Team B5
April 6, 2022

Nectarfy

Technical Report

Table of Contents

Product Brief

Nectarfy's flagship product will be a revolutionary smart hive that will help bees and beekeepers thrive. By including four key components, Nectarfy will be able to provide a holistic solution for the largest problems faced by the beekeeping community.

<table>
<tr><th colspan="2">Varroa Mite Detection</th><th colspan="2">Climate Control</th></tr>
<tr><td colspan="2">Why?</td><td colspan="2">Why?</td></tr>
<tr><td colspan="2">- Varroa mites weaken bees, leaving them more susceptible to other risks
- Current solutions do not alert beekeeper of varroa mite levels</td><td colspan="2">- Bees are sensitive to temperature
- No current solution to protect hives against extreme weather events leading to significant losses</td></tr>
<tr><td colspan="2">Key Features & Benefits</td><td colspan="2">Key Features & Benefits</td></tr>
<tr><td>Must Haves:
- Detection of varroa mites at various levels
- Alerts beekeeper to treat varroa mites</td><td>Better Ifs:
- Application of treatment remotely or automatically</td><td>Must Haves:
- Monitor temperature and humidity
- Maintain livable hive environment during extreme weather for several hours</td><td>Better Ifs:
- Always control hive environment
- Allow user to set hive environment factors</td></tr>
<tr><td colspan="2">Technology</td><td colspan="2">Technology</td></tr>
<tr><td colspan="2">- Gas sensing technology proved to work by Polish research team</td><td colspan="2">- Temperature and humidity sensors
- Heating and cooling elements in hive walls</td></tr>
<tr><th colspan="2">Improved Feeding</th><th colspan="2">App Integration</th></tr>
<tr><td colspan="2">Why?</td><td colspan="2">Why?</td></tr>
<tr><td colspan="2">- Climate change limits bees' ability to find food
- No feeders on the market that mix syrup to promote activity or alert when empty</td><td colspan="2">- Hard to diagnose the health of the hive
- Difficult to find beekeeping community
- Events that affect hive sometimes missed</td></tr>
<tr><td colspan="2">Key Features & Benefits</td><td colspan="2">Key Features & Benefits</td></tr>
<tr><td>Must Haves:
- Access for bees
- User set ratios of sugar to water mixed
- Alert beekeeper when food low</td><td>Better Ifs:
- Ensure at least 10 lbs of honey is always in the hive
- Nutrients included in mixture</td><td>Must Haves:
- Alert beekeepers of important events
- Community page
- Easy to understand data from hive</td><td>Better Ifs:
- Marketplace for equipment or goods
- Education regarding bees and beekeeping</td></tr>
<tr><td colspan="2">Technology</td><td colspan="2">Technology</td></tr>
<tr><td colspan="2">- Sensors to detect when tank empty
- Mixing ingredients using an air pump</td><td colspan="2">- Communication between app and hive
- Data analysis</td></tr>
</table>

Product Illustration

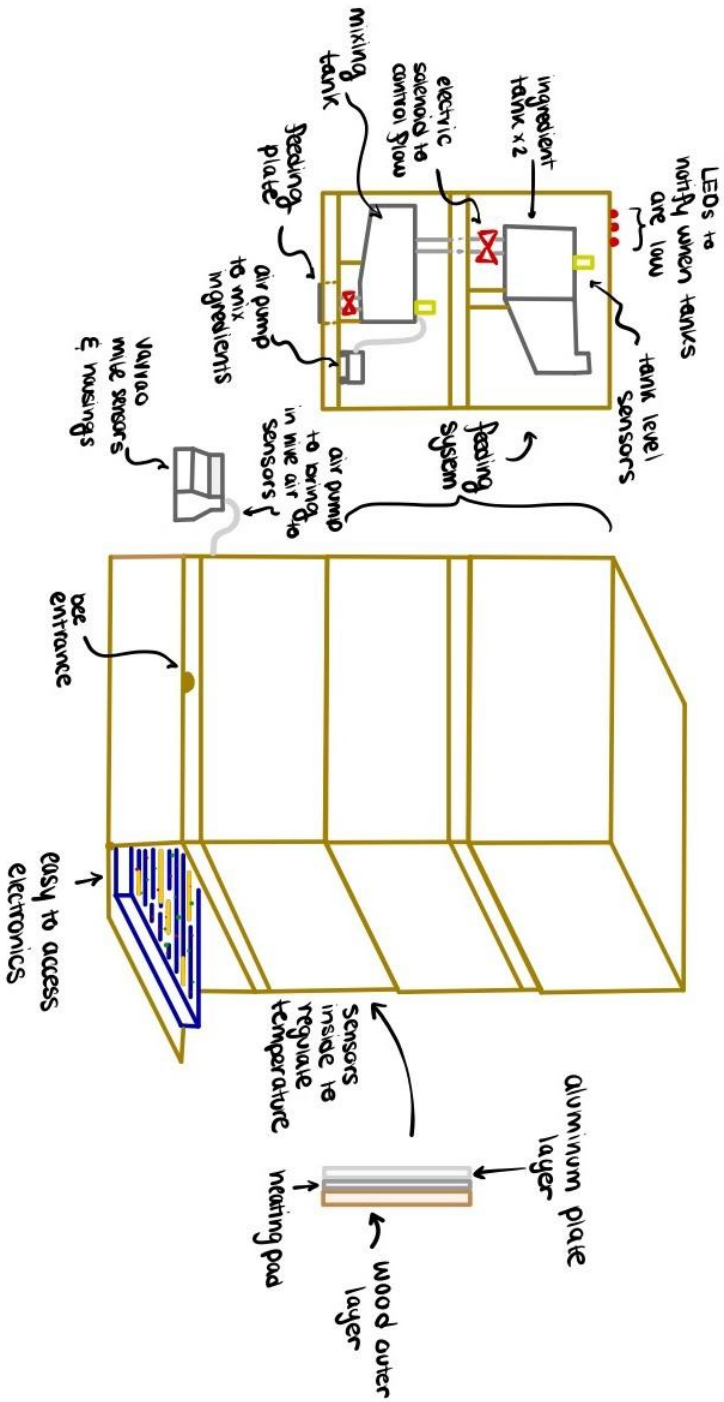


Figure 1:Product Illustration

Requirements

Motivation

The motivation behind Nectarfy and its features are based on the vision to create a product which solves various problems faced by our key stakeholders: urban and professional beekeepers. These people are our primary stakeholders, and whom we have based our solutions and product design around. We are aiming to appease two sets of stakeholders through our product: the urban beekeepers, who we aim to sell our product to, and professional beekeepers, who are experts in the field and will thus have a deep understanding of the problems and how our solutions will resolve those problems. Through our requirements, our product blends the lines between these two stakeholders to create a holistic solution.

Product Vision

50 billion bees were killed in the winter months of 2019, seven times the world population at the time. Most of these deaths are due to various factors, but based on our customer validation it comes down to three factors: internal beehive climate, parasitic varroa mites, and feeding difficulties. To resolve each of these issues, the technical team developed the following solutions:

- **Climate Control:** Thermocouples and humidity sensors monitor the internal temperature while two heating plates change the internal temperature
- **Varroa Mite Detection:** A gas pump and probe samples air within the beehive, which are analyzed by an array of gas sensors and a machine learning model to identify infection by varroa mite.
- **Improved Feeding System:** Two storage tanks controlled by solenoids contain feeding solutions which can be dispensed at custom ratios into a mixing tank, where a high-power pneumatic agitator mixes the solution to be dispensed to the bees.

All these systems are integrated together via a smartphone application, where the user of the hive can monitor all of its status and take corresponding action when required.
Given the distinct capabilities of each subsystem within the beehive, we have split up requirements into each subsystem, as well as a few general requirements for the overall beehive. These requirements were discovered after thorough primary and secondary research, consulting beekeepers and other industry experts

Requirements

Given the drastic amount of requirements needed for every section of our technology, we have provided a reduced list in the report with a more exhaustive list in the appendix.

Beehive Requirements

Requirement	*Evaluation Criteria*	*Justification*
Maintain a variable living space for bees	Check to see if bees live within beehives for prolonged periods of time.	If bees cannot live within the beehive, it will render the product useless
Cohesively integrate all subsystems, including but not limited to, feeding system, climate control system, varroa mite detection system, any other electronic or mechanical system which contributes to functionality of any or none of the previously listed subsystems.	Test the entire beehive system for functionality, ask users to use the beehive and rate clunkiness.	Without proper integration of all the desired subsystems, the product would not function to the degree at which the customer expects.

Future Requirements	*Justification*
Allow for stacking of multiple instrumented boxes on one base	Beekeepers often change the size of their beehives with regards to how many colonies they wish to maintain. If the beehive is unable to change its size at will for the beekeeper, they are less likely to use them.

Improved Feeding System Requirements

Requirement	*Evaluation Criteria*	*Justification*
Notify the beekeeper when ingredient tanks or mixing tanks are empty.	When the tank is empty, the user is notified, preferably through an app notification, but at the bare minimum by some visual indication on the hive.	Beekeepers must know in a timely manner when either the ingredients or mixing tanks are running low to either provide more ingredients for the system to make simple syrup with or to create a new batch of simple syrup to ensure the bees always have access to food.
Mix user set ratios of ingredients.	System must be able to mix colored water to produce different color mixtures that correspond to user set ratios.	Different ratios of water and sugar promote different behaviors in the bees, it is important for the beekeeper to have control of the ratios so they are able to promote the behaviors they desire through-out the year.
Tanks and other ingredient	Water must run through the	Any component that comes into

components must be completely leak-proof.	entire system with no to minimal leaks for the first prototype iteration.	contact with the ingredients must be completely leak-proof to ensure no damage to electronics or the hive itself. Moisture in particular can significantly affect overall hive health so it's important for no water to leak.
The holes the bees feed from must be able to hold surface tension.	When the system is sealed, no fluid should leak from the feeding area.	The bees must be able to access their food without risk of liquid spilling into the hive.

Future Requirements	*Justification*
System must be of an ergonomic size and weight.	It is important for the system to be easy to remove from the hive so that the beekeepers can access their bees with ease. The MVP is focused on proving functionality, therefore, size can be overlooked at the moment but must be considered in the future before releasing as a product.
System must include three tanks.	Many beekeepers like to add additives to their bee's food so there must be an additional tank to store these additives so they can be added to the mixture when desired.
System must be able to send notifications and allow for ratios to be set remotely.	Currently, the system requires the beekeeper to be near the unit and have a computer plugged into it to see tank empty lights, set ratios of food, and dispense food. To become a true solution, all of these tasks must be done remotely.

Climate Control System Requirements

Requirement	*Evaluation Criteria*	*Justification*
Detect temperature accurately (± 2 C)	The measured temperature must be close to the actual temperature of the environment, so a small margin of error is allowed.	The climate control system needs to know the temperature in order to turn on the heater when the temperature is too cold and turn the heater off when the temperature has reached the ideal temperature.
Heating plate can reach 40C	When plugged into power (120VAC), the heater can reach 40C	Bees thrive between 30-35 C. This temperature is optimal to raise brood and promotes colony fitness.
Can control temperature inside the hive	When the system has power, it should be able to reach the ideal	The ability to heat up the hive from the inside would save

	temperature inside the hive and maintain the temperature	beekeepers a lot of time, and also ensures the bees would not die of the cold snaps in the winter.
Report to user with the basic condition inside the hive	The bluetooth module can send data via bluetooth low energy to bluetooth devices	The user needs to know the overall health of the hive without opening the hive and checking it themselves. Beekeepers validated that opening a hive and checking on the bees conditions is disturbing for bees.

Future Requirements	*Justification*
User's could control the temperatures in the hive themselves.	Beekeepers could have the ability to change the temperature inside their hive depending on what they want and the weather condition. Maybe beekeepers in the winter decide they do not want to have their hives at 30 C all the time in order to save electricity, or just to turn off heating during the spring and summer.

Varroa Mite Detection System Design Requirements

Design Requirement	*Evaluation Criteria*	*Justification*
Distinguish between different gaseous compositions.	Reading resistance values from the sensors should change and reach a steady state when obvious gasses like ethanol are present.	Without the ability to detect changes in the atmospheric gas composition, the system does not work.
Machine learning model can learn and predict different gaseous compositions	The machine learning model will be trained on known but imprecise gaseous composition changes, such as human breath and air within a room. The machine learning model should have at least 90% accuracy when making predictions.	Without the ability to learn and identify different gaseous compositions, the system does not work.
Can self-cleanse detection chamber	After the successful prediction and detection of a non-air gas, running the system and prediction again should result in an "air" or base-gas prediction.	If the chamber in which the gas being sampled is not cleansed, it would void the accuracy of the results, rendering the product useless.

Future Requirements	*Justification*
Distinguish between gaseous compositions of infected and non-infected hives	Because the purpose of this system is to detect varroa mite, it needs to be able to achieve this requirement. This requirement was waived in favor of the requirement to distinguish between different gaseous compositions due to the timeframe and seasonality of the industry, which would not allow us to build an MVP in time for major deadlines.
Able to act upon the detection system to trigger another system which can treat beehives for varroa mite.	This is the next step in the varroa detection system, based on the interviews and validation conducted. The beekeepers cared most about detecting AND treating varroa mite. However, we found that treating Because of the limitations of our skills and time, we were not able to integrate this technology into our current MVP, but will be a requirement for the future MVP.

Mobile Application Requirements

Requirement	*Evaluation Criteria*	*Justification*
Connection from user's smart device to the Nectarfy Hive	The mobile app could find, connect, and receive data from the hive via bluetooth low energy	The users need to be able to see what is going on inside the hive without going out and opening the hives.
Authentication: users could sign up, sign in, and log out of the app.	Frontend could send requests creating users and logging in by comparing passwords in the database while ensuring the password is encrypted	Authentication needs to exist so that users could log into their Nectarfy account from any device.
Users could interact with each other in the community of beekeepers: users could make a post, like a post and comment on a post.	Users can interact with the frontend, and actions could be converted into RESTful APIs, making requests to the database	Users interacting with each other is essential to building beekeepers' community, providing people with common interests a place to share their knowledge and help others.

Future Requirements	*Justification*
Recommended actions based on the current condition of the hive and the current weather condition in the area.	Providing the users with what to do next is important in educating new beekeepers. This feature will be added once the team gathers enough expertise for each scenario.
Ability to control climate control, feeding system, and test varroa levels from their phones.	This would make beekeeping simple and allow it to fit into people's busy schedules, only needing to take care of the bees when it is absolutely needed.

Design

Product Architecture

The Nectarfy hive has of three key systems to provide a holistic solution to urban beekeepers.

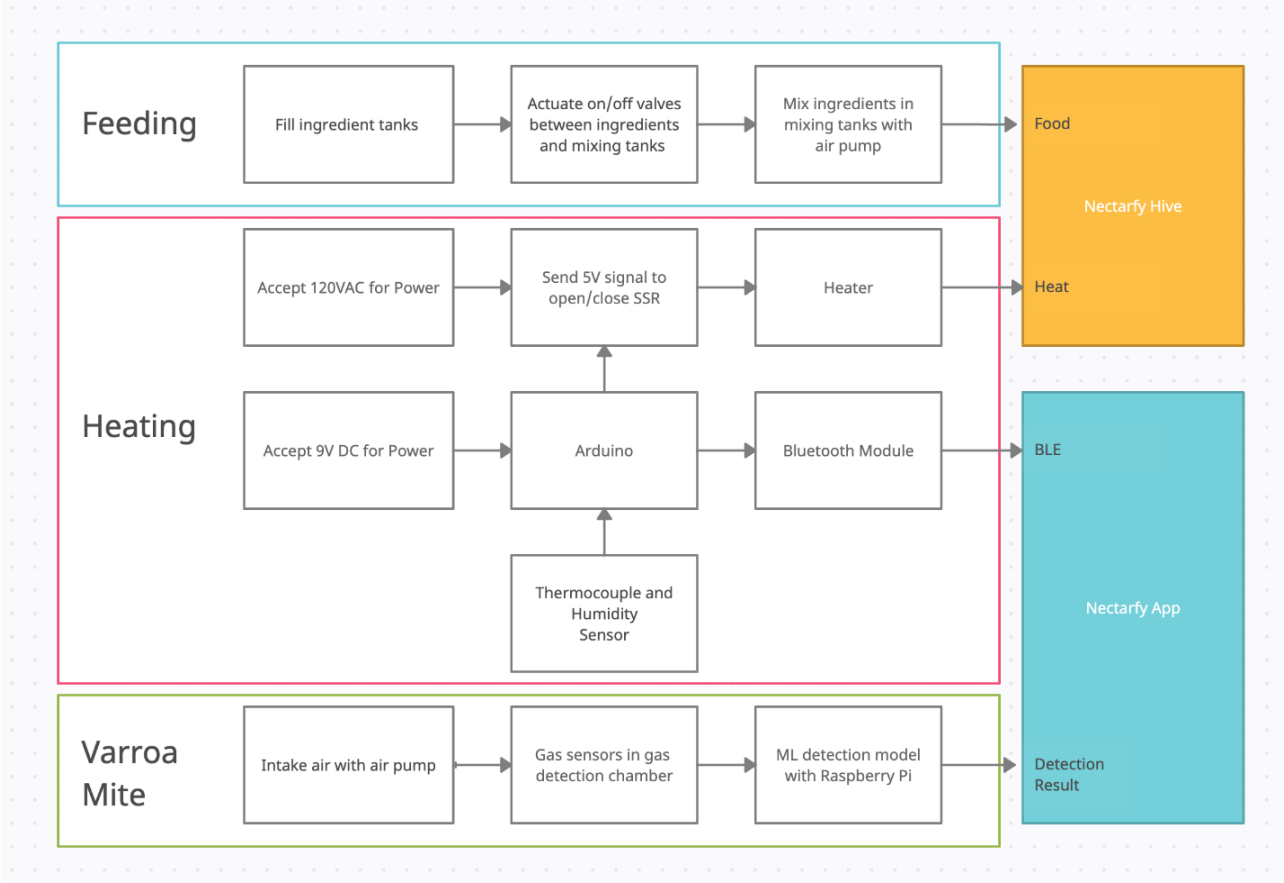


Figure 2: Function Structure Diagram

Component Descriptions

Improved Feeding System Design

One of the key aspects of the smart hive design included the improved feeding system. This system is responsible for automatically mixing different ratios of water and sugar that is then fed to the bees.

- Tank level sensors that indicate when a tank is empty with a red LEDs
- Leak-proof ingredient system including tanks, connections, and solenoids
- Feeding plate design that allows for fluid to be held in feeding volume through surface tension
- Controllable fluid flow between mixing tank and feeding area to ensure no leakage while mixing
- Mixing of ingredients done through an air pump which creates turbulence in system

- Ingredient ratios may be set by controlling the amount of time the solenoid between the ingredient tank and mixing tank is open for

This functionality was achieved through many rounds of design, iteration, and testing.

Overall System Design

The current design of the improved feeding system is as follows:

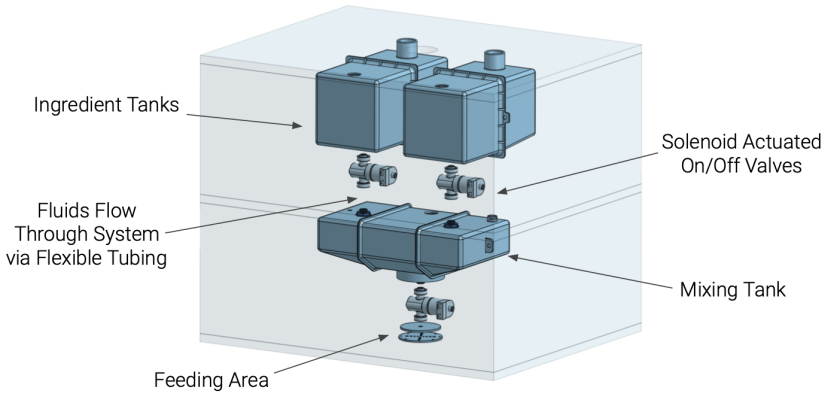


Figure 3: Overall System Design

Mounts and fittings are omitted in the CAD, but can be seen in the MVP.

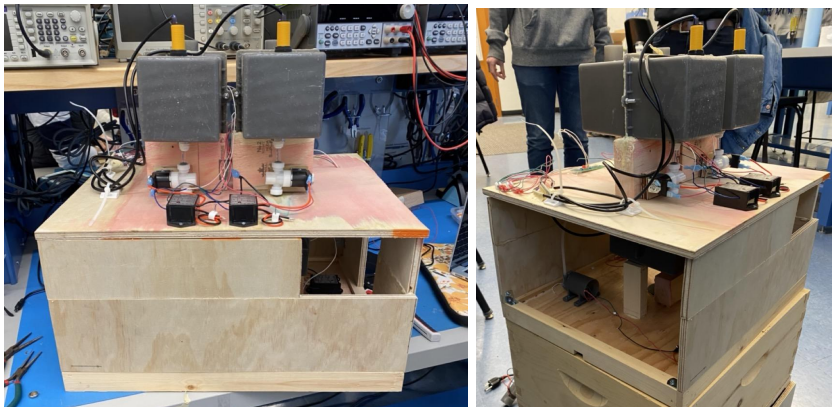


FIgure 4: Photos of Feeding MVP

There are two ingredient tanks which can be actuated separately. The fluid fills into a mixing tank where a pneumatic agitator mixes the fluids. A solenoid connected to the mixing dispenses the fluid into the feeding area for bees. IR sensors will report if the tank is empty via LEDs.

Ingredient Tank Design

While a simple premise, there were several key design choices made when creating the ingredient tank:

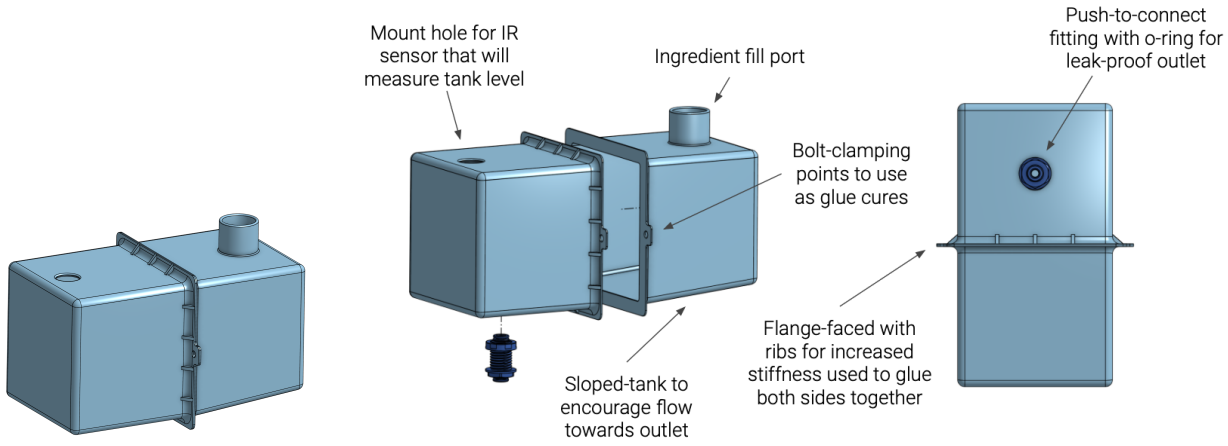


Figure 5: Complete Ingredient Tank CAD (Left), Breakout View of Ingredient Tank (Right)

The tanks are made of multiple parts which are sealed together. All other joints use O-rings and other sealants to prevent leaking. Ribs maintain rigidity when the tanks are full.

Mixing Tank Design

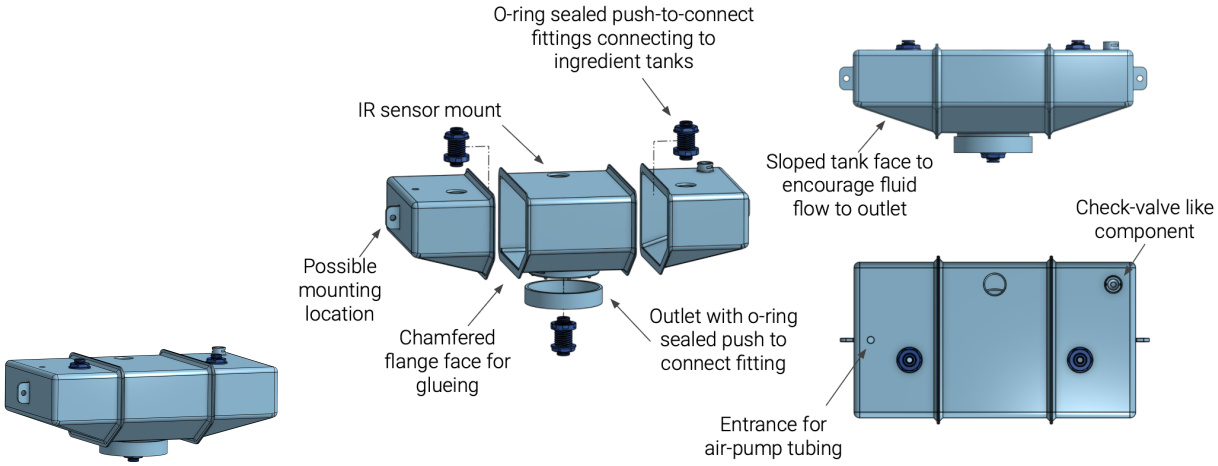


Figure 6: Complete Mixing Tank CAD (Left), Breakout View of Mixing Tank (Right)

The mixing tank is made of three parts joined with sealant. Each aperture for water is sealed with O rings and sealants. The tank has a sloped base to encourage fluid flow and ribs for structural rigidity. A check-valve and pump tubing entrance are made to ensure pressure for fluid flow and mixing agitation, respectively. When there is no air, surface tension in the feeding area prevents leakage.

Feeding Area Plates

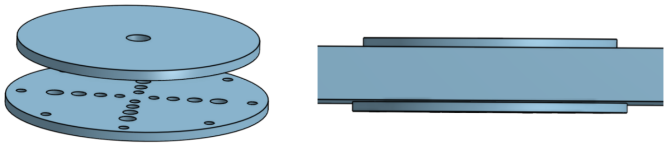


Figure 7: Breakout Feeding Area Plates (Left), Assembled Feeding Area plate(Right)

The feeding area consists of two plates encasing a through-hole on the bottom board of the system. The design for both plates is fairly simple:

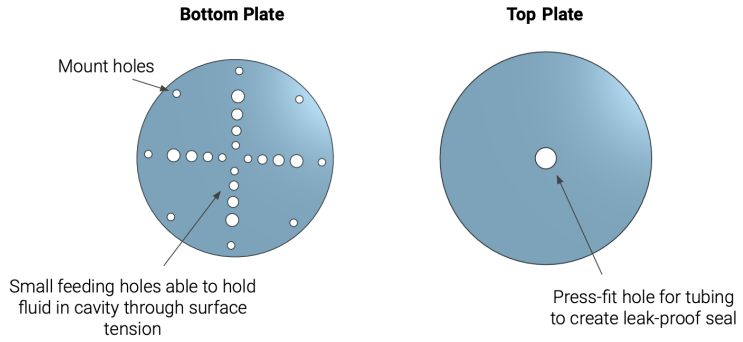


Figure 8: Bottom Plate Overwi (Left), Assembled Feeding Area plate(Right)

The top plate press fits to the solenoid, creating a seal. It is completely sealed to prevent any pressure causing leakage. The diameters were specifically designed such that surface tension would retain and allow maximum bee feeding access. The bottom plate was also glued to the bottom board to ensure a tight seal.

Solenoids

The solenoids used in the system were chosen based on their ability to actuate electronically, push-to-connect compatibility, normally-closed valve position, power requirements, food-safe status, and cost. Tubing was dependent on the solenoids and chosen as such. Relays were used to actuate the solenoids and ensure operation with an Arduino. All part numbers can be found in the appendix.

Figure 9: MVP Solenoid Implementation

The circuit diagram and the Arduino code used to run the solenoids can be found in Appendix C.

Tank Level Sensors

In the system, IR distance sensors were installed on every tank so that it could be used to monitor and alert the user when the tanks are empty. Below is an actual image of them in the system:

Figure 10: MVP IR Sensors

For this current version of the prototype, each sensor simply connects to an LED that turns on when the tank is empty:

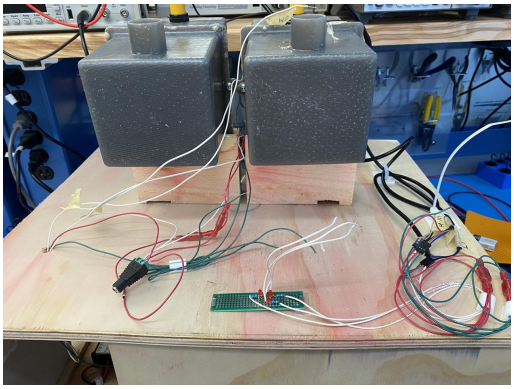


Figure 11: MVP LED Indication Boards

The circuit diagram for this system can be found in Appendix C.

Air Pump Design

A custom designed DC motor with an enclosure to push high volumes of air was designed to mix the syrup, described in the figure below.

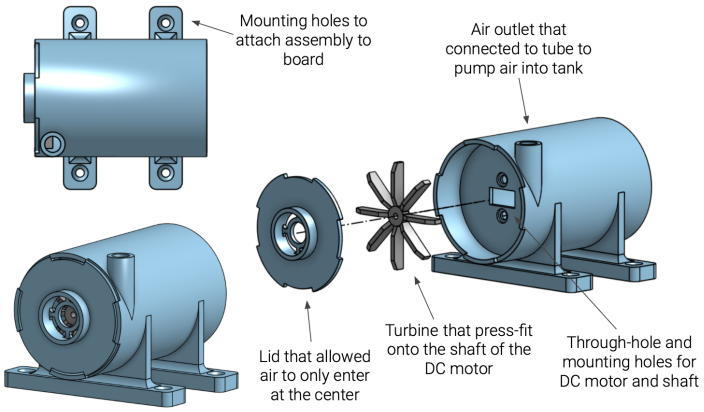


Figure 12: Pneumatic Agitator Motor Design and Breakout Diagram

On the current MVP, the DC motor rotor was damaged due to excessive vibration in transport, causing it to draw magnitudes more than its rated current. The circuit was modified to account for this issue to be a simple push button, for the current MVP only, the intended driving circuit is the same one used for varroa mite found in Appendix C.

Future Design Considerations

The future design implementations for the improved feeding system include significantly reducing the size and weight of the unit. This can be done by reducing tank size and potentially using pumps instead of gravity so all the tanks are on one level. A third tank must also be introduced for nutrients but this is a simple task as it can be the same as the other ingredient tanks. The electronic systems must also be improved so that all data is read to the microcontroller and shared with the app. Lastly, design for manufacturing practice can be applied.

Climate Control Design

Sensor Selection and Set Up

For moisture detection, DHT11 humidity sensor was used since it was popular for detecting the air moisture and the most readily available to use for Arduino.
A K-type thermocouple with a MAX31855 Thermocouple to Digital Converter was chosen for sensing the temperature.The sensitivity of the thermocouple was tested in various conditions, and was found to be promising. The error was found to be an average of 2%, see appendix for details. The setup is below:

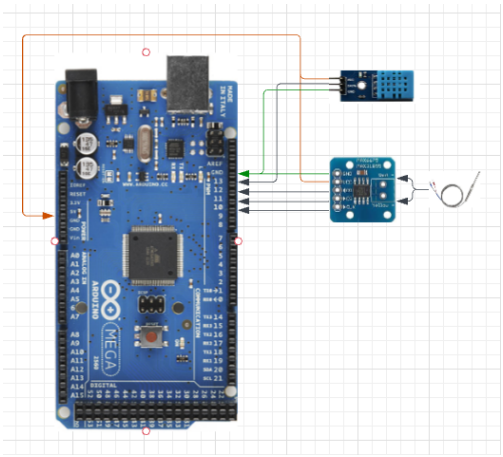


Figure 13: Climate Control Sensor Circuit Diagram

Heating Element and Final Hive Design

The design for the heaters was iterated upon based on key requirements. The final design had the heaters, which are thin and able to reach 40C, directly integrated with the walls. The heater has an adhesive for easy manufacturing and can be powered with 120VAC and a relay. The electronics will be housed in a drawer like compartment below the beehive while the sensors remain in the beehive. This design minimizes the risks of bees going near the electronics and uses the space inside the hive optimally. It uses three thermocouples for robustness and redundancy, taking the average values of the sensors, a humidity sensor, a bluetooth module, and an Arduino for control.

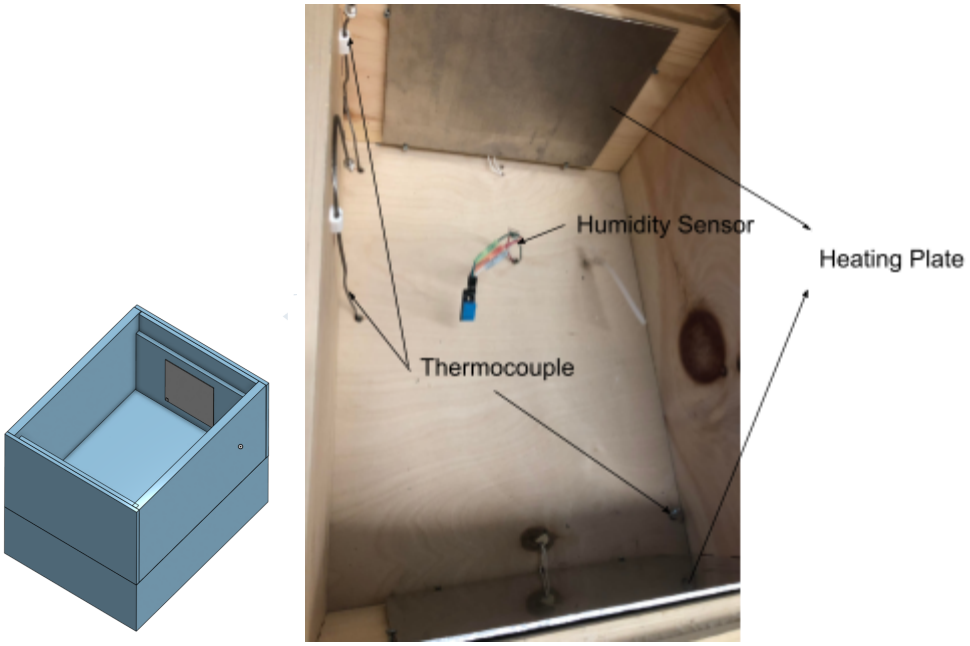


Figure 14: CAD of Heater System(Left), MVP Implementation(Right)

Controlling Heating Element

The heating element is controlled with the arduino, since the arduino will gather information regarding the temperature inside the hive, it will turn on the heater when the temperature is not in the ideal range and turn off the heater once ideal temperature is reached. This could be controlled via a (SSR)Solid State Relay using the arduino to send 5V signals to the SSR, the heaters could be turned on, and when the arduino does not send 5V signals, the heaters will be off.

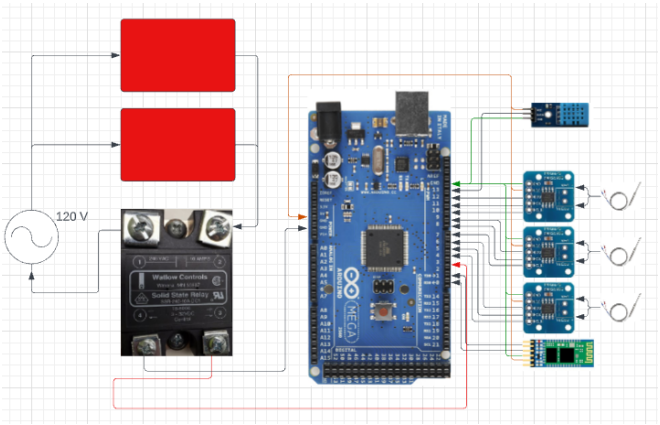


Figure 15: Climate Control Heating Circuit Diagram

Arduino Code for Sensors, Heaters, and Bluetooth

The Arduino Mega 2560 was used in climate control, its purpose is to gather sensor data from the humidity sensor and the thermocouples, communicate this information to the user via bluetooth connection, as well as controlling the status of the heater based on the current temperature of the hive. The code used can be found in the respective appendix.

Meeting Requirements

In terms of the requirements of climate control, all requirements have been met with the exception of the humidity requirement. It was due to a lack of resources to test as the team was running low on budget and time to perform legitimate tests.
The requirements have been met:

- Detect temperature accurately (± 2 C)
    - Through testing the thermocouple thoroughly, and verifying that it could sense temperature consistently.
- Heating plate could reach 40 C
    - By connecting the adhesive heaters to 120VAC, it could reach 40 C and above.
- Can control temperature inside the hive
    - By using a SSR with an arduino that will take the temperature of the beehive constantly, the heaters will be turned on when the temperature from the thermocouple indicates that it is not at the ideal temperature and will turn off when it reaches the ideal temperature.
- Report to user with the basic condition inside the hive
    - The arduino is equipped with a Bluetooth module and it will constantly send the humidity and temperature information to the users via Bluetooth Low Energy.

Future Design Considerations

Moving forward, it would be a great value add to have a stackable design for the beehive, because it is a common practice in the beekeeping industry. The stackable design would allow for heaters on the top hive to connect via a parallel connection to the power, reducing the amount of outlets needed for power.
The climate control's efficiency would greatly improve if insulation could be added to the walls of the hive. For instance, insulation could be an extra layer of foil and wood that could attach to the beehive via magnets, and beekeepers could take it off easily from the beehive.
Additionally, users able to control the temperature of the hive would be a great integration for the software. Having the ability to change the temperature inside their hive depending on what they want and the weather condition could be a value add for beekeepers.

Varroa Mite Detection Design

Sensor Selection and Instrumentation

Inspired by existing research[B] the TGS gas sensor series by Figaro Sensors[C] were used. These gas sensors are highly sensitive, robust, and have varying degrees of detection ranges, including but not limited to alcohols, butane, VOCs, refrigerants, and COs.
The gas sensors used in our system from Figaro Sensors were:

- TGS832-A00: Tin dioxide semiconductor, high sensitivity to Chlorofluorocarbons.
- TGS2600-B00: Metal-oxide semiconductor, high sensitivity to air contaminants.
- TGS2602-B00: Metal-oxide semiconductor, high sensitivity to ammonia, H2S, VOCs.
- TGS2603-B00: Metal-oxide semiconductor, high sensitivity to amine-series and sulfurous odors.

- TGS2620-C00: Metal-oxide semiconductor, high sensitivity to vapors of organic solvents and other volatile vapors.

Each sensor varies its resistivity value when detecting different types of gas. Based on data sheets supplied by Figaro, the sensors were built into the following circuit:

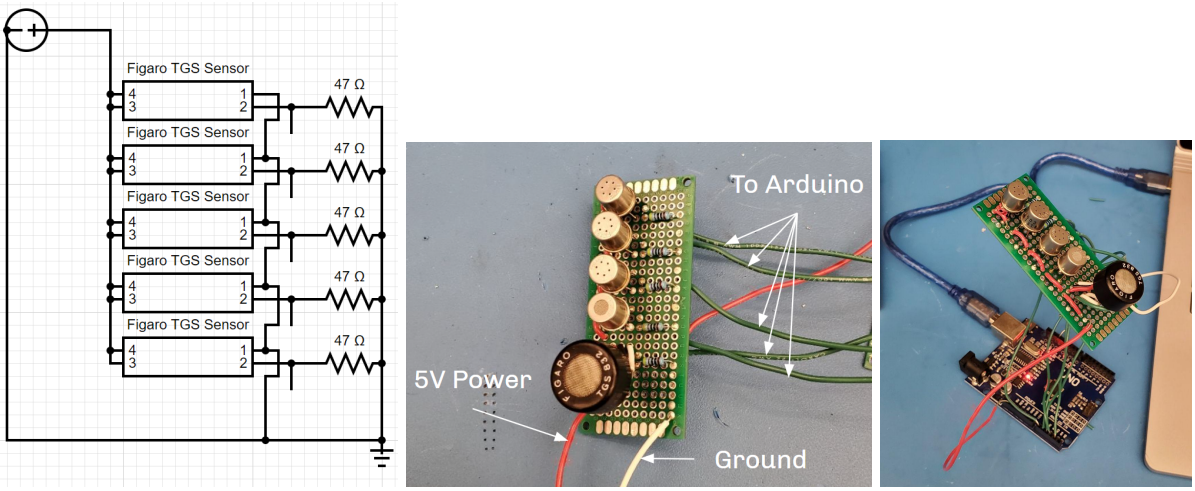


Figure 16: Varroa Mite Circuit Diagram (Left), MVP Diagram (Center), Arduino Attachment (Right)

Where each disconnected pin before the resistor is a wire going to the Arduino.
The resistors used were 47 Ohms, as per the datasheet, and each output pin was read by the Arduino by using the Analog In pins.

Motor Selection and Instrumentation

A DC motor and tubing is used to pull gas from the beehive into a semi-isolated chamber where the gas sensors are able to process and read the data.
To control and ensure safe operation of the motor, a MOSFET and driving circuit was used.

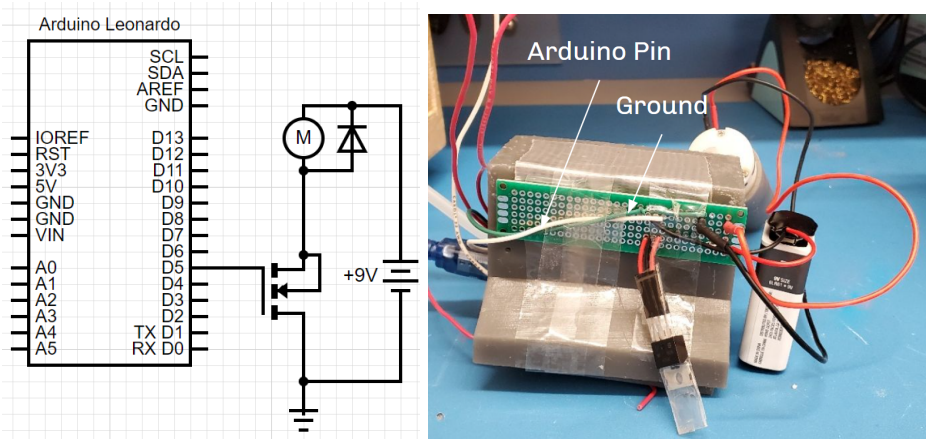


Figure 17: Motor Driver Circuit Diagram (Left), MVP Housing (Right)

A 9V battery was used to be the power supply for the motor, and the Arduino is plugged in to control the MOSFET, therefore being able to turn on and off the motor as required.

Sensor Housing CAD Design

The sensor housing was designed to house the sensors and Arduino in an semi-isolated space to provide greater robustness to the sensor reading data. The package was aimed to be created such that all parts could be easily assembled and contained for ease of use and maintenance.

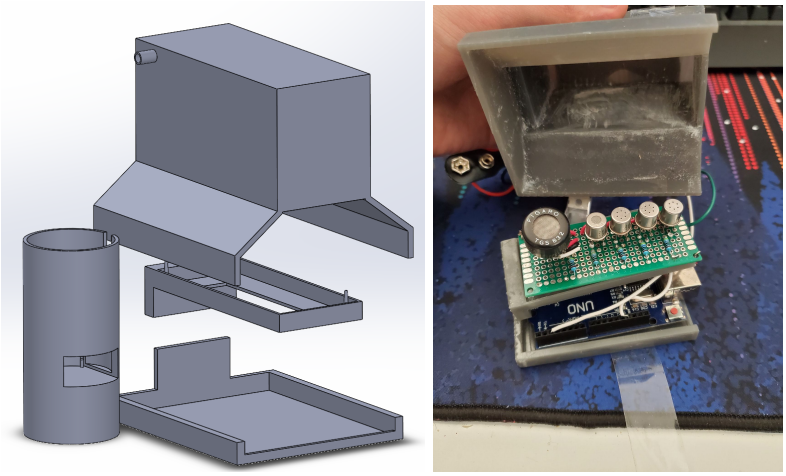


Figure 18: Varroa Mite Housing CAD (Left), MV Implementation (Right)

Mechanical Beehive Integration

As the beehive has an easy-access electronics shelf, the original intent of the design was to hold the entire sensor system and the computer. However, due to time and budget constraints, the sensory system was unable to fit within the beehive, but acts as a discrete part of the beehive. The Raspberry Pi computer was successfully able to fit within the beehive, as well as the indicator LEDs which show the status of the beehive to the user. These indicator LEDs were simply two LEDs attached to long wires which are connected to Digital Output Pins to the Arduino.

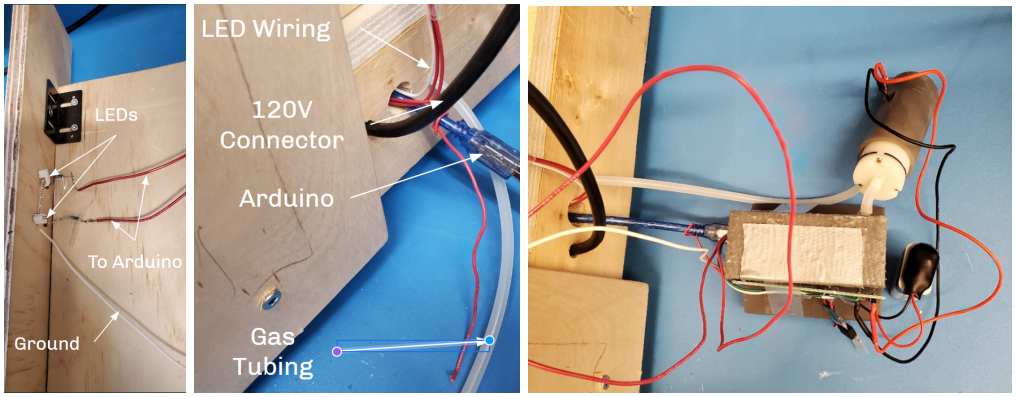


Figure 19: LED Wiring Diagram (Left), Rear Wiring(Center), Varroa Detection Setup (Right)

The tubing and wiring both within the beehive to the sensor array, to the Raspberry Pi, to the indicator LEDs were done through drilling holes and cable management.
Careful consideration was made to ensure there was no interference with the other electrical systems within the “easy-access” shelf.

Software

The Arduino will power the sensors as well as read data from such sensors, control the motor, and indicator LEDs.
For the computer, Python scripts were written. The full code and explanations is in appendix B. There are four different scripts. They are listed below:

1. Gas Sensor Library
    a. A library holding functions which operate various aspects of the gas system.
2. ML (Machine Learning) Model
    a. A machine learning script which builds a ML model based on the K-Nearest Neighbors algorithm.
3. Gas Sensor Interface
    a. A wired computer interface for controlling the gas sensor array for data collection, debugging, and improving robustness.
4. Gas Sensor Interface BT (Bluetooth)
    a. A script which piggybacks off of scripts 1 and 3, for headless user operation of the gas system through a bluetooth connection and Android app.

Meeting Requirements

Given the overview of how the varroa mite detection system is built, we can clearly see each requirement met:

- Distinguish between different gaseous compositions.
    - The gas sensor array, code, and machine learning model prove that the system is able to detect different gas compositions. The resistivity values which change based on composition proves there is a change detection, and the machine learning model makes it further interpretable for users.
- Able to report to the user when a recognized change is detected.
    - The Machine learning model and the LEDs built into the beehive evidently do this, the machine learning model able to show what it thinks the beehive is via text, and the LEDs letting the user know if they are not able to access a computer.
- Can automatically sample air data
    - The GasSensorInterface and GasSensorInterfaceBT show that the system can act automatically as soon as the user prompts the system, either via Bluetooth or through a computer. One that button is pressed, the system requires no more human interaction
- Can self-cleanse detection chamber
    - The DC motor which runs to collect the gas can be run again to cleanse out the chamber in which the gas sensor reading is made. This requirement was seen to be met through the repeated collection of data, which would not have been possible if the gas motor was unable to clean out the system, introducing inaccuracies and noise.

Future Design Considerations

The biggest problems with the system were the motor controller circuit not being very reliable, and the housing of the sensors and controller not actually being able to fit inside the beehive itself. These two issues could be resolved with greater care and consideration in their design, perhaps using a PCB with a motor controller chip instead of a discrete part circuit.
Moreover, the housing could be modeled in such a way where the Arduino is housed in a discrete, separate housing, which would greatly reduce the clearance within the system. Furthermore, the system could be further abstracted through the use of the ATMEGA microcontroller itself and custom PCBs, which would not only streamline the design and manufacturing process..
Another design consideration would be to have a self-learning ML model, one which will be able to update itself on collected data as opposed to relying on a single static set of previous data collected. Finally, the bluetooth integration could be designed to be more elegant and incorporated into the software application already designed, and thus become device agnostic.

Mobile App Design

The Nectarfy app is developed with Flutter Framework for software agnosticism. MongoDB was used for infor storage, and a Node Server with Express was used for backend to connect frontend to the database. Detailed implementation can be found in appendix C3.

Overall Architecture

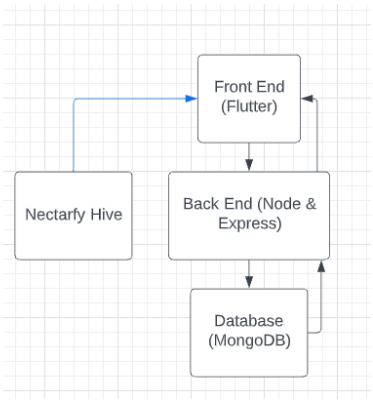


Figure 20: Mobile App Flowchart

The frontend of the app interacts with the backend of the app via API requests, POST requests, and using JSON objects. More details are available in the appendix.
The app currently can connect to devices via Bluetooth Low energy which is handled in the background and relevant information will be displayed to the user refreshing every 10 seconds.

Meeting Requirements

- Connection from user's smart device to the Nectarfy Hive
    - The app will automatically search nearby bluetooth devices and would automatically connect the the device with our name, making connecting to Nectarfy hives as easy as possible

- Authentication: users could sign up, sign in, and log out of the app
    - The app connects to our database via our backend, which is built with Node js and Express as the server. The app also has an internal state management for the user, so that different things would display depending on if the user is signed in or not.
- Users could interact with each other in the community of beekeepers
    - Each registered user has the ability to post, like and comment in our community module, and all requests are being sent to and processed by our backend server that would update the database in real time.

Future Design Considerations

The app in the future will have additional features such as deploying actions from the phone, and setting ideal temperature for the hive in the mobile app. The app will also provide the users with what to do next depending on the current condition of the hive. This is important in educating new beekeepers. This feature will be added once the team gathers enough expertise for each scenario.

Verification and Validation

Introduction

One important distinction to make about validation for our system is that our validation will not be able to include true in-field validation. This is due to the seasonality of the industry we are attempting to enter. As bees are inactive during the winter months and only start to wake up late March, which is when the MVP and demo must be completed, and thus near the end of the course. Because most of these features are meant to be functional during the winter months, and the varroa mite system would require time to collect and train on healthy and infected hives, we aim to have these verification tests done in the future when we are able to line up with the seasonality of the industry. Likewise, future validation for these systems are discussed, including working with beekeepers and apiaries.

Improved Feeding System

Requirement Verification

Significant Requirement	*Verification*
Notify the beekeeper when ingredient tanks or mixing tanks are empty.	**Interview Support:** Henninger Hill Apiary, Purrfect Honey Current feeding solutions do not notify the beekeeper when empty so would be a competitive advantage.
Mix user set ratios of ingredients.	**Interview Support:** Henninger Hill Apiary, Purrfect Honey **Literature Support:** Honey Bee Hobbyist

	Automatic mixing would provide a competitive advantage since current solutions require beekeepers to make simple syrup at home.
Tanks and other ingredient components must be completely leak-proof.	As electronics will be used, this is a constraint put on the assembly since the simply syrup consumed by bees has a large percentage of water and could cause significant damage to the electronics and hive if it leaked. Interview Support Regarding Humidity Concerns: Jarret (BC Hobbyist), Scott (BC Hobbyist), Henninger Hill Apiary, BCB Honey Farm, East Van Bees
The holes the bees feed from must be able to hold surface tension.	This is a constraint as if all the liquid were to fall out of the system it would hurt the hive and provide no way for the bees to access food.

Design Verification

Requirement	*Design*	*Test(s)*
Notify the beekeeper when ingredient tanks or mixing tanks are empty.	Integrated IR sensors into all tanks that output high when the tank is empty and low when the tank is full. An LED is connected in series with the output signal so it lights up to alert the user when the tank is empty.	1. **Initial Test:** Began with an empty tank and turned on power to the system. The LED was initially on. As water was added to the tank, the LED turned off. 2. **Continuous Test:** Through-out prototyping, LED status lights were monitored to ensure they only turned on when the corresponding tank was actually empty.
Mix user set ratios of ingredients.	Use of electronic on/off valves controlled via solenoids allow for precise control over how long the valve is open for and ingredients are being dispensed. By setting different times for each valve to be open, one can set the ratio of ingredients to be mixed.	To test the system, each tank was filled with different colored water, one blue and one red. Then the below tests were carried out: 1. **Single Ingredient Dispensed:** Turned on the solenoid valve corresponding to a specific ingredient tank to allow fluid to flow to mixing tank. Turned on solenoid between mixing and feeding area to dispense liquid and visually confirm if it was the correct color. 2. **Mixed Ingredients Disepensed:** Turned on each solenoid valve for the same amount of time so the water in both tanks flowed to the mixing tank. Ran the pump for a few seconds. Turned on solenoid between mixing and feeding area to dispense liquid and visually confirm it was purple.
Tanks and other ingredient components must be completely leak-proof.	All components were designed to minimize leaking by using sealant and integrating o-rings where necessary.	1. **Component Testing:** Once fully assembled, every component was checked to see if it was water tight by filling the component fully and observing it for 10 minutes to see if any leaks formed. If there were, additional sealant was applied, cured, and the component was tested until it no longer leaked.

		2. **System Testing:** After individuaully testing all the components, the entire system was assembled to ensure no leaks as the liquid flowed through-out the system. Colored water was used to help identify source of of leaks and sealant and other measures were used to eliminate leaks as they appeared through many rounds of testing until the system was able to produce three batches of mixed water without leaks.
The holes the bees feed from must be able to hold surface tension.	The feeding plate has several holes cut into it so the bees can access the simple syrup. By eliminating all areas where air can enter the system, these holes are able to hold the fluid in the feeding area through surface tension.	1. **Hole Sizing Test:** A simple test using a coffee cup was done to see how well surface tension could hold liquid in the container. A hole was made at the bottom of the coffee cup. The cup was then filled with water and flipped so that the water dispensed out of the drinking hole in the lid. When the hole on the bottom was covered, no water would spill out. When the hole was not covered, water spilled out very quickly. Therefore, holes in the feeding plate had diameters smaller than that of the drinking hole. 2. **Plate Check:** First, the check-valve hole is blocked off. Then, the valve between the mixing tank and feeding area is opened and water flows into the feeding area. After, the valve is closed. The entire time, the plate plate is observed to for any leaks.

Design Validation

The next steps that need to be taken in the prototype design involve validating the design with actual beekeepers to gather feedback, observe what features they find useful and do not find useful. The validation testing would first begin in the lab and as the prototype progressed, would extend to actual field testing where beekeepers would be given an option to use the assembly on their own hives. Validation would occur at this stage by speaking with the beekeepers directly and observing how they interact with the product over the course of several weeks.

Improved Feeding Prototyping Risks

The largest prototyping risks in the improved feeding assembly would be the system not being water-tight. This is a risk since there are many electronics in the system that can be damaged by water leaks. This risk is mitigated by having sealant to fill any leaks and by actively testing the components ability to hold water. Some smaller risks include having imperfect 3D prints which affect the system's ability to hold water and increase friction in the custom air pump design which would then require more power to run. These risks are mitigated by having various types of sealants to fill holes and by sanding down components to ensure no interference. Another key risk was unexpected results from the electrical components like ones that arise from damage. A way this was mitigated was by actively reviewing the circuits with a digital multimeter/oscilloscope, understanding failures and changes in performance, insulating exposed wires and circuits to

prevent disturbance,and updating the circuits to work with their electronics current states (i.e. motor damage forcing a modification from a motor driver circuit to a simple button).

Varroa Mite Detection System

Significant Requirement	*Verification*
System can detect varroa mite with little intrusion into the beehive	Interview Support: Jarret (BC Hobbyist), Scott (BC Hobbyist), Henninger Hill Apiary, BCB Honey Farm, East Van Bees, Fr. Foster, Dr. Bixby, Carolyn (BC Apiary Operator), Alberta Urban Beekeeper, Beekind, Country Bee Apiary, Bee Natural Apiaries, EquiFlora Micro Apiary, Purrfect Honey
System can report varroa mite infection to user	Interview Support: Jarret (BC Hobbyist), Scott (BC Hobbyist), Henninger Hill Apiary, BCB Honey Farm, East Van Bees, Fr. Foster, Dr. Bixby, Carolyn (BC Apiary Operator), Alberta Urban Beekeeper, Beekind, Country Bee Apiary, Bee Natural Apiaries, EquiFlora Micro Apiary, Purrfect Honey

Current System Verification and Validation

System Verification	*Validation*
Test that the gas sensors are fully functioning and able to detect changes within the gaseous compositions	Speak with beekeepers about proposed solutions to gather opinion, generally positive feedback based on theoretical discussions.
Test that the algorithm is functional and can detect up to three different types of gas.	Validated methodology with existing research from researchers done by the University of Hohenheim.
Test that the motor collection and cleansing function can successfully intake a significant concentration of gas., and is subsequently able to clear a the gas concentration	

Future Verification and Validation

One of the first things we would aim to validate is to ensure that our hypothesis of being able to use the gas sensors to detect infected and non infected hives.

Future Verification	*Future Validation*
Test that the machine learning model is able to self-learn	Test prototype with beekeepers to see if design is acceptable
Test the ability for the system to operate wirelessly and software agnostically.	Get real-world testing by offering beekeepers the option to test the system on their infected/uninfected hives
	Receive feedback from beekeepers through their use of the device

	Ensure the gas-detection system is robust enough to work in the field
	Ensure the gas-detection system can learn and improve in cases of increased pollution or alternative environments.

Varroa Mite Verification Tests

All further verification tests were documented in the appendix.

Requirement	*Verification Test*	*Metric*
Distinguish between different gaseous compositions.	Expose sensors to different chemical compositions and read resistance values.	Change in resistance values beyond a 10% threshold
Machine learning model can learn and predict different gaseous compositions	Train machine learning models on various gaseous compositions, then run test and prediction analysis.	Training model confidence of at least 90%
Able to report to the user when a recognized change is detected.	Run prediction and check if the system is able to report the gas	Run multiple tests, visually verify that the system is reporting to the user.
Can automatically sample air data	Upon activation of the system, do not touch the system until collection and prediction is complete, and check results.	Test system multiple times, successful operation without intervention rate 90% or better.
Can self-cleanse detection chamber	Collect a sample of data, verify that the system identifies it as not-air (base sample). Run the pump system a second time, verify that the system identifies it as air (base sample).	Test system multiple times, success rate 90% or better.

Varroa Mite Validation Tests

Given the seasonality of our market, it was not possible to validate the varroa mite by testing it in the field and collecting data to train our ML model. However, we were able to validate that this type of system was one with the potential to function, based on existing research.
Moreover, we were able to validate that this solution was one which was desired by many beekeepers, indicating our progress is in the right direction.

Varroa Mite Most Significant Prototyping Risks

Shipping was a large risk as the sensors came from the USA. We mitigated this risk by ordering sensors very far ahead in advance, and preparing as much of the design as possible without the requirement of the sensors, such as the code and serial communication protocol. Ultimately, the package took a month to arrive, far longer than the anticipated shipping time, but because of these mitigation practices we were able to maintain the schedule.

The largest risk due to this being a novel system is the development of software and integration of hardware components. While it would be best to have everything designed on its own custom PCB, for the sake of this prototype we aimed to use OTS components and easily accessible microcontrollers.
We mitigate the risk of incorporating high degrees of complexity through the use of existing software libraries, such as BlueDot for bluetooth control, and the sklearn library for Machine Learning models. This way, we would have a viable prototype.
Finally, another large prototyping risk for the varroa mite system would be the mechanical design and manufacturing, which we were able to mitigate having access to a wide variety of private and public 3D printers. This gave us a high degree of flexibility to rapidly create designs as required. Evidently, given the functionality of our prototype, these mitigation methods have proved fruitful.

Climate Control System

Significant Requirement	*Verification*
Detect temperature	**Interview Support:** Jarret (BC Hobbyist), Scott (BC Hobbyist), Henninger Hill Apiary, East Van Bees, Fr. Foster, Dr. Bixby, Carolyn (BC Apiary Operator), Alberta Urban Beekeeper, Beekind
Can control temperature inside the hive	**Interview Support:** Jarret (BC Hobbyist), Scott (BC Hobbyist), Henninger Hill Apiary, East Van Bees, Fr. Foster, Dr. Bixby, Carolyn (BC Apiary Operator), Alberta Urban Beekeeper, Beekind
Report to user with the basic condition inside the hive	**Interview Support:** Jarret (BC Hobbyist), Scott (BC Hobbyist), Henninger Hill Apiary, East Van Bees, Fr. Foster, Dr. Bixby, Carolyn (BC Apiary Operator), Alberta Urban Beekeeper, Beekind

Climate Control Verification Tests

Requirement	*Verification Test*	*Metric*
Detect temperature accurate	Expose thermocouple to different environments and make sure that they are reading the correct temperature	Difference between actual temperature and the measured temperature from the thermocouple
Sense humidity	Expose humidity sensor to different environment with different humidity condition, and make sure it is giving the correct values	Difference between actual humidity and the measured humidity from the humidity sensor
Heating plate could reach 40C	Turning on the heater and measure it with a thermometer	Heating element can reach 40 C
Can control temperature inside the hive	Connect the system to power and make sure the internal temperature	The heater will turn on when the temperature is not it the ideal range,

	of the hive reaches and maintains at its ideal temperature	and will turn off when ideal temperature is reached
Report to user with the basic condition inside the hive	Connect to bluetooth via bluetooth low energy	The mobile app could detect and connect to the bluetooth module via bluetooth

Future Validation

We hope to test and validate the system in the field and get quantitative results on the percentage of bees that we are saving beekeepers.

Future Validation
Test prototype with beekeepers to see if design is acceptable and make sure that the placement of the heaters is ideal for the bees
Get real-world testing by offering beekeepers the option to test the system through the winter, get quantitative results on the bees we could save in the winter with our climate control system
Receive feedback from beekeepers through their use of the device
Ensure the climate control system is robust enough to work in the field

Climate Control Most Significant Prototyping Risks

In terms of risks, the most significant risk was using the heaters, the heaters are connected to 120VAC directly, and this in turn heats up the hive quickly. Subsequently, the heaters get too hot and could end up setting the beehive on fire. We are mitigating risks by having the thermocouples close to the heaters, and turning off the heater when the ideal temperature is reached.

IP Assessment

The beehive IP landscape has very recently begun to see more technologically innovative solutions appear in the past decade.
We will aim to patent each of our subsystems as its own solution, and also file a top-level patent which will cover the aspects of the smart hive as a whole. We are basing this methodology on existing companies and their patent patterns within this space.

Beehive IP Assessment

We will be able to patent our implementation of temperature and humidity sensors, as well as our use of wireless transmission and reception so long as the frameworks for each of these integrations do not overlap with existing implementations, which we are confident is not the case.

List of Patents

(1) US20210400925A1: BEEHIVE
(2) US10721919B2: SMART BEEHIVE SYSTEM AND METHOD OF OPERATING THE SAME

Feeding System IP Assessment

The feeding mechanism which is designed for Nectarfy is unique in various aspects giving it large differentiation from existing patented feeding solutions.Our uniqueness comes from the following factors:

- The use of motorized components
- The use of three separate tanks
- The use of a mixing tank which allows for mixing of various feeding solutions
- The use of sensors to detect fluid volume within two storage tanks
- Integration with wireless communication to report volume of fluid in the system to user.

While we have some similarities with other patents, such as using holes to drip-feed the food, as well as mounting the system on the top of hives, these claims have been verified via precedent to not be an issue. These factors give our solution a high degree of freedom to operate without infringing on any prior art, some listed below.

List of Patents

(1) US20210337774A1: Drip-feeding device which has holes in a plate.
(2) US11109575B2: Simple drip feed system with a maze-like structure preventing intruders.
(3) US6830499B1: Drip-feeding device with a heat-absorptive housing with a hamper, and has a plurality of holes on the bottom to let feed through.

Climate Control IP Assessment

The climate control system within Nectarfy has more prior art to consider. However, there are unique aspects to Nectarfy's approach which provides flexibility in patenting:

- Integration of heating plates into the box portion of the beehive structure
- Thermocouples and other heating sensors to self-regulate beehive temperatures
- Integration with wireless communication to control and report beehive temperature

Other heating methods, while often also using similar technologies, implement those technologies differently. For instance, some heating methods use heaters to circulate hot air, while others use heating elements within a frame inside the box, different from our implementation of incorporating the heater on the sides of the hive within the bee box. The list of

patents below highlights some of the most similar inventions, but given the explanations above, we are able to claim no infringement.

List of Patents

(1) US20080064298A1: Replaces a frame with a ceramic block with integrated heating.
(2) JP6624624B2: Replaces a frame with a system which absorbs and emits IR radiation.
(3) US20190289830: Uses a bottom plate with an electric heating element and material.

Varroa Mite Detection IP Assessment

Nectarfy's varroa mite detection system is unique compared to existing solutions for the following aspects:

- Use of gas sensors
- Use of a air pump and motor
- Use of a machine learning model to calculate gassed on gas sensor data
- Automated deception without the need of human intervention

Other methods of varroa mite detection include using physical products which assist in physical examination or using cameras to detect the mites. These methods are drastically different from our approach which uses gas sensors. Moreover, based on precedent on other patents, we aim to patent our use of machine learning for gas sensor data for varroa mite detection, something which is also novel and shows little prior art.

List of Patents

(1) US20150049919A1: (Abandoned) An device which uses cameras to detect varroa.
(2) US20210289765A1: Automatic detection and elimination of mites via lasers.
(3) US10874090B2: A box which causes varroa to fall off bees for accounting.

IP Strategy

Given the wide variety of possible patents applicable to the Nectarfy hive, the Nectarfy team aims to approach patenting in a very formulaic approach. While we are acquiring money to prepare these patents, we plan to operate in stealth mode for as long as possible.

The first patent we aim to file is the patent which protects our entire hive, much like the patent US20210400925A1, which protects our entire beehive system as a whole. This would allow Nectarfy to carve a spot in a quickly-growing patent landscape. In parallel, the Nectarfy team aims to trademark the Nectarfy company and its assets. "Nectarfy" is currently not a registered trademark.

Following such, funds would be allocated to secure the most unique aspect of the beehive, which is the varroa mite detection system. Because of the importance of this feature for our customers, alongside the unique method in which Nectarfy aims to detect varroa, this piece of technology would be highly important to patent.

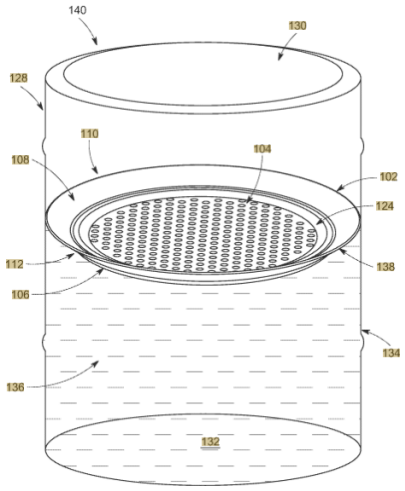
Finally, we would aim to hold patents for the feeding and heating system respectively. Further down the line once more cash has been acquired and key patents, which include subsystem hives, we plan to protect the "look and feel" of the software application and mechanical design of the beehive using design patents, trade dress, and industrial designs.

Appendices

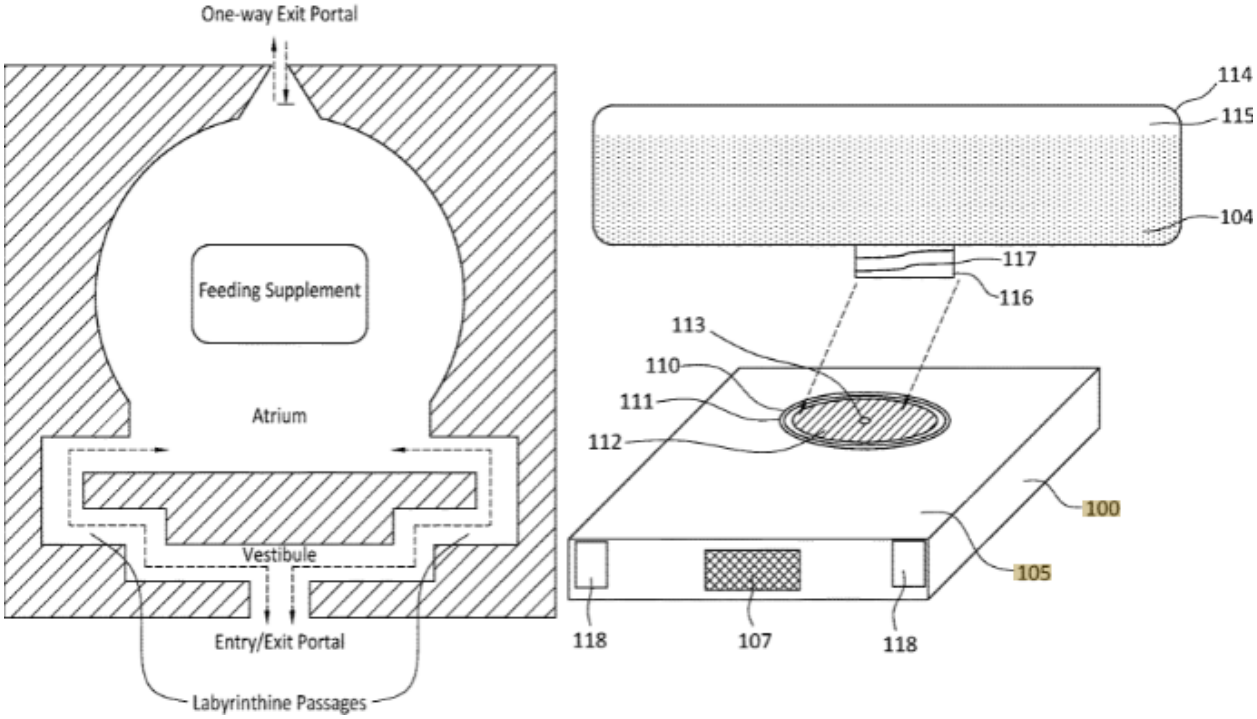
Appendix A: IP Assessment List

Feeding System

US20210337774A1: Drip-feeding device which has holes in a plate resting on top of syrup.



US11109575B2: Simple drip feed system with a maze-like structure prevents intruders from entering.



US6830499B1: Drip-feeding device with a heat-absorptive housing with a hamper, and has a plurality of holes on the bottom to let feed through.

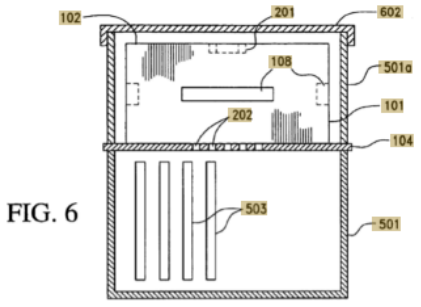


FIG. 6

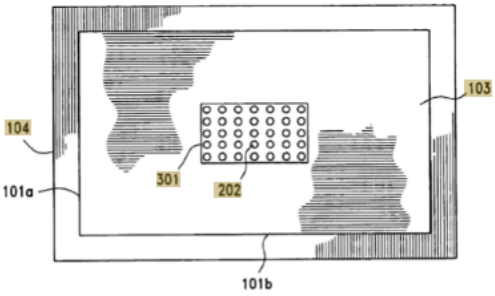


FIG. 3

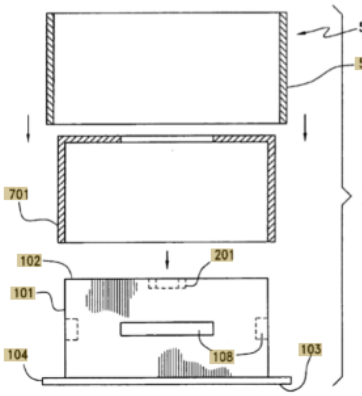
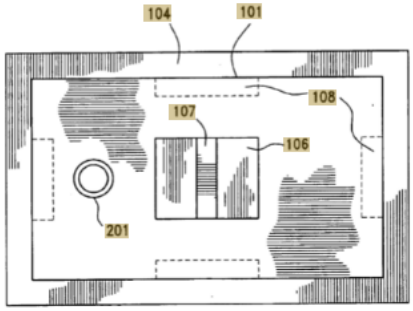
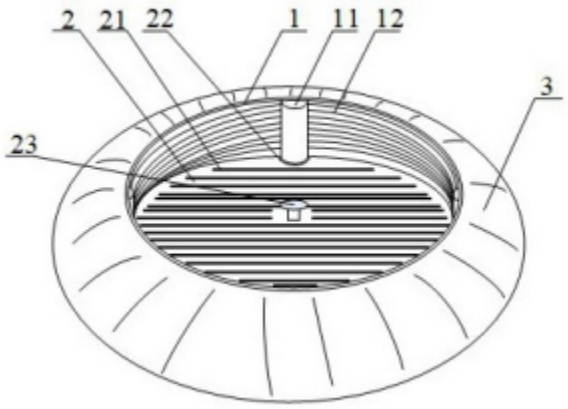


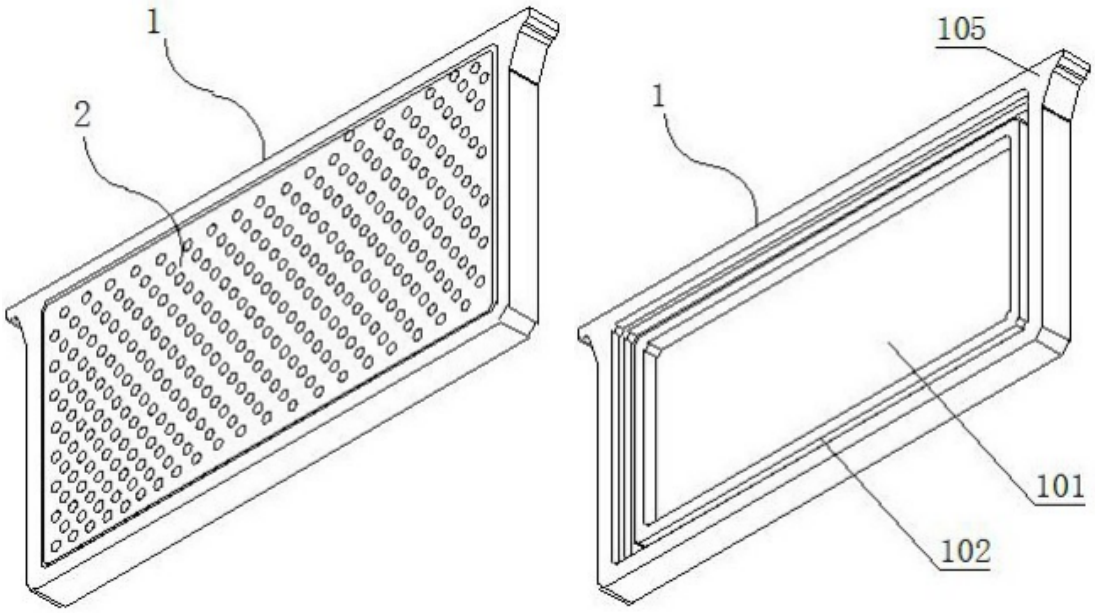
FIG. 7



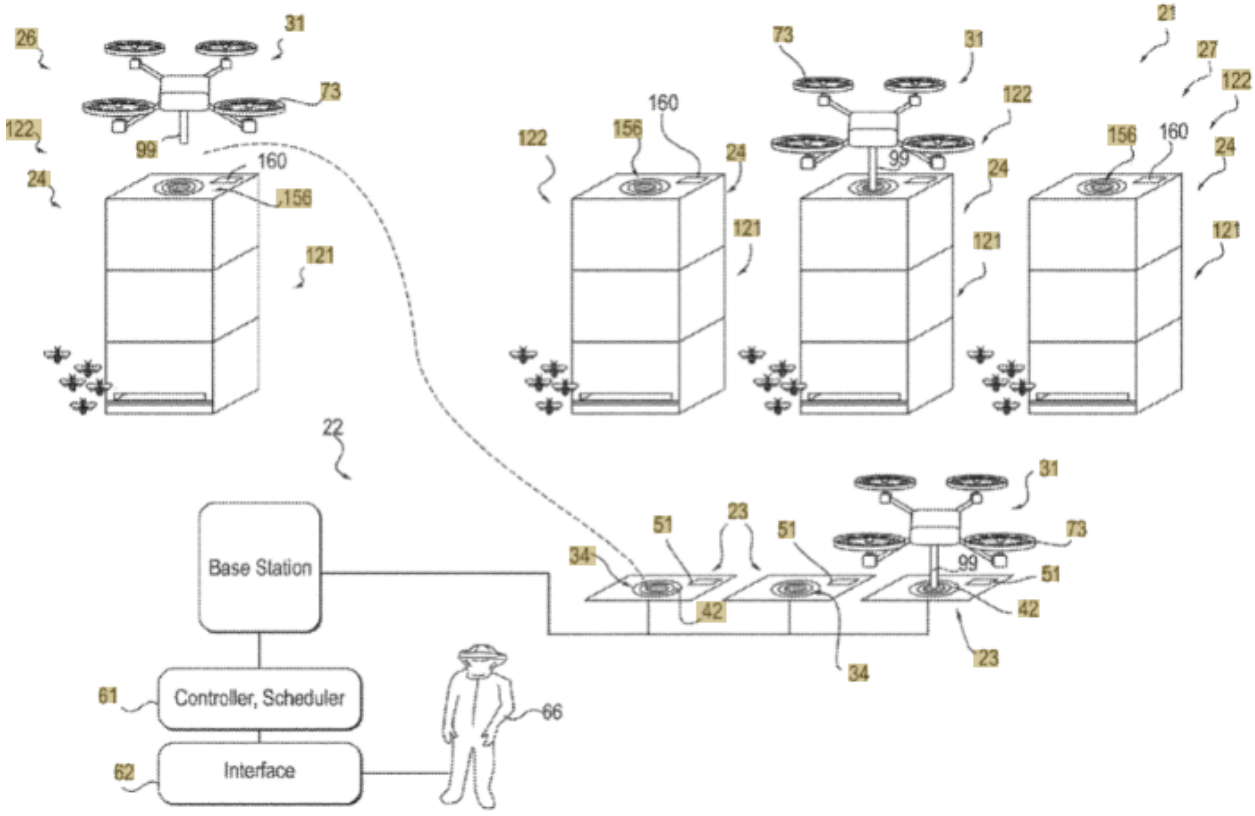
CN213523475U: The utility model provides a bee feeder, which comprises a box body, a floating plate and a fixed plate; the box body is cylindrical and is placed in the fixing plate, two pairs of semi-cylindrical clamping strips which are symmetrical to each other are arranged on the inner side wall of the box body, and the inner side wall of the box body is also provided with grain strips, so that bees can climb to fetch food conveniently.



CN210247979U:The utility model relates to a multifunctional bee feeder, belonging to the technical field of artificial breeding and feeding of bees; the feeding device comprises a feeder bottom plate and a feeder cover plate, wherein a concave trough is manufactured in the feeder bottom plate, and a circle of cover plate fixing clamping grooves are formed in the periphery of the concave trough. The feeding replaces a frame within the beehive to feed bees instead of dripping from the top.

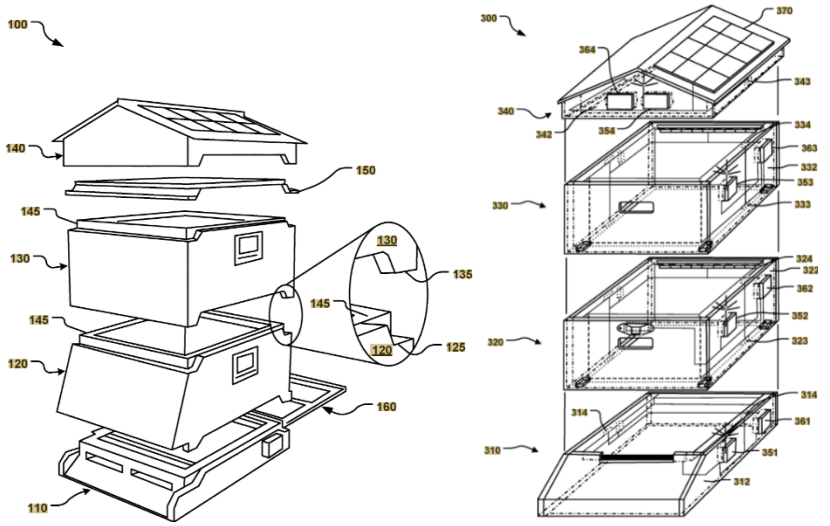


EP2523549A1: Bee feeder with automatic pressure control, consisting of a container, ducts, feed head with pressure control, pump, filter, timer switch, pressure indicator and closing valves.
US10717529B2: Uses a drone to apply feeding fluid to beehives.

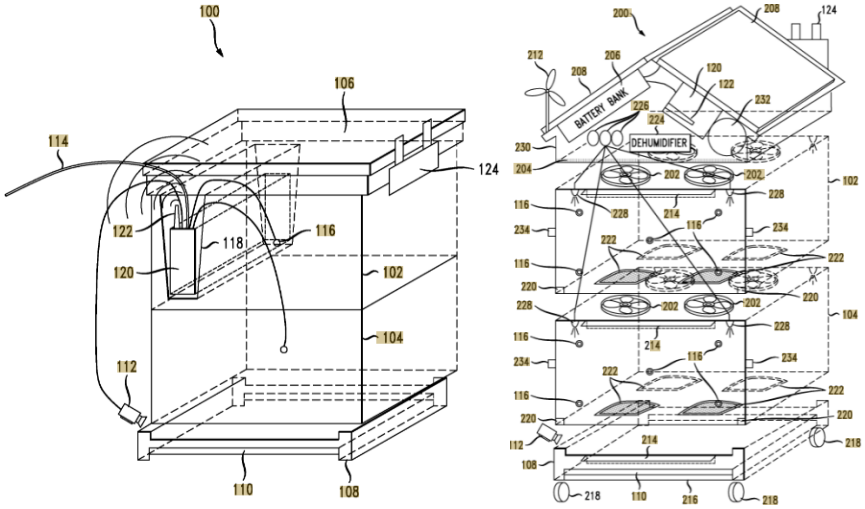


Beehive

US20210400925A1: BEEHIVE, Applicant HyperHyveLLC, Inventor Micheal James Harvey (Published (2021))

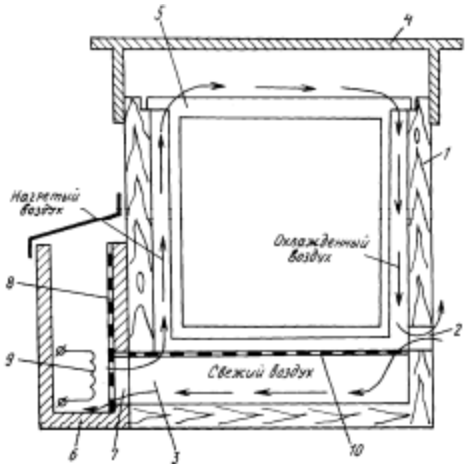


US10721919B2: SMART BEEHIVE SYSTEM AND METHOD OF OPERATING THE SAME, Applicant: Best Bees Company, Inventory: Noah Wilson-Rich (Published 2020)

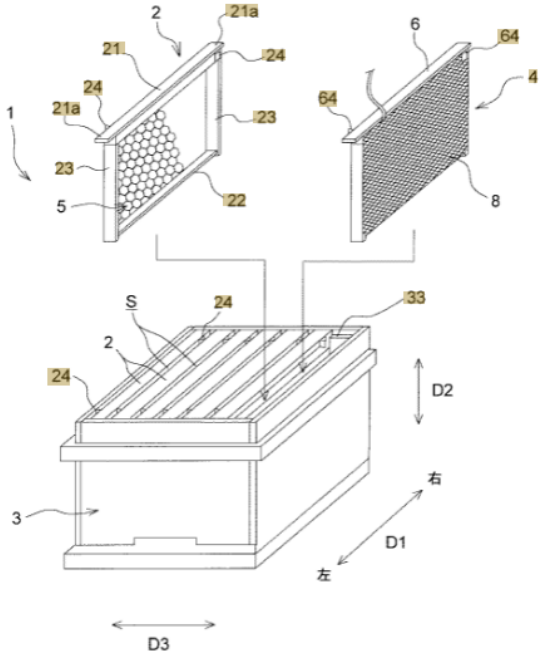


Climate Control:

RU2271654C2: A bee hive with electric heating, comprising a hull with a notch, a cover, frames, a metal screen in the bottom and an electric heater which is closed by a metal shielding net.



JP6624624B2: The present invention relates to a honeycomb housing heater and a honeycomb housing provided with the same. A honeycomb box heater that heats the inside of the box by absorbing and absorbing the far-infrared energy emitted from the panel heater by the wood constituting the box.



US20190289830: A heating tray substantially inserted through the slotted entrance of said hive body and residing adjacent the floor of said hive body, said heating tray comprising an electric heating element and a heat-conductive material designed to distribute heat generated by said heating element across a surface area of said heating tray.

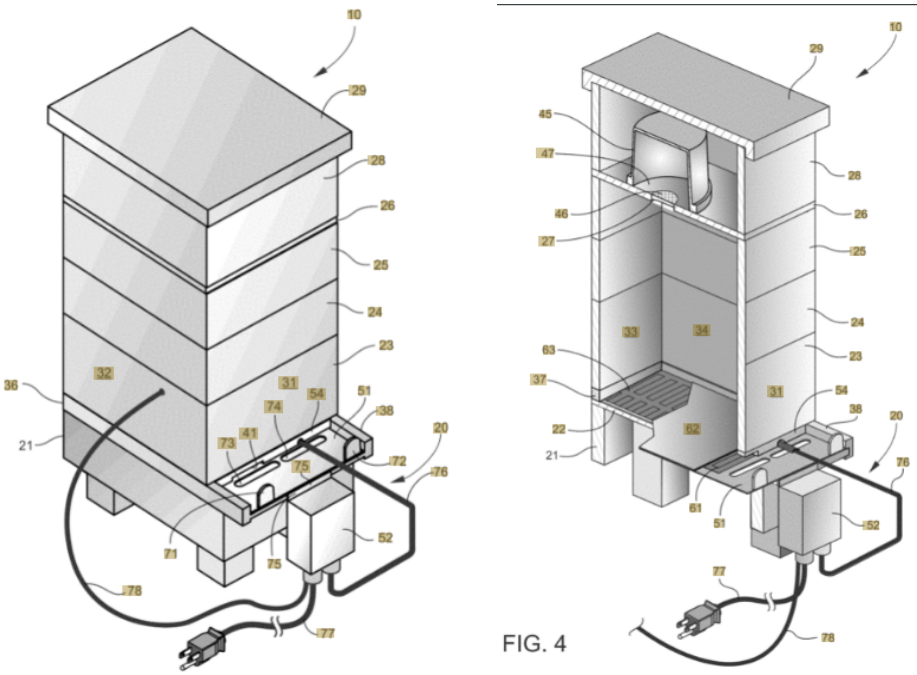
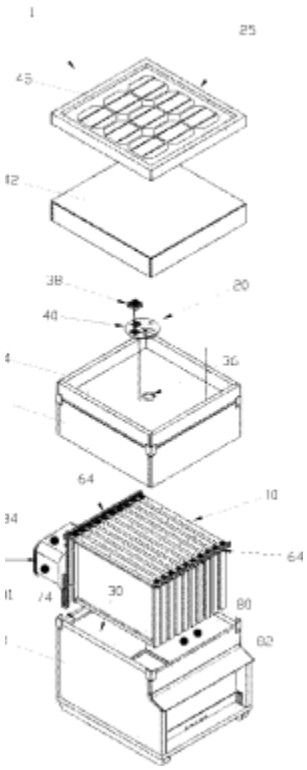


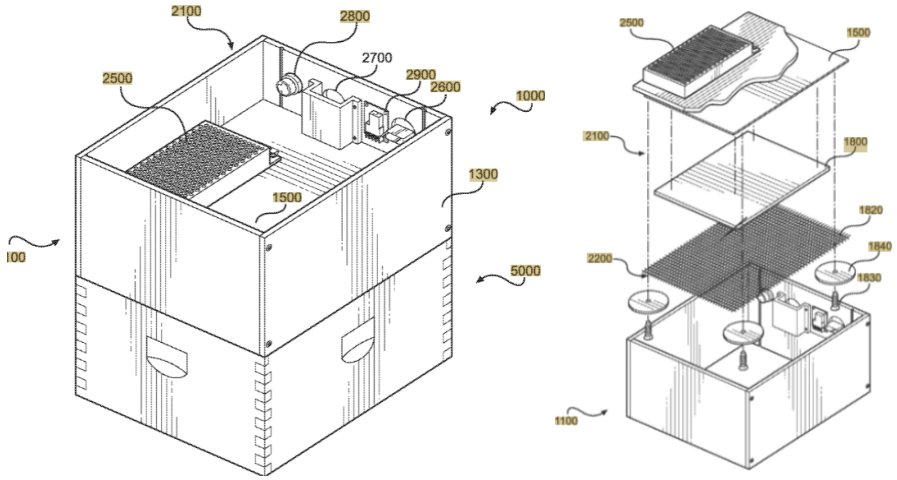
FIG. 4

PL230342B1: A system for heating bee colonies, characterized in that it comprises a frame heater II, a heating plate I and a universal heater III, which elements are intended for independent use, using a safe voltage up to 50V AC or 120V DC.

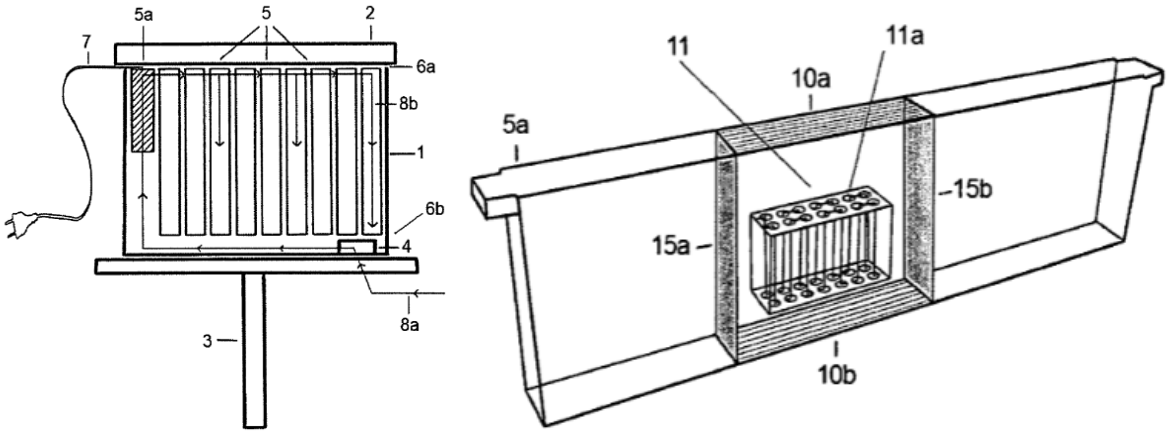
WO2015087198A1:Has heating frames which sit within the device, replacing a honeycomb frame.



US11129370B1: A thermodynamic terminator and a method of eliminating mites and parasites within a bee box. The thermodynamic terminator includes a housing having a lower panel with a fan aperture, an air intake, and an interior wall for supporting a divider panel. This patent is more focused on varroa mite treatment, but it still heats the hive and thus should be considered.



US20080064298A1: DEVICE FOR HYGIENIZATION, WARMING, AND DEHUMIDIFYING A BEEHIVE. Uses one frame for inserting a ceramic block 11 provided with at least one row being thermally insulated from the remaining frame with the aid of wings and traversed by conducting filaments connected to an energy source through wire, whereby whenever electrical current traverses said filaments, the temperature inside said channels attains from 300-350° C.



Varroa Mite Detection:

US 20210289765 A1: Automatic detection and elimination of mites through laser exposure is provided. In various embodiments, light is emitted into a target area from at least one light source. Light reflected from a target within the target area is detected by at least one photodetector.

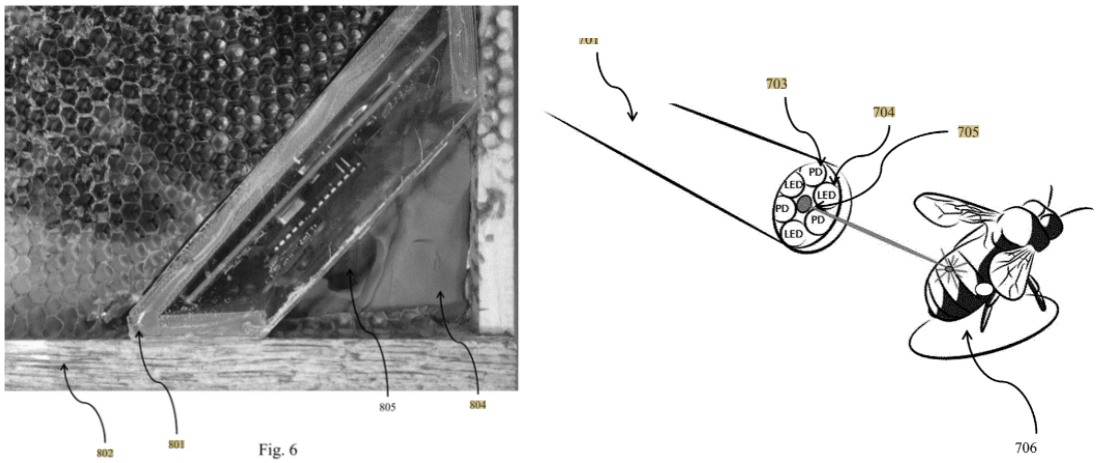
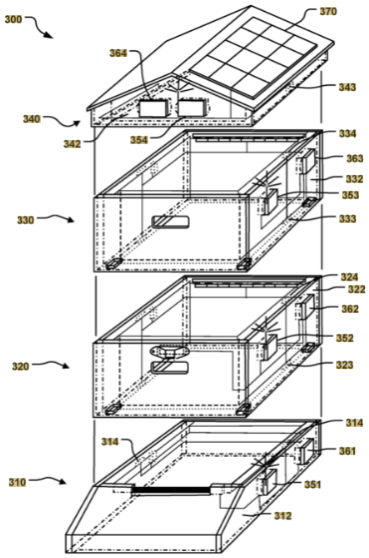
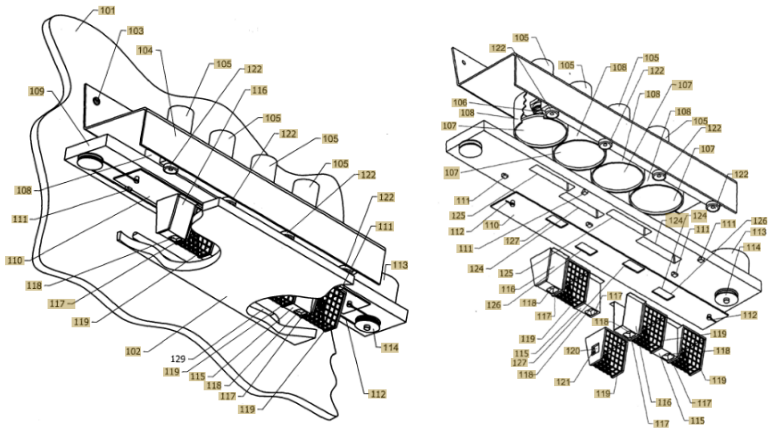


Fig. 6

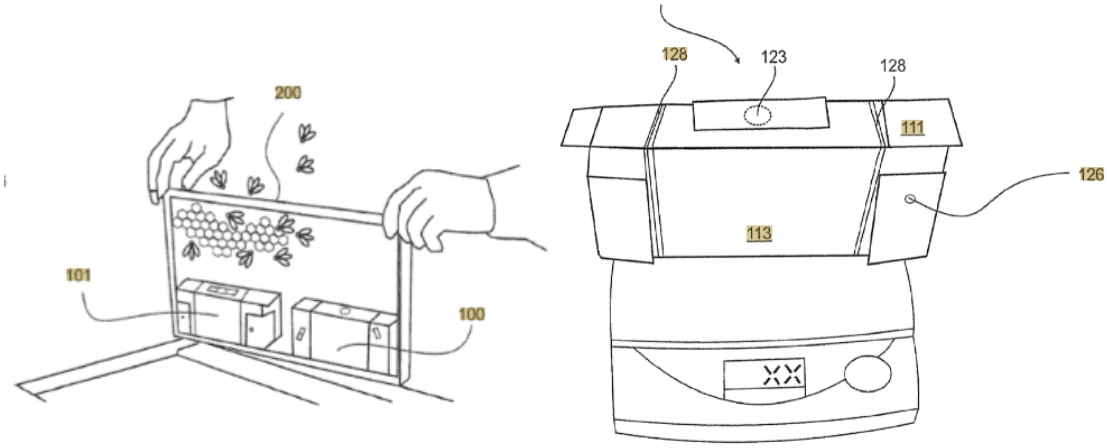
US 20210400925 A1: The system also includes a video and/or audio component, with connectivity to remote server(s), that allows for remote data collection, video and/or audio analysis and can provide hive command(s) such as heat treatments, ventilation, and similar.



US 20150049919 A1: (Abandoned) An device for diagnosis and control of honeybee varroatosis (Varroa mite infection) comprises one or several cameras (310), connected to image processor (311). The software of the device is capable of processing the picture of honeybee, recognition the presence or absence of varroa mite on the body of the bee on the field-of-view of the cameras or in the chamber for observation, counting the infected and non-infected bees, separating the infected bees from non-infected bees or killing the recognized varroa mites.



US10874090B2: The device contains a container with a bottom, side walls and a cover. Within the container there is a means which causes the Varroa mites to drop off the bees, but does not harm the bees, and a means for fixing the dropped Varroa mites to the bottom of the container.



CN104749219A: AI for honey monitoring, A honey detection method by support vector machine classifier parameter selection based on particle swarm optimization. This patent is one of the patents currently on the market which patents machine learning, which we can use to learn how we can patent our machine learning algorithm for varroa mite detection.

Appendix A1: Full IP Assessment

The beehive IP landscape has very recently begun to see more technologically innovative solutions appear in the past decade.
We will aim to patent each of our subsystems as its own solution, and also file a top-level patent which will cover the aspects of the smart hive as a whole. We are basing this methodology on existing companies and their patent patterns within this space. Moreover, this method will allow us a greater degree of control when it comes to selling our products as modular components, if our company wishes to sell each subsystem as its own standalone product.

Beehive IP Assessment

There are currently multiple patents which exist at least in the US patent office which have patent entire beehive systems. While these patents have similar claims to each other as the subsystems and communication protocols of each beehive are not unique, they are able to protect the overall function and certain specific aspects of their design.

We are confident given this precedence that we will be able to patent the entire system of our smarthive, which includes how the electronic systems within the beehive are constructed how the hive itself is built and its mechanical function which includes our “easy-access” electronic shelf and how it integrates its subsystems through the hardware of the beehive itself. Moreover, we would also be able to patent the method in which the beehive communicates and connects with other smart devices including the user-facing application.

Given the precedence seen in these claims of the beehive, we will be able to patent our implementation of temperature and humidity sensors, as well as our use of wireless transmission and reception so long as the frameworks for each of these integrations do not overlap with the existing implementations, which we are confident is not the case.

List of Patents

(3) US20210400925A1: BEEHIVE
(4) US10721919B2: SMART BEEHIVE SYSTEM AND METHOD OF OPERATING THE SAME

Feeding System IP Assessment

The feeding mechanism which is designed for Nectarfy is unique in various aspects giving it large differentiation from existing patented feeding solutions.Our uniqueness comes from the following factors:

- The use of motorized components
- The use of three separate tanks
- The use of a mixing tank which allows for mixing of various feeding solutions
- The use of sensors to detect fluid volume within two storage tanks
- Integration with wireless communication to report volume of fluid in the system to user.

While we have some similarities with other patents, such as using holes to drip-feed the food, as well as mounting the system on the top of hives, these claims have been verified via precedent to not be an issue. There was only one other patent which used pressure controls and pumps in their solution, but the patent was withdrawn in 2012, however, that patent did not outline any mixing features.
These factors give our solution a high degree of freedom to operate without infringing on any prior art, some listed below.

List of Patents

(4) US20210337774A1: Drip-feeding device which has holes in a plate.
(5) US11109575B2: Simple drip feed system with a maze-like structure preventing intruders.
(6) US6830499B1: Drip-feeding device with a heat-absorptive housing with a hamper, and has a plurality of holes on the bottom to let feed through.

Climate Control IP Assessment

The climate control system within Nectarfy has more prior art to consider. However, there are unique aspects to Nectarfy's approach which provides flexibility in patenting. These unique aspects include:

- Integration of heating plates into the box portion of the beehive structure
- Thermocouples and other heating sensors to self-regulate beehive temperatures
- Integration with wireless communication to control and report beehive temperature

Other heating methods, while often also using similar technologies, implement those technologies differently. For instance, some heating methods use heaters to circulate hot air, while others use heating elements within a frame inside the box, different from our implementation of incorporating the heater on the sides of the hive within the bee box. The list of patents below highlights some of the most similar inventions, but given the explanations above, we are able to claim no infringement.

List of Patents

(4) US20080064298A1: Replaces a frame with a ceramic block with integrated heating.
(5) JP6624624B2: Replaces a frame with a system which absorbs and emits IR radiation.
(6) US20190289830: Uses a bottom plate with an electric heating element and material.

Varroa Mite Detection IP Assessment

An important distinction for this subsection is that we are aiming to look at methods of how technology is used to detect varroa mite as opposed to the technology which treats varroa mite. Of course, for the future prototype, the treatment system will need an IP assessment, some patents which are discussed in the appendix. Nectarfy's varroa mite detection system is unique compared to existing solutions for the following aspects:

- Use of gas sensors
- Use of a air pump and motor
- Use of a machine learning model to calculate gassed on gas sensor data
- Automated deception without the need of human intervention

Other methods of varroa mite detection include using physical products which assist in physical examination or using cameras to detect the mites. These methods are drastically different from

our approach which uses gas sensors. Moreover, based on precedent on other patents, we aim to patent our use of machine learning for gas sensor data for varroa mite detection, something which is also novel and shows little prior art.

List of Patents

(4) US20150049919A1: (Abandoned) An device which uses cameras to detect varroa.
(5) US20210289765A1: Automatic detection and elimination of mites via lasers.
(6) US10874090B2: A box which causes varroa to fall off bees for accounting.

IP Strategy

Given the wide variety of possible patents applicable to the Nectarfy hive, the Nectarfy team aims to approach patenting in a very formulaic approach. While we are acquiring money to prepare these patents, we plan to operate in stealth mode for as long as possible.
The first patent we aim to file is the patent which protects our entire hive, much like the patent US20210400925A1, which protects our entire beehive system as a whole. This would allow Nectarfy to carve a spot in a quickly-growing patent landscape.

In parallel, the Nectarfy team aims to check the trademark for the Nectarfy company. After some searching on the Canadian and United States Trademark system, “Nectarfy” is currently not a registered trademark.

Following such, funds would be allocated to secure the most unique aspect of the beehive, which is the varroa mite detection system. Because of the importance of this feature for our customers, alongside the unique method in which Nectarfy aims to detect varroa, this piece of technology would be highly important to patent.

Finally, we would aim to hold patents for the feeding and heating system respectively. Further down the line once more cash has been acquired and key patents, which include subsystem hives, we plan to protect the “look and feel” of the software application and mechanical design of the beehive using design patents, trade dress, and industrial designs.

Beehive IP Assessment

There are currently multiple patents which exist at least in the US patent office which have patent entire beehive systems. While these patents have similar claims to each other as the subsystems and communication protocols of each beehive are not unique, they are able to protect the overall function and certain specific aspects of their design.

We are confident given this precedence that we will be able to patent the entire system of our smarthive, which includes how the electronic systems within the beehive are constructed how the hive itself is built and its mechanical function which includes our “easy-access” electronic shelf and how it integrates its subsystems through the hardware of the beehive itself. Moreover, we

would also be able to patent the method in which the beehive communicates and connects with other smart devices including the user-facing application.

Given the precedence seen in these claims of the beehive, we will be able to patent our implementation of temperature and humidity sensors, as well as our use of wireless transmission and reception so long as the frameworks for each of these integrations do not overlap with the existing implementations, which we are confident is not the case.

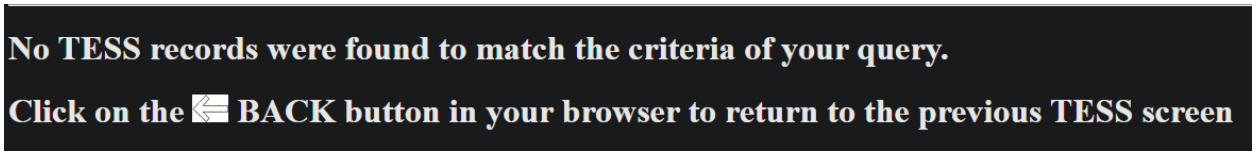
List of Patents

(5) US20210400925A1: BEEHIVE
(6) US10721919B2: SMART BEEHIVE SYSTEM AND METHOD OF OPERATING THE SAME

Comparison of Patents for Beehive
As an example of the closely aligned claims between each patent, we can see that the first patent (1) claims:
An insulated beehive assembly comprising :
a base section , at least one box , and a roof section ...**the monitoring system comprising a humidity sensor , a temperature sensor , a**
wireless communication receiver and a wireless communication transmitter ; and
a power source operably connected to the monitoring system . The insulated beehive assembly of claim 1 , wherein each of the base section , the at least one box , and the roof section **has a humidity sensor and a temperature sensor.**

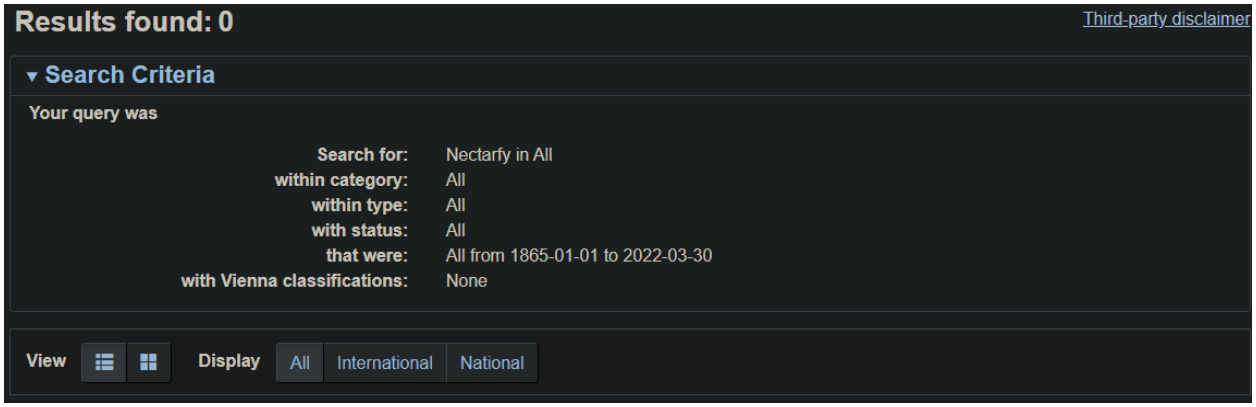
On the other hand, patent (2) claims:
A beehive comprising … **one or more sensors; a receiver/transmitter device** coupled with the computing device; and at least one memory that stores instructions that are executed by the at least one processor to: **receive signals from the one or more sensors, wherein the one or more sensors comprise an ion trap mass spectrometer or a humidity detector**;
2. The beehive of claim 1, wherein the one or more sensors comprises a **temperature sensor…**

Trademarks:
We will name each subsystem based on the “Nectarfy” name, and thus the first search should be to check if Nectarfy is already trademarked.
From the US Patent and Trademark Office:
Record List Display (uspto.gov)
Searching for “Nectarfy” yields:



From the Canadian Trademarks Database:
Canadian Trademarks Database

Searching for Nectarfy yields:



Therefore, we are confident we can use Nectarfy as a trademark, and thus name the subsystems simply based on the Nectarfy name. For instance, the “Nectarfy Varroa System”, or the “Nectarfy App” could be easily trademarked as well in the current climate.

Appendix B: Varroa Mite Detection System Code

The final code used on the Arduino and Raspberry Pi are found here. With this code, an engineer would be able to replicate the system we devised for the varroa mite detection system.
See repository (Here) for all code including legacy and updated notes.
Arduino Code:

```
int ZeroVal=0;
int ThreeVal=0;
int TwoVal=0;
int TwentyVal=0;
int EightVal=0;
int ebyte=255;
int incomingByte=0;
int runtime=0;
int i=0;

int ZeroPin=A0;//TGS2600
int ThreePin=A1;//TGS2603
int TwoPin=A2;//TGS2602
int TwentyPin=A3;//TGS2620
int EightPin=A4; //TGS832
int pumpPin=2;
int airPin=10;
int ethPin=11;

void setup() {
  Serial.begin(9600);
  pinMode(LED_BUILTIN, OUTPUT);
  pinMode(pumpPin,OUTPUT);
  pinMode(airPin,OUTPUT);
  pinMode(ethPin, OUTPUT);


}

void loop() {
  if (Serial.available() > 0) {
    incomingByte = Serial.read();
    if (incomingByte=='H'){
      while(true){
        ZeroVal=analogRead(A0);
        ThreeVal=analogRead(A1);
        TwoVal=analogRead(A2);
```

```
        TwentyVal=analogRead(A3);
        EightVal=analogRead(A4);

        Serial.println(ZeroVal);
        Serial.println(ThreeVal);
        Serial.println(TwoVal);
        Serial.println(TwentyVal);
        Serial.println(EightVal);
        Serial.println(ebyte);
        delay(500);
        incomingByte = Serial.read();
        if(incomingByte=='L'){
          digitalWrite(LED_BUILTIN, HIGH);
          break;
        }
      }
    incomingByte=0;
    }
    if (incomingByte=='M'){
      while(true){
        digitalWrite(airPin,LOW);
        digitalWrite(ethPin,LOW);
        digitalWrite(pumpPin,HIGH);
        incomingByte = Serial.read();
        if(incomingByte=='O'){
          digitalWrite(pumpPin,LOW);
          break;
        }
      }
    incomingByte=0;
    }
    if(incomingByte=='A'){
      digitalWrite(airPin,HIGH);
    }
    if(incomingByte=='E'){
      digitalWrite(ethPin,HIGH);
    }
  }

}
```

Python Gas Sensor Self-Made Library

```
# -*- coding: utf-8 -*-
```

```
"""
Created on Wed Mar  2 15:46:16 2022
Collect the location and address of the Arduino board by using:
ls /dev/tty* in the terminal
@author: Thomas
"""
#Libraries are in!
import serial;
import time;
import csv;
from queue import Queue
from MLModel import gasPredictor
#Serial Port Name
     #Change the name here which will control the name in subsequent
declarations
     #For Thomas' Laptop, it is COM8
     #For the RPi, it is /dev/ttyUSB0
#Initialize queues
# dataQueue=Queue(maxsize=0)
# ZeroData=Queue(maxsize=0)
# ThreeData=Queue(maxsize=0)
# TwoData=Queue(maxsize=0)
# TwentyData=Queue(maxsize=0)
# EightData=Queue(maxsize=0)
#Initialize label array
arrayLabel=[]
#Functions
def avgQueue(queue):
    size=queue.qsize()
    arraySum=0
    while True:
        try:
            arraySum=arraySum+queue.get_nowait()
        except:
            avg=float(arraySum/size)
            break
    return avg
def CreatePredArray(self,
stateval,ZeroData,ThreeData,TwoData,TwentyData,EightData):
    if(stateval==1):
        #Average all the queues
        ZeroAvgVal=avgQueue(ZeroData)
        ThreeAvgVal=avgQueue(ThreeData)
```

```
        TwoAvgVal=avgQueue(TwoData)
        TwentyAvgVal=avgQueue(TwentyData)
        EightAvgVal=avgQueue(EightData)

predArray=[ZeroAvgVal,ThreeAvgVal,TwoAvgVal,TwentyAvgVal,EightAvgVal]
    else:
        print("You will be prompted to enter five values.")
        predArray=[]
        for i in range(0, 5):
            while True:
                try:
                    val = int(input("Please input value: "))
                    break
                except:
                    print("That is not a valid integer. Try again.")
            predArray.append(val)
    return predArray
def dataCollect(self,timeCollect,dataQ,sPort):
    #Init Serial
    #Here, the port needs to be modified.
    #/dev/ttyUSB0
    arduino = serial.Serial(port=sPort, baudrate=9600)
    arduino.flushInput()
    print("Synchronizing for sensor reading....standby")
    time.sleep(3)
    arduino.write(b'H')
    ##!! Add Safety Feedback when failure occurs!!#
    print("Synchronization complete")
   #Change the time value as required
    #Currently gives 6 full rows of data and one row of garbage values
    #Time delay after every sent packet is 500ms on Arduino End
    #May need to calc for complexity for more accurate results
    t_end=time.time()+timeCollect
    try:
        while time.time()<t_end:
            data=arduino.readline()
            decoded_bytes = float(data[0:len(data)-2].decode("utf-8"))
            dataQ.put(decoded_bytes)
        arduino.write(b'L')
        arduino.close()
    except:
        print("\n !!!WARNING There was a data collection issue. Please
reboot kernel. WARNING!!!\n")
```

```
        arduino.close()
def titlePrint():
    isOverwrite=input("Do you want to overwrite existing csv?(Y/N)")
    if(isOverwrite=="Y"):
        with open("test_data.csv","w", newline='') as f:
            writer = csv.writer(f,delimiter=",")

writer.writerow(["TGS2600","TGS2603","TGS2602","TGS2620","TGS832A",
"Label"])
    elif(isOverwrite=="N"):
        with open("test_data.csv","a", newline='') as f:
            writer = csv.writer(f,delimiter=",")

writer.writerow(["TGS2600","TGS2603","TGS2602","TGS2620","TGS832A",
"Label"])
    else:
        print("That is not a valid input.")
        titlePrint()
def titlePrintBT():
    with open("test_data.csv","w", newline='') as f:
        writer = csv.writer(f,delimiter=",")
        writer.writerow(["TGS2600","TGS2603","TGS2602","TGS2620","TGS832A",
"Label"])
def
dataPrint(self,label,dataQueue,ZeroData,ThreeData,TwoData,TwentyData,EightD
ata):
    while(dataQueue.empty()==False):
        try:
            val=dataQueue.get_nowait()
            if (val==255.0):
                try:
                    zeroval=dataQueue.get_nowait()
                except:
                    zeroval=None
                try:
                    threeval=dataQueue.get_nowait()
                except:
                    threeval=None
                try:
                    twoval=dataQueue.get_nowait()
                except:
                    twoval=None
                try:
```

```
                    twentyval=dataQueue.get_nowait()
                except:
                    twentyval=None
                try:
                    eightval=dataQueue.get_nowait()
                except:
                    eightval=None
                with open("test_data.csv","a", newline='') as f:
                    writer = csv.writer(f,delimiter=",")

writer.writerow([zeroval,threeval,twoval,twentyval,eightval,label])
                ZeroData.put(zeroval)
                ThreeData.put(threeval)
                TwoData.put(twoval)
                TwentyData.put(twentyval)
                EightData.put(eightval)
        except:
            print("No more items in Queue")
def motorRun(self,motortime,sPort):
    #Init Serial
    #The serial port in the function will have to change for the device
plugged into the Arduino
    #The location of the Arduino is at /dev/ttyUSB0, and on Thomas' PC it
is on "COM8"
    arduino=serial.Serial(port=sPort, baudrate=9600)
    arduino.flushInput()
    print("Synchronizing for motor control...standby")
    time.sleep(3)
    print("Sychronization complete. Motor Start...")
    arduino.write(b'M')
    time.sleep(motortime)
    arduino.write(b'O')
    print("Motor stopped.")
    arduino.close()
def ledTrigger(self,ledval,sPort):
    arduino=serial.Serial(port=sPort, baudrate=9600)
    arduino.flushInput()
    time.sleep(3)
    #Change values here
    if(ledval==0):
        arduino.write(b'A')
        print("AIR")
    if(ledval==1):
```

```
        arduino.write(b'E')
        print("ETH")
    arduino.close()
```

Machine Learning Model Python Code

The following code contains the machine learning model used for the Varroa mite system. Please ensure that the CSV used for “dataAir” (line 50) is updated.

```
# -*- coding: utf-8 -*-
"""
Created on Fri Mar  4 12:16:58 2022
9783
@author: Thomas
"""
#Import libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

#Import from SKLearn
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import confusion_matrix,accuracy_score

#Define predictor function (CHANGE!!)
def gasPredictor(array):
    #Change the predictor machine when necessary
    predictionNT=knn_clf.predict(np.array([array]))
    #predictionNT=nb_clf.predict(np.array([array]))
    prediction=le.inverse_transform(predictionNT)
    totpred=[prediction,predictionNT]
    return(totpred)

def gasModelNaiveBayes():
    nb_clf = GaussianNB()
    nb_clf.fit(X_train, Y_train)
    Y_pred=nb_clf.predict(X_test)
    scores = cross_val_score(nb_clf, X_train, Y_train, scoring='accuracy',
cv=10)
    print(scores)
```

```
    print('Averaged prediction accuracy for Naive Bayes = ',
np.average(scores))
    #Check confusion matrix
    cm=confusion_matrix(Y_test,Y_pred)
    ac=accuracy_score(Y_test,Y_pred)

def gasKNN():
    #Try K Nearest Neighbors
    from sklearn.neighbors import KNeighborsClassifier
    knn_clf = KNeighborsClassifier(n_neighbors=20) # change n_neighbors;
boundary becomes smoother with increasing value of K
    knn_clf.fit(X_train, Y_train)
    scores = cross_val_score(knn_clf, X_train, Y_train, scoring='accuracy',
cv=5)
    print(scores)

#Import csvs, add more if needed
dataAir=pd.read_csv('Data_0329_AirEth.csv')

#Split into input and output
inputData=dataAir.iloc[:,:-1].values
outputData=dataAir.iloc[:,-1].values

#Encode the output label
le=LabelEncoder()
outputData=le.fit_transform(outputData)

#Split Data
X_train, X_test, Y_train, Y_test = train_test_split(inputData, outputData,
test_size = 0.2, random_state = 42)

#Feature Scaling
#I am attempting feature scaling now
#Use Standardscaler
#sc=StandardScaler()
#X_train=sc.fit_transform(X_train)
#X_test=sc.transform(X_test)
#Training models
#I will first try using a Naive Bayes classifier

#Okay Naive Bayes kinda sucked I'm going to try using ANN instead
#ANN stored within the function, because it kinda sucked as well.
#Update 03-29 I'm going to try Naive Bayes again but with scaling
```

```
#KNN still best
from sklearn.neighbors import KNeighborsClassifier
knn_clf = KNeighborsClassifier(n_neighbors=20) # change n_neighbors;
boundary becomes smoother with increasing value of K
knn_clf.fit(X_train, Y_train)
scores = cross_val_score(knn_clf, X_train, Y_train, scoring='accuracy',
cv=5)
print(scores)
```

Python Computer Code (Non-Bluetooth)

This code is used for computer control of the gas sensor array, and also used to collect data to save as a csv. This code can also be used to run predictions as a form of debugging for the hardware system. Ensure that the COM port (Line 18) is correct for use, and if a csv is to be saved, to make sure that "test_data.csv" is caved as a different file name for retention.

```
# -*- coding: utf-8 -*-
"""
Created on Wed Mar 16 15:25:43 2022
@author: Thomas Guan
"""
#Libraries are in!
import serial;
import time;
import csv;
from queue import Queue
from MLModel import gasPredictor
from GasSensorLib import
titlePrint,motorRun,dataCollect,dataPrint,CreatePredArray,ledTrigger
#Serial Port Name
     #Change the name here which will control the name in subsequent
declarations
     #For Thomas' Laptop, it is COM8, for Desktop is COM5
     #For the RPi, it is /dev/ttyUSB0

sPort='COM5'
#Initialize queues
dataQueue=Queue(maxsize=0)
ZeroData=Queue(maxsize=0)
ThreeData=Queue(maxsize=0)
TwoData=Queue(maxsize=0)
TwentyData=Queue(maxsize=0)
EightData=Queue(maxsize=0)
#Initialize label array
arrayLabel=[]
```

```
#State Value Variables
stateVal=0
numSample=0
motortimeCollect=10
timeCollect=0

#Start of Program: Ask for state of program:
while True:
    state=input("Please select: Collection mode(C) or Prediction Mode(P) or
Quit(Q)\n")
    #Collection Mode
    if(state=="C"):
        input("Entering collection mode. Press enter to continue.")
        while True:
            try:
                timeCollect=int(input("Please enter how long to collect
data per sample: "))
                break
            except:
                print("That is not an integer.")
        while True:
            try:
                numSample=int(input("Please enter how many samples to
collect: "))
                break
            except:
                print("That is not an integer.")
        for x in range(0,numSample):
            arrayLabel.append(input("Please enter the label for sample
#"+str(x+1)+": "))

        titlePrint()
        #Call Functions for loop
        for x in range(0,numSample):
            motorRun(motorRun,motortimeCollect,sPort)
            dataCollect(dataCollect,timeCollect,dataQueue,sPort)

dataPrint(dataPrint,arrayLabel[x],dataQueue,ZeroData,ThreeData,TwoData,Twen
tyData,EightData)
            print("\nData for "+arrayLabel[x]+" has been collected.\n")
            if(x==numSample-1):
                print("Data collection is complete.\n")
                arrayLabel.clear()
```

```
            else:
                print("Waiting for user to collect data for
"+arrayLabel[x+1]+".\n")
                input("Press Enter to proceed.")
    elif(state=="P"):
        #Prediction Mode
        input("Entering prediction mode. Press enter to continue.")
        if(numSample==1):
            print("Singular collection entry detected. Defaulting to
collected data.")
            stateVal=1
        else:
            print("Defaulting to manual entry.")
            stateVal=0

predArray=CreatePredArray(CreatePredArray,stateVal,ZeroData,ThreeData,TwoDa
ta,TwentyData,EightData)
        Pred=gasPredictor(predArray)
        print("The prediction is "+Pred[0])
        ledTrigger(ledTrigger, Pred[1],sPort)
        stateVal=0
    elif(state=="O"):
        print("Change the motor runtime\n")
        print("The default runtime is currently 10 seconds")
        motortimeCollect=int(input("Please enter new value"))
        input("The current motor runtime has been changed. Press Enter to
continue")
    elif(state=="Q"):
        print("Quitting...")
        break
    else:
        print("That is not a valid entry. Please try again: ")

print("GoodBye!")
```

Python Computer Code (Raspberry Pi, Bluetooth)

This code uses a bluetooth connection and is meant to be run on a Raspberry Pi, paired with an Android device. This code was running on the final MVP, to show that the headless computer was able to run this system and report autonomously without requiring any peripherals being plugged into the Raspberry Pi.

To run this, an Android device with “The Blue Dot” application is required. In the future, the Bluetooth integration will be more agnostic.

```
# -*- coding: utf-8 -*-
```

```
"""
Created on Wed Mar  16 2022
Does the same stuff as GasSensorInterface, but boiled down to be controlled
via bluedot
@author: Thomas
"""
#Libraries are in!
import serial;
import time;
import csv;
from queue import Queue
from MLModel import gasPredictor
from GasSensorLib import
titlePrintBT,motorRun,dataCollect,dataPrint,CreatePredArray,ledTrigger
#BlueDot Setup
from bluedot import BlueDot
#Setup Bluedot
bd=BlueDot(cols=5)
#Hide the two buttons for separation
bd[1,0].visible=False
bd[3,0].visible=False
#Change colors
bd[2,0].color="green"
bd[4,0].color="red"

#Serial Port Name
     #Change the name here which will control the name in subsequent
declarations
     #For Thomas' Laptop, it is COM8
     #For the RPi, it is /dev/ttyUSB0
sPort='/dev/ttyUSB0'
#Initialize queues
dataQueue=Queue(maxsize=0)
ZeroData=Queue(maxsize=0)
ThreeData=Queue(maxsize=0)
TwoData=Queue(maxsize=0)
TwentyData=Queue(maxsize=0)
EightData=Queue(maxsize=0)
#Initialize label array
arrayLabel=[]
#State Value Variable
stateVal=0
numSample=0
```

```
killvar=0
#Modify these variables if needed
motortimeCollect=60
timeCollect=30
#Create exception
class Exit(Exception): pass
#Program comprised of functions based on BlueDot functionality
def Collect(pos):
    timeCollect=30
    arrayLabel.append("CollectedData")
    titlePrintBT()
    #Just run once
    motorRun(motorRun,motortimeCollect,sPort)
    dataCollect(dataCollect,timeCollect,dataQueue,sPort)

dataPrint(dataPrint,arrayLabel[0],dataQueue,ZeroData,ThreeData,TwoData,Twen
tyData,EightData)
    print("\nData for "+arrayLabel[0]+" has been collected.\n")
    print("Back at main")
def Predict(pos):
    stateVal=1

predArray=CreatePredArray(CreatePredArray,stateVal,ZeroData,ThreeData,TwoDa
ta,TwentyData,EightData)
    Pred=gasPredictor(predArray)
    print("The prediction is "+Pred[0])
    ledTrigger(ledTrigger,Pred[1],sPort)
    stateVal=0
    print("Back at Main")
def Kill(pos):
    raise Exit
#Start of Program: Ask for state of program:
try:
    while True:
        bd[0,0].when_pressed=Collect
        bd[2,0].when_pressed=Predict
        bd[4,0].when_pressed=Kill
except Exit:
    print("GoodBye!")
```

Appendix B1: Climate Control System Code

```
#include "max6675.h"
#include <dht.h>
dht DHT;

// initialize pins
#define DHT11_PIN 13

int soPin1 = 4;
int csPin1 = 5;
int sckPin1 = 6;

int soPin2 = 7;
int csPin2 = 8;
int sckPin2 = 9;

int soPin3 = 10;
int csPin3 = 11;
int sckPin3 = 12;

int pinOut = 3;

MAX6675 Module1(sckPin1, csPin1, soPin1);

MAX6675 Module2(sckPin2, csPin2, soPin2);

MAX6675 Module3(sckPin3, csPin3, soPin3);

void setup() {
    Serial.begin(9600);
    Serial3.begin(9600);
    pinMode(3,OUTPUT);
}

void loop() {
// read from DHT sensor
DHT.read11(DHT11_PIN);

double temp = DHT.temperature;
double humid = DHT.humidity;
```

```
// printing the temperature and the humidity to the serial monitor
Serial.print("Temperature From DHT: ");
Serial.println(temp);
Serial.print("Humidity: ");
Serial.println(humid);

// read temperature from thermocouple and print it to serial monitor
float temp_from_Therm1 = Module1.readCelsius();
Serial.print("Temperature From Thermocouple #1: ");
Serial.println(temp_from_Therm1);

float temp_from_Therm2 = Module2.readCelsius();
Serial.print("Temperature From Thermocouple #2: ");
Serial.println(temp_from_Therm2);

float temp_from_Therm3 = Module3.readCelsius();
Serial.print("Temperature From Thermocouple #3: ");
Serial.println(temp_from_Therm3);

// sorting the three different temperature from the thermocouples and
setting the median_temp to the median of the three values
if (temp_from_Therm1 > temp_from_Therm2 and temp_from_Therm1 <
temp_from_Therm3){
  float median_temp = temp_from_Therm1
}else if (temp_from_Therm2 > temp_from_Therm1 and temp_from_Therm2 <
temp_from_Therm3){
  float median_temp = temp_from_Therm2
}else {
  float median_temp = temp_from_Therm3
}

// sending the median temperature and the humidity measured via bluetooth
to the user's mobile app

Serial3.print("T:");
Serial3.println(median_temp);
Serial3.print("H:");
Serial3.println(humid);

// send a low signal if the temperature is above 30, turning off the
heater, and send a high signal if the temperature is below 25, turning on
the heater
if (median_temp > 30){
```

```
  digitalWrite(pinOut, LOW);
}else if (median_temp < 25){
  digitalWrite(pinOut, HIGH);
}

delay(2000);

}
```

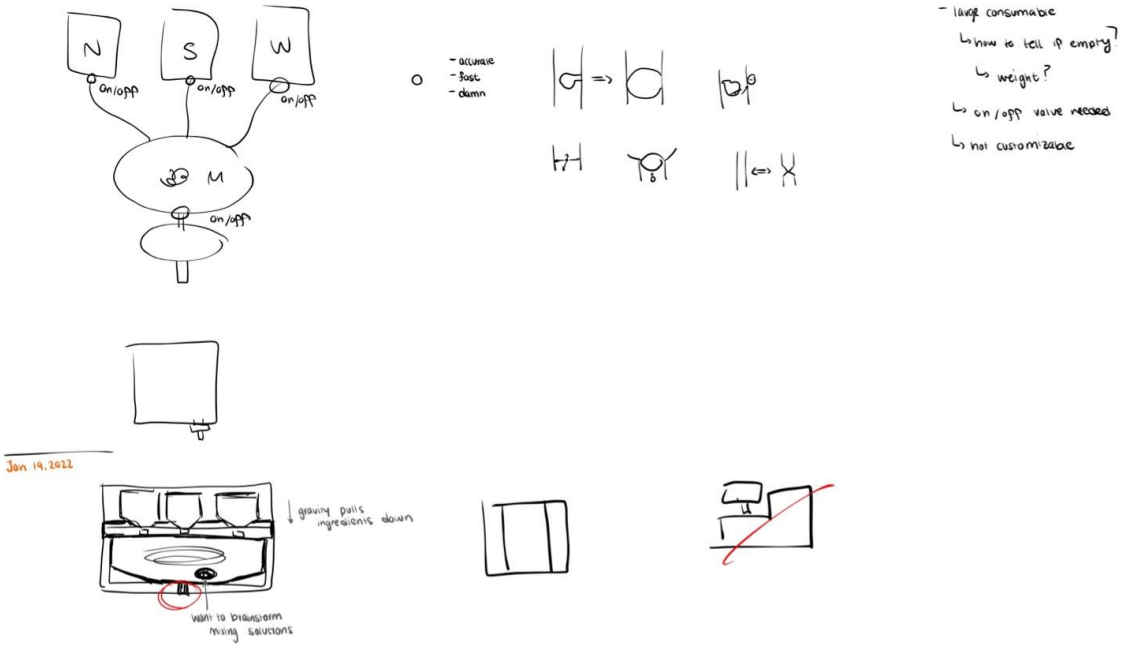
Appendix B2: Mobile app github

Frontend:
https://github.com/thiagolee1999/nectarfy-app

Backend:
https://github.com/leowei31/nectarfy-app-backend

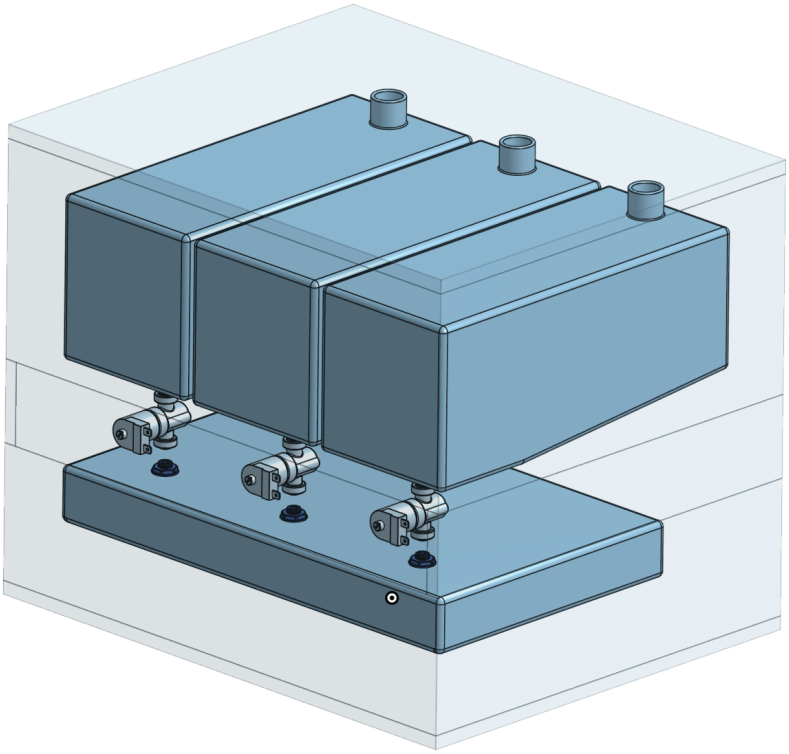
Appendix C: Additional Information for Improved Feeding System

Early Design Concepts

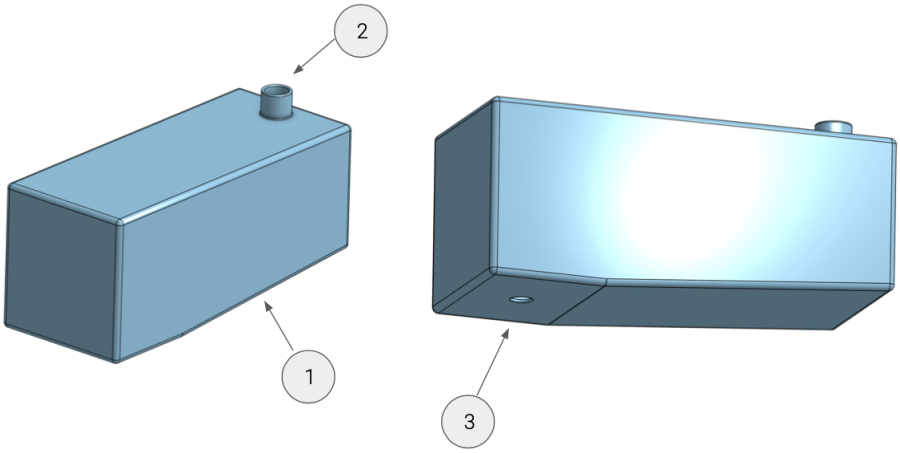


The design for the feeding system began by understanding the most basic requirement that had to be accomplished which was mixing two ingredients. From this requirement, it became clear that a minimum of three tanks would be required in the design: two ingredient tanks and one mixing tank. Next, it became obvious that ingredients would have to flow from the ingredient tanks to the mixing tank which introduced the use of flexible tubing and on/off valves between the ingredient and mixing tanks. Furthermore, it became crucial to ensure isolation of the mixing tank from the feeding area so no fluid would spill into the hive which introduced the requirement of another on/off valve between the mixing tank and the feeding area. Lastly, there had to be a way to ensure flow between all the tanks. However, since I wanted to limit the amount of power required for the system, I designed the system such that the ingredient flow relied on gravity rather. This meant that the ingredient tanks had to be above the mixing tank and the mixing tank had to be above the feeding area.

Reviewing the design, it became clear that it was best to make this components out of a combination of 3D printed parts and wood which led to the first major iteration of the design on OnShape:



In this CAD image, one can see the transparent shape of the enclosure along with the ingredient tanks, solenoids to control the on and off valve, flexible tubing connections and the mixing tank. To understand some design choices, I will go into more detail regarding each component. First, I want to review the design of the ingredient tank:

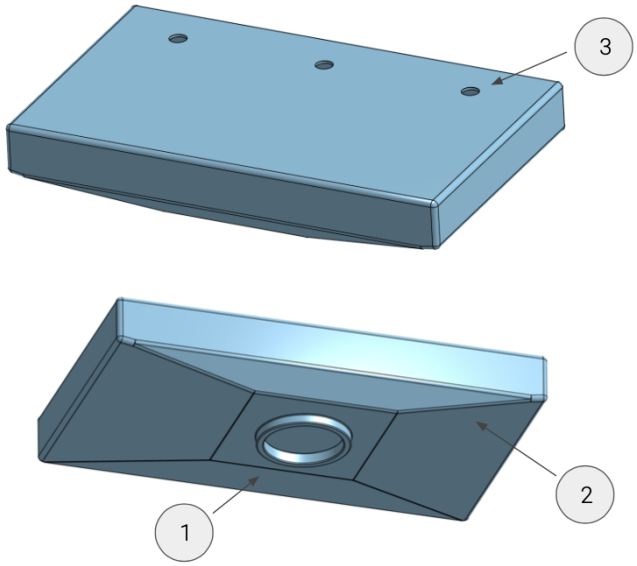


Several key design choices were made in the design of this tank:

1. Angled tank to encourage fluid flow towards outlet
2. Fill port at the top of tank for easier access for the beekeepers to fill

3. Flat face at bottom of tank to allow for easy installation of the push-to-connect fitting for outlet flow

Next, I want to review the design of the mixing tank:



Some important design choices regarding the mixing tank design include:

1. Feeding area connection
2. Angled tank to encourage fluid flow towards the feeding area outlet
3. Flat face at top of tank with three clearance holes for push-to-connect fittings that will connect to the three respective solenoids and ingredient tanks

Solenoid, Fittings, & Voltage Adapter Selection

Reviewing the initial concept of the design, one of the major next steps was to select solenoids that would act as the on and off valve through-out the system. To find the solenoids that were to be used in this system, a search on McMaster was conducted. Three key characteristics were selected to reduce the number of options:

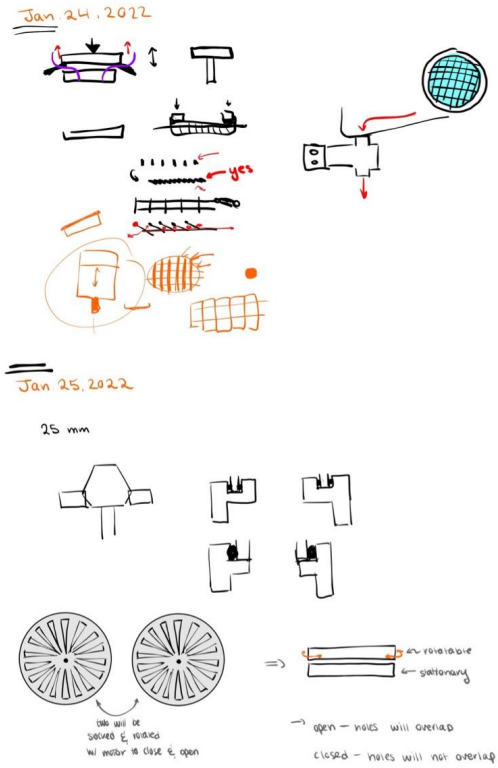
- Actuation: Electric
    - Justification: Wanted to be able to control solenoids remotely using an Arduino rather than manually opening and clothing valves
- Valve Starting Position: Normally Closed
    - Justification: Valves would normally need to be closed to keep ingredients from flowing into mixing tank and it is important to control how long the valves are opened for to dispense proper amount of ingredients
- Connection Style: Push-to-Connect

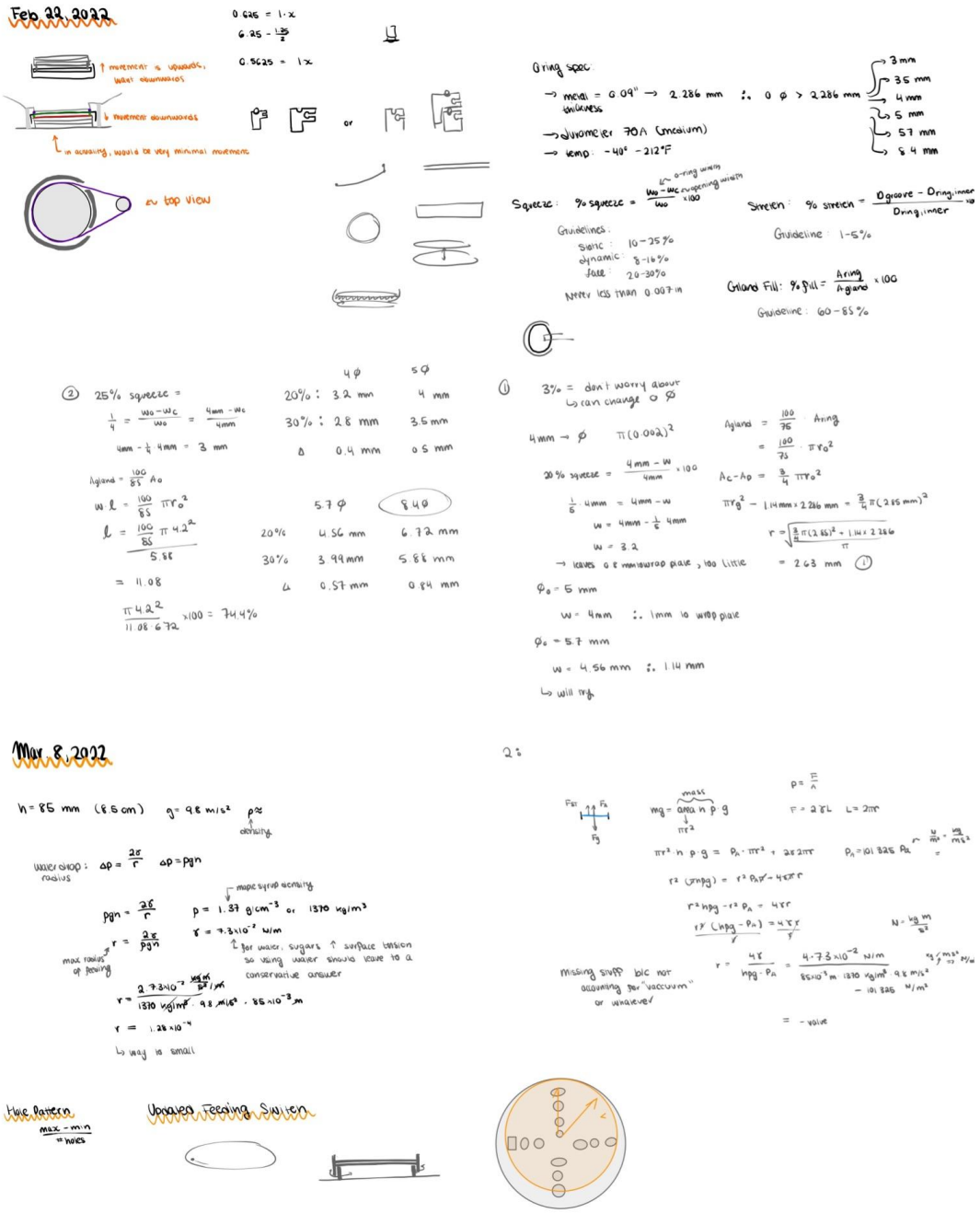
- Justification: Push-to-Connect would be easy to set-up and would allow for the use of flexible tubing which provide more freedom in tolerances in the system

These characteristics quickly limited the options of available solenoids to only one food-safe solenoid model. Comparing the options of this model, the 24 VDC model (part number: 5489T633) was selected based on price and voltage requirements. This solenoid only worked with ¼" tubing which quickly restricted the options of tubing and corresponding push to connect fittings to choose from. After reviewing factors including costs, food-safety, and more, the tubing (part number: 5231K952) and fittings were selected (part number: 9087K48). Since these solenoids required 24 VDC, an affordable but reliable 120 VAC to 24 VDC adapter was found online as there was no other way to provide the solenoids with the required voltage.

Development of Feeding Area On/Off Valve Design

One of the key features of the assembly involves connecting the mixing tank to the rest of the hive through some kind of on/off valve. Since the feeding hole diameter was much larger than the ¼" tubing, a custom on/off valve-like component had to be made. This proved to be a very challenging design problem and as part of the design cycle, many concepts were brainstormed:



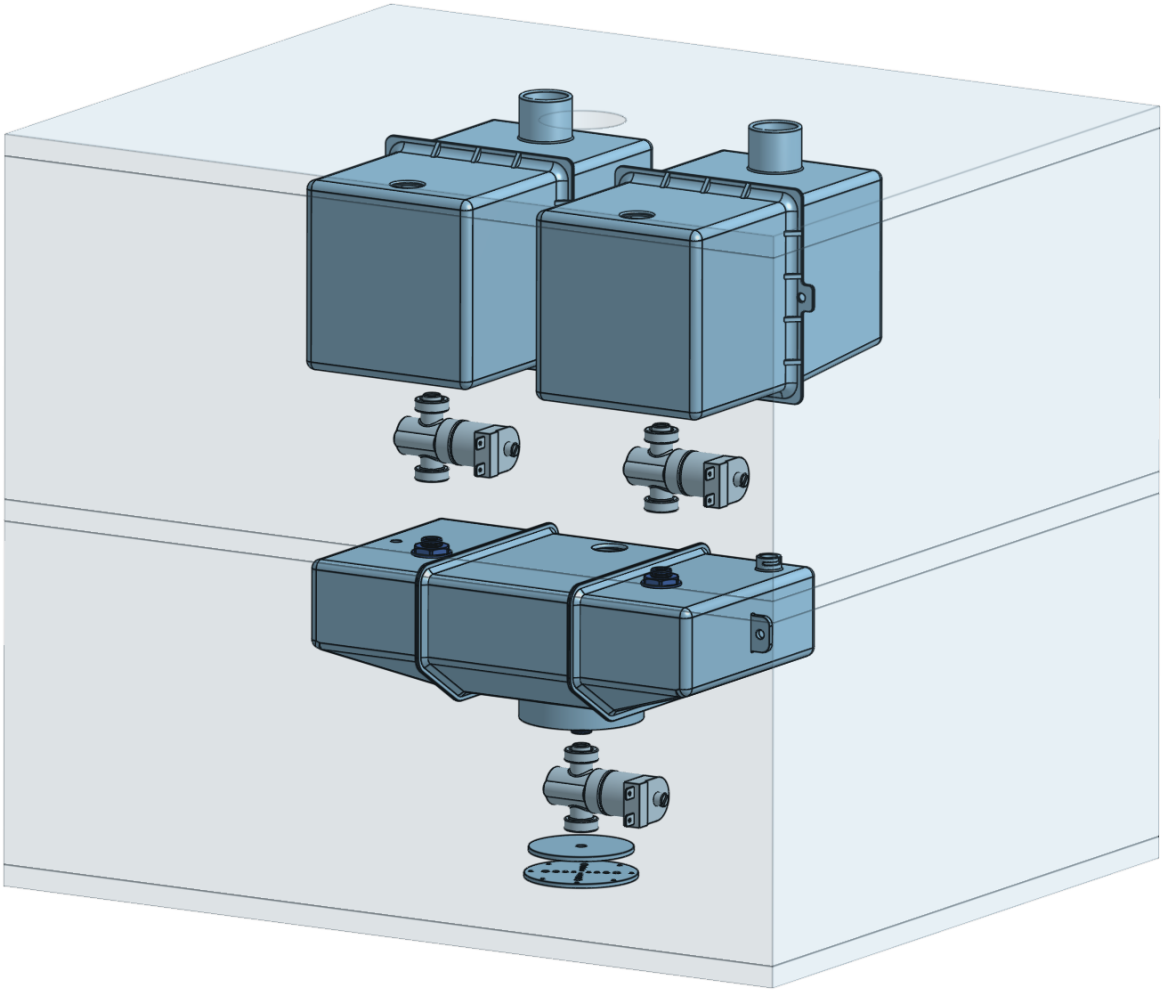


After a design review with the team, a decision was made to move forward with the double-plate rotatory concept (seen at the bottom of the above figure). This concept was chosen because it took advantage of the rotational motion created by a motor (rather than translating it to linear motion), was simpler to manufacture compared to other concepts, and minimized the complexity and moving parts in the system. The CAD for this feeding mechanism can be seen here:

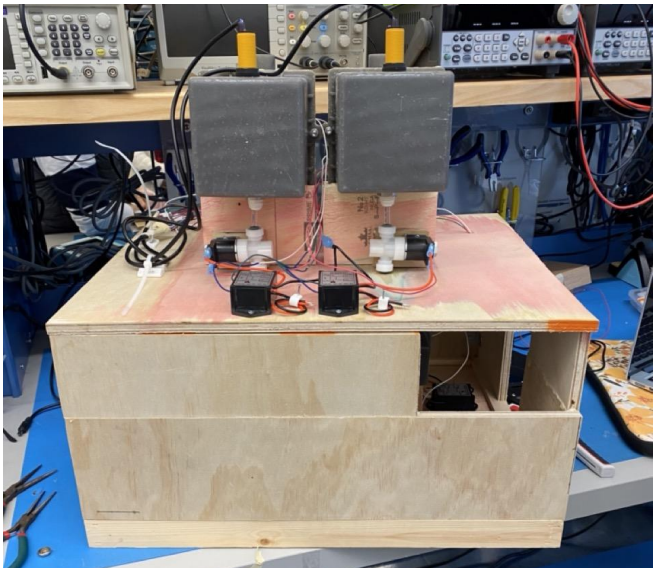
Unfortunately, this design did not end up working when tested as liquid was able to leak through the plates due to imperfections in the plates and assembly making it near impossible to produce a perfectly sealed surface. To solve this issue, another part was designed that had a through hole for a push-to-connect fitting so that a solenoid could be used between the feeding area and mixing tank.

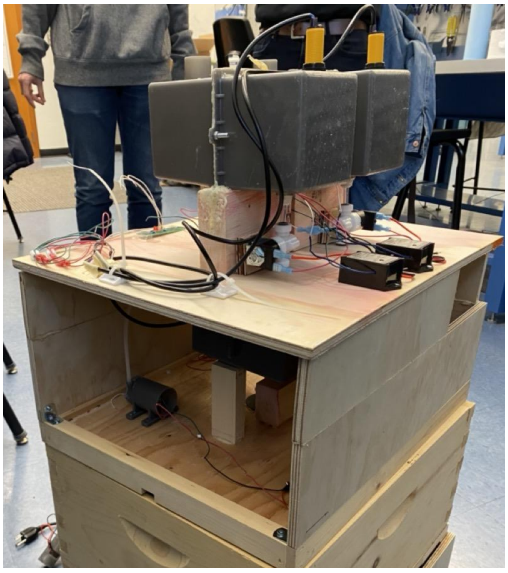
Extended Overall System Design

The current design of the improved feeding system is as follows:



Not pictured in this image includes the mounts, electronics, and air pump design which can be seen in images of the real, current prototype:

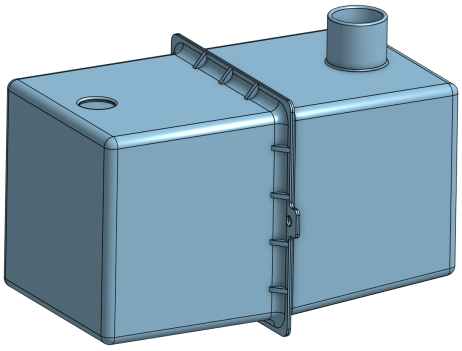




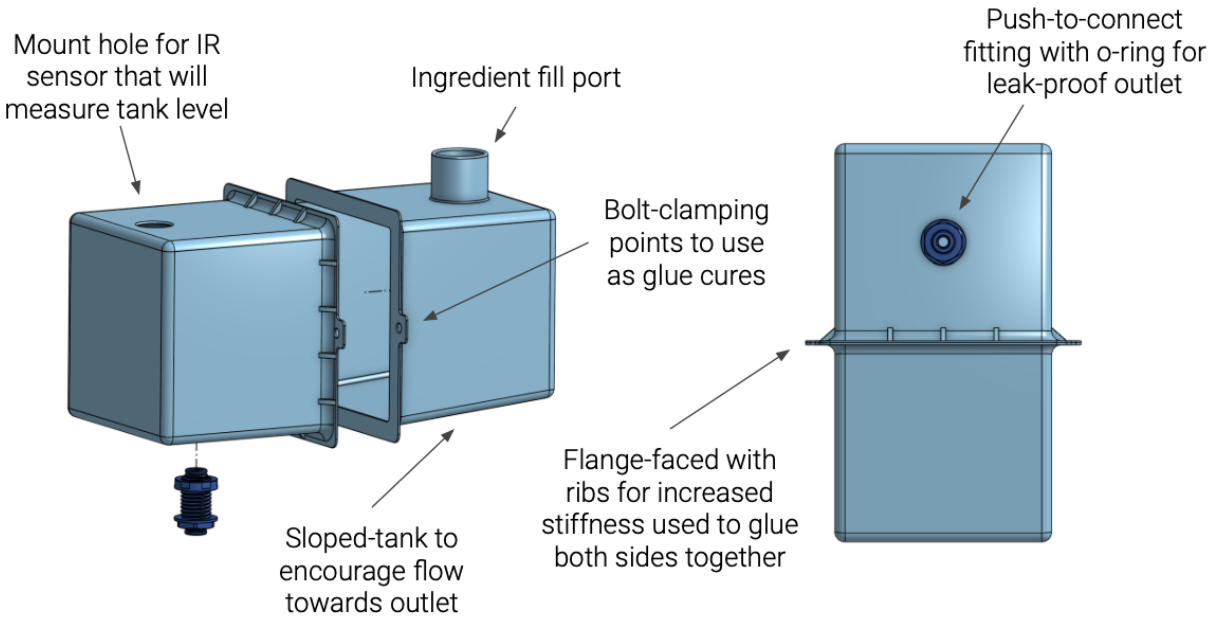
In general, the idea of the design was to have two ingredient tanks (one for sugar and one for water) from which ingredients in either tank can flow into the mixing tank through a solenoid on/off valve and gravity. Once in the mixing tank, an air pump mixes the two ingredients together by pumping air into the tank at a high rate to create turbulence. Then, a solenoid on/off valve connecting the mixing tank to the feeding area will periodically turn on and off to refill the feeding area with simple syrup for the bees to consume to ensure the liquid inside of the feeding area is able to be held in place through surface tension. To ensure that all the tanks are not empty when the system runs, IR distance sensors are used to determine when the tanks are empty. If they are empty, the LED corresponding to the empty tank is turned on to notify the user to refill the tank either by adding the ingredients to the ingredient tanks or by starting another mixing cycle for the mixing tank.

To fully understand the design, all the design and justification of all key components will be briefly discussed.

Extended Ingredient Tank Design

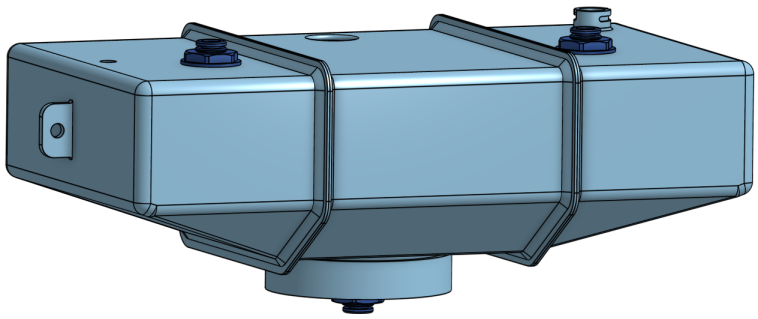


While a simple premise, there were several key design choices made when creating the ingredient tank:

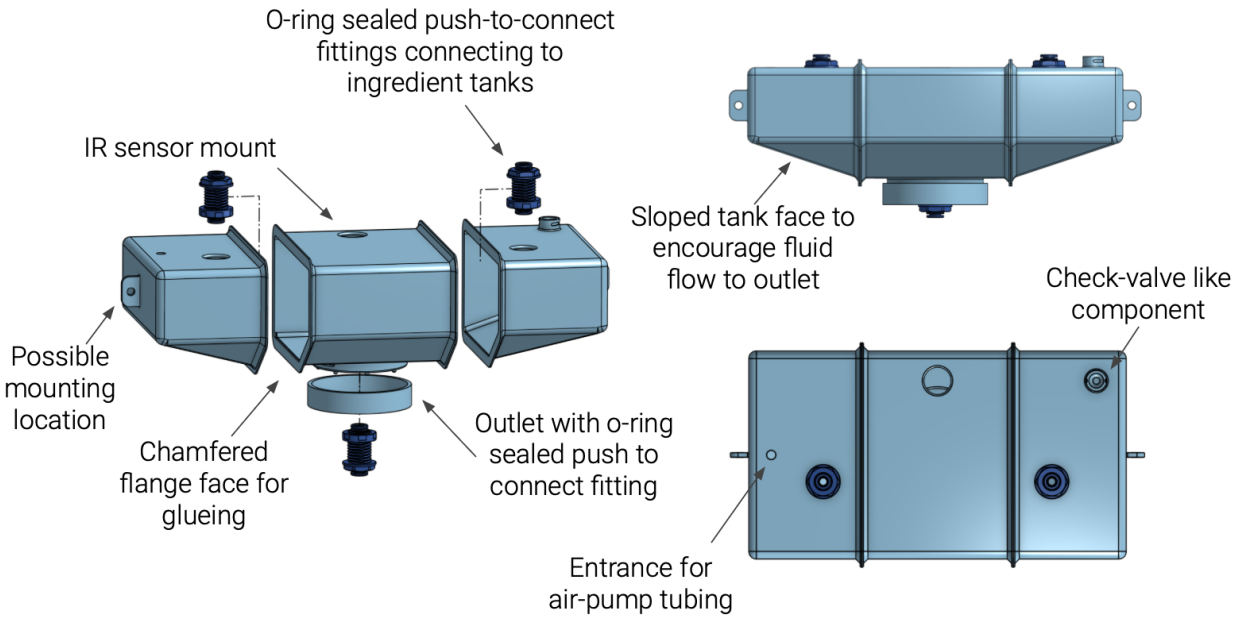


The tank itself is made out of two parts so that it would be easier to 3D print and because it would allow for installation of both the push-to-connect fitting and the the IR sensor. The push-to-connect fitting also as an o-ring seal to stop any leaking from the outlet. The tank includes a large inlet at the top to allow for the user to easily fill it with an ingredient. On one of the sides, there is also a small slope downwards to encourage the motion of the ingredient towards the outlet. The parts are joinged along the flange faces using glue as it is a simple method to connect to parts and it would ensure that the assembly would not leak water. Holes are provided along the flange face to help clamp the assembly while the glue is curing. Lastly, there are ribs integrated along the flange to add stiffness to the parts to ensure the flange face remains flat and doesn't warp as its printed.

Extended Mixing Tank Design

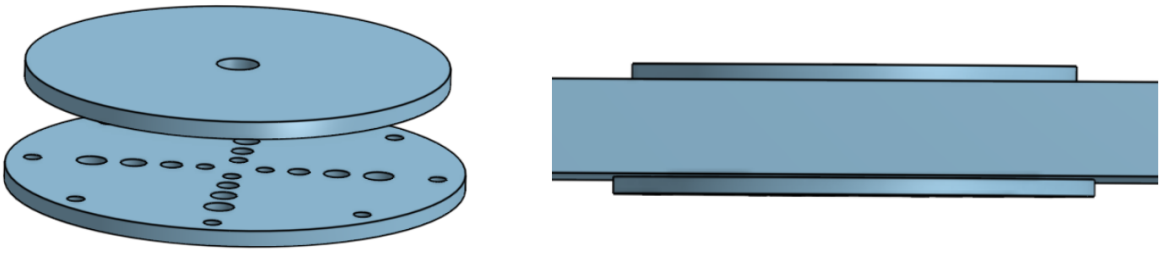


In general the design is very similar to the ingredient tank:

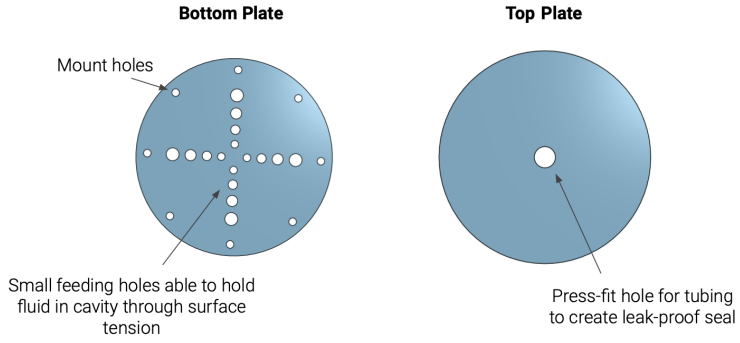


Instead of two parts, the mixing tank is made out of three parts to allow for easier 3D printing and installation of fittings, sensors, and air pump. All three push-to-connect fittings are installed with o-rings to ensure they are leak proof. The tank also includes a sloped base to encourage fluid flow towards the outlet. Each part also has a flanged face as a glueing surface, but instead of having ribs to increase thickness, there is a chamfer between the tank sides and the flange face. In the mixing tank, there is also a hole where tubing from the air pump can enter which is then glued to the side of tank such that when the air pump runs, the air entering the system, enters at the bottom of the tank. A check-valve like feature is also included in the design to ensure any air entering the tank is able to leave so that it never becomes pressurized. However, when there is no air being pumped into the tank, it acts like a seal so that when the feeding area is being filled with water, no air is entering the system and the fluid is able to stay in place by surface tension. Additionally, there are possible mounting locations on either side to help with part installation in enclosure.

Extended Feeding Area Plates



The feeding area is comprised of two plates encasing a through-hole on the bottom board of the system. The design for both plates is fairly simple:

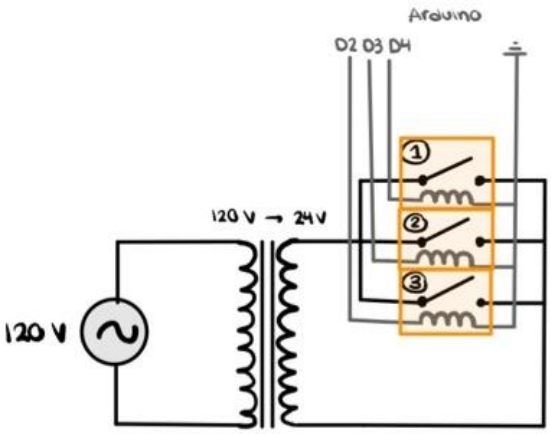


The top plate has a single press-fit hole the tubing from the solenoid enters. By making it a press fit, the material of the tube is able to push along the circumference of the hole to create a seal. It is attached to the bottom board via glue. The bottom plate has a pattern consisting of several small holes. The diameters of these holes were selected to ensure that surface tension would be able to hold the fluid in the feeding area and to maximize the amount of area for the bees to access the simple syrup from. Along the diameter, the bottom plate has another hole pattern that can be used for mounting holes to the board, however, to ensure no leaks, the bottom plate was also glued to the bottom board.

Extended Solenoids Design

The solenoids used through out the system have part number 5489T633 and can be found on McMaster (MM). This solenoids were selected based on their ability to actuate electronically, push-to-connect compatibility, normally-closed valve position, power requirements, food-safe status, and cost. This solenoid selection also limited the selection of tubing (MM part number: 5231K952), push-to-connect fittings (MM part number: 9087K48), and voltage adapter. Since the solenoids ran on 24 VDC, a relay was also required so that the solenoids could be controlled using a 5V Arduino. Below is an image of the solenoids and relays:

The circuit diagram for this system is as follows:



The Arduino code used control the solenoids is as follows:

```
const int middleSolenoidPin = 2;
const int leftSolenoidPin = 3;
const int rightSolenoidPin = 4;

int mix = 0; //can be 1 for blue, 2 for red, 3 for purple

void setup() {
  pinMode(middleSolenoidPin, OUTPUT);
  pinMode(leftSolenoidPin, OUTPUT);
  pinMode(rightSolenoidPin, OUTPUT);
  Serial.begin(9600);
}

void loop() {
    Serial.println("Please enter 1 for blue, 2 for red, 3 for purple:"); //Prompt
user for input

    while (Serial.available() == 0) {
      // Wait for User to Input Data
    }

    mix = Serial.parseInt();

    if (mix == '1') {
      //flowing blue
      digitalWrite(leftSolenoidPin, HIGH);
      delay(5000);

      digitalWrite(leftSolenoidPin, LOW);
      delay(1000);

      //showing blue
      digitalWrite(middleSolenoidPin, HIGH);
      delay(6000);

      digitalWrite(middleSolenoidPin, LOW);
      delay(1000);

    } else if (mix == '2') {
      //flowing red
      digitalWrite(rightSolenoidPin, HIGH);
      delay(5000);

      digitalWrite(rightSolenoidPin, LOW);
      delay(1000);
```

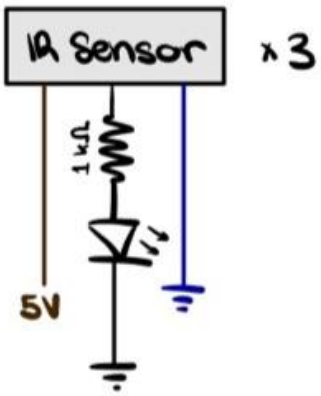
```
      //showing red
      digitalWrite(middleSolenoidPin, HIGH);
      delay(6000);

      digitalWrite(middleSolenoidPin, LOW);
      delay(1000);

    } else if (mix == '3') {
      //flowing red
      digitalWrite(rightSolenoidPin, HIGH);
      delay(2500);

      digitalWrite(rightSolenoidPin, LOW);
      delay(1000);

      //flowing red
      digitalWrite(leftSolenoidPin, HIGH);
      delay(2500);

      digitalWrite(leftSolenoidPin, LOW);
      delay(1000);

      //delay to manually run motor with button
      delay(6000);

      //showing purple
      digitalWrite(middleSolenoidPin, HIGH);
      delay(6000);

      digitalWrite(middleSolenoidPin, LOW);
      delay(1000);

    } else {
      Serial.println("Not an option, please select 1, 2, or 3.");
    }
  }
}
```

Extended Tank Level Sensor Design

In the system, IR distance sensors were installed on every tank so that it could be used to monitor and alert the user when the tanks are empty. Below is an actual image of them in the system:

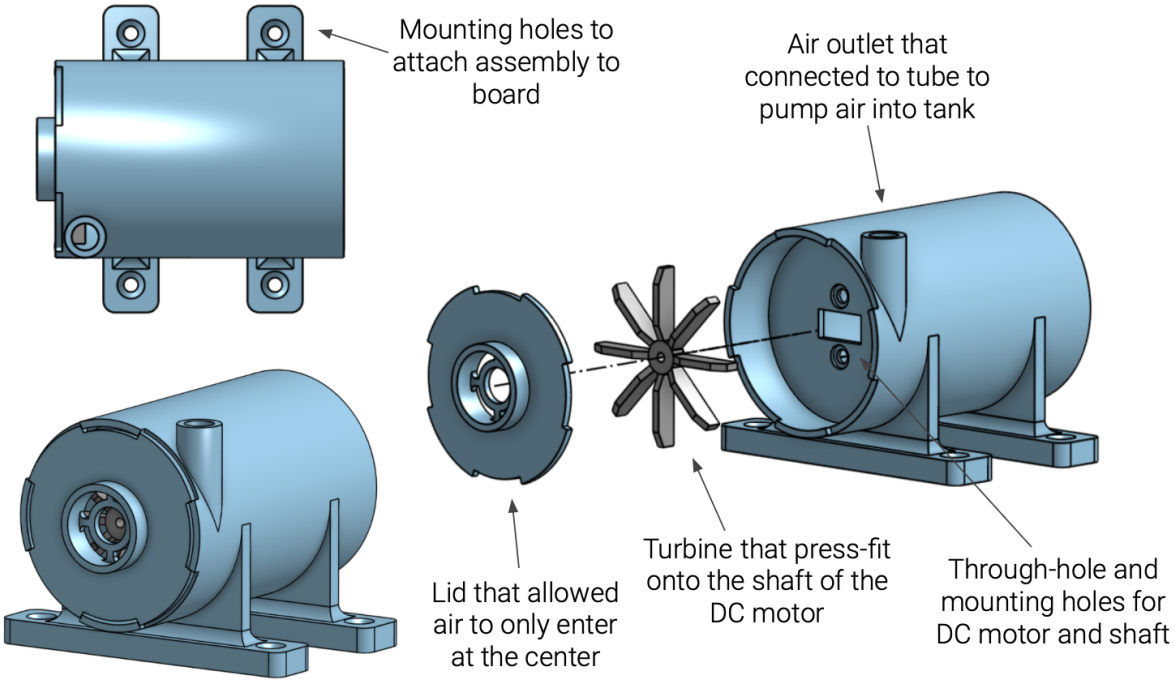
For this current version of the prototype, each sensor simply connects to an LED that turns on when the tank is empty:

The circuit diagram for this set-up is as follows:

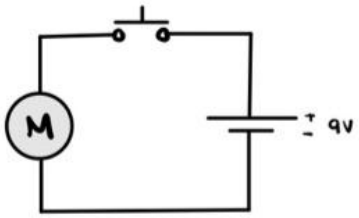


Extended Air Pump Design

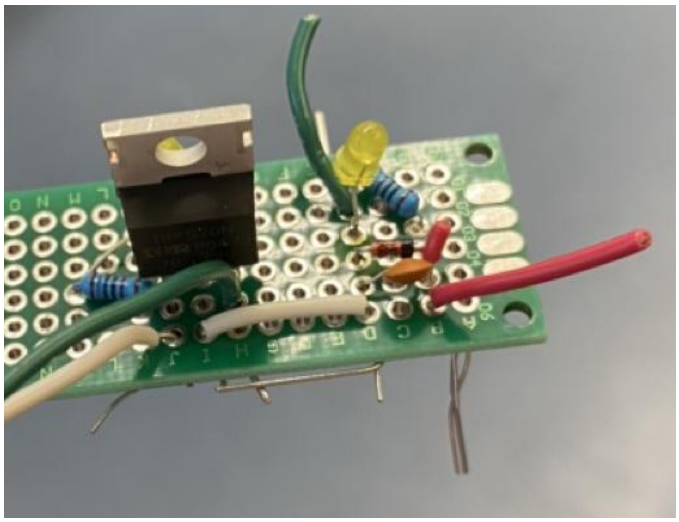
The air pump used in this system had to be custom designed in order to mix the simple syrup mixture at a fast rate. The design consisted of housing a DC motor in an enclosure with a separate area that the shaft passed through to spin a turbine which sucked in air through the center and pushed air out of the side outlet:



Due to damages from significant vibrations when transporting the prototype, the DC motor rotar seems to have become misaligned causing it to now draw excess current. This caused the destruction of a the motor driving circuit that was originally being used to control the motor. To overcome this challenge, the motor is currently being run using a battery and button circuit (circuit diagram below) until a new motor arises and the unit can be replaced:



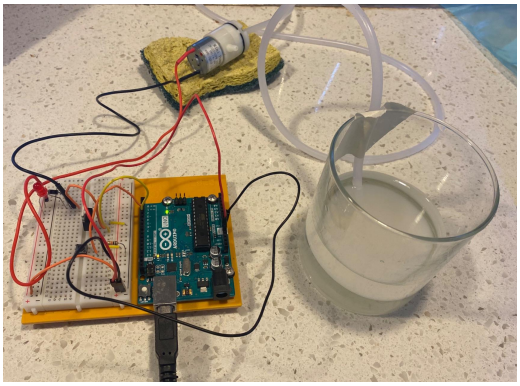
The circuit diagram of the motor driving circuit can is the same as the one referenced in the varroa mite section. Images of the completed protoboard are below:



Verification Details

Mixing Sugar & Water With an Air Pump

Purpose: To determine whether an air pump is capable of mixing sugar and water together
Set-Up: A motor driving circuit is built in an Arduino breadboard:



The motor is run on a 50% duty cycle using the following code:

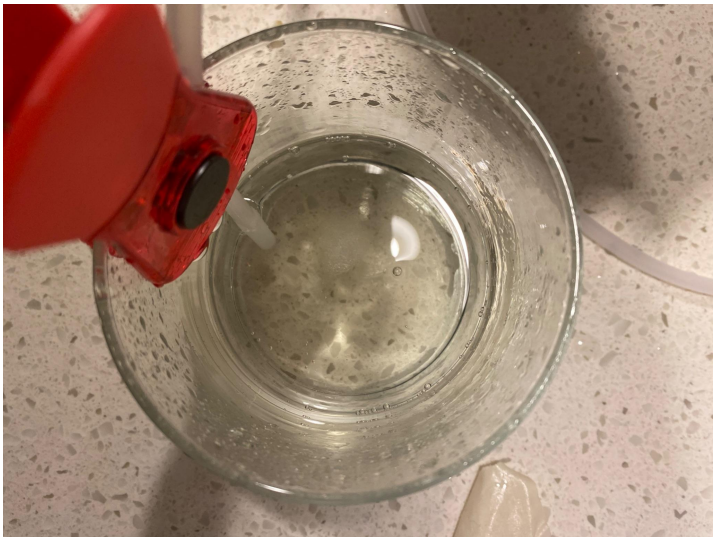
```
const int pumpPin = 2;      //pump connected to digital pin 2

void setup() {
  pinMode(pumpPin, OUTPUT);
}

void loop() {
  digitalWrite(pumpPin, HIGH);
  delay(2000); //delaying 5 seconds
  digitalWrite(pumpPin, LOW);
  delay(2000);
}
```

Execution: The pump ran for a total of 6 hours on a 50% duty cycle. Images were taken every 15 minutes but only key ones are shown below:

Beginning of Test (Time = 0)

2 Hours Into the Test (Time = 2 Hrs)

4 Hours Into the Test (Time = 4 Hrs)



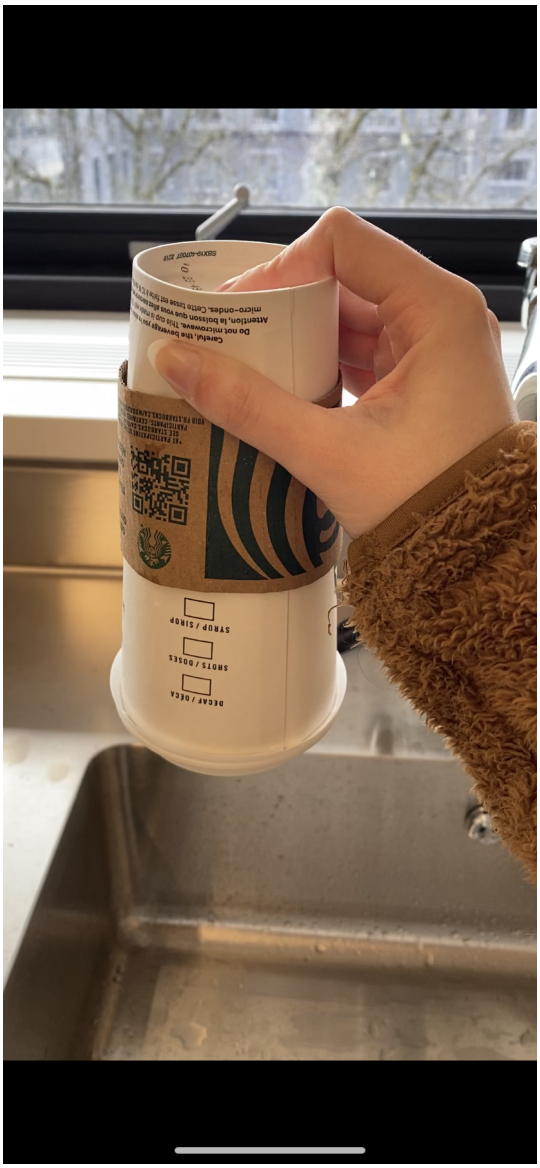
End of the Test (Time = 6 Hrs)

Results: From the images, it is clear that an air pump is able to mix sugar and water together. However, it is clear from this test a stronger pump will be needed to mix the solution together faster and to mix larger batches.

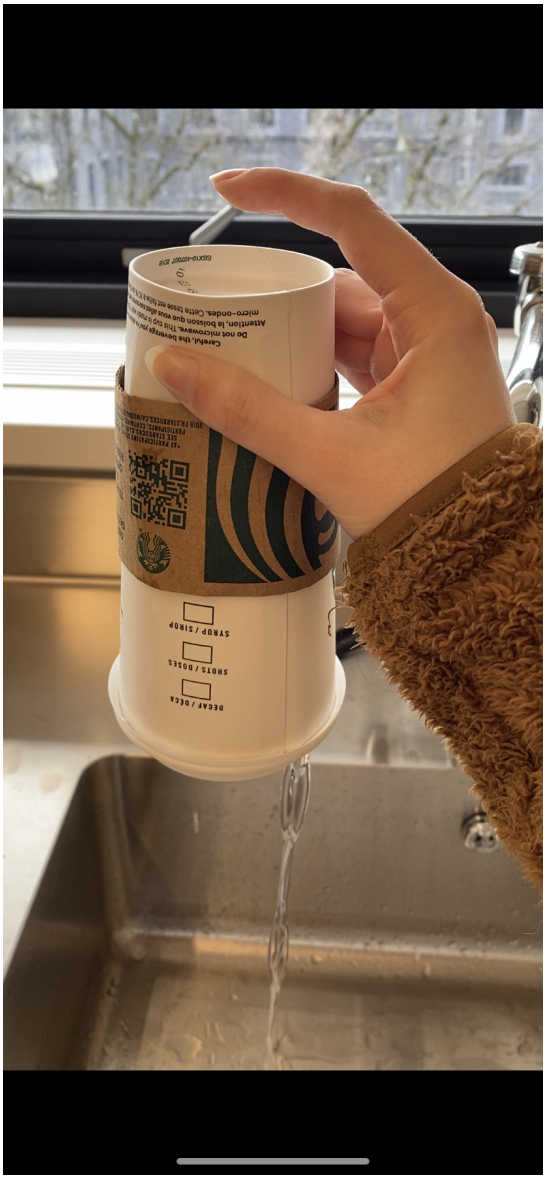
Coffee Cup Surface Tension Test

Purpose: To determine whether a 15 mm hole is able to hold liquid in a water-tight container.
Set-Up: A hole is made at the bottom of a coffe cup that has an opening on the lid where one normally drinks from. This hole is approximately 15 mm wide which would be the maximum diameter of the holes in the feeding platform if no water leaks when the system is closed from atmospheric pressure.
Execution: Hole is covered and uncovered to see if water leaks from drinking hole:

Hole Covered - Container Closed



Hole Not Covered - Container Open



Results: From these two images, it is clear that surface tension and internal pressure are able to keep fluids inside a compartment if the opening is 15 mm or smaller.

Appendix C1: Detailed Design for Varroa Mite System

The current requirements for the varroa mite detection system include:

- Distinguish between different gaseous compositions.
- Able to report to the user when a recognized change is detected.
- Can automatically sample air data
- Can self-cleanse detection chamber

First, the overall design methodology for the subsystems on the MVP are outlined. Then after the design is articulated, the requirements will be reviewed and proven that they are met.

Sensor Selection and Instrumentation

The Arduino natively has a OTS(off the shelf) gas sensor which can be purchased, the MQ-2 gas sensor[A]. This sensor is sensitive to LPG, butane, propane, methane, alcohol, hydrogen, and smoke. It's typically used to detect leakages. However, this is the only sensor available native for Arduino, which would not be sufficient or robust enough for our needs to detect a variety of changes for the detection system.

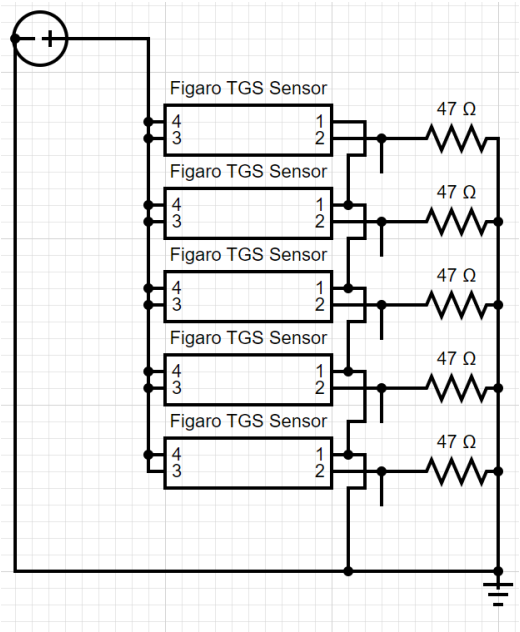
Eventually, a search for gas sensors, inspired by existing research[B] landed upon the TGS gas sensor series by Figaro Sensors[C]. These gas sensors are highly sensitive, robust, and have varying degrees of detection ranges, including but not limited to alcohols, butane, VOCs, refrigerants, and COs.

The gas sensors used in our system from Figaro Sensors were:

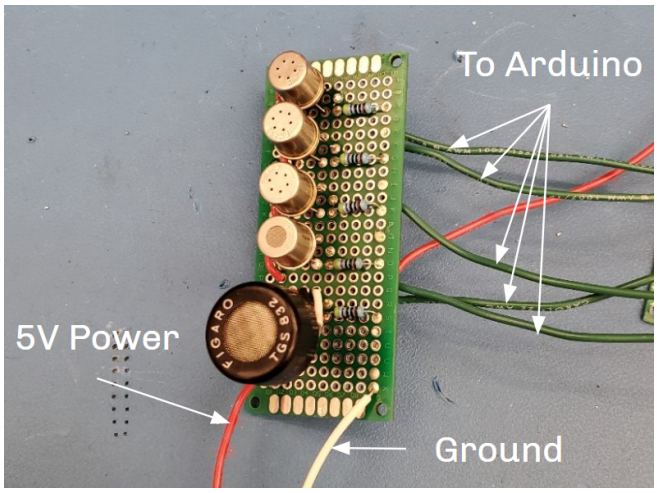
- TGS832-A00
    - Tin dioxide semiconductor, high sensitivity to Chlorofluorocarbons.
- TGS2600-B00
    - Metal-oxide semiconductor, high sensitivity to air contaminants such as hydrogen and CO.
- TGS2602-B00
    - Metal-oxide semiconductor, high sensitivity to ammonia, H2S, VOCs.
- TGS2603-B00
    - Metal-oxide semiconductor, high sensitivity to amine-series and sulfurous odors.
- TGS2620-C00
    - Metal-oxide semiconductor, high sensitivity to vapors of organic solvents and other volatile vapors.

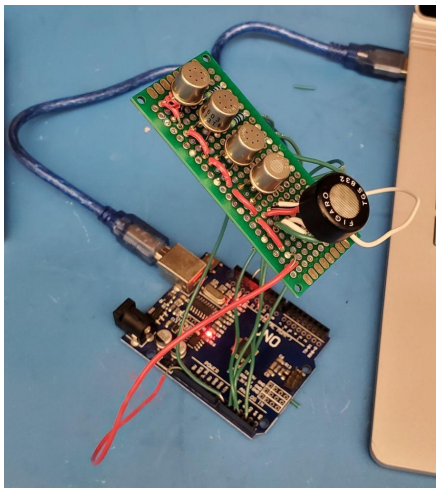
This selection of sensors made were based on the existing research in the area of collecting gaseous data from beehives. With a proven precedence for efficacy, we aimed to use sensors with a similar detection range.

Each sensor varies its resistivity value when detecting different types of gas. Based on data sheets supplied by Figaro, the sensors were built into the following circuit:



Where each disconnected pin before the resistor is a wire going to the Arduino.



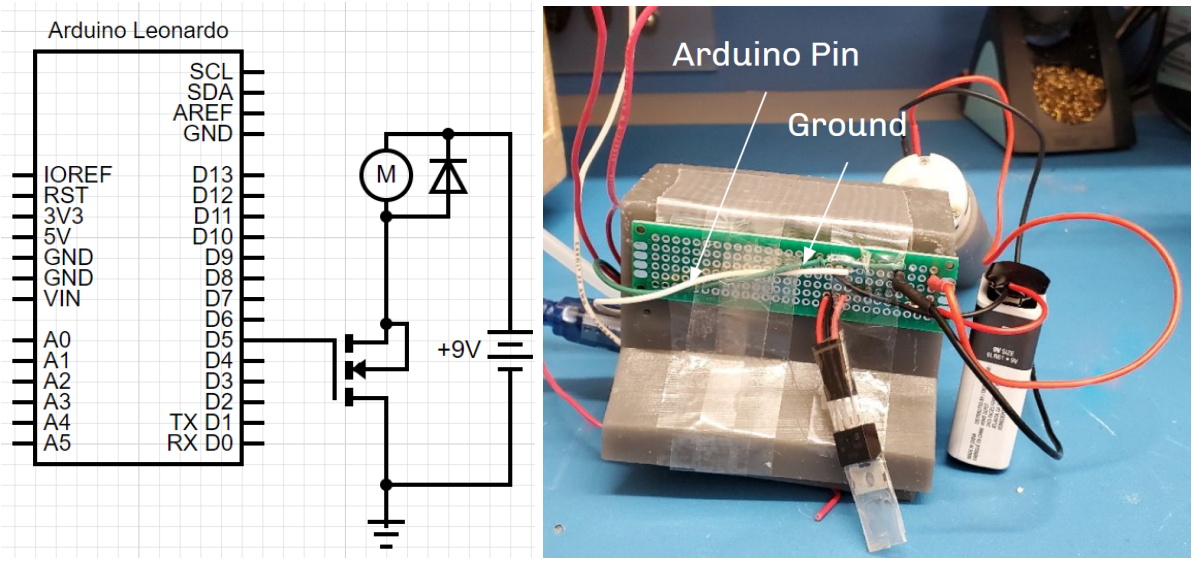


The resistors used were 47 Ohms, as per the datasheet, and each output pin was read by the Arduino by using the Analog In pins.

Motor Selection and Instrumentation

A DC motor and tubing is used to pull gas from the beehive into a semi-isolated chamber where the gas sensors are able to process and read the data. The DC motor was chosen simply based on availability and prior known compatibility with the Arduino.[1]

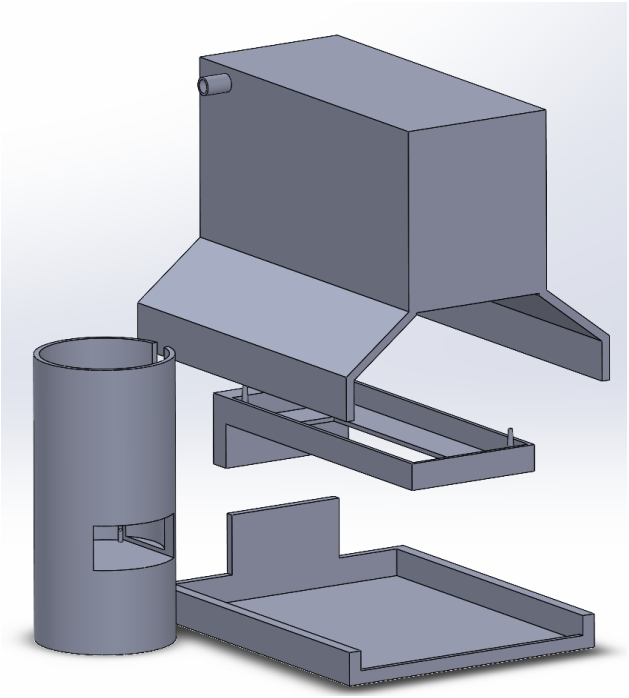
To control and ensure safe operation of the motor, a MOSFET and driving circuit was used. A motor controller chip could also be easily used in place of this circuit, but due to time and budget constraints, the system was built using discrete components.

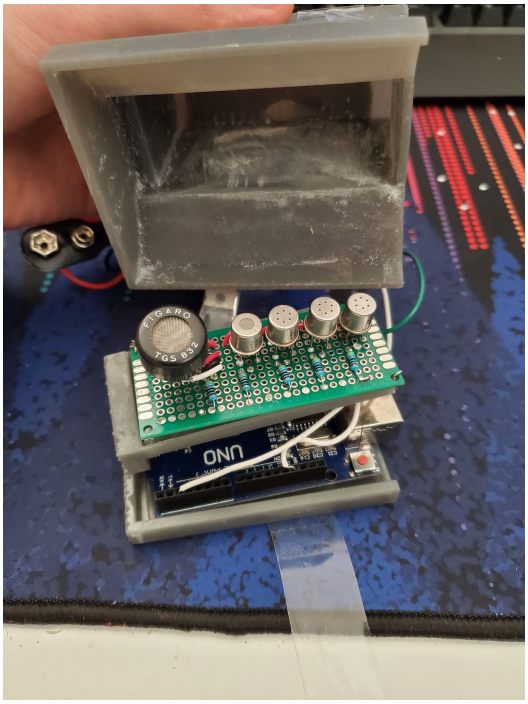


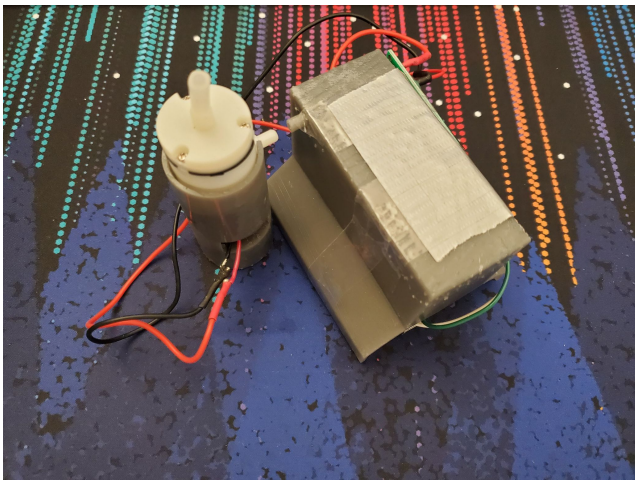
A 9V battery was used to be the power supply for the motor, and the Arduino is plugged in to control the MOSFET, therefore being able to turn on and off the motor as required.
The pump would have tubing which would probe into the beehive box, and be pumped into a small chamber where the gas sensors would be to minimize noise. It would run for a user-determined amount of time, based on the Arduino control.

Sensor Housing CAD Design

The sensor housing was designed to house the sensors and Arduino in an semi-isolated space to provide greater robustness to the sensor reading data. The package was aimed to be created such that all parts could be easily assembled and contained for ease of use and maintenance. While the CAD was designed with these integrations in mind, due to great uncertainty in the function of the rapid 3D printer used, the final hardware faced implementation issues. Moreover, there was a great change in the electronic implementation, and due to budget and time constraints some required modifications could not be completed.
The general intent for the CAD assembly below was to fit within the beehive and ensure compatibility of all of the system parts. It should be noted that the Raspberry Pi is not designed into this integration, as the sensor array simply needs a USB connection to a computer for operation.



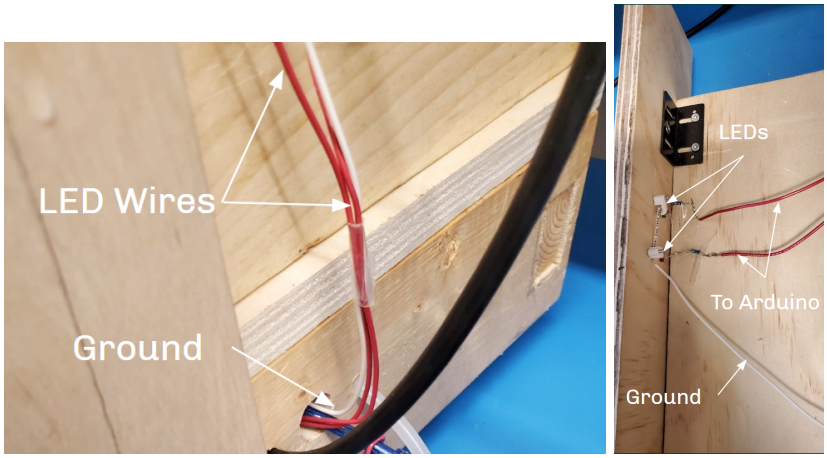




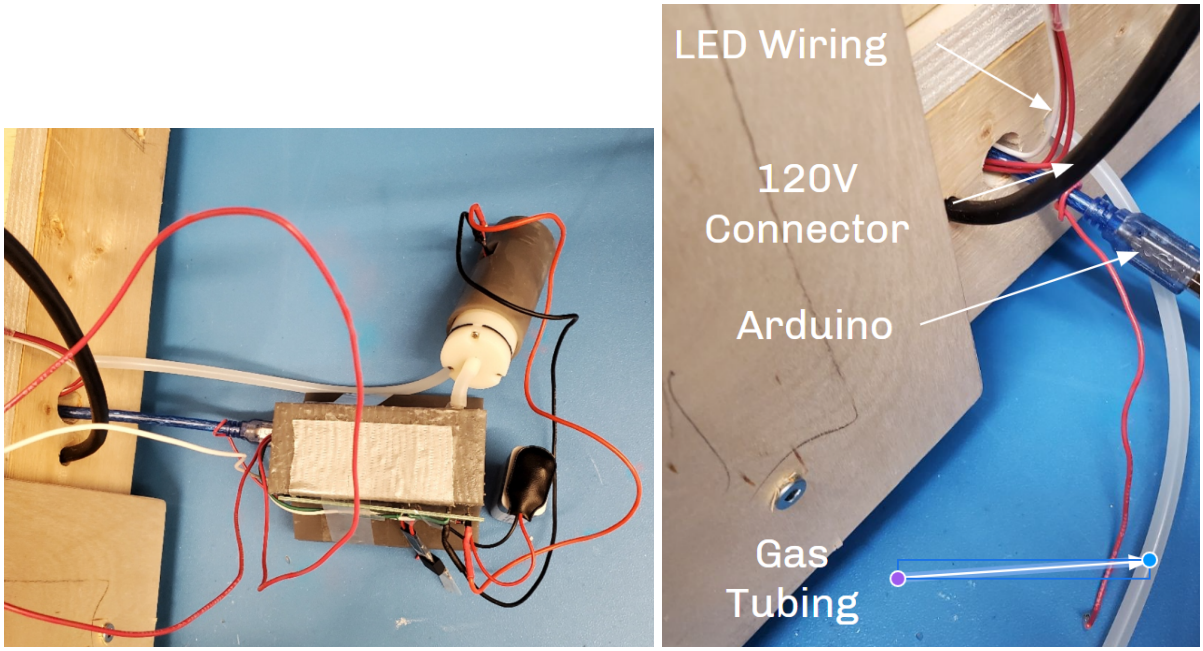
Mechanical Beehive Integration

As the beehive has an easy-access electronics shelf, the original intent of the design was to hold the entire sensor system and the computer. However, due to time and budget constraints, the sensory system was unable to fit within the beehive, but acts as a discrete part of the beehive. However, this allowed for greater demonstration purposes for demos on the MVP.

The Raspberry Pi computer was successfully able to fit within the beehive, as well as the indicator LEDs which show the status of the beehive to the user. These indicator LEDs were simply two LEDs attached to long wires which are connected to Digital Output Pins to the Arduino.



The tubing and wiring both within the beehive to the sensor array, to the Raspberry Pi, to the indicator LEDs were done through drilling holes and cable management.



Careful consideration was made to ensure there was no interference with the other electrical systems within the “easy-access” shelf.

Software

The Final Code written for the varroa mite detection system can be found in Appendix B. Any reference to the final code is made to Appendix B.

The Arduino will power the sensors as well as read data from such sensors. In order to collect the data from the Arduino, the built-in serial port will be used to send the collected data via wire to a computer, which will have a program to read, store, and interpret the incoming data. The Analog In pins are where the sensor readings are read. The Arduino is also controlled via the serial port, where it listens for various ASCII characters to be sent by the controlling computer, and acts upon those messages. For example, if the Arduino reads a character “H” on the serial port from the

controlling computer, it will run data collection and sends the values back to the computer, until it reads a character “L”, upon which it will stop writing to the serial port, and the comm port is closed.
The Arduino is also supplied with code to run the motor which supplies the sensors with gas, as well as the indicator LEDs built into the beehive, which run on the same “listen for character” logic as the code above.

On the side of the computer, there was a little bit more involvement.
There are essentially four different scripts. They are listed below:

5. Gas Sensor Library
    a. A library holding functions which operate various aspects of the gas system.
6. ML (Machine Learning) Model
    a. A machine learning script which builds a ML model which can also predict.
7. Gas Sensor Interface
    a. A wired computer interface for controller the gas sensor array for data collection, debugging, and improving robustness.
8. Gas Sensor Interface BT (Bluetooth)
    a. A script which piggybacks off of scripts 1 and 3, for headless user operation of the gas system.

First, the Gas Sensor Library was opted for because there were two different sets of code which would use the same functions, thus it was easier to create a library. The following functions were created:

def avgQueue(queue):
- Takes a queue of int() type and returns the average value.

def CreatePredArray(self, stateval,ZeroData,ThreeData,TwoData,TwentyData,EightData):
- Creates an array to use for prediction purposes from user-input data.

def dataCollect(self,timeCollect,dataQ,sPort):
- Calls the Arduino to read and write data for a specified amount of time

def titlePrint():
- Prints the title and initializes the .csv file for data.

def titlePrintBT():
- Prints the title and initializes a csv for data while running wirelessly.

def dataPrint(self,label,dataQueue,ZeroData,ThreeData,TwoData,TwentyData,EightData):
- Prints the data into the .csv created by “titlePrint”/”titlePrintBT”.

def motorRun(self,motortime,sPort):
- Calls the Arduino to run the motor for a specified amount of time.

def ledTrigger(self,ledval,sPort):
- Calls the Arduino to activate an LED depending on the predictions made.

The full library can be found in Appendix B.

Secondly, the ML Model script was created, and a function from the Machine learning model was used in Gas Sensor Interfaces scripts which could run predictions based on collected data.
For this portion, almost hours of data was collected from the Gas Sensor Interface script, and different algorithmic models were used to create a predictor.

- ANN(Artificial Neural Network)
- KNN(K-Nearest Neighbors)
- Naive Bayes Regression

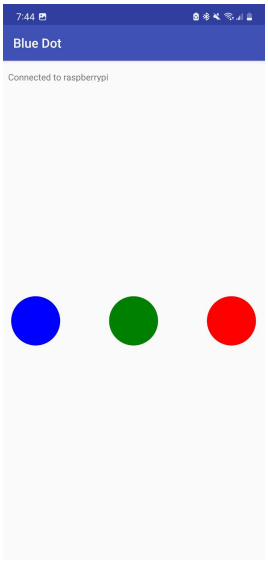
These three different methods were used to find the best machine learning model. The ANN was first created as an experiment, but because the model was meant to classify, it did not perform very well.
Secondly, the Naive Bayes was attempted as it was more suitable for a classification model, but for some reason its accuracy was quite wanting, perhaps due to the lack of noise within the collected data.
Finally, KNN was used, which was the most suitable approach to the system. Given the data which was collected and its discreteness, with a significant lack of noise, the KNN was able to give great degrees of accuracy on the data collected.
A function was created for the Gas Sensor Interface scripts to call for prediction purposes:
def gasPredictor(array):

- Takes an array of sensor data, then predicts what the gas is based on a previously-trained ML model.

The full code for the ML model can be found in Appendix B.

Thirdly, the Gas Sensor Interface script was created. This code is the primary way for the user to interact with the sensor array system, and is very friendly for the user. It functions on user-inputs and allows for the user to control how much data they collect, for how long, and create .csvs. Moreover, it allows for testing of the system by accepting either user-input values, or takes the average of its collected data to run a prediction on.
Quite often, this code was used first to create, debug, and test the sensor system for robustness of creating new code, before writing the final controller code (Gas Sensor Interface BT).
The full code for the Gas Sensor Interface script can be found in Appendix B.

Finally, the Gas Sensor Interface BT script is a lightweight version of the Gas Sensor Interface, but instead of accepting user text-based input, it is controlled by an Android application connected to the Raspberry Pi via Bluetooth. On this application, there are three buttons, one which tells the sensor array to collect data based on preset values(blue) , and another which runs a prediction and notifies the user of the change in gas composition(green). This script allows the entire system to run wirelessly, and is the end goal for user interaction with the device, acting semi-autonomously.



The full code for the Gas Sensor Interface BT can be found in Appendix B.

Meeting Requirements

Given the overview of how the varroa mite detection system is built, we can clearly see each requirement met:

- Distinguish between different gaseous compositions.
    - The gas sensor array, code, and machine learning model prove that the system is able to detect different gas compositions. First, the resistivity values which change based on composition proves there is a change detection, and the machine learning model makes it further interpretable for users.
- Able to report to the user when a recognized change is detected.
    - The Machine learning model and the LEDs built into the beehive evidently do this, the machine learning model able to show what it thinks the beehive is via text, and the LEDs letting the user know if they are not able to access a computer.
- Can automatically sample air data
    - The GasSensorInterface and GasSensorInterfaceBT show that the system can act automatically as soon as the user prompts the system, either via Bluetooth or through a computer. One that button is pressed, the system requires no more human interaction
- Can self-cleanse detection chamber
    - The DC motor which runs to collect the gas can be run again to cleanse out the chamber in which the gas sensor reading is made. This requirement was seen to be met through the repeated collection of data, which would not have been possible if the gas motor was unable to clean out the system, introducing inaccuracies and noise.

Future Design Considerations

For the next prototype, there are a variety of considerations which are required to be made.

First, the motor controller citrus needs to be more robust, perhaps utilizing a controller chip instead of discrete parts. With that, the hardware in which all of the electronics are integrated also needs to be further updated. The biggest problems with the system were the motor controller circuit not being very reliable, and the housing of the sensors and controller not actually being able to fit inside the beehive itself. These two issues could be resolved with greater care and consideration in their design.

For instance, the housing could be modeled in such a way where the Arduino is housed in a discrete, separate housing, which would greatly reduce the clearance within the system. Furthermore, the system could be further abstracted through the use of the ATMEGA microcontroller itself and custom PCBs, which would not only streamline the design process, but the manufacturing and assembly issues as well.

Another design consideration would be to have a self-learning ML model, one which will be able to update itself on collected data as opposed to relying on a single static set of previous data collected. Finally, the bluetooth integration could be designed to be more elegant and incorporated into the software application already designed, and thus become device agnostic.

Appendix C2: Climate Control Detailed Design

Climate Control Design

The core functionality of the climate control system includes:

- Sense currently temperature and humidity accurately
- Heating plate could reach the optimal temperature for bees to thrive in
- Control heater (on/off) depending on the current temperature inside the hive
- Ability to relay information from sensors to the users' mobile app

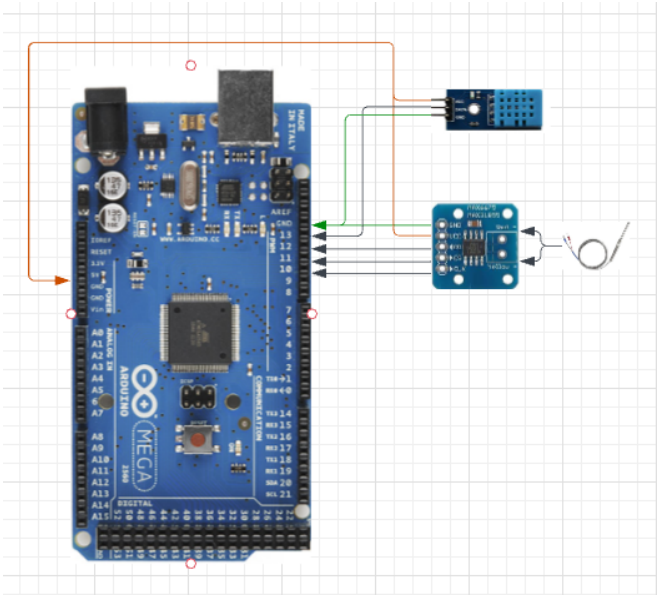
Sensor Selection and Set Ups

To fulfill the requirements of being able to sense temperature and humidity, OTS(off the shelf) sensors could be purchased for the arduino. For temperature, multiple sensors was determined as the best option since temperature is crucial for the climate control system and having multiple inside the hive will mitigate the risks where one of the temperature sensors fails.

For moisture detection, DHT11 humidity sensor was used since it was popular for detecting the air moisture and the most readily available to use for arduinos. The DHT11 sensor came with both temperature and moisture measurement; however, its measurement of temperature only ranges from 0 C to 50 C, and since in the winter, temperatures will usually be much colder than 0 C, we needed another sensor for temperature.

The temperature sensor needed to handle cold and warm temperatures, and thus the decision was to go with a thermocouple. A K-type thermocouple with a MAX31855 Thermocouple to Digital Converter was chosen for sensing the temperature. A total of three thermocouple is used in the design as it allows us to measure the temperature at the sides of the hive as well as the temperature at the center of the hive.

The setup could be seen below with the Arduino.



The sensitivity of the thermocouple was tested in various conditions, and was found to be promising.

Actual temperature	Thermocouple temperature
23 C	22.50 C
25 C	25.25 C
40 C (Warm water)	40.50 C
0 C (Ice water)	1.25 C

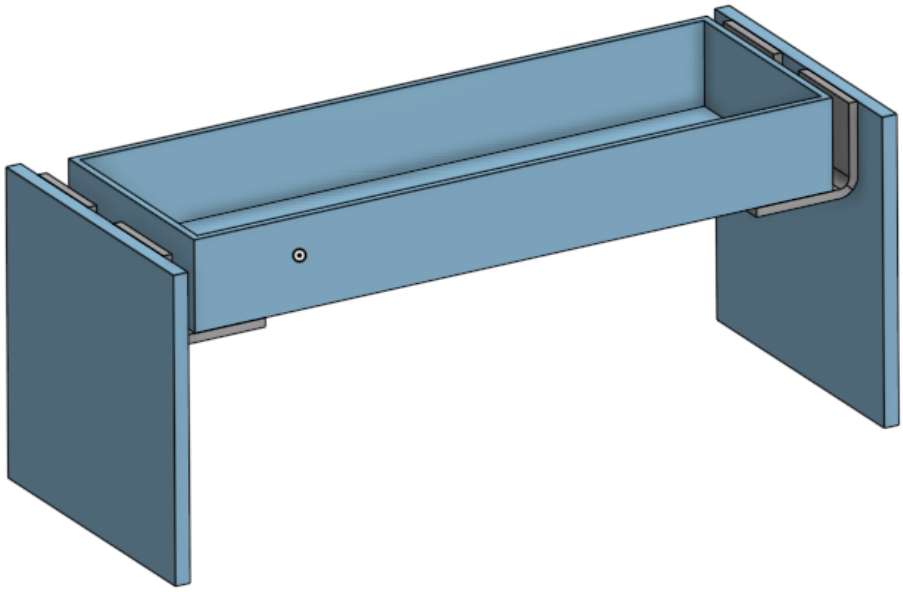
Through testing the thermocouple, it could be concluded that the thermocouple was in fact accurate enough for our application and it fulfills the requirement for temperature sensing capability.

Heating Element in Hive Design

The design of the placement of the heaters has gone through a couple of iterations before reaching our final design. The key requirement that the heating system needs to fulfill is to reach and maintain the ideal temperature for the beehive.

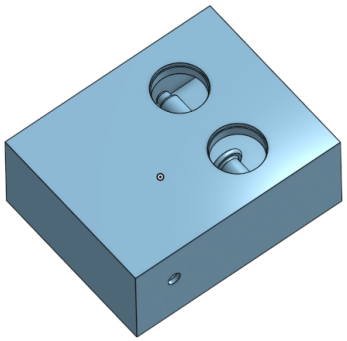
Design #1

Initially, the design of the heater was focused on a simple approach where we would just need to place the heaters inside the beehive without making any changes to the hive. There are OTS beehive heaters that could fit in between the space between the two frames inside the beehive. This design features a housing for sensors and the arduino alongside a battery inside for power, and the heaters would be connected to the housing via two slotted L brackets each, mounted with one screw in each slot, this would allow slight movement in the horizontal and the vertical direction for the heaters so that beekeepers could have a little bit of flexibility of how many frames could be in between the two heaters. Ultimately, this design's placement of sensors and all the electronics inside the beehive makes it difficult, since there are very little space inside the hive and it will be difficult for bees



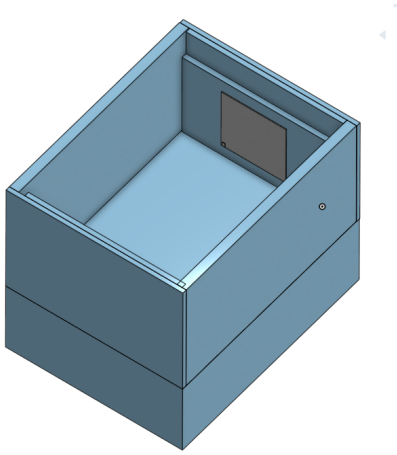
Design #2

Inspired by the way a lot of DIY beekeepers currently keep their bees warm in the winter solution. Since heat flows upwards, this heater design was meant to be on the bottom of the hive unit, with holes(with a metal net to keep bees away) to allow the heat to flow from the bottom chamber to the top. Two ceramic heating lamps would be used to generate heat. After reviewing this design, it was dropped because it will heat up the electronics in the bottom chamber, and could cause problems for the arduino and the sensors.



Final Design

This design was ultimately selected as the design to move forward with because of its placement for the heaters and its ability to overcome the challenges the previous two designs faced. By having the heaters directly integrated with the walls, it will only take minimal space, and the electronics will be housed in a drawer like compartment below the beehive while the sensors will stick out into the beehive, this design minimizes the risks of bees going near the electronics and uses the space inside the hive optimally.



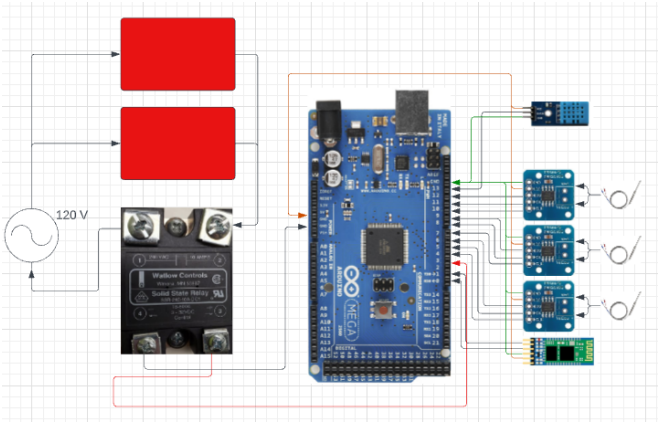
The final heating element inside the hive needed to be thin and be able to reach a temperature of at least 40C. An adhesive heater was finally chosen because of its thickness and its ability to stick onto the side of the beehive.

By connecting the heater to 120VAC, the heater was able to heat up quickly and it was able to reach 40 C easily, and fulfill the requirement.

Trials	Temperature of the heater
Trial 1	41.00 C
Trial 2	44.25 C
Trial 3	42.75 C

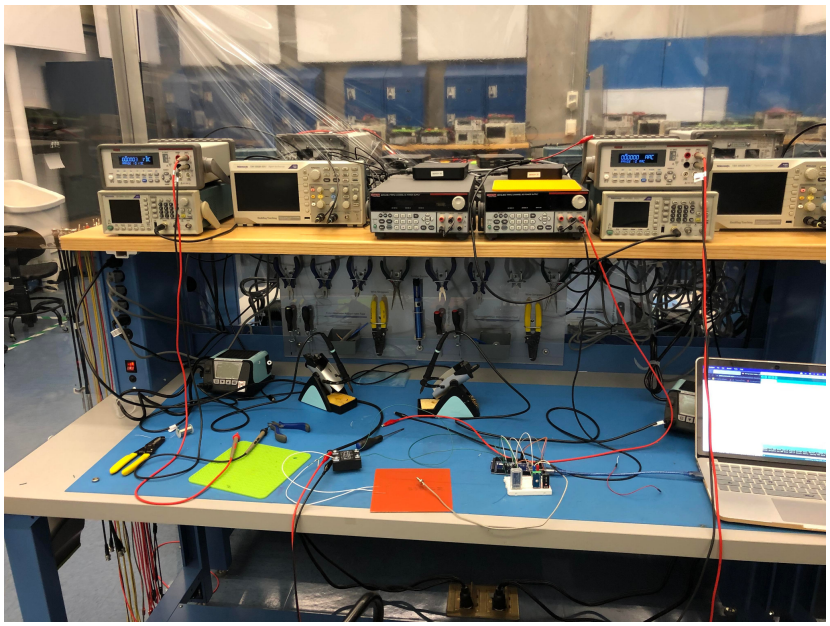
Controlling Heating Element

The heating element is controlled with the arduino, since the arduino will gather information regarding the temperature inside the hive, it will turn on the heater when the temperature is not in the ideal range and turn off the heater once ideal temperature is reached. This could be controlled via a (SSR)Solid State Relay using the arduino to send 5V signals to the SSR, the heaters could be turned on, and when the arduino does not send 5V signals, the heaters will be off.



Arduino Code for Sensors, Heaters, and Bluetooth

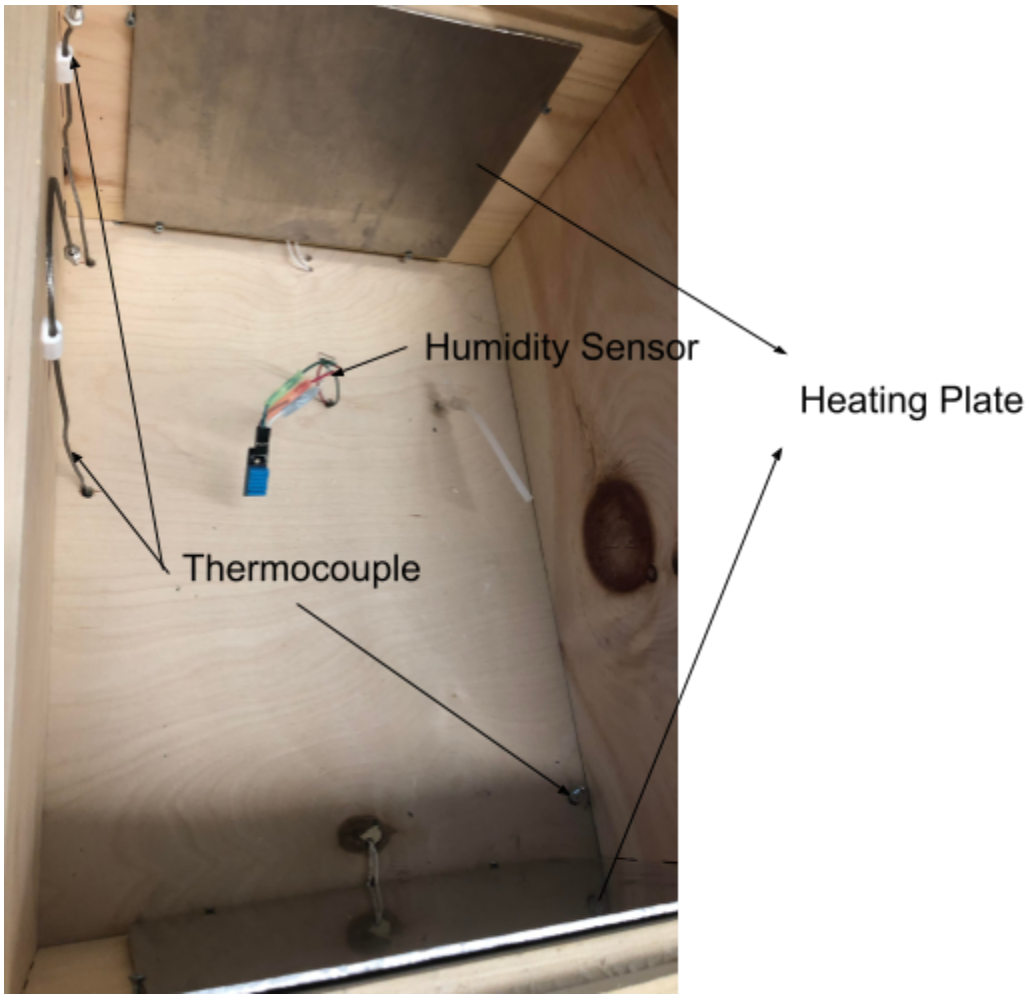
The Arduino Mega 2560 was used in climate control, its purpose is to gather sensor data from the humidity sensor and the thermocouples, communicate this information to the user via bluetooth connection, as well as controlling the status of the heater based on the current temperature of the hive. The code used can be found in Appendix



The setup was tested with a multimeter across the two knobs measuring the voltage difference. When the thermocouple's reading is not in the ideal temperature range, the arduino would send a 5V control signal to the solid state relay and the heater would turn on and the voltage reading on the multimeter would produce 0 VAC, and when the heater has reached the ideal temperature, the arduino would turn off the heater, and the reading on the multimeter would read 120 VAC.

The final design of the heating system consists of two heating plates on both sides of the hive, three thermocouples, one humidity sensor, one bluetooth module, one solid state relay and an arduino. The choice of three thermocouples is to minimize risks, the arduino will take the median

of the three temperatures so that in the case that one of the thermocouples fails, the system would still function as planned.



Meeting Requirements

In terms of the requirements of climate control, all requirements have been met with the exception of the humidity requirement. Which is due to a lack of resources to test as the team was running low on budget and we did not have anything to measure the humidity against.

The requirements have been met:

- Detect temperature accurately (± 2 C)
    - Through testing the thermocouple thoroughly, and verifying that it could sense temperature consistently.
- Heating plate could reach 40 C
    - By connecting the adhesive heaters to 120VAC, it could reach 40 C and above easily.
- Can control temperature inside the hive
    - By using a SSR with an arduino that will take the temperature of the beehive constantly, the heaters will be turned on when the temperature from the

thermocouple indicates that it is not at the ideal temperature and will turn off when it reaches the ideal temperature.

- Report to user with the basic condition inside the hive
    - The arduino is equipped with a Bluetooth module and it will constantly send the humidity and temperature information to the users via Bluetooth Low Energy.

Future Design Considerations

Moving forward, it would be a great value add to have a stackable design for the beehive, because it is a common practice in the beekeeping industry to have beehives stacked on top of each other. The stackable design would allow for heaters on the top hive to connect via a parallel connection to the power, reducing the amount of outlets needed for power.

The climate control's efficiency would greatly improve if insulation could be added to the walls of the hive. The hives are built with wood and although that wood is not a bad insulator, the walls where the heaters are placed are relatively thin; thus, having insulation in the winter would be able to save electricity for the beekeepers. Insulation could be an extra layer of foil and wood that could attach to the beehive via magnets, and beekeepers could take it off easily from the beehive when the weather is warm.

Additionally, users able to control the temperature of the hive would be a great integration for the software. Having the ability to change the temperature inside their hive depending on what they want and the weather condition could be a value add for beekeepers that want to be more hands-on. Maybe beekeepers in the winter decide they do not want to have their hives at 30 C all the time in order to save electricity, or just to turn off heating during the spring and summer.

Appendix C3: Detailed Design for Mobile App

Detailed Requirements

name	use_case_id	sub_use_case_id	description
Authentication & Profile	1	1	A new user wants to register a new account into the platform (may or may not have a Nectarfy hive).
		2	An existing user wants to login to an existing account.
		3	An existing user wants to manage their account
Hive Management	2	1	An existing user wants to register a new Nectarfy hive to monitor through the app.
		2	An existing user wants to manage all the Nectarfy hives he/she has.
		3	An existing user wants to access the individual information of a registered Nectarfy hive.
		4	An existing user wants to be notified about possible actions to do regarding his/her Nectarfy hive.
Community	3	1	An existing user wants to navigate through different topics in the discussion board.
		2	An existing user wants to engage with specific discussion topics in the discussion board.
		3	An existing user wants to manage posts and comments made by the user.
		4	An existing user wants to flag innapropriate content in the discussion board.

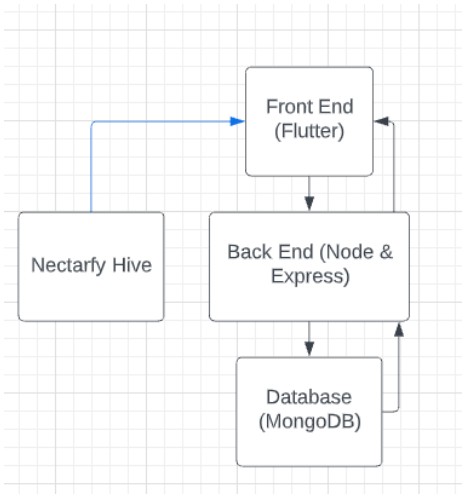
Database

We are using MongoDB Atlas for our project and all information is stored in a singular database, and in that one database, there are multiple collections each representing different information that is being stored.

Mobile App Design

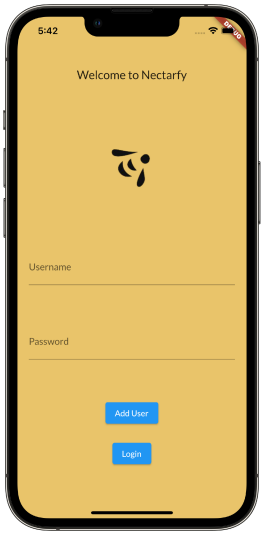
The Nectarfy mobile app is developed with the use of the flutter framework, its advantage being that it could run on both IOS and Android systems while having a single codebase. We used MongoDB for storing information about users, posts, etc. For the backend that connects the frontend and the database, we are using a Node Server with Express framework.

Overall Architecture



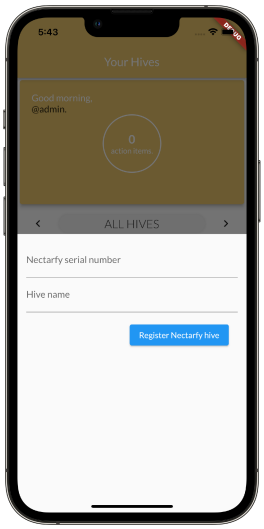
The front end of the app would interact with the backend of our app through making API requests, for example, when the user wants to create an account, a POST request is sent to the backend, and when the user navigates to the community section of our app, a GET request is send to the backend, the backend would search through the posts and give the fontend a JSON object with the posts that are going to be seen, afterwards, frontend would parse the JSON object, and display it on the screen.
The app currently can connect via bluetooth low energy, when the user is on the hive module section, the app would look for a bluetooth low energy device with our name, and automatically connect in the background, and after it connects, it would display the information on the screen; furthermore, we have put a timer for the bluetooth to refresh and get information again in 10 seconds, so the user will always have the most updated condition of the hive.

Features

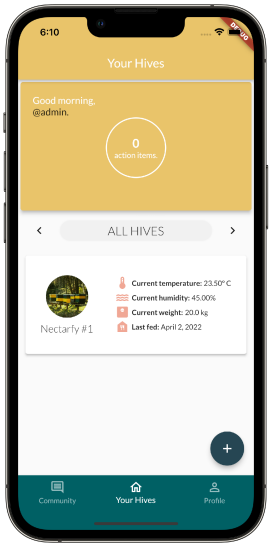


Register and login

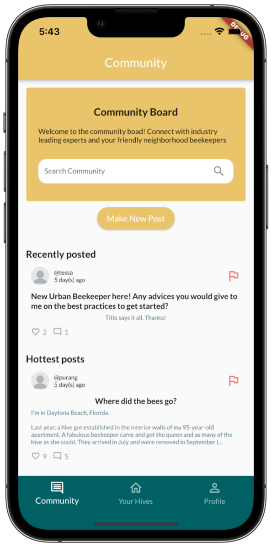
- User could register and login on this page



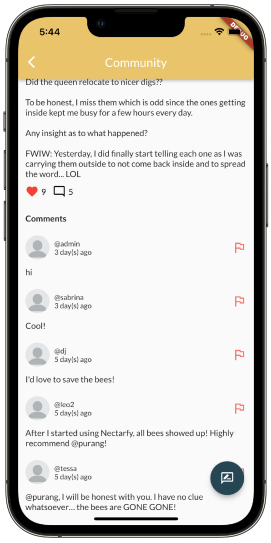
User could register new hive on their app



Bluetooth automatically discover and connect to nectarfy hive via BLE

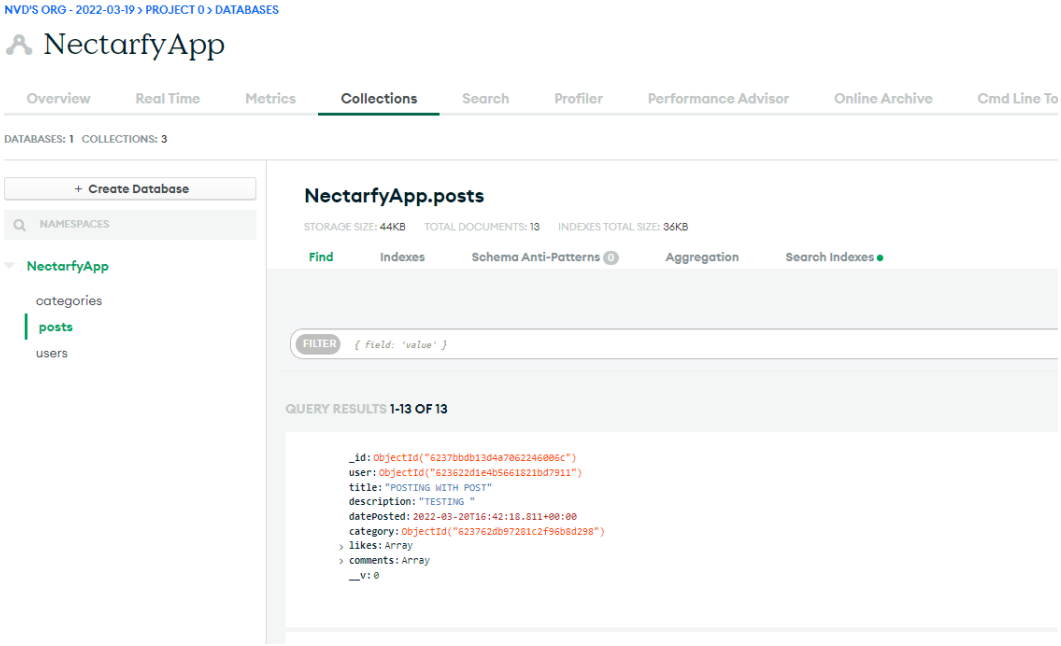


Community module: on this page, a get request is sent and posts information is fetched via the backend and displayed on the screen



Interacting with posts: users could like and comment on the posts

Database



Meeting Requirements

Given the overview of how the mobile app is built, we can clearly see each requirement met:

- Connection from user's smart device to the Nectarfy Hive
    - The app will automatically search nearby bluetooth devices and would automatically connect the the device with our name, making connecting to Nectarfy hives as easy as possible
- Authentication: users could sign up, sign in, and log out of the app

    - The app connects to our database via our backend, which is built with Node js and Express as the server. The app also has an internal state management for the user, so that different things would display depending on if the user is signed in or not.
- Users could interact with each other in the community of beekeepers
    - Each registered user has the ability to post, like and comment in our community module, and all requests are being sent to and processed by our backend server that would update the database in real time.

Appendix D: Prototyping, Verification/Validation Test Results, and Risk

Varroa Mite Verification Test Results

Verification Test: Expose sensors to different chemical compositions and read resistance values. The chart below is a sample of the data being recorded by the sensor array, which shows there are significant resistance changes. It can easily be seen that the resistance values between different gasses is beyond 10% differences between the different gas samples.

TGS2600	TGS2603	TGS2602	TGS2620	TGS832A	Label
10	13	7	3	21	Air
10	14	7	3	21	Air
10	14	7	3	21	Air
10	14	7	3	21	Air
10	13	7	3	21	Air
10	13	7	3	21	Air
10	13	7	3	21	Air
10	13	7	3	21	Air
10	13	7	3	21	Air
116	237	255	13	115	Eth
111	230	249	13	109	Eth
107	224	242	15	105	Eth
103	219	236	46	103	Eth
101	221	232	58	122	Eth
101	221	228	53	129	Eth
99	217	223	43	127	Eth
96	211	217	35	122	Eth
93	206	210	29	117	Eth
89	201	204	24	112	Eth
86	196	197	21	107	Eth
15	31	18	23	80	Breath
15	34	20	27	91	Breath
15	35	21	27	99	Breath
16	37	22	22	99	Breath
16	38	23	19	96	Breath
16	40	24	18	91	Breath
16	42	25	16	87	Breath
16	45	26	16	82	Breath
17	46	26	16	80	Breath
17	48	28	19	83	Breath
17	52	29	23	95	Breath

Verification Test: Train machine learning models on various gaseous compositions, then run test and prediction analysis.
After running the machine learning model on the collected data, we find the following prediction accuracies:

```
In [2]: runcell(0, 'C:/NVD/nectarfy-engineering/PythonBeeHiveGasSensor/MLModel.py')
[0.9986911  1.         1.         0.99868938 0.99868938]
```

Evidently, these accuracies are absurdly high. This is due to the lack of variation within data and lack of amount of data collected. For future further analysis, more complex gaseous compositions will yield greater differences within the data, and as such would yield lower accuracies.

Verification Test: Run prediction and check if the system is able to report the gas.

The computer program consistently reports its prediction back to the user. Furthermore, there is an indicator LED which can also report conditions to the user.

```
['The prediction is Air']
```

Verification Test: Upon activation of the system, do not touch the system until collection and prediction is complete, and check results.

The following test steps were used to test activation of the system.

1. Connect to the system via Bluetooth. Ensure connection to the Pi is made.
2. Press the activation button (blue button)
3. Check with audio cues if the motor is running, check software feedback messages to see what state it is at.
4. After collection is complete, press the prediction button (green button)
5. Check to see if a prediction is reported. Check the .csv file for test data as well as rotating gas from the chamber each test to ensure the gas is being collected.

Only upon full completion all five steps is a test considered to be successful.

TEST #	**RESULT**
1	Pass
2	Fail (Motor Failure)
3	Fail (Motor Failure)
4	Pass
5	Pass
6	Pass
7	Pass
8	Pass
9	Pass
10	Pass

It can be seen that the 80% success rate is lower than the desired threshold. After running these 10 tests, the motor failure was diagnosed to be due to exposed connections which shorted.These were rectified by insulating the motor driver circuit. Another set of tests were run with a 100%

success rate, indicating the issues had been resolved.

Verification Test: Collect a sample of data, verify that the system identifies it as not-air (base sample). Run the pump system a second time, verify that the system identifies it as air (base sample).
The two gas samples used for this were room air as the base sample, and hand sanitizer as the alternative gas. This test was also unofficially done in the previous verification test to check prediction values.
The following table shows the results of this test

TEST #	**REAL GAS**	**PREDICTION**
1	Hand Sanitizer	Hand Sanitizer
2	Room Air	Hand Sanitizer
3	Hand Sanitizer	Hand Sanitizer
4	Room Air	Room Air
5	Hand Sanitizer	Hand Sanitizer
6	Room Air	Room Air
7	Hand Sanitizer	Hand Sanitizer
8	Room Air	Room Air
9	Hand Sanitizer	Hand Sanitizer
10	Room Air	Room Air

There was a single instance in which the prediction was incorrect. This was found not to be the case of the system not being able to cleanse itself, but that one of the wires in the system became loose. After rectifying this issue, the 10 tests were run again with a 90% success rate, which passes our threshold.

Varroa Mite Product Risks

For the varroa mite system, there are not too many critical risk harms to humans, but there are a few risks which may occur and can affect the functionality of the product.

1. DC Motor fails to pull gas from tubing
2. Tubing becomes clogged
3. Motor runs out of power
4. Sensors stop functioning properly
5. Computer fails or encounters a bug

Mitigation Steps

1. Have redundancy in the electronic circuit system.
2. Maintain ability for user to SSH/VNC/Control/Communicate with the onboard computer.
    a. Subsequently, ensure peripherals can be attached to the onboard computer, including a mouse, keyboard, and monitor if the above process fails.
3. Design for easy access and maintenance if required for parts of the beehive.
4. Design deadlock protections within software and give easy hardware reset access for software.
5. Design software which can reverse pump flow in an attempt to push out debris while working with the heating system to expand pipe for better debris movement.

Appendix E: Complete Requirement List

Beehive Requirements

Requirement	*Evaluation Criteria*	*Justification*
Maintain a variable living space for bees	Check to see if bees live within beehives for prolonged periods of time.	If bees cannot live within the beehive, it will render the product useless
Cohesively integrate all subsystems, including but not limited to, feeding system, climate control system, varroa mite detection system, any other electronic or mechanical system which contributes to functionality of any or none of the previously listed subsystems.	Test the entire beehive system for functionality, ask users to use the beehive and rate clunkiness.	Without proper integration of all the desired subsystems, the product would not function to the degree at which the customer expects.

Future Requirements	*Justification*
Allow for stacking of multiple instrumented boxes on one base	Beekeepers often change the size of their beehives with regards to how many colonies they wish to maintain. If the beehive is unable to change its size at will for the beekeeper, they are less likely to use them.

Improved Feeding System Requirements

Requirement	*Evaluation Criteria*	*Justification*
Notify the beekeeper when ingredient tanks or mixing tanks are empty.	When the tank is empty, the user is notified, preferably through an app notification, but at the bare minimum by some visual indication on the hive.	Beekeepers must know in a timely manner when either the ingredients or mixing tanks are running low to either provide more ingredients for the system to make simple syrup with or to create a new batch of simple syrup to ensure the bees always have access to food.
Mix user set ratios of ingredients.	System must be able to mix colored water to produce	Different ratios of water and sugar promote different

	different color mixtures that correspond to user set ratios.	behaviors in the bees, it is important for the beekeeper to have control of the ratios so they are able to promote the behaviors they desire through-out the year.
Tanks and other ingredient components must be completely leak-proof.	Water must run through the entire system with no to minimal leaks for the first prototype iteration.	Any component that comes into contact with the ingredients must be completely leak-proof to ensure no damage to electronics or the hive itself. Moisture in particular can significantly affect overall hive health so it's important for no water to leak.
Mixing tank must be closed off from the hive while mixing is underway.	While the mixing mechanism is running, no movement of liquid must be observed from the mixing tank to the feeding area.	To ensure that no liquid enters the hive and creates moisture and mold in the hive which can hurt the bees, the mixing tank must be sealed off from hive while mixing as this will be when the liquid is the most turbulent and likely to leak.
The holes the bees feed from must be able to hold surface tension.	When the system is sealed, no fluid should leak from the feeding area.	The bees must be able to access their food without risk of liquid spilling into the hive.
Ingredients must be mixed autonomously.	Ingredients must be mixed without any physical mixing done by the beekeeper.	In the future, the beekeeper will be able to create a new batch of simple syrup remotely, therefore, the system must be able to mix ingredients without any one nearby in the most efficient manner possible.

Future Requirements	*Justification*
System must be of an ergonomic size and weight.	It is important for the system to be easy to remove from the hive so that the beekeepers can access their bees with ease. The MVP is focused on proving functionality, therefore, size can be overlooked at the moment but

	must be considered in the future before releasing as a product.
System must include three tanks.	Many beekeepers like to add additives to their bee's food so there must be an additional tank to store these additives so they can be added to the mixture when desired.
Reduced power consumption.	Currently, the prototype requires a minimum of 24 VDC to function which would be unpractical for a unit that is designed to work off solar power, therefore, significant improvements must be made in the electronics department to ensure maximum efficiency.
System must monitor the exact volume of each substance in the tanks.	Instead of simply notifying the beekeeper when the tank is low, the beekeeper should know exactly how much of each substance is in the tank to plan activities and manage their system most efficiently,
System must be able to send notifications and allow for ratios to be set remotely.	Currently, system requires the beekeeper to be near the unit and have a computer plugged into it to see tank empty lights, set ratios of food, and dispense food. To become a true solution, all of these tasks must be done remotely.

Climate Control System Requirements

Requirement	*Evaluation Criteria*	*Justification*
Detect temperature accurately (± 2 C)	The measured temperature must be close to the actual temperature of the environment, small margin of error is allowed.	The climate control system needs to know the temperature in order to turn on the heater when the temperature is too cold and turn the heater off when the temperature has reached the ideal temperature.
Sense humidity	Measured humidity must change when more moisture is present in the environment.	The optimal humidity for bees is 50% - 60%, and this information should be given to the users for them to monitor the overall hive's

		health.
Heating plate could reach 40C	When plugged into power (120VAC), the heater can reach 40C	Bees thrive between 30-35 C. This temperature is optimal to raise brood and promotes colony fitness.
Can control temperature inside the hive	When the system has power, it should be able to reach the ideal temperature inside the hive and maintain the temperature	The ability to heat up the hive from the inside would save beekeepers a lot of time, and also ensures the bees would not die of the cold snaps in the winter.
Report to user with the basic condition inside the hive	The bluetooth module can send data via bluetooth low energy to bluetooth devices	The user needs to know the overall health of the hive without opening the hive and checking it themselves. Beekeepers validated that opening a hive and checking on the bees conditions is something bees do not appreciate.

Future Requirements	*Justification*
Stackable design of the integrated heaters	Beehives are often stacked on top of each other to allow room for expansion. This is a common practice across the industry and thus it is important for us to provide the ability for Nectarfy hives to stack on top of each other while having a single source of power.
User's could control the temperatures in the hive themselves.	Beekeepers could have the ability to change the temperature inside their hive depending on what they want and the weather condition. Maybe beekeepers in the winter decide they do not want to have their hives at 30 C all the time in order to save electricity, or just to turn off heating during the spring and summer.

Varroa Mite Detection System Design Requirements

Design Requirement	*Evaluation Criteria*	*Justification*

Distinguish between different gaseous compositions.	Reading resistance values from the sensors should change and reach a steady state when obvious gasses like ethanol are present.	Without the ability to detect changes in the atmospheric gas composition, the system does not work.
Machine learning model can learn and predict different gaseous compositions	The machine learning model will be trained on known but imprecise gaseous composition changes, such as human breath and air within a room. The machine learning model should have at least 90% accuracy when making predictions.	Without the ability to learn and identify different gaseous compositions, the system does not work.
Able to report to the user when a recognized change is detected.	Indicate preferably through a mobile application, but either as a text-based report on a computer or physical indicator on the hive when a distinct variation of gas is detected.	Without being able to notify the user when changes occur, and subsequently when the hive is in a healthy or unhealthy state defeats the purpose of the system. The system will recognize changes based on a machine learning model which can process the collected data and perform predictions on them.
Can automatically sample air data	Upon command given, show that the air pump runs and collects a significant amount of gas. Significant amount would be enough gas such that the resistivity values show a change, and the detection	If the beekeeper has to go to the beehive to check for mites,most beekeepers would check for mites their own way, which would render the mite detection system useless. Beekeepers have emphasized that a system they would purchase must be autonomous and require little labor from them.
Can self-cleanse detection chamber	After the successful prediction and detection of a non-air gas, running the system and prediction again should result in an “air” or base-gas prediction.	If the chamber in which the gas being sampled is not cleansed, it would void the accuracy of the results, rendering the product useless.

Future Requirements	*Justification*
Distinguish between gaseous compositions of infected and non-infected hives	Because the purpose of this system is to detect varroa mite, it needs to be able to achieve this requirement. This requirement was waived in favor of the requirement to distinguish between different gaseous compositions due to the timeframe and seasonality of the industry, which would not allow us to build an MVP in time for major deadlines.
Self-update machine learning prediction model?	If this system is to be effective against parasites, it must be able to continuously learn and update the prediction model it uses to check if beehives are infected or not. WIthout this requirement, it would render the system useless after a change in atmospheric conditions.
Able to act upon the detection system to trigger another system which can treat beehives for varroa mite.	This is the next step in the varroa detection system, based on the interviews and validation conducted. The beekeepers cared most about detecting AND treating varroa mite. However, we found that treating Because of the limitations of our skills and time, we were not able to integrate this technology into our current MVP, but will be a requirement for the future MVP.

Mobile Application Requirements

Requirement	*Evaluation Criteria*	*Justification*
Connection from user's smart device to the Nectarfy Hive	The mobile app could find, connect, and receive data from the hive via bluetooth low energy	The users need to be able to see what is going on inside the hive without going out and opening the hives.
Authentication: users could sign up, sign in, and log out of the app.	Frontend could send requests creating users and logging in by comparing passwords in the database while ensuring the password is encrypted	Authentication needs to exist so that users could log into their Nectarfy account from any device.
Users could interact with each other in the community of	Users can interact with the frontend, and actions could be	Users interacting with each other is essential to building

beekeepers: users could make a post, like a post and comment on a post.	converted into RESTful APIs, making requests to the database	beekeepers' community, providing people with common interests a place to share their knowledge and help others.

Future Requirements	*Justification*
Recommended actions based on the current condition of the hive and the current weather condition in the area.	Providing the users with what to do next is important in educating new beekeepers. This feature will be added once the team gathers enough expertise for each scenario.
Ability to control climate control, feeding system, and test varroa levels from their phones.	This would make beekeeping simple and allow it to fit into people's busy schedules, only needing to take care of the bees when it is absolutely needed.

Appendix F: Product overview