

Team 8 (Smart Hive)

Design Document

Nicholas Paschke, Ahmed Alsamahi, Neil Sankineni, Sidney Amber-Messick,

Ismael Torres, Ergi Masati

12/2/2022

Introduction

The honeybee population is critical for the health of wild plants and important for the fruit and vegetables we consume. It is estimated that one-third of the food we eat in America is pollinated by honeybees. However, honey bees are dying at unprecedented rates due to invasive species, pesticides, and air pollution. They require our intervention to prevent extinction. Our Senior Design team has chosen to contribute to the Smart Hive Project at George Mason University - an effort aimed to remotely monitor bee hives to measure hive health and prevent bee swarms. This will be accomplished through the deployment of an array of sensors (Carbon Dioxide, temperature, humidity, and possibly weight level sensors) to collect data, store the data on a database, and display the data on a web dashboard. The stored data can be used to analyze trends and can be made available to beekeepers to keep the bee population healthy and separate colonies when they grow too large.

Requirements specification

The following list was gathered from the customer of note in a direct interview and cited within a previous project conducted on the exact grounds.

1. The power source provided on premises is an array of 4 nickel batteries (3.3 V Power supply required for Raspberry Pi)
2. The general maintenance of the apiary is short intermittent visits 4 times each week
3. Hives during acclimate temperatures, are opened to inspect the hive's health
4. A range of 1 to four sensors per frame was found to be accurate enough for the information required of the customer.
5. Light is found to be a key disturbance for the hives and must be kept to a minimum if not completely nullified
6. 0.375 of an inch was found to be the minimum gap which
7. The expected temperature within the hive is 90-93 degrees Fahrenheit provided by the client, while the expected humidity within the hive to support healthy colonies is 40%-60%. However, the temperature can fall far below this up to 32 degrees.
8. The distance from the main power supply to the hives is less than 15 feet
9. The system should be functional for 1 month, as specified by the client

System design/architecture

The generic case for the overall functionality of the system is broken down into four main components: The smart frame, the “beebox”, the database, and the website. These four parts work together in order to provide a web app service for the customer. The smart frame is designed to be modular and low-cost to provide advantages over the current systems in place. The frames will use a GPIO pinout in order to deliver data from the frames to the beebox. The beebox acts as a central hub for processing the sensor data. This controller then formats the information and sends it to the MERN stack for the website on the front end to be able to pick it up with an Axios API. The API will allow the front end to create use-states and webhooks in order to generate graphs for the website. Additionally, the website utilizes libraries in order to format CSV and excel spreadsheets sent by the sensors and as requested by the customer.

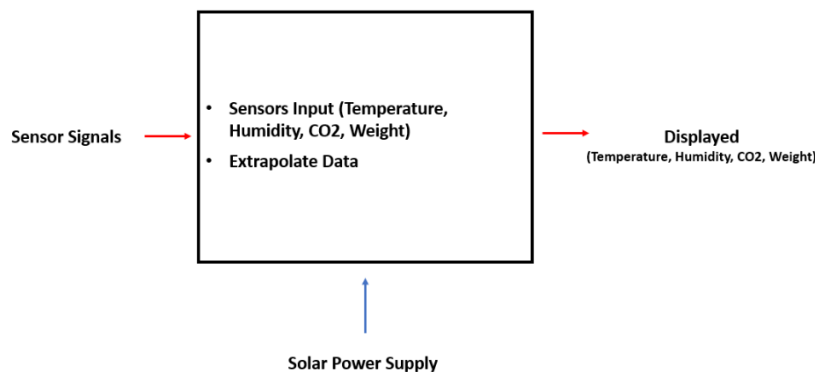


Figure 1: Level 0 Function Design

Node-red is an API-based system that is preinstalled on all raspberry pi devices. Node-red is also known to have rather simple commands in order to have highly flexible functions. The application is completely open source and utilizes javascript react in order to give itself many available libraries to create more functionality between components. One, in particular, we are interested in is the CSV format style as well as the MongoDB server module. This allows us to post records onto a MongoDB server.

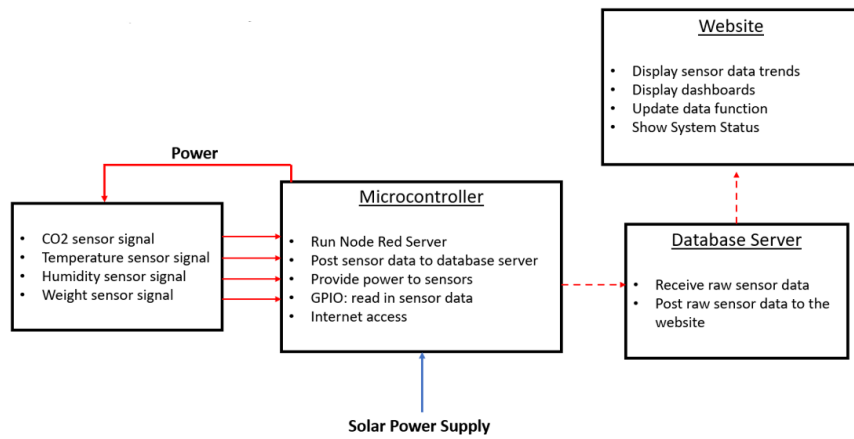


Figure 2: Level 1 Function Design

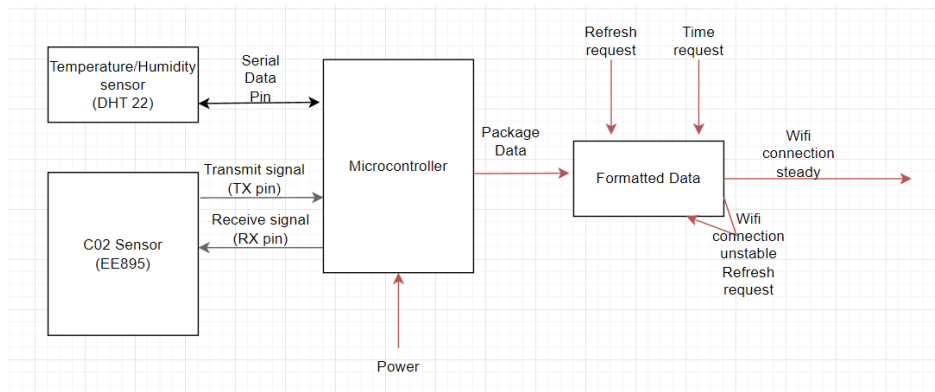


Figure 3: Level 2 Function Design

For the requirements, we have included a functional state diagram which includes a generic version of the smart frame design. This design is not dependent on which sensors are included from a microcontroller or website data schema perspective but may be physically different to accommodate the sensor itself.

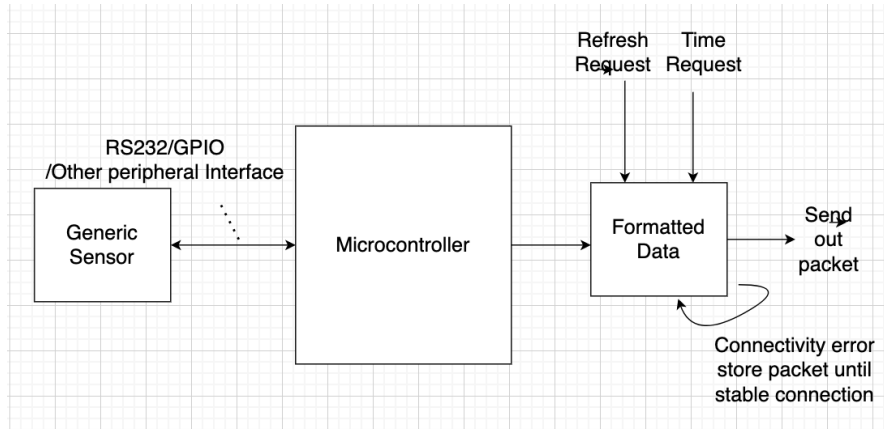


Figure 4: Level 2 Function Design for the Website

As for general Webhook functionality, the functional diagram is shown below:

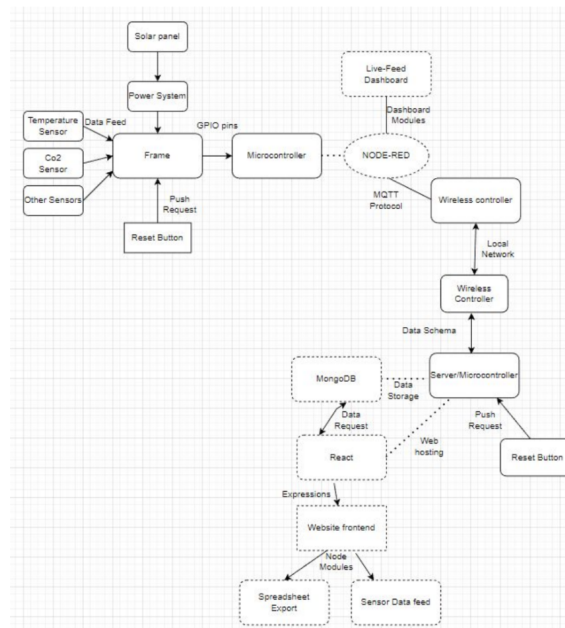
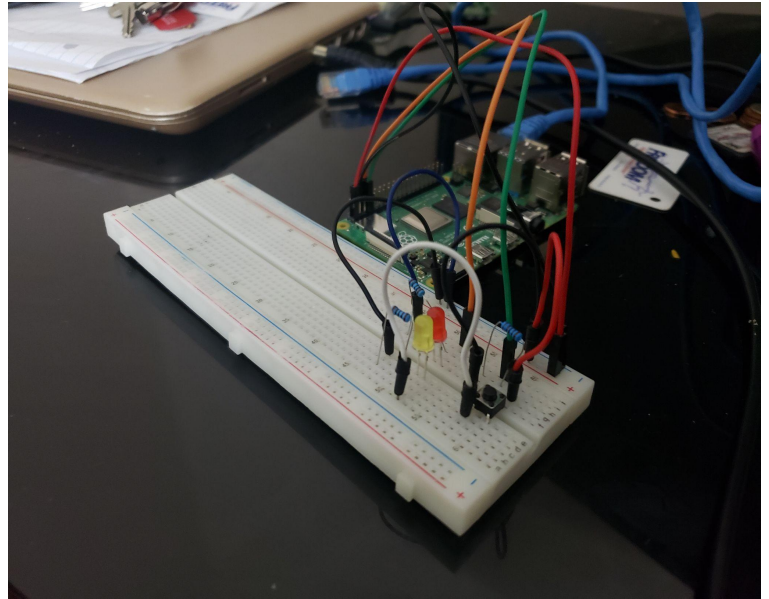
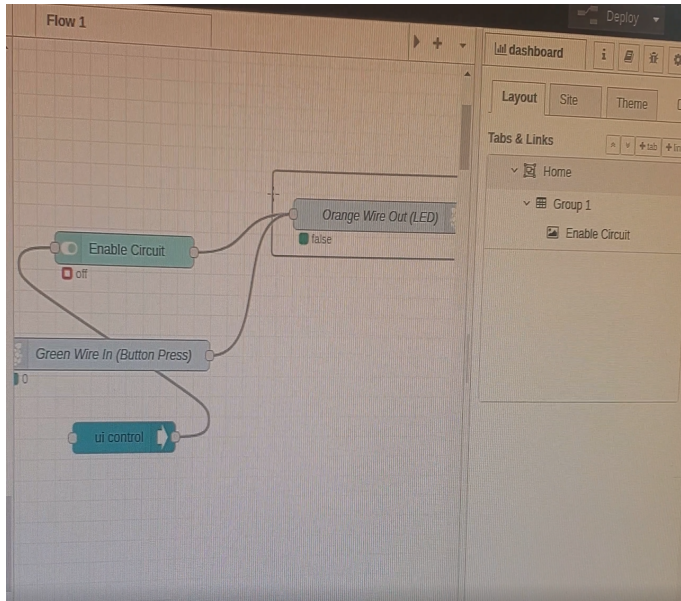


Figure 5: General Function Design

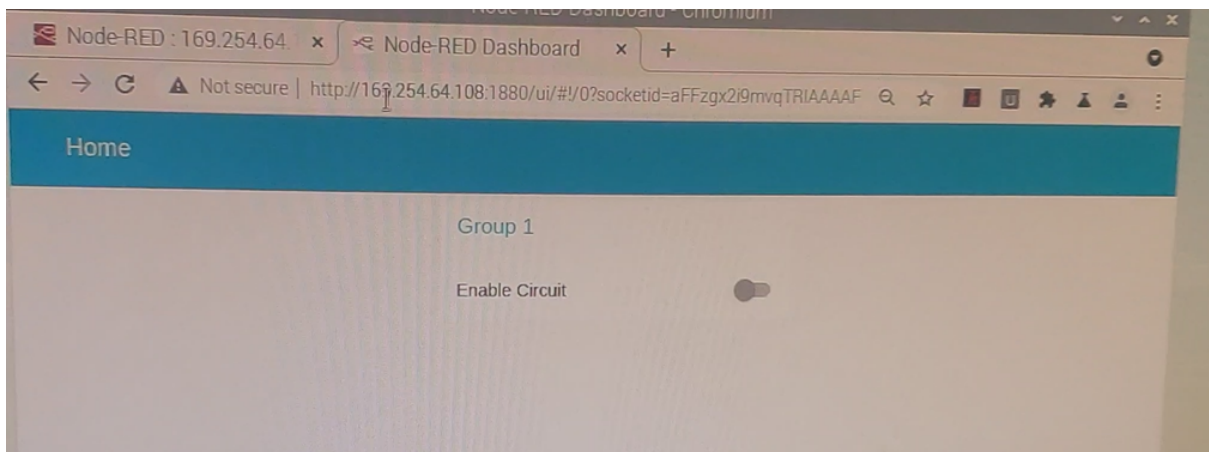
Prototyping Progress Report

Acquired components so far: DHT22 Sensors (2), SCD30 Sensor (1), EE895 Sensor is in transit and has limited testing availability due to its cost as opposed to the SCD30 sensor.

NODE RED TESTING:



To test the functionality and interconnectivity of Node-Red with our sensors and raspberry pi, we are testing the general-purpose inputs and outputs of the node-red dashboard. The breadboard circuit above is the simple button and LED breadboard circuit used to test the GPIO pins on the raspberry pi. The circuit in the leftmost figure above is the node-red circuit which maps the input pin signals received to the output pins on the physical raspberry pi. Finally, the simple dashboard below is the NodeRed dashboard which hides the technical mappings and allows for simple control of the LED. We were able to repeatedly test the functionality and verified that our physical circuit button will only turn on the LEDs when our NodeRed circuit is mapped as such. We were also able to verify that our NodeRed dashboard controls the LEDs from the internet action to the raspberry pi.



SCD30 - CO2 SENSOR TESTING:

To demonstrate the functionality of the Carbon dioxide sensor (SCD30), we repeatedly tested data capture from the sensors to the raspberry pi console output through the I2C serial communication protocol. The SCD30 can measure Carbon Dioxide in parts per million as well as temperature and humidity from the environment. For functionality demonstration, there were 2 tests done showing results for all three variables. The first test involved placing the SCD30 in close proximity to an ice pack to verify that the reported temperature was correct. Our results show that the temperature readings fall from room temperature to 12 degrees celsius in correspondence with the ice pack. The second test involved placing the SCD30 sensor within a Ziploc bag filled with breath. The reason for this is that human breath contains higher carbon dioxide concentration (up to 40,000 ppm) versus clean, ambient air as well as hotter than room temperature. The outputs for this test repeatedly showed humidity going up to near 98%, temperature up to 22.82 degrees Celsius and carbon dioxide levels reaching 40,000 ppm. The results of each test are shown below. Due to the accurate readings and repeatability of these experiments, we can certify that the SCD30 sensor is working and is stamped to be used in our SmartFrame.

Test 1:

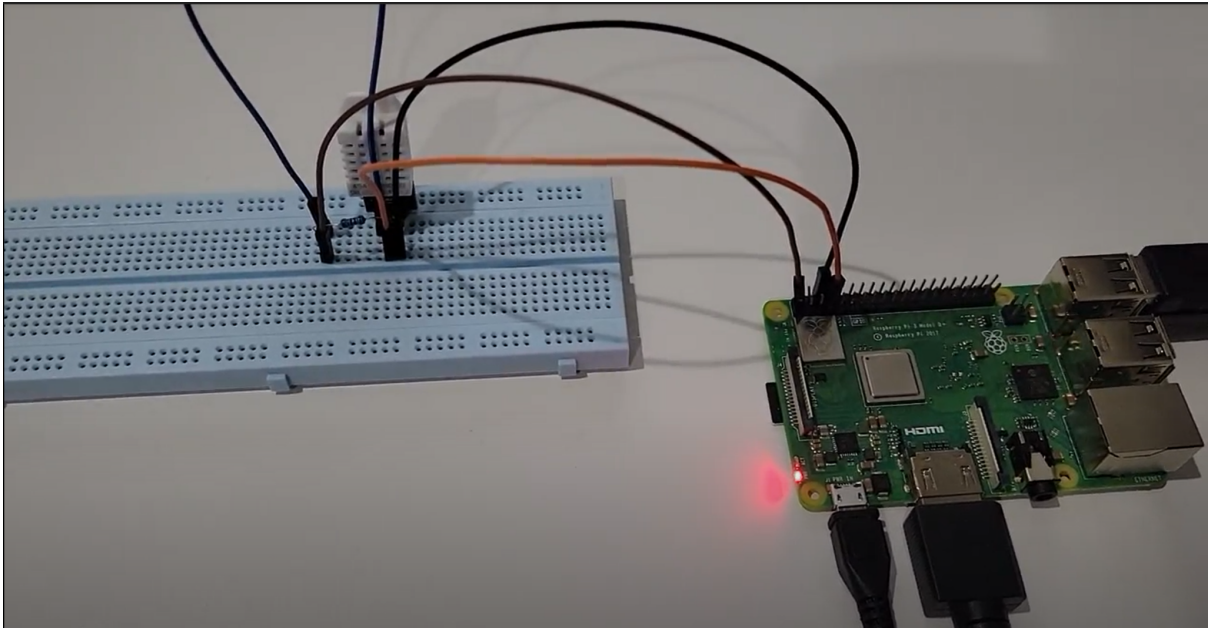
```
pi@TorresRPI:~/Desktop $ python3 CarbonDioxideSensorTest.py
INFO:root:Probing sensor...
INFO:root:Link to sensor established.
INFO:root:Getting firmware version...
INFO:root:Sensor firmware version: 834
INFO:root:Setting measurement interval to 2s...
INFO:root:Enabling automatic self-calibration...
INFO:root:Starting periodic measurement...
INFO:root:ASC status: 1
INFO:root:Measurement interval: 2s
INFO:root:Temperature offset: 0.0'C
CO2: 1380.21ppm, temp: 12.60'C, rh: 47.75%
CO2: 1360.17ppm, temp: 12.53'C, rh: 44.97%
CO2: 1360.19ppm, temp: 12.47'C, rh: 43.66%
CO2: 1350.41ppm, temp: 12.39'C, rh: 42.72%
CO2: 1360.31ppm, temp: 12.32'C, rh: 42.06%
CO2: 1365.79ppm, temp: 12.26'C, rh: 41.55%
CO2: 1363.99ppm, temp: 12.19'C, rh: 41.25%
CO2: 1363.84ppm, temp: 12.16'C, rh: 41.07%
^CINFO:root:Stopping periodic measurement...
```

Test 2:

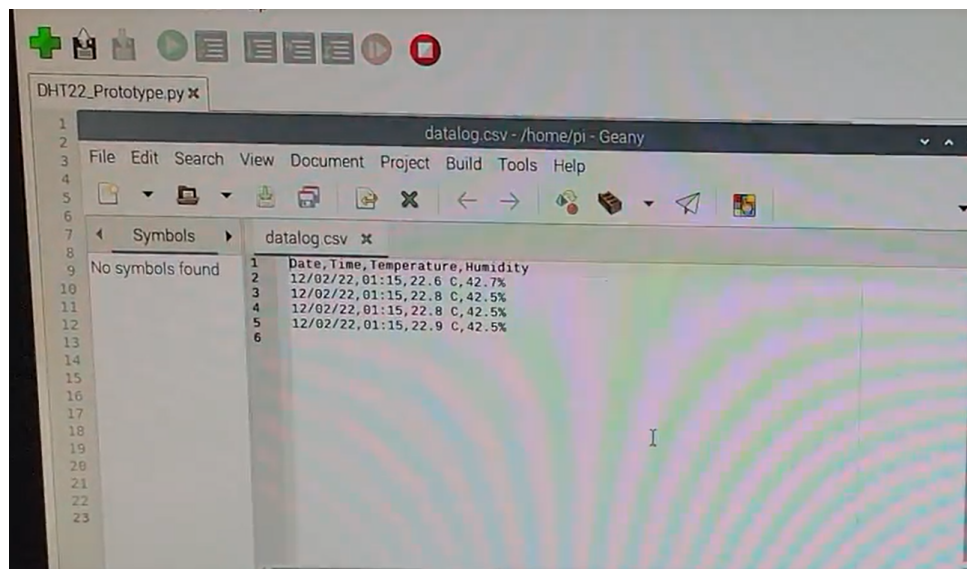
```
pi@TorresRPI:~/Desktop $ python3 CarbonDioxideSensorTest.py
INFO:root:Probing sensor...
INFO:root:Link to sensor established.
INFO:root:Getting firmware version...
INFO:root:Sensor firmware version: 834
INFO:root:Setting measurement interval to 2s...
INFO:root:Enabling automatic self-calibration...
INFO:root:Starting periodic measurement...
INFO:root:ASC status: 1
INFO:root:Measurement interval: 2s
INFO:root:Temperature offset: 0.0'C
CO2: 1368.85ppm, temp: 12.11'C, rh: 41.00%
CO2: 40000.00ppm, temp: 22.75'C, rh: 98.60%
CO2: 38682.30ppm, temp: 22.74'C, rh: 98.59%
CO2: 38277.46ppm, temp: 22.72'C, rh: 98.62%
CO2: 38200.09ppm, temp: 22.74'C, rh: 98.65%
CO2: 37872.04ppm, temp: 22.74'C, rh: 98.69%
CO2: 37720.89ppm, temp: 22.76'C, rh: 98.66%
CO2: 37643.21ppm, temp: 22.76'C, rh: 98.68%
CO2: 37531.22ppm, temp: 22.78'C, rh: 98.73%
CO2: 37533.79ppm, temp: 22.79'C, rh: 98.73%
CO2: 37481.27ppm, temp: 22.82'C, rh: 98.71%
CO2: 37423.14ppm, temp: 22.82'C, rh: 98.75%
^CINFO:root:Stopping periodic measurement...
pi@TorresRPI:~/Desktop $ █
```

DHT22 - Temperature/Humidity SENSOR TESTING:

We tested the DHT22 Sensor by using the I2C serial communication protocol to send the data received from the sensor to the Raspberry Pi. The data including date, time, temperature, and humidity was then saved in a .csv file to be formatted correctly to be sent to the website for graphing eventually. A figure of the circuit is shown below:

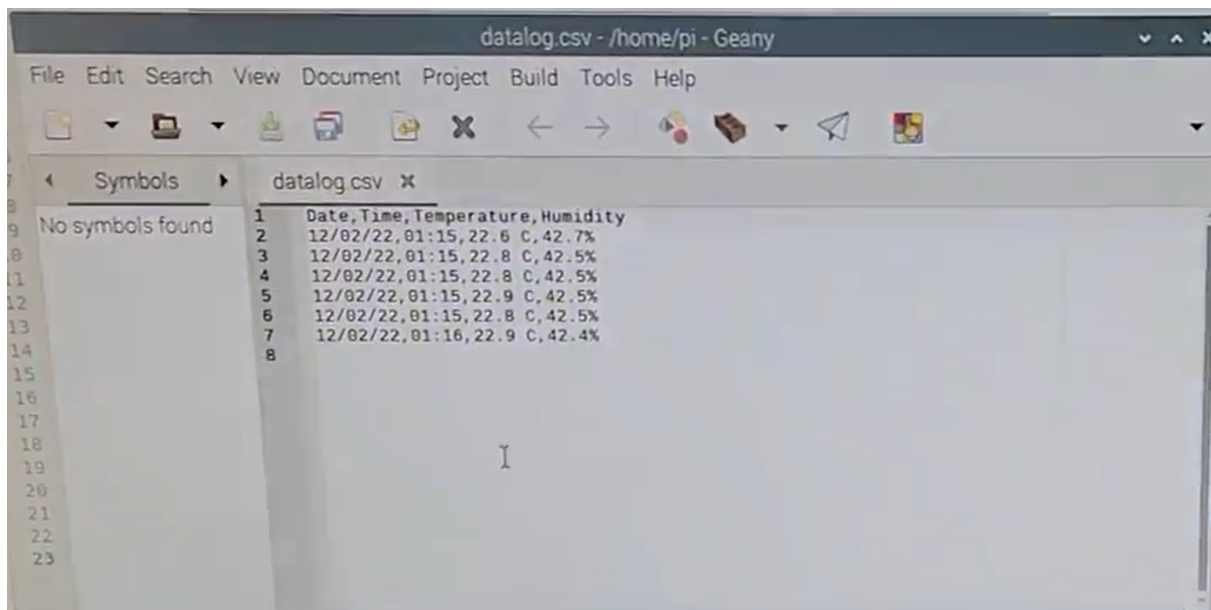


Pin 1 of the DHT22 is connected to the 3.3V power supply, while pin 3 is connected to the ground and pin 2 is connected to GPIO4. A 10k resistor is connected between pin 1 and pin 2. The Adafruit library was used to code the program in python. The program samples temperature and humidity every five seconds (alterable)s, and writes this data to the .csv file. The .csv file for one run of the program is shown below:



```
DHT22_Prototype.py x
1
2
3 File Edit Search View Document Project Build Tools Help
4
5
6
7 Symbols
8 No symbols found
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
datalog.csv - /home/pi - Geany
1 Date,Time,Temperature,Humidity
2 12/02/22,01:15,22.6 C,42.7%
3 12/02/22,01:15,22.8 C,42.5%
4 12/02/22,01:15,22.8 C,42.5%
5 12/02/22,01:15,22.9 C,42.5%
```


A key feature is that when the program is ended for any reason, the .csv file is saved with all of the days, times, temperatures, and humidities. Once the program is started up again, it writes data to the same .csv file so all of the data is still present in one location for sending and graphing. These tests were performed at room temperature but will be experimented with outdoors and in the hive during week 1 of next semester. After ending the programming and then starting it again, the .csv file with a few more entries is shown below:



```
datalog.csv - /home/pi - Geany
File Edit Search View Document Project Build Tools Help
No symbols found
1 Date,Time,Temperature,Humidity
2 12/02/22,01:15,22.6 C,42.7%
3 12/02/22,01:15,22.8 C,42.5%
4 12/02/22,01:15,22.8 C,42.5%
5 12/02/22,01:15,22.9 C,42.5%
6 12/02/22,01:15,22.8 C,42.5%
7 12/02/22,01:16,22.9 C,42.4%
```

Testing plan/Tasks

Further testing will be required on several components of the system:

1. Starting at the highest level of testing, the API must be tested for edge cases as well as the general amount of storage that will be required for the customer to store an appropriate amount of information needed onto the site. This will give us insight into what exactly would be a necessary amount of storage that will be needed on the database. However, cloud-based storage will not require more testing due to the ability of MongoDB to be agnostic towards hosting and casting capabilities when posting records. Cloud-based testing has already been completed, however, the actual record calculation and speed of data transferred from the apiary has not been tested. Success will be if our MERN stack stays reliable over a month-long period of testing as given in our project requirements.

2. NODE-RED has demonstrated the ability to send and receive records from our database on a local level. Next semester we will need to approach the overall database with more physical testing as one of our main challenges is connectivity in case of weather or other aspects. Success will be defined as Node-RED sending data with 100% accuracy and reliability over a month-long period.
3. While the temperature and Co2 sensors are operating correctly individually and sending data to .csv files, we need to test the operation of both sensors when connected to the same raspberry pi. This includes uploading all data to the same .csv file. Success will be indicated by having all of the data present in a single .csv file from both sensors, over a month-long period.
4. While we tested the temperature and CO2 sensors indoors, we will need to utilize headless mode on the Raspberry Pi to test them outside with a power supply. This will be important to test if the sensors stay accurate within the hive. Testing outdoors is a subtask. Testing in the hive is the overall task before final implementation. Success will be indicated by our design by working outdoors for a week-long period (subtask), and then a month-long period in the hive.
5. While the Node-RED interface is working with dummy data, we need to pass data from the sensor to it and see if the data is maintained through the Node-RED interface. This is a subtask of the overall goal, which is to test if the data is sent by the sensors to Node-RED, then sent by Node-Red to the website, and outputted in graphical form. Success in this task will indicate the correct functionality of our overall design, and we will test if all of these subtasks and components operate for a month-long period.

Background knowledge

We utilized the 3-pin DHT22, with the normal “null” pin in the 4-pin DHT22 omitted, the DHT22 sensor hardware schematic is as follows:

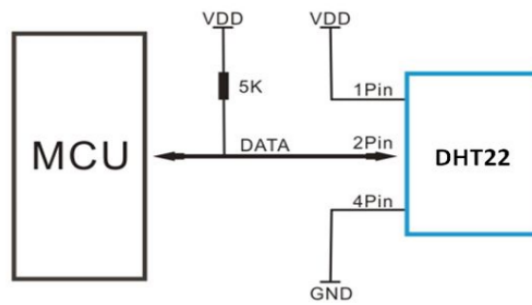
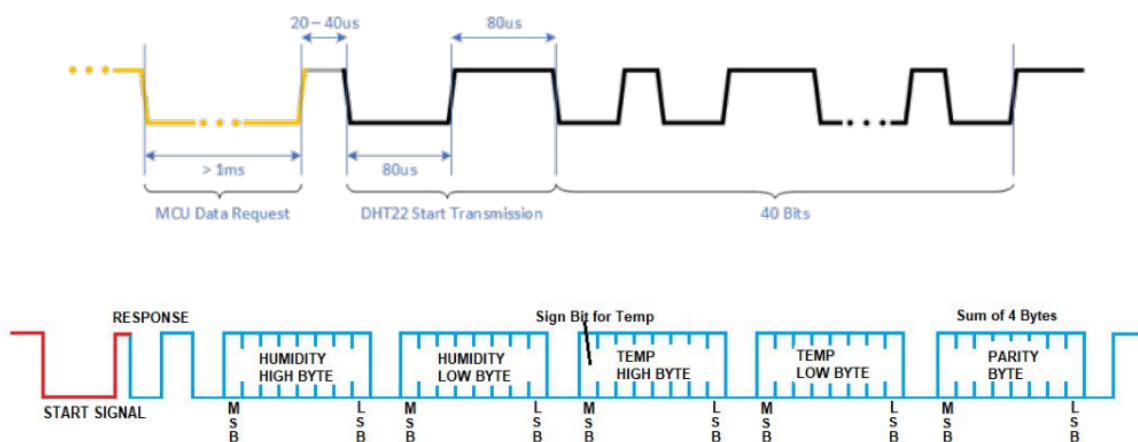


Figure 6: DHT22 sensor connected to a microcontroller

And the DHT22 sensor specification as follows:

- Dimensions: 15.3mm (0.6") length x 7.8mm (0.3") width x 25.3mm (1.0") height.
Weight: 2.4g (0.08oz) [24]
- Operating Voltage: 3.5V to 5.5V
- Operating current: 0.3mA
- Output: Serial data
- Resolution:
 - Temperature: 16-bit
 - Humidity: 16-bit
 - Sum: 8-bit
- Accuracy: $\pm 0.5^{\circ}\text{C}$ and $\pm 1\%$

From the specification, there are 40 bits that are somehow needed to convert to temperature and humidity. The DHT22 timing diagrams below show how these 40 bits are transmitted.



Single-bus data format is used for communication and synchronization between MCU and DHT22 sensor. One communication process is about 4ms. Data consists of decimal and

integral parts. Complete data transmission is 40-bit, and the sensor sends higher data bit first. To calculate temperature and humidity from the serial data, we simply need to add the first two bytes and the result is the humidity, we add the second two bytes and the result is the temperature. As shown below:

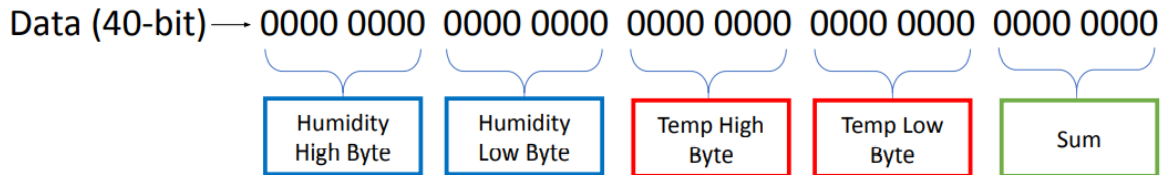


Figure 7: Example of a complete data transmission

So humidity will equal to $= (\text{Humidity High Byte} + \text{Humidity Low Byte}) / 10$, and temperature will equal to $= (\text{Temp High Byte} + \text{Temp Low Byte}) / 10$. And for the other sensors, we are supposed to read its specification first and then program to read them.

Detailed design

The following system architecture describes the structure and behavior of our smart hive design:

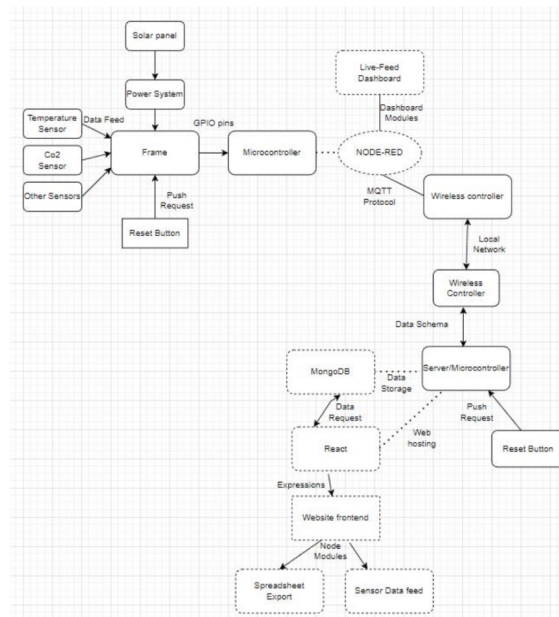


Figure 8: Design of Overall System

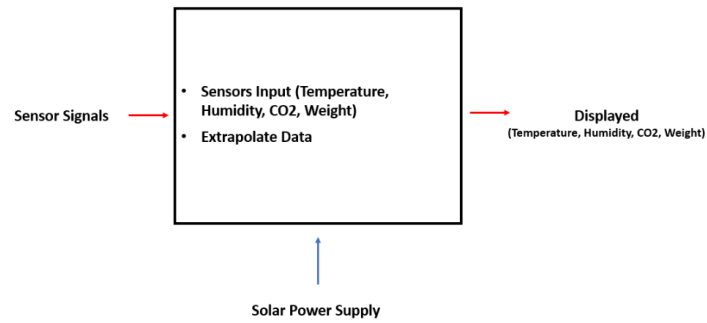


Figure 9: Level 0 Function Design

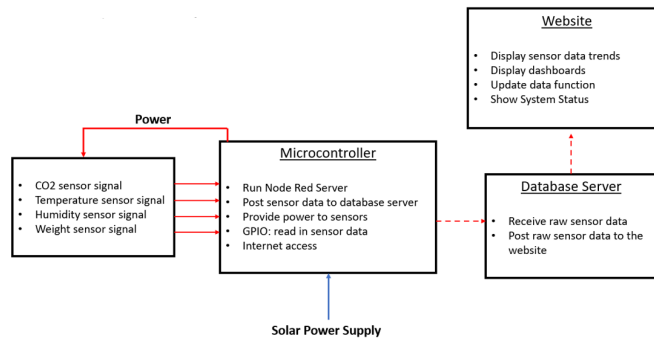


Figure 10: Level 1 Function Design

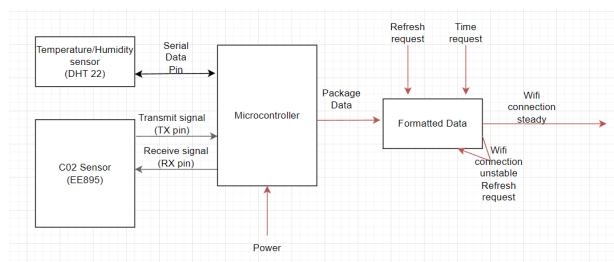


Figure 11: Level 2 Function Design

B. Hardware

The DHT22 as well as the DHT-11 have nearly identical footprints and have been implemented in a “Smart frame lite” like the design as seen below. Due to our smart frame utilizing several different designs to cut cost and redundancy, we have decided to make two different frames. The smaller design is seen below in the symbolic form:

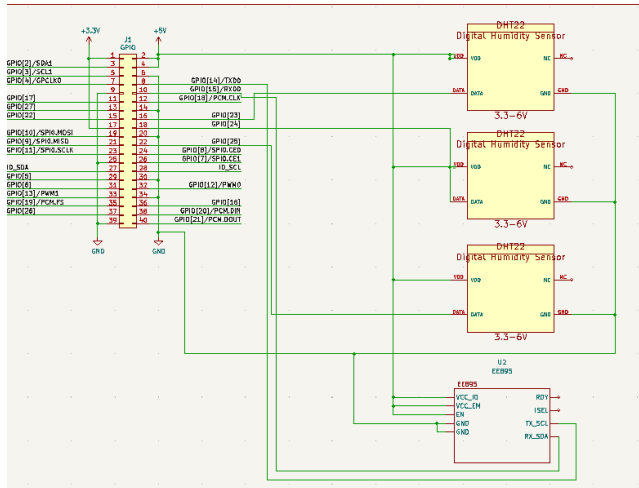


Figure 12: Frame Schematic

The following shows how our design is derived from the symbolic reference and is a non-routed design for the DHT sensor to the raspberry pi. The DHT sensor is required to measure temperature, but the importance of the choice of this sensor is not high as the data schema formed for this design can implement any sensor of choice or additional sensors as long as they are formatted correctly.

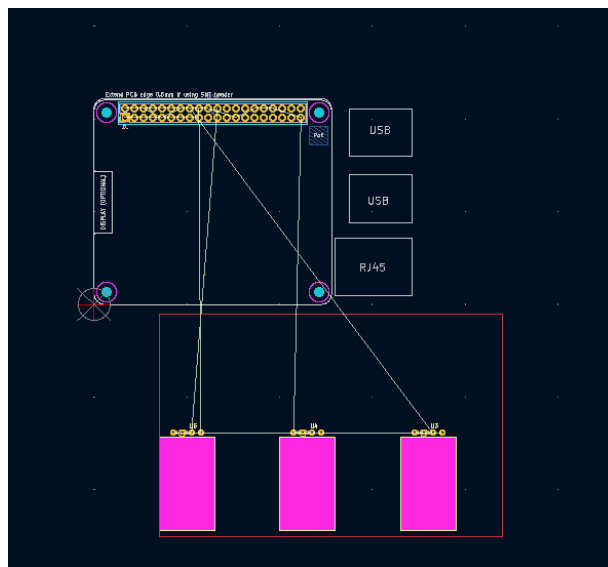


Figure 13: DHT Sensor

One of the problems we had with our sensor selection was the lack of a footprint for our CO2 sensors. So, we have implemented a custom breakout board in our design in order to account for it missing.

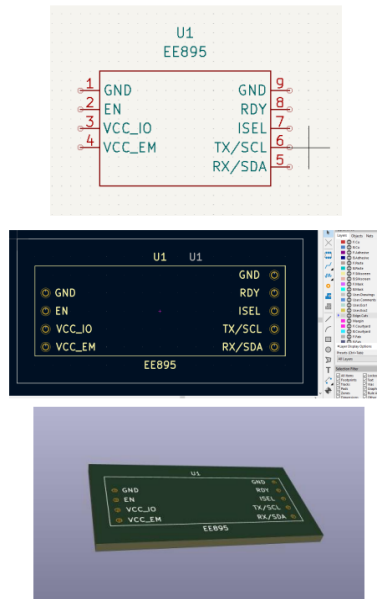


Figure 14: EE895 KiCad Integration

Decision Matrix of Components with Justification

We derived a shortlist for our sensors, needing to measure temperature, humidity, and Co2 levels. Weight and illumination were specified as bonuses, but not required for our client. We included decision matrices for them as well. We used five factors to determine the suitability of each sensor for our implementation (cost, durability, size, digital/analog, and accuracy). We assigned different weights for each of them according to how important each factor was for each sensor in our design, with higher scores pertaining to more importance. Higher scores in each category reflect more preference for use in our design.

- Cost- Preferably lower cost corresponding to a higher rating
- Durability- Important for interaction with bees/material in the hive, a higher score indicates more durability due to material/sensor layout/hardware
- Size- Not as important, but preferably smaller size to fit on smart frame in the hive
- Digital/Analog- Both can be implemented, but preferably digital for Raspberry PI (Digital GPIO Pins)
- Accuracy- Very important, including measurements taken per second to send data to the server/website. Datasheets observed.


1. Temperature Sensor:

Our best candidate is the DHT22 because of its relatively cheap cost compared to the other sensors. It also appears more durable than the other sensors including the


DS18B20 which contained a long, thin wire [5]. The sensor must be durable to withstand the bees and different materials/chemicals in the hive. The DHT22 was also more accurate compared to the DHT11 and had a lower sampling rate, and was digital as compared to analog for the LM35. The datasheets for all four parts that include dimensions and accuracy are listed in the references section [3][5][7][8].

	Weight ->	9	10	5	2	10	
Part Number:		Cost	Durability	Size	Digital/Analog	Accuracy	Final Rank
DHT22		9	6	9	10	9	43
DS18B20		10	3	7	10	8	38
DHT11		9	6	9	10	7	41
LM35		6	4	9	5	10	34

Table 1: Temperature Sensor Decision Matrix



DHT11



DHT22

Operating Voltage	3 to 5V	3 to 5V
Max Operating Current	2.5mA max	2.5mA max
Temperature Range	0-50°C / ± 2°C	-40 to 80°C / ± 0.5°C
Humidity Range	20-80% / 5%	0-100% / 2-5%
Sampling Rate	1 Hz (reading every second)	0.5 Hz (reading every 2 seconds)
Advantage	low cost	More Accurate

Figure 15: Temperature Sensors Comparison

2. Humidity Sensor:

We chose the DHT22 sensor to measure humidity in our design. The cost savings were the main reason as we are using this sensor to measure temperature as well. Additionally, accuracy was the most important factor for us and the DHT22 fulfilled this criterion the best. The datasheets for all four parts that include dimensions and accuracy are listed in the references section [7][8][9][16].

	Weight ->	9	10	5	2	10	
Part Number:		Cost	Durability	Size	Digital/Analog	Accuracy	Final Rank
BME280		5	6	9	10	7	37
SHT40		6	7	10	10	8	41
DHT22		9	6	9	10	9	43
DHT11		9	6	9	10	8	42

Table 2: Humidity Sensor Decision Matrix

3. Weight Sensor (Additional):

We chose the HX711 weight sensor for our design due to its relatively low cost, ability to connect multiple of the same HX711 sensors together to measure weight, and improved accuracy over the other sensors. The datasheets for all three parts that include dimensions and accuracy are listed in the references section [1][4][12].

	Weight >	9	10	5	2	10	
Part Number:		Cost	Durability	Size	Digital/Analog	Accuracy	Final Rank
HX711		7	7	9	5	10	38
H26R0		7	5	10	5	8	35
MF02A-N-221-A01		8	3	8	0	8	27

Table 3: Weight Sensor Decision Matrix

4. CO2 Sensor:

We chose the EE895 sensor for our design because of its improved accuracy over the other sensors and increased durability as the sensor contains a covering. The trade-off was the cost as this sensor is more expensive than the others. The datasheets for all four parts that include dimensions and accuracy are listed in the references section [2][6][10][13]. However, we are currently testing with the SCD30 due to its cheaper value and availability for orders.

	Weight >	9	10	5	2	10	
Part Number:		Cost	Durability	Size	Digital/Analog	Accuracy	Final Rank
K30		4	5	7	10	8	34
SCD30		6	5	7	10	8	36
EE895		5	6	8	10	9	38
CCS811		7	4	8	5	7	31

Table 4: CO2 Sensor Decision Matrix

5. Illumination Sensor (Additional):

We chose the LM393 light sensor because of its cheaper cost compared to the UUGEAR LSM sensor. The sensor can also run for longer than the others and is

small, being able to fit in a smart frame inside the hive well. The datasheets/websites for all three parts that include dimensions and accuracy are listed in the references section [14][17].

	Weight >	9	10	5	2	10	
Part Number:		Cost	Durability	Size	Digital/Analog	Accuracy	Final Rank
LM393		10	6	8	10	8	42
UUGEAR LSM		9	6	8	10	8	41
KY-018		9	3	9	10	8	39

Table 5: Illumination Sensor Decision Matrix

Website Implementation:

The database implementation unlike other designs has the capability of creating a custom data schema that can adjust as needed. For our design, it looks as such:

```

{
  "TIMESTAMP": {
    "HIVE NUMBER": {
      "SENSOR": {
        {
          "data": 15
        }
      }
    }
  }
}

```

Figure 16: Website Implementation

This implementation allows for multiple (2) frames, temperature and humidity sensors and additional room for additional weight and illumination sensors, and .csv files to be read for a month long period. This approach has allowed us to implement newer sensors if need be into the design. For example, if an illumination sensor is required later, it can be added to the node-red sensor and therefore added to the components list. Our Node-red nodes will be open source and available on GitHub for future students to use and update. Users will be able to select which sensor's data they want to create plots for, and download data for different time ranges to save for future use.

Javascript React is used with HTML5-based components as well as JSX. This allows us to render charts dynamically based on API calls of the sensors without having to worry

about previous data or sensors which have been added after the initial implementation was created.

The raspberry pi was chosen as our board due to its small form factor, low energy requirements, and flexibility across our network to act as our database and general infrastructure microcontroller. The low cost allowed us to utilize this component in addition to its high amount of documentation to ensure further future-proofing.

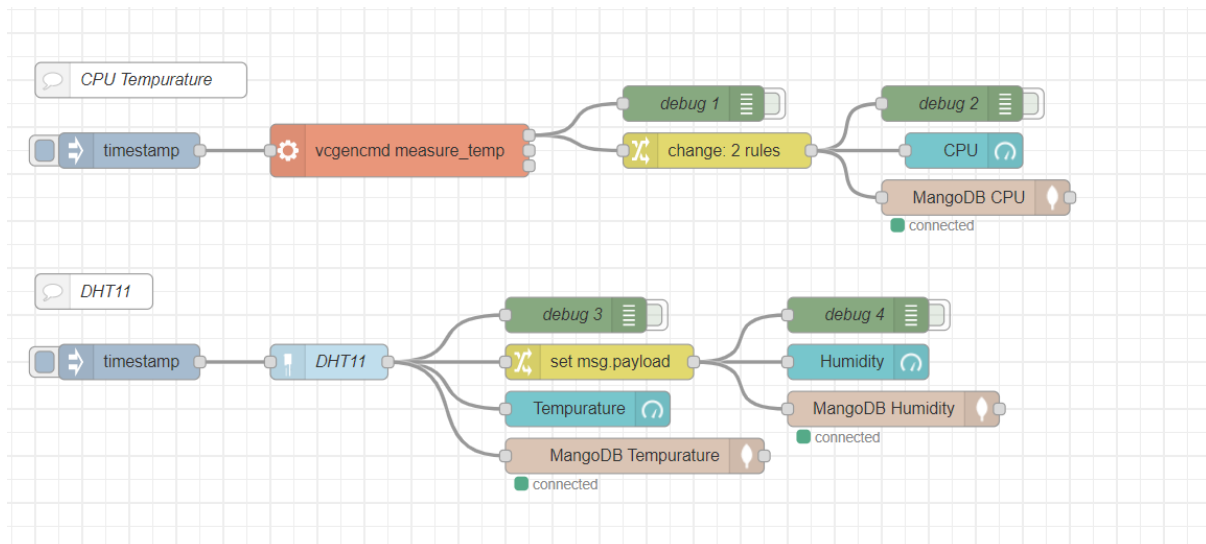


Figure 17: Node-Red Design

Node-red allows one to implement an internet of things approach to networks with minimal troubleshooting. We will be using node-red to gather data from our sensors in a .csv format and send this data to the server. The flow has nodes, and each node we can program to achieve the desired outcome. Using node-red we will have a live feed dashboard as shown in the architecture. It will be used to establish a connection to the database which then sends the data to the website. MongoDB is the database chosen because it is well-suited to node-red. They both use JavaScript to allow for clean mapping.

Schedule and Milestones

Over the course of this semester we have made exceptional progress with our Smart Hive project. Utilizing a top-down approach we were able to effectively break down the main components our project required and gain an understanding of what is required in each sub-system which will all ultimately work coherently to deliver our end product. Throughout the semester, we made use of a Gantt chart to manage our tasks related to the project. The key experiments we have completed but are not limited to, the following:

- NODE-RED GPIO Connectivity
- Optimize MERN stack running concurrently on the same server
- MongoDB postman unit tests
- Rechart.js/Software flow test
- Schematic Footprints
- Temperature and humidity sensor sample gathering,
- Various component dimension gathering

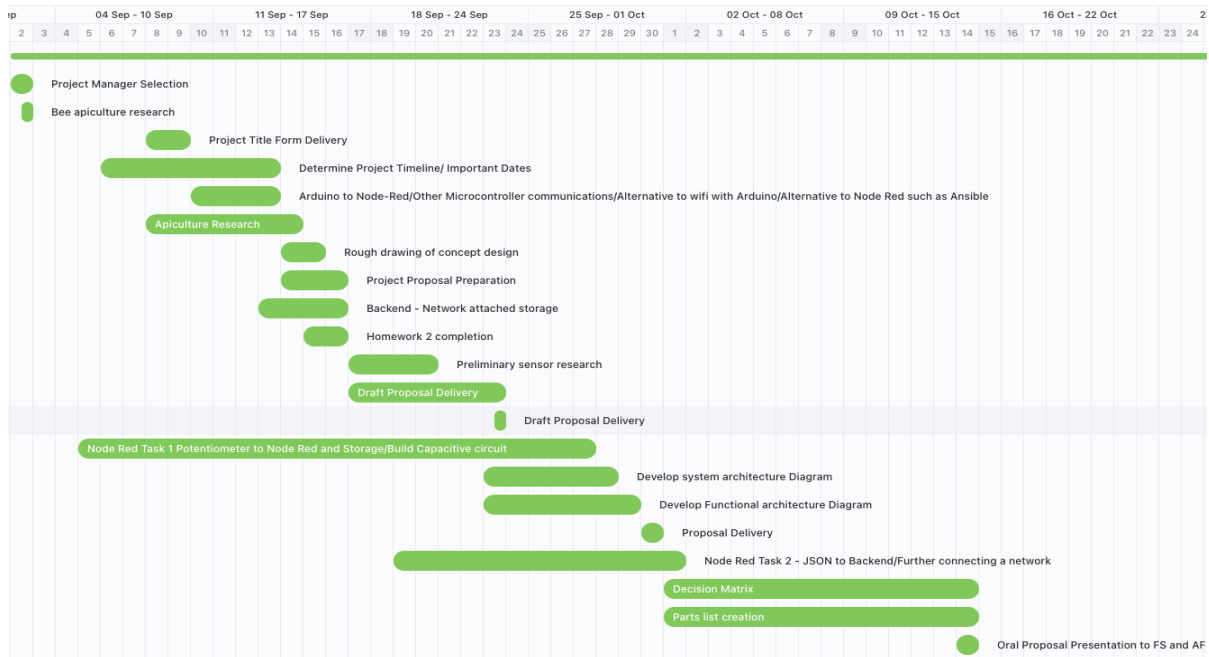


Figure 18: Gantt Chart for SmartHive Fall 2022 Semester Part 1

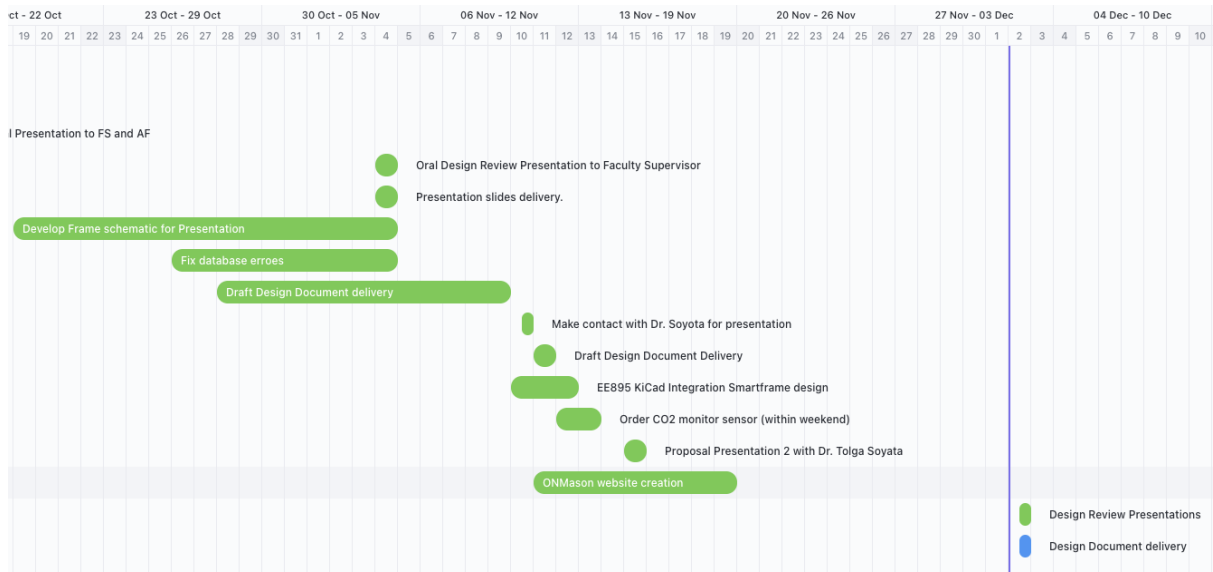


Figure 19: Gantt Chart for SmartHive Fall 2022 Semester Part 2
 *Vertical blue line indicates the current day

As we continue to progress with the design of our Smart Hive, we have created a timeline of important tasks to be completed throughout the upcoming Spring semester. Next semester, our efforts will be geared towards the integration of components, refinement of software, prototyping, and an extended period of time for data collection and display. Through iterative testing, we will be able to complete the requirements described by the customer. In addition to the Gantt chart, we have also included a plan in a list.

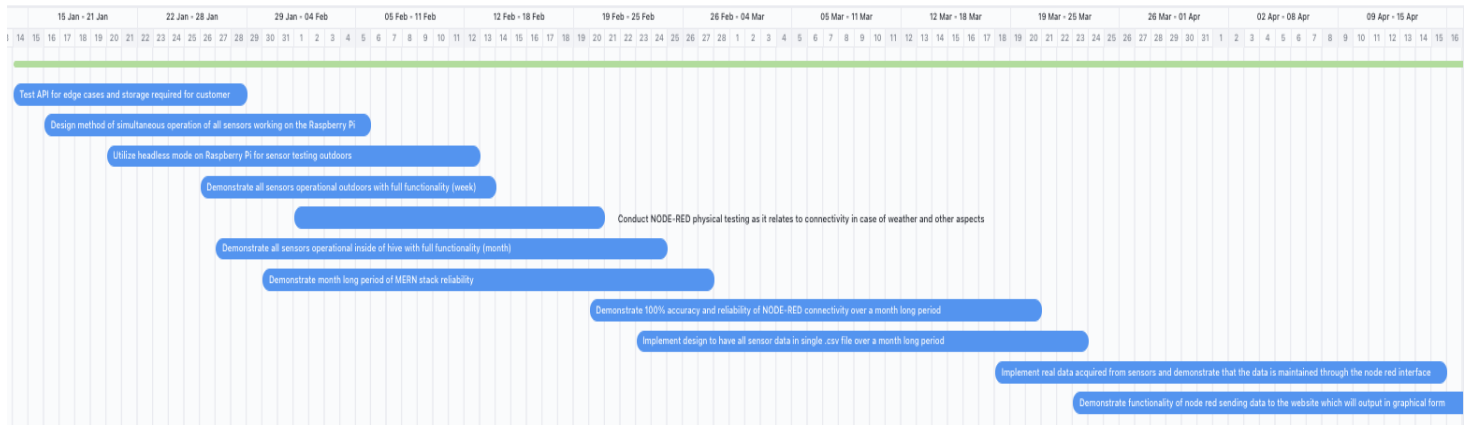


Figure 20: Gantt Chart for Spring 2023

Phase 2 - Spring 2023 Semester  + NEW TASK












TO DO	11 TASKS	ASSIGNEE	DATE STARTED	DUE DATE
■	Test API for edge cases and storage required for customer		1/14/23	1/28/23
■	Demonstrate month long period of MERN stack reliability		1/30/23	2/27/23
■	Conduct NODE-RED physical testing as it relates to connectivity in case of weather and other aspects		2/1/23	2/20/23
■	Demonstrate 100% accuracy and reliability of NODE-RED connectivity over a month long period		2/20/23	3/20/23
■	Design method of simultaneous operation of all sensors working on the Raspberry Pi		1/16/23	2/5/23
■	Utilize headless mode on Raspberry Pi for sensor testing outdoors		1/20/23	2/12/23
■	Demonstrate all sensors operational outdoors with full functionality (week)		1/26/23	2/13/23
■	Demonstrate all sensors operational inside of hive with full functionality (month)		1/27/23	2/24/23
■	Implement design to have all sensor data in single .csv file over a month long period		2/23/23	3/23/23
■	Implement real data acquired from sensors and demonstrate that the data is maintained through the node red interface		3/18/23	4/15/23
■	Demonstrate functionality of node red sending data to the website which will output in graphical form		3/23/23	4/17/23

Figure 21: List View of Tasks for Spring 2023

References:

Note: Much of the additional information requested for the proposal document was required for design document as well, so it was used heavily.

[1] “39.6mm square active area application - mouser electronics.” [Online]. Available: https://www.mouser.com/datasheet/2/13/MF02A__c3_a2_c2_96_c2_a1_A01-2634350.pdf. [Accessed: 10-Nov-2022].

[2] “ams Datasheet CCS811 Ultra-Low Power Digital Gas Sensor for Monitoring Indoor Air Quality.” [Online]. Available: https://cdn.sparkfun.com/assets/learn_tutorials/1/4/3/CCS811_Datasheet-DS000459.pdf

[3] “Analog | Embedded Processing | Semiconductor Company | ti.com.” [Online]. Available: <https://www.ti.com/lit/ds/symlink/lm35.pdf>. [Accessed: 10-Nov-2022].

[4] Avia Semiconductor, “AVIA SEMICONDUCTOR 24-Bit Analog-to-Digital Converter (ADC) for Weigh Scales DESCRIPTION,” 2009. [Online]. Available: https://cdn.sparkfun.com/datasheets/Sensors/ForceFlex/hx711_english.pdf

[5] “Click DS18B20 proramale resoltion 1-wire diital thermometer.” [Online]. Available: <https://datasheets.maximintegrated.com/en/ds/DS18B20.pdf>. [Accessed: 10-Nov-2022].

[6] “Datasheet Sensirion SCD30 Sensor Module,” 2020. Accessed: Nov. 10, 2022. [Online]. Available: https://sensirion.com/media/documents/4EAF6AF8/61652C3C/Sensirion_CO2_Sensors_SCD30_Datasheet.pdf

[7] “Digital-output relative humidity & temperature sensor/module DHT22 ...” [Online]. Available: <https://www.sparkfun.com/datasheets/Sensors/Temperature/DHT22.pdf>. [Accessed: 10-Nov-2022].

[8] “Electronic Components Distributor - Mouser Electronics.” [Online]. Available: <https://www.mouser.com/datasheet/2/758/DHT11-Technical-Data-Sheet-Translated-Version-1143054.pdf>. [Accessed: 10-Nov-2022].

[9] “Itbrainpower.net.” [Online]. Available: <https://itbrainpower.net/downloadables/BST-BME280-DS002-1509607.pdf>. [Accessed: 10-Nov-2022].

[10] “K30 pdf, K30 Description, K30 Datasheet, K30 view ::: ALLDATASHEET *pdf1.alldatasheet.com*. <https://pdf1.alldatasheet.com/datasheet-pdf/view/611300/SPSEMI/K30.html> (accessed Nov. 10, 2022).

[11] K. S. Yeo, M. C. Chian, T. C. Wee Ng, and D. A. Tuan, “Internet of things: Trends, challenges and applications,” 2014 International Symposium on Integrated Circuits (ISIC), 2014.

[12] “Load Cell Sensor (H26R0x),” *Hexabitz*, Jun. 30, 2019. <https://hexabitz.com/product/load-cell-sensor-h26r0x/> (accessed Nov. 10, 2022).

[13] “Miniature Sensor Module for CO 2 Temperature and Barometric Pressure EE895.” Accessed: Nov. 10, 2022. [Online]. Available: https://www.epluse.com/fileadmin/data/product/ee895/datasheet_EE895.pdf

[14] M. Zolnierczyk, “Light sensor modules KY-018 and LM393 (3 and 4 pin) for a Raspberry PI or Arduino,” *NotEnoughTech*, Aug. 21, 2016.

<https://notenoughtech.com/raspberry-pi/light-sensor-lm393-ky018/> (accessed Nov. 10, 2022).

[15] N. Kalaburgi, "Working of DHT sensor - dht11 and DHT22," *NerdyElectronics*, 27-Jan-2021. [Online]. Available: <https://nerdyelectronics.com/working-of-dht-sensor-dht11-and-dht22/>. [Accessed: 10-Nov-2022].

[16] "SHT4x 4 th Generation, High-Accuracy, Ultra-Low-Power, 16-bit Relative Humidity and Temperature Sensor Platform." Accessed: Nov. 10, 2022. [Online]. Available: <https://download.mikroe.com/documents/datasheets/SHT40%20Datasheet.pdf>

[17] "UUGear Light Sensor Module (4-Wire, with both Digital and Analog Output) | UUGear." <https://www.uugear.com/product/uugear-light-sensor-module-4-wire-with-both-digital-and-analog-output/> (accessed Nov. 10, 2022).

[18] J. Cazier, "Peering into the future a path to the genius hive," *Peering into the Future a Path to the Genius Hive*, 26-Mar-2018. [Online]. Available: <https://www.beeeculture.com/peering-into-the-future-a-path-to-the-genius-hive/>. [Accessed: 12-Nov-2022].

[19] A. lofaro, "Smart hive 2.0 installed at GMU and USDA," *Lofaro Labs Robotics*. [Online]. Available: <https://lofarolabs.org/2022/02/14/smart-hive-2-0-installed-at-gmu-and-usda/>. [Accessed: 12-Nov-2022].

[20] *Smarthive.lofaro.net*. [Online]. Available: <https://www.smarthive.lofaro.net/>. [Accessed: 12-Nov-2022].

[21] F. Linton, "Welcome to Colonymonitoring.com!," *colonymonitoring.com*. [Online]. Available: <https://colonymonitoring.com/>. [Accessed: 12-Nov-2022].

[22] R. F, "Humidity in the Hive," *Arnia*, 26-Apr-2021. [Online]. Available: <https://www.arnia.co/post/humidity-in-the-hive>. [Accessed: 12-Nov-2022].

[23] "Smart hives: A radical rethink of Beekeeping," *The Best Bees Company*, 17-Mar-2022. [Online]. Available: <https://bestbees.com/2021/07/27/smart-hives/>. [Accessed: 30-Sep-2022].

[24] "Addicore DHT22 (AM2302) temperature and humidity sensor," *www.addicore.com*. [Online]. Available: [https://www.addicore.com/DHT22-Temperature-and-Humidity-Sensor-p/182.htm#:~:text=Dimensions%20\(excluding%20pins\)%3A%2015.3,%3A%202.4g%20\(0.08oz\)](https://www.addicore.com/DHT22-Temperature-and-Humidity-Sensor-p/182.htm#:~:text=Dimensions%20(excluding%20pins)%3A%2015.3,%3A%202.4g%20(0.08oz).). [Accessed: 02-Dec-2022].