Instrumented Beehive

Manali Palwankar, CSE, Max Aukshunas, CSE

***Abstract*—There are many things on this planet that we must take care of, but bees are one of the most important. Bees play an important role in the pollination of the world's plants that us humans consume. We have beekeepers to monitor them, but with the astronomical amount of bees and relatively small amount beekeepers, this proves to be a difficult task. Our system allows beekeepers to monitor the most important conditions, temperature and humidity, in their already existing beehives. We have a dummy beehive frame equipped with sensors and a microcontroller which beekeepers will insert into their existing hives. The beekeepers will be able to analyze the conditions of their hives using our web interface as well as receive real-time alerts when abnormalities occur.**

I. INTRODUCTION

During the recent decade, bees' population has been declining by an alarming rate, up to 30 percent per year, with a total loss of domesticated honey bee hives in the United States worth an estimated $2 billion. Major causes of the decline include insecticides, pesticides, habitat loss and general decline in their diet. There has been a major habitat loss due to climate change and monocultural agriculture practices. This has greatly affected the world as bees play a major role in pollination. It has affected farming and major food production [1].

The EPA has been working to save honey bees and other pollinators from pesticide exposure including major policies. However, little has been done about the habitat loss as it requires a lot of resources and time. Beekeeping is commercialized on a large scale and hence Beekeepers can provide enough habitat as they can also earn profit by farming honey.

Our product is aimed to provide a cheap solution to providing a safe and quality environment for bees which can monitor their health and can also be used to provide significant data for research on bee heath.

A major thing that sets us apart from competitors[5] is that our system is portable. That is, it can be moved within a beehive as well as hive to hive. Our competitors have created a completely new hive that beekeepers must buy and replace their old hives with. That plan which involves replacing all old hives with new hives is infeasible because there are over 80 million hives kept globally [2].

Our system will be very portable in that it can be moved within the hive to account for the Brood[see footnote #1]. This is the most important location in the hive to record temperature and the location varies over time. The location also varies from hive to hive so it's essential that the beekeeper can put our system where ever they see fit. As the default, our sensors will collect readings every 15 minutes because we don't mind if there are temperature or humidity changes for periods of less than 15 minutes. Bees can withstand those short changes but those changes start to be important once they are constant for at least 15 minutes. If the beekeeper wishes to change that interval, they can easily do so on the website. Each reading must have the date and time taken with it in order for beekeepers to perform the correct analysis of their hives. These readings are uploaded to the server every time they are recorded. Once at the server, the beekeeper will be able to see all the data for each of their hives. They will be able to view graphs of how the temperature and humidity has varied over time. Based on those graphs, the beekeeper will be able to see if the bees are constantly living in the proper conditions. If there are abnormalities at a certain time and hive, the beekeeper will notice this and try to improve living conditions at the hive. Our system uploads data to the server as long as the hive is in WiFi range. The required power for our system is generated at the hive via solar panels. In order to account for lack of sun and inclement weather, we will have a rechargeable battery that will power the system. This rechargeable battery will be charged by the solar panels and have enough capacity to power the system for 13 days without being charged by the solar panels. Our system will be able to operate in all weather conditions and will be working year round. We have encased our system with epoxy to protect our system from the harsh elements of winter in New England. Table 1 shows the specifications for our system.

M. Aukshunas from Lowell, MA (e-mail: jaukshunas@umass.edu)
M. Palwankar from Mumbai, India (e-mail: mpalwankar@umass.edu)

[1] A nest within Beehive where Queen Bee lays eggs

Table 1: Requirements and Specifications

Parameter	**Value**
Sensor Accuracy	+/- 1 °F
Alerts	Temperature change +/- 5 °F Low battery
Data Collection	Every 15 minutes
Data Transfer	Every 15 minutes
Data Accessibility	Remote
Renewable Energy Source	Solar
Power Limitations	13 Days
Operability	All weather conditions, all year
Range	Within 300 feet of WiFi

I. Design

A. *Overview*

This technology should solve our problem because it provides a simple, cost-effective way for beekeepers to monitor their hives' well-being. We considered making a completely new hive with sensors built into it but this would be very expensive for beekeepers. One of them would be expensive, and if they wanted to replace all of their hives it would cost a fortune. Our system costs less than competitors ($470 vs $550), but most the cost comes from the power supply; see Appendix B. We also considered using cellular data to account for lack of WiFi in remote locations but that would mean working with cellular networks such as Verizon and AT&T to get cellular data on each individual system. We could have also used LoRa and other long range communication methods but these would have been more complicated and more expensive than WiFi. Figure 1 shows our final block diagram which consists of 6 components.

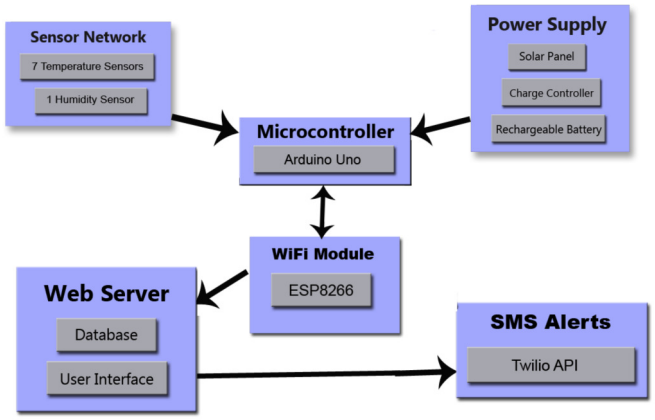


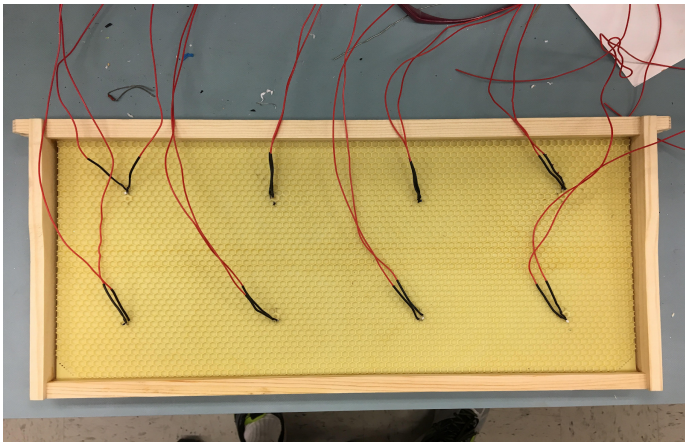
Figure 1: Block Diagram

The Sensor Network will be placed on a dummy frame and it's what is going to take all the readings. The Microcontroller will tell the sensors to read every 15 minutes and it will store the readings as well as date and time stamp them. The Microcontroller will connect to the WiFi Module which will then upload the data to the Web Server's database. The Power Supply will generate power as well as provide constant power to the microcontroller. Using Twilio's API, we will be able to send real-time text message alerts to beekeepers to inform them of irregularities.

B. *Sensor Network*

The sensor network (Refer to figure 2) will act as the primary resource of our data as it will read temperature and humidity. The sensor network consists of 5 thermistors that record temperature, 1 waterproof sensor that records temperature, and a temperature and humidity sensor for the housing. In total, there are 7 sensors that record temperature and 1 that records humidity. The original plan was to lay out 8 thermistors on the frame, but because of the limited number of analog inputs on the Arduino, we could only use 5. Once the recordings have been taken in from the sensors, the data is then sent to the server via WiFi for further analysis. Data will be read every 15 minutes or how ever long the beekeeper chooses.

With expertise of our beekeeper sponsor and experiments using different temperature sensors, we decided to use 8 temperature sensors on a dummy frame inside the hive (only 5 can be used). The sensor network is shown in in Figure 2.



Figue 2: Sensor network

We performed tests on three types of sensors which include thermistors, waterproof sensors, and the actual temperature via a thermometer. Our goal for the experiment was to get accurate and consistent readings. We used a digital thermometer for reference. Few of our experimental readings are shown below in figure 3.

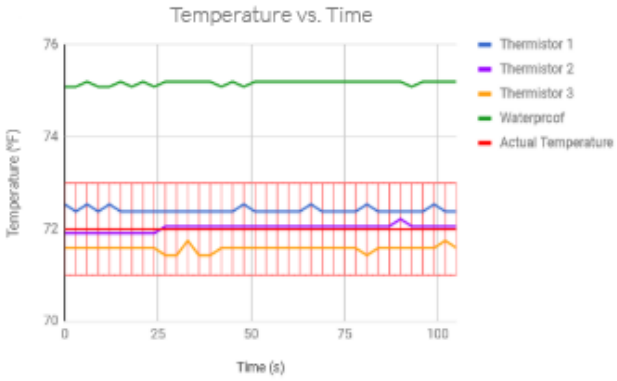


Figure 3: Temperature vs Time testing

C. *Microcontroller*

For initial testing purposes, we used Arduino Uno R3 SMD. It is driven by ATMega328 Chip which contains a 1KB Data storage and 32KB flash memory[13]. Due to lack of storage space on the chip, we are looking at options of external storage that can store data at least worth 15 days in the case of failure in data transfer. The lack of storage also prevented us from adding more features into the Arduino code like being able to change the WiFi from the website.

We planned to design a circuit layout using the ATMega 2560 chip[13] and use it to make a PCB for our final microcontroller. Due to lack of time, we were unable to get the PCB working properly or even get a new Arduino with more storage space. Our microcontroller and circuitry was placed outside the hive in the housing which is attached to the dummy frame. The wires from the sensors run through a slot on top of the frame that then leads to the housing. The design can be explained in the figure 4 below.

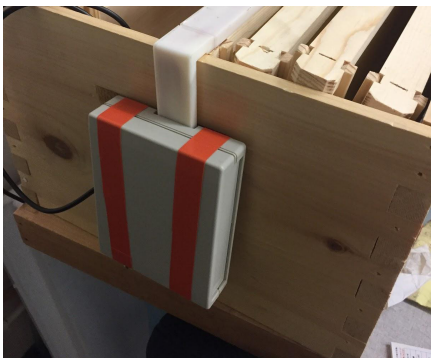


Figure 4: Design for Housing

D. *Power Supply*

The power supply is one of the major components of our project. We are going to use solar energy to drive the whole system. The prototype will consist of one 50W solar panel, 55Ah 13V rechargeable battery, a switching voltage regulator circuit[12], and a solar charge controller[11].

We used a power supply and with 5V of power, the Arduino drew 290 mA of current. At 5V, the total power consumption of the system will be 1.45W considering the microcontroller draws 290mA of current. Assuming, the system will run constantly, it will require 452.3 WH. Because our battery is 13V, the system will need 34AH to run for 13 days. However, battery does not discharge all the way to 0V. If we assume that the state of charge is approximately linear, and considering the power cutoff after the battery reaches 8V, the system will use up the battery in 13 days.

The output from the charge controller is 13V and is further converted to 5V using a switching voltage regulator LM2596[12]. This regulator is 80% efficient and has minimal power loss,

The charge controller is a major component of our solar system. It regulates the charging and power suply between the Solar panel, microcontroller and the Rechargeable battery. The Charge controller stops outputting power once the Battery reaches 8V.

Table 2: Power cost analysis

Components	**cost($)**	**Specifications**
Solar Panel[19]	105	50W, outputs 12V-22V
Charge Controller[11]	20	20A, 12/24V
Rechargeable Battery[18]	118	55Ah, 13V

D.1. *Battery Percentage*

State of charge of the battery can be determined by the Voltage of the battery has accross its nodes. This is the easiest way to determine the State of Charge, although it is not the most accurate. Whether the system is under load or not affects the voltage across the nodes but we decided to stick to the estimated state of charge.

We read the voltage across the terminals of the battery and convert it to a voltage reading and convert it to state of charge.

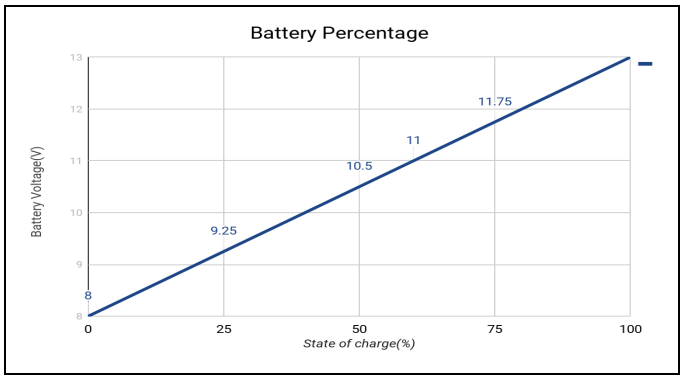


Figure 4: Battery percentage

Because our charge controller stops outputting voltage after the battery reaches 8V, we set our 0% as 8V. Our 25% is 9.25V. Our system sends sms alerts in case the battery goes below 25%. Figure 4 shows approximation for our system.

E. *WiFi Module*

The ESP8266 WiFi Module [7] first connects to WiFi using the given name and password. Since UMass's school-wide WiFi, eduroam, is a WPA2 encrypted network [14], we had to set up a hotspot using one of our phones. The data from the microcontroller will be sent to the WiFi module using AT commands, then that will create a TCP connection with our server and send HTTP commands containing the data. While we have learned similar things in ECE 374 Computer Networks and the Internet, most of these commands and algorithms have been taken from the WiFi module data sheet [3]. The WiFi module sends messages back to our microcontroller and if we receive the right messages (normally an "OK"), we know it's working correctly. Every time the Arduino loops back through the same set of code, it checks to make sure it is connected to the WiFi. If it wasn't connected to the WiFi and we had available storage space on the Arduino, we would have stored the data until it re-connected to the WiFi.

F. *Web Server*

We have rented a server from a hosting website (HostWinds [6]) that has a MySQL [15] database to contain the all the data that ever gets collected. With our current hosting plan, we have unlimited database storage which provides us with the ability to scale our product if we ever wished to. The database contains all the readings along with the dates/times as well as the beekeeper's settings for the delay, phone number, and alert thresholds. The database contains all the data which can then be seen on the user interface of our website; see Appendix A. All of the files for uploading/outputting data and the rest of the functionality in the back-end were written in PHP. The user interface was designed using HTML, CSS, and JavaScript and has a bee theme to it (black and yellow colors). We learned how to use JavaScript via YouTube videos and coding tutorial websites like W3Schools [16]. To make the plots of the data, we had to use a JQuery library called jqPlot [8]. This was also a new technology to us so we used the same methods of learning how to use it.

When the beekeeper first visits the website, they will land on the home page, which has a few things.

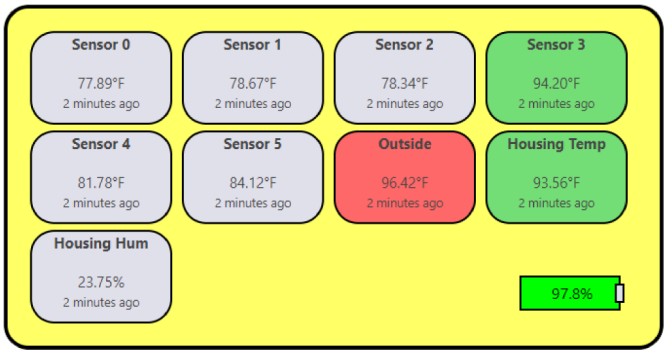


Figure 5: Current readings on home page

It has all the current readings for each sensor and how long ago they have been updated. Each sensor should update at the same time, so if there is a sensor with a different last updated time, we know there is something wrong with the functionality of that sensor. If a temperature is above 95° F, the sensor's background turns red to show that it's currently hot. If a temperature is between 93° F and 95° F, the background turns green to show that its close to being hot. The background stays at its default color, gray, if the temperature is below 93° F. Also, the current battery percentage can be seen in the bottom right corner. The battery percentage as well as the readings are all a part of the data that gets uploaded to the server during each loop. Below this section is the settings section, where the beekeeper can change a few things. They can change the phone number that they wish to get alerts sent to as well as change the threshold for the temperature alert and battery alert. They can also change the status of each alert, turning it on or off. The last thing they can do in the settings section is change the delay of the readings. The current default is every 15

minutes, but if they want to make it shorter or longer, they just simply have to enter in an integer and click the update button. The delay for the readings is controlled by the code on the Arduino, so every time the readings are recorded, the WiFi module creates a TCP connection with the website and passes a GET command (HTTP request method) [17], to get the current delay from the website. This is different from when the data gets uploaded to the website, because in that case, we are sending a POST command [17]. The final thing the beekeeper can do on this page is if they want to see data from the past, they can click on any of the sensors of the top of the page and choose a time period for which they want to see data for. Choosing these options and clicking the "Display Graph" button will bring the beekeeper to a new page where the data gets outputted.

There are a few things the beekeeper can do/see on the output data page. If they wish to see all the individual data points for the specific time period and the selected sensors, all they have to do is click the "Toggle Data Table" button and each data point is outputted. This is useful to the beekeeper if they wish to see what the exact value of a sensor is at a certain date/time. Below that, is the graph of the raw data for each selected sensor.

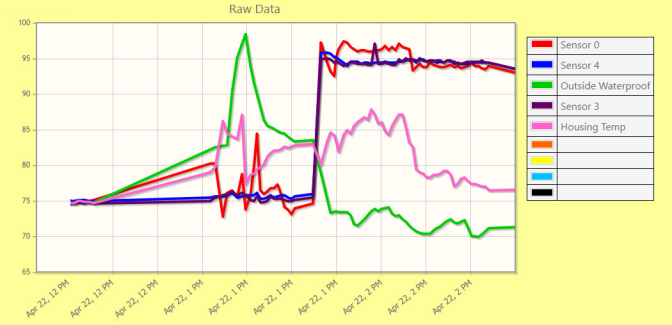


Figure 6: Raw Data Graph

Each sensor on the graph is color coded so the beekeeper can see which sensor is at what value at what time. Sometimes the raw data can be quite choppy and hard to read. For this reason, we created an exponentially smoothed graph that really helps shows the trends of each sensor and is overall a lot easier to look at.

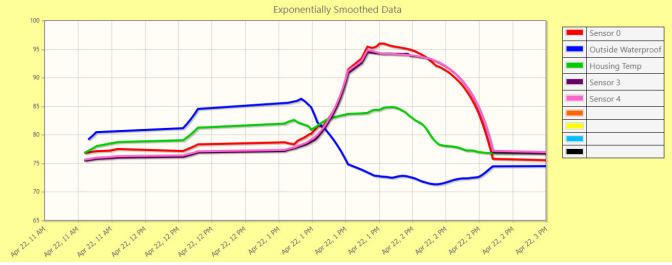


Figure 7: Exponentially Smoothed Graph

With these graphs, the beekeeper can easily analyze the trends over time for each sensor. Below the exponentially smoothed graph is the sensor metrics section. This provides useful statistics for each sensor like the average, maximum, minimum, average change between readings, standard deviation as well as how many readings are multiple standard deviations away. This provides the beekeeper with a quick and easy way to see the most important values for the set time period. The standard deviation part really helps show the beekeeper how much the sensors values vary (variation is normally a red flag).

G. *SMS Alerts*

We have rented a phone number from cloud communications platform, Twilio [10]. We use this phone number to send text messages to the beekeeper when certain conditions are satisfied. Each time we record data and upload it the website, each one of those readings goes through a set of checks. The last 8 readings for a sensor are taken from the database and the average for those are calculated. The value chosen by the beekeeper is then added and subtracted from that average. If the current reading is outside that range, the alert gets triggered and the beekeeper will get a text saying that specific sensor is at the current value. We call this alert the "moving average" alert.

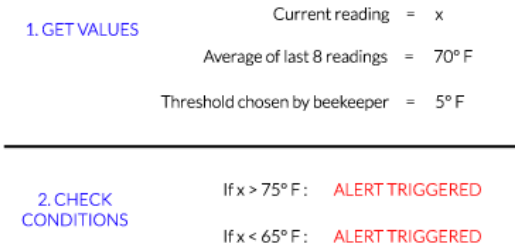


Figure 8. Moving Average Alert Example

This kind of alert isn't very effective in detecting gradual increases or decreases so we also have a "trending" alert. This is calculated by checking the current reading with the previous reading and seeing if it is an increase or decrease. If there is a majority (at least 7 increases or at least 7 decreases) over the last 10 readings, we consider that a trend and an alert will get triggered. Also, we have an alert set up for the battery level. It's default trigger value is 25% so

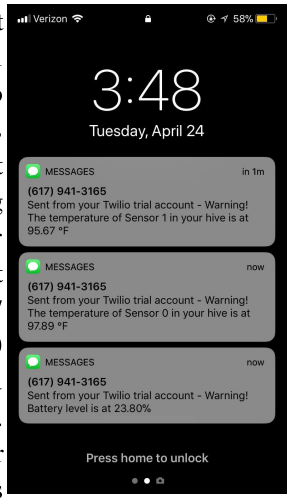


Figure 9: Alert Examples

if the battery level drops below that, the beekeeper will receive a text message. All of these alerts are checked in the same PHP file that gets used when data gets uploaded to the website. We have used Twilio's API to create messages and send them using the phone number they gave us. Again, if the beekeeper wishes to change the values on these alerts or turn them on/off, they can.

II. Project Management

Table 3: Final Project deliverables

Deliverable	Status
Recording temperature, date/time stamped	Completed: Recording/storing temperature readings from 5 sensors.
Data uploaded to server/UI	Completed: Each reading uploaded separately from server's database
Real-time SMS alerts	Completed: Test alert able to be sent to phone
Prototype of power supply	Complete
Battery Percentage	Complete

The final prototype has 7 sensors that record temperature and one humidity sensors which records housing humidity. We also have a circuit which reads the voltage across our battery which is then converted to state of charge. Because of our input limitations, we don't have more temperature sensors which is what we would have liked.

Our temperature sensors are thermistors that take in a voltage reading and deliver that reading to the analog inputs of the Arduino. We converted those voltage levels to temperatures in Fahrenheit. The battery percentage reading is uploaded to the server along with other sensor readings and battery percentage. The data is time stamped and uploaded to the server. These reading points are analyzed for irregularities and sms alerts are sent if the data is above or below the thresholds. Once the readings have been taken in, we have the Arduino communicating with the WiFi module. The Arduino passes commands to the WiFi module first telling it to connect to the WiFi. Instead of using the complicated process of connecting to UMass's campus-wide, WPA2-Enterprise network, we decided to create a hotspot via cell phone that acts like a normal router. Once connected, the WiFi module can create a TCP connection with our website. Once that connection has been established, we send a string containing the HTTP command we want to use (POST [11]), the PHP script, and the values of the data readings. If the transfer is successful, the PHP script should have been executed and we should be able to see the data in the backend of the server. We have multiple tables in the database for each sensor so we can choose which sensor we want to look at the data for. We can then see the data in the user interface by choosing which sensor we want to look at.

The team has overall worked well together. We met twice a week and discussed what we'd accomplished and what our short term as well as long term goals were. We communicated in a group chat regarding meeting up to work on the project for the individual parts. We were in contact with our Sponsors Frank Linton and Brant Cheikes regarding our progress. It helped us to stay on track. Lacking an Electrical Engineering major definitely was a drawback as we took longer than expected to design our PCB. However, working on an interdepartmental project gave us interested insights on how the different components of the project should work together.

III. Conclusion

Our final prototype is a retrofit frame that can be put into any existing beehive. The system records temperature and humidity data and is powered by solar energy. The system is able to withstand inclement weather conditions and can run without sun for 13 days. We were able to test our systems on live bees in Warm Colors Apiary[20] in South Deerfield. We collected valuable data from our visit and helped us verify our specifications and also realize some drawbacks of the system.

We put in hard work to design the sensor layout, Frame and housing design and system layout with the help of Mechanical Engineers on the team. The prototype has been verified as convenient and user friendly by Dan Colon from Warm Colors Apiary[20].

[A] *Further development.*

If we are able to further develop our system, there is room for development in the following areas,

1. Add a camera to hive for security purposes
2. Add a weighing scale to weigh honey
3. Make the connection between our frame and power system retractable.

IV. Appendix

A. http://instrumentedbeehive.website/

B.

Parts	Development	Production (1000)
Sensor Network	$48.57	$27.17
Arduino/WiFi Module	$36.90	$31.21
Solar Panel	$105.00	$105.00
Battery	$118.39	$89.75
Rest of Power System	$109.65	$98.11
Housing	$51.94	$44.23
Total	$470.45	$395.47

V. Acknowledgment

Special thanks to our faculty advisor, Prof. Lixin Gao. We would also like to recognize our evaluators, Profs. William Leonard and Yadi Eslami for feedback that greatly improved our system. We would like to thank Dan Conlon of Warm Colors Apiary for feedback and live testing. We would also like to thank our sponsors Frank Linton and Brant Cheikes.

VI. References

[1] McDonnell, Tim. “Here's Why All the Bees Are Dying.” Mother Jones, 23 June 2017, www.motherjones.com/environment/2015/07/climate-change-killing-bumblebees/
[2] http://www.safechemicalpolicy.org/the-number-of-honeybee-hives-have-increased-globally/
[3] https://www.sparkfun.com/datasheets/Cellular%20Modules/AT_Commands_Reference_Guide_r0.pdf
[4] APSBiker. “Arduino Uno - R3 SMD.” DEV-11224 - SparkFun Electronics, www.sparkfun.com/products/11224.
[5] The Arnia remote hive monitoring system was developed by beekeepers, for beekeepers, http://www.arnia.co.uk/
[6] Hostwinds, https://www.hostwinds.com/shared.php
[7] ESP8266 WiFi Module, https://www.sparkfun.com/products/13678
[8] jqPlot plotting software, http://www.jqplot.com/
[9] Solagris, https://solargis.info/imaps/
[10] Twilio, https://www.twilio.com/
[11] All Powers Charge Controller, http://iallpowers.com/index.php?c=product&id=371
[12] LM2596 Voltage regulator http://www.ti.com/lit/ds/symlink/lm2596.pdf
[13] ATMega 2560,https://cdn.sparkfun.com/datasheets/Components/General%20IC/2549S.pdf
[14] WPA2 Network https://www.webopedia.com/TERM/W/WPA2.html
[15] MySQL database https://www.mysql.com/
[16] W3Schools https://www.w3schools.com/js/default.asp
[17] HTTP Request methods https://www.w3schools.com/tags/ref_httpmethods.asp
[18] Battery https://www.batterysharks.com/Universal-Power-UB12550-45825-p/UB12550_B12-55.htm?gclid=CjwKCAiAweXTBRAhEiwAmb3Xu6qSirzOrMgumBe5qeg6yew7taOqN_O8jOR9I2QNXYIEieBRDh_JMhoCXxQQAvD_BwE
[19] Solar Panel https://www.homedepot.com/p/Ramsond-50-Watt-12-Volt-Monocrystalline-PV-Solar-Panel-SP-50/203423806
[20] http://www.warmcolorsapiary.com/
[21] Battery state of charge https://www.energymatters.com.au/components/battery-voltage-discharge/