

Bilkent University Electrical and Electronics Engineering Department
EEE 494 - Industrial Design Project II
COMMITTEE MEETING 4 REPORT
GROUP A2
BEESMART

KAREL



Project Group Members:

Mert Erkul

Elif Ceren Fitoz

Deniz Can Oruc

Dogan Parlak

Mert Sayar

Arda Sarp Yenicesu

Academic Mentor: Prof. Ezhan Karasan

Company Mentor: Dr. Alper Sarikan

Teaching Assistant: Arda Atalik

Date: 02.07.2020

Karel Electronics: Operates in the visual communication and security systems field, and performs sales and project based applications for the intruder alarm systems market.

Abstract

This project aims to create a virtual environment for beekeepers so that they can monitor the hive conditions and also try to predict the medical conditions of the bees by processing their sounds. The solution uses hardware components such as temperature, moisture, tilt, accelerometer, GPS, GSM and microphones for monitoring the hive conditions with addition to recording and processing sounds present in the hive and transferring them over GSM to a cloud server. After the sounds are processed with edge computing, the information are sent to beekeepers and other third-party users via a phone or web application. We used training and validation data for deciding on the correct learning algorithm to use, wrote individual testing modules for each hardware component to see if that particular component gives expected results prior to sending the data to servers over GSM module. GSM module was tested by sending arbitrary data to servers and visualizing them. There are six milestones designated for this project, hardware design, embedded software design, machine learning algorithm design, Android Application design, Web Application design and Cloud-Application connection design. After all these milestones are completed, as a result, we have successfully acquired a fully functional hardware that is in connection to applications that notifies beekeepers and helps them correctly classify and monitor hive conditions so that they can plan their visitations more precisely and thus be more efficient in terms of honey production and collection overall.

Key Words: Sound Processing, Hives, Machine Learning, Cloud Servers, Embedded Modules, Edge Computing

Contents

- 1 Motivation & Novelty** **2**
 - 1.1 Motivation 2
 - 1.2 Novelty 3

- 2 Design and Performance Requirements** **4**
 - 2.1 Functional Requirements 4
 - 2.1.1 Hardware Requirements 4
 - 2.1.2 Software Requirements 5
 - 2.2 Non-Functional Requirements 6

- 3 Big Picture** **8**

- 4 Methods and Implementation Details** **9**
 - 4.1 Work Breakdown Structure and Project Plan 10
 - 4.1.1 Milestones 10
 - 4.1.2 Work Packages / Tasks 13
 - 4.2 Methods and Progress 16
 - 4.2.1 Progress up to CM3 16
 - 4.2.2 Progress between CM3 and Present 33
 - 4.2.3 Task/Work Packages That Could not be Finished 37
 - 4.2.4 Risk Analysis & Backup Plan 37

- 5 Results, Discussions and Future Directions** **39**
 - 5.1 Results 39
 - 5.1.1 Hardware Integration to Hive 39
 - 5.1.2 Hardware-Cloud-App Connection 39
 - 5.1.3 Sensor Measurements and Machine Learning Prediction 41
 - 5.2 Discussions and Lessons Learned 46
 - 5.3 Future Directions 48

- 6 Equipment List** **49**

- 7 Appendices** **51**
 - 7.1 Appendix A 51
 - 7.2 Appendix B 53
 - 7.3 Appendix C 54
 - 7.4 Appendix D 55
 - 7.5 Appendix E 56
 - 7.6 Appendix F 58

- References** **61**

List of Figures

2	Designed Project Hive	7
3	Big Picture of the Project	8
4	Work Breakdown Structure for Sections -Two Phases	13
5	Several Designed Component Demonstrations	17
6	Designed PCB	17
7	Debugger and hive lid	18
8	PCB Soldering	18
9	Overall Designed Circuitry	19
10	Sensory Components Used	21
11	Results Obtained from K-Means Clustering Unsupervised Learning	22
12	Sample Sounds and Obtained Mel-Frequency Spectrogram Images	23
13	Properties for Transfer Learning Model	24
14	Properties for Edge Computing with TF Lite	24
15	Observations on the Predictions in Overall and Class-wise	26
16	Spectrogram Comparisons	27
17	Mel-Frequency Spectrograms for Hive Conditions	27
18	Screenshots on the Emulator for Application Designed	31
19	Sample Sound Recordings	34
20	Message Structure	35
21	Completed Circuitry on Hive Lid	39
22	Screenshot for Inbox	40
23	Example Temperature Notification	40
24	Example Tilt Notification	41
25	Results when no sound provided	42
26	Results when healthy and full hive recordings provided	42
27	Results when healthy and half-full hive recordings provided	43
28	Results when unhealthy and full hive recordings provided	43
29	Location provided, Karel Electronics, Cyberpark	44
30	Location provided, Turkey Map	44
31	Location provided, Karel Electronics, Cyberpark	45
32	Web Application Graphics	45
33	Android Application Humidity Graphics	46
34	Previously Modified Timeline	51
35	Previously Modified Timeline, Continued	52
36	Last-Modified Timeline	53
37	Sound Data Obtained from Microphone Sensor	54
38	Connection Established to beesmart Server	55
39	Database Configuration Verification	55
40	Modified Android Application Screenshot- Hive Status	56

41	Modified Android Application Screenshot- Hive Data	56
42	Modified Android Application Screenshot- Hive Graphics	57
43	Server Publishing Data and Application Listening via MQTT	57
44	Web Application Screenshot -Homepage	58
45	Web Application Screenshot -Hive Location	58
46	Web Application Screenshot -Hive Status	59
47	Online Demonstration -Live Changes Through Database	59
48	Online Demonstration -Settings Menu	60

List of Tables

1	Comparison between Several up-to-date Projects	4
2	Time chart until CM3	15
3	Final Time chart	16
4	Used Sensors on Raspberry Pi 2	20
5	Table for Obtained Sound Data Classifications	22
6	Specifications of the Model	25
7	Accuracy Obtained	25
8	Compact Equipment list	49
9	Equipment list	50

1 Motivation & Novelty

1.1 Motivation

Even though, technological and industrial revolutions have been affecting different sectors, their overall impact over the beekeeping sector can be denoted as weak compared to their impact over many other sectors. Beekeepers still have to manually visit hives and check regularly for both internal and external conditions to optimize honey production. This project aims to apply state of the art algorithms and designs to integrate technology to beekeeping sector so that beekeepers can not only monitor the external and internal conditions of the hives, but also use the predictive aspect of the design to ease their timetable planning and optimize honey production. This project serves as a pioneer in terms of sound processing and applying machine learning techniques to hives for Turkey and such a comprehensive design has not been investigated or modelled in literature so far [1]. The sounds collected in the hives will be processed on a run-time fashion so that the beekeepers can act as soon as possible in order not to lose their bee colonies. Besides sound processing, the design will also help beekeepers by sending real-time data of external conditions that also affect honey production such as temperature and humidity to a phone or web application that they can easily use.

As a result of the project, the bee health will be optimized and wasting time of and honey will be prevented as a result of the instant monitoring of the hive environment which varies according to the climate, natural life and environmental conditions. Through the analysis of the sound of the beehive, the measurement of the amount of honey in the beehive is the most innovative aspect of the project. In addition, the real-time analysis of the amount of honey allows the hives to be emptied at the right time to achieve functional efficiency. Our managing company plans to integrate this project as a commercial product which will be sold to individual beekeepers and large-scaled honey production companies. This project can be further improved if the users wishes to also display possible medications that can fix the health issues of the bees if necessary, by not only classifying bees as healthy and ill, but also predicting their medical inflammation. The company can also further improve the project by integrating modules that can also fix or normalize the unusual states detected in external conditions such as decreasing a sudden temperature increase by using MCU controlled fans built in the hive. The project aims to appeal to the people that are active in the beekeeping sector. In order to monitor continuously without the necessity of visiting the hives, the users will connect over the internet with the options of web and android applications. They will be able to access the medical conditions of the specific hive they are concerned with. Thus, this implementation will lead to an advanced bee-monitoring system that can be purchased by the honey-producing firms. Without the need of a detailed technical background for the users, the health conditions will be monitored and the associated responsible will take action accordingly. Inter-operating with our company, KAREL, a novel product for hive monitoring will be designed.

1.2 Novelty

To discuss the novelty of the project, similar products and projects should be discussed. According to the literature research, five different products or projects were found: BeeHive Lab Project [1], Intelligent BeeHive [2], EDA Hives [2], BigbrotherBees [3] and BeeState [4] monitor. BeeHive project is more like a monitoring project which contains the monitoring of sensory information (temperature, weight, accelerometer) coming inside the hives and provides a data transfer over a GPS module. However, this project does not have any sound related concerns which is our main focus, or any GUI related application features. Intelligent BeeHive also contains a hardware with sensors implemented and an additional GSM module in order to have a data transfer over an Android Application to see live hive conditions [2]. Internet of Things was implemented the project where this project still lacks any machine learning & deep learning algorithms to process the sound coming from the hives. For the main focus of our project, idea behind EDA hives is more similar to BeeSmart since it uses a sound process using artificial neural networks [2]. However, any hardware implementation or sensory information or a web application does not exist. Project also lacks GPS or GSM module. BigBrotherBees uses Internet of Things and machine learning algorithms in order to process the sound but this project doesn't have any concern about sensory information or GPS& GSM modules [3]. Finally, while BeeStateMonitor tracks the sensory information with the help of a web application, it lacks sound processing and machine learning algorithms [4]. So, in terms of novelty, BeeSmart project is the most innovative project when compared with other products and projects in the literature [5]. It is the first project which combines these six different and independent features which are tilt & accelerometer sensors, microphone & sound analysis, temperature and moisture sensors, GPS & GSM modules, Android & Web Applications and Internet of Things & Machine learning algorithms. Following figure represents the comparison of BeeSmart with other projects found in the literature in terms of the listed features above. Additionally, cost will not be increased since the same hardware equipment will be used to monitor the conditions inside the hives. Sound processing and machine learning algorithms will be implemented by the members of the group so no additional budget will be needed other than the hardware equipment. The main differences and combinations of systems of our project compared with the current researches conducted is given via table below.

Projects / Systems	Tilt & Accelerometer	Microphone & Sound Analysis	Temperature & Moisture	GPS & GSM	Android & Web Application	IoT & Machine Learning
BeeSmart	Included	Included	Included	Included	Included	Included
BeeHive Lab Project [1]	Included	Included	Included	Not Included	Not Included	Partially Included
Intelligent BeeHive [2]	Partially Included	Not Included	Included	Not Included	Included	Not Included
EDA Hives [2]	Included	Included	Included	Not Included	Not Included	Not Included
Big Brother Bees [3]	Not Included	Included	Not Included	Not Included	Included	Included
BeeState Monitor [4]	Partially Included	Not Included	Partially Included	Partially Included	Included	Not Included

Table 1: Comparison between Several up-to-date Projects

In order to reiterate its importance, it is very likely to receive a patent for this project. There exists no other patented product or project which tracks the life conditions with sensory information inside the hive using edge-computing of Neural Network models. Hence, in terms of novelty and originality, this project will have an prominent impact in the beekeeping sector, with proper implementation.

2 Design and Performance Requirements

Regarding the necessities of the project, it is divided into two main categories as Functional and Non-Functional requirements. As further analysis done on the project, these main categories were also observed under two subcategories, hardware and software sections. This distinction was essential since the project did branched out considering this basis.

2.1 Functional Requirements

2.1.1 Hardware Requirements

As it was suggested in the previous sections, the project aims to combine several systems in order to observe the health conditions in the hive. Hence, main consideration on the hardware section is to combine and collect the essential sensory components along with strong Micro Controller Units (MCU's) to operate in a robust way with the implemented softwares. With the machine-learning algorithms that is aimed be done with edge-computing, any additional memory is again one of the main considerations at this stage. Overall, the

list of the required hard-ware components during the project can be seen below with their brief explanations for their importance throughout the project.

- 1. Tilt Sensor:** The system should detect any force that might lead to displacement by using tilt and three axis accelerometer sensor such as bear attack to the hive without any human observer.
- 2. Weight Sensor:** The system should measure the weight of the hive in order to provide the beekeeper necessary honey amount information.
- 3. Microphone:** The system should record the sound created by the bees in the hive since the project aims to make live health-status predictions based on sound of the bees in the hive.
- 4. Temperature & Moisture sensor:** The system should measure the temperature and moisture present in the hive in order to inform the user about the physical conditions.
- 5. RAM & ROM:** The system should store the data obtained from the sensors for edge processing and alternative scenarios such as the server connection.
- 6. GPS and GSM:** The project aims to report the location of the hive to the users by using the GPS module. In addition, the constructed system should provide a mobile communication channel for the transfer of the data obtained at the hive using a GSM module. Sending message over MQTT without delay is essential for a successful notification system.

As it will be significant in the latter sections of the report, the initial hardware requirements were essential to shape our hive system. Therefore, there were no minor nor major changes in the hardware requirements in any period of project implementation.

2.1.2 Software Requirements

For the software section of the project, several implementations plays a primary role. Starting with the machine learning algorithms that is developed, embedded software design is essential considering the accurate implementation of sensory components located on the hive. A user-friendly interface on android and web along with strong server & GSM connections are one of the main considerations on software side. The brief description for the necessary software designs is given below.

- 1. Cloud Servers:** The system should interact with a cloud server in order to store the data obtained at the edge and general storage for other applications. After CM3, especially connectivity to internal MySQL database from any IPv4 address, simultaneous saving of the received MQTT messages and E-Mail based alert system were the most significant

requirement of our cloud server as our server's main role was to unify the board located in the hive and the applications.

2. Learning and Sound Conversion Algorithms: The system should develop learning algorithms in order to make predictions based on the sensory readings. As the core of the project, the system should develop sound conversion algorithms in order to make the records be available for any algorithm.

3. Android Application & Web Application: The system should develop Android and web application to enable the interaction between the user and the system.

4. Sensor Modules: The system should develop a sensor module in order to collect the information from the sensory modules.

Again, it will become conspicuous that indeed the project was built upon these software implementation milestones. Thus, there were no major changes made on the requirements and all of them were full-filled during project implementation.

2.2 Non-Functional Requirements

The project aims to design a system which is then be located on the hives. Hence, considering the location of the system, there is a constraint of compactness. Further, from the national perspective, the hives the project is designed on are located in Köyceğiz. This again requires decent connections to immediately inform the end-users. In this perspective, some of the main considerations the non-functional requirements of the project are given below. Further, it is essential to state these requirements are still main concerns, without any change from third committee meeting.

1. Cost: The total cost of the system components will be compensated by KAREL. However, some additional costs like transportation to Köyceğiz will be funded by group members individually. On the overall, the project requires the fond of nearly 10.000 TL to succeed in terms of purchasing the necessary components. Current analysis done for the cost of the project is given in Equipment List section.

2. Transportability & Ergonomic Constraints: The hardware system constructed will be located on the hives. For this reason, the trans-portability and the structure of the hives are essential constraints. A standard hive can weight up to 20-30 kg in mass with the honey produced inside. There is no maximum size constraint however the minimum size of the top part of the hive must be 25x30(cm) in order to place the hardware module. Mobility is another requirement since hives are moved to different forests seasonally (i.e. pine forests or orange groves). Hive picture where the project is designed based on is given below.



Figure 2: Designed Project Hive

3. Reliability & Power Constraints: Regarding the power-constraints, after some literature reviews, we concluded that 24/7 GSM connection is no longer necessary. However, durable power is still a main concern. Thus, it is planned to use battery with 9 amperes/hour to make the hardware long lived. In addition, the hive is going to be placed in different locations through the production season for different kinds of honey; hence, the system should be reliable to perform during non-supervised duration such as 1 week or more depending on the condition of the beekeepers.

4. Health constraints: There are two main health measures that must be recognized during the project which are the bees and people in the surrounding environment. Hence, the health of bees will be taken into consideration by measuring the temperature and humidity continuously and there will be authorized bee-vet for the supervision. The hives are located where the surrounding environment contains beekeepers. In addition, there will be medicine stored near to hives in order to prevent the indications caused by possible bee attack.

5. Safety Issues: A possible bee attack will lead to harmful consequences in case of intolerance; hence, the hives will be located near to other hives where the society is educated about the bees and any human interaction with the hive will be done in a protective suit under the supervise of the bee-vet.

6. Environmental & Social Aspects: The environment that contains the microphones are not suitable for the sound recording because of the material created by the bees. The microphones are required to record even in these conditions. Also, the new bee hives should not distribute the other beekeepers so the hardware design should be well-integrated in order to prevent any possible negative effect on society.

The final product of the project will be compatible with standardization that stated at IEEE Guide for Developing System Requirements [6]. Specifications paper which designed to maximize the reliability of the materials, products, methods, and/or services people use every day. Moreover, since the product being developed with The Ministry of Agriculture and Rural Affairs, it will further be compatible with beekeeping machinery standardization identified by the ministry office -ISO 12824:2016 [7].

3 Big Picture

The Big Picture for the project is given below.

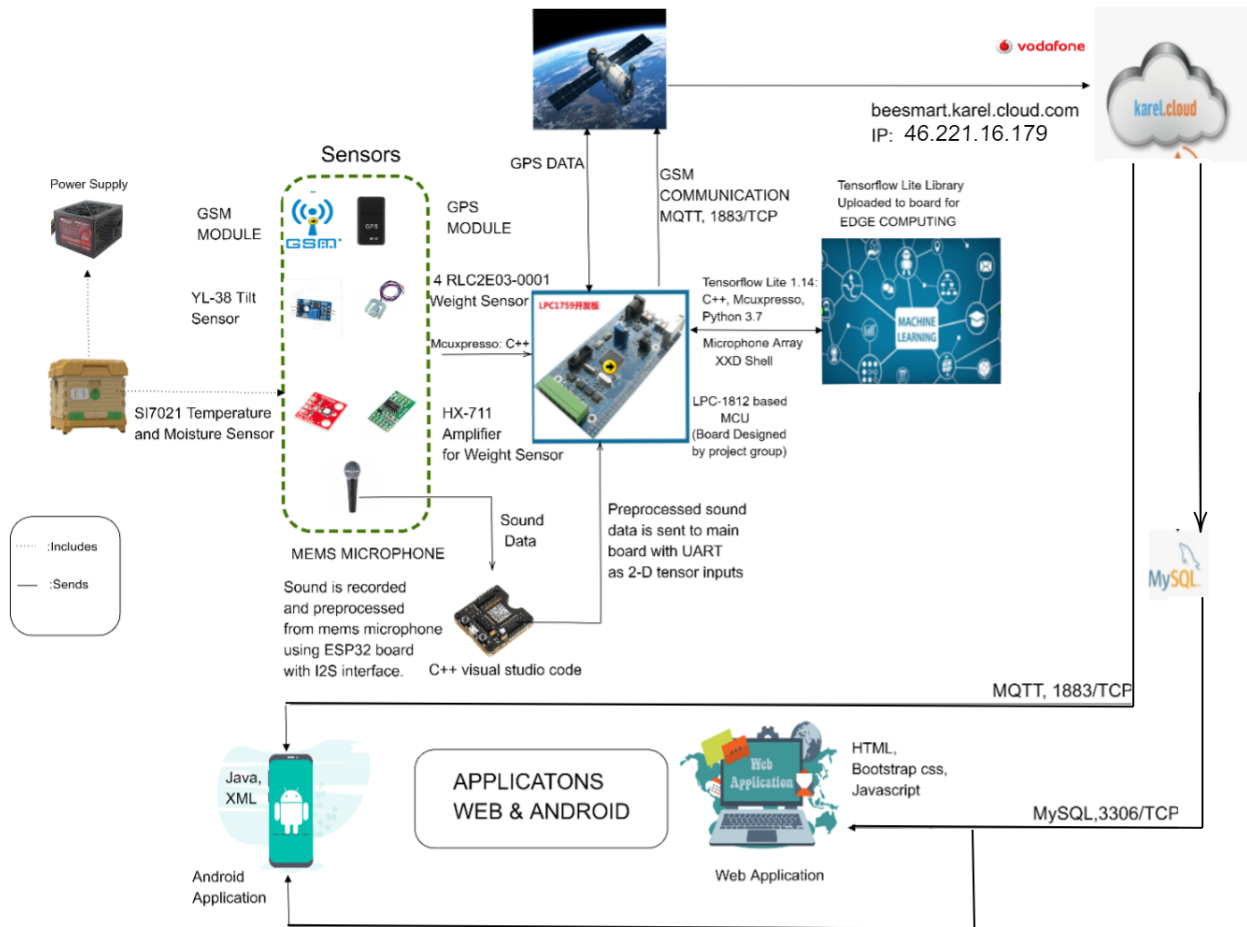


Figure 3: Big Picture of the Project

The project is divided into three main parts. First of all, the area where the BeeSmart device will be located is considered. After that, the server interface is kept in view. Lastly, the application interface is taken into account.

To begin with, location of the device BeeSmart is crucial since it will be mounted to the hives where we simultaneously obtain data from distinct sensors. That yields a sensor network interface between with the hives. This network contains hardware implementation of power supply and all of the sensor configurations as the fundamental subsections. To be more precise in terms of the sensors, it consists of YL-38 tilt sensor, HR-202L Moisture Sensor, DS18B29 temperature sensor, ARD-MDL 12-29 weight sensor, MEMS microphone, GPS module and finally GSM module. In addition to these sensors, additional memory might be required therefore, there will be a room reserved for this case. All of the components mentioned will be connected to the printed board which will have a connection with the power supply.

After the hardware specifications, it is essential to point out the building stone of the project, which is the sound processing using edge computing. Especially sound process is

completed via microphone array that includes edge computing. After CM2, further improvements made in efficient edge computing processes, which the specifications can be seen in the above picture. After several difficulties faced in regard to register storage issues, the essential elements in Tensorflow Lite library was used to eliminate Flash Memory overwhelming. XXD Shell commands were used in opcode process.

Furthermore, utilizing from the 3G and LTE properties of the device, the data obtained from several sensors and modules can be transmitted to the cloud of Karel. Any significant change in the condition of the hives can be detected by the alarm generator that employs the output of the tilt sensor. In cooperation with Karel Company, the necessary ports were opened for connection. These are 1883 and 3306 for MQTT and database connections respectively. TCP connections were made for this purpose. MySQL was configured in order to access the database. In the final section, utilizing from the server-web protocol, the server will be used to inform the users by providing them the web and android applications. The application can be accessed in three different ways. Bee keepers will utilize from user-web interface. On the other hand, for any technical requirement, a technical-web interface can be used. This phase was again accomplished starting from the second semester, where the specifications for these applications can be seen via big picture. There is one minor change of switching to Vodafone – Karel Cloud servers to be able to open port 3306/TCP without compromising the security of Karel servers. This then allowed our IoT components to be merged. Briefly, Cloud with new IP of 46.221.16.179 saves data to MySQL via Auto Subscriber, Android App (Java) and Web App (JavaScript & PHP) take data from the database MySQL with port 3306/TCP.

4 Methods and Implementation Details

In order to reiterate, the project aims to show progress simultaneously considering six main sections. In each of the sections, the group members are working concertedly. Hence, the milestones, work breakdown structure along with the project plan will be analyzed considering these six main sections of the project, throughout this report. These main sections are given as follows:

- Hardware Design
- Embedded Software Design
- Machine Learning Algorithms Development
- Android Application Design
- Web Application Design
- Application to Cloud Connection

4.1 Work Breakdown Structure and Project Plan

Throughout the project implementation, the group members worked in collaboration. Hence, considering the above sections, we can see during progress explanations that all members worked in a trans-disciplinary manner. However, to specify the main responsible of the above sections following list is provided:

Mert Erkul: Android Application Design Responsible

Elif Ceren Fitoz: Hardware Design Responsible

Deniz Can Oruc: Machine Learning Algorithms Development Responsible

Dogan Parlak: Application to Cloud Connection Responsible

Mert Sayar: Embedded Software Design Responsible

Arda Sarp Yenicesu: Web Application Design Responsible

Again, each members contributions will be explained throughout the progress section. Before explicitly demonstrating the work packages and tasks assigned for each of these main sections provided above, it is essential to specify the milestones of the project. Hence, the progress in overall can be briefly seen.

4.1.1 Milestones

The milestones for the project are listed based on the previously mentioned categories. Since we conducted the project in these categories, there were significant achievements to conclude our project is on schedule in regard to them. Hence, for each category we have defined milestones to assure essential operation and success of the category.

Hardware Design: For the hardware design, the main concern was to be able to develop and design a compact Printed Circuit Board (PCB). Since the sensor modules will be connected to this PCB at the end of the project, it was an important section to critically measure several milestones to achieve during this process. Another significance arises from the fact that an initial market research is essential for the sensor modules, since the PCB layout designed after selected all the components. Hence, from the beginning of the project the turning points in this section are given below.

- **Finishing Market Research for Necessary Components:** It was the primary step to specify all the components in order to start the design of the layout.
- **Embedded Circuit Design and PCB Layout:** Another important aspect is to obtain the printed and integrated board of the designed layout.
- **Prototype Testing:** It has a significant importance for the project since it will be visible how accurate the designed circuitry with our embedded software implementation.
- **Finalization of PCB and Integration to Hives:** This final milestone in the hardware section of the project is mainly shows the finalization of the overall implementation.

By achieving this milestone it can be understood that the designed layout is compatible with the overall software implementations in a compact manner and thus can be located on the lid of the hives.

Embedded Software Design: For this section of the project, one of the primary concern is the working principle of each of the components. Since the project requires many sensors to obtain information from the hive, a robust data gathering process must be implemented. Considering this, the following sub-milestones were determined.

- **Collecting and Processing the Data from Sensors:** Simultaneously with the hardware progress, the project shows further development regarding this section by obtaining the necessary modules and test to see they are working correctly on a temporary module. Hence, this is a turning point to conclude the module is able to operate successfully.
- **Conversion of Data for GSM Transmission:** Apart from the other modules, a decent GSM connection has a pronounced importance for further implementations of server and present data to end users.
- **Compatible Implementation for the Designed PCB:** This milestone again shows the finalization of the embedded software design section since achieving this will show that all the sensor modules located on the designed PCB can accurately operate in a robust manner.

Again with this milestones, it is expected to be able to successfully combine all the sensory modules on a board, then send and process the data received from the board via GSM connections and able to show these accurate implementations on the designed board respectively. Hence, the criteria of success is visible that there should be a valid GSM transmission along with accurate data gathering from all the sensors located on the boards in case of any visible change happens in the hive.

Machine Learning Algorithm Development:

- **Pre-processing of the Data:** It is one of the turning points in Machine Learning algorithm developments due to understand the sound signal obtained and implement data cleaning, transformation and finally reduction if necessary.
- **Machine Learning Algorithm Tests:** It is another turning point since the evaluations of the state-of-art algorithms on the data and select with high accuracy predictions along with robustness.

Again, with these turning points for the Machine Learning Algorithms Development, it is expected that the model is able to learn the fundamental difference between the obtained sounds for bees classified differently. Another performance expectation is to implement this in a timely manner without using a major storage. Hence, the criteria for success is again

visible: the project must be able to classify the conditions of the hive with a high accuracy, in a effective way. In addition, with the weight measurements, it should be able to perform a regression task and interpret the expected honey production in the hive.

Android Application Design: Another progress that the project aims is to present a user-friendly android application to monitor the health conditions of the hive without the necessity of visiting the hives. Considering this need, the milestones for this section is given below:

- **Android Application Development for Monitoring:** It is the primary turning point of this section since developing the android application will result as a high progress.
- **Notification of Processed Data to Application:** Another essential concern is to be able to obtain notifications from the application implemented in case of sudden changes in the hive, without the need of accessing to the system manually.

With these turning points in the implementation of android application, it is expected to be used in an effective way by the end users and the application will be able to show accurate results along with visual supports. Another expectation is to obtain notifications from the application. Hence, the criteria of success is the design of an application that automatically informs users with a successful user interface.

Web Application Design: Another fundamental aspect of the project is to provide a web site that can also be used for monitoring the hives. With this purpose, the milestone is given below.

- **Web Site Implementation:** It is an essential progress for the project since online access to end users will also provided via the web site implementation.

By achieving this milestone, it is expected to observe an accurate implementation with further visual support as it is concerned in Android Application Design. The criteria for success is again the user-friendly implementation for the signed up users.

Application to Cloud Connection Configuration: Another significance of the project is the storage of data obtained from the sensor modules located on the hives, on the cloud provided by KAREL Electronics. Hence, the milestone is provided below.

- **Server Implementation:** The server implemented will be able to cluster the data regarding the conditions in the hive and transmit to the android and web applications when necessary.

By accomplishing this turning point for this section, it is expected that the server is able to cluster the data accurately and transmit to the applications. Hence, the criteria of success is again visible, if the received data from the server is the same with the original data send to the cloud; then it can be concluded that a successful cloud connection with a server implementation is achieved.

4.1.2 Work Packages / Tasks

Initially, the tasks for each of the packages provided were divided into two main phases to be focused on. In the first phase, the planned focus for the project were the progress made in the Hardware Design, Embedded Software Design and Machine Learning Algorithms Improvements. Whereas in the second phase, which started from the second semester, after achieving sufficient amount of progress regarding the sections mentioned in the first phase, the main focus was on the server implementation along with web and android design. The initial work breakdown structure can be seen from the below graph.

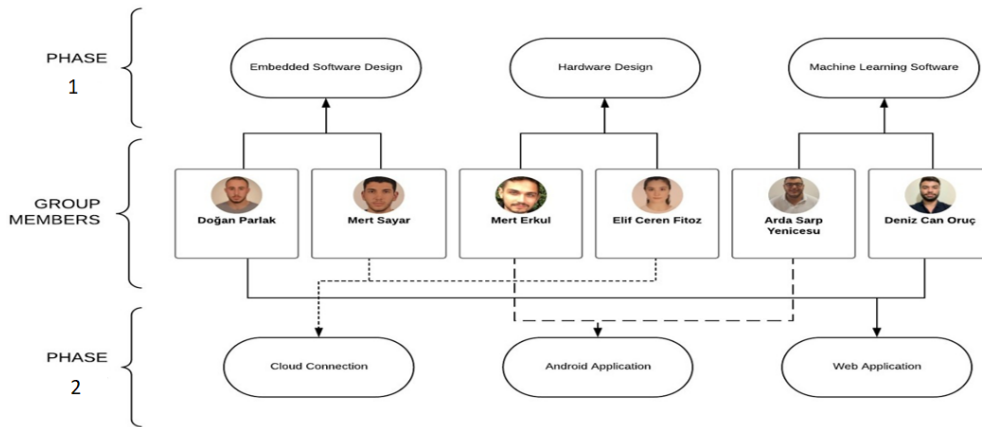


Figure 4: Work Breakdown Structure for Sections -Two Phases

Considering their timelines, the first and second phases were arranged in a way to show parallelism between the semesters. The planned achievement regarding the first phase is thus had the deadline of end of Fall Semester. The first proposed timeline regarding these phases are given below in the Project Plan section with phases separated.

In order to briefly state the division strategy for the tasks assigned between our members, each of these categories regarding the first and phase and the tasks associated is given with responsible members. It is clear that up to CM2, most of the hardware related tasks and work packages were almost completed.

Hardware Design: For the hardware design, group members Mert Erkul and Elif Ceren Fitoz were responsible. As the initial step of the project, they worked on determining the sensor requirements and designing the PCB board. After the layout design review and completing several tests, the PCB board was verified that indeed it was able to perform the sensory components in an accurate manner. The PCB layout with a small delay was obtained right after the second Committee Meeting, and the testings were held up to COVID-19 outbreak.

Embedded Software Design: The responsible group members were Dogan Parlak, Mert Sayar and Arda Sarp Yenicesu. Again, right after the MCU specifications made by hardware

responsibles, the protocol tests were done. After working on several boards including Raspberry Pi and LPC1812, they understood the principles of sensory components and worked on compatible implementation to the designed PCB. The final task was started from the end of Fall semester and majorly completed up to the third progress meeting. Until the COVID-19 outbreak, major focus was on the microphone compatibility analysis.

Machine Learning Algorithms Development: Responsible group members for the tasks associated with Machine Learning and edge computing were Mert Erkul and Deniz Can Oruc. After obtaining the sound data gathered from hives in Köyceğiz, they worked on obtaining the optimal model with several tests associated. After performing feature extraction, they implemented training process on the model.

Before moving on to the second phase, it is again essential to discuss the modified table with the setbacks arose during the second Committee Meeting. Since the arrival of the designed PCB and its soldering process took longer time than it was expected, the Android Application and GSM implementation which was planned to be done in the second phase did became the focus in order to compensate the loss on the Hardware Design section. Further, implementations on the server started and necessary configurations were made. Hence, although we were not able to check the validity of the connections made for out designed PCB, the loss fully compensated with these further progress made, which was initially assigned for the second phase. This modified time-line for completed tasks and reorganized works are provided in Appendices part A.

Starting from the second semester, the main focus was on making the sensors compatible along with performing edge computing on the designed board. The group members were then divided uniformly to these two sub-groups to effectively proceed in these sections. Up to the third Progress Meeting, the group were able to successfully show progress in embedding software components and implementing machine learning algorithms on the designed board. The overall progress is presented in detail in the following sections.

Although there were minor tasks associated with the hardware components, the second phase focus was on the Server, Android and Web Applications sections due to the outbreak. There were several modifications related to server connection configurations made during this time interval, up to PM3, and accelerated during the outbreak. The modified tasks then assigned among the group members for the second phase with including COVID-19. After this modification in the timeline, the project was on schedule and no further modifications were necessary. The project is completed according to this time-line rescheduled for Committee Meeting 3.

Application to Cloud Connection: Responsible members were Mert Erkul, Elif Ceren Fitoz and Mert Sayar. Main aim was to operate the server in a way that it is able to get data from a specific client, -there is a possibility of having a temporary publisher at this stage since the publisher requires our designed board-, and serve to the subscribers on request. After contacting with Karel and opening the necessary ports for the server, the

database and MQTT connections was the primary task of this part. Then, depending on whether successfully operation the publisher-server-subscriber mechanism achieved, several adjustments to be made on the server to keep data up to a specific period and obtain yearly, monthly or daily characteristics of our recordings at the client edge were also assigned. The server configurations were done up to the third progress meeting.

Android Application Design: Responsible group members were Mert Erkul and Deniz Can Oruc. Although there were an initial progress shown prior in this section, further improvements were available after server configurations. Then, the primary task for responsible group members was to obtain the weekly, monthly and yearly graphs with the data obtained from the database and display for the selected hive -which requires the connectivity to server and database essentially.

Web Application Design: Responsible group members were Dogan Parlak and Arda Sarp Yenicesu. Again, primary aim is to obtain connection to the cloud server configured. There were again some progress regarding graphical user interface in the earlier period. However, the suitable interface must be used for connection and handling the up-to-date data was the concern during this modified phase.

The modified project plan for the second Committee Meeting, along with the final version with further modifications made in regard to COVID-19 is presented in the Appendices sections A & B for better, magnified visualizations. However, a brief explanation for changes occurred due to the outbreak is presented below.

Considering all parts individually up and running, combining them all together was the remaining task after CM3. However, for Android and Web applications, major work packages were completed, with the final step of testing and integration to the hardware components. Thus, the tasks that were incomplete right after CM3 is presented below for brief observation.

WORK PACKAGE	TIMELINE COMPLETENESS CORRESPONDENCE
Hardware Design	Completed – 12.02.2020
Embedded Software Design	Individually Completed – 10.03.2020
Machine Learning Algorithm Design	Completed – 28.02.2020
Cloud Server Establishment	Completed – 01.05.2020
Web Application	Partially Completed – 03.05.2020
Android Application	Partially Completed – 03.05.2020

Table 2: Time chart until CM3

As we have finalized our project during this time interval by achieving the specified milestones, our project completeness can be seen below.

WORK PACKAGE	TIMELINE COMPLETENESS CORRESPONDENCE
Hardware Design	Completed – 12.02.2020
Embedded Software Design	Completed – 15.06.2020
Machine Learning Algorithm Design	Completed – 28.02.2020
Cloud Server Establishment	Completed – 01.05.2020
Web Application	Completed – 16.06.2020
Android Application	Completed – 16.06.2020

Table 3: Final Time chart

The progress report will be given in a detailed manner with the following section, however, the tables 2 and 3 demonstrate the summary of what we have accomplished between our online CM3 meeting and until the finalization of our project, which is dated as 25th of June. During the interval, the incomplete tasks along with work packages that need revisions can be noticed, which were Embedded Software Design, Web and Android Application. The overall merge was accomplished after completing these sections.

4.2 Methods and Progress

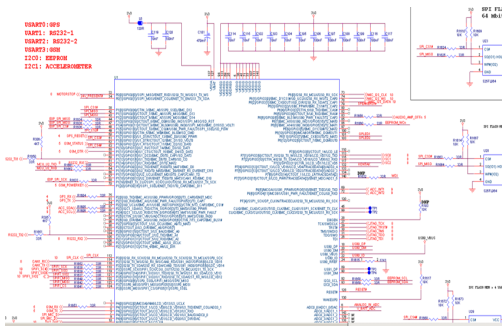
For the methods used and progress achieved until and up to the third committee meeting (CM3), the solution strategies regarding each accomplished tasks along with the tools will be specified in the following categories: Progress up to Committee Meeting 3 (CM3), Progress between CM3 and present and tasks/work packages that could not be finished. After these specifications, the present risks along with their planned precautions with effects on the project will be discussed.

4.2.1 Progress up to CM3

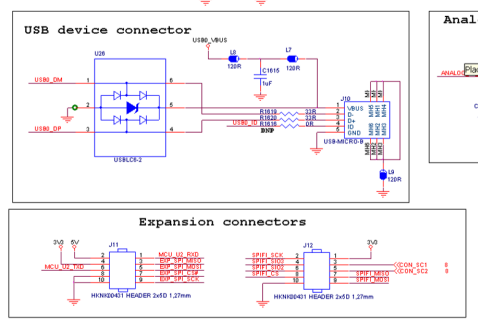
The progress made up to the third committee meeting will be discussed regarding the sections specified for the project. It is essential to remind that the progress mainly focuses on the sections considered to be implemented in the first phase.

a. Hardware Design: With the help of KAREL, the group members Elif Ceren Fitoz and Mert Erkul received tutorials on how to use OrCAD for PCB design. After the tutorials, the PCB design was made simultaneously with the market research conducted for sensor modules. Below some of the designed component modules along with their connections to the Micro Controller Unit NXP board is given.

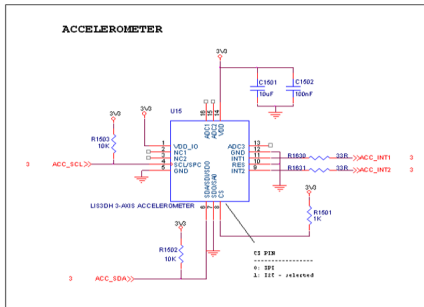
After accomplishing designing schemes for each components, several revisions made. The overall designed circuitry is then sent for layout order. Hence, two of the milestones in the hardware development section were achieved until the first committee meeting. The ordered two PCBs have arrived during the interval and immediately sent to our firm KAREL Electronics for soldering the necessary components. After Figure 5, the designed PCB is depicted.



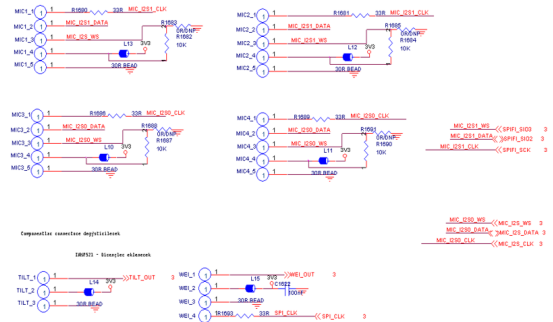
(a) MCU Connections



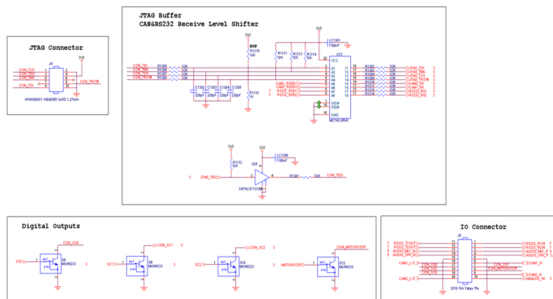
(b) USB Connections



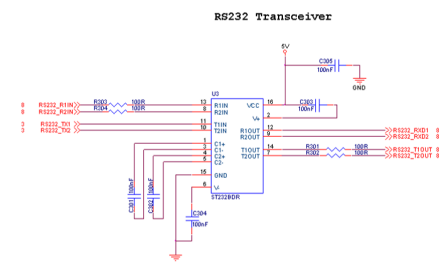
(c) Accelerometer



(d) Microphone Array, Tilt, Weight Sensors



(e) I/O Port Connections, Digital Outputs



(f) GSM Antenna Design

Figure 5: Several Designed Component Demonstrations

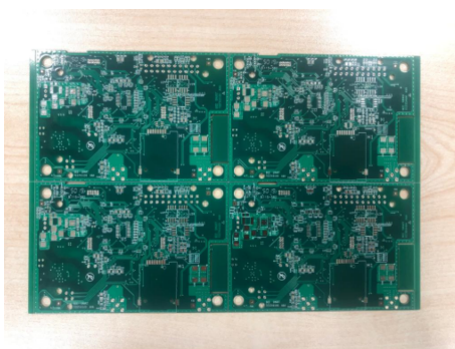


Figure 6: Designed PCB

The other devices necessary for later implementations like debugger which was used

to check the validity of connections within the designed board were again ordered. The debugger device ordered for this purpose is again given below. Another progress achieved in this interval was the placement of the battery and the temporary board on hive lid with exact dimensions. The placed battery along with the board can be seen below.



(a) Debugger

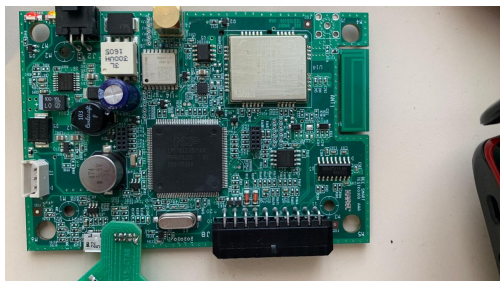


(b) Placed Devices on the Lid of the Hive

Figure 7: Debugger and hive lid

After CM2, we obtained our board with all components being soldered by our firm Karel Electronics. Validity of the connections of components with our MCU were checked. Debugger was already obtained, board was being used for programming purposes after CM2. Additional sockets were soldered during this process. The lid was already designed for our board unit including the power supply. In addition, the hive which will be used in the demonstrations was also obtained. Below, the soldering process along with one of the soldered, complete board is demonstrated.

As it can be seen from the visuals, that main and final milestone of the hardware design part was achieved after ensuring the successful operation of the designed board. Considering the pandemic, it was a major advantage to finalize the PCB design in the first phase, hence no further changes or updates in the layout was made after CM3.



(a) Complete PCB



(b) PCB Soldering

Figure 8: PCB Soldering

b. Embedded Software Design: On the Embedded software section, three STM32-F407VG boards were provided to members Elif Ceren Fitoz, Dogan Parlak and Mert Sayar by the KAREL company. The board at first used for several fundamental but necessary protocols such as Timer Interrupts, USART connection, transmitting and receiving bit implementations since it does not have the same MCU unit with the designed one. The members Dogan Parlak and Mert Sayar are then worked on the ordered sensor module software implementations on the temporary STM32 board provided.

As we could not get our hands on our completed board at that time, we implemented necessary components on Raspberry Pi Version 2.B provided by our company, rather than losing time on the STM32 board. It is essential to note the subtle changes in the Embedded Software during this time interval. Although we have started to work on the fundamental connections on our temporary STM32F407vg board, the GSM and GPS modules were also obtained to compensate for the late PCB arrival. Hence, in order to test the connections effectively, we have implemented along with all the sensors on Raspberry Pi 2 Model V1.1.

Another change arising from this modification was the microphone usage. Since Raspberry does not have Inter-IC Sound Bus (I2S) bus interface, we decided to use microphone instead of MEMS that will be used on our original board. The resulted combination of all the modules to Raspberry Pi board is given below.

Overall, we were able to successfully implement all the sensory components that will be used during the project managed to observe notification transmission from GSM connection in sudden changes occurring in the hive for the second Committee Meeting.

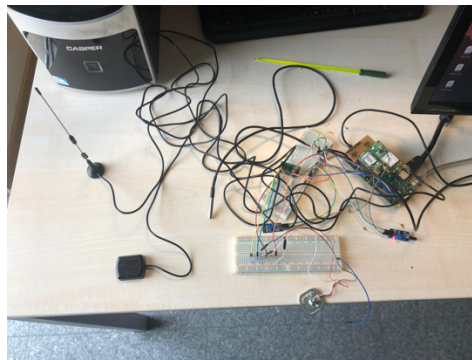


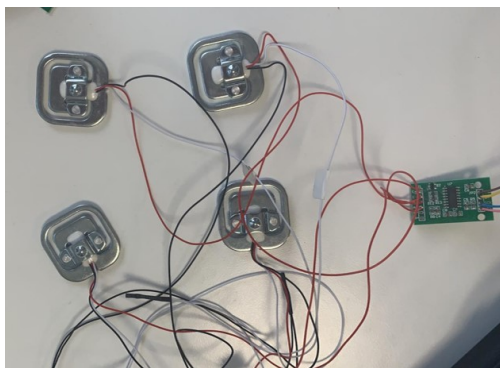
Figure 9: Overall Designed Circuitry

Figure 9 demonstrates the connections of each component used in the temporary Raspberry Pi board. Upper left is the GSM antenna that is currently being used. Further, the GPS Antenna is located at the lower left in the figure. Two breadboard contains the other sensory modules which is listed in Table 4. Finally the Raspberry Pi board is located on the right with GSM and GPS modules attached on. The overall sensors used during the implementation is given via table below.

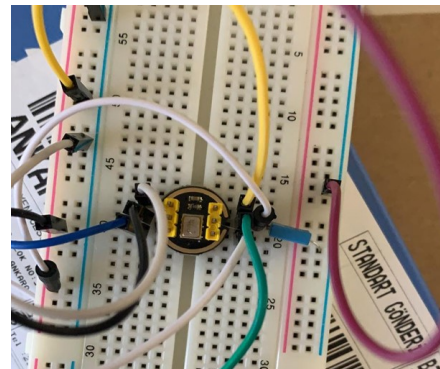
Temperature: DS18B20 (One wire)	Tilt: YL-38
Humidity: HR202L	GPS Antenna (1575.42 Mhz)
GPS Module u-blox LEA-6S-0-001	GSM Antenna
Telit-GL865	Trust Mico Microphone
HX711 weight sensor	ARD-MDL 1229 load cell

Table 4: Used Sensors on Raspberry Pi 2

This implementations made on Raspberry Pi was essential in order to fully integrate and plan the operations of several sensors to be used on the designed board thoroughly. Since the soldered PCB was obtained after CM2, the main goal was to make all the sensors compatible with this designed board. In the mean time up to outbreak, temperature, humidity, tilt sensors along with GPS and GSM modules were successfully implemented on the board. The microphone and weight sensor implementations were disrupted due to outbreak, however significant progress was shown with these sensors. As a modification, temperature and humidity were obtained from a single component using I2C protocol rather than obtaining individually via one-wire configuration. The sensors used for the implementation overall is shown below.



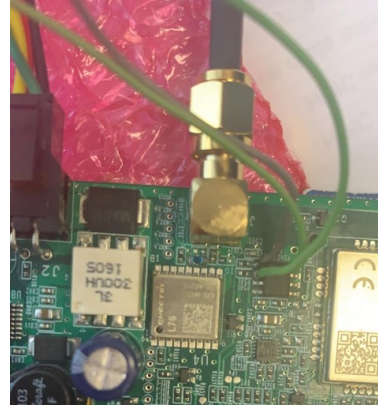
(a) Weight Sensors with HX711 Amplifier



(b) INMP441 Low Volume I2S MEMS Microphone



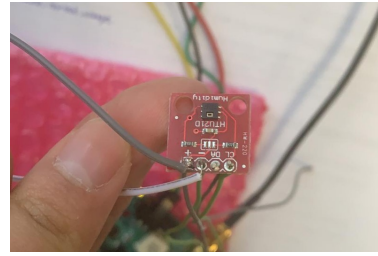
(c) GSM Module –QUECTEL M95



(d) GPS Module – QUECTEL L76



(e) Tilt Sensor-YL38

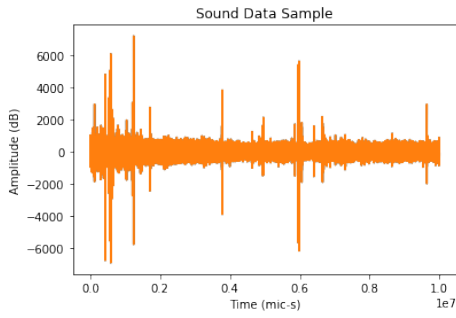


(f) Temperature and Humidity -HTU21D

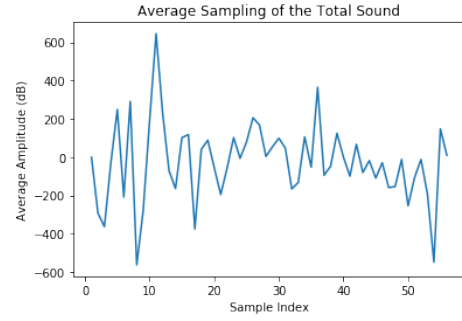
Figure 10: Sensory Components Used

Although the microphone implementation was interrupted due to pandemic, the significant progress can be seen with the screenshot taken in Appendix C, where obtained sound data is presented. One important design modification made is essential to point out. The microphone implementation was conducted on ESP32 board provided and custom-made by KAREL Electronics. The main reason is due to memory considerations on pre-processing and MCU board specifications. The modification is explained in the following Machine Learning section, in a detailed manner. Therefore further experiments on microphone implementation was essential in the latter stage, where company access was provided during pandemic.

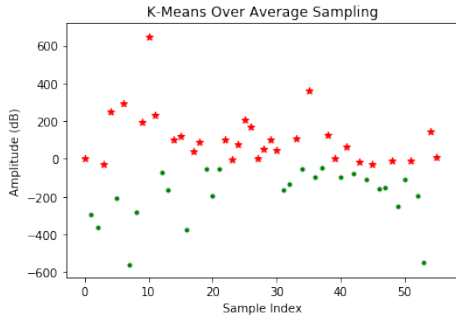
c. Machine Learning Algorithms Development: With the help of our supervisor Alper Sarikan, the unlabeled bee sounds were recorded directly inside the hives and obtained to start Machine Learning Algorithms implementations immediately in the first phase. Since the data did not contain the labels, the initial implementation was determined as sampling the data for every 50 seconds and then running an unsupervised learning algorithm, which is K-Means Clustering. The initial stage was to distinguish the unhealthy and healthy sounds from themselves. Below, there are some observations on the sound data and Algorithm results were given.



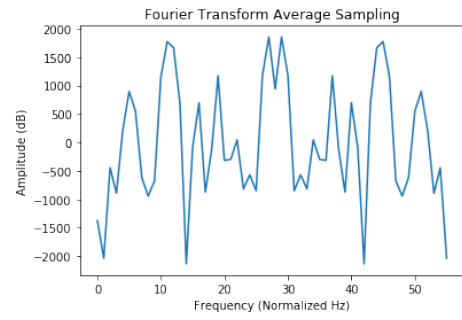
(a) Data Properties



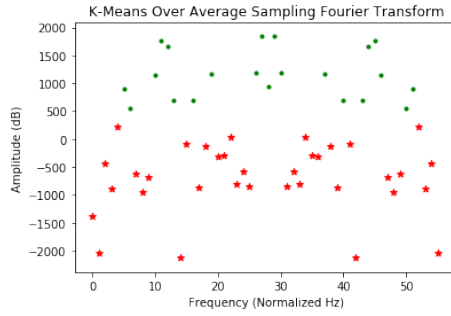
(b) Sampling Results



(c) K-Means Clustering



(d) Fourier Transform of the Sampled Sound



(e) K-Means Clustering on the Transformed Data

Figure 11: Results Obtained from K-Means Clustering Unsupervised Learning

During this interval, we also received labelled sounds in the hives for a total of 30 Hours. The data and their labels, corresponding duration can be seen in the table below after augmentations:

Labels	Durations	One-Hot Encoded Labels
Hive Full, Bees Healthy	7 h 25 m	[1 0 0 0 0]
Hive Half Full, Bees Healthy	6 h 13 m	[0 1 0 0 0]
Hive Empty	6 h 24 m	[0 0 1 0 0]
Hive Full, Bees Unhealthy	4 h 41 m	[0 0 0 1 0]
Hive Half Full, Bees Unhealthy	2 h 08 m	[0 0 0 0 1]

Table 5: Table for Obtained Sound Data Classifications

In order not to create an unbalanced data set, we implemented circular shifting and sampling to the 5th label, which is half full hive with unhealthy bees. As our solution strategy, after literature reviews, we selected to implement Convolutional Neural Networks on the Mel-Spectrogram Images of sounds parsed at every 6 seconds [8]. With this implementation, total of 1500 images were obtained with 3000 images for each classes. Below the sound samples for each classes, for mel-spectrogram image generation along with the obtained mel-frequency spectrogram images are depicted.

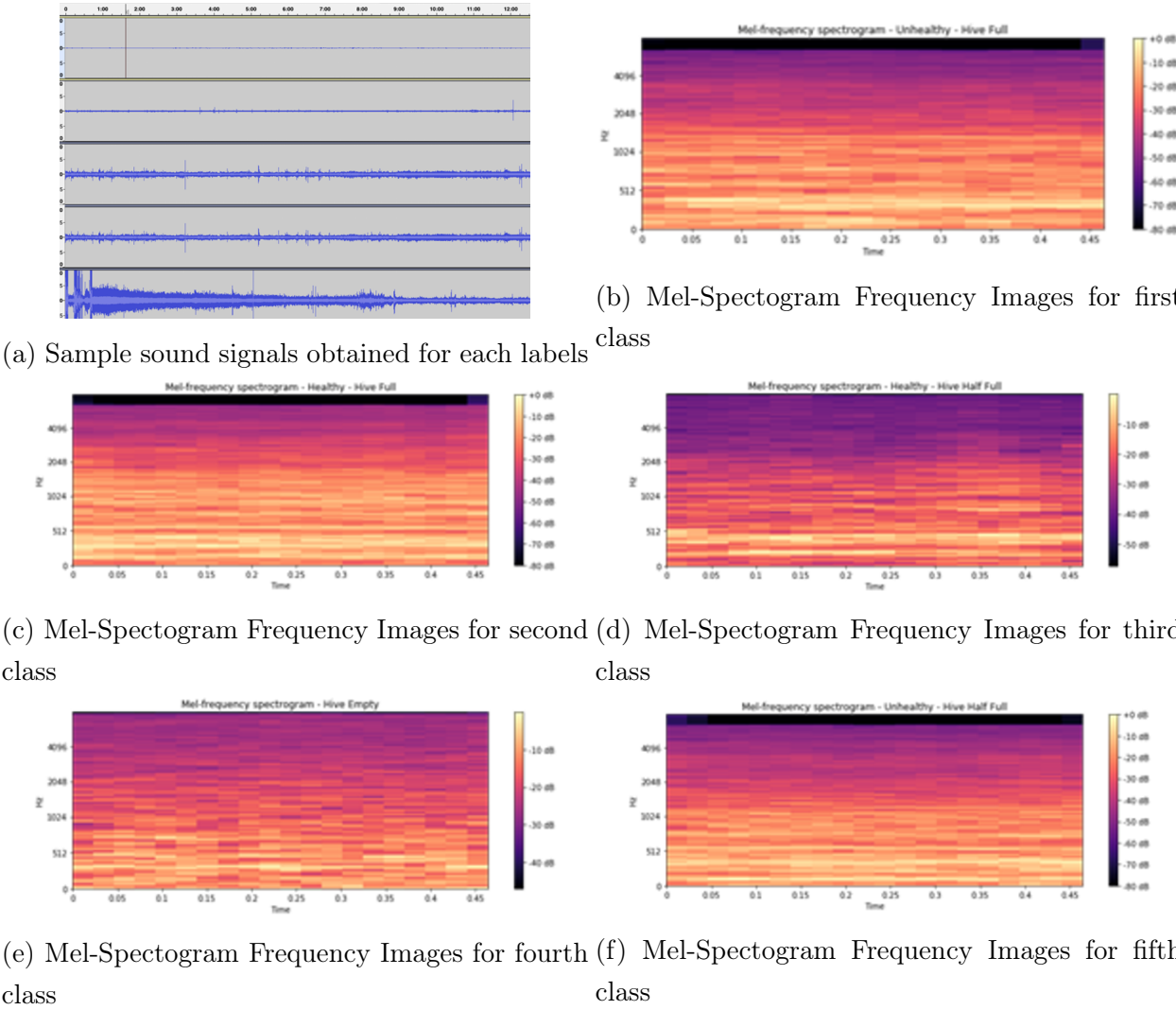


Figure 12: Sample Sounds and Obtained Mel-Frequency Spectrogram Images

Initially, we considered Transfer Learning for our model, by freezing the output layer and adding an additional Convolutional Sandwich Layer in the input layer of ResNet-50 [8]. However, a major downside of this initial implementation was 261 MB storage needed to store all the parameters of the model. Secondly, as a 6 second 16-bit unsigned stereo WAV recording is approximately 0.2 MB, which requires over 32 GB of GSM Usage monthly. Considering the budget constraints for further implementations, instead of transferring the sounds over GSM, we decided to implement a simpler model for Edge Computing with TensorFlow Lite or TFLearn (both TensorFlow models), where CNN architectures can be

implemented on MCUs [9]. Hence, we were able to decrease the storage necessary for parameters to 288 KB and substantially decrease the GSM usage with edge computing. Below figure represents the comparison of both models applied for this algorithm respectively.

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	(None, 120, 320, 3)	0	
conv1_pad (ZeroPadding2D)	(None, 126, 326, 3)	0	input_1[0][0]
conv1 (Conv2D)	(None, 60, 160, 64)	9472	conv1_pad[0][0]
bn_conv1 (BatchNormalization)	(None, 60, 160, 64)	256	conv1[0][0]
activation_1 (Activation)	(None, 60, 160, 64)	0	bn_conv1[0][0]
pool1_pad (ZeroPadding2D)	(None, 62, 162, 64)	0	activation_1[0][0]
max_pooling2d_1 (MaxPooling2D)	(None, 30, 80, 64)	0	pool1_pad[0][0]
res2a_branch2a (Conv2D)	(None, 30, 80, 64)	4160	max_pooling2d_1[0][0]
bn2a_branch2a (BatchNormalization)	(None, 30, 80, 64)	256	res2a_branch2a[0][0]
activation_2 (Activation)	(None, 30, 80, 64)	0	bn2a_branch2a[0][0]
res2a_branch2b (Conv2D)	(None, 30, 80, 64)	36928	activation_2[0][0]
bn2a_branch2b (BatchNormalization)	(None, 30, 80, 64)	256	res2a_branch2b[0][0]
activation_3 (Activation)	(None, 30, 80, 64)	0	bn2a_branch2b[0][0]
res2a_branch2c (Conv2D)	(None, 30, 80, 256)	16640	activation_3[0][0]
res2a_branch1 (Conv2D)	(None, 30, 80, 256)	16640	max_pooling2d_1[0][0]
bn2a_branch2c (BatchNormalization)	(None, 30, 80, 256)	1024	res2a_branch2c[0][0]
bn2a_branch1 (BatchNormalization)	(None, 30, 80, 256)	1024	res2a_branch1[0][0]
add_1 (Add)	(None, 30, 80, 256)	0	bn2a_branch2c[0][0] bn2a_branch1[0][0]
activation_4 (Activation)	(None, 30, 80, 256)	0	add_1[0][0]
res2b_branch2a (Conv2D)	(None, 30, 80, 64)	16448	activation_4[0][0]
bn2b_branch2a (BatchNormalization)	(None, 30, 80, 64)	256	res2b_branch2a[0][0]
Total params: 65,679,105			
Trainable params: 65,625,985			
Non-trainable params: 53,120			

Figure 13: Properties for Transfer Learning Model

Layer (type)	Output Shape	Param #
conv2d_5 (Conv2D)	(None, 62, 8, 3)	78
batch_normalization_5 (Batch Normalization)	(None, 62, 8, 3)	12
conv2d_6 (Conv2D)	(None, 29, 2, 6)	456
batch_normalization_6 (Batch Normalization)	(None, 29, 2, 6)	24
flatten_3 (Flatten)	(None, 348)	0
dense_7 (Dense)	(None, 348)	121452
dense_8 (Dense)	(None, 64)	22336
dense_9 (Dense)	(None, 5)	325
Total params: 144,683		
Trainable params: 144,665		
Non-trainable params: 18		

Figure 14: Properties for Edge Computing with TF Lite

Continuing with our edge computing model, the model specifications along with the accuracy obtained are given in tables. Further predictions for each of the classes after tuning the necessary parameters are given below in Figure 15.

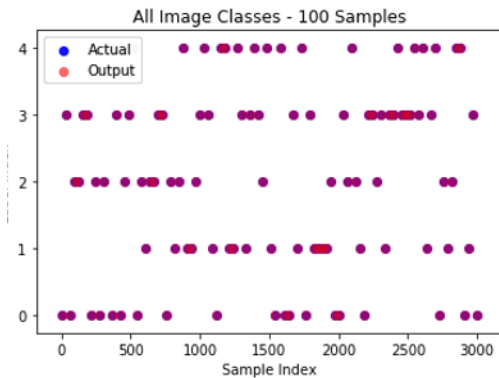
Optimizer	Adam
Hidden Layer Activation Function [9]	Unipolar Sigmoid
Output Layer Activation Function	SoftMax
Kernel Size	5*5
Convolution Strides	2
Mini-Batch Size	50
Sample Rate	22050
Input Image Dimension	120*24*1 (Normalized Grayscale)

Table 6: Specifications of the Model

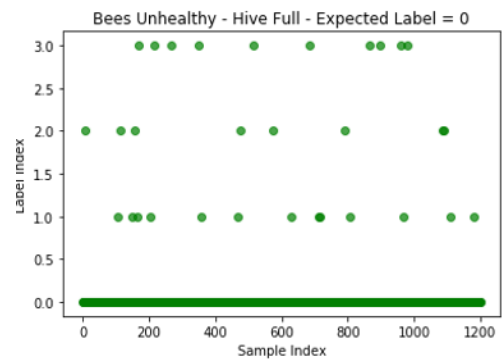
Inputs	Accuracy (%)
Test Set (3000 Images – 600 Each Class)	96.9
Only Label 1 (1200 Images)	94.7
Only Label 2 (1200 Images)	91.9
Only Label 3 (1200 Images)	95.6
Only Label 4 (1200 Images)	92.4
Only Label 5 (1200 Images)	99.1

Table 7: Accuracy Obtained

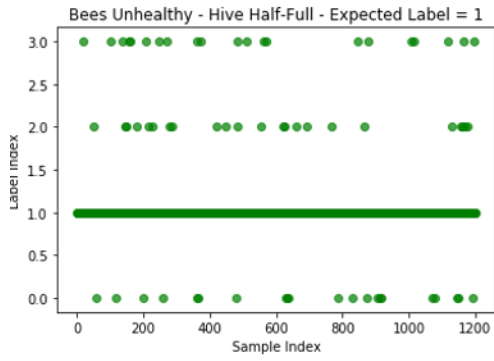
It can be seen from Table 7, the CNN model constructed with the specifications given in Table 6, succeeded in predicting health states along with differentiating the occupancy of the hive. Again, visual support for these predictions above with further observations on the predicted values are given in the figure below.



(a) Overall Predictions Made



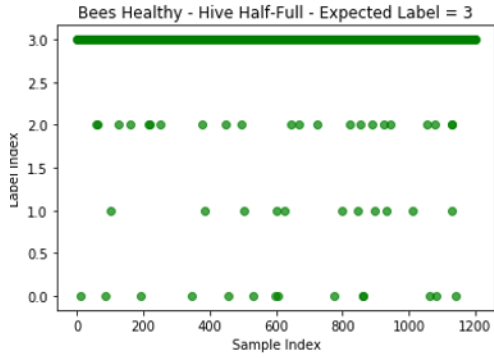
(b) Predictions for Class 1



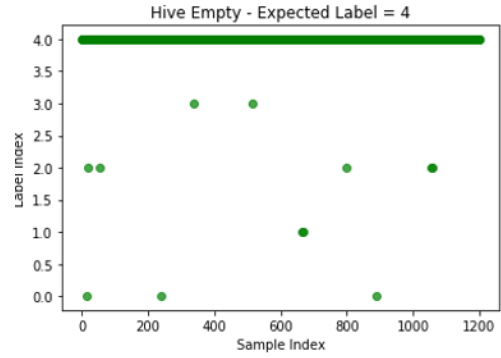
(c) Predictions for Class 2



(d) Predictions for Class 3



(e) Predictions for Class 4

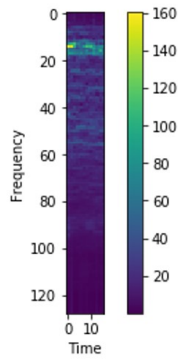


(f) Predictions for Class 5

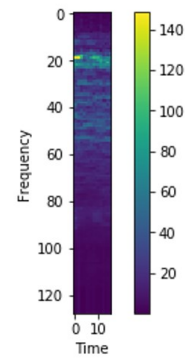
Figure 15: Observations on the Predictions in Overall and Class-wise

After constructing the initial model that works optimally, it was an essential step to successfully implement sound processing using edge computing on the designed, newly arrived board. Since our selected MCU Family (LPC1812) doesn't support SDK for NXP eLQ TensorFlow Lite Library and Librosa Library for pre-processing of the obtained wav data, we needed to create C and C++ Static Libraries. Project building on our IDE MCUXpresso required many inclusions and SEGSI (Segmentation Fault Signal) debugging. In addition, XXD Shell commands were used for .tflite to OPICODE C++ byte arrays.

As a preprocessing step of the obtained sound data, Mel-frequency spectrogram conversion was made likewise the previous implementations. Hence, Short Time Fourier Transforms of the sound signals were achieved and then filtered with a Mel-Filter bank. This step requires a simple matrix multiplication thus does not overwhelm the system computationally. For this purpose, FFTW –a C subroutine library for computing the Discrete Fourier Transform (DFT) was used. Mel-Filter Bank was obtained which has the maximum frequency of 11050 Hz and total of 128 filters. In order to verify the accuracy of the resulted mel spectrograms, the direct library was again used for comparison. The results can be seen below.



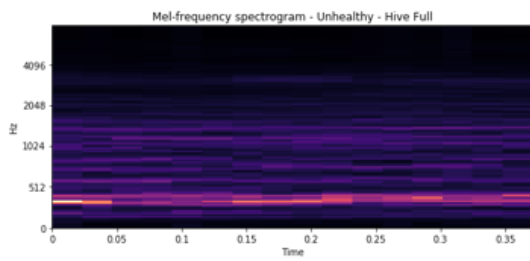
(a) Results using Librosa



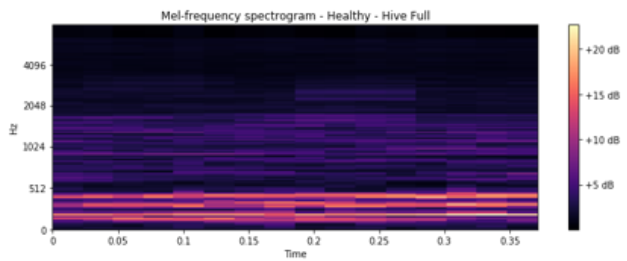
(b) Implementation Results

Figure 16: Spectrogram Comparisons

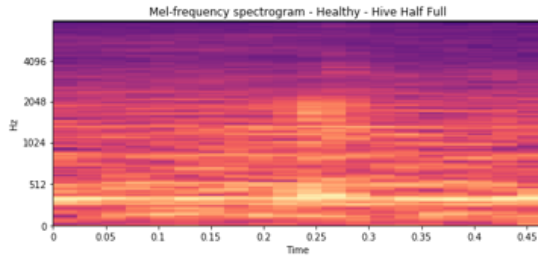
The final implementation results are finally presented below.



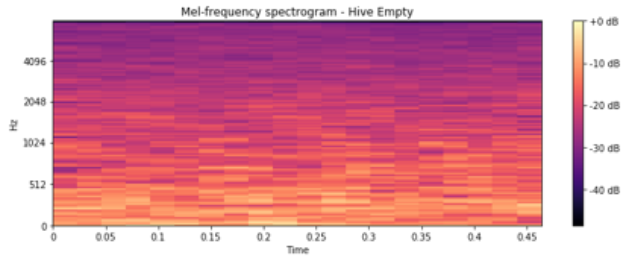
(a) For Unhealthy, Full Hive



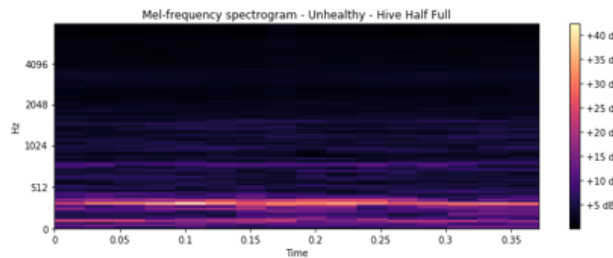
(b) For Healthy, Full Hive



(c) For Healthy, Half-Full Hive



(d) For Empty Hive



(e) For Unhealthy, Half-Full Hive

Figure 17: Mel-Frequency Spectrograms for Hive Conditions

Finally, the output achieved with both Python and MCU output for comparison is given in the table below.

Probabilities	Label 0	Label 1	Label 2	Label 3	Label 4
MCU Prediction	2.01e-6	9.25e-6	1.28e-3	9.99e-1	2.27e-6
Python Prediction	2.00e-6	9.25e-6	1.28e-3	9.99e-1	2.25e-6

Initially, because of TensorFlow Lite and Keras Backend compatibility issues, we received register storage issues, heap was overflowed and interrupt routines were overridden. We decided to change the PYPATH to Tensorflow-Keras instead of regular Keras and save the C byte array to flash memory (ROM) for space clearing. Since the entire Tensorflow Lite library is consists of more than 400 .c, .cc, .h extentions, we only used CONV_2D, FULLY_CONNECTED, LOGISTIC and SOFTMAX and supporting OPCODE functions in order not to overwhelm the Flash Memory (ROM), which is 512 KB. Since our MEMS microphone was implemented on ESP32 board, in order to see whether our board can handle edge-computing, we compared the outputs of embedded machine learning model with our Keras model in PC. Although we were not able to work with microphone unit due to outbreak for a while, we have conducted real time classification using the data obtained from the I2S MEMS Microphone to see whether the Heap & Stack space allocation is sufficient or not after CM3 presentation. Further studies on real-time sound classification using our MEMS microphone, without overflowing the heap on our board is provided in the next section, where progress made after CM3 is discussed.

d. Application to Cloud Connection: As it is crucial and one of the main tasks of our project to provide real time data transfer for users to monitor hive states, with addition to viewing medical conditions of the hives through analyzing and processing their sounds, simultaneity was a necessity in our project. For this to be accomplished, we initially got a dedicated ubuntu machine located in IPv4 address 90.158.24.100, which is an IP address owned by our firm Karel Electronics. With agreement of our company, our designated data transfer protocol was decided to be MQTT, which provides reliable data transfer using the transport layer protocol TCP on the port 1883. Besides having a real time functionality, we also required to have past data viewing section, for observing the improvements, honey amount increases and effects of external tools and conditions, hence, we also decided to implement a MySQL server that automatically saves the data listened and that has passed through our server using a specified client credentials over a specified MQTT topic. For general usage and possible extensions as a product, we decided to implement an Auto Subscriber at the backend of our server through Ubuntu native language Python, which will be further explained in the following subsections. After successful integration of MQTT – Auto Subscriber – Board communication modules were completed, for a better User Experience, we required the second leg of this connection to function correctly and accurately as well, which will all be done through MySQL connection, as it is our reliable channel for viewing the data sent through all of the boards. The process will be explained further below.

Server Establishment for MQTT: Initially, we were given a completely empty Ubuntu 16.04 machine running on the IP 90.158.24.100. Initially to avoid intruders, we

were provided with an SSH RSA key designed especially for PUTTy communication. Prior to telling us the entire IPv4 address, we were given a DNS ID, which was *beesmart.karel.com.tr*, however, for that particular domain, the hardware part for port forwarding was not established, such that, this DNS forwarded us to 90.158.24.90 instead, which didn't have port 1883 (MQTT/TCP) not opened yet. We initially tried to open 1883 through bash commands such as "sudo ufw allow any to 1883", yet the port was still closed, hence, we figured out the DNS solution instead, but even though the ports were opened, as there weren't any listeners attached (such as a MQTT broker), the port still seemed closed. After realizing this, we decided to use an open-source MQTT broker, Mosquitto, that runs indefinitely on the cloud, restarts as the machine restarts and never shuts down. Using "*mosquito_conf.d*" file, we restricted connections to the Broker as well, so that the unique identifiers of the Broker were to be only given to client applications, specifically, Android and Web. After conducting initial tests with open-source client applications such as Mosquitto_Sub and Mosquitto_Pub using our own server and broker, we concluded that the broker service was complete and correct.

Database and Auto Subscriber Functionality in the Servers: After acknowledging that the servers are up and running for the MQTT connection, we realized that the data that is to be streamed, should exist and to be saved even if there isn't a live subscriber through a client that is listening to our particular MQTT topic. Data saving part was simple, as this task was only accomplishable through MySQL, which is a database service that uses TCP/IP for reliable data transfer and communication, other services such as SQLite are built to function on local host servers only. So, what we did, after installing necessary MySQL connectors, was to grant access to all IPv4 addresses to the database "beesmart" located in the server. Now, we only required a "hypothetical" listener that is always active and on the correct MQTT channel, after listening the channel and obtaining the appropriate JSON Dictionary (our selected data structure for MQTT transfers from the board), parsing the appropriate strings and saving them into the correct tables in the database, which is why we created an Auto Subscriber system. Since Linux variants are Python native, we decided to implement a Python 2.7 script, using MySQL - Python adapter and a MQTT client (once again open source Paho MQTT client), we can establish this task as Linux OS machines allow for Python Scripts to continue as long as the machine is up and running. After this, we implemented this as well, a script that runs indefinitely and always, listening to the MQTT Broker, and saving the data streams to our database.

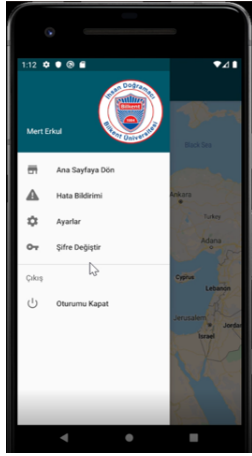
Database Connectivity outside Local Host (90.158.24.100): As specified earlier, MySQL supports TCP/IP connectivity, using a username, password, and other unique identifiers, one can request and pullup data from any MySQL Database established, in any remote machine, anywhere. However, this is only accomplishable given that the MySQL Database Master machine gives access for that database. So once again, using bash commands at our Ubuntu Machine in the remote servers, we "GRANT ALL PERMISSIONS TO MYDATABASE.* FOR 'beesmart'@* WITH IDENTIFIER 'beesmart2020'". This MySQL command grants us to communicate with the database from any IP address using the user-

name beesmart and password beesmart2020, as long as communication through the MySQL designated port 3306/TCP is opened. As a MySQL database is established, we already knew that there was a listener for the port 3306 readily available at the server, however, once again hardware part for the port was not established. An example screenshot for accessing the beesmart server and saving upcoming data to database is given in the Appendices section D.

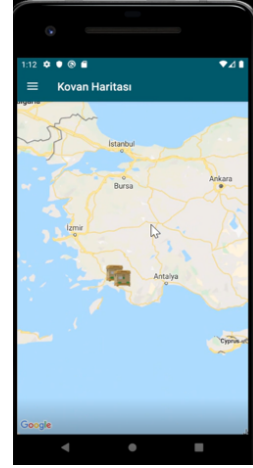
After negotiations with our firm and conducting intensive and comprehensive research on the topic, we learned that not all TCP ports are (like 1883) data-only port, such that they can be used for malicious purposes. As Karel's R&D uses a certain portion of the 90.158.24/24 subnet, and all addresses are connected, thus the entire Karel Domain is vulnerable if port 3306 and its listener is opened for universal access over the entire internet, even if we allowed for specific usernames and passwords, even if we somehow managed to implement a certificate that is necessary to bypass the firewall. As Karel's domain is not for all IP's, we decided to give permission to our static IP addresses (Home IP addresses) for external connectivity. Initially, after the company lowered down the hardware firewall, and we gave permission to our IP's through bash commands (software firewall was lowered down), we were able to communicate with the database. What we didn't realize initially, on the other hand, was the fact that our routers, in fact every team members routers, use DHCP (Dynamic Host Connection Protocol), such that whenever we shut down the machines that we use for development, or there is an issue with the internet connection, we are directly given a different IP address. For us to have distinct and completely static IP addresses, we needed to be in either a cooperation domain or buy monthly subscription to that unique IP using our machines' MAC addresses by communicating with our internet service providers. Initially, to bypass this connectivity issue, after discussions with both our Academic Mentor and our Company Mentor, we decided to develop in our respective local machines in the first stage, after that find a reasonable solution for connecting to our Ubuntu 16.04 Machine's database, which is and has already been updated through the Auto Subscriber Python Script.

After discussing with our coordinator, TA's, Prof. Karasan and Dr. Sarikan prior to the meeting and afterwards, there still existed the prior issue of the risk being prevalent in our 90.158.24.100 IPv4 server, as Karel R&D domain is completely exposed through the respective subnet as Karel owns a proportion of it. After the online CM3, as research indicated that cyber security would be an issue for Karel, and as opening up the servers for certain IP addresses (for port 3306/TCP connectivity) even if us developers bought static IP addresses from our ISP's, is not a generalizable solution, Dr. Sarikan reached an agreement with Vodafone, who is a regular provider for our firm, for our Ubuntu Machine to be transferred over to their domain with TCP ports 1883 and 3306 being opened for every IPv4 address in the Internet Domain, while us still keeping the servers for experimental purposes as well. Hence, remaining transfer operation was left as a progress requirement after CM3 meeting. However there is no need for configurations from start since Broker, Auto Subscriber and MySQL database are individually up and running already.

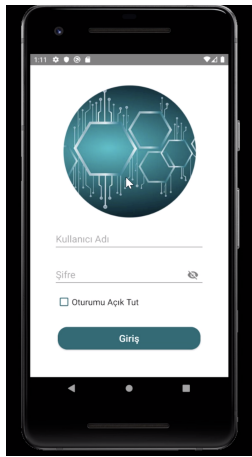
e. Android Application: As we could not implement the Embedded Software on our designated board for a while in the first phase, we decided to start working on the Android Application Interface initially in the meantime. We have designed an application with JAVA and using Android Studio. The emulator simulations and screenshots are given below.



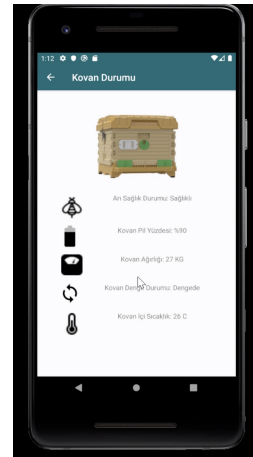
(a) Menu Option



(b) Hive Map



(c) Log In Page



(d) Hive Information Page

Figure 18: Screenshots on the Emulator for Application Designed

Our initial screen allows registered users to login to the application using their username and password. If desired, we also provided an option to remain logged in. The home page opens up with a map indicating the locations of the hives simulated as markers with the actual pictures of our hives. The main page also includes a navigation drawer menu for users to send error feedbacks through taking images and indicating the error it self in an editable text-field, changing their passwords, refreshing the connection and logging out. The Google Map API provided in the main page shows the hives and if the user decides to check the hive condition on a particular location, he/she clicks the desired hive and is directed to the status page of that particular hive. In the status page, the user can display the medical conditions of the bees, battery status of the module, orientation and GPS/GSM connectivity status,

with addition to exogenous variables such as temperature and moisture, whether they are optimal or not. The UI is always subject to be altered; however, for this stage, initial UI design is implemented as demonstrated in the figures above.

Up to CM2, we managed to implement the main UI & UX design requirements that our company required us to complete such as Hive Markers using GPS connectivity, Login Pages, displaying the conditions of the selected hives through clicking the respective marker, and these were advanced, because printing processes of our PCB layouts were delayed. We acknowledged that, until the server was established, no further improvements could be done on this milestone. As during the COVID-19 outbreak, we established main functionalities of our server, we also managed to implement both MySQL – Android Connectivity and MQTT Broker – Android Connectivity by establishing a built-in subscriber that listens to the Broker located in the server. The two new functionalities added to the Application, thus, can be broken down to these modules.

Android – MySQL Connectivity: As indicated earlier, after holding discussion sessions with our Mentors, we decided to work on local hosts for MySQL interface connectivity, as later connecting them to our servers would be relatively simple given that we use the exact mechanisms except the IP addresses, so locally, we established the same databases for Android and Web Applications, with same MySQL credentials and identifiers we used in the server side. After port 3306/TCP is opened, changing the in-app-integrated IP address would be sufficient. To make the user to obtain past data for every sensor output we completed and view them in a suitable fashion, whether it is a list or it is a graph (yearly, monthly and weekly as our time line selection), we required generate additional Android Activity pages for these. Now, we grant clients to view the last 100 data streams that the particular hive registers to our databases after clicking the respective hive using the Google Maps API we programmed earlier. The initial screen where the user displays the current hive conditions also utilize the latest data entry for that specific hive to display them immediately, if the user decides not to display the past data, and this was relatively useful, since we already derive the database upon user selection for a hive using the Maps.

Android – MQTT Connectivity: We also wanted to establish another activity, which provides our clients with the functionality to observe the hive in a live fashion, such that the data that is being registered to our database using the Auto Subscriber Functionality is also used here (Coded in JAVA instead, as it is Android Studio’s native language), without the registration steps. Paho MQTT also provides the users with the option to listen any broker after connecting through the 1883/TCP port, and display the traffic for JAVA, and this subscriber functionality was also implemented after selecting the hive to listen through the Maps API. We thought that this was also a good idea in order to empirically display the connection speed of that hive, as through live session listening, one can get the date and time the board has sent the data to the server, and after displaying that particular data, one can experimentally observe the average transmission and mirroring rates for Board-Server connection (transmission) and Server-Application’s Live Listener (mirroring). Once again, it is important to stress out the fact that the data that the live listener displays is also

displayed at the Auto Subscriber at the server, with addition to them being saved in the MySQL database, hence, after listening the hive in a ‘real-time’ fashion, the user can also display the data using the previous part, which is Android – MySQL connectivity. Example screenshots taken are provided in Appendix E.

f. Web Application Design: The solution strategy for the web application is to use well-established standards in the field of web development and use open source libraries in the scenario that 3rd part implementation is needed. In the company of this perspective, we decided to implement HTML based website. We implemented our CSS templates with addition to Bootstrap’s open source CSS templates. We used JavaScript for the visualization type of application purposes and PHP for the cloud server connection purposes. Google Maps API is used for the location services and MySQL is used for the backend part of the web application.

Up to CM2, we did not work on the web application as planned. Between CM2 and CM3, we completed the front-end of the web application as shown in the screenshots relating to web application using the tools described above. After the front end is finished, we started to work on backend part of the web application. Two problems encountered during this progress. First one is, we were planning to use MQTT type of approach to monitor live data similar to Android application; however, JavaScript does not support the native MQTT connection that uses TCP and PHP stopped its MQTT support. Hence, similar method with the MQTT- Auto Subscriber used at server side is adopted. The database is checked periodically in a time period which it is selected as 30 seconds in our application and if there is a new entry in the database resulted from the new MQTT transfer naturally, the analytics page is updated dynamically with the new entry. The second problem encountered during this vulnerability of the Karel’s data port. The port that supports MySQL transfer is not accessible by dynamic IPs since it creates a backdoor to Karel’s cloud server which creates vulnerability in terms of data protection. Local MySQL is used at development phase to not expose Karel’s cloud which is the same MySQL structure used in the cloud server. To wrap up, it is possible to say that changing the IP address used in the application for the MySQL database is left which is a trivial assignment but needs a change in Karel’s administrative side. The screenshots regarding web application design are provided via Appendix F.

4.2.2 Progress between CM3 and Present

The progress made after CM3 is again analyzed in detail in the below subsections. Since the hardware design was dedicated to be completed in the first phase, no further work was necessary. Although we have obtained the successful implementation of all sensory components in the Embedded Software Design part, further study on our MEMS microphone was essential and hence was one of our main focus. Likewise, we have finalized our implementations in Machine Learning part. Hence, the main progress was achieved in Embedded Software Design, Android and Web application along with server implementation.

a. Embedded Software Design: The implementations on ESP32 are consist of three main parts which are using MEMS microphone to collect sound in hives, preprocess the sound and obtain the mel-spectograms and finally sending the Mel-spectograms with then proper format to main board through UART connection. However, as explained above, the main reason to implement these features on ESP32 is memory limitations of main board where possible improvements will be discussed regarding the issue in the following parts of the report.

Collecting sound data with MEMS microphone: Since the MEMS microphone uses the I2S communication protocol, custom library of ESP32 for I2S was used to collect the data. The I2S was configured with 16000 sample rate, 4 buffers and 8 samples per buffer. The collected data length is 24 bits and two's compliment with MSB first. So, in order to get the normalized floating format of the data, the raw data obtained from microphone divided (by $2^{30} - 1$). The visuals for obtained sound data are provided below.

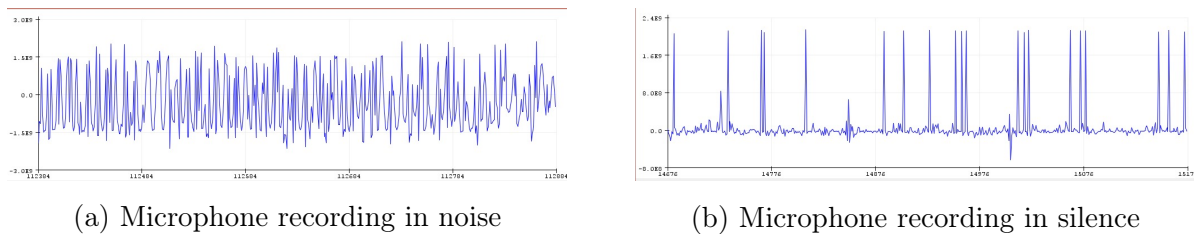


Figure 19: Sample Sound Recordings

Preprocessing the sound data: Since the overall strategy of the project is to use Mel-spectograms as inputs for neural network, the conversion to mel-spectogram procedure also implemented on ESP32. Also, this is the part that responsible of major storage complexity of the project hence the need for ESP32 which have a SRAM of size <216 Kbyte. The procedure can be break up to three parts which are obtaining the Short Time Fourier Transform (STFT) of the collected sound, creating the Mel Filter Bank and finally filtering STFT by matrix multiplication between STFT and created filter. The STFT of the sound calculated by using ArduinoFFT library. The first idea regarding the Mel Filter Bank was to calculate it for every process, however the filter has a memory requirement about 400 Mbyte which was overly exceeding the limitations of the ESP32, so filter calculated from the PC saved to Flash Memory of ESP32. Finally, matrix multiplication was implemented with three nested loops. The final version of the code needs about 200 Kbyte of dynamic storage where most of it is STFT of the sound.

Transmitting the Mel-Spectrogram: The calculated mel-spectrogram sent to main board with a URAT connection where the implemented protocol is as given below. Also used baud rate is 115200 for this communication:

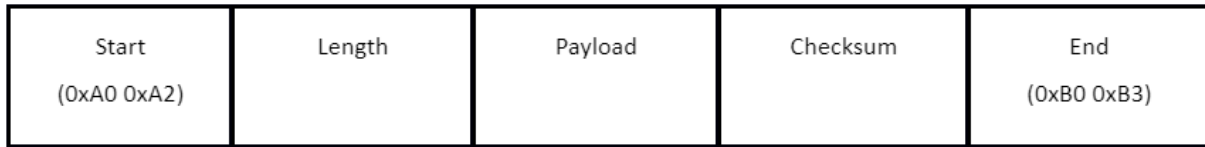


Figure 20: Message Structure

Every payload consists of 128 float data which is a complete row of calculated mel-spectrogram with since maximum payload is 1048 Bytes. So, the overall process of sending complete mel-spectrogram takes 16 serial messages and there is a 250 ms delay between each row in order to overcome possible issues regarding the excessing baud rate.

b. Application to Cloud Connection: Prior to CM3, because of cybersecurity vulnerability of the Karel Cloud’s infrastructure, which are all connected to a central super node as a subnet, we couldn’t establish port 3306/TCP that was essential and a necessity for IoT connectivity. Right after CM3, with agreement of our company and Vodafone, we managed to move our cloud remote machine to another IP address, 46.221.16.179, which had additional port controls already set up by Vodafone, we managed to open both the hardware socket and software socket for MySQL 3306/TCP connection. Since our remote machine and cloud was transferred to another IPv4 address, some already established privileges regarding MySQL rendering and MQTT broker needed to change. We initially had the broker service reinstalled manually to the server, as after the transfer, only the SSH port was functional for our engineering purposes. After logging in using SSH, we set the broker with exact same privileges and opened the port 1883/TCP for MQTT connectivity with our board. After MQTT messages were successfully rendered and were being received correctly via our board, we transferred the already established auto subscriber functionality that was implemented prior to CM3, as this functionality was implemented as a Python 2.7 code, using the right bash commands such as chmod and nohup, the server started to save the data to the database located in the cloud servers that was sent through the MQTT topic “beesmart/database”. To complete the IoT tasks fully, we now required to give privileges to any connection with correct password and username from any IPv4 address through MySQL, for our applications to successfully pullup data, simultaneously and synchronously, where the cloud server was the catalyst for the hardware autonomy of the hives and user experience through the applications. Once again, for the CM3 demos, we had already implemented database pullups in our applications, we just needed to change the IPv4 addresses that were hardcoded in the applications. After this was also successfully replaced with the new IPv4 address, we decided to dedicate the prior root name server “beesmart.karel.com.tr” to our newly established server as well. After DNS arrangements were also completed, by using a custom built MQTT subscriber that listened to the server for the topic “beesmart/database”, and simultaneously running the Web and Android applications to test the board – server and server – application connectivity, MQTT broker and MySQL functionality, auto subscriber functionality, we opened the interface that showed our hive and utilized the live

listener functionalities. After seeing that all components were working simultaneously and synchronously, we dedicated our time and energy to implementing an alert system.

We had implemented an SMS alarm system for threshold values for beekeepers to directly intervene with productivity decreasing situations such as temperature changes, stabilization of the hive and humidity conditions in CM2 as we didn't have a server to do this task for us. With suggestion of our company mentor Alper Sarikan, we decided to reimplement this functionality in our cloud servers, instead of our board which was initially responsible of sending SMS through its GSM module through AT commands, we have implemented this functionality to our auto subscriber. After parsing the JSON dictionaries that are sent to the server through MQTT and GSM, and prior to saving them to our database, our auto subscriber checks whether the values received through our hive are in the non-risky range, if any of them possesses a risk that might affect hive productivity, the server sends an email to beesmart client email address that we created. So, our server, not only functions as a monitor system through the applications, but also serves as an alert system as well, which concluded our implementations in the network and IoT department of the project.

c. Android Application: After CM3, as indicated earlier, because of switching to a fully functional and communicative server, connecting the Android Application to the server was relatively straightforward. In CM3 demos, we have shown that the application can pullup data from the local MySQL database, which we filled up with random data, and show them in a graph format, as well as listening to the hive live and monitoring the data both through MQTT messages received from our board and through a database, which was built locally at the time as 3306/TCP wasn't allowing connections from any other IPv4 address. A functionality that formed the main activity page of the Android Application wasn't implemented during CM3, which was to render the GPS connection from the servers as well. During preparation for CM4, we changed the table structure of the MySQL database, so that latitude and longitude of the hive was also sent as a JSON dictionary over MQTT, which is added to the MySQL table via our auto subscriber. After correct login to the application, the Android Application now connects to the database immediately, and places hive shaped markers on the map for user interaction, which different GPS locations that are in the MySQL table. After clicking on the selected hive, the application pulls up the data using the latitude and the longitude of that particular hive, and in the live listener interface, displays the last entry to the database, in addition to displaying the past data only for that selected hive. It can be said that, because of accurate and generic code engineering in the process of creating the application, the remote server change didn't alter much of the functional aspects of the application, and instead of replacing or removing any important requirements, we added new functionalities and increased the robustness of the application during the time between CM3 and CM4. Overall, after CM3, we increased the robustness and stability of the application with the establishment of new remote machine, we added location pullup with addition to other pullups that were implemented earlier, the UI part didn't experience major changes.

d. Web Application: There are several changes in terms of infrastructure of the web application. As it is stated in the previous CM, the cloud server and web application connection could not be achieved due to security concerns of the Karel server. This problem is resolved by establishing our remote machine to Vodafone network and giving necessary permissions for the port used for MySQL data transfer. Hence, the first major change in the web application was using the Karel's remote machine located in Vodafone's network instead of our local machines. As an additional note, since we finished the integration and configuration of the other components and our board, we are now monitoring the actual live data transferred from our board to the remote server such as temperature, humidity, tilt and hive condition predicted by edge learning using the sound data obtained by microphone. There are also two addition to the functionality of the web application. First one is displaying the unique IMEI number associated with the hive. The second update is instead of just showing the health condition of the hive, it also displays the occupancy of the hive based on edge learning prediction. To sum up, the web application is ready and functional in terms of design requirements and needs testing in neutral setting to observe new day-zero bugs for future updates and improvements.

4.2.3 Task/Work Packages That Could not be Finished

One of our major wills while establishing the foundation of this project was to implement regressive analysis of the hive weight and the honey amount as well as conducting classification analysis of the hive medical conditions and the number of bees inside the hive. In order to accomplish this, we planned to implement a self-trained labelling using the weight sensors and implement a biplanar Neural Network which receives same Mel spectrogram images as the inputs, which are the processed sounds that are received from our I2S microphone, outputs both the classification results and the regression results in parallel and in a single forward propagation, similar to the state-of-the-art CNN model of Google's Inception V3. However, as this required for us to travel to Köyceğiz, which we couldn't do because of COVID-19, this work package was neglected. In addition, we also wanted to enhance the performance of our convolutional neural network model, which was responsible of the classification analysis, by using the dataset we would have formed using the same I2S microphone that we have shown in the demo, for precision purposes. This was also not possible, because of the ongoing pandemic. Besides these, as denoted earlier, we had 6 milestones, because of our strategic planning that we conducted in the first semester that required us to work primarily on software in the second semester, outbreak didn't prevent us to complete our major work packages and didn't disallow us to build a fully functional edge computing smart hive that is capable of IoT communication.

4.2.4 Risk Analysis & Backup Plan

From the beginning, the main concerns of the project are the coverage of microphones with honey. This will result as interrupted in sound processing. In addition, data deliv-

ery can be interrupted because of the dependence of the GSM connection to hive location. Further, appropriate UI design for all users are necessary along with the risk of Application-Cloud connectivity delays. In addition to cloud-application connectivity risks, there are two possible events. First one is the change in the stability of the tools we are using in the applications which can be solved by period checks to web application that does not create a problem in terms of project. The second one is a possible server failure of data transfer failure between the board and server which will affect the information displayed in the web and android application. However, this is not a concern for the web application as it performs a slave kind of role in master-slave relationship. At this point, in order to prevent any setbacks in the project regarding these risks specified above, several precautions and B-Plans have been generated. Firstly, for the risk arising from the use of microphones, some specialized materials to cover is the major recovery plan of the project. The analysis for the possible materials that can protect microphones from propolis was made and some of them is ordered by our company KAREL during this process. Second major backup plan specified regarding the GSM connectivity is the adjustment of network usage. Since the data transmitted consists mainly voice and sensory data, 2G network will be chosen over the 3G network to allow increase in data transmission reliability. Next backup plan regarding the user platform is the implementation of cross-platform applications. Hence, the application is compatible with every possible device that end-user aims to obtain information with. Compared with the purely native application that is not compatible to every environment, the implementation will prevent from further complications. After cloud server establishment to the new IPv4 address 46.221.16.179, we have changed the MQTT message format and MySQL database table entries to also include the latitude and the longitude of the hive location, which also required an accurate GPS connectivity. GPS functionality is important, such that if the GPS antenna cannot receive NYMEA broadcasts from 4 different satellites, the algorithm cannot compute the exact location of the hive and directly assign the location to be gyrocenter of the world, with longitude and latitude of 0.00000 and 0.00000 to be exact. This causes the auto subscriber and the database subroutines to assign and save the hive location in the Southern Atlantic Ocean, and since the applications pullup the data from our MySQL database located in the cloud, it causes the hive marker to appear in the ocean. Reliability of the GPS sensor is thus another risk in this project. Our plan B for this case was to add a pullup protocol in the applications that utilizes the latest non-zero GPS entry and to use the latest sensor entry in the database as sensor outputs are reliable, so that the marker shows the last known location of the hive with the most recent sensor data, if the user wishes to display and monitor the hive even if the GPS sensor provided false information.

5 Results, Discussions and Future Directions

5.1 Results

5.1.1 Hardware Integration to Hive

After the design, production and test phases of main board, other sensors and components that are not included on board were connected to main board. Specifically, sensors mentioned before such as temperature, tilt, humidity and weight were connected to main board. Further, MEMS microphone is connected to ESP32 and two boards are then communicating with each other serially. This communication is made with UART protocol and a settled format where the accuracy checked through by sending a test array and printing the received array through serial monitor for main board. Then final software of both boards flashed to them. Finally, a power circuit is implemented to convert 15 Volt output of battery and main board powered by this circuitry. All of the other parts of the project including ESP32 were powered by DC outputs of main board. The overall designed hive is presented below.

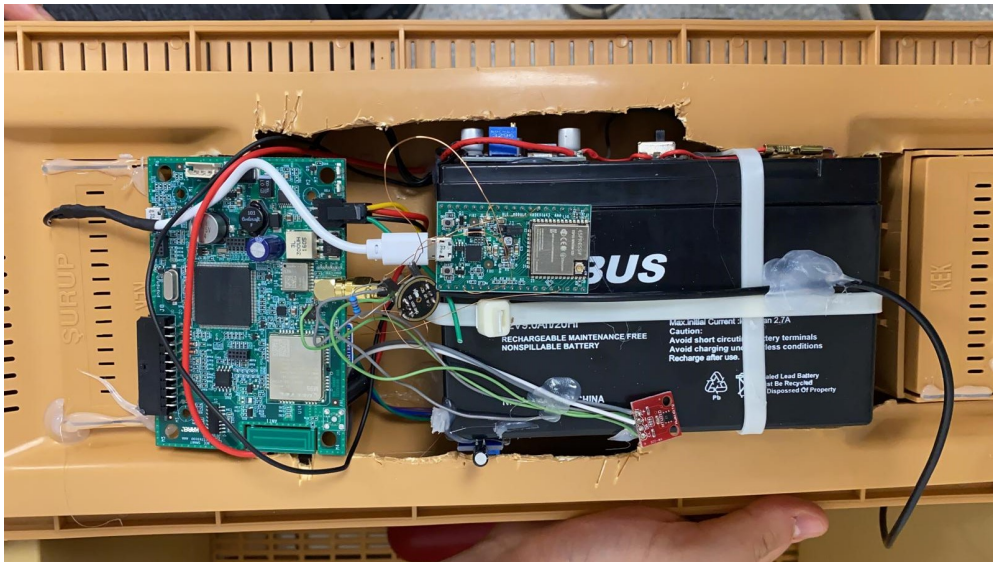


Figure 21: Completed Circuitry on Hive Lid

From the Figure 21 above, the battery system is powered with can be seen. On the side of the battery which is at the upper part in the figure, the power circuitry is attached which is controlled with a switch next to the circuitry. On the left, the designed board can be noticed, along with its sensory connections. The ESP32 module is also attached on top of the battery.

5.1.2 Hardware-Cloud-App Connection

After the individual implementation of cloud and apps, we were able to open port 3306/TCP which allowed our IoT components to be merged. Also using the GSM connection of main board, the hardware part also connected to server to send the measurements

and predictions. With the connection of individual parts, board can send MQTT packages, which include the ML Predictions, Sensory Readings, Location information via GSM to Servers. These servers parse the data and save them to MySQL database and applications can connect to the server and receive parsed data from the MySQL built in the servers. Moreover, an extra functionality added to Auto Subscriber subroutine to check critical values, send email to check tilt, humidity, temperature notifications for enhancing the instant interventions. Below, a sample e-mail delivery is depicted. Inbox is provided along with two sample notifications which are when above-threshold temperature or tilt on the hive observed.

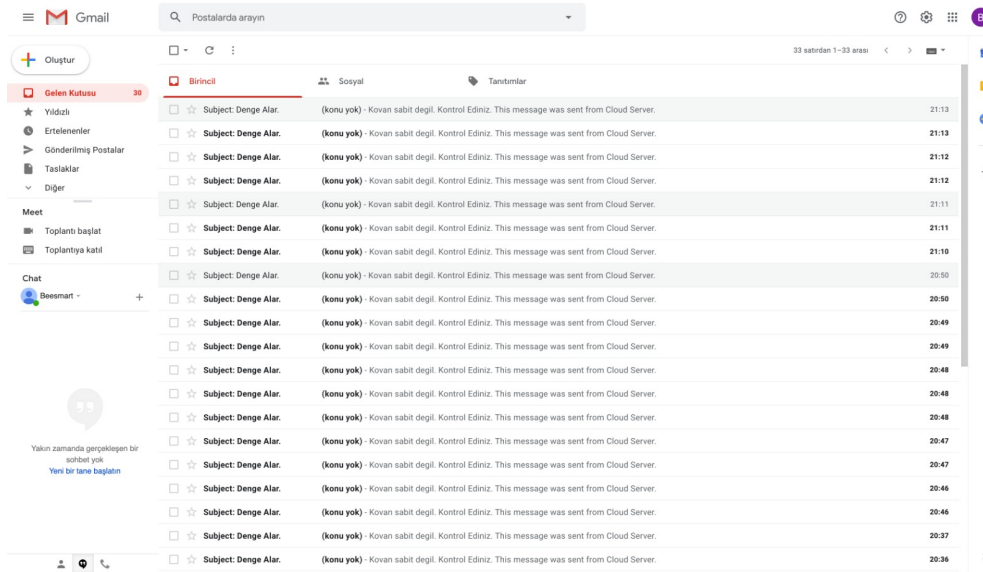


Figure 22: Screenshot for Inbox

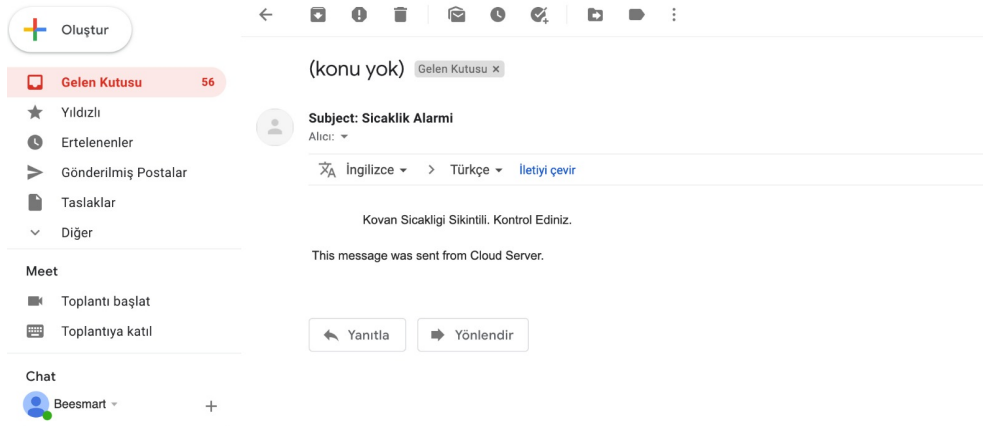


Figure 23: Example Temperature Notification

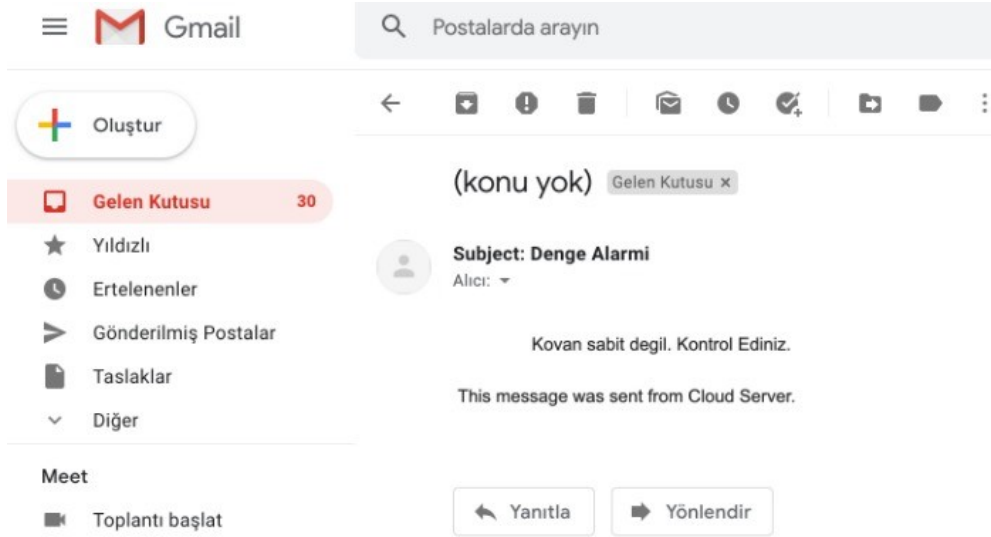
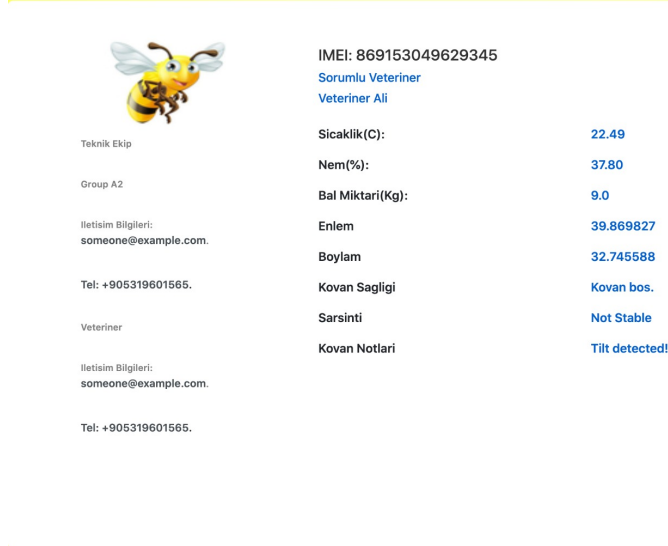


Figure 24: Example Tilt Notification

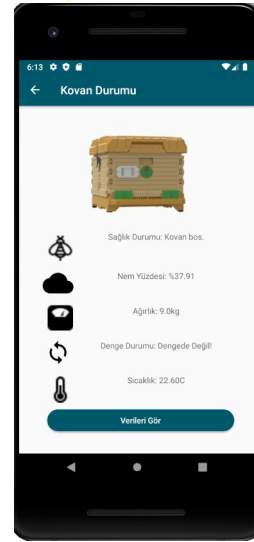
5.1.3 Sensor Measurements and Machine Learning Prediction

Both sensor measurements and results of processing of sound observed from Android and App applications after testing the performance of the Hardware-Cloud-App connection. The measurements of all sensors are accurate with calculations and expectations. The temperature and humidity measurements of board were compared with known values from air conditioning unit of KAREL department. Also, the weight measurements were same with the weight of empty hive and overall hardware of the project. Finally, in case of tilts or temperature or humidity conditions exceeding the threshold values in the hive, board sent instantaneous alarms to server which were observable from both apps and notification mails. The screenshots were provided in Figures 22, 23 and 24.

On the machine learning part of the project, test sounds that separated form training data used to see the final predictions of overall project where for all 5 different labels, the predictions where correct. All of different situations for prediction and measurements of sensors can be observable the figures given below via both Android and Web Application. In order to demonstrate the accurate operation of prediction, the sounds separated for testing were used. The labeled sounds are fed into the system using speaker inside hive. For the sample health conditions, following results were obtained which is observed via both web and android applications.

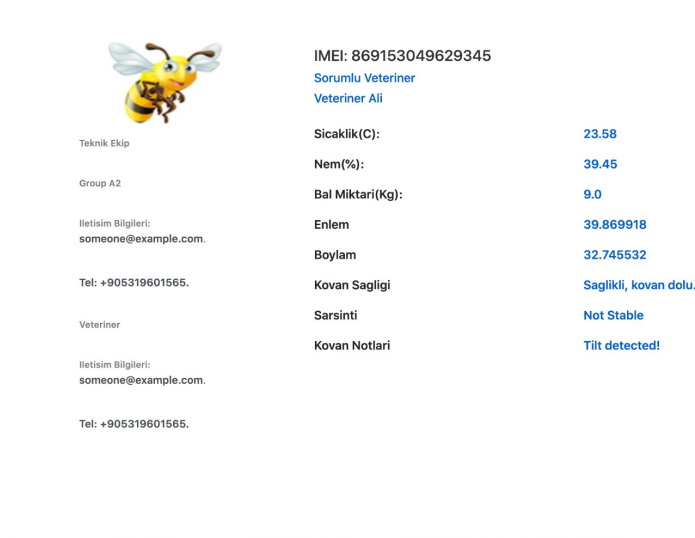


(a) Hive Empty, Web

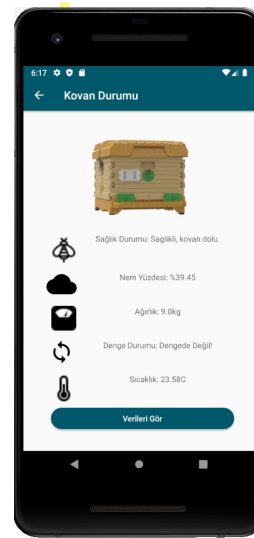


(b) Hive Empty, Android

Figure 25: Results when no sound provided




(a) Hive Healthy & Full, Web



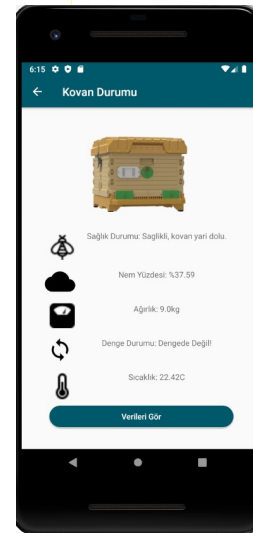
(b) Hive Healthy & Full, Android

Figure 26: Results when healthy and full hive recordings provided




Teknik Ekip	IMEI: 869153049629345
Group A2	Sorumlu Veteriner
iletisim Bilgileri: someone@example.com.	Veteriner Ali
Tel: +905319601565.	Sicaklik(C): 31.12
Veteriner	Nem(%): 50.68
iletisim Bilgileri: someone@example.com.	Bal Miktari(Kg): 9.0
Tel: +905319601565.	Enlem 39.869883
	Boylam 32.745545
	Kovan Sagligi Saglikli, kovan yari dolu.
	Sarsinti Not Stable
	Kovan Notlari Tilt detected!

(a) Hive Healthy & Half-Full, Web



(b) Hive Healthy & Half-Full, Android

Figure 27: Results when healthy and half-full hive recordings provided



Teknik Ekip	IMEI: 869153049629345
Group A2	Sorumlu Veteriner
iletisim Bilgileri: someone@example.com.	Veteriner Ali
Tel: +905319601565.	Sicaklik(C): 22.96
Veteriner	Nem(%): 38.47
iletisim Bilgileri: someone@example.com.	Bal Miktari(Kg): 9.0
Tel: +905319601565.	Enlem 39.869612
	Boylam 32.745557
	Kovan Sagligi Sagliksiz, kovan dolu.
	Sarsinti Not Stable
	Kovan Notlari Tilt detected!

(a) Hive Unhealthy & Full, Web



(b) Hive Unhealthy & Full, Android

Figure 28: Results when unhealthy and full hive recordings provided

As it can be seen from above results and live-demonstrations, the android and web applications agrees on the hive health status predictions as expected. Further, machine-learning model is able to identify the test sounds in the correct category, which strengthens the accuracy in general. It is important to note that tilt was present in the system since it was our motivation to observe in addition to sound predictions and the hive was inclined to capture tilt.

Finally, accuracy of GPS module was verified from web and android applications as follows.

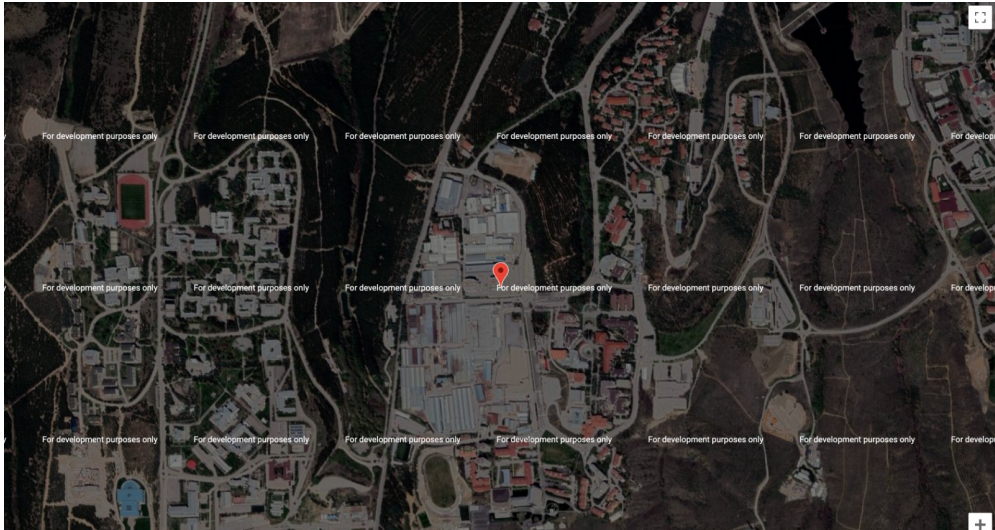


Figure 29: Location provided, Karel Electronics, Cyberpark

We can see the output of Android application from below. First, general locations of the hives are mapped in country wise. Although a single prototype is tested in here, under Karel Company's supervision in Cyberpark, the sample hives are also located in Muğla, Köyceğiz since it is the original operation location.

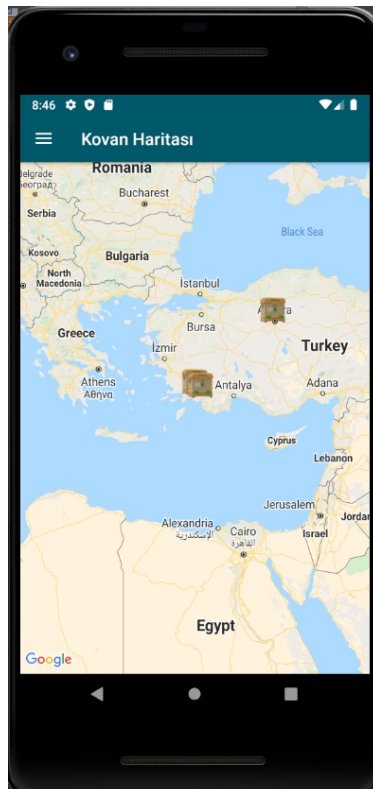


Figure 30: Location provided, Turkey Map

Next, a zoomed-in version is presented in order to verify successful data transfer of the GPS module. The Cyberpark is again can be viewed in detail. This is what we have expected

naturally, since web and android applications are completely consistent and receiving the same data messages.

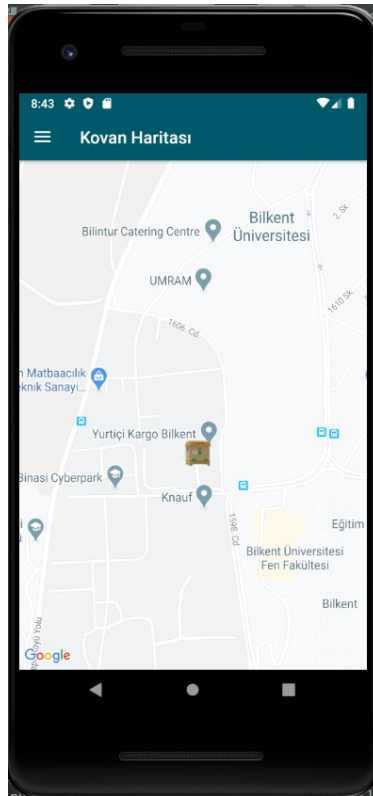


Figure 31: Location provided, Karel Electronics, Cyberpark

In order to see the operation of other sensory components and graphics that will be provided to users via web and android applications, a sample one-year sensory information was arranged. An example temperature, humidity and weight information on a weekly basis via web application is presented below. Again, for humidity information, weekly monthly and yearly based graphics are presented in android application.

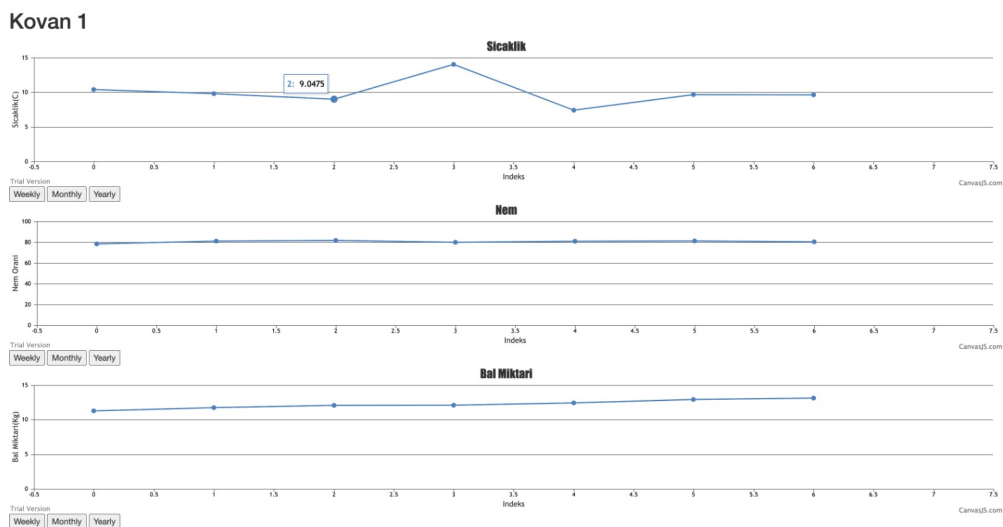


Figure 32: Web Application Graphics

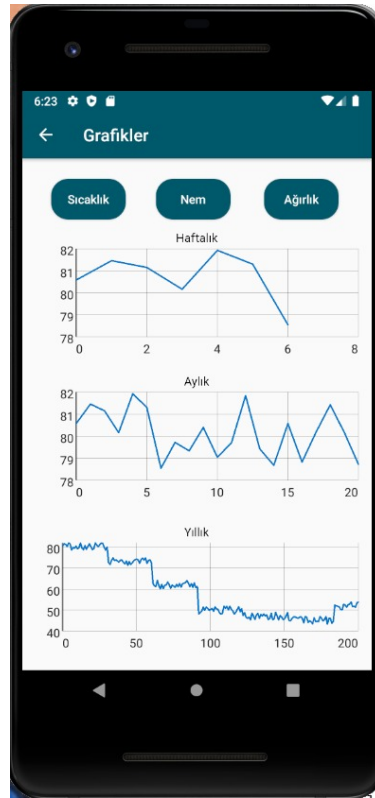


Figure 33: Android Application Humidity Graphics

5.2 Discussions and Lessons Learned

Finalizing the BeeSMART project, remarkable learning outcomes are obtained from the project while all of the goals are attained. Overall, we have a smart hive which is able to analyze the bee sound coming inside the hive with nearly perfect accuracy which is equipped with crucial sensors in terms of beekeeping sector requirements. We also have a functional web and android application where the user can continuously monitor the live conditions of the hive utilizing from the data obtained from the moisture, temperature, tilt and weight sensors. Despite some hesitance points, each step of the project plan went well and we were able to successfully achieve what we proposed in the beginning of the project. Hardware part went well since we obtained a multi-functional PCB on which the hardware of the project was implemented. Machine learning and especially edge learning section was also remarkably successful since sounds coming inside the hive are correctly classified with the related label. For instance, our model can recognize whether the colony is healthy or not or the colony is inside the hive or not. Likewise, server connections are successfully established between the applications and real-time monitoring is possible. However, the group was stopped up due to unexpected consequences at some points which were handled later.

Coronavirus pandemic was the most remarkable surprise for the group in the spring semester which unfortunately decelerated the group since the union of the project group was not possible for at least two months. Technically, memory concern was one of these consequences since the memory limitation because of the RAM of the board was a big

challenge for the group. We needed to employ an additional ESP board for sound pre-processing purposes in order to ease the memory on the main board. We also required to reduce the layers in the neural network which is equivalent to reduce the number of parameters in order to ease the memory again. In addition to that, ROM of the board was used for the same purpose. I2S microphone was another unexpected challenge for the group and cost at least three weeks which resulted with an unexpected delay. A temporary Raspberry PI was used at the first phase and microphone was successfully implemented in the spring semester with the help of the engineers in KAREL where we later realized that the problem was the wrong configuration of the clock in the microphone. As mentioned above, we lost several weeks due to hardships faced on implementing microphone. Employment of the ESP32 board might be initiated earlier. What is more, tilt sensor and weight sensor could implemented and tested earlier so that we might detect the problems regarding their output earlier. On the other hand, the shift to Vodafone subnet might be done at the first stage of server implementation. Terminally, the board we configured might be designed in order to provide more RAM so that the need for ESP32 board can be easily eliminated. To continue, we have gained a great experience with this project and have plenty of recommendations to give others who might involve in a similar project in the future. First of all, initial acceleration of the group was crucial and we were ahead of our calendar from the beginning of the first semester. Our project was spread to two semesters but a potential project in the future may require more time. Availability of time shouldn't relax or misdirect any individual who takes part in the project. Our group worked really hard from the beginning, but still had problems and drawbacks at some points which resulted with some delay. For a multi-disciplinary project like our project, we would recommend to complete the hardware part of the project first. Till coronavirus outbreak, hardware part (excluding the microphone) was completed which was a significant advantage for us and we were able to make progress on the outbreak since web-application and android application including server implementation did not require any physical attendance. Furthermore, we would also advise an effective use of the debugger since it was really handy while solving the problems existing in our code.

Throughout the project, everything was not in our control. Coronavirus Pandemic caused some communication problems between the group and the company since our company shifted to home office. These communication problems caused some delay for the server part, since the central machine for the server was located in the company and there was no access. Printing and soldering process of our boards were also took longer time than scheduled and caused shifts in the project calendar. Taking into account the surprises, we were expecting lots of problems with our board with the first arrival. Members Mert Erkul and Elif Ceren Fitoz had no previous knowledge about the PCB design and they designed the board with the tutorial led by the company. However, the board was perfectly functioning as we desired and planned which surprised the group a lot. We anticipated to arrive Muğla, Köyceğiz and put the hive ourselves and planned to employ real bee sounds for testing purposes. It was also a motivation for the group since we would be able to observe our own project on the location where it would be serving. As mentioned previously, we

completed the most significant section of the hardware implementation initially which we consider as the most challenging part. This allows us to avoid any kind of chaos during the project especially during in pandemic.

Furthermore, we faced the difficulties of edge computing section in mid-stages of the project so that we had time to analyze and convert the TF lite library and implement it to our board. Moreover, we were aware of the limitations of the RAM located in our board which led us to use the memory efficiently from the very beginning of the implementations. We lost a considerable amount of time while importing TF lite library into our board. Library did not have a useful import tool so we implemented the library manually by understanding the most basic logic behind its own configuration. There were dozens of classes and methods in the library and automation of importing TF lite would save remarkable time for us. As mentioned above, our group lacked of hardware skills such as PCB design, sensor implementation or I2S interface usage. So, the group needed tutorials and a detailed research and lots of trials on these components. Hardware part of the project was a great achievement for each individual in the group and now, all members have proficiency on the hardware components which are used in the project.

Even though all of our group members have profound knowledge in the area of Machine Learning and Artificial Intelligence, with unlimited hardware and cloud computing resources available in the industry at the moment with powerful CPU's and online GPU's, during our experience in course projects, efficiency was never the initial goal for us, it was all about performance. During the implementation of the edge learning aspect, both efficiency and performance became our primary concerns, and conducting complex operations with such a simple MCU was also one of the greatest achievements and challenges faced in this project. As taking courses and obtaining hands on experiences in ML and AI tasks have already shown us that data augmentation and preprocessing is the cumbersome aspect in such projects, dealing with space requirements in these processes was never an issue prior to this project and these tasks became even more cumbersome, because of space requirements, we implemented some of the complex operations using the built in C and Arduino libraries, which were individually and exhaustively challenging, but taught us a lot and enlightened us in terms of the approach industrial machine learning projects are leaning towards.

5.3 Future Directions

Even though we can get accurate results through playing the test sounds we received that were recorded and labelled by a vet in the first semester, these results are theoretically accurate. To build a completely robust prediction system, saving and labelling the sound data through our own custom board after going to Köyceğiz would idealize the results. In addition, when starting this project, one of our main tasks was to regressively label the data and also predict the honey amount through sounds. Because of COVID-19, our trip to Köyceğiz was cancelled, so this subtask was neglected for completing the project, a future direction would also be to implement this task, as we couldn't receive the self-labelled data,

it was impossible for us to complete this task.

As this project will be used by our company, increasing the efficiency of the e-mail alarm system to serve multiple clients, adding developer functionalities to the applications in order to overwrite database threshold values and testing the robustness of the cloud databases with multiple hives under different conditions to enhance autonomy of the project can also be considered as a future direction, even though the currently implemented systems are written generically, because of the COVID-19 lockdown and restrictions, testing in an actual environment couldn't be completed. Locating the deficiencies under the current circumstances were therefore impossible and the project was fully functional under the experimental environment we worked on, receiving and perhaps building a feedback system via the applications might also help in the generic case where users are able to provide feedbacks according to the deficiencies and drawbacks might be a implementation that could increase the gradual development of this project.

6 Equipment List

The complete components used up to the current state of the project along with some other components ordered for future implementations are given below in the table. Further observations on the table yields the total cost spent up to the current state of the project is nearly 8400 TL. However, it should be noted that some of the components written below is used during testing purposes when group members were gaining familiarity to the subject. Thus, during the final version itself, a more compact list was obtained. This can be viewed in Table 8.

Component/Equipment Name	Individual Price (TL)	Amount	Total Price (TL)	Method to Obtain	Status
I2S Microphone (SPH0645LM4H I2S MICROPHONE)	39.99	8	319.96	Company Provided	Attained
GSM Module (QUECTEL MC60CA GSM-GPRS-GNSS)	92.99	2	185.98	Company Provided	Attained
GPS Module (GY-NEO6MV2 with Antenna)	52.31	2	104.62	Company Provided	Attained
PCB Boards	320.00	8	2560.00	Company Provided	Attained
Load Cell with HX-711 Amplifier (RLC2E03-0001)	17.99	8	144.00	Self-Bought	Attained
Temperature & Moisture Sensor (SI7021)	47.08	1	47.08	Self-Bought	Attained
Tilt Sensor (RLA1B46-1750)	5.62	4	22.48	Company Provided	Attained
Hives (Apimaye Ergo Hive)	800.00	3	2400.00	Company Provided	Attained
9Ah Accumulator Battery (TTEC 9Ah)	120.00	3	360.00	Company Provided	Attained
ESP 32	56,66	1	56,66	Company Provided	Attained

Table 8: Compact Equipment list

Component/Equipment Name	Individual Price (TL)	Amount	Total Price (TL)	Method to Obtain	Status
USB Mems Microphone (TRU22191 GXT212 MICO USB MICROPHONE)	129.99	1	129.99	Self-Bought	Attained
I2S Microphone (SPH0645LM4H I2S MICROPHONE)	39.99	8	319.96	Company Provided	Attained
GSM Module (QUECTEL MC60CA GSM-GPRS-GNSS)	92.99	2	185.98	Company Provided	Attained
GPS Module (GY-NEO6MV2 with Antenna)	52.31	2	104.62	Company Provided	Attained
PCB Boards	320.00	8	2560.00	Company Provided	Attained
NXP LPC1812 MCU	32.99	8	264.00	Company Provided	Attained
Turk Telekom Multi-Sim Card with 10 GB Data	99.99	1	99.99	Self-Bought	Attained
Load Cell with HX-711 Amplifier (RLC2E03-0001)	17.99	8	144.00	Self-Bought	Attained
Temperature Sensor (DS18B20)	19.99	2	39.98	Self-Bought	Attained
Tilt Sensor (RLA1B46-1750)	5.62	4	22.48	Company Provided	Attained
Raspberry Pi 2 Model B	296.11	1	296.11	Company Provided	Attained
Moisture Sensor (HR202)	8.83	1	8.83	Self-Bought	Attained
Breadboards	15.00	3	45.00	Self-Bought	Attained
Hives (Apimaye Ergo Hive)	800.00	3	2400.00	Company Provided	Attained
Three-axis Accelerometer	120.00	2	240.00	Company Provided	Online Ordered
Debugger (STLink V2)	66.34	1	66.34	Company Provided	Attained
9Ah Accumulator Battery (TTEC 9Ah)	120.00	3	360.00	Company Provided	Attained
Samsung Galaxy S4 (I9500)	1000.00	1	1000.00	Self-Bought	Attained
ESP 32	56.66	1	56.66	Company Provided	Attained
Temperature & Moisture Sensor (SI7021)	47.08	1	47.08	Self-Bought	Attained
TOTAL			8391.02		

Table 9: Equipment list

Thus, we can see that production of the finalized project costs nearly 1552.64 TL, considering the components will be used once. Again, this can be viewed in detail from Table 8.

7 Appendices

7.1 Appendix A

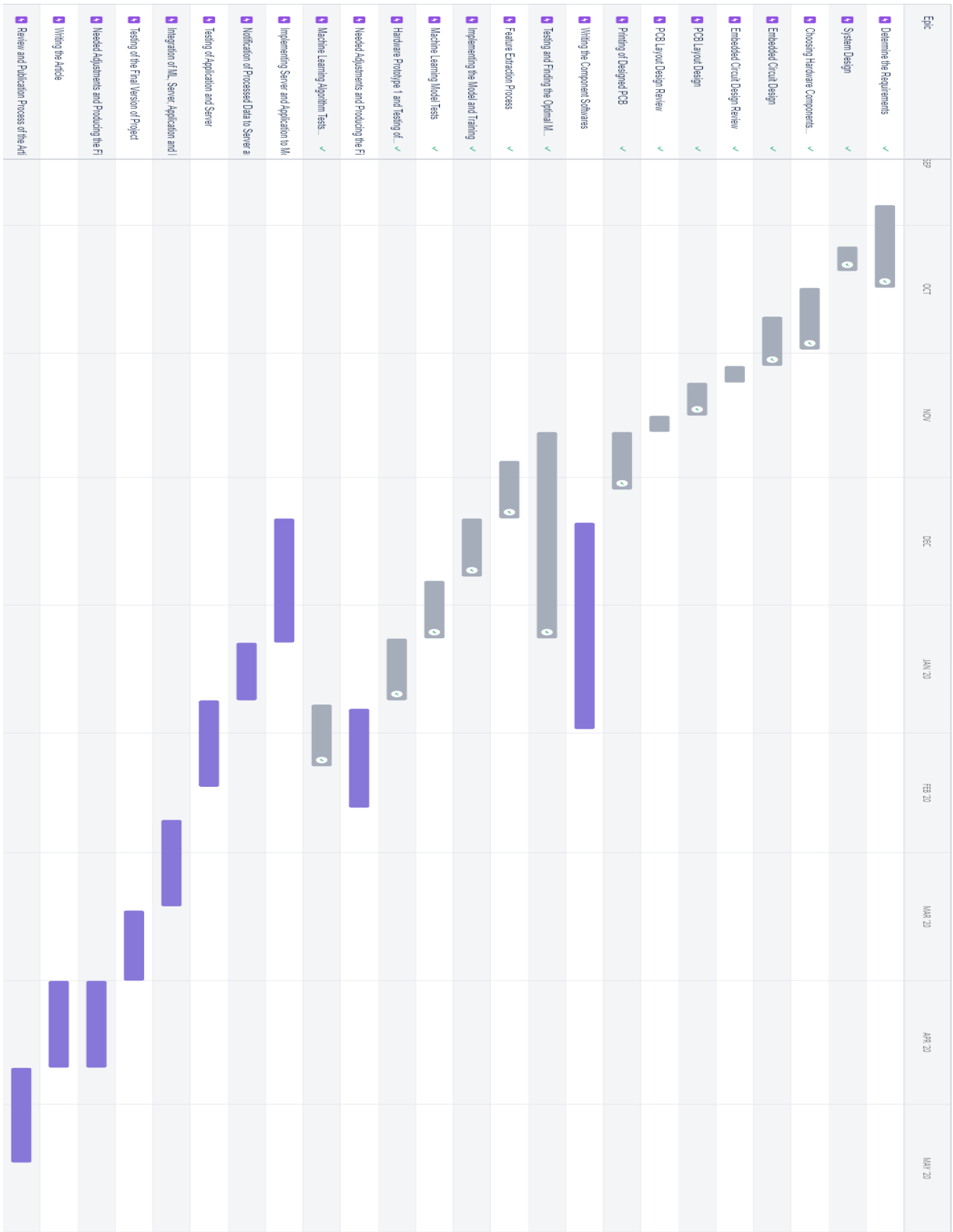


Figure 34: Previously Modified Timeline

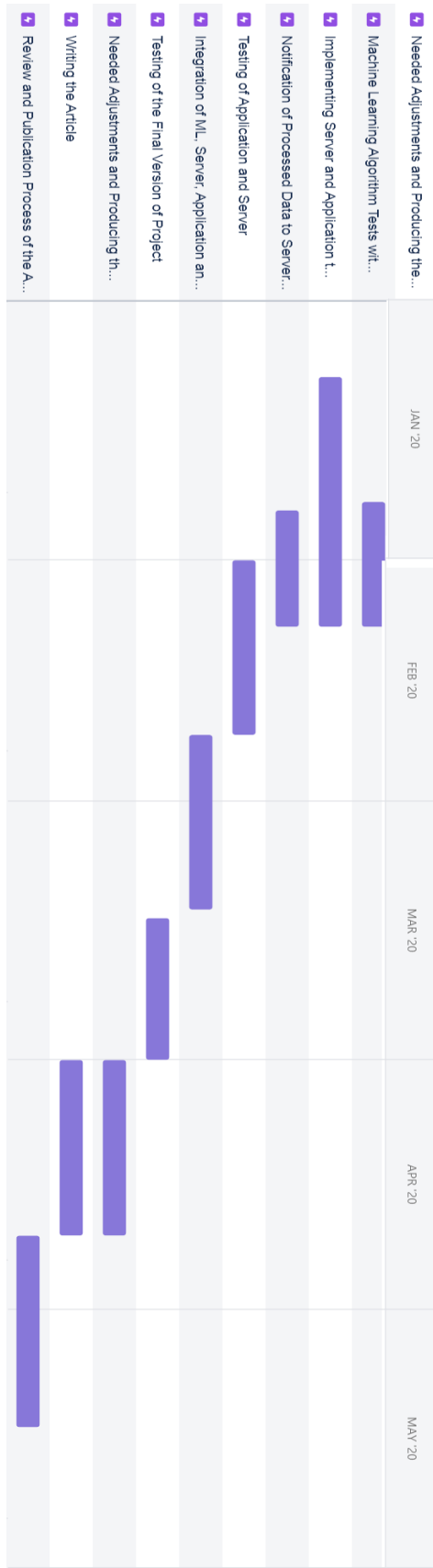


Figure 35: Previously Modified Timeline, Continued

7.2 Appendix B



Figure 36: Last-Modified Timeline

7.3 Appendix C

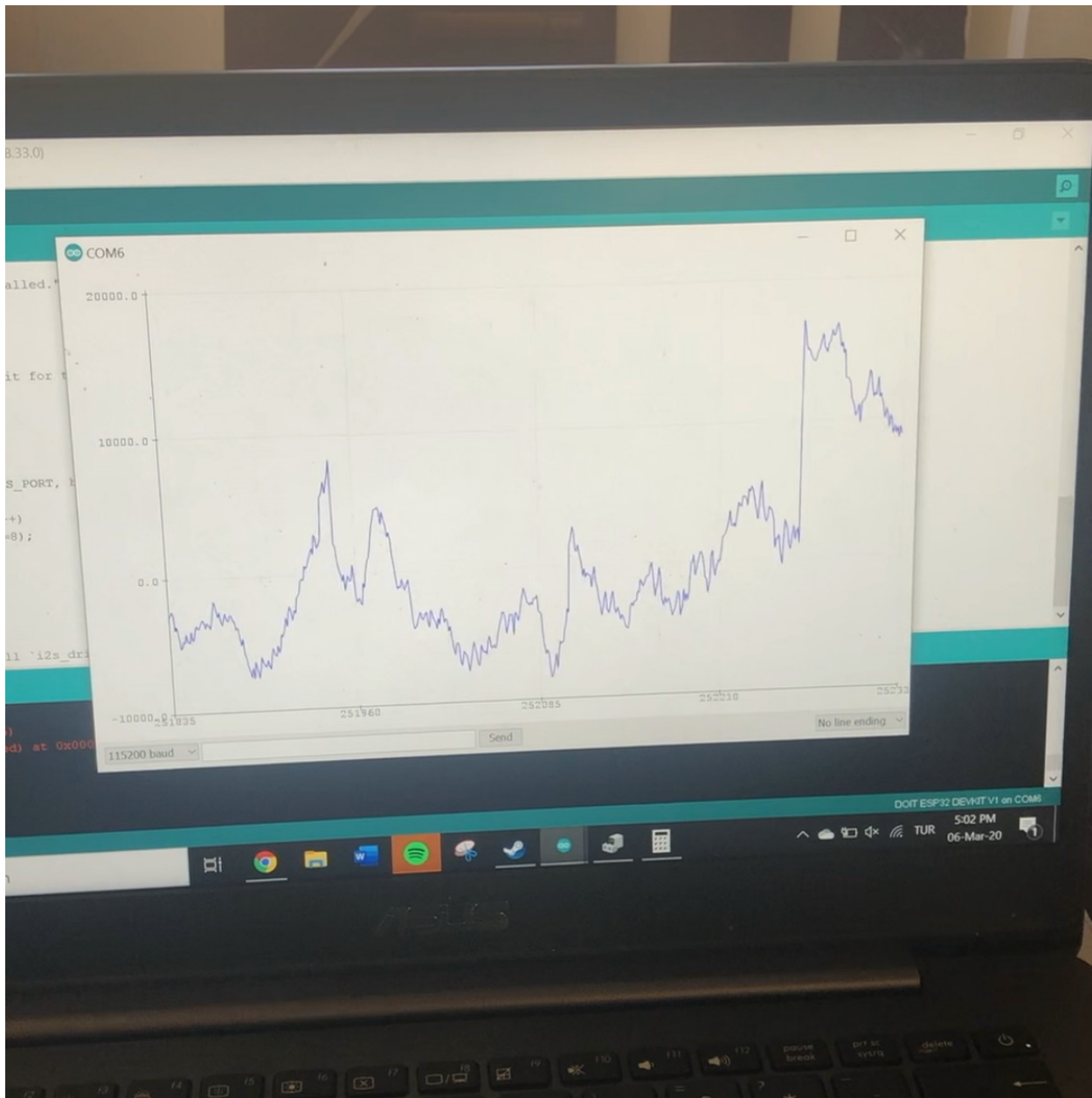


Figure 37: Sound Data Obtained from Microphone Sensor

7.4 Appendix D

```

beesmart@beeserver: ~
login as: beesmart
Authenticating with public key "rsa-key-20200222"
Passphrase for key "rsa-key-20200222":
Last login: Sat May 2 16:13:39 2020 from 176.43.126.61
beesmart@beeserver:~$ systemctl status mosquitto
● mosquitto.service - Mosquitto MQTT v3.1/v3.1.1 Broker
   Loaded: loaded (/lib/systemd/system/mosquitto.service; enabled; vendor preset
   Active: active (running) since Fri 2020-04-10 06:32:49 +03; 3 weeks 4 days ag
         Docs: man:mosquitto.conf(5)
               man:mosquitto(8)
   Main PID: 3502 (mosquitto)
        Tasks: 1 (limit: 4915)
   CGroup: /system.slice/mosquitto.service
           └─3502 /usr/sbin/mosquitto -c /etc/mosquitto/mosquitto.conf

Apr 10 06:32:49 beeserver systemd[1]: Starting Mosquitto MQTT v3.1/v3.1.1 Broker
Apr 10 06:32:49 beeserver systemd[1]: Started Mosquitto MQTT v3.1/v3.1.1 Broker.
Apr 23 16:15:44 beeserver mosquitto[3502]: warning: can't get client address: Co
Apr 24 20:32:32 beeserver mosquitto[3502]: warning: can't get client address: Co
Apr 30 00:07:27 beeserver systemd[1]: Reloading Mosquitto MQTT v3.1/v3.1.1 Broke
Apr 30 00:07:27 beeserver systemd[1]: Reloaded Mosquitto MQTT v3.1/v3.1.1 Broker
Apr 30 16:26:44 beeserver mosquitto[3502]: warning: can't get client address: Co
lines 1-17/17 (END)

```

Figure 38: Connection Established to beesmart Server

The screenshot displays three windows related to database configuration verification:

- Command Prompt:** Shows a series of "Publishing fake Humidity Value" messages. Each message includes a "Sensor_ID" (Dummy-1), a "Date" (e.g., "16-Apr-2020 17:54:04:651645"), and a "Data" value (e.g., 65.83).
- beesmart@beeserver:~/Desktop/MQTT_LISTENER:** Shows the corresponding MQTT listener logs. It receives "MQTT Data Received..." for each topic "test" and then "Inserted Humidity Data into Database." The logs also show the raw MQTT data received, including the sensor ID and date.
- DB Browser for SQLite:** Shows a table named "Data" with the following columns: "id", "SensorID", "Date_n_Time", and "Data". The table contains 11 rows of data, corresponding to the 11 published messages in the Command Prompt.

id	SensorID	Date_n_Time	Data
2	Dummy-1	16-Apr-2020 ...	63.97
3	Dummy-1	16-Apr-2020 ...	65.83
4	Dummy-1	16-Apr-2020 ...	79.82
5	Dummy-1	16-Apr-2020 ...	99.36
6	Dummy-1	16-Apr-2020 ...	94.53
7	Dummy-1	16-Apr-2020 ...	85.34
8	Dummy-1	16-Apr-2020 ...	53.45
9	Dummy-1	16-Apr-2020 ...	83.42
10	Dummy-1	16-Apr-2020 ...	59.36
11	Dummy-1	16-Apr-2020 ...	56.06

Figure 39: Database Configuration Verification

7.5 Appendix E

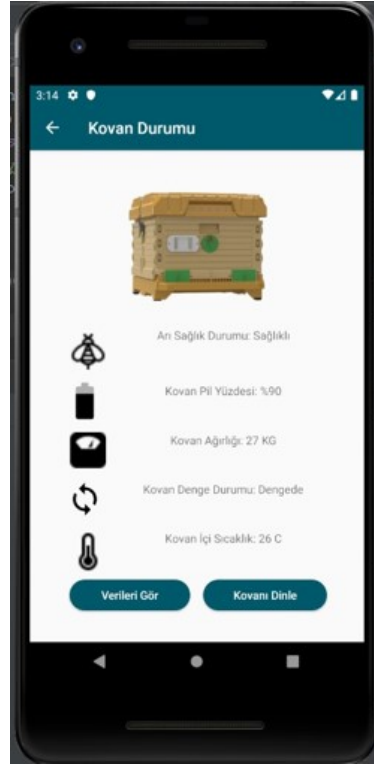


Figure 40: Modified Android Application Screenshot- Hive Status



Figure 41: Modified Android Application Screenshot- Hive Data

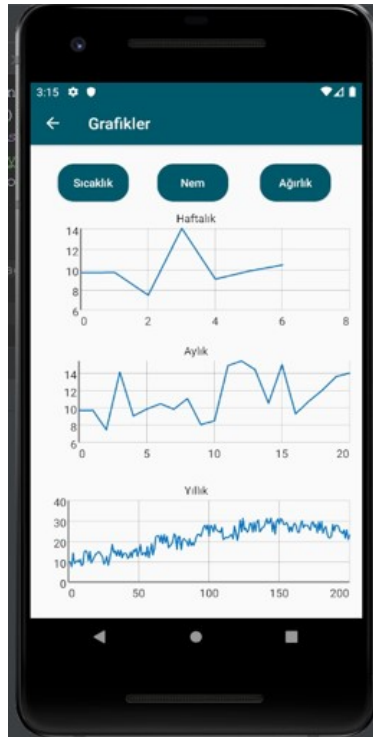


Figure 42: Modified Android Application Screenshot- Hive Graphics

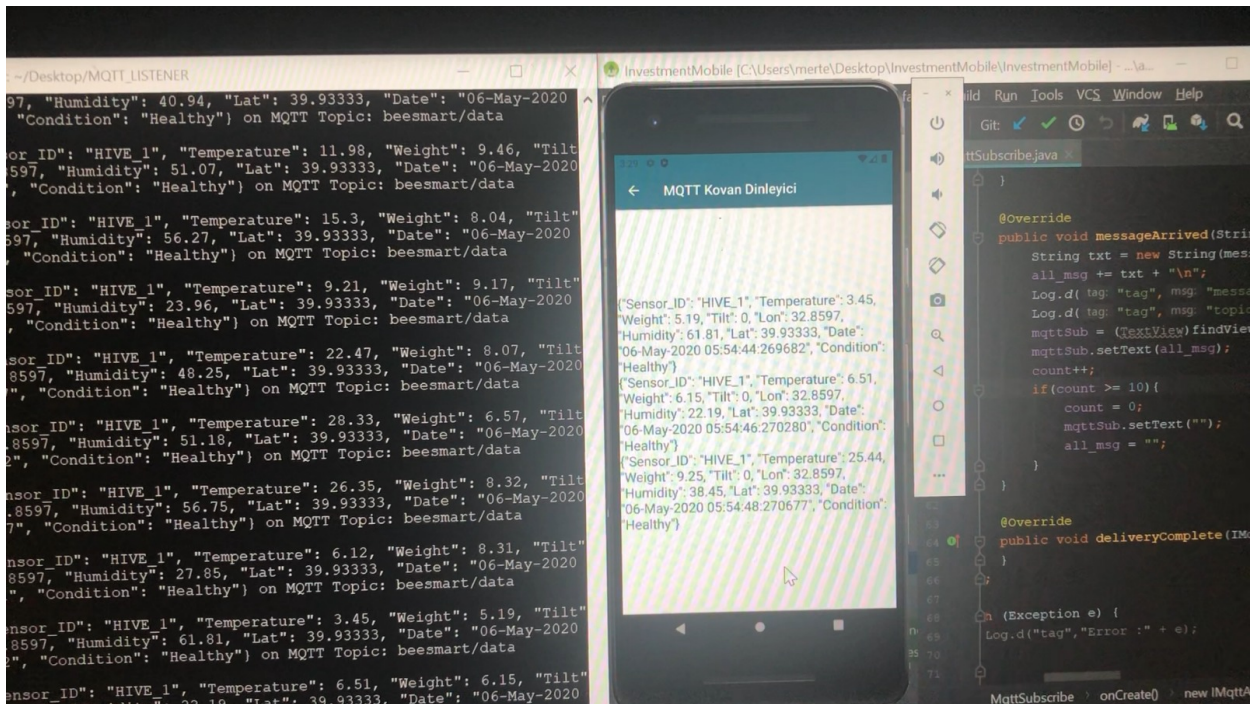


Figure 43: Server Publishing Data and Application Listening via MQTT

7.6 Appendix F

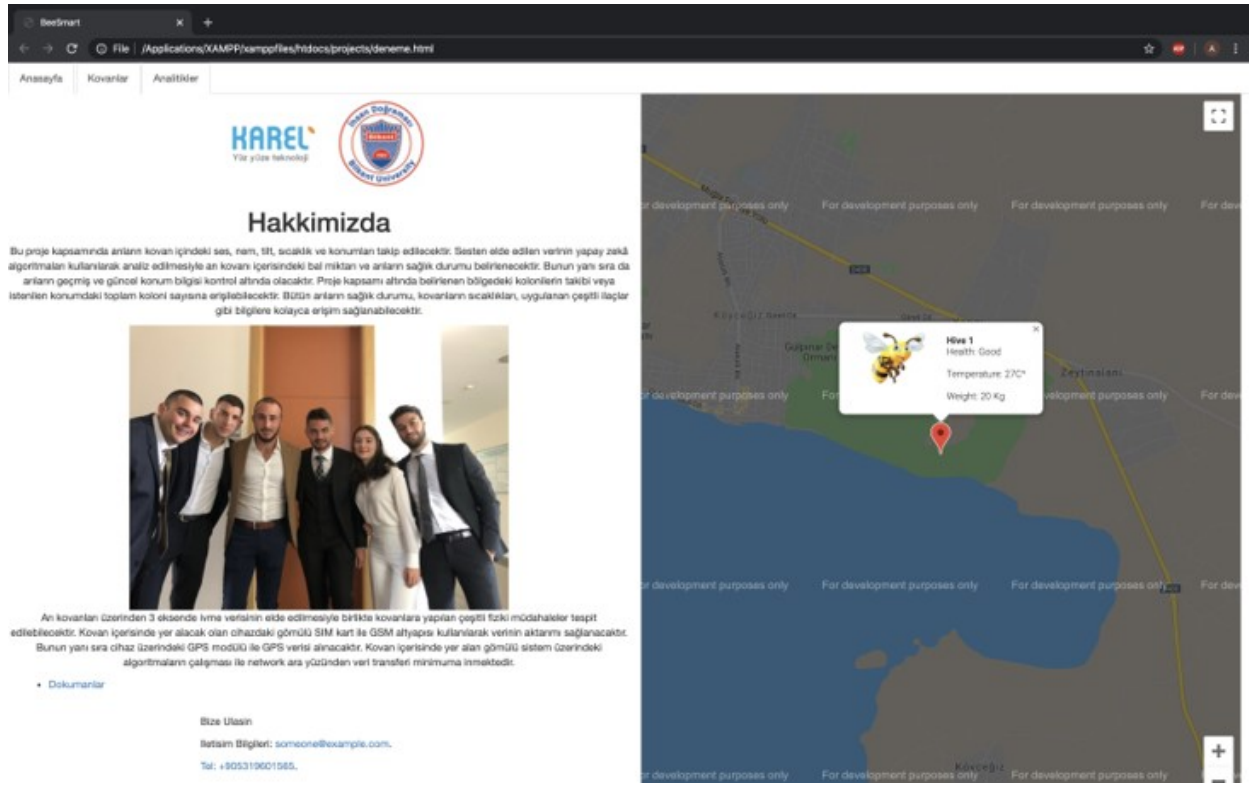


Figure 44: Web Application Screenshot -Homepage

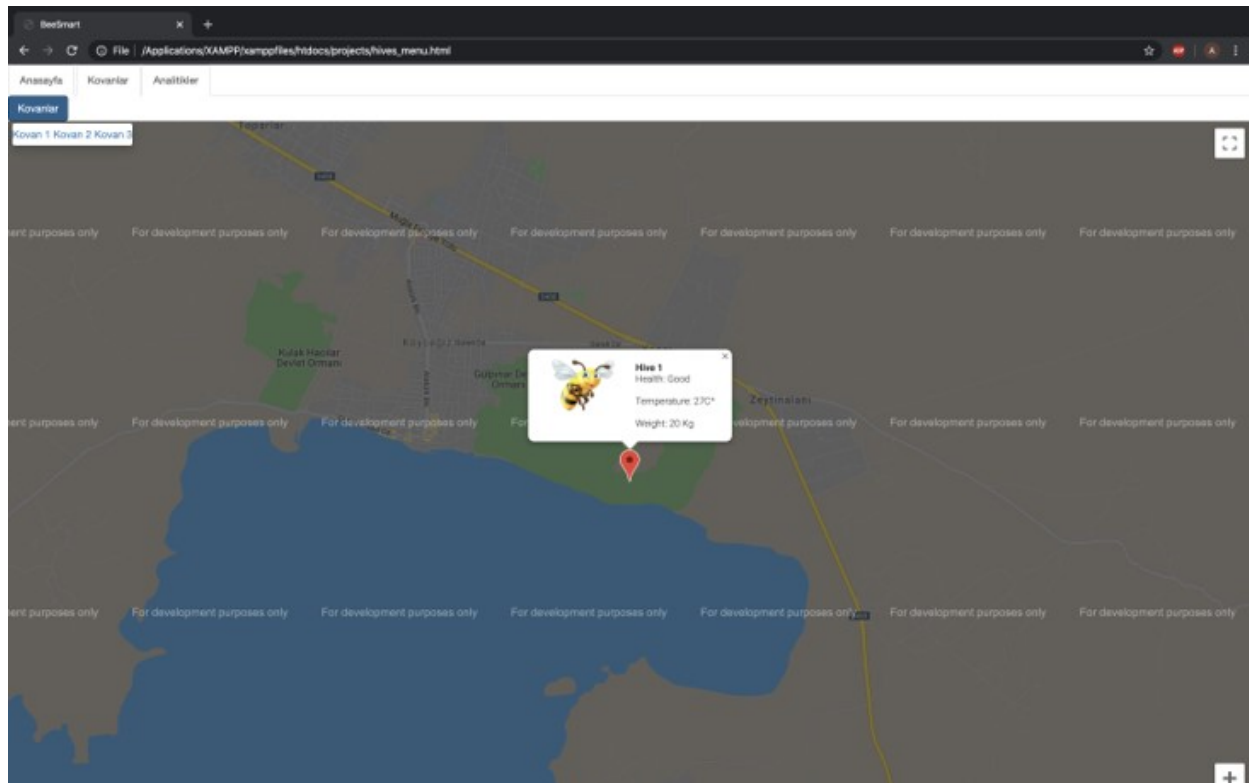


Figure 45: Web Application Screenshot -Hive Location

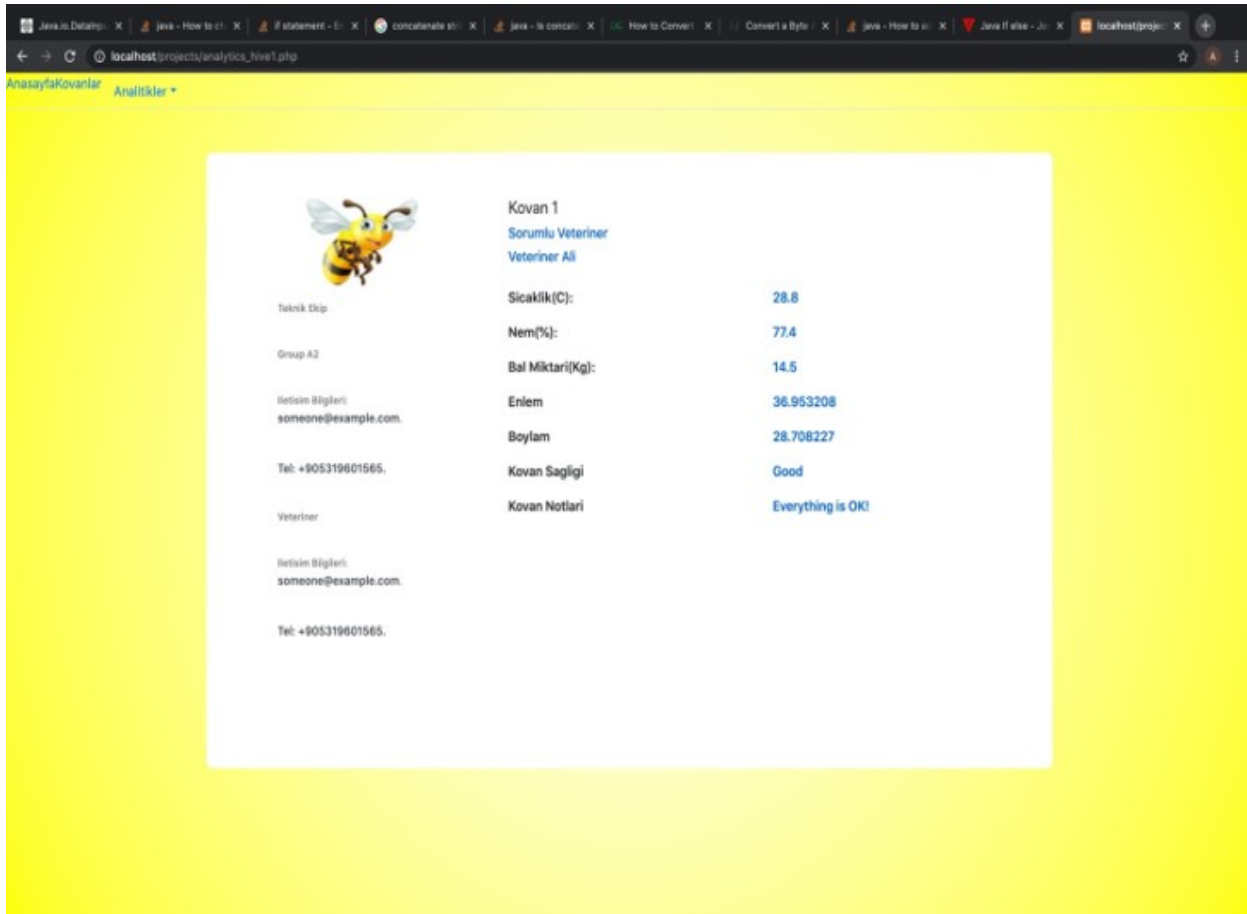


Figure 46: Web Application Screenshot -Hive Status

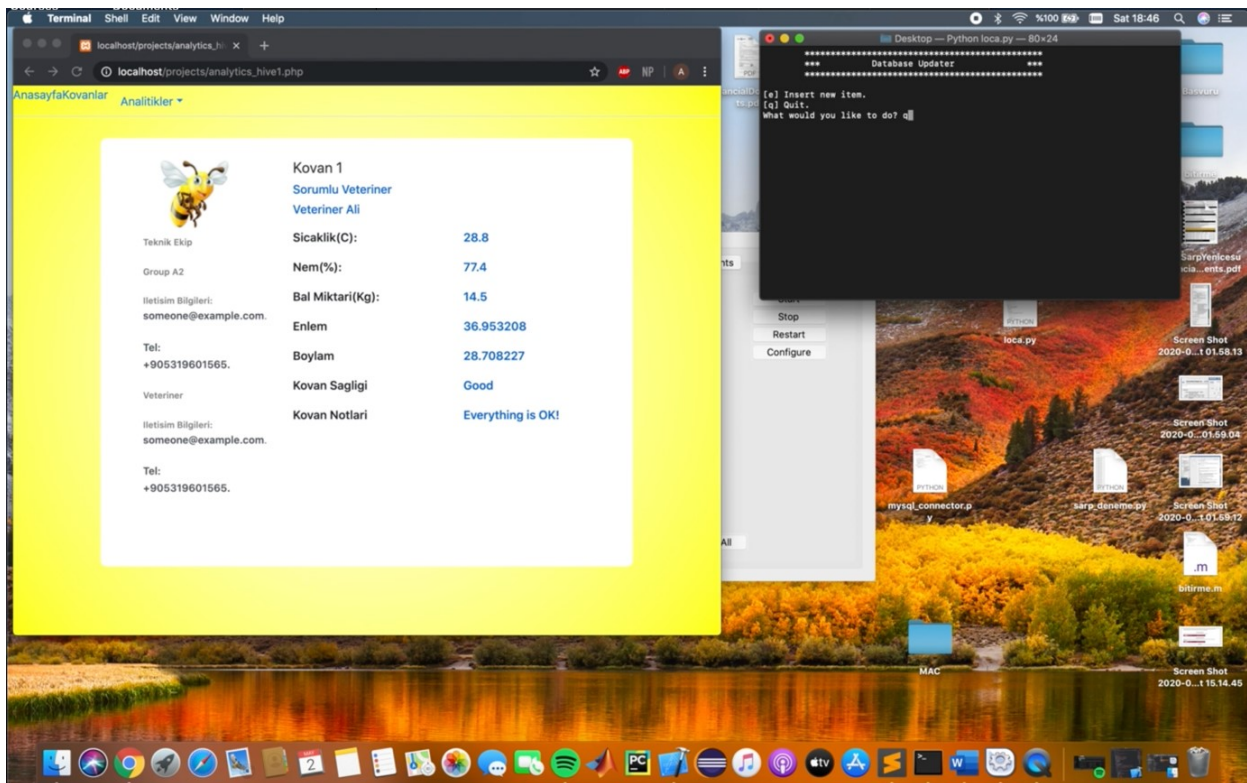


Figure 47: Online Demonstration -Live Changes Through Database

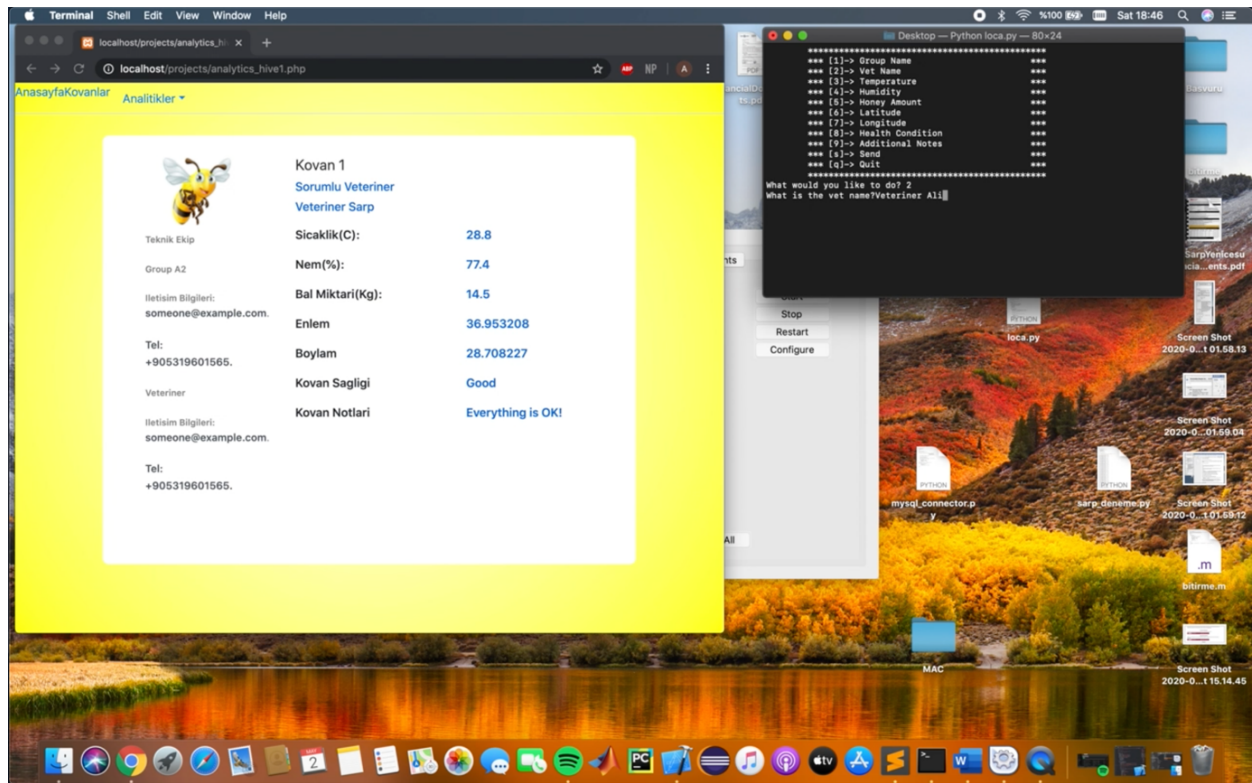


Figure 48: Online Demonstration -Settings Menu

References

- [1] O. Parlakay, H. Yilmaz, B. Yasar, A. Secer, B. Bahadir *Prospects for the Future of the Present Situation and Trend Analysis Method of beekeeping activities in Turkey*
- [2] E. Murphy, M. Magno, L. O’Leary, P. Whelan, E. Popovici *Big brother for bees (3B) - Energy neutral platform for remote monitoring of beehive imagery and sound* 2015.
- [3] V. Rybin, D. Butusov, T. Karimov, D. Belkin, M. Kozak *Embedded Data Acquisition System for Beehive Monitoring* IEEE, 2017.
- [4] Arnia *Better Knowledge for Bee Health, BeeState Monitor*
<https://www.arnia.co.uk>
- [5] M. Aizen, L. Harder *’The Global Stock of Domesticated Honey Bees Is Growing Slower Than Agricultural Demand for Pollination*
Current Biology, vol. 19, no. 11, pp. 915-918, 2009.
- [6] *IEEE Guide for Developing System Requirements Specifications* IEEE Std 1233, 1998 Edition(R2002)
- [7] ISO. *ISO12824:2016*.
<https://www.iso.org/standard/65648.html>
- [8] Su Y, Zhang K, Wang J, Madani K *Environment Sound Classification Using a Two-Stream CNN Based on Decision-Level Fusion. Sensors* (Basel). 2019 Apr 11;19(7):1733. doi: 10.3390/s19071733. PMID: 30978974; PMCID: PMC6479959.
- [9] S. Ö. Arik, H. Jun, and G. Diamos *Fast Spectrogram Inversion Using Multi-Head Convolutional Neural Networks* IEEE Signal Processing Letters, vol. 26, no. 1, pp. 94–98, 2019.