

T.R.
GEBZE TECHNICAL UNIVERSITY
FACULTY OF ENGINEERING
DEPARTMENT OF COMPUTER ENGINEERING

**VOICE RECOGNITION SYSTEM FOR
BEEKEEPING HARDWARE + DATA COLLECTION**

ŞEYDA ÖZER

**SUPERVISOR
PROF. DR. YUSUF SINAN AKGÜL**

**GEBZE
2023**

T.R.
GEBZE TECHNICAL UNIVERSITY
FACULTY OF ENGINEERING
COMPUTER ENGINEERING DEPARTMENT

VOICE RECOGNITION SYSTEM FOR
BEEKEEPING HARDWARE + DATA
COLLECTION

ŞEYDA ÖZER

SUPERVISOR
PROF. DR. YUSUF SINAN AKGÜL

2023
GEBZE

 <p>GEBZE TECHNICAL UNIVERSITY</p>	<p>GRADUATION PROJECT JURY APPROVAL FORM</p>
--	--

This study has been accepted as an Undergraduate Graduation Project in the Department of Computer Engineering on 21/06/2023 by the following jury.

JURY

Member

(Supervisor) : Prof. Dr. Yusuf Sinan Akgül

Member : Dr. Yakup Genç

ABSTRACT

Beekeeping is a sector of critical importance for global food security and requires significant effort. In this study, we designed a system aiming to alleviate the daily workload of beekeepers using a Jetson Nano 2GB Developer Kit. This system continuously records the sounds produced by bees inside the hive through a microphone and stores this data. The recordings are named with hive number and time information. Furthermore, we developed a mobile application to assist beekeepers in identifying hive conditions. The mobile application uses QR codes to identify hives and allows beekeepers to label hive states. These labels are used to make sense of the sound data, providing the capability of remote hive condition monitoring. This system provides a wealth of data for the analysis and interpretation of bee behavior and sounds using deep learning models. This could potentially provide a significant increase in efficiency for the beekeeping sector and be used for monitoring bee health. The motivation for this project is to remotely monitor the condition of hives using sound data and thereby ease the daily tasks of beekeepers.

Keywords: Jetson Nano 2GB Developer Kit, deep learning, mobile application, QR code.

ÖZET

Arıcılık, küresel gıda güvencesi açısından kritik bir öneme sahip olan ve büyük bir çaba gerektiren bir sektördür. Bu çalışmada, bir Jetson Nano 2GB Developer Kit kullanarak, arıların günlük işlerini hafifletmeyi amaçlayan bir sistem tasarladık. Bu sistem, kovan içerisine yerleştirilen bir mikrofon aracılığıyla arıların çıkardığı sesleri sürekli olarak kaydeder ve bu verileri depolar. Kayıtlar, kovan numarası ve zaman bilgisi ile isimlendirilmiştir. Ayrıca, arıların kovan durumunu belirlemelerine yardımcı olmak için bir mobil uygulama geliştirdik. Mobil uygulama, QR kodu aracılığıyla kovanları tanımlar ve arıların kovan durumlarını etiketlemelerini sağlar. Bu etiketler, ses verilerinin anlamlı hale getirilmesinde kullanılır ve bu sayede uzaktan kovan durumu izlemi olanağı sağlar. Bu sistem, arı davranışları ve seslerinin derin öğrenme modelleri kullanılarak analiz edilmesi ve yorumlanması için büyük miktarda veri sağlar. Bu, arıcılık sektörü için potansiyel olarak büyük bir verimlilik artışı sağlayabilir ve arı sağlığını izlemek için kullanılabilir. Bu projenin motivasyonu, kovanların ses verilerini kullanarak arıların durumunu uzaktan izleyebilmek ve bu sayede arıların günlük işlerini kolaylaştırmaktır.

Anahtar Kelimeler: Jetson Nano 2GB Developer Kit, derin öğrenme, mobil uygulama, QR kod.

ACKNOWLEDGEMENT

I would like to my special thanks of gratitude to my supervisor Prof. Dr. Yusuf Sinan Akgül for his guidance and support in completing my project.

Şeyda Özer

LIST OF SYMBOLS AND ABBREVIATIONS

Symbol or
Abbreviation : Explanation

CONTENTS

Abstract	iv
Özet	v
Acknowledgement	vi
List of Symbols and Abbreviations	vii
Contents	ix
List of Figures	x
List of Tables	xi
1 Introduction	1
1.1 Project Description	2
1.2 Project Purpose	2
2 Literature Review	4
2.1 Audio Recordings from Beehives	4
3 Method and System Architecture	6
3.1 System Requirements	6
3.1.1 Hardware Components	6
3.1.2 Software Components	7
3.1.3 Network Components	8
3.2 Architecture and Implementation Details	8
3.2.1 Setting Up the Jetson Nano 2GB Developer Kit	8
3.2.1.1 Unsuccessful Attempts at Setting Up the Jetson Nano 2GB Developer Kit	9
3.2.1.1.1 Attempt with Windows using Etcher In- structions	9
3.2.1.1.2 Attempt with Ubuntu using Etcher Instruc- tions	9
3.2.1.1.3 Attempt with Ubuntu using Command Line Instructions	9

3.2.1.1.4	Attempt with Initial Setup in Headless Mode	9
3.2.1.1.5	Investigation into JetPack SDK	9
3.2.1.2	Successful Setup Using NVIDIA SDK Manager . .	10
3.2.2	Setting up the Environment for Data Collection at the Beehive	11
3.2.2.1	Power Supply to the Device	11
3.2.2.2	Establishing Internet Connectivity	11
3.2.2.3	Device and Microphone Installation	12
3.2.3	Audio Recording	12
3.2.3.1	Python Code for Audio Recording	13
3.2.3.1.1	Requirements for running Python code . .	13
3.2.3.2	Implementation of Bash Script for Auto-Restart . .	14
3.2.3.3	Creating a systemd Service to Execute the Bash Script	14
3.2.4	Remote Access to the Jetson Nano with TeamViewer	15
3.2.5	Mobile Application	15
4	Experiments	17
4.1	Results	17
4.1.1	Recording of Audio Files	17
5	Evaluation of Success Criterias	18
6	Discussion and Conclusion	19
	Bibliography	21

LIST OF FIGURES

1.1	beekeeper[1]	1
1.2	beekeeper[1]	2
3.1	Jetson Nano 2 GB Developer Kit[6]	6
3.2	USB Microphone	6
3.3	Case for Jetson Nano 2 GB Developer Kit	7
3.4	Getting Started with Jetson Nano 2GB Developer Kit[6]	8
3.5	Jetson Nano under the beehive	12
3.6	Microphone inside the beehive	12
3.7	Microphone entry point into the hive	12
3.8	Log file format	13
3.9	sudo apt-get install portaudio19-dev	13
3.10	sudo pip3 install pyaudio	14
3.11	Recorder Bash Script	14
3.12	Recorder Service	14
3.13	Main Page	16
3.14	Scanning QR Page	16
3.15	Select Status Page	16
4.1	Daily folders	17
4.2	bee audio files on Drive	17

LIST OF TABLES

1. INTRODUCTION

Beekeeping, is a crucial industry contributing significantly to global food security through pollination. However, it demands intensive labor and meticulous observation to maintain hive health and productivity. Traditionally, beekeepers must physically inspect hives regularly to monitor the state of the hive, including the presence or absence of the queen bee, activity levels, and potential disease or pest outbreaks. For those managing a large number of hives, this task can be exceedingly time-consuming and labor-intensive.



Figure 1.1: beekeeper[1]

Emerging technologies in machine learning, artificial intelligence, and sensor technology offer new avenues to automate and streamline this monitoring process. Particularly, the application of sound analysis in beekeeping has shown promise in recent research. The sound generated within a bee hive can provide critical insights into the hive's status, potentially indicating the presence of a queen, general activity levels, and even disease outbreaks.

In this project, we developed a remote hive monitoring system using the Jetson Nano 2GB Developer Kit and a sound recording setup. This system continuously records and stores the sounds produced within the hive. Additionally, we designed a mobile application allowing beekeepers to identify each hive using QR codes and label the state of the hive. The combination of these features aims to alleviate the burden of manual hive monitoring, enabling beekeepers to manage their hives more efficiently.



Figure 1.2: beekeeper[1]

By collecting sound data alongside beekeeper annotations, our system provides a substantial dataset for further analysis. The collected data can be used for training deep learning models, aiming to automate the process of hive state recognition based on sound patterns. This project serves as a foundation for future research into applying machine learning for the automation and optimization of beekeeping practices.

1.1. Project Description

The project comprises the development and implementation of a remote hive monitoring system utilizing the Jetson Nano 2GB Developer Kit. The system features a sound recording setup, which includes a microphone placed inside the beehive to minimize disturbance to the bees. The Jetson Nano is encased in a 3D-printed case to protect it from natural elements and insects. It is programmed to record hive sounds continuously, storing them with filenames indicating hive number and timestamp.

In case of a power outage or system reboot, a bash script is used to automatically restart the recording process. This script is executed through a systemd service, ensuring that the recording starts every time the device boots up.

In addition to the sound recording setup, a mobile application was developed for beekeepers. The application uses QR codes to identify individual hives, and it allows the beekeepers to label the state of the hive. Beekeepers can label the hives as "No Queen Bee", "Queen Bee", "Missing Queen Bee", or "Active Day". This labeled data, combined with the sound recordings, is used for meaningful interpretation and analysis.

1.2. Project Purpose

The primary purpose of this project is to alleviate the challenges faced by beekeepers in monitoring their hives. Manual inspection of hives is a laborious and time-consuming task, especially for beekeepers managing a large number of hives. By

utilizing sound data from the hives, this project aims to provide beekeepers with a remote monitoring capability.

This project also serves to build a dataset combining hive sounds and labeled data from beekeepers. This dataset can be invaluable for training deep learning models to recognize patterns and indicators in the sounds produced within the hive. These patterns could potentially be used to automatically identify the presence of a queen, general activity levels, and early signs of disease or pests.

Ultimately, the goal is to use technology to make beekeeping more efficient, accurate, and sustainable. By easing the workload of beekeepers and providing them with more information about the state of their hives, this project has the potential to positively impact the beekeeping industry and contribute to food security through healthier and more productive hives.

2. LITERATURE REVIEW

Similar projects on how they recorded the sounds of bees in the hives were researched. How they collected data from the hives was examined. [2]–[5]

2.1. Audio Recordings from Beehives

In the article [3], the audio recordings from bee hives were collected using a multi-sensor Electronic Bee Monitoring (EBM) system called "BeePi". This system is comprised of components such as a Raspberry Pi computer, a miniature camera, a microphone splitter, four microphones, a solar panel, a temperature sensor, a battery, and a hardware clock. The BeePi unit records 30-second audio clips every 15 minutes using the four microphones and saves these recordings to the Raspberry Pi. The data collected by this system was saved locally, either on the Raspberry Pi's SD card or on a USB storage device connected to the Raspberry Pi.

In summary, the BeePi, a specially designed system, was used to collect audio recordings from the bee hive, and this system used four microphones to periodically record audio, and this data was stored on the Raspberry Pi.

In the article [4], techniques used for recording sound from bee hives are discussed. Microphones and accelerometers are used for capturing sound, and are positioned inside the hive. The article includes examples showing how microphones and accelerometers are placed in different locations within the hive (Figure 2). For instance, in one approach, the microphone is placed above the cage where the queen bee is located, while in another approach, microphones are positioned outside the hive.

In contrast to various studies that have predominantly utilized Raspberry Pi for their projects, this project has elected to employ the Jetson Nano 2GB. The rationale behind this choice lies in Jetson Nano's substantial advantages. Firstly, Jetson Nano is endowed with a more powerful processor, which is pivotal for running complex calculations and deep learning models. Specifically designed for AI applications, it delivers superior performance compared to Raspberry Pi. Secondly, Jetson Nano features an integrated GPU, which is crucial for applications such as deep learning that require parallel computation. The GPU is capable of executing these applications at a much faster rate than a CPU. Moreover, Jetson Nano 2GB boasts a larger RAM, which is essential for rapidly processing a large volume of data, particularly in running deep learning models. Additionally, NVIDIA provides libraries optimized for AI and deep

learning applications for Jetson Nano, such as TensorRT. Jetson Nano is also more adept at processing high-resolution video streams, which is vital for image and audio processing applications. Given that this project entails recording audio from beehives throughout the day and processing this data, the selection of Jetson Nano 2GB, with its high processing capacity and ability to locally run deep learning models, is a prudent decision.

3. METHOD AND SYSTEM ARCHITECTURE

3.1. System Requirements

The system requirements of this project have been meticulously detailed, encompassing hardware, software, and network components.

3.1.1. Hardware Components

- Jetson Nano 2 GB Developer Kit: A compact board with GPU support, essential for processing and storing the hive audio data.

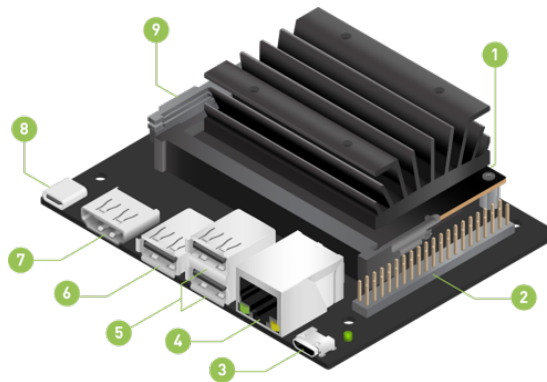


Figure 3.1: Jetson Nano 2 GB Developer Kit[6]

- USB Microphone: A compact, microphone that is capable of capturing sound within the hive without disturbing the bees.



Figure 3.2: USB Microphone

- Micro SD Card: that is used as a boot device and for main storage. The minimum requirement is a 32GB UHS-1 card. 64GB or larger microSD cards are recommended. [6]
- QR Codes: Individual QR codes for each hive to allow identification through the mobile application.
- 3D-Printed Case: Protective casing for the Jetson Nano 2GB Developer Kit to shield it from natural elements and insects [7].



Figure 3.3: Case for Jetson Nano 2 GB Developer Kit

- Power Source: A reliable power source of powering the Jetson Nano for extended periods. Also USB-C power supply (5V=3A) is needed. [6]
- Smartphone: A smartphone with camera capability for the beekeepers to use the mobile application.

3.1.2. Software Components

- Python3 [8] and PyAudio library [9]
- Android Studio : used for Mobile Application development [10]
- Nvidia SDK Manager : used for setup Jetson Nano [11]
- TeamViewer (optional) : used for remote connection to Jetson Nano [12]

3.1.3. Network Components

Internet Connection: An Internet connection will be required as data is regularly transferred to Drive and remote connection.

3.2. Architecture and Implementation Details

To accomplish this project, I used the Jetson Nano 2GB Developer Kit. Jetson Nano installed. I printed a case for my device with a 3D printer. I chose a microphone that wouldn't attract the attention of bees while listening. The Jetson Nano is programmed with a Python script to continuously record audio from the microphone placed within the hive.

3.2.1. Setting Up the Jetson Nano 2GB Developer Kit

During the initial phase of my project, I faced numerous challenges in setting up the Jetson Nano 2GB Developer Kit [6], [13]. Despite rigorously following the setup instructions provided by NVIDIA on their official website, I was unable to boot the device. Below is a detailed account of the steps I took and the challenges I faced.

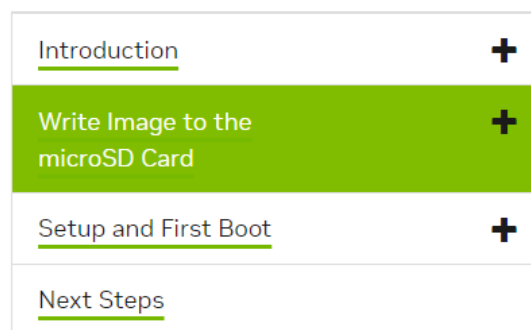


Figure 3.4: Getting Started with Jetson Nano 2GB Developer Kit[6]

3.2.1.1. Unsuccessful Attempts at Setting Up the Jetson Nano 2GB Developer Kit

3.2.1.1.1 Attempt with Windows using Etcher Instructions

- I followed the "Initial Setup with Display Attached" mode instructions using a Windows operating system.
- I used the Etcher software to write the image to the SD card as per NVIDIA's guide.
- Upon completing the image writing process, I inserted the SD card into the Jetson Nano and attempted to boot the device.
- Unfortunately, this attempt was unsuccessful, as the device did not boot up.

3.2.1.1.2 Attempt with Ubuntu using Etcher Instructions

- To eliminate the possibility of OS-specific issues, I switched to Ubuntu and repeated the steps using Etcher.
- Just like before, I followed the "Initial Setup with Display Attached" instructions.
- Despite trying with a different operating system, the device still did not boot up.

3.2.1.1.3 Attempt with Ubuntu using Command Line Instructions

- Thinking that the issue might be with the Etcher software, I decided to try writing the image using command-line instructions on Ubuntu.
- This attempt also did not yield a successful boot-up of the Jetson Nano.

3.2.1.1.4 Attempt with Initial Setup in Headless Mode

- As a troubleshooting step, I tried setting up the Jetson Nano in "Headless Mode".
- However, this approach also did not result in a successful boot-up.

3.2.1.1.5 Investigation into JetPack SDK

- At this stage, I started to explore the JetPack SDK in an attempt to understand if it could assist in resolving the issues.
- However, understanding and setting up the JetPack SDK proved to be complex, and I could not ascertain if it would address the boot-up issue.

In conclusion, despite multiple attempts and variations in the setup process, I was unable to successfully boot the Jetson Nano 2GB Developer Kit using NVIDIA's provided instructions. This presented a significant hurdle in the progression of my project, and alternative solutions or support would be needed to overcome this challenge. After numerous failed attempts to boot the Jetson Nano 2GB Developer Kit, I discovered that flashing the image to the SD card alone was not sufficient for the setup. The setup needed to be done using the NVIDIA SDK Manager [11], [14], [15] while the SD card was inserted in the device. This information was not included in the NVIDIA's official setup instructions, which led to the initial confusion and challenges.

3.2.1.2. Successful Setup Using NVIDIA SDK Manager

- I downloaded and installed the NVIDIA SDK Manager on my computer.
- I inserted the SD card into the Jetson Nano 2GB Developer Kit.
- However, when I launched the SDK Manager, it did not recognize the Jetson Nano 2GB Developer Kit.
- I realized that I needed to put the device into recovery mode for the SDK Manager to recognize it.
- To do this, I short-circuited the GND and PWR BTN pins on the device to enter recovery mode.
- While in recovery mode, the SDK Manager successfully recognized the Jetson Nano 2GB Developer Kit.
- With the device recognized by the SDK Manager, I proceeded with the installation by following the prompts in the SDK Manager.
- After the installation was completed, I was able to successfully boot the Jetson Nano 2GB Developer Kit.

In conclusion, the key to successfully setting up the Jetson Nano 2GB Developer Kit was utilizing the NVIDIA SDK Manager and placing the device in recovery mode, which was a step not covered in the initial NVIDIA instructions. Through perseverance and additional resources, I was finally able to overcome the setup challenges and properly boot the Jetson Nano, enabling me to proceed with my project.

3.2.2. Setting up the Environment for Data Collection at the Beehive

In order to collect data with the device, I began the quest for a beehive. Fortunately, I was able to get in touch with a beekeeper on campus who had a beehive. After identifying the location of the beehive on campus, the next challenge was ensuring that the device had access to a power source since it needed to operate continuously throughout the day.

3.2.2.1. Power Supply to the Device

- The device needed to operate continuously throughout the day, necessitating a constant power supply.
- After surveying the site, it was decided to draw a power cable from the nearest building to the beehive.
- A 50-meter long power cable was strung from the building to the beehive, running through the trees to avoid any damage and to minimize any disturbance to the daily activities of individuals on campus.
- This ensured that the Jetson Nano received the required power supply for continuous operation.

3.2.2.2. Establishing Internet Connectivity

- The location of the beehive did not have Wi-Fi coverage, which posed an issue as internet connectivity was essential for remotely accessing the device.
- I then decided to also run an Ethernet cable from the building.
- Initially, an existing hole in the building's wall was used for the power cable, but this hole was too small for the Ethernet cable, requiring me to drill a new hole.
- After obtaining the necessary permissions, I procured approximately 60 meters of Ethernet cable, which was a challenging task. I managed to find two different cables and connected them together using an adapter.
- Like the power cable, the Ethernet cable was also routed over trees through the newly-drilled hole to the beehive.

3.2.2.3. Device and Microphone Installation

- I concluded that it would not be suitable to place the device directly inside the beehive. As such, I opted for a USB microphone with a cable 3.2.
- The microphone was installed inside the beehive, while the Jetson Nano was placed beneath the beehive.



Figure 3.5: Jetson Nano under the beehive



Figure 3.6: Microphone inside the beehive



Figure 3.7: Microphone entry point into the hive

- To protect the device and its pins from damage, a casing was fabricated using a 3D printer 3.3.

This arrangement allowed the Jetson Nano to be securely positioned for collecting audio data from the beehive environment, with a steady power supply and internet connectivity.

3.2.3. Audio Recording

For the purpose of audio data collection, I developed a Python code that records audio continuously for 24 hours. The python code by recording audio, each 1 minute in length, and then goes into idle mode for 10 minutes before resuming recording. This cycle continues throughout the day.

3.2.3.1. Python Code for Audio Recording

- The script was written in Python and was configured to record 1-minute-long audio.
- After recording each audio, the code pauses for 10 minutes before recording the next audio. This is to ensure that not only continuous data is collected but also to manage the amount of data being stored.
- Additionally, I incorporated a logging feature in the code. A log file was created to keep track of the recorded audio files. All outputs were also logged into this file for monitoring purposes.

```
log_filename = "/home/gtunano3/Desktop/beekeeping/recording_log.log"
log_format = "%(asctime)s - %(levelname)s - %(message)s"
logging.basicConfig(filename=log_filename, level=logging.INFO, format=log_format)
```

Figure 3.8: Log file format

3.2.3.1.1 Requirements for running Python code

- Python 3: As the script is written in Python, it is essential to have Python installed on the system.
- PortAudio: The script relies on the pyaudio library for audio recording, which in turn depends on the PortAudio development files. PortAudio is a cross-platform audio I/O library and must be installed prior to installing the pyaudio library. On Debian-based Linux systems, for example, this can be achieved by running the following command:

```
sudo apt-get install portaudio19-dev
```

Figure 3.9: `sudo apt-get install portaudio19-dev`

- Pyaudio: Once PortAudio is installed, the Pyaudio library must be installed. This can be done through Python's package manager, pip, by executing the following command:


```
sudo pip3 install pyaudio
```

Figure 3.10: `sudo pip3 install pyaudio`

3.2.3.2. Implementation of Bash Script for Auto-Restart

A bash script is implemented to restart the Python recording script in case of system reboots or power interruptions. The bash script is executed automatically on system startup through a systemd service.

```
#!/bin/bash

sleep 60 &&

echo "Start recording.." &&

python3 /home/gtunano3/Desktop/beekeeping/recorder.py
```

Figure 3.11: Recorder Bash Script

3.2.3.3. Creating a systemd Service to Execute the Bash Script

To ensure that the bash script is executed automatically upon system startup, and to maintain high reliability in case of any system interruptions, I created a systemd service. Systemd is a system and service manager for Linux operating systems and is designed for better control and management of background processes.

After saving this file in the systemd directory, I enabled and started the service. This ensures that the script will run automatically every time the system is booted up. [16]

```
[Unit]
Description=Beekeeping recorder script
After=sound.target

[Service]
ExecStart=/bin/bash -c '/home/gtunano3/Desktop/beekeeping/record_script.sh'
Restart=always

[Install]
WantedBy=multi-user.target
```

Figure 3.12: Recorder Service

This setup ensures the consistent and automated recording of audio data, and also guarantees that unexpected system shutdowns do not hinder the recording process, as the script will automatically restart upon system boot.

3.2.4. Remote Access to the Jetson Nano with TeamViewer

To enable remote access to the Jetson Nano, TeamViewer was utilized. The "teamviewer package for ARM64" was downloaded and installed on the Jetson Nano for this purpose [17].

3.2.5. Mobile Application

The designs for the mobile application screens were created [18]. Once the designs were finalized, the mobile application code was written using Kotlin in Android Studio.

- The mobile application, accessible on beekeepers' smartphones, is designed for hive identification and status labeling.
- It uses the smartphone's camera to scan QR codes placed on the hives.
- Once a hive is identified, beekeepers can input the state of the hive using the options "No Queen Bee", "Queen Bee", "Missing Queen Bee", or "Active Day".
- The application saves this information with the hive number, timestamp, and label (hivenumber-timestamp-label.wav).



Figure 3.13: Main Page

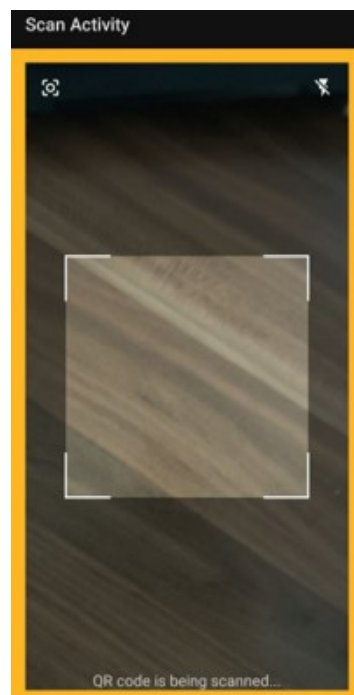


Figure 3.14: Scanning QR Page

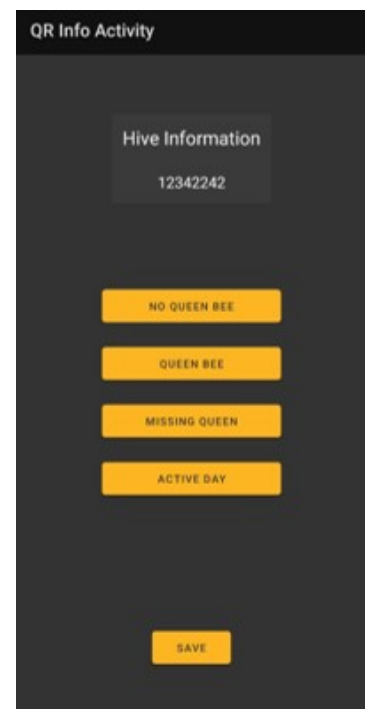


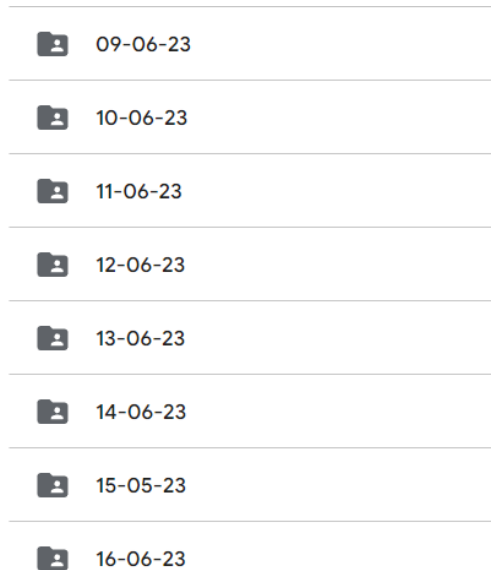
Figure 3.15: Select Status Page

4. EXPERIMENTS

4.1. Results

4.1.1. Recording of Audio Files

The recordings are stored locally on the Jetson Nano's storage with filenames containing the hive number and timestamp (hivenumber-timestamp.wav). The data collected by the Jetson Nano is regularly transferred to Drive. The data on Drive is organized into daily folders.











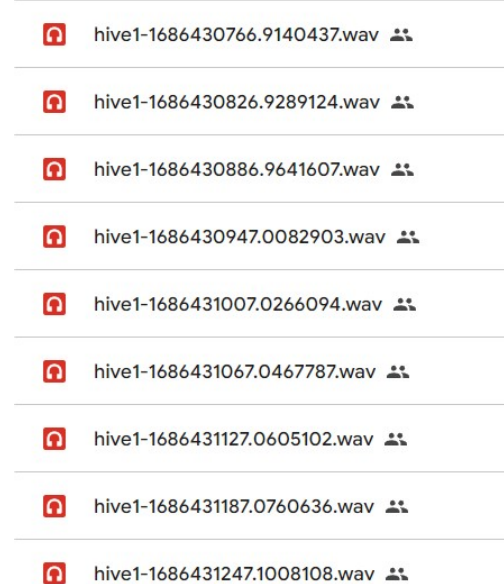
	09-06-23
	10-06-23
	11-06-23
	12-06-23
	13-06-23
	14-06-23
	15-05-23
	16-06-23

Figure 4.1: Daily folders





















	hive1-1686430766.9140437.wav	
	hive1-1686430826.9289124.wav	
	hive1-1686430886.9641607.wav	
	hive1-1686430947.0082903.wav	
	hive1-1686431007.0266094.wav	
	hive1-1686431067.0467787.wav	
	hive1-1686431127.0605102.wav	
	hive1-1686431187.0760636.wav	
	hive1-1686431247.1008108.wav	

Figure 4.2: bee audio files on Drive

Initially, audio was recorded every minute throughout the day. The size of the 1440 audio files is 7 GB. After 3 days, there was no more space left on the device. so I reduced the number of audio files. Currently, 1 minute of audio is being recorded, followed by a 10 minute sleep. The size of 131 audio files is 700 MB.

5. EVALUATION OF SUCCESS CRITERIAS

- %75 memory usage in Jetson Nano per day: Less than %75 of the memory is used. However, it may vary depending on the size of the SD card used in this device. 64GB SD card was used in this project.
- %20 GPU usage in Jetson Nano (128-core NVIDIA Maxwell GPU in Jetson Nano): It shows a variation between %10 and %25. Jetson Nano's overall operating performance is %20 CPU, %10 GPU.
- The mobile application use less than %10 of the CPU: CPU usage is %14 on *Pixel_3a_API*.

6. DISCUSSION AND CONCLUSION

This project presents an innovative approach to hive monitoring by leveraging the capabilities of the Jetson Nano 2 GB Developer Kit, sound recording technology, and a custom mobile application. The system offers a promising alternative to traditional hive monitoring techniques by collecting continuous sound data from the hives. Furthermore, the integration of a mobile application with QR code scanning functionality adds an additional layer of data collection and user interaction, facilitating the labeling process for beekeepers.

The potential of this system in real-world applications is vast. However, further research and development are necessary to refine the sound data processing and analysis components. The training of deep learning models with the collected data is crucial in moving towards automated hive state recognition based on sound patterns. This project serves as a foundation upon which future developments can be built, aiming to make beekeeping more efficient and sustainable.

As next steps, it would be worthwhile to conduct field trials with the participation of experienced beekeepers to validate the effectiveness and practicality of this system.

BIBLIOGRAPHY

- [1] *Microsoft bing image creator*. [Online]. Available: <https://www.bing.com/create>.
- [2] A. Qandour, I. Ahmad, D. Habibi, and M. Leppard, "Remote beehive monitoring using acoustic signals," *Acoustics Australia / Australian Acoustical Society*, vol. 42, pp. 204–209, Dec. 2014.
- [3] V. Kulyukin, S. Mukherjee, and P. Amlathe, "Toward audio beehive monitoring: Deep learning vs. standard machine learning in classifying beehive audio samples," *Applied Sciences*, vol. 8, no. 9, 2018, ISSN: 2076-3417. DOI: 10.3390/app8091573. [Online]. Available: <https://www.mdpi.com/2076-3417/8/9/1573>.
- [4] A. Terenzi, S. Cecchi, and S. Spinsante, "On the importance of the sound emitted by honey bee hives," *Veterinary Sciences*, vol. 7, no. 4, 2020, ISSN: 2306-7381. DOI: 10.3390/vetsci7040168. [Online]. Available: <https://www.mdpi.com/2306-7381/7/4/168>.
- [5] C. G. on Kaggle, *To bee or not to bee*. [Online]. Available: <https://www.kaggle.com/datasets/chrisfilo/to-bee-or-no-to-bee/code>.
- [6] Nvidia, *Get started jetson nano 2gb developer kit*. [Online]. Available: <https://developer.nvidia.com/embedded/learn/get-started-jetson-nano-2gb-devkit>.
- [7] Thingiverse, *Jetson nano 2gb full case*. [Online]. Available: <https://www.thingiverse.com/thing:4649425/files>.
- [8] *Python3*. [Online]. Available: <https://www.python.org/downloads/>.
- [9] *Pyaudio*. [Online]. Available: <https://pypi.org/project/PyAudio/>.
- [10] *Android studio*. [Online]. Available: <https://developer.android.com/studio>.
- [11] Nvidia, *Sdk manager*. [Online]. Available: <https://developer.nvidia.com/sdk-manager>.
- [12] *Teamviewer*. [Online]. Available: <https://www.teamviewer.com/tr/>.
- [13] Nvidia, *Jetpack sdk*. [Online]. Available: <https://developer.nvidia.com/embedded/jetpack>.
- [14] Nvidia, *Sdk manager documentation*. [Online]. Available: <https://docs.nvidia.com/sdk-manager/install-with-sdks-jetson/index.html>.

- [15] Waveshare, *Jetson nano dev kit with sdk manager tool*. [Online]. Available: https://www.waveshare.com/wiki/JETSON-NANO-DEV-KIT#Method_One:_Adopt_SDK_Manager_Tool.
- [16] Tutorialspoint, *Run a script on startup in linux*. [Online]. Available: <https://www.tutorialspoint.com/run-a-script-on-startup-in-linux#:~:text=Make>.
- [17] N. Forum, *Teamviewer on jetson nano*. [Online]. Available: <https://forums.developer.nvidia.com/t/teamviewer-on-jetson-nano/230624>.
- [18] Figma, *Qr code scanner app design*. [Online]. Available: <https://www.figma.com/community/file/1214837612730924876/QR-Code-Scanner-App>.