A CONTROL AND ANALYSIS SYSTEM FOR HONEY BEE HIVES

A Thesis
by
EDWARD SCOTT SHUFFLER JR

Submitted to the Graduate School
Appalachian State University
in partial fulfillment of the requirements for the degree of
MASTER OF SCIENCE

August 2017
Department of Computer Science

A CONTROL AND ANALYSIS SYSTEM FOR HONEY BEE HIVES

A Thesis
by
EDWARD SCOTT SHUFFLER JR
August 2017

APPROVED BY:

Rahman Tashakkori, Ph.D.
Chairperson, Thesis Committee

James B. Fenwick Jr., Ph.D.
Member, Thesis Committee

Raghuveer Mohan, Ph.D.
Member, Thesis Committee

Rahman Tashakkori, Ph.D.
Chairperson, Department of Computer Science

Max C. Poole, Ph.D.
Dean, Cratis D. Williams School of Graduate Studies

Abstract

A CONTROL AND ANALYSIS SYSTEM FOR HONEY BEE HIVES

Edward Scott Shuffler Jr
B.S., Appalachian State University
M.S., Appalachian State University

Chairperson: Rahman Tashakkori, Ph.D.

Honey bees are an important part of our everyday lives. Many foods we eat on a daily basis are dependent on some form of pollination. These foods include nuts, berries, fruits, and vegetables. In recent years, the population of honey bees has been declining due to a phenomenon called Colony Collapse Disorder. Research on this disorder is lacking and there is still much to learn. This thesis provides the details on research and development of a system that manages the data acquisition of one or more honey bee hives. This system could be used by researchers and beekeepers to monitor their beehives. The system allows users to setup parameters on various data acquisition components. The basic hive data will include audio, temperature, and humidity from inside the hive, as well as video recording the beehive entrance. The system (BeeCon) also provides basic analysis options that the hive owners can use to monitor and determine the health of their bees.

Acknowledgements

My thanks go to Dr. Tashakkori for being my mentor, without his guidance and support this thesis would not have been possible. My special thanks go to all of my committee members for reading this thesis. I would also like to thank Devin Sink, Chris Smith, and Zach Andrews; without whom this thesis would not have been as pleasant. My thanks go to my family: my mother, father, grandparents, and siblings. Without their support through my entire education, I don't feel I would have ever been as successful as I have been. My thanks go to the NSF S-STEM program for the scholarship support throughout my time at Appalachian State University. I would also like to thank the Department of Computer Science and the The Lowe's Distinguished Professor Research Fund for providing summer support to conduct my research.

Contents

Abstract **iv**

Acknowledgements **v**

1 Introduction **1**
1.1 Beekeeping 1
1.2 Beehive Maintenance 2
1.3 Graphical User Interfaces 3
1.4 Objectives 4
1.5 Thesis Overview 5

2 Background **7**
2.1 Data Acquisition System 7
2.2 Software 8

3 Literature Review **11**

4 Methodology **22**
4.1 Interfacing with PyBeemon 26
4.2 Network Scanning 27
4.3 Logging into BeeCon 30
4.4 Manual Control 33
4.5 Temperature and Humidity analysis 39
4.6 Changing Preferences 41
4.7 Downloading Audio and Video 45

5 Results **48**
5.1 Deployment 48
5.2 BeeCon Usage 49

6 Conclusion and Future Work **60**
6.1 Conclusions 60
6.2 Future Work 60

Bibliography **62**

Vita **65**

List of Figures

1.1 A sample beehive with two empty frames 2

2.1 Basic flow of data acquisition through PyBeemon 9

4.1 FXML used to create the About panel 23
4.2 About panel created by FXML 24
4.3 Flowchart diagram for BeeCon 25
4.4 Connecting BeeCon and PyBeemon and sending a command 26
4.5 An example of a successful scan listed within BeeCon 29
4.6 An unsuccessful scan resulting in an error message 30
4.7 The Login panel for connecting to connect to PyBeemon 31
4.8 Saved Hosts panel for previously used hosts 33
4.9 Running stop_video 34
4.10 The code for the connect function 35
4.11 The code for the sendCommand function 37
4.12 Locked buttons 38
4.13 Warning when unlocking buttons 39
4.14 Temperature and humidity panel 40
4.15 Code for the download function 42
4.16 Configuration file on the the Raspberry Pi 44
4.17 Preference panel showing the video preferences 45

5.1 An example of a Raspberry Pi 3 data acquisition system 48
5.2 Login screen on Linux, Mac, and Windows 50
5.3 Network Scan screen on Linux, Mac, and Windows found two IPs 51
5.4 Main panel displayed on Linux, Mac, and Windows 52
5.5 Main panel unlocked and audio started displayed on Linux, Mac, and Windows 53
5.6 Downloaded video playing in BeeCon on Linux, Mac, and Windows . . . 54
5.7 Preference panel on Linux 55
5.8 Preference panel on Mac 55

5.9 Preference panel on Windows 56
5.10 Temperature and humidity analysis panel on Linux 57
5.11 Temperature and humidity analysis panel on Mac 57
5.12 Temperature and humidity analysis panel on Windows 58

Chapter 1 - Introduction

1.1 Beekeeping

Honey bees have a significant impact on the world around us, as they help the pollination of crops, vegetables, and fruits. Almonds, in particular, are completely dependent on honey bee pollination, and because of this, a large portion of the United States' honey bees are sent to California [1]. However, almonds are not the only crop that requires pollination from honey bees. Grapes and strawberries also require pollination from honey bees. Pollination for crops is one of the reasons beekeepers are interested in keeping bees.

Another reason that some beekeepers are keeping beehives is to research the sudden collapse of bee colonies over the past decade. Population loss may stem from colony collapse disorder (CCD), has been recorded for several years and the first dramatic loss was noted in the Winter between 2006 and 2007. Some researchers believe CCD is caused by worker bees abandoning their queen, food, and young bees suddenly [2]. Colony collapse disorder is still largely unknown, and research is still ongoing to determine the cause.

Beekeepers provide bee colonies with a shelter to live. These shelters can come in all shapes and sizes, and can be constructed from many types of material. Frames are provided to the bees, which are support structures for the bees to build upon. In Figure 1.1, two frames inside an empty beehive are shown. These frames do not have the foundation sheet that is often installed on the frames to encourage the build up of the hives.

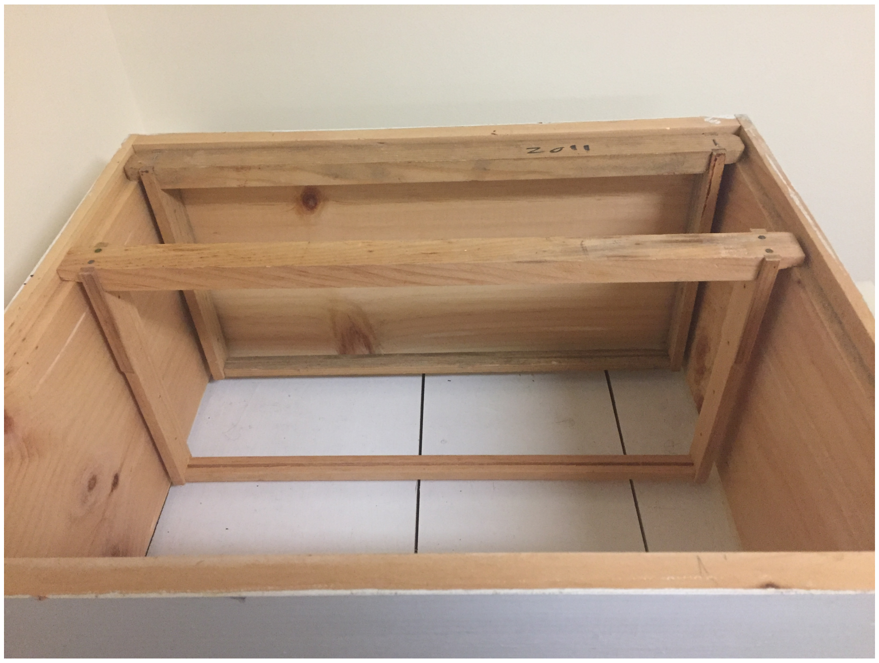


Figure 1.1: A sample beehive with two empty frames

This is an example of a basic beehive built out of wood. Over the course of the season, from Spring to Fall, bees will fill these frames with honeycombs, which are wax structures in which they will store honey, pollen, and their larvae. Artificial foundations could also be provided to new bee colonies to improve their build up.

1.2 Beehive Maintenance

While honey bees do not require constant monitoring in order to thrive, beekeepers will often monitor the hives to ensure the hives are healthy. Two key areas beekeepers monitor are the entrance of the hive and the frames within it. Monitoring the entrance of the hive can show beekeepers if there's a healthy amount of pollen entering the hive, or if the hive is in trouble from being robbed. Inspecting the frames will notify the beekeeper of the health of the queen, the ratio of larvae and sealed brood, as well as if there are any pests, such as Varroa mites, inside the hive. These inspections can be time consuming and has the inherent danger of being stung from defensive bees within a hive. An application that allows users to be able to monitor their beehives from the comfort of their own homes would simplify the beekeeper's job. Such an application should have an interface that is easy to understand and be used to set the necessary parameters.

1.3 Graphical User Interfaces

Graphical User Interfaces, or GUIs, are a medium for casual users to interact with a complex system. This allows users to use advanced software in a simplified point-and-click interface. An example of a GUI is a digital calculator. In a digital calculator GUI, a user is shown multiple buttons. These buttons are mapped to a variety of tasks, ranging from simply displaying a digit on the screen to calculating division. In this example, a user is not required to understand the mathematics behind an action. However, through a series of clicks, a user can still complete the task.

1.4 Objectives

The main goal of this thesis is to create a honey beehive control system (BeeCon) with an extensive graphical user interface (GUI) where a user can control Raspberry Pi devices that monitor his or her beehives. This user interface will allow inexperienced users to control and monitor their beehives without having any advanced knowledge of the underlying system. This application will also need to be cross-platform, allowing use on the most common operating systems (OS). The main features of the user interface include: the ability to locate Raspberry Pi monitoring systems on a local area network (LAN), manual control of the system, setting preferences, the ability to download data, and analyze temperature data.

The ability to set preferences, such as the beginning and ending times for recording data and being able to manually start or stop recordings is essential to the operation of the acquisition software. The system can run independently with minimal or no user interaction once preferences have been set. Manual operation is also achievable within the system.

Downloading data to a local machine is important for beekeepers as it will allow a quick access to audio and video files being recorded. Since some parameters being set can change the way some of the data is seen or heard, for example the resolution of images or videos or frequency of the audio recordings, it is necessary for the system to allow for samples of audio and video to be downloaded. The system also allows beekeepers to observe their hives throughout the day.

Finally, giving the users the ability to search for their Raspberry Pis on a network will make logging into the available systems simpler. A typical user is not expected to know specific information about their network. Allowing them a way to scan their network and select a Raspberry Pi would prove to be a simpler experience.

Among the contributions of this thesis are:

- Determining how best to interface with the PyBeemon monitoring system.
- Developing a graphical user interface that makes the interactions between beekeepers and their hives very efficient.
- Providing basic analysis capabilities of the temperature and humidity data.
- Making accessibility of Raspberry Pis on home networks very convenient.

1.5 Thesis Overview

Over the course of this thesis, a system will be outlined that allows for beekeepers to monitor their beehives through a simplified user interface. In Chapter 2, background information will be briefly discussed. Chapter 3 will detail previous work done in habitat monitoring, as well as analysis of habitat data. Chapter 4 describes how BeeCon was implemented for beehive monitoring. Chapter 5 will overview the final product that was created. Chapter 6 will discuss conclusions and future expansion of BeeCon.

Chapter 2 - Background

2.1 Data Acquisition System

Raspberry Pis are microcomputers approximately the size of a credit card. The Raspberry Pi 3 Model B is used in this research. The Raspberry Pi 3 has a 1.2 GHz quad-core ARMv8 processor, 1GB of RAM, 4 USB ports, 40 GPIO ports, a camera interface, an SD card slot, and an ethernet port [3]. The storage space on the Raspberry Pi 3 is provided by the size of the SD card. This version of the Raspberry Pi allows use of USB microphones, GPIO temperature sensors, and the Raspberry Pi Camera module.

The Raspberry Pi Camera V2 was used in the project to record videos at the entrance of the beehive [4]. The Raspberry Pi Camera V2 has an 8 megapixel camera and can capture still images at a resolution of 3280 by 2464 pixels, while video capture can record at 1080p. The camera utilizes a Sony IMX219 sensor on board. The camera is fully compatible with every version of the Raspberry Pi. There is also a Python library named picamera for working with the camera [5]. The picamera library provides an interface for manipulating the camera in Python.

The AM2302 sensor was utilized to gather temperature and humidity data from the beehive [6]. The AM2302 sensor can detect temperatures from -40° to 80° Celsius (-40° to 176° Fahrenheit) with a 0.5° Celsius percent error. Humidity can be read with the 0-100% range with an accuracy of 2-5%. The sensor can be polled every 2 seconds for temperature and humidity data. Three wires are used on the sensor: power, ground, and data. Data is transferred to the microcontroller unit via a one-wire interface.

Audio recording was done with an eBerry USB microphone [7]. The connection to the Raspberry Pi 3 is made over a USB 2.0 connection. The microphone has a sensitivity of -67 dBV/pBar,-47 dBV/Pascal +/-4dB.

2.2 Software

This thesis describes the research and development for creating a control system (BeeCon) for an application called PyBeemon that has been developed in the Visual and Image Processing Lab at Appalachian State University. PyBeemon runs on a Raspberry Pi microcomputer that allows for recording of video, temperature, humidity, and two channels of audio. The hardware recommended by PyBeemon is the Raspberry Pi camera, an AM2302 temperature humidity sensor, and USB microphones. Users can set variables, such as start and end time, in a configuration file. The system will then read this file and run without any further interaction from the user. Figure 2.1 shows the flow of PyBeemon's data acquisition.

As shown in Figure 2.1, a Cron job, which is a Linux utility, allows scheduling of commands or scripts at specified times, starts the server process. The server is responsible for starting processes to record data such as audio. A client can then be used to relay commands from a terminal connected via a Secure Shell (SSH) connection to the server allowing control over the system. At the bottom of the image, it is shown how the system uses the configuration file in order to operate.

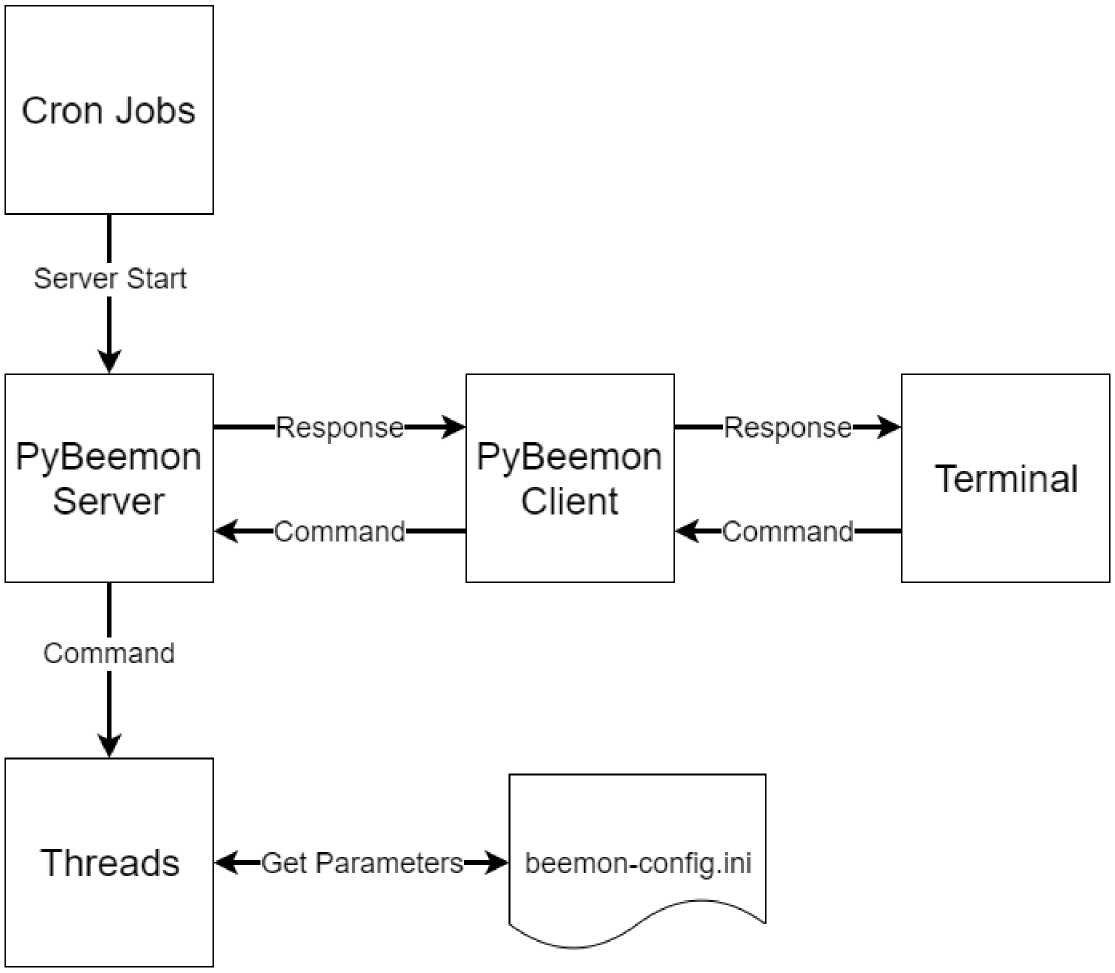


Figure 2.1: Basic flow of data acquisition through PyBeemon

PyBeemon allows usage of the software in two ways. The first is an autonomous mode where the system will run using only the information from the configuration file. The second is a manual mode where users can override the automated settings. In order to override these settings, a user logs in to the system from any command line. Once logged into the system, users are able to run commands, such as `start_video`, in order to control the system. This allows a GUI to control the system by sending commands to PyBeemon. The collection of software components and analysis tools written as part of this thesis make the control of PyBeemon possible. This collection of software is referred to as BeeCon, which is a product of this thesis.

Chapter 3 - Literature Review

Habitat monitoring systems have been utilized in many different fields of science. The most common uses for these systems are for studying animals and the environment. The system is utilized by this thesis is classified as a habitat monitoring system. Applying the requirements and recommendations from the research done throughout this section, a GUI is created to control and monitor the beehive systems.

Busse et al. have tried to analyze the full life cycle of animal monitoring technologies [8]. Specifically, this group has studied the development, generation, and use of animal monitoring technologies. Based on expert interviews, workshops, and a Delphi survey, they were able to conclude a number of steps to ensure proper growth within the systems. These steps include: resource-based fostering of network management, high level of scientific education and research, and openness to new challenges and demands. This thesis focuses on the development of such systems.

Lynch et al. studied the use of high resolution cameras to investigate the behavior of seabirds [9]. The goal of the paper was to build a low-cost and weather proof system that could reliably monitor the nesting habits of seabirds. During their experiment, they used a camera system called Gigapan which allowed them to take a panoramic photograph of the entire area of the nesting grounds. This allowed them to get a much better view of the entire flock at once instead of using many individual cameras. Once the pictures were captured, they were sent to a server in their laboratory. It was concluded that the pictures taken could extend or enhance their data collection methods.

Kumar et al. developed a system that could monitor both the environmental and animal data [10]. This data included body temperature, heart rate, air temperature, and humidity. They also claimed to be able to read stress levels of individual animals associating with the thermal humidity index. They used a ZigBee and an attached microcontroller to relay the readings which are read through a GUI in real time. The sensors are placed around the neck of the animal inside a weatherproof container and the data is transmitted via ZigBee to a relay station. The relay station then interfaces directly with a PC where the GUI displays the data. They were able to detect the health of the animals using this system.

Szewczyk et al. outlined a system that they used successfully in three different environments [11]. They described the layout of their system, which includes an on-site data center, transit network, patch network, and verification network. The data center is where all of the data from the nodes are sent, which will be stored locally and then sent to the web. The transit network is the network that runs between the sensor patches and the data center. The patch network is the group of nodes that have been deployed in the field. Finally, the verification network is used to verify the correctness of the patch network by reading data around the patch. This system was utilized on an island off the coast of Maine, University of California's James Reserve in the San Jacinto Mountain, and around California Redwoods in Sonoma County California.

Yaashuwanth et al. created a system for monitoring the health of dairy animals in real time [12]. To achieve this, they mounted several sensors around the necks of the animals. These sensors included a motion, airflow, temperature, piezoelectric, and an RFID tag. Once the controller had the readings, they were transferred to a robot utilizing a ZigBee. Once the robot has the data, it is then transferred to the user via SMS. The robot would follow a line on the ground around the pen of animals so that each module could transfer its data. They concluded that they could accurately read all data and quickly receive any warnings about the cows well being.

Zlinszky et al. discussed the importance of remote sensing within habitats [13]. They argue that studying the environment will be shifting from isolated experiments to remote sensing. They stated that traditional testing of habitats can be extremely time consuming, can lack precision, and can be very difficult to access. Remote sensing can be used to monitor a wide area of a habitat with little human interaction. This data is also being used at a national, regional, and local scale for city planning. They concluded that remote sensing could be used to quantify biodiversity in the same way satellite images quantified climate science.

Cerpa et al. discussed the importance of remote sensors environmental data collection [14]. Since in these small ecosystems power consumption is the most limiting factor.

They suggest using something called the Frisbee model, where in a perfect system every node would go into a low power mode while there is no data available. However, this is impossible since nothing would ever bring the nodes back into their regular mode. In the Frisbee Model if an object moves over a field of nodes all nodes within the radius of the moving object will be observing the target, while any node not within the radius will be in low power mode. This would save power and only cause a power drain on the nodes that need to be awake.

Polastre discussed his experiences using a wireless sensor network deployed on Great Duck Island [15]. They outline their requirements, architecture, implementation methods, and results. The Great Duck Island had many requirements that the sensor network had to adapt to in order to be effective. Some of these requirements were long lasting sensors, being able to manage the network from a distance, inconspicuous operation, and internet access. Their system architecture included a base station used for backing up data and connecting to the internet via satellite. The next step in the system is the transit network, which connects the gateways of the sensor networks to the base station. The gateway of the sensor networks receives data from the individual sensors and sends them across the transit network to the base station and finally to an off-site database. In order to deploy their network, they used a device called a Mica Mote with a weather shield. These devices were chosen for their robustness and processing power. During the writing of their paper the network had been running for four weeks and indicated that they could continue to power themselves for another six months. They implemented a system to replicate the database from the base station to an off-site location every fifteen minutes to offload the bandwidth of the satellite collection. This also helped them keep their data secure if the connection is ever lost.

Using wireless sensor networks, Qandour et al. discuss a method of analyzing audio data from honeybee hives in order to monitor health [16]. Their systems architecture consists of a sensor node, in their case a Beagleboard microcontroller, a microphone attached to the board, and a ZigBee to transmit data. Using this, they were able to collect data for analysis. They used different algorithms including: Principal Component Analysis, Support Vector Machines, and Linear Discriminant Analysis to classify their data. Using these methods, they were able to predict the infestation of a beehive solely by audio recordings.

Kale proposed a system for monitoring the traffic at the entrance of the beehives using videos taken above the entrance of a beehive utilizing computer vision [17]. The two main goals of that research were to be able to track bees and detect whether they are entering or leaving the hive, and count the number of bees entering or leaving the hive. The flow of the system implemented can be broken down into four main steps: acquisition, background subtraction, object detection, and tracking. It was concluded that using background subtraction on the video, an accurate segmentation was achievable. Using blob analysis on these segmented frames allowed precise detection of bees within a frame. However, for motion tracking, both implemented algorithms performed poorly and undercounted arrivals and departures.

Ghadiri monitored honeybees activity in front of their hives using surveillance cameras [18]. Furthermore, weather data to correlate with the videos is also obtained from the internet. Once the videos were recorded, they were sent to a server for analysis. A illumination-invariant change detection algorithm is implemented with the purpose of detecting motion of bees in front of a beehive. This method proved to be troublesome in that the threshold values needed to be optimized. Temperature and humidity data was also evaluated in tandem with signal to noise ratio algorithm in an attempt to find a correlation between the two data points. Using this data, it was shown that bee activity is negatively correlated with humidity.

A study by Mainwaring et al. on wireless sensor networks took place on an island near Maine [19]. During this study, a network of 43 nodes was used that would transmit data from the island to the internet. While the deployment time was relatively short, over a million readings from the nodes were transmitted. The paper argues that wireless sensor networks are a much more practical solution to habitat data collection. Some benefits that are listed are price, instant access to data as opposed to onsite data loggers, less habitat impact, and little to no need for human intervention. The authors concluded that nodes had a very high failure rate and didn't provide any meaningful data for their research. It did show that when viewing sensor data, one could use the readings to predict when a node would fail.

An analysis of two different subtypes of wireless sensor networks performed by Szewczyk et al. was done during a four month period [20]. The two types of networks were single hop networks and multi hop networks. Single hop networks are nodes connected directly to a single source of internet, while a multi hop network would allow nodes to pass data between other nodes with the end goal being the router connected to the internet. This experiment used two different types of probes to record data: one recorded weather data and the other recorded data of birds inside burrows. The conclusions were that these networks were based on their battery life expectancy over the course of the experiment. The single hop network outperformed the life of the multi hop network. The majority of the weather nodes for the single hop network died around the 125 day mark while the multi hop nodes died around the 65 day mark. The burrowed nodes for both single hop and multi hop networks died around the 40 day mark. This was concluded to be an issue with a sensor drawing more power than expected.

YingMing et al. focused on the uses of wireless sensor networks and outlines a framework of use [21]. While they found that most previous versions of wireless sensor networks used 8-bit MCUs at a low clock rate, they decided to use a 32-bit system that is compatible with a ZigBee. A ZigBee is a low power and inexpensive component that allows communication between other ZigBees within a certain radius. As their gateway node, they used two different systems. The first being a ZigBee to allow communication through the network to the sensors. The second was a General Packet Radio Service device that allowed data to be transmitted to a smartphone. They concluded that power consumption plays the biggest role within these networks and that everything within the system has to be optimized for power consumption.

Polastre proposed that wireless sensor networks are the future for scientific communities doing research on the environment [22]. Since each sensor can be apart of the environment and have a very small impact it can prove to be much more advantageous than other more bulky equipment. Local storage of data allowed these sensors to run aggregation, filtering, and compression algorithms. The ability for multiple sensors to communicate allows a group of nodes to perform more complex tasks, such as statistical sampling, data aggregation, and checking the system health. However, a full support system for these networks must be in place for the network to perform at full capacity. These include power management, sampling mechanisms, and a protocol for communication. Their system was deployed in two environments studying different types of data and the system was able to handle the data acquisition and transmission.

Rajendran et al. proposed the use of a collision free protocol within wireless sensor networks called TRAMA (traffic-adaptive medium access protocol) [23]. TRAMA claimed that it was both energy efficient and collision free. This protocol used a distributed election system to determine which node it can send to next. This allows nodes to go into idle the moment they have no data to send. The results were shown through a simulation of networks. Throughout the simulation, it was shown that battery performance can increase through allowing nodes to sleep for approximately 87% of their uptime. They claim an increase in overall throughput around 40% increase over S-MAC OR CSMA, as well as a 20% increase over 802.11.

Wang et al. introduce a system for preprocessing data within a tiered network designed for habitat monitoring [24]. This design used a two tiered system: the first tier consists of a macro node and the second consist of micro nodes. Within the ecosystem, there were many lower power micro nodes and only a few more powerful macro nodes. Two types of preprocessing were developed within the micro nodes in order to decrease transmissions between micro and macro nodes. The first type of preprocessing was event filtering, where the micro node parses incoming audio for relevance, discarding anything that was not. The second preprocessing technique was data reduction or compression. Once the relevant data was found, it was encoded and compressed and sent to a macro node. The experiment was conducted using speakers and ambient noise, such as traffic and wind to simulate the wild. This experiment, presented preliminary evidence that their method was both valid and effective.

Yong-Min et al. discussed the architecture and characteristics of wireless sensor networks [25]. In their paper, they discuss the best way to setup a wireless sensor network. The main characteristics of a wireless sensor network were explained including computing capabilities, communication capabilities, battery usage, dynamic network, self organizing network, and allowing multi hop. The network architecture shows five different layers: application, transport, network, data link, and physical. These all apply to the three different planes of the project, which are power management, mobility management, and task management. Different deployment protocols, among them were static and stochastic was also explained. The Static Protocol is used when you can predetermine the placement of sensors and stochastic is used when placement cannot be determined.

Krishnamachariet al. asserted that wireless sensor networks differ from traditional networks because they have energy constraints, redundant low rate data, and many-to-one flows [26]. During their analysis, they compared data-centric routing versus end-to-end routing. Data-centric routing is defined as a node sending data collected from sensors to the internet. This had the benefit of everyone node along the way can view this data and add its own. However, end-to-end routing simply sent its data independently of all other nodes and finds the shortest path available. As they reveal in their analysis, data-centric had two main issues, which are placement of sources and the network topology.

Ferdoush et al. discussed the use of Raspberry Pis and Arduinos as an open source and inexpensive way to leverage a wireless sensor network [27]. The network architecture uses an Arduino with RHT03 temperature and humidity sensors reading data. The data was then sent to the base station via a ZigBee module. The base station, a Raspberry Pi with an attached ZigBee, receives the data sent by the Arduinos, and sends the data through a router to a web app. The data was then displayed on a web app that was built with both PHP and HTML. A sample network in the Department of Electrical Engineering office area of the UNT Discovery Park Facility consisted of three nodes around the building to record temperature and humidity data. They were able to monitor the data from their web app during the entirety of their experiment.

Chapter 4 - Methodology

The Java programming language [28] is used to create BeeCon. Compiled Java code runs inside a virtual machine, allowing any operating system with Java installed the ability to run the Graphical User Interface. Java also allows packaging of applications into Java Archive (JAR) files, that are archive files storing the Java classes and metadata about the project. These files have the advantage of allowing users to simply double click to launch the application. This is one of the major advantages of developing in Java over other programming languages, such as C++ or Visual Basic.

An advantage of creating the BeeCon application in Java is the ability to make the application cross-platform. Cross-platform is defined as allowing users to be able to use the application on any operating system (OS). This will allow users to interact the application on Windows, Mac, and Linux operating systems. Once a user learns the application on one OS, they will not need to relearn the interface for a different OS, which will create an enhanced user experience.

A build system named Gradle was used for this project [29]. Gradle is a build automation tool designed for projects of all sizes. One of Gradle's best features is that it will only build as much of the project as needed. This means if portions of the project are not updated between builds, Gradle will not recompile these portions, which saves time. Gradle also allows the use of dependency repositories, which simplify the inclusion of libraries in a project. Instead of manually downloading and placing the library within the project, Gradle can handle all of this automatically by placing a command within the build file called gradle.build.

In order to design the interface, JavaFX was used [30]. JavaFX builds panels in FXML, which is a markup language based on XML. FXML can be written manually or it can be generated by software called Scenebuilder, which is a drag-and-drop implementation [31]. JavaFX allows for customization of panels using the CSS language, providing more control over the look of components.

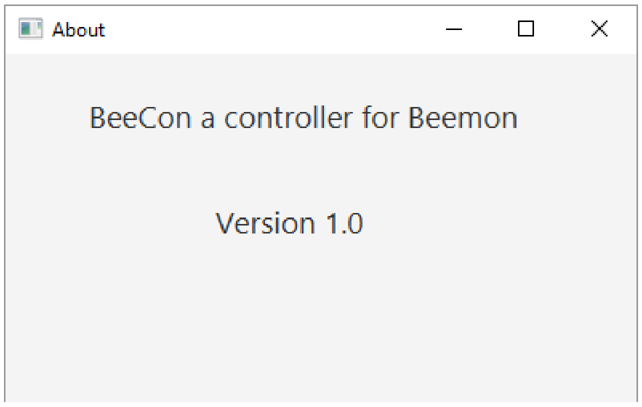
```
<?xml version="1.0" encoding="UTF-8"?>

<?import java.lang.*?>
<?import java.util.*?>
<?import javafx.scene.control.*?>
<?import javafx.scene.layout.*?>
<?import javafx.scene.paint.*?>
<?import javafx.scene.text.*?>

<VBox prefHeight="217.0" prefWidth="394.0"
    xmlns="http://javafx.com/javafx/8" xmlns:fx="http://javafx.com/fxml/1">
  <children>
    <AnchorPane maxHeight="-1.0" maxWidth="-1.0" prefHeight="329.0"
        prefWidth="640.0" VBox.vgrow="ALWAYS">
         <children>
            <Label layoutX="52.0" layoutY="25.0" text="BeeCon a controller for Beemon">
               <font>
                  <Font size="19.0" />
               </font>
            </Label>
            <Label layoutX="131.0" layoutY="91.0" text="Version 1.0">
               <font>
                  <Font size="19.0" />
               </font>
            </Label>
         </children>
    </AnchorPane>
  </children>
</VBox>
```

Figure 4.1: FXML used to create the About panel

In Figure 4.1 an example is of FXML is shown. The first seven lines of code set up all packaged dependencies. The next line of code begins a VBox pane which will contain the contents of this pane. The width is set to 394 pixels and the height is set to 217 pixels. Next, an Anchor Pane is created, and two labels are made within it. The font size is set within these Label tags.



BeeCon a controller for Beemon

Version 1.0

Figure 4.2: About panel created by FXML

In Figure 4.2, you can see the final product of the FXML. It shows a simple panel with two labels displaying the text set within the FXML. Each panel detailed within this chapter was designed with simplicity in mind. For the purpose of being user friendly. In order to achieve this each panel was made with the least amount of unnecessary components. Each text field and button was given a specific descriptor. The flowchart diagram of BeeCon is shown in Figure 4.3.

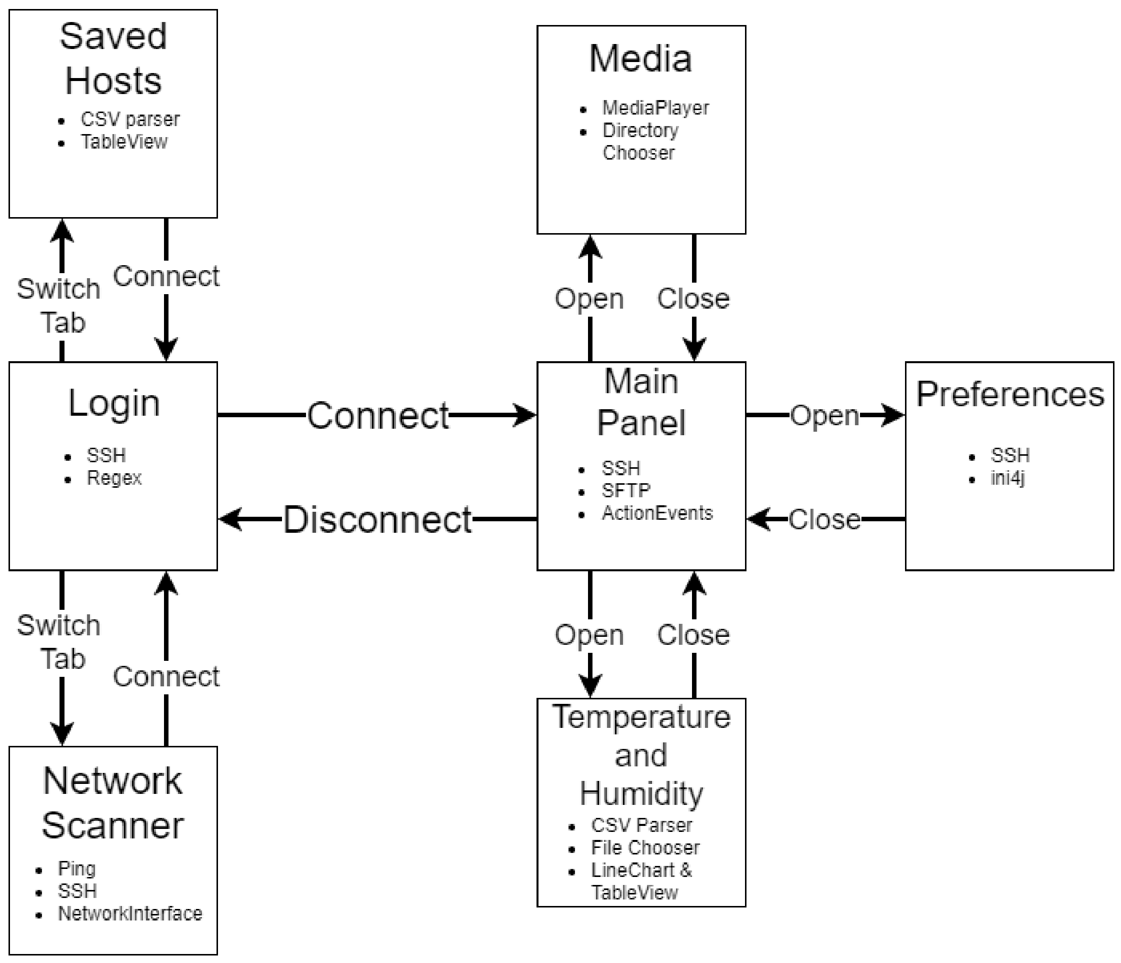


Figure 4.3: Flowchart diagram for BeeCon

BeeCon was designed to be as simple as possible. Before users are logged in they can use three different panels: Login, Saved Hosts, or Network Scanner. Each of these panels assists users in either finding information on their Raspberry Pis or logging into them. Once a user is logged into BeeCon the Main Panel is displayed. The Main Panel allows users to start and stop recordings as well as download snapshots of data. From this panel the Media, Preferences, or Temperature and Humidity panel can be opened. Once these panels are closed the Main Panel will be redisplayed. Each of these panels will be discussed in detail in the remainder of this chapter.

4.1 Interfacing with PyBeemon

Interfacing with PyBeemon only requires an internet connection from a computer to the Raspberry Pi controller. A connection has to be made over Secure Shell (SSH) in order to communicate with PyBeemon. Figure 4.4 shows how BeeCon connects to PyBeemon to send a command.

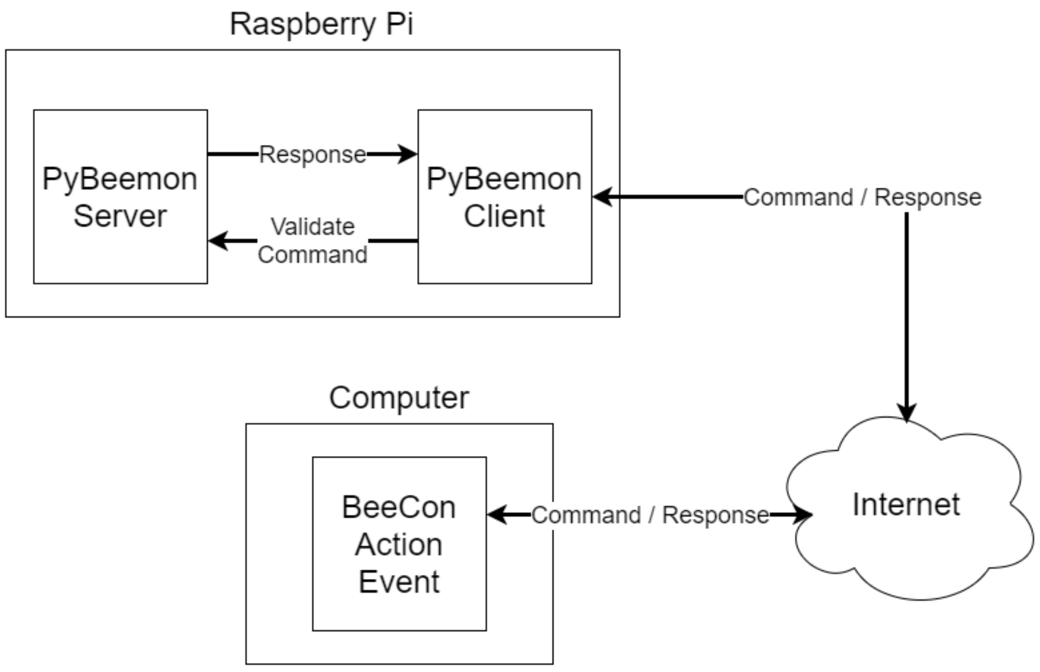


Figure 4.4: Connecting BeeCon and PyBeemon and sending a command

At the bottom of Figure 4.4, BeeCon has an action event, which is launched from a computer. Every action event has a predefined command bound to it. The command will then be sent over an SSH connection, which is connected to the Raspberry Pi that PyBeemon is hosted on. Once the command goes through the internet to PyBeemon, the PyBeemon client will read the command and send it to PyBeemon's server for validation. If the command is found to be valid, an appropriate command will be executed on PyBeemon and a response will be sent back to the client. If the command is found to be wrong, a bad command response will be returned. The response will then be routed back to BeeCon over the SSH connection. The action event will then read the response and modify BeeCon appropriately.

4.2 Network Scanning

In order to log into the Raspberry Pis via SSH, an IP address is required. Users of BeeCon are not expected to know what the IP address of the Raspberry Pi is or how to find it on their network. Having a network scanner solves this issue. This scanner will need to look for all devices on a network, and then display the found devices in BeeCon. This will allow users to be able to quickly log into their desired Raspberry Pis without any knowledge of the underlying system.

Initially, a naive solution was implemented. The entire range of Internet Protocol (IP) addresses was scanned and pinged in order to generate a list of machines connected to a network. However, this solution was discarded due to its inefficiency. Scanning from IP 0.0.0.0 to 255.255.255.255 allowing 10 ms for the code to determine if the machine is connected would take 1.3 years to fully scan. A new solution was implemented where the IP of the machine running BeeCon would be scanned instead. This allows the program to simply search through 256 different IPs at 10 ms per IP which is 2.56 seconds.

The network scanner first finds the host computer's IP address. The IP address is needed in order to scan the network's used IP range. In order to accomplish this, the NetworkInterface class, provided by Java, is used. The getNetworkInterfaces method returns all network interfaces on the current machine, in this case the computer running BeeCon. Once a list of interfaces is generated, each interface is tested to ensure they are not the local computer's IP, 127.0.0.1, or an inactive IP, as this would result in no Raspberry Pis being found. The result will be an IP address and a MAC address. The MAC address is filtered out and only the IP address of the computer on the network will be reported.

Using the IP found above, for example 192.168.1.12, the IP address is split on the last period so that it is now 192.168.1. Since IP ranges are from 0-254, this new IP is sent to a function which will scan from 192.168.1.0 to 192.168.1.254. For each of these IPs, the program will determine whether port 22 is open, which is the port used for SSH on Raspberry Pis. Scanning for this open port will give a list of all Linux machines with SSH enabled. If the scan finds an IP with this port open, it will then attempt to log into the machine with that IP with a guest account. If successful, the IP is added to the final list. Successfully logging into the guest account guarantees that the Raspberry Pi is running PyBeemon. An unsuccessful log in attempt will allow the system to discard the device from the list. Once the scan completes, a list of IP's for the available Raspberry Pis will be returned and listed within BeeCon, which can be seen in Figure 4.5.

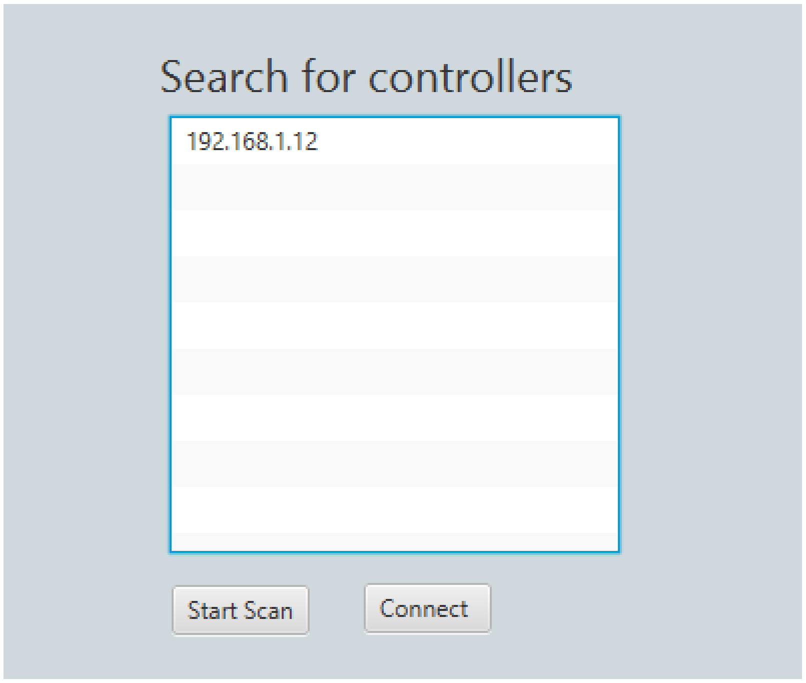
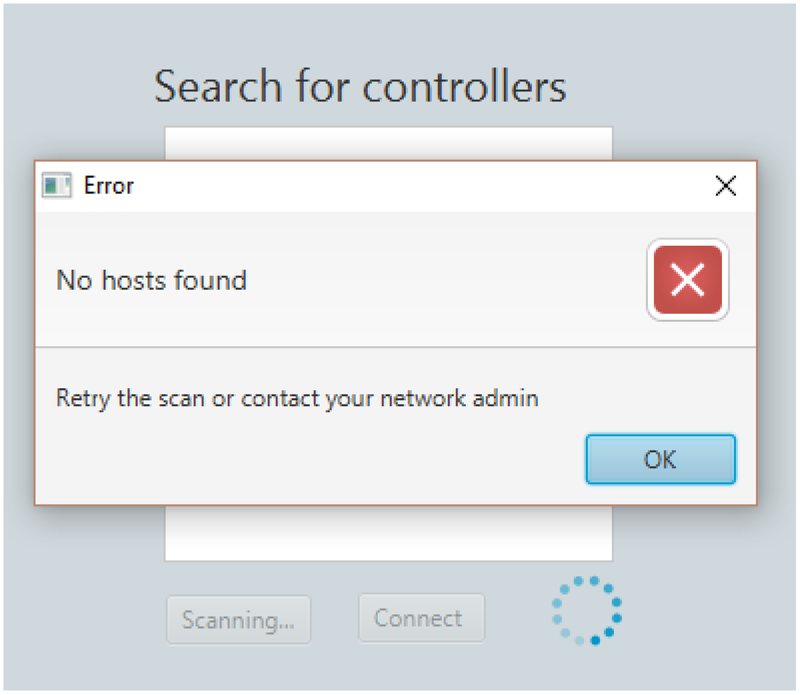


Figure 4.5: An example of a successful scan listed within BeeCon

In figure 4.5, a single Raspberry Pi was found and is listed inside a TableView. If none are found, an alert will show that the scan was unable to find any usable machines, which can be seen in Figure 4.6.



No hosts found

Retry the scan or contact your network admin

Figure 4.6: An unsuccessful scan resulting in an error message

In Figure 4.6, an error message is displayed communicating to the user that no device was found on the network. The user is then requested to try the scan again or contact their network admin. If the scan does find devices to connect to, the user will be able to select it and be taken to the login screen. The IP address and port are inserted into the appropriate text fields.

4.3 Logging into BeeCon

Before a user can control PyBeemon, they must log into the BeeCon system. Logging into BeeCon is synonymous to logging into PyBeemon. As shown in Figure 4.7 in order to log in, users must input a username, IP address, password, and port number. If the user does not know how to get the IP address, he/she can follow the instructions in Section 4.2 to find the IP address of their Raspberry Pis without any technical knowledge.

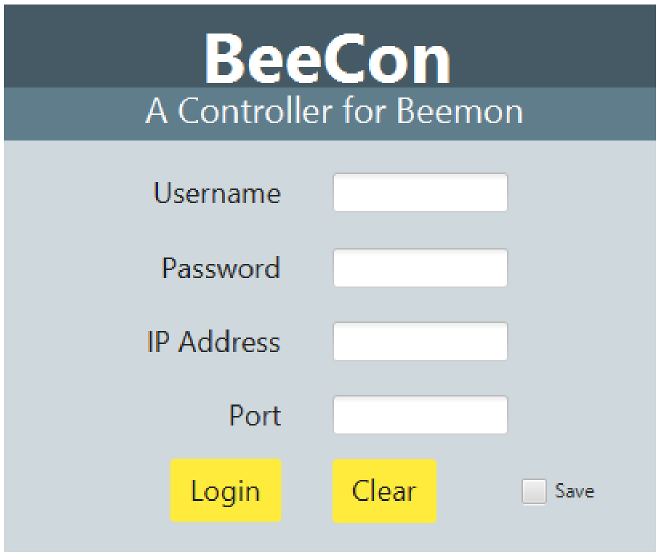


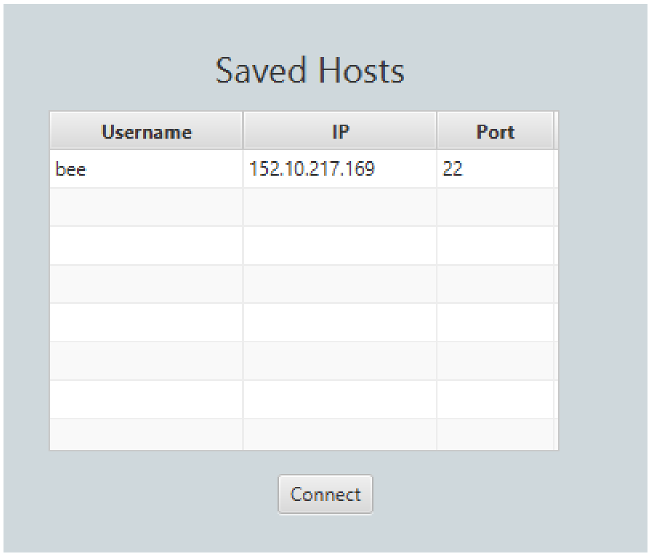
Figure 4.7: The Login panel for connecting to connect to PyBeemon

Figure 4.7 shows the panel used for logging into the system. The four text fields are used to set the parameters described by their labels specifying their use. The two buttons are below: the Login button will initiate the log in process and the Clear button simply resets all the text fields. To the right of the buttons, there is a Save check box that will be discussed later. Once a user has entered their information into the appropriate text fields, the Login button can be pressed and a login function is called.

This function first checks that all the user information is provided and is valid. If the username, IP, or password text fields are empty, the user will receive an error requesting all fields to be completed. Next, the IP address will be validated. Validation is done through the Apache Commons library InetAddressValidator. InetAddressValidator allows the isValidInet4Address and isValidInet6Address functions to be run on the user submitted IP. If the IP is not valid, an alert will be displayed asking for a valid IP. Otherwise, the port will be checked. If the port text field was left blank, the login function will default to port 22, since this is the default port used by SSH. If the user enters their own port, it will be checked to ensure it is a number within the range of valid ports, which is 1 - 65535. If the number entered is outside of this range, an error will be displayed asking the user to enter a valid port number. If the port is valid but is not open on the device, the log in process will fail resulting in an alert message requesting the user to check their input.

Once all user information has been validated, an SSH connection will be attempted. If the connection is successful, a session variable will be set. The users information will then be stored and the session is disconnected. The main control panel is displayed and the Login panel is hidden. If the connection could not be made, an alert will be displayed informing the user. If the user selected the Save checkbox and the connection attempt was successful, the username, IP address, and port will be stored in a Comma Seperated Value (CSV) file. This file is stored within the BeeCon directory, which is created when BeeCon is first launched. This CSV file is used by another panel called Saved Hosts. Saved Hosts allows users to be able to quickly log in to Raspberry

Pis that have been logged into previously. In Figure 4.8, a Raspberry Pi's information can be seen stored within the Saved Hosts panel.



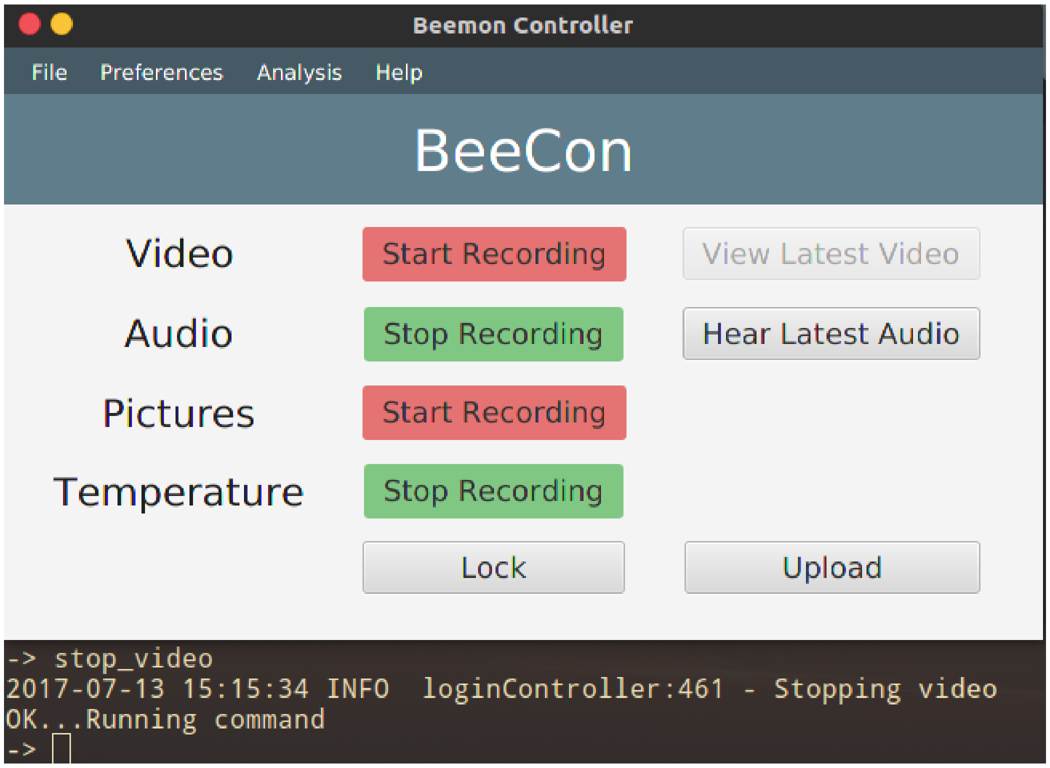
Username	IP	Port
bee	152.10.217.169	22

Figure 4.8: Saved Hosts panel for previously used hosts

The Saved Hosts list is scrollable in the case that users owns more than eight Raspberry Pis. The user can simply double click a host in the list, or single click and then press the connect button. The username, IP, and port will be automatically entered into the Login panel, then the user will be sent to the Login panel.

4.4 Manual Control

The ability for users to send commands, such as starting or stopping video, is essential for users. While PyBeemon can be run autonomously, users may need to manually override the system in order to perform routine maintenance to the controllers. Sending commands from the controller to the Raspberry Pi must be done through a SSH connection. The Java Secure Channel (JSCH) library was used in order to make a SSH connection and send commands from Java to a Linux machine. PyBeemon allows these commands to override the automatic mode. For example, if video was being recorded automatically and the controller sent a stop_video command, PyBeemon would not attempt to restart the video until the next day. This allows for the controller to simply map commands to button events. In Figure 4.9, the Stop Recording button for video was just pressed.



```
-> stop_video
2017-07-13 15:15:34 INFO  loginController:461 - Stopping video
OK...Running command
->
```

Figure 4.9: Running stop_video

When the Stop Recording button is pressed the stop_video command is sent to PyBeemon over the SSH connection. This can be seen at the bottom of 4.9. In order for the controller to send these commands, a connect function was created, which can be seen in Figure 4.10.

```java
/**
 * Starts a session, channel, and commander
 */
private void connect() {
    logger.info("Connecting to shell");
    JSch ssh = new JSch();
    try {
        // Create new SSH session with user information
        setSession(ssh.getSession(getUsername(), getServer(), getPort()));
        getSession().setConfig(config);
        getSession().setPassword(getPassword());
        getSession().connect(30000);

        // Create a new channel, which will be set to shell
        // This mimics a terminal
        setChannel(getSession().openChannel("shell"));
        OutputStream input = getChannel().getOutputStream();

        // Set the commander, this has the ability to send
        // commands to the mock shell
        setCommander(new PrintStream(input, true));
        getChannel().setOutputStream(System.out, true);

        // Connect to the channel
        getChannel().connect(3 * 1000);

        // Set up buttons and windows in GUI

    } catch (JSchException e) {
        logger.error("Could not connect to server");
        // Display alert
    }
}
```

Figure 4.10: The code for the connect function

Figure 4.10 shows the code written for the Connect function. After a message is logged in the system, a JSCH object named ssh is created. Using this object, a new session is created using credentials entered by the user. These include username, password, IP address, and port number. A channel is then created with the parameter of "shell". This will allow the program to mimic a shell, allowing commands to be entered and the subsequent responses to be displayed. An OutputStream is created and used to create a PrintStream that is used for sending commands. The channel's output is sent to System.out, and finally the channel is connected to. If any errors occur during this connection period, they are handled in the catch statement below the code. In this error code, a log message displaying the error is logged, as well as an alert box simply stating that the connection has been interrupted. Followed by a call to the disconnect function, which will disconnect from any sessions and channels, and force the user back to the login screen. Once the connection has been established, commands can be sent to a sendCommand function.

The code shown in Figure 4.11 shows how a command is sent to PyBeemon. The sendCommand function takes in a String value, such as start_video. The connection is then tested, and if the connection is still open, then the message will be sent through the mock shell created by the connect function. A delay is then executed for 100 ms, which stops all of the commands from being sent too frequently. If the connection is not open, then the function will call the disconnect function, which will close all of the open connections and will send the user back to the Login screen.

```
/**
 * Sends a pre-made command to the terminal for executing and delays for 100ms
 *
 * @param command command string that will be sent to the terminal
 */
private void sendCommand(String command) {
    if (getSession().isConnected()) {
        getCommander().println(command);
        delay(100);
    } else {
        disconnect();
    }
}
```

Figure 4.11: The code for the sendCommand function

Once a button is clicked, the appropriate click event function is called. These functions will call the sendCommand function with start or stop commands for their respective button name. For example, the Audio button will call sendCommand with start_audio if the audio is currently not recording. However, if the audio is recording, stop_audio would be called. The current state of the button, recording or not recording, is displayed by the buttons. If audio is currently recording, the text of the button will say “Stop Recording” and the color of the button is green. If the audio is currently not recording, the button will display the text “Start Recording” with a red background. This can be seen in Figure 4.9. Since the video is not currently recording, the button is red. In contrast, while the audio is recording the button is green.

Since manual override of the system is not corrected by the system until the next day, a lock system was needed in order to prevent users from making accidental changes to their recordings. An example of the buttons being locked can be seen in Figure 4.12.

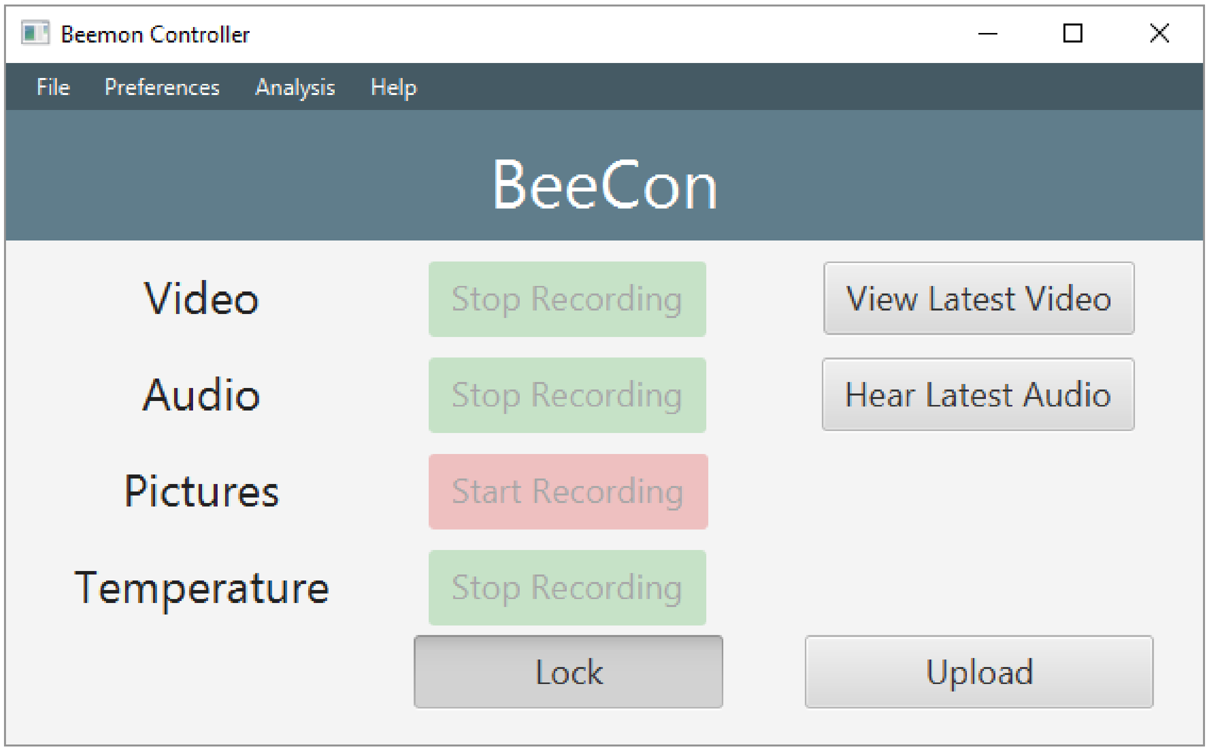
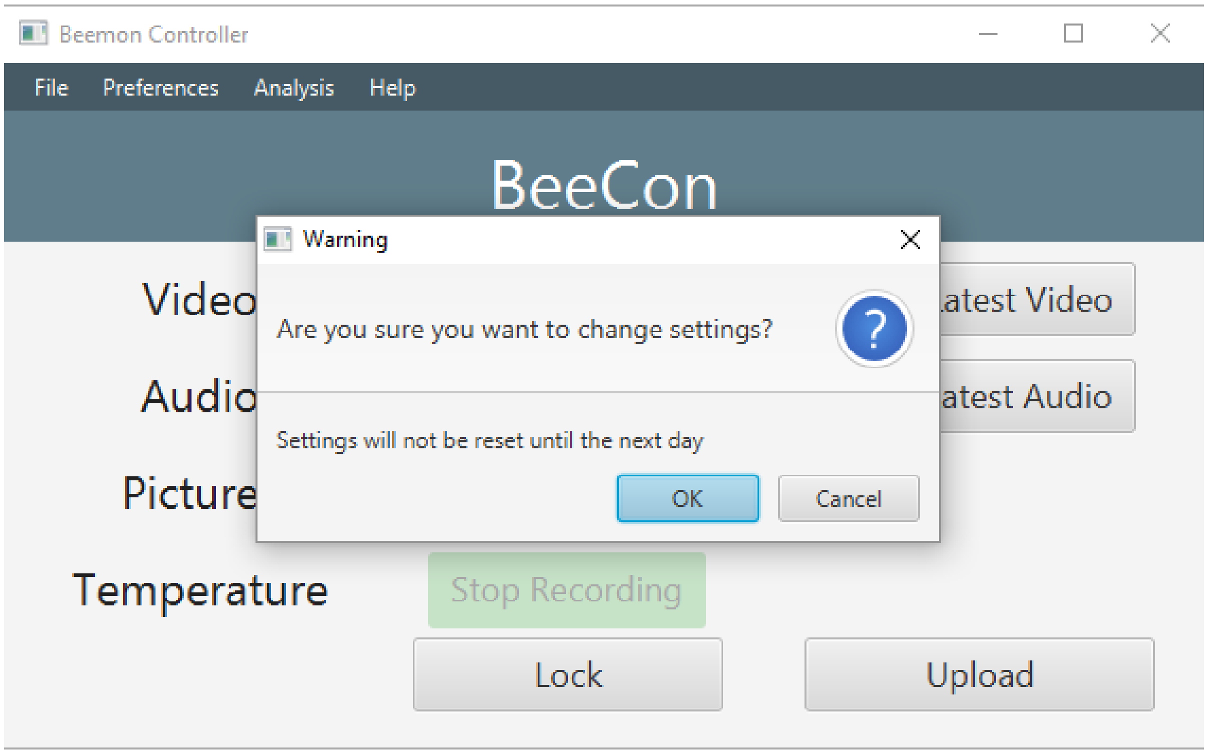


Figure 4.12: Locked buttons

The lock button is toggled on when the user logs into the system. In order to intuitively show the user that the lock is currently on, the buttons are slightly greyed out, the button is unable to be clicked, and the lock button is shown as toggled. When a user presses the toggle button, an alert will show that informs a user that if manual changes are made, these changes will not be reverted until the next day. This alert can be seen in Figure 4.13.



Are you sure you want to change settings?

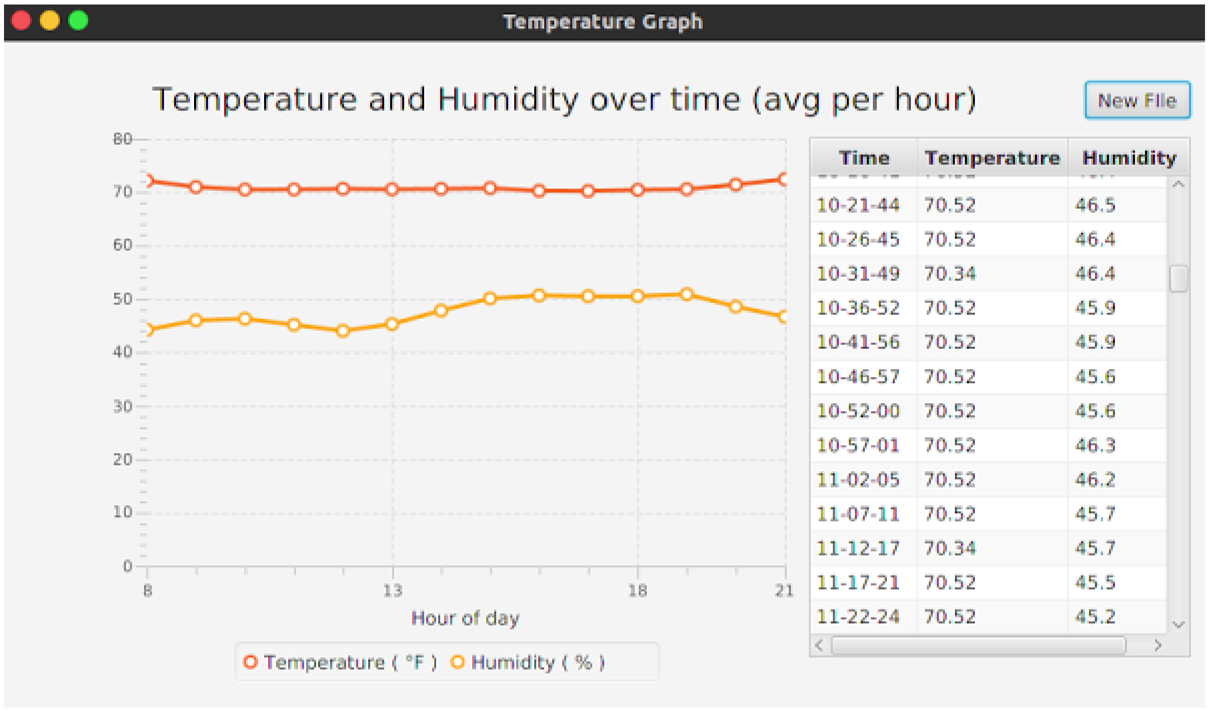
Settings will not be reset until the next day

Figure 4.13: Warning when unlocking buttons

If the user clicks Ok, the buttons will no longer be disabled and will show their original colors. The lock button will also no longer be toggled to be shown as on. If the user hits Cancel, the alert will disappear and the buttons will remain in their same state.

4.5 Temperature and Humidity analysis

PyBeemon records temperature and humidity data into CSV files. These files get automatically uploaded to a location specified by a user over File Transfer Protocol (FTP) each night. BeeCon allows a user to open these CSV files and see a brief analysis of the data contained within them. On the panel, a line graph showing both temperature and humidity over the duration of the day is displayed, as well as a table that shows all of the data as a list. This can be seen in Figure 4.14.



Time	Temperature	Humidity
10-21-44	70.52	46.5
10-26-45	70.52	46.4
10-31-49	70.34	46.4
10-36-52	70.52	45.9
10-41-56	70.52	45.9
10-46-57	70.52	45.6
10-52-00	70.52	45.6
10-57-01	70.52	46.3
11-02-05	70.52	46.2
11-07-11	70.52	45.7
11-12-17	70.34	45.7
11-17-21	70.52	45.5
11-22-24	70.52	45.2

Figure 4.14: Temperature and humidity panel

A graph is displayed showing the average temperature per hour, while the table will show every reading that was recorded for the entire day. Users can scroll down through the TableView in order to analyze the readings.

When the temperature and humidity panel is opened, it requests a CSV file to take in. If a CSV file is not given, the panel will still be displayed. However, the graph and table will be empty. A button is available to open a CSV. When a CSV is selected, BeeCon will store all of the readings and times into an array. Then, using the array, averages of temperature and humidity are calculated and stored in a separate array. The array of averages is passed to a graph function where it is inserted into a graph and displayed, while the list array is sent to a TableView class where it will be displayed.

4.6 Changing Preferences

PyBeemon utilizes a .ini file in order to save all of the user preferences. Some of these preferences include start and stop times for video, audio, and temperature. While the user could log in to the Raspberry Pi and edit this file through the command line, this cannot be expected of less technical users. Implementing a preference editor is needed in order to allow these users to edit their preferences without needing to know the technical aspects behind it.

In order to download a copy of this .ini file from the Raspberry Pi to the local computer running BeeCon, the SSH File Transfer Protocol (SFTP) was used. Since SSH is needed in order to connect to PyBeemon, SFTP is a simple solution for downloading and uploading files.

Figure 4.15 shows the code for downloading a .ini file. In the beginning of the code the download function creates a session in the same way as Figure 4.10. Once a session is created, then the channel is created, and is set to SFTP and connected. The .ini file is stored within the /etc folder on the Raspberry Pi's file system. A cd command is ran in order to change the directory from the home folder, where the SFTP connection will take

```java
/**
  * Gets .ini file from server you're connecting to and stores it inside a tmp file
  * Sets the file to the correct field for reuse once finished
  */
 private void getRemoteFile() {
     JSch jsch = new JSch();
     try {
         // Create a session and connect
         sess.connect();
         // Create a channel, set it to SFTP, and connect
         ChannelSftp channelSftp;
         channelSftp = (ChannelSftp) sess.openChannel("sftp");
         channelSftp.connect();
         // Change the directory to /etc
         channelSftp.cd("/etc");
         // Create an InputStream and run a .get command for the .ini
         InputStream in = channelSftp.get("beemon-config.ini");
         // Create a temporary file and download contents
         File localFile = File.createTempFile("beemon-config-", ".ini");
         FileOutputStream targetFile = new FileOutputStream(localFile);
         int count;
         while ((count = in.read()) != -1) {
             targetFile.write(count);
         }
         // Disconnect and close
     } catch (Exception e) {
        // Log error and exit
     }
 }
```

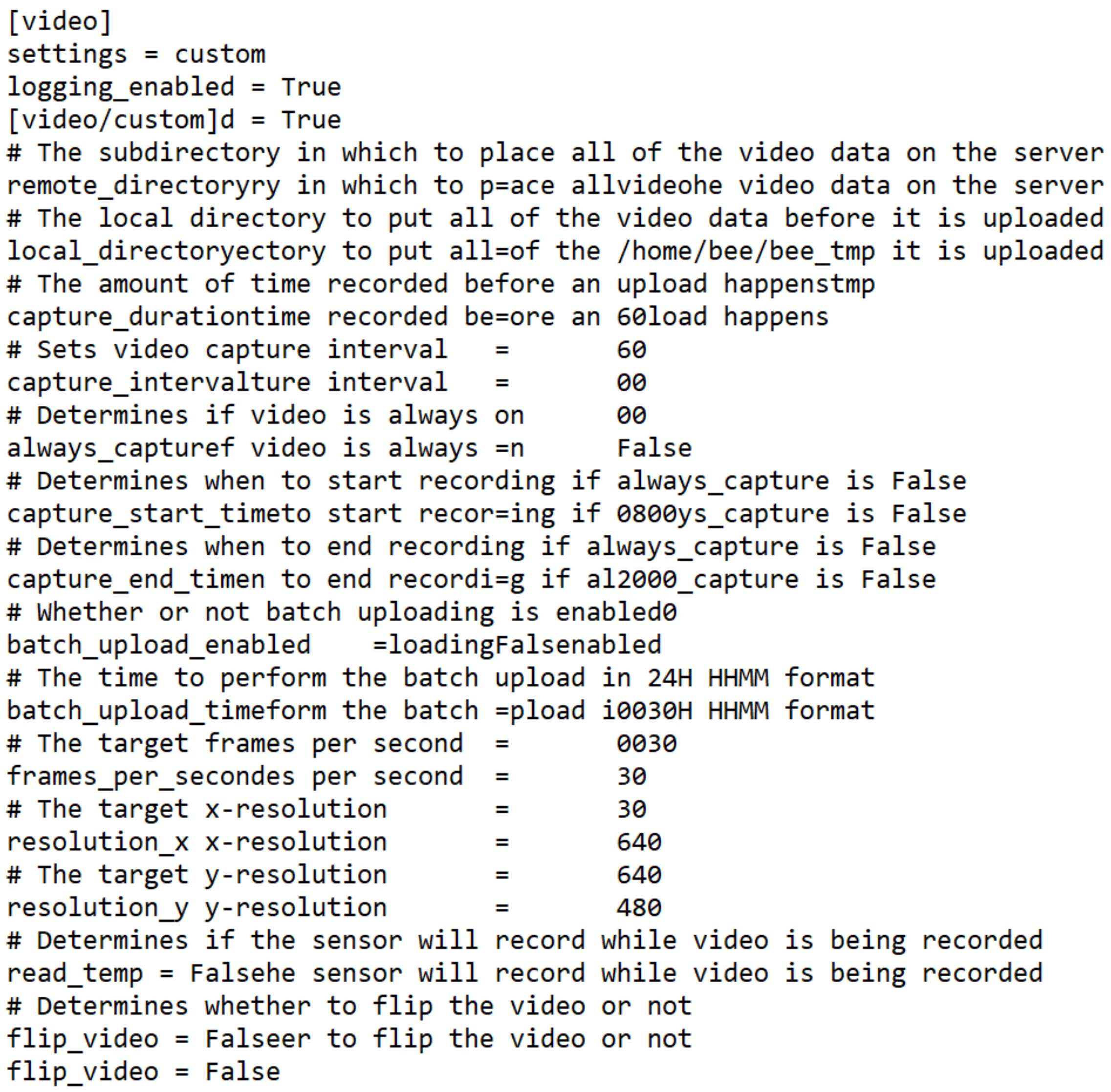
Figure 4.15: Code for the download function

you by default, to the /etc folder. The SFTP connection will run a get command for the .ini file and store the contents inside an InputStream. A temporary file is created using the Java File command createTempFile. This command will create an empty file in the temporary file directory native to the OS, with a given prefix and suffix. For example, the result could be beemon-config-1234567.ini stored in the temporary directory. An

OutputStream is created with the target being the temporary file. Then, the contents are read from the InputStream file and passed to the OutputStream file. Once the download is completed, all connections, channels, and streams are closed. The newly created temporary .ini file's location is saved within a variable.

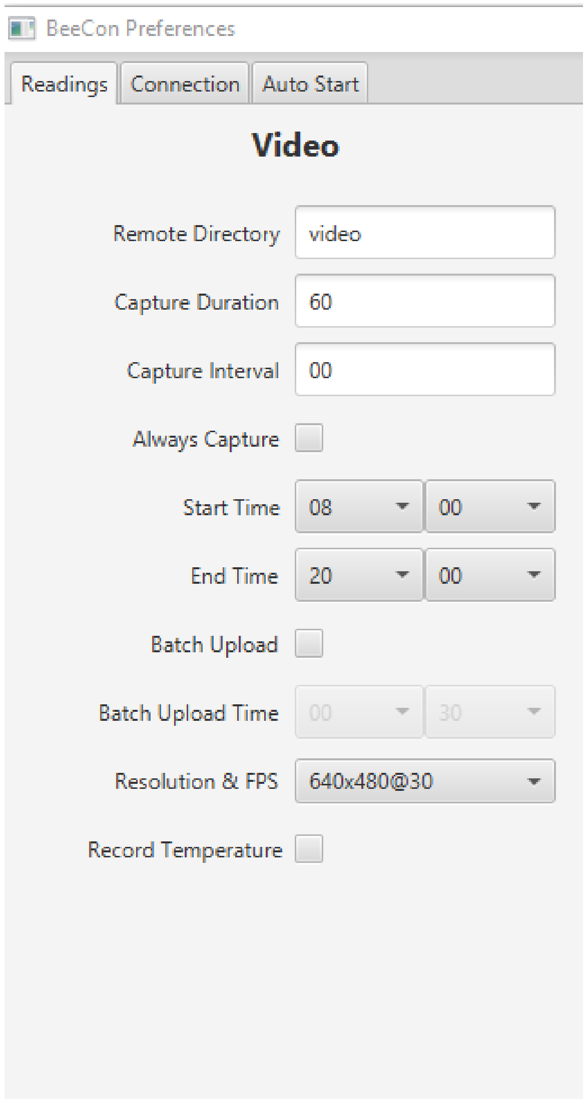
Once the .ini file has been downloaded and the location is set, the different fields within the file are set in BeeCon. Figure 4.16 shows how the fields are set on the Raspberry Pi.

Figure 4.17 displays how BeeCon presents the same information. Fields that need to be bound within certain parameters, such as time, resolution, and Frames Per Second (FPS), are put in drop down menus. This ensures that users cannot set a value that is outside the bounds of the program. Users can then save their settings locally, download a fresh copy of the configuration file, or upload their changes to replace the old configuration file. Sending the new copy of the configuration file to the server is implemented similarly to the download function shown in Listing 4.15. An SSH connection is made and is then defined as an SFTP connection. The file is uploaded into the appropriate directory on the Raspberry Pi using a put command. This command will overwrite the old file, leaving the new changes in its place.



```
[video]
settings = custom
logging_enabled = True
[video/custom]d = True
# The subdirectory in which to place all of the video data on the server
remote_directoryry in which to p=ace allvideohe video data on the server
# The local directory to put all of the video data before it is uploaded
local_directoryectory to put all=of the /home/bee/bee_tmp it is uploaded
# The amount of time recorded before an upload happenstmp
capture_durationtime recorded be=ore an 60load happens
# Sets video capture interval   =       60
capture_intervalture interval   =       00
# Determines if video is always on      00
always_capturef video is always =n      False
# Determines when to start recording if always_capture is False
capture_start_timeto start recor=ing if 0800ys_capture is False
# Determines when to end recording if always_capture is False
capture_end_timen to end recordi=g if al2000_capture is False
# Whether or not batch uploading is enabled0
batch_upload_enabled    =loadingFalsenabled
# The time to perform the batch upload in 24H HHMM format
batch_upload_timeform the batch =pload i0030H HHMM format
# The target frames per second  =       0030
frames_per_secondes per second  =       30
# The target x-resolution       =       30
resolution_x x-resolution       =       640
# The target y-resolution       =       640
resolution_y y-resolution       =       480
# Determines if the sensor will record while video is being recorded
read_temp = Falsehe sensor will record while video is being recorded
# Determines whether to flip the video or not
flip_video = Falseer to flip the video or not
flip_video = False
```

Figure 4.16: Configuration file on the the Raspberry Pi


BeeCon Preferences

Readings | Connection | Auto Start

Video

Remote Directory	video
Capture Duration	60
Capture Interval	00
Always Capture	☐
Start Time	08 : 00
End Time	20 : 00
Batch Upload	☐
Batch Upload Time	00 : 30
Resolution & FPS	640x480@30
Record Temperature	☐

Figure 4.17: Preference panel showing the video preferences

4.7 Downloading Audio and Video

PyBeemon can store all video, audio, and weather files of the user's choosing. This is done through any FTP server. However, a user may want to quickly check in on their beehives and see how the hive is doing without having to access the server. BeeCon allows users a simple way of downloading the latest audio or video recording in order to give the beekeepers a quick snapshot of their hive's health. The media files are displayed within BeeCon in order to provide a simplified experience for less technical users. Once the media is loaded, the user will be able to decide if they want to keep the file and save it to their computer, or if they would like to delete the file.

A download button was added to the main panel that is clickable when audio or video files are currently being recorded. These concepts are explained in further detail in Section 4.4, and an example of these buttons can be seen in Figure 4.9. Since downloading audio and video use the same process only the download for video will be detailed. Once a user clicks this button, it will call a function called download. Downloading will need to utilize a SFTP connection similar to the one described in Section 4.6. However, once the SFTP connection has been made, other steps must be taken. PyBeemon stores files within a directory named bee_tmp located on the Raspberry Pi.

An example path to a video file could be: bee_tmp/video/2017-07-07/12-30-21.h264. In order to find the latest video recording, BeeCon must get the current date and change to that date's directory within bee_tmp/. Once this has been achieved, the contents of this folder will be sorted in ascending order by name, and the first in the list will be the currently recording file. JavaFX does not support playing .h264 video files, so the video file will need to be converted from .h264 to .mp4. This step will not occur for audio files. The tool avconv will be used in order to do this [32]. avconv is a Linux command line tool used to convert both audio and video files quickly. The video file will be converted and saved within the same directory and created with the same name. The new .mp4 file will then be downloaded into the BeeCon directory described in Section 4.3. The mp4 file is removed and the SFTP connection is closed.

The newly saved video file is sent to another function named showVideo. The function showVideo will open a new panel that will display the video within a MediaPlayer [33]. MediaPlayers provide controls for media files, which include the playing, stopping, and pausing of files. The video is placed within the MediaPlayer and is set to automatically play and loop continuously. Within this panel, a user can choose to save their file, which will open a directory chooser for ease of use. A user can also choose to delete this video, which will simply delete the file from the BeeCon folder.

Chapter 5 - Results

5.1 Deployment

In order to fully test the system, a Raspberry Pi 3 was deployed at a beehive. An Ethernet cable was ran from the router inside the beekeeper's home to the Raspberry Pi mounted onto the beehive. An AM2302 sensor was attached to the Raspberry Pi to collect temperature and humidity data inside the beehive. A Raspberry Pi Cam V2 was used to collect video data and audio recording was achieved by an eBerry USB microphone placed within the beehive. In Figure 5.1, the Raspberry Pi attached to a bee hive can be seen.

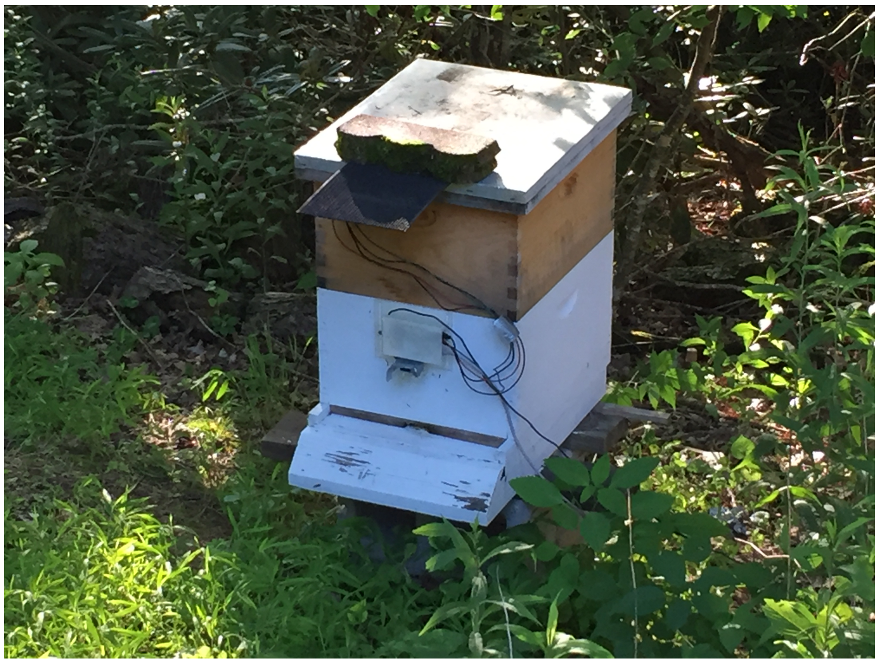


Figure 5.1: An example of a Raspberry Pi 3 data acquisition system

In order to attach the Raspberry Pi and keep it safe from the elements, a custom 3D printed case was designed. The microphone and temperature sensor can be seen being routed through the top of the hive. The camera is fixed to the Raspberry Pi case and placed over the entrance to the beehive.

The latest version of PyBeemon was installed on the Raspberry Pi. The default configuration was used for collecting audio, video, temperature, and humidity data. PyBeemon was configured to send all data recorded to a server on Appalachian State University's campus. Automatic start was also turned on for each recording. This allowed the Raspberry Pi to run autonomously while testing of BeeCon occurred.

5.2 BeeCon Usage

Two laptops were used in order to test BeeCon. One laptop was using macOS Sierra, while the other dual booted the Windows 10 and Ubuntu 16.04 operating systems. The setup for each was the same, the Java Development Kit (JDK) 8 needed to be installed. Once installed, all three operating systems were able to launch BeeCon successfully. However, the official JDK from Oracle was needed, instead of OpenJDK [34]. OpenJDK is an open source alternative for Linux operating systems. If OpenJDK was installed instead of the official JDK, BeeCon could not be launched.

After the installation of the JDK, BeeCon was launched and the initial screen was displayed. This screen can be seen on each OS in Figure 5.2.

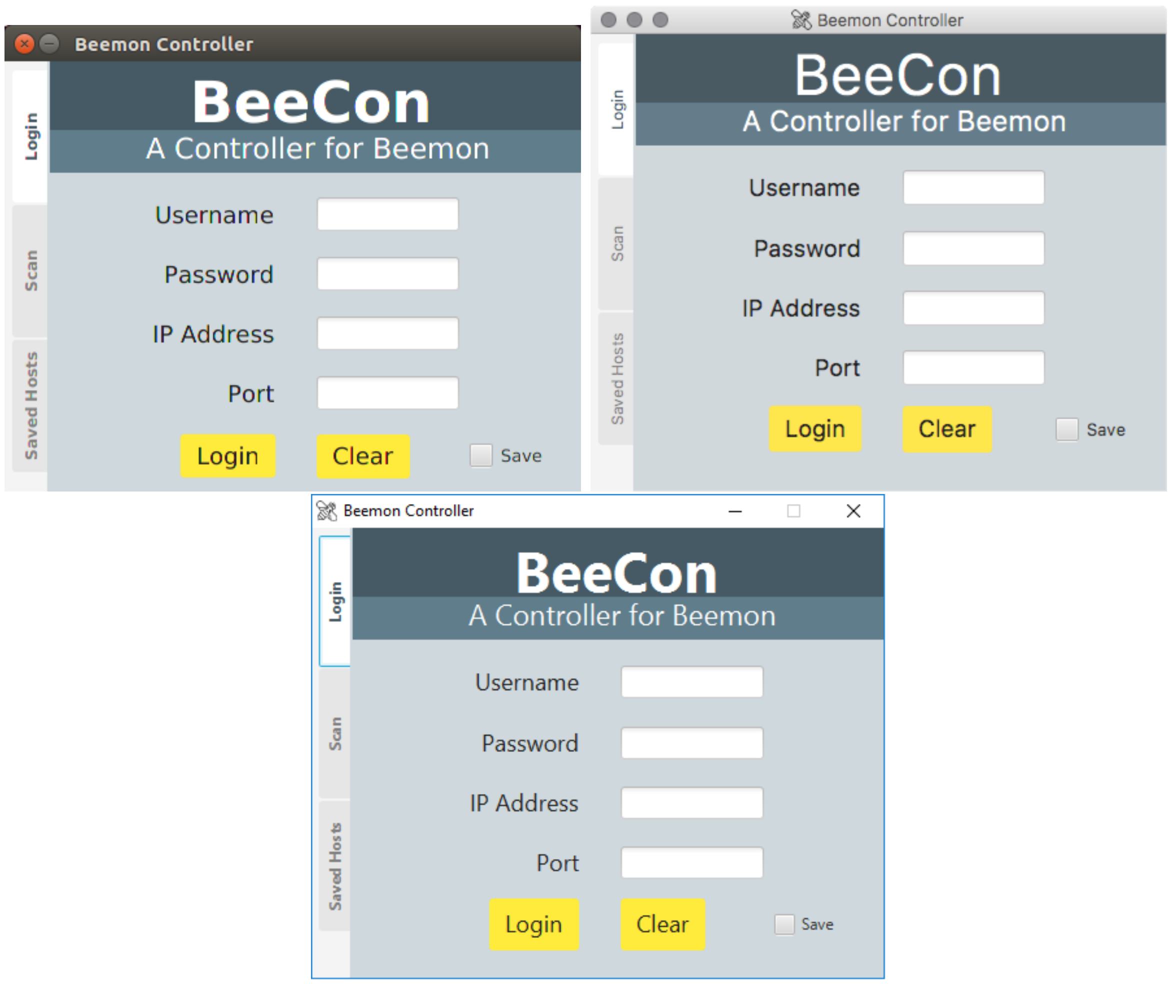


Figure 5.2: Login screen on Linux, Mac, and Windows

The initial screen displayed to a user is the Login panel, which is described in detail in Chapter 4 Section 4.3. From this screen, users can log in to their Raspberry Pi, or they can select from one of the other tabs on the left of this screen. From these tabs, users can scan their network, or select a saved host in order to log in to a Raspberry Pi. In Figure 5.3, the network scanner is shown to have found two Raspberry Pis.

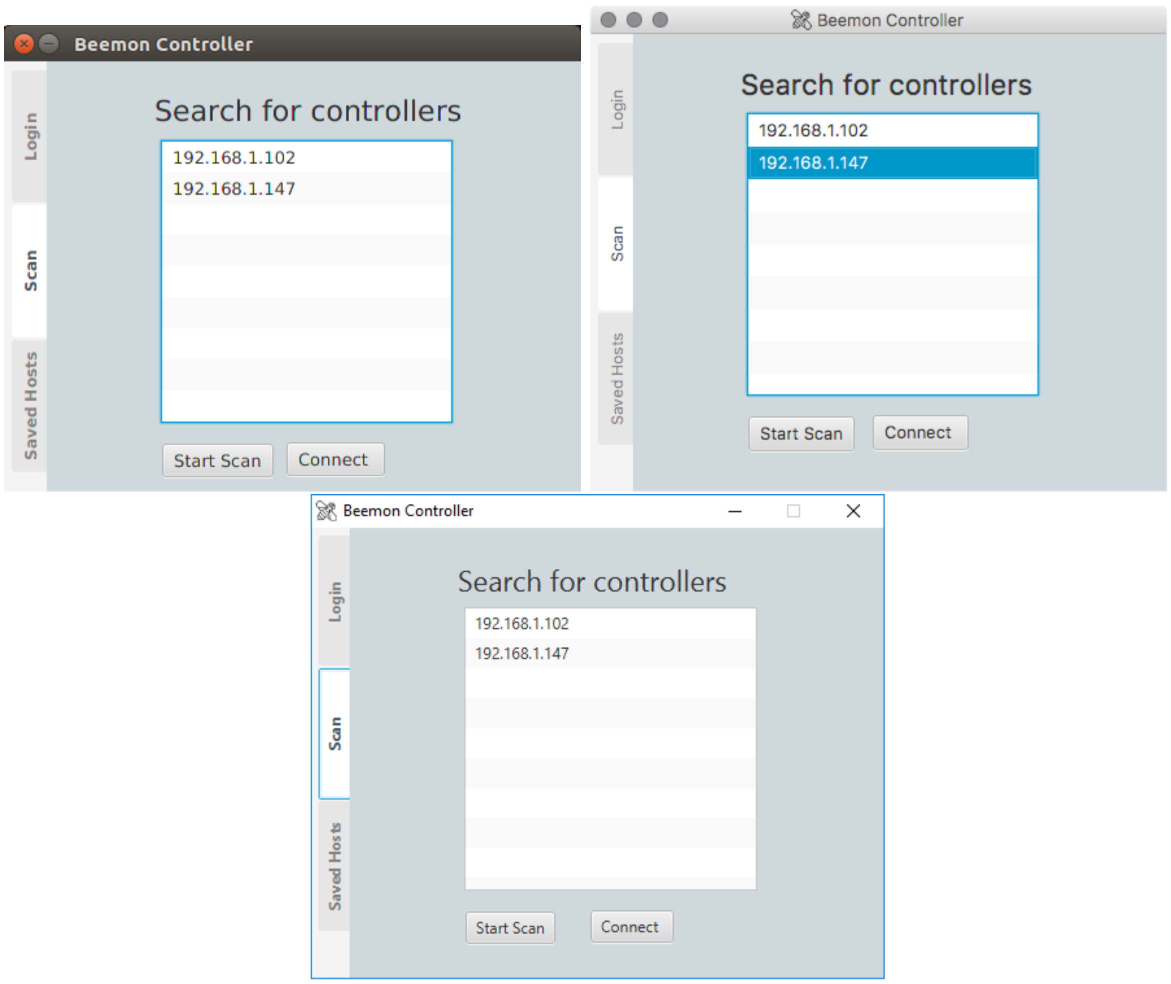


Figure 5.3: Network Scan screen on Linux, Mac, and Windows found two IPs

Two Raspberry Pi IP's were found on all three operating systems and are listed on the panel shown in Figure 5.3. The IP address 192.168.1.147 was selected and the Connect button was pressed. The IP found in the network scanner is copied to the Login tab and inserted into the hostname text field. The username and password were inserted into their appropriate fields and a connection was established. Upon successful connection, the main panel is displayed. This panel can be seen in Figure 5.4.

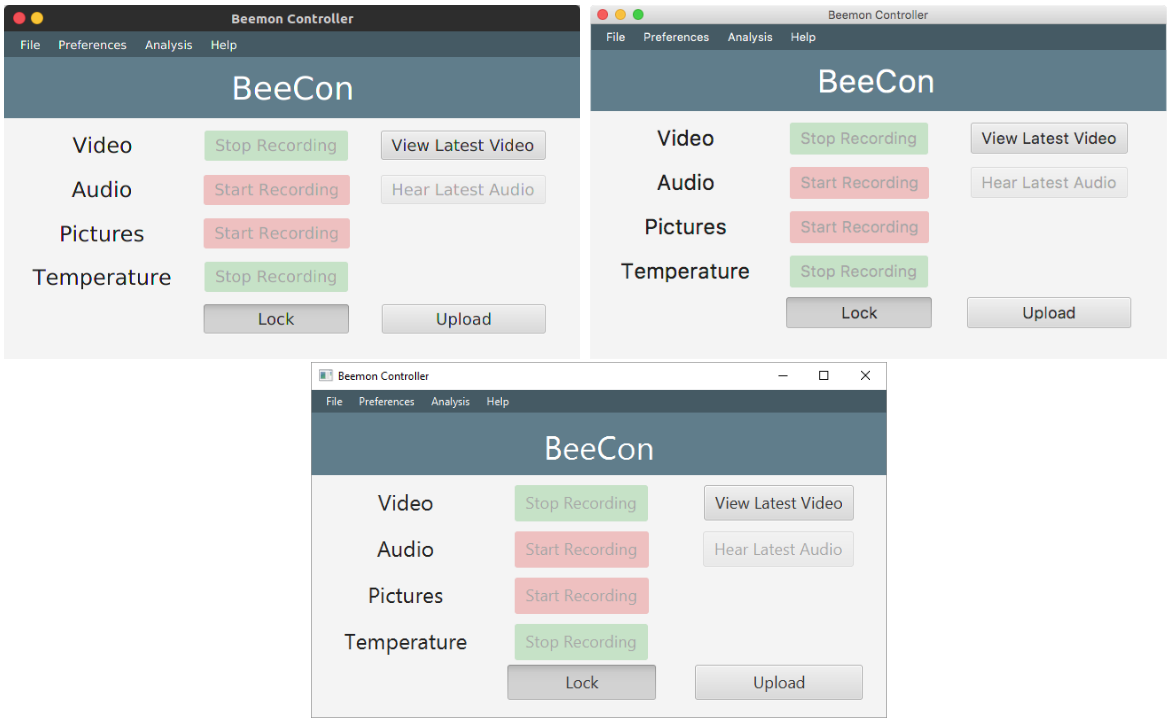


Figure 5.4: Main panel displayed on Linux, Mac, and Windows

On this panel, users can control their Raspberry Pis or download sample videos. In Figure 5.4, the video can be seen recording, while audio, temperature, and pictures are currently not recording. When the user first logs in, the panel is locked in order to protect users from unwanted button presses. In order to unlock the panel, the lock button must be pressed and the buttons can then be used to control PyBeemon. Figure 5.5 shows the main panel unlocked and with the audio button pressed.

With the panel unlocked and the audio button pressed, the text and color will change to indicate that it is currently recording. While video is currently recording, the View Latest Video button is available to be pressed. Pressing this button will download

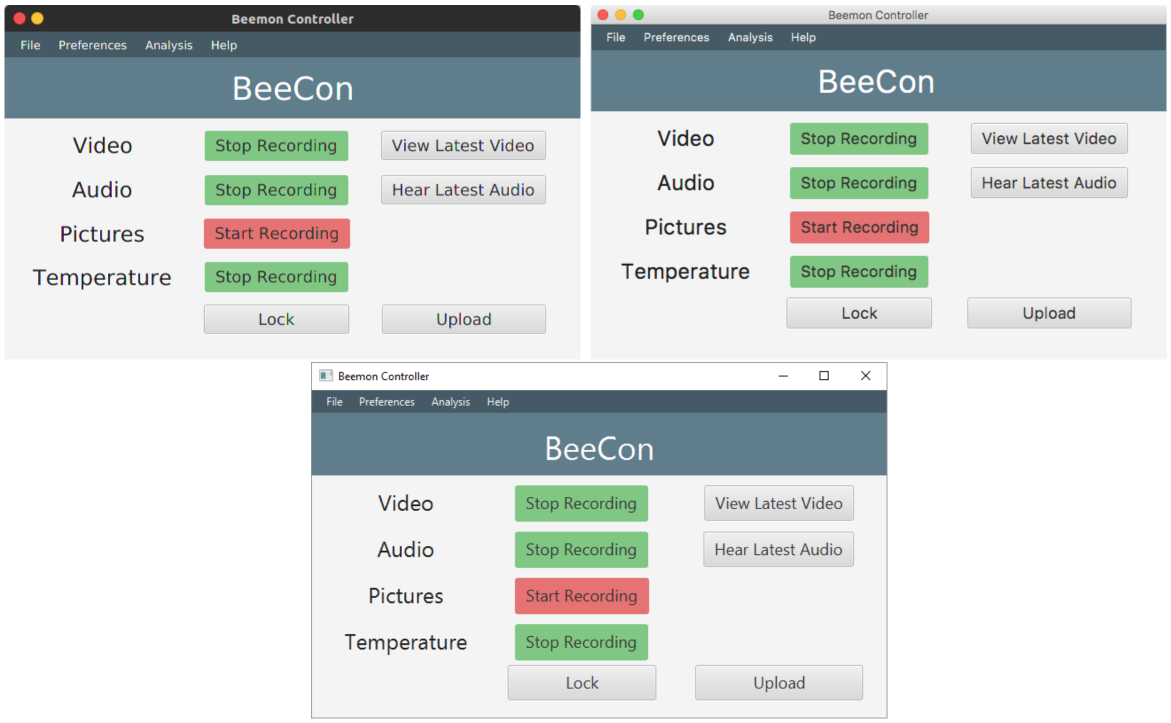


Figure 5.5: Main panel unlocked and audio started displayed on Linux, Mac, and Windows

the currently recording video and display it within a panel in BeeCon. The resulting panel can be seen in Figure 5.6. It should also be noted that downloading the latest audio file would function the same as video.

Once the video panel opens and displays the video, the user can either press the Save or Delete button. Pressing Save will open a directory chooser that allows users to save the video wherever they would like. Delete will simply remove it from the BeeCon directory. When either is pressed, the video panel will close and the main panel will be displayed once again.

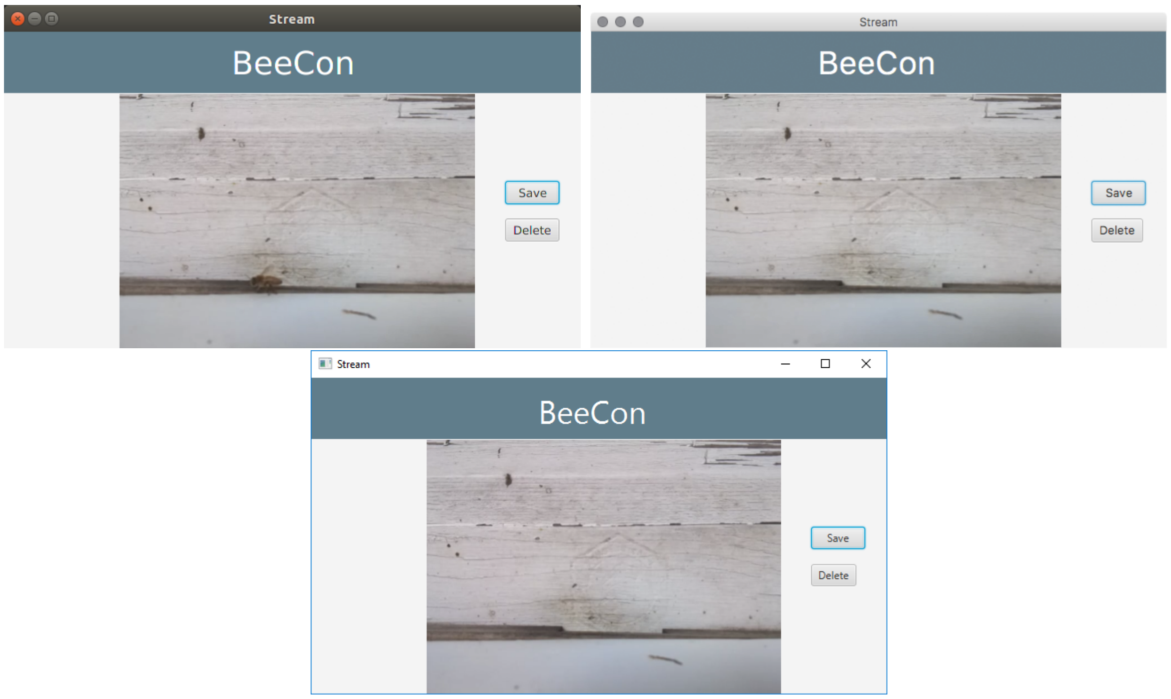
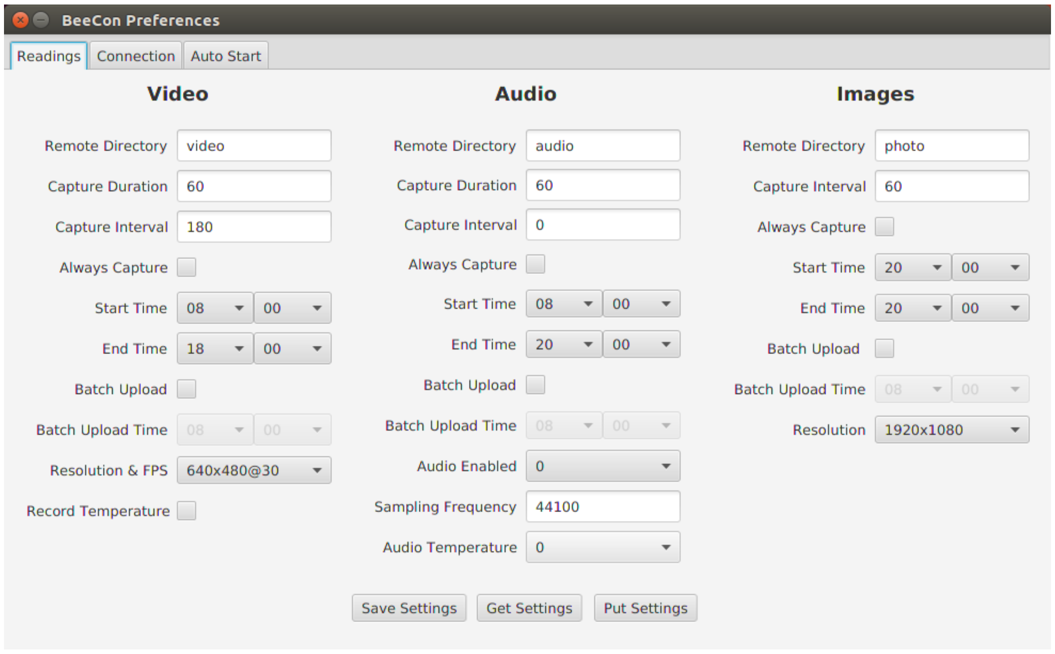


Figure 5.6: Downloaded video playing in BeeCon on Linux, Mac, and Windows

The menu bar on top of the main panel allows users to change the preferences of PyBeemon or analyze their temperature and humidity data. Selecting the Preferences option in the menu and clicking Edit Preferences will display the Preferences panel. This panel be seen in Figures 5.7, 5.8, and 5.9.



Video

Remote Directory: video
Capture Duration: 60
Capture Interval: 180
Always Capture: ☐
Start Time: 08 : 00
End Time: 18 : 00
Batch Upload: ☐
Batch Upload Time: 08 : 00
Resolution & FPS: 640x480@30
Record Temperature: ☐

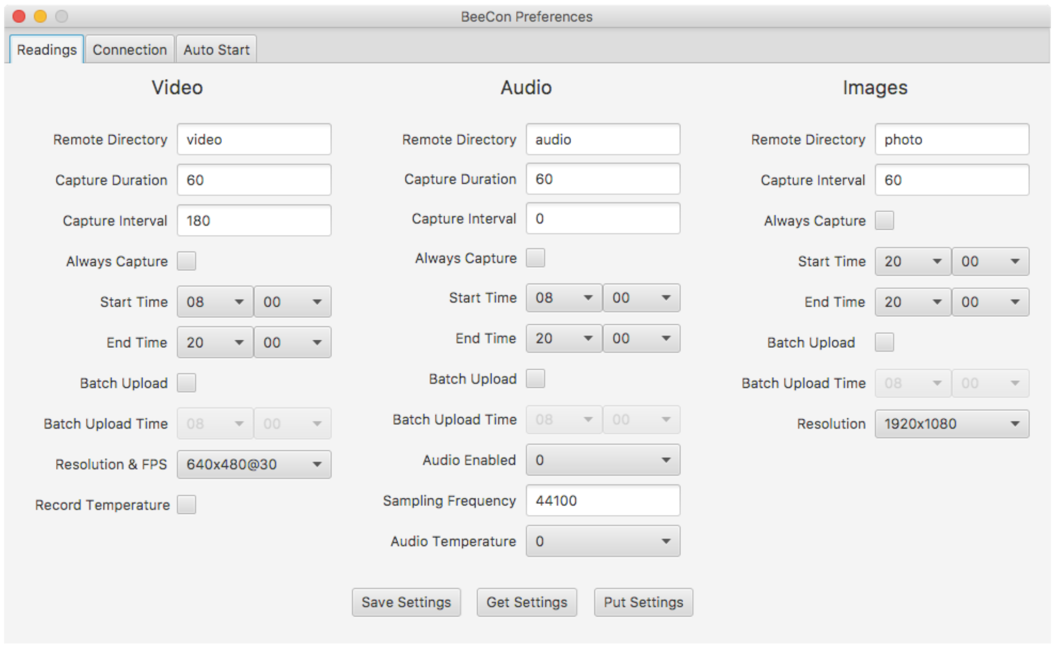
Audio

Remote Directory: audio
Capture Duration: 60
Capture Interval: 0
Always Capture: ☐
Start Time: 08 : 00
End Time: 20 : 00
Batch Upload: ☐
Batch Upload Time: 08 : 00
Audio Enabled: 0
Sampling Frequency: 44100
Audio Temperature: 0

Images

Remote Directory: photo
Capture Interval: 60
Always Capture: ☐
Start Time: 20 : 00
End Time: 20 : 00
Batch Upload: ☐
Batch Upload Time: 08 : 00
Resolution: 1920x1080

Figure 5.7: Preference panel on Linux



Video

Remote Directory: video
Capture Duration: 60
Capture Interval: 180
Always Capture: ☐
Start Time: 08 : 00
End Time: 20 : 00
Batch Upload: ☐
Batch Upload Time: 08 : 00
Resolution & FPS: 640x480@30
Record Temperature: ☐

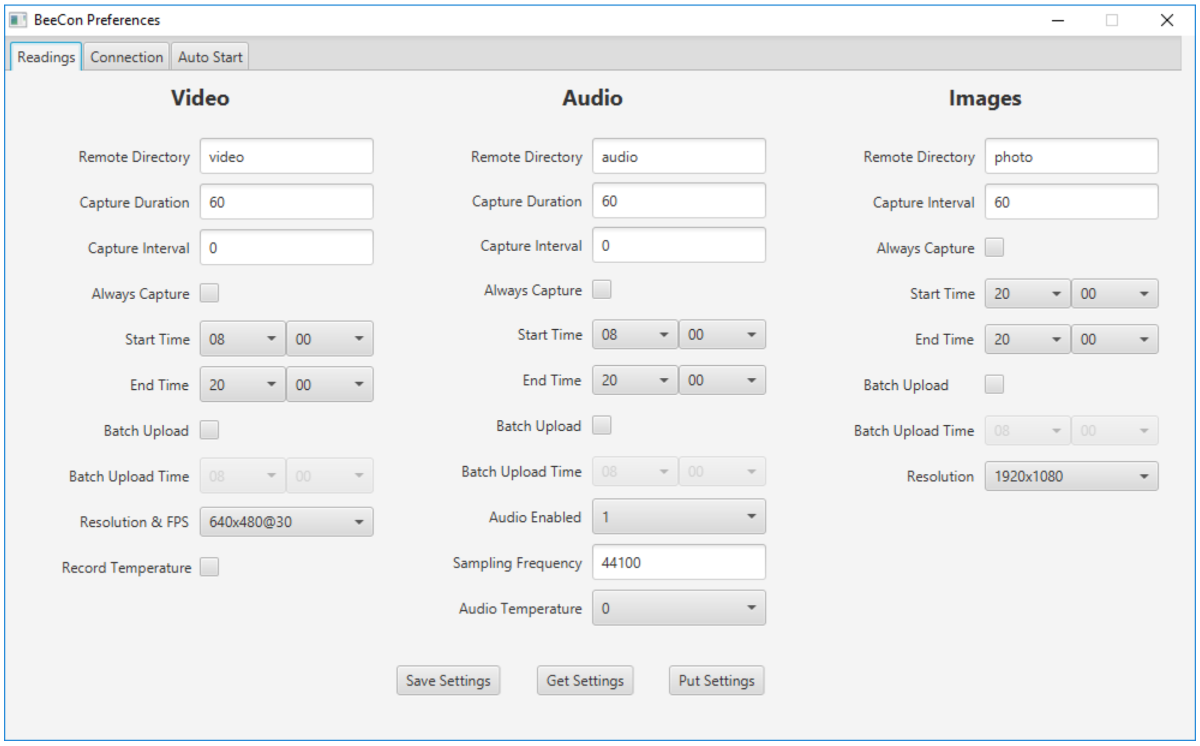
Audio

Remote Directory: audio
Capture Duration: 60
Capture Interval: 0
Always Capture: ☐
Start Time: 08 : 00
End Time: 20 : 00
Batch Upload: ☐
Batch Upload Time: 08 : 00
Audio Enabled: 0
Sampling Frequency: 44100
Audio Temperature: 0

Images

Remote Directory: photo
Capture Interval: 60
Always Capture: ☐
Start Time: 20 : 00
End Time: 20 : 00
Batch Upload: ☐
Batch Upload Time: 08 : 00
Resolution: 1920x1080

Figure 5.8: Preference panel on Mac



Video

Remote Directory	video
Capture Duration	60
Capture Interval	0
Always Capture	☐
Start Time	08 00
End Time	20 00
Batch Upload	☐
Batch Upload Time	08 00
Resolution & FPS	640x480@30
Record Temperature	☐

Audio

Remote Directory	audio
Capture Duration	60
Capture Interval	0
Always Capture	☐
Start Time	08 00
End Time	20 00
Batch Upload	☐
Batch Upload Time	08 00
Audio Enabled	1
Sampling Frequency	44100
Audio Temperature	0

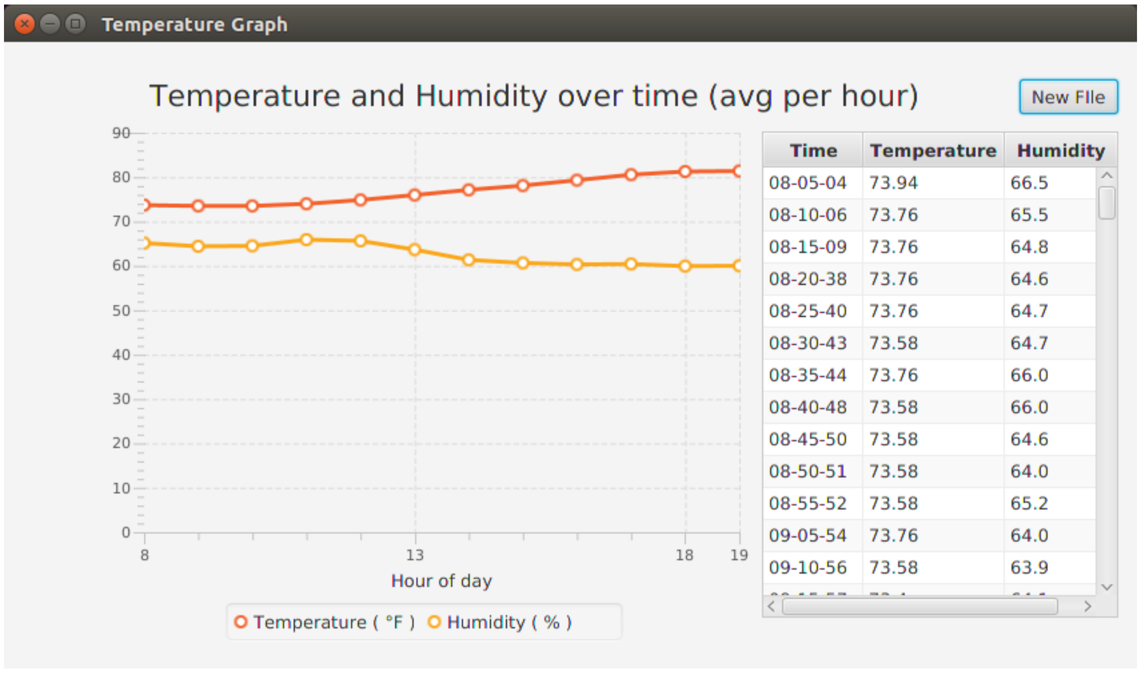
Images

Remote Directory	photo
Capture Interval	60
Always Capture	☐
Start Time	20 00
End Time	20 00
Batch Upload	☐
Batch Upload Time	08 00
Resolution	1920x1080

Figure 5.9: Preference panel on Windows

On this panel, users can edit all of their preferences in PyBeemon. Once a user is done editing their changes, they can press the Put Settings button to save their new preferences to PyBeemon. The Save Settings button will save a temporary copy of their preferences and the Get Settings button will download a new copy of the preferences.

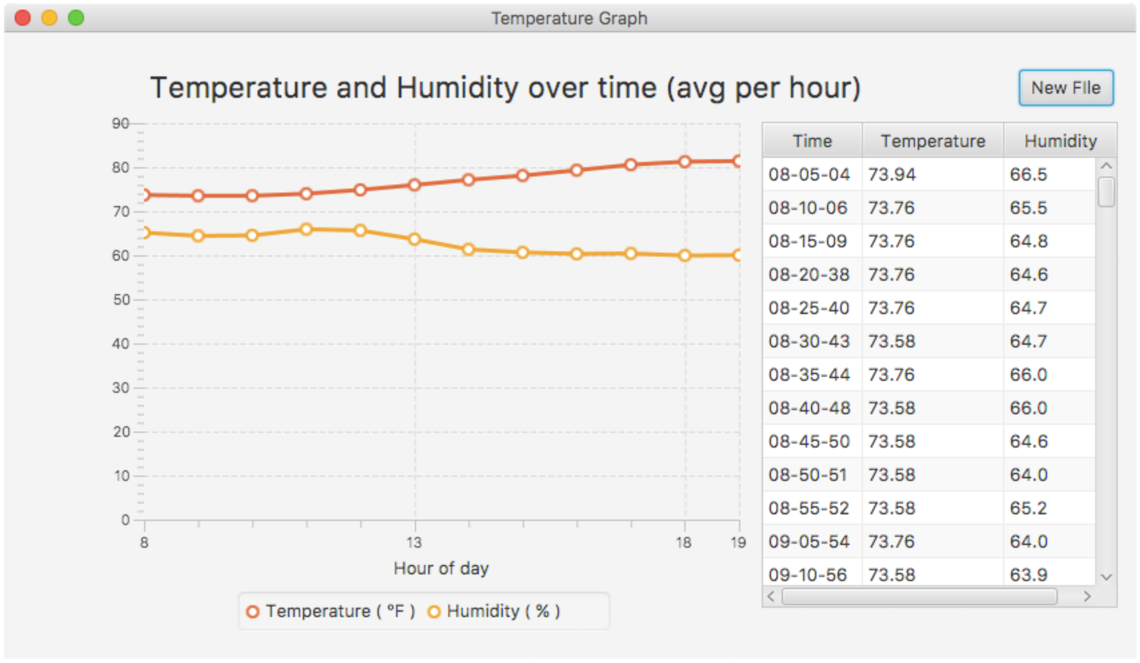
From the main panel, the Analysis menu option can be selected and the Temperature option can be pressed. A file chooser is displayed, which allows a user to upload their CSV file into BeeCon. Once the file has been selected, BeeCon will open the analysis panel.



Temperature and Humidity over time (avg per hour)

Time	Temperature	Humidity
08-05-04	73.94	66.5
08-10-06	73.76	65.5
08-15-09	73.76	64.8
08-20-38	73.76	64.6
08-25-40	73.76	64.7
08-30-43	73.58	64.7
08-35-44	73.76	66.0
08-40-48	73.58	66.0
08-45-50	73.58	64.6
08-50-51	73.58	64.0
08-55-52	73.58	65.2
09-05-54	73.76	64.0
09-10-56	73.58	63.9

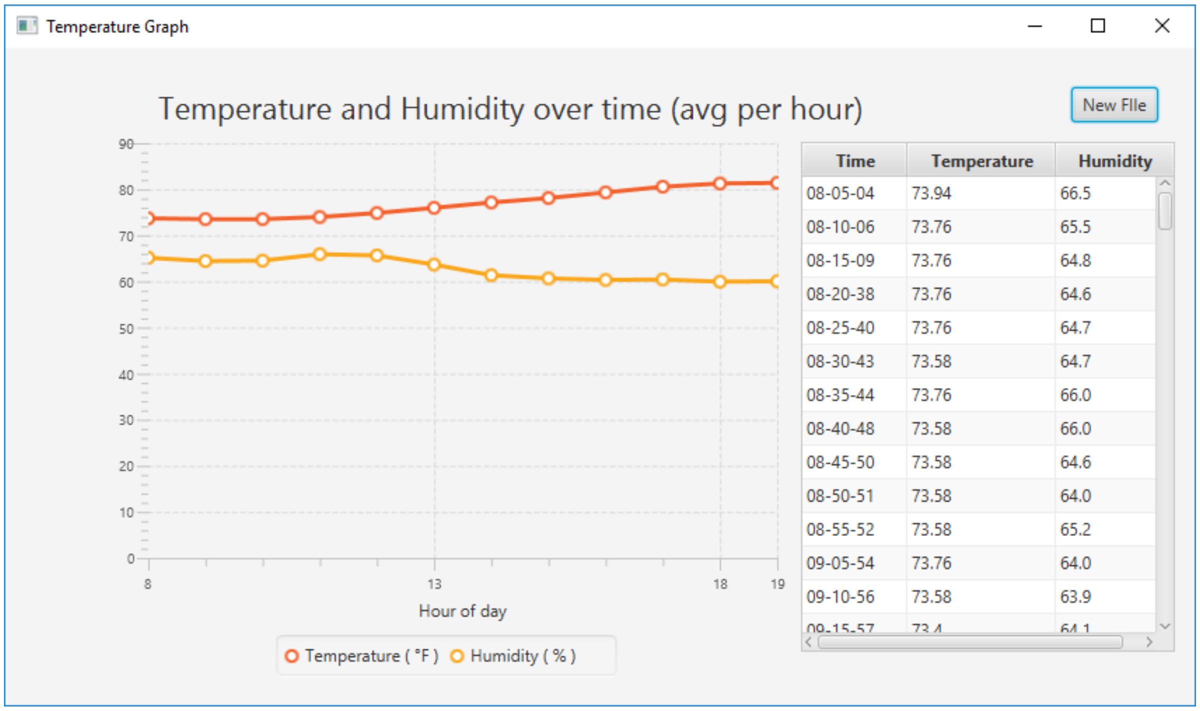
Figure 5.10: Temperature and humidity analysis panel on Linux



Temperature and Humidity over time (avg per hour)

Time	Temperature	Humidity
08-05-04	73.94	66.5
08-10-06	73.76	65.5
08-15-09	73.76	64.8
08-20-38	73.76	64.6
08-25-40	73.76	64.7
08-30-43	73.58	64.7
08-35-44	73.76	66.0
08-40-48	73.58	66.0
08-45-50	73.58	64.6
08-50-51	73.58	64.0
08-55-52	73.58	65.2
09-05-54	73.76	64.0
09-10-56	73.58	63.9

Figure 5.11: Temperature and humidity analysis panel on Mac



Time	Temperature	Humidity
08-05-04	73.94	66.5
08-10-06	73.76	65.5
08-15-09	73.76	64.8
08-20-38	73.76	64.6
08-25-40	73.76	64.7
08-30-43	73.58	64.7
08-35-44	73.76	66.0
08-40-48	73.58	66.0
08-45-50	73.58	64.6
08-50-51	73.58	64.0
08-55-52	73.58	65.2
09-05-54	73.76	64.0
09-10-56	73.58	63.9
09-15-57	73.4	64.1

Figure 5.12: Temperature and humidity analysis panel on Windows

Figures 5.10, 5.11, and 5.12 show an example of this panel. On the left-hand side, a line graph of the average temperature and humidity per hour is displayed. On the right-hand side, a table of every reading in the CSV file is displayed for user analysis. The headings can be clicked to sort the data from ascending to descending. The New File button in the top right corner allows users to open a new CSV file or the panel can be closed.

On the main panel, the File menu item can be selected and the user can select either Disconnect or Quit. Disconnect will log the user out and send them back to the

Login panel to access a different Raspberry Pi. The Quit option will completely exit BeeCon.

Chapter 6 - Conclusion and Future Work

6.1 Conclusions

Monitoring beehives is an essential part of beekeeping. Monitoring beehives gives beekeepers vital information they need in order to determine the health of their beehives. A system that allows beekeepers to monitor their beehives from their homes can allow beekeepers to not disturb their hives as often. The proposed system, BeeCon, provides beekeepers with a GUI called BeeCon that allows them to monitor the Raspberry Pis on their beehives easily on multiple operating systems. Simple temperature and humidity analysis is also provided by the system. As illustrated in Chapter 5 BeeCon can allow beekeepers to analyze their data, monitor their beehives, and control the type of data that is recorded as well as when and how the data is recorded.

6.2 Future Work

In order to provide beekeepers with better analysis of their beehives, audio and video analysis should be researched and included in an advanced edition of BeeCon. Video analysis could alert beekeepers of events that are of interest. These events could include: swarming, being attacked by other bee colonies, and other animals robbing the hive. Audio analysis could also alert beekeepers of important events within the beehive. These events could be tooting, quacking, hissing, or piping [16]. These events can let you know how the hive is doing and if there is a possibility of a swarm.

Real-time video, audio, and temperature data could be implemented in order to give beekeepers a better snapshot of their beehives. Currently users can download the latest video and audio files. This system be expanded to include temperature and humidity as well. A system could also be implemented that streams video, audio, and temperature for constant monitoring of beehives. This would allow beekeepers to stay up-to-date with their beehives throughout the day without disturbing the hives.

Bibliography

[1] ARS: ARS Honey Bee Health and Colony Collapse Disorder. http://www.ars.usda.gov/News/docs.htm?docid=15572.

[2] OCSPP US EPA. Colony Collapse Disorder. https://www.epa.gov/pollinator-protection/colony-collapse-disorder.

[3] Raspberry Pi 3 is out now! Specs, benchmarks & more, February 2016. https://www.raspberrypi.org/magpi/raspberry-pi-3-specs-benchmarks/.

[4] Camera Module - Raspberry Pi Documentation. https://www.raspberrypi.org/documentation/hardware/camera/.

[5] picamera — Picamera 1.13 Documentation. https://picamera.readthedocs.io/en/release-1.13/.

[6] Thomas Liu. AM2302.pdf. https://cdn-shop.adafruit.com/datasheets/Digital+humidity+and+temperature+sensor+AM2302.pdf.

[7] Microphones. eBerry® Plug and Play Home Studio Adjustable USB Desktop Microphone Compatible w/ PC and Mac,ideal for Chatting, Skype, MSN, Yahoo Recording (Black), January 2016.

[8] M. Busse, W. Schwerdtner, R. Siebert, A. Doernberg, A. Kuntosch, B. König, and W. Bokelmann. Analysis of Animal Monitoring Technologies in Germany from an Innovation System Perspective. *Agricultural Systems*, 138:55–65, September 2015.

[9] Tim P. Lynch, Rachael Alderman, and Alistair J. Hobday. A High-resolution Panorama Camera System for Monitoring Colony-wide Seabird Nesting Behaviour. *Methods in Ecology & Evolution*, 6(5):491–499, May 2015.

[10] A. Kumar and G. P. Hancke. A Zigbee-Based Animal Health Monitoring System. *IEEE Sensors Journal*, 15(1):610–617, January 2015.

[11] Robert Szewczyk, Eric Osterweil, Joseph Polastre, Michael Hamilton, Alan Mainwaring, and Deborah Estrin. Habitat Monitoring with Sensor Networks. *Communications of the ACM*, 47(6):34–40, 2004.

[12] C. Yaashuwanth, C. Gopinath, and P. Prabhavathy. Health Monitoring and Management System for Dairy Animals. *Indian Journal of Animal Research*, 48(6):625–627, December 2014.

[13] András Zlinszky, Hermann Heilmeier, Heiko Balzter, Bálint Czúcz, and Norbert Pfeifer. Remote Sensing and GIS for Habitat Quality Monitoring: New Approaches and Future Research. *Remote Sensing*, 7(6):7987–7994, June 2015.

[14] Alberto Cerpa, Jeremy Elson, Deborah Estrin, and Lewis Girod. Habitat Monitoring: Application Driver for Wireless Communications Technology. In *Proceedings of the 2001 ACM SIGCOMM Workshop on Data Communications in Latin America and the Caribbean*, Kota Kinabalu, Malaysia, 2001.

[15] Joseph Robert Polastre. *Design and Implementation of Wireless Sensor Networks for Habitat Monitoring*. Master's thesis, Citeseer, 2003.

[16] Amro Qandour, Iftekhar Ahmad, Daryoush Habibi, and Mark Leppard. Remote Beehive Monitoring Using Acoustic Signals. *Acoustics Australia*, 42, December 2014.

[17] David Kale. *Automated Beehive Surveillance Using Computer Vision*. Master's thesis, Appalachian State University, August 2015.

[18] Ahmad Ghadiri. *Implementation of an Automated Image Processing System for Observing the Activities of Honey Bees*. Master's thesis, Appalachian State University, August 2013.

[19] Alan Mainwaring, David Culler, Joseph Polastre, Robert Szewczyk, and John Anderson. Wireless Sensor Networks for Habitat Monitoring. In *Proceedings of the 1st ACM International Workshop on Wireless Sensor Networks and Applications*, WSNA '02, pages 88–97, New York, NY, USA, 2002. ACM.

[20] Robert Szewczyk, Alan Mainwaring, Joseph Polastre, John Anderson, and David Culler. An Analysis of a Large Scale Habitat Monitoring Application. In *Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 214–226, Baltimore, MD, USA, 2004. ACM.

[21] G. YingMing and J. RenCheng. A Novel Wireless Sensor Networks Platform for Habitat Surveillance. In *Proceedings of the 2008 International Conference on Computer Science and Software Engineering*, volume 4, pages 1028–1031, Wuhan, China, December 2008.

[22] Joseph Polastre, Robert Szewczyk, Alan Mainwaring, David Culler, and John Anderson. Analysis of Wireless Sensor Networks for Habitat Monitoring. In *Wireless sensor networks*, pages 399–423. Springer, 2004.

[23] Venkatesh Rajendran, Katia Obraczka, and Jose Joaquin Garcia-Luna-Aceves. Energy-efficient, Collision-free Medium Access Control for Wireless Sensor Networks. *Wireless Networks*, 12(1):63–78, 2006.

[24] Hanbiao Wang, Deborah Estrin, and Lewis Girod. Preprocessing in a Tiered Sensor Network for Habitat Monitoring. *EURASIP Journal on Advances in Signal Processing*, 2003(4):1–10, 2003.

[25] L. Yong-Min, W. Shu-Ci, and N. Xiao-Hong. The Architecture and Characteristics of Wireless Sensor Network. In *Proceedings of the 13th International Conference on Computer Technology and Development, 2009. ICCTD '09*, volume 1, pages 561–565, Kota Kinabalu, Malaysia, November 2009.

[26] L. Krishnamachari, D. Estrin, and S. Wicker. The Impact of Data Aggregation in Wireless Sensor Networks. In *Proceedings of the 22nd International Conference on Distributed Computing Systems Workshops, 2002. Proceedings*, pages 575–578, Vienna, Austria, 2002.

[27] Sheikh Ferdoush and Xinrong Li. Wireless Sensor Network System Design Using Raspberry Pi and Arduino for Environmental Monitoring Applications. *Procedia Computer Science*, 34:103–110, 2014.

[28] Java Software | Oracle. https://www.oracle.com/java/index.html.

[29] Gradle Build Tool. https://gradle.org/.

[30] 1 JavaFX Overview (Release 8). http://docs.oracle.com/javase/8/javafx/get-started-tutorial/jfx-overview.htmJFXST784.

[31] JavaFX Scene Builder Information. http://www.oracle.com/technetwork/java/java-se/downloads/javafxscenebuilder-info-2157684.html.

[32] Libav documentation : avconv. https://libav.org/avconv.html.

[33] MediaPlayer (JavaFX 2.2). https://docs.oracle.com/javafx/2/api/javafx/scene/media/MediaPlayer.html.

[34] OpenJDK. http://openjdk.java.net/.

Vita

Scott Shuffler was born to Scott Shuffler SR. and Diane Walden in 1990 in the state of North Carolina, United States of America. He attended West Davidson High school in 2008, and then attended Davidson County Community College where he received an Associates of Applied Science Computer Programming in 2012. He was accepted by Appalachian State University in the Fall of 2013 where he earned his Bachelor of Science in Computer Science in the Fall of 2015. He then pursued a Master of Science in Computer Science and graduated in August 2017.