



## Requirements Specifications Version 1.0

**Date:** November 20, 2020

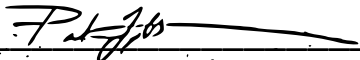
**Team Name:** SongBird

**Project Sponsor:** Paul Flikkema

**Team's Faculty Mentor:** Andrew Abraham

**Team Members:** Kevin Imlay, Daniel Mercado, Yasmin Vega-Nuno, Anqi Wang

Accepted as baseline requirements for the project:

Client:  Date: 20 Nov 2020

Team:  Date: 20 Nov 2020

# Table of Contents

---

<b>Introduction</b>	<b>2</b>
<b>Problem Statement</b>	<b>3</b>
<b>Current Workflow</b>	<b>3</b>
<b>Solution Vision</b>	<b>5</b>
<b>Project Requirements</b>	<b>7</b>
Functional Requirements	7
User Interface	10
<b>Challenges and Risks</b>	<b>15</b>
<b>Project Plan</b>	<b>17</b>
<b>Conclusion</b>	<b>19</b>
<b>Glossary</b>	<b>20</b>
<b>Appendix</b>	<b>22</b>
<b>Works Cited</b>	<b>23</b>

# 1. Introduction

Birds play an essential role in the health and development of many ecosystems around the world. Pest insect populations are regulated by bird predation, dead animals are disposed of by bird scavenging, and many seeds rely on birds for distribution and priming for sprouting. For example, in India, vultures are responsible for an estimated \$34 billion worth of clean-up of cow carcasses [1]. In Sweden, it is estimated that it will cost \$9,400 per hectare for human seed dispersal services if the Eurasian Jay were to disappear from their oak forests [2]. In Jamaica, bird predation on pest insects increases coffee bean production and quality by \$310 per hectare per year [3].

Despite the extreme importance of birds, the scientific community still has numerous questions surrounding the behavior of birds and the ways that they communicate. Questions such as where birds eat and nest, how they move from area to area, and how they communicate with each other remain unanswered. Possibly more importantly, it is not known how these behaviors are changing in response to sound pollution, contact with humans, and climate change.

Many organizations and societies dedicated to studying birds exist around the world. Within the United States, there are over 450 local chapters of the National Audubon Society (an organization dedicated to the study and preservation of birds) [4], and over 180 ornithological institutions and societies around the world [5]. Over the past century, the Bird Banding Laboratory has banded over 77 million birds in North America [6] and uses these bands to track migration, estimate population sizes, estimate survival rates, to name a few [7].

Dr. Flikkema, a professor of Computer Science and Electrical Engineering at Northern Arizona University, is working with a research team at Politecnico di Milano, in Italy, with the aim of helping scientists conduct more meaningful research on birds. This team is creating an Artificial Intelligence tool to classify bird sounds based on audio recordings. This tool will be able to identify bird species, individual birds, and how many birds are present. Currently, however, they are developing this tool using a library of audio recordings that don't include features such as environmental and human-made noise such as vehicles, people, and wind.

## 2. Problem Statement

### 2.1. Current Workflow

We will be discussing the general workflow of recording bird vocalizations using the AudioMoth, the cheaper and most popular alternative of the other audio sensors on the market, in order to gain an understanding of the present problem. To illustrate the workflow, we will be going through the process displayed in Figure 1. First, AudioMoth sensors are placed in the areas of interest, where they will be recording bird vocalizations using their microphone. The AudioMoth's microphone will then capture the audio and store it into its SD card. Once a sufficient amount of audio is collected, the scientist or researcher will use the SD card to download the audio that was collected onto their computer. They will then analyze this audio using one of two methods: annotation, the method of using a human listener to identify the song, or a machine learning algorithm, in which a computer identifies the song. If the researcher wishes to perform another round of data collection and analysis, they must take the AudioMoth sensor, return to the place of interest, deploy the sensor, and then repeat the process.

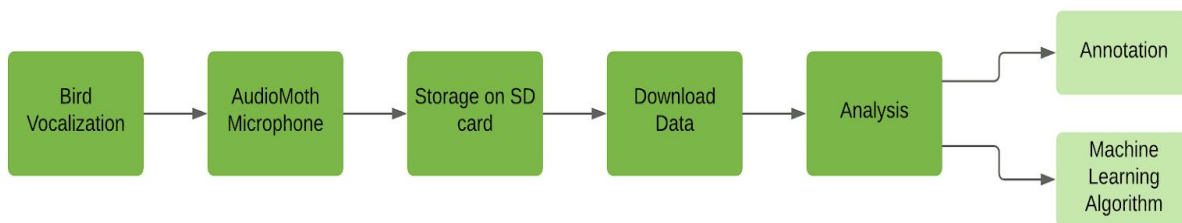


Figure 1. Flow chart of the general process of gathering audio data with an AudioMoth.

### 2.2. The Problem

In the above existing production flow of researching birds lies deficiencies and missing elements important to our sponsor. This approach isn't desirable given the main deficiencies of the current approach:

- AudioMoth sensors as well as other sensors on the market, do not have the extensibility our client is looking for. These sensors do not have the ability to be wirelessly networked for multi-sensor localization and forwarding data, nor do they have the ability to filter out audio that does not contain bird vocalizations.

- The ability to download data is not possible until the sensor is picked up from its location. This is more overhead than if the sensor were able to wirelessly send the data to the desktop application.
- Analysis is also not possible until the sensor is picked up from its location and after the audio files have been downloaded. It would be more convenient for the sensor to improve data collection by discarding data that does not contain bird vocalizations.

What our sponsor ultimately needs is a low-cost, open-source sensor, capable of recording bird vocalizations and analyzing those vocalizations to see if they contain bird calls. The sensor should make the data available to the desktop application for further analysis using MATLAB through wireless communication, serial communication, or by storing the data in the sensor's memory.

### 3. Solution Vision

Our team, Team SongBird, is researching and developing a front-end-back-end sensor system to fit our client’s needs. The back-end will consist of sensors capable of recording bird vocalizations and storing or forwarding these recordings. Working in conjunction with the back-end, the front-end will be a desktop application which is capable of managing these sensors and retrieving recorded data.

Our sensors will be developed on a microcontroller by Silicon Labs that includes two on-board microphones. This board will:

- Record audio in segments.
- Save or stream the segments that contain bird vocalizations.
- Be open-source and modular to allow users to define functionality.

Our desktop application will be compatible with MATLAB, a numerical computing environment, to enable users with powerful analysis tools. This application will provide a user interface that can:

- Connect to and, control settings on, the sensors.
- Download and manage data from the sensors.
- Bring data into MATLAB for advanced analysis.

Below, Figure 2 shows the major steps that our system will take from bird to end user.

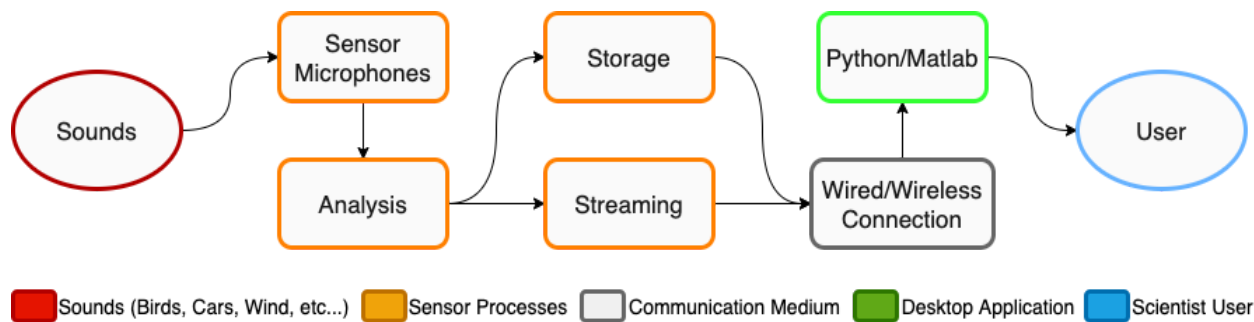


Figure 2. Flow chart showing the major steps of data collection from a bird to the end user.

The system begins with the sensor recording from its microphones to capture a bird vocalization. Once the audio is recorded, a basic analysis will be performed to determine if the audio contains a bird vocalization. Based on this analysis, the recording will be saved if a bird vocalization is present, or discarded if there is no bird vocalization. From here, the board may forward

the recording to the desktop application over a wired or wireless connection. The desktop application will import the data into MATLAB so that the end user is able to perform further advanced analysis of the recorded audio.

This project is mainly research in nature, meaning that we are attempting to test the feasibility of this system. The current version of our system has a crucial issue regarding how little audio can be stored on the sensor (see our Technological Feasibility Report for details). Without the addition of secondary memory to the sensors, the sensors can only hold 25.6 seconds of audio. The limited storage available on the board is the largest risk for our project not succeeding, and is very likely to cause issues with development. While this is an issue for this version of our system, it is not an issue for future versions where sensors will wirelessly forward their data to a server for storage and access.

## 4. Project Requirements

Overall we separated the project requirements into two main categories: a User Interface and an Embedded System. First we will discuss the Functional Requirements related to the User Interface and Embedded System. Then, we will outline the Performance Requirements for each of the categories and address how the project should perform. Lastly, we will discuss the Environmental Requirements which are the constraints and limitations we must abide by with this project. As was discussed in the Solution Vision section, this project is mainly research in nature. As such, it is important to note that goals are integrated into these requirements.

While creating this document we had conversations with our sponsor Dr. Flikkema about the differences between requirements and goals. We concluded that this project is seen as a research project to our sponsor which allowed us to create these requirements and goals. We have a number of goals to see how close we can get to them, however, we are not setting them as requirements because we are not sure if they are possible to achieve without additional research.

### 4.1. Functional Requirements

#### 4.1.1. Sensor

Essentially, the sensor should be able to record audio, analyze audio, store audio, and utilize its power efficiently. We dive into detail about this high level view below. Note that the following should be read as being under the label of “requirement” unless explicitly stated as a “goal” or “stretch goal”. Requirements are necessary to complete a successful product. Goals would greatly improve our product, but are not necessary.



- 4.1.1.1. The sensor will be able to record audio from microphones. Our goal is to include an option to record from one or both microphones.
- 4.1.1.2. The sensor will be able to split audio recordings into configured length segments (approximately 4 seconds). If available storage only allows for less audio than the segment length, then only the length that is available starting from the beginning of segment will be saved.
- 4.1.1.3. The sensor will be able to analyze audio segments and determine if it should be saved or not. The analysis should occur in real time. During the process of recording, one of our stretch goals is for analysis to be improved by using both microphones, by combining streams into one, which can improve signal to noise ratio. If recording from both microphones, streams from both will be saved instead of their combination of streams.
- 4.1.1.4. The sensor will be able to utilize a Fast Fourier Transformation (FFT). The FFT is a mathematical algorithm used to transform data recorded into frequencies that can be analyzed, often used in audio analysis.
- 4.1.1.5. The sensor will be able to do frequency filtering. This will include banding to search between 160 Hz and 600 Hz for example, and removing frequencies outside of the band. The majority of bird calls fall within 160 Hz and 600 Hz.
- 4.1.1.6. The sensor will be able to store audio segments in on-board memory. This includes storing the time and date of the recording segment and in a custom file format. Also, our sensor will look more into the feasibility of a micro-SD card or QSPI FLASH for additional secondary memory as a stretch goal.
- 4.1.1.7. The sensor will be able to send saved audio segment files to the desktop application over the wired or wireless connection. This includes the audio saved and it's associated

metadata. This metadata will contain date, time, sensor ID, and sensor number.

- 4.1.1.8. The sensor will be able to delete saved audio segment files from the on-board memory with the command from the desktop application.
- 4.1.1.9. The sensor will use a communication schema (same as the desktop application) as a set of protocols for standardized communication between the sensor and the desktop application. This communication schema will be high-level and sit above whatever communication medium that is in use. A goal of this communication schema is to incorporate user authentication to prevent illegitimate users from accessing and altering the sensor.
- 4.1.1.10. The sensor will live stream audio segments to the desktop application as they are being recorded. Our goal is for the sensor to perform its analysis and only send segments that are flagged as containing bird vocalizations so as to only send useful data to the desktop application.
- 4.1.1.11. Our goal is for the sensor to provide the user with time intervals of recording. This will let the user set what times of day, and for how long, to record audio. Included in this goal is for the sensor to provide options for the date as well as time for recording, and the sensor should allow for continuous steps for the time intervals.
- 4.1.1.12. The sensor will enter the Sleeping mode for power saving when there is no storage space left to save audio recordings. The sensor will also enter the Sleeping mode while not in the time interval of recording.
- 4.1.1.13. A stretch goal is for the sensor to test the power source for power input conditions and safely shut down when low power is provided, and start up when sufficient power is provided. This helps to prevent data corruption in the event of power loss.

- 4.1.1.14. A stretch goal is for the sensor to incorporate a trigger system that can start recording. These triggers may be audio based with frequency or amplitude, or may be light based. This can let the sensor start recording outside of recording intervals or give the option to not use recording intervals.
- 4.1.1.15. A stretch goal is for the sensor to compress audio for saving in on-board memory. This compression is required to be lossless. Compression can help increase the amount of audio saved on the board.

## 4.1.2. User Interface

Overall, a key requirement is for our user interface to interact with the board. This will be the end cycle for most of the data in exporting, deleting or analyzing the data itself. It will need to be easy to use for all users and allow users to view the data that is stored on the board. In addition, the user interface will need to integrate MATLAB for advanced analysis of the audio data. We break down these ideas into details below. They should be read as being under the label of “requirement” unless explicitly stated as a “goal”. Requirements are necessary to complete a successful product. Goals would greatly improve our product, but are not necessary.

- 4.1.2.1. The user interface will be able to view the data that has been recorded on the sensor. The audio data will be displayed in a list format with details such as File Name, Sensor ID, and Date-Time.
- 4.1.2.2. The user interface will be able to filter the data shown based on sensor ID or time.
- 4.1.2.3. The user interface will be able to stream the data from the sensor when set to “Streaming” mode. This will allow audio to be directly streamed over to the user interface. A later goal will be to have only analyzed data streamed to the user interface.

- 4.1.2.4. The user interface will be able to export the audio files. This will allow the user to export any of the files stored on the sensor to the user's choice of destination. This will be done using the default file explorer.
- 4.1.2.5. The user interface will be able to display a spectrogram of the audio data it is viewing. This is so the user can see which parts of the audio the sensor qualified as "interesting" enough to save.
- 4.1.2.6. The user interface will be able to download audio files from the sensor. This will be done by allowing the user to select specific audio files to download with an option to download all audio files.
- 4.1.2.7. The user interface will be able to delete audio files that are stored on the sensor. This will be done by allowing the user to select specific audio files to delete with an option to delete all audio files.
- 4.1.2.8. The user interface will be able to review the status of the sensor. This will allow the user insight into the current behavior of the sensor. Statuses of the sensor include Streaming, Idle, Recording, and Sleeping.
- 4.1.2.9. The user interface will have configurable settings for the sensor that include the ability to set the recording intervals and the ability to change the status of the sensor.
- 4.1.2.10. The user interface will use a communication schema (same as the sensor) as a set of protocols for standardized communication between the sensor and the desktop application. This communication schema will be high-level and sit above whatever communication medium that is in use. This communication schema should incorporate user authentication to prevent illegitimate users from accessing and altering the sensor through the desktop application.

## 4.2. Performance Requirements

### 4.2.1. Sensor

Below we describe the performative requirements regarding the sensor. The following should be read as being under the label of “goals” unless explicitly referred to as a “requirement”. The sensor will be able to function as expected and be efficient. We dive into the details below.

- 4.2.1.1. The sensor’s code should use a small footprint to leave space for audio storage and additional code modules.
- 4.2.1.2. The sensor should be able to record audio with a significant signal-to-noise ratio within a cleared 30 foot radius.
- 4.2.1.3. The sensor should record audio between 0 kHz and 20 kHz.
- 4.2.1.4. The sensor should be able to store at least 15 seconds of audio recording on the FLASH storage native to the sensor board.

### 4.2.2. User Interface

The user interface will be easy to use for both our client and other scientists that will have the opportunity to use it. This is essential in both a design aspect and the speed at which tasks are done in the user interface such as modifying files or downloading files. This will be further expanded on in the points labeled below and should be read as being under the label of “goals” unless explicitly referred to as a “requirement”. We came up with these requirements and goals with the help of Dr.Flikkema and additional research as a group on how long these tasks would normally take.

- 4.2.2.1. The user interface will be able to view data on the sensor within 4 mouse clicks of the user.
- 4.2.2.2. The user interface will be able to stream audio from the sensor with less than a 500 millisecond delay.
- 4.2.2.3. The user interface will be able to export the audio within 5 mouse clicks of the user.

- 4.2.2.4. The user interface will be able to download all audio files from the sensor within 20 seconds.
- 4.2.2.5. The user interface will be able to delete all audio files from the sensor within 20 seconds.
- 4.2.2.6. The user interface will be able to change the configuration of the board in 30 seconds.
- 4.2.2.7. A first-time user will be able to download all audio files from the sensor in 5 mouse clicks.
- 4.2.2.8. An experienced user will be able to download all audio files from the sensor in 3 mouse clicks.
- 4.2.2.9. A first-time user will be able to navigate to the option of downloading selected files in 7 mouse clicks.
- 4.2.2.10. An experienced user will be able to navigate to the option of downloading selected files in 5 mouse clicks.
- 4.2.2.11. A first-time user will be able to delete all audio files from the sensor in 5 mouse clicks.
- 4.2.2.12. An experienced user will be able to delete all audio files from the sensor in 3 mouse clicks.
- 4.2.2.13. A first-time user will be able to navigate to the option of deleting selected files in 7 mouse clicks.
- 4.2.2.14. An experienced user will be able to navigate to the option of deleting selected files in 5 mouse clicks.

### 4.2.3. Both Sensor and User Interface

Both the sensor and the user interface share performative requirements. The following should be read under the label of “requirements” unless explicitly referred to as a “goal”. Below we discuss in detail the ideas that center around ease of use for the user, extensibility, and shared communication.

- 4.2.3.1. The communication schema used between the board and the user interface will be extensible through modularity.

- 4.2.3.2. The code for the sensor and desktop application will be open source and documented to provide users with a complete understanding of the system and the ability to add and modify modules.
- 4.2.3.3. The code for the sensor and desktop application will be modular to allow the user to add and switch modules.
- 4.2.3.4. The sensor and board will be required to communicate through a wired USB connection, and should be able to communicate over a wireless bluetooth connection as a goal.

### 4.3. Environmental Requirements

Environmental requirements are divided into two categories below: Sensor and User Interface. These are constraints that are imposed on this project. As such, they should be read as being under the label of “requirements” instead of goals. Below we dive into the specifics of what equipment we are to use and other restrictions.

#### 4.3.1. Sensor

- 4.3.1.1. The Thunderboard EFM32GG12 is required as the sensor of choice for this project.
- 4.3.1.2. The accompanying Simplicity Studio IDE is required to code in.

#### 4.3.2. User Interface

- 4.3.2.1. The user interface should support MATLAB functionality.

## 5. Challenges and Risks

Some of the major challenges we can see right now are working within the limited storage, memory, and processing power of the sensor board.

Without any modifications to the sensor board, only about 25 seconds of audio can be saved. In practice, this value will likely be less than 20 seconds of audio storage. The likelihood of this being a problem is high, but it would not have a major impact on the project as a whole. Dr. Flikkema plans to further develop this sensor system to forward all the data collected to a server, so the limited amount of storage for audio will eventually not be needed causing the severity to be low.

The limited memory on the board will be a constant challenge to work with. The sensor board only has 196 kb of RAM, so we will have to design the board to use very little memory for operations. The likelihood of this being a problem is high, and the impacts can be major. If the memory is too limited for our requirements, the whole project can fail.

Similarly, the limited processing power on the board will be a constant challenge. While the board's processor is fairly powerful, it is still nowhere as powerful as a desktop computer. The largest issue with the processing power is whether or not it will be able to perform the audio analyses required to mark audio as containing bird vocalizations. If the processor is unable to perform where it needs to, the impacts can be major on the success of this project. We think there is a medium likelihood of this happening.

Aside from the challenges of our hardware, there are the risks of competitors and of the performance ability of our sensors.

At the time there is one competitor that has developed a device similar to what we want to develop, they are known as AudioMoth. The risk of another competitor coming into the market is moderate and the severity is low since we are making this device open source for others to improve/develop new solutions.



Our sensors will be challenged with recording bird vocalizations that overlap with many other sounds. Ideally, the sensors would always be able to determine if some audio contains bird vocalizations, but many factors can diminish the sensor's ability to do so. Some of these factors are weather like wind and loud man-made sounds. We believe the likelihood of these causing an issue to be medium, but we will constantly be thinking about them because our audio analysis is likely the largest aspect of this project. Audio analysis should be able to diminish this risk. Therefore, the severity should be low.

Some of the other risks involved in this project are the Thunderboard going out of production, Matlab going out of production, and our programs only being able to work on the Thunderboard. These risks are seen as low for the project we are developing, but the severity could be high considering that our project relies on these components to function.

## **6. Project Plan**

Preparing a plan for the future of this project is crucial to overcome potential risks and challenges. This is why we created milestones and phases to ensure that we stay on track. We have separated our project into twelve milestones corresponding to our requirements and three phases which track our progress in “releases”. This will allow us to be able to test the software and fix any bugs after phases so they do not accumulate.

### **6.1. Tech Demo**

Our goal for this phase is to prove that our project is possible with the current technologies that we have. Each milestone for the Tech Demo phase is listed in the light blue section of Gantt Chart 1 located in the Appendix. This release is relatively short compared to the Post Demo and Final Product phases simply because we have less time to create the Tech Demo. It will last from November 6th to November 20th which is only three weeks with each of the milestones lasting two weeks. We would like to ensure that the project is achievable within this phase. This phase is one of the most important since it confirms that the technologies we chose can interact with each other smoothly and allow a usable workflow. This is why we have milestones such as User Interface Communication with Board and Storage Capabilities to prove that our Thunderboard can communicate well with the user interface. After we show that it is possible to use these different technologies together, we will then go into the Post Tech Demo where we will start using these technologies for the project itself.

### **6.2. Post Tech Demo**

During this phase we will want to achieve a usable product from our research on the previous phase. Each milestone for the Post Tech Demo phase is listed in the light red section of Gantt Chart 1 located in the Appendix. This maps out some of the work we will be doing over winter break continuing into the

spring semester which lasts a total of ten weeks with the longest milestones lasting only five weeks. This phase we will be developing some of the core milestones such as User Interface Communication protocols and basic Audio Analysis on the board itself, both are predicted to take five weeks. Consequently, this will help us develop some of the more complicated features in time. Continuing this phase will be the Final Product phase that will contain most of our more advanced features.

### **6.3. Final Product**

In the Final Product phase, we will develop some of the more advanced features as well as the stretch goals that we have in place. Each milestone for the Final Product phase is listed in the purple section of Gantt Chart 1 located in the Appendix. This maps out the work we will be doing in the final half of the Spring semester, around nine weeks, which primarily consists of developing advanced features such as Adjustable Analysis and Streaming Mode which both are predicted to take about five weeks. Overall, this is the final sprint to get bugs fixed and a product that is not only usable, but has features that make it easy for a user to personalize how they would like to use the product.

## 7. Conclusion

Birds play a very important role in ecosystems around the world. Without birds, many plants and animals would not survive and ecosystems would fall apart. Consequently, it is imperative to learn more about the behaviors of birds and how human interactions are influencing them.

Currently, scientists use audio recording devices, such as the AudioMoth, to track what birds are where. The sensors can be expensive and cumbersome, or they are cheap and inflexible. We are designing BiVo to solve this issue. Our team is researching and developing a front-end-back-end sensor system to fit our client's needs. BiVo will be cheap and flexible to allow the user much more control over what the sensor does than other similar sensors.

Overall, regarding the iteration of requirements, we separated the project requirements into two main categories: the User Interface category and the Sensor category. We discussed the Functional Requirements related to the User Interface and Sensor and outlined the Performance Requirements for each of the categories by addressing how the project should perform. We also discussed some limitations of Environmental Requirements.

Our project comes with a few risks that can impact its production and final result. Following, we have included the recent challenges, like the storage of sensors and the processing capacity of sensors, and some potential risks, like the competitor and the environment, we see that may impact our progress.

Preparing a plan for the future of this project is crucial to avoid potential risks and challenges. Table 1 in the Appendix displays our schedule. We have separated our project into twelve milestones corresponding to our requirements and three phases which track our progress in "releases". This will allow us to be able to test out the software and fix any bugs efficiently after phases.

When complete, BiVo will be the foundation for an improved bird vocalization sensor system for scientists. Dr. Flikkema and his research team will be able to more efficiently gather better data for their bird-vocalization-identification tool, creating a better tool. With BiVo, and this tool, scientists around the world will be able to conduct more meaningful research on birds and just how important they are.

## 8. Glossary

**Amplitude:** in acoustics, amplitude is the distance of the peak or trough from the vertical center of the waveform. This can be perceived as the loudness of a sound.

**Annotation:** the method of using a human listener to identify bird songs.

**Artificial Intelligence:** a class of computer algorithms that can be trained to perform complex tasks. In general, the more training an artificial intelligence receives, the better it will perform at its task.

**AudioMoth:** a low-cost, open-source acoustic monitoring device.

**FLASH Memory:** a form of solid state memory that is non-volatile, or doesn't lose its data when power is lost.

**Frequency:** in acoustics, frequency is the number of peaks or troughs that pass a given point in a given set of time. This can be perceived as the pitch of a sound.

**Hertz:** a measure of frequency in a waveform, abbreviated as Hz. Kilohertz, or kHz, is a unit of base 1000 hertz.

**Lossless Compression:** in data compression, lossless compression is a form of compression that, when uncompressed, is identical to the original data before compression.

**Machine Learning Algorithm:** an algorithm in which the computer learns from data and improves with experience without human intervention.

**MATLAB:** a proprietary numerical computing environment and programming language used by many professionals and researchers to perform complex and powerful computations.

**Metadata:** information about a particular set of data. This could be time and date, file name, file location, and all other data besides the data being referenced.

**Microcontroller:** a specialized chip that contains all the components of a simple computer. Microcontrollers are sometimes abbreviated as MCU, for microcontroller unit.

**Modular:** in software, modular code is code that is able to be switched, altered, removed, or added to change or add functionality without the need to heavily alter unrelated code.

**Open Source:** software whose source code is publicly available and can be redistributed and modified.

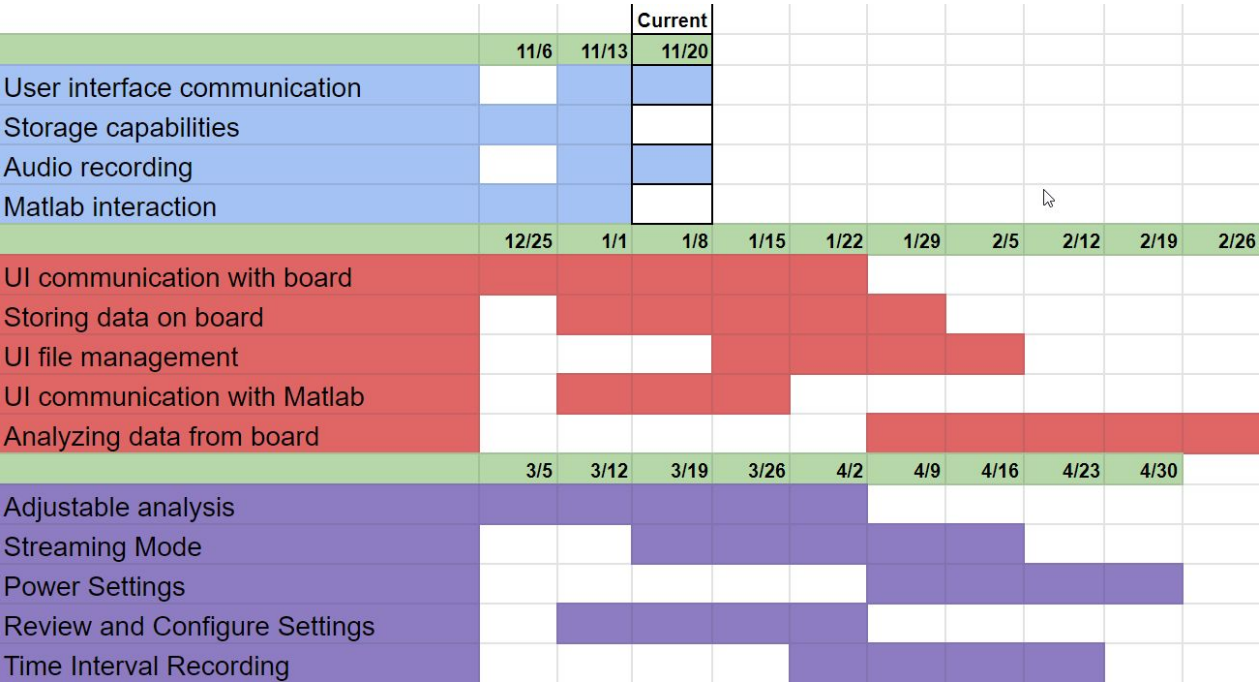
**Ornithological:** a branch of zoology that is concerned with the study and knowledge of birds.

**Protocol:** in software, a protocol is a set of procedures that are performed to accomplish a goal, usually associated in communication between two or more devices.

**Schema:** a formalized model or theory.

**Workflow:** the sequence of tasks in a process from start to finish.

# 9. Appendix



Gantt Chart 1. Gantt Chart explaining our timeline for completing and researching the project. Blue means Tech Demo phase, red means mid development stage, and purple is final development. The current date is shown with a black outline and the text above.

## 10. Works Cited

1. Peisley, R. K., Saunders, M. E., Robinson, W. A., & Luck, G. W. (2017). The role of avian scavengers in the breakdown of carcasses in pastoral landscapes. *Emu-Austral Ornithology*, 117(1), 68-77.
2. Hougner, C., Colding, J., & Söderqvist, T. (2006). Economic valuation of a seed dispersal service in the Stockholm National Urban Park, Sweden. *Ecological economics*, 59(3), 364-374.
3. Johnson, M. D., Kellermann, J. L., & Stercho, A. M. (2010). Pest reduction services by birds in shade and sun coffee in Jamaica. *Animal conservation*, 13(2), 140-147.
4. National Audubon Society - Audubon Near You, Accessed October 31, 2020, <https://www.audubon.org/about/audubon-near-you>.
5. International Ornithologists Union - Ornithological Societies and Institutions, Accessed October 31, 2020, <https://www.internationalornithology.org/other-ornithological-societies>.
6. USGS - Bird Banding Laboratory, Accessed November 3, 2020, [https://www.usgs.gov/centers/pwrc/science/bird-banding-laboratory?qt-science\\_center\\_objects=0#qt-science\\_center\\_objects](https://www.usgs.gov/centers/pwrc/science/bird-banding-laboratory?qt-science_center_objects=0#qt-science_center_objects).
7. USGS - Why Do We Band Birds?, Accessed November 3, 2020, [https://www.usgs.gov/centers/pwrc/science/why-do-we-band-birds?qt-science\\_center\\_objects=0#qt-science\\_center\\_objects](https://www.usgs.gov/centers/pwrc/science/why-do-we-band-birds?qt-science_center_objects=0#qt-science_center_objects).