Bird Strike Monitor User Guide

as of 2/22/2022

For any information not listed, contact either:

gragleas@iu.edu

wyemicha@iu.edu

1 Parts Needed

To use the bird strike monitor, you will need a few pieces of hardware and some tools on your computer to use them.

Hardware

- 2 x Raspberry Pi 3 B+ (other versions of Pi may work with troubleshooting, compatibility issues likely)
- 2 x 5V 2.5A Raspberry Pi 3 B+ Power Supply/Adapter
- 2 x Large (32GB) MicroSD Card
- 1 x USB MicroSD Adapter
- 1 x Seeed ReSpeaker 2-Mics Pi Hat
- 1 x Adafruit TSL2591 High Dynamic Range Digital Light Sensor – STEMMA QT
- 1 x Adafruit MPU-6050 6-DoF Accelerometer and Gyro Sensor – STEMMA QT Qwiic or Equivalent (ours)
- 1 x Grove 4 Pin to Male Jumper Cable
- +4 x Male to Male Jumper Wires
- 1 x Small Breadboard

Tools Required

- GitHub (how we will push updated software to the Raspberry Pi's and how you will pull the updated software to your Pi's)
    - Will need to create an account if you don't have one and contact us to be added as collaborators to the repository
    - GitHub Desktop provides a user-friendly GUI
    - Can install git and use via command line/terminal, quick and few commands but hard to troubleshoot and can't see much of what's going on
- Command Line Access
    - Linux – Can open terminal from apps
    - Mac OS – CMD + Space brings up search bar, search "terminal" or "command prompt" to bring up the terminal
    - Windows – Install PuTTY (64-Bit x86 version most likely) and run the downloaded PuTTY.exe file or search for PuTTY in the lower left corner

2 Using the Pi’s

Initial Setup (Fresh Raspberry Pi’s)

The Raspberry Pi board needs an operating system in order to actually run the programs, which must be downloaded from a computer and move onto a MicroSD card. For this step, you will need to be able to connect a MicroSD card to your computer (likely with a USB Adapter).

1. Insert the blank MicroSD card into the USB MicroSD Adapter, and insert the USB Adapter into your computer.
2. Navigate to https://www.raspberrypi.com/software/ and download the Raspberry Pi Imager for your computer version (Windows/Mac/Linux). Install it with the .exe file when it is finished downloading.
3. Open up the Raspberry Pi Imager with your SD card inserted into your computer, and select the Raspberry Pi OS/Raspbian. Afterwards, choose your storage and select the MicroSD card. Press CTRL + SHIFT + X to open up the Advanced Options Panel.
4. Here you can set several important settings for the Pi including your **hostname** which is used to connect to the Pi via **ssh**, actually enabling **ssh** and setting your **password**, and configuring your **WiFi** settings for your local network. Create a **hostname** and **password** and write them down somewhere. Enter the **WiFi** settings for the network you will be using. Click save.

5. Click Write and when it is finished it will tell you that you can now remove the SD card from the reader. Remove the SD card from the reader and insert the MicroSD card into your Raspberry Pi. You are now ready to connect to the Pi.

To connect to the Pi's and start monitoring, or to edit files as needed, you will need to connect to them over ssh (Secure Shell Protocol). This will allow you to access the file contents of the Pi's from anywhere as long as they are plugged in.

Getting Connected

- Linux/Mac OS – Once you have a terminal open, type the following command and press enter:

  ssh pi@[hostname].local

  - Enter the password that you set earlier when prompted for it.
  - After connecting, you may be prompted if you want to continue connecting, type in either "yes" or "y" and press enter depending on what you are asked

- Windows – Once you have PuTTY open, in the "Host Name (or IP address) field, type in one of the following commands and click Open:

  ssh pi@[hostname].local

- Enter the password that you set earlier when prompted for it.
- After connecting, you may be prompted if you want to continue connecting, type in either "yes" or "y" and press enter depending on what you are asked

For all users: After connecting via one of the ssh commands you should see a screen with a line that says something like:

pi@[hostname]:~ $

Helpful Commands

Everything we do on the Pi is controlled from commands on the command line, we typically only use the same handful of commands regularly, but it may take some getting used to as you won't be using a mouse to navigate things.

sudo

sudo is a keyword that you may sometimes add before a command which essentially just runs that command as an administrator which bypasses some annoying permissions requirements. Anytime this happens you may be prompted to enter your password, do so and press enter.

Tabbing

Pressing **Tab** will autocomplete the word you are typing if there is a match for it. This is helpful because a lot of the files are named with a combination of words and numbers connected by dashes or underscores (which are treated as entirely 1 word). This works for command names, data files, directories, and so on.

For example, if the directory I was in had a file called scipy_1.0.0_version_4.tar, I could begin typing "sci" and press **Tab**, and it would automatically fill in the rest. If there was another file in the directory with a similar name like scipy_1.0.0_example.tar, pressing **Tab** would automatically fill in the parts that are shared, i.e., "scipy_1.0.0_", and you would specify the rest. You could additionally just type in the next letter and press **Tab** again, or whatever differentiates it from the other files.

- **cd** – Change Directory
  - This command will change the directory (folder) you are currently looking at, by default the Pi's will be in the /home/pi directory, and inside that directory is the directory Engr-2022-Capstone. This is the folder for the GitHub repository where all the code and data are located.
  - For example, if you wanted to change directories from /home/pi to /home/pi/Engr-2022-Capstone, you would run the command: cd Engr-2022-Capstone while within the /home/pi directory.

  - Running the command “cd” at any time will bring you back to the /home/pi directory.
  - Running the command “cd ..” will bring you up one directory from where you currently are.
- **ls** – List
  - This command lists all of the contents of the current directory, color coded by type (may differ across systems).
  - This command will be used very frequently with **cd**, as you will want to know what is in a directory when you change to it.
- **clear**
  - This command will simply clean up the screen by clearing all text currently on the terminal. Sometimes there is just simply too much information to look at and looking at one thing at a time is helpful.
- **python3** [optional filename here]
  - Running the command “python3” followed by a space and then a python filename (which will end in “.py”), will execute the given file until it is completed, or it is canceled by the user.
    - Sometimes, a specific version must be specified, so instead of just running “python3 filename.py”, you’ll need to run “/usr/bin/python3 filename.py”
    - To kill a currently running python program, press Ctrl + C
    - To pause a currently running python program, press Ctrl + Z

        - Pausing/Killing a program may not always work correctly and cause an error the next time you try to run the program, which you can fix by simply killing the process with the command "**kill %**"
    - For most python programs we will be working with, the terminal will show you the output of the Pi's as they are running, this is a good way to tell whether things are working or not and is useful when testing or bug fixing.
    - Running the command "python3" by itself will bring you to a python3 interpreter, which will allow you to run and test python commands in real time, which may be useful for bug fixing or math operations.
- **vim** [optional filename]
    - vim is a text editor that we use to edit files. Ideally you shouldn't need to edit anything, but there are several parameters you might like to tune for testing such as microphone or accelerometer sensitivity or video clip length.
    - You can edit a file by running the command **vim filename.xxx**, which will open up the contents of the file.
    - The controls for vim are **NOT** intuitive or user-friendly, and you will likely have to search up for commands you are looking for. To make things easy, try to only do the following if you must use vim:
        - When the file opens you will see your cursor blinking but will be unable to type. To begin editing you must first enter "Insert" mode by pressing the "i" key.

- You may navigate throughout the file with the arrow keys and type as usual, when you are done typing, press the Escape key to exit "Insert" mode. Then, press the ":" key to bring up a prompt at the bottom of the screen. If you wish to save changes and exit, type in "wq" and press enter. If you wish to exit without saving, type in "q!" and press enter.
- Once you've made any changes and exited the file, if it was a python program you may rerun it to test the changes you've made.
- If you fear that you've broken the code, no worries, you can use the command **git pull** to get the most recent stable version of the repository from GitHub. **Please do not push anything to the master branch, if you've made a change that you'd like to keep, let us know and we will change the master branch.**

If you are experiencing an error or find yourself needing commands not listed here, please feel free to contact us. There are some very specific commands that may be used later, but these are the most commonly used ones.

Will I break everything if I type in the wrong command?

The commands above are safe, and there is always a backup of the code on GitHub so there is little danger of things seriously going wrong. Things **may** go wrong if packages are installed incorrectly or removed. To avoid this, please do not use any **pip** commands or **sudo apt install** commands.

Clone Repository/Getting Required Files Configured

After connecting to either Pi, you will need to run some scripts to get the code setup on the Pi. To get the required files on the Pi, you will need to **git clone** the repository from GitHub, for which you will need to install **git** with the command **apt-get install git**. Once it is finished installing, make sure you are in your home directory by running the command **cd**. Now run the command **git clone** https://github.com/wyethmich/Engr-2022-Capstone.git. Now enter the new folder with the command **cd Engr-2022-Capstone** and run the following 2 commands: **sh setup.sh** and **sh setup2.sh**. These will automatically download, install, and configure the necessary files to run the system. Now we need to set up a static IP address and we will be good to go.

Static IP Setup

We need a static IP address so that our Pi's have the same IP address at all times so that they know how to communicate with each other. To do this we will need to edit a file with vim. Run the command **sudo vim /etc/dhcpdc.conf** which will open up the file to be edited. Scroll down all the way to the bottom of the file and add the following lines:

`interface wlan0`

`static ip_address=192.168.0.200/24` < Whatever you want it to be (in this format)

`static routers=192.168.0.1` < (Obtained by running the command **ip r | grep default** this will be the first IP address given)

`static domain_name_servers=192.168.0.1` < (Obtained by opening a file with the command **sudo vim resolv.conf** and it's the IP address immediately after **nameserver**)

Save and quit the file and then run the command **sudo reboot**. Wait a minute or so and then connect back to the Pi using the command **ssh pi@[hostname].local** and enter your password when prompted. Now if you run the command **ifconfig**, you should be able to see that static IP address that you set under the network interface that you specified above (likely wlan0).

3 Starting the Bird Monitoring

To get the system working, you will need to start a python program on **both** Pi's so that they are running at the same time. The order in which you start them shouldn't matter. For this you must open **two** different terminals, one for each Pi.

Sensor Pi

To start the sensor monitoring, you must run the command **python3 superscript.py** located in the directory **/home/pi/Engr-2022-Capstone**. This may take a moment to start up, and this file handles the microphone listening, accelerometer detection, light sensor, and machine learning classification. The accelerometer is tuned so that it compares reading against the average of the last hour of data, so the detection will become more accurate the longer the program is left running. When a sound above a certain threshold **and** a vibration above a certain threshold are detected at the same time, the audio is processed in a machine learning classifier, and if the audio sample is determined to be a positive, the audio is saved, the light level is logged, and a request is sent to the camera to save video.

Camera Pi

To start the camera monitoring, you must run the command **/usr/bin/python3 camerascript.py** which is located in the **/home/pi** directory. This may take a moment to start up, once it does it will wait until it has received a request from the sensor Pi and record a video including a period of time before the window strike, and a period of time afterwards. The video does take 30-40 seconds to be processed after it's been recorded, so during this time it will be unable to receive any more requests.

Important Note for Camera Pi

Please do not close the terminal or kill the python program with Ctrl + C/Z if there is a video still being processed. In the terminal while the python program is running, you'll be able to see whether a video request has been triggered or not. If no request has been made or if the video processing has completed and you see a line that says "moviepy has completed" or "saved the file to filename.mp4", you can kill the program. If you kill the program before it is finished processing, excess files may be left in the home directory named after.xxx or before.xxx. To get rid of these, run the command **sh cleanup.sh**.

Locating the Data

For right now, the data recorded is located on the Pi's SD cards, which can then manually be moved to another location. In the future we'd like to have the data be saved directly to a server somewhere, but for right now it is all local.

- **Sensor Pi Data**
    - **Time Data**
        - All audio/video files will have the date and time in the name of the file, for example an audio file from 2/22/2022 at 5:47PM might be named 02_22_05_47_30_PM.wav.

  - **Audio Data**
    - Located at /home/pi/Engr-2022-Capstone/audioconfig
  - **Light Data**
    - Located at /home/pi/Engr-2022-Capstone/logs
- **Camera Pi Data**
  - **Video Data**
    - Located at /home/pi/videos

4 Training the Classifier

Currently, the last check before data is recorded to disk is an SVM classifier. This takes in the audio file and uses a trained machine learning model to predict a simple True/False, is it a bird or not a bird. The accuracy of this classifier highly depends on the amount of data it's given, which is limited when working with bird strikes. We have recorded a small training dataset with our bird prototype bird here, but we will need both a higher quantity and quality of training data if the system is going to work well.

Once more data has been collected, the model can be retrained with the additional data, and loaded into the python program so that it may evaluate audio samples in real time.

Gathering Data

On the Sensor Pi, there are a couple folders relating to the SVM classifier, firstly there is a **data** folder in **/home/pi/Engr-2022-Capstone** that has 4 directories in it. 2 of them (**test_positives** and **test_negatives**) are for training the model, and 2 of them (**test_positives** and **test_negatives**) are for testing the model. There is a python program called **data_collect.py** which helps with the automation of data collection. **Before you run data_collect.py, please do a git pull of the most recent version of the repository from GitHub, and when you are finished recording data, you can go ahead and push the changes. This is done by doing the following 3 commands: git add /home/pi/data, git commit -m "added data", git push origin master.**

BEFORE YOU RUN THIS FILE: Open up the file with the command **vim data_collect.py** and look near the top of the file for several rows of comments (denoted by # in Python). There is a number 1-4 corresponding with each possible directory you can save audio files to, make sure to change this to match the audio you are currently recording. For example, if I was about to record training positives by having the bird prototype hit the window, I would set the number to 1. If I was about to record testing negatives by recording ambient or classroom noise, I would set the number to 4.

Running this program with **python3 data_collect.py** will continuously loop, giving you a 5 second window to record a simulated bird strike, followed by a 5 second window to reset, and then it starts over again until stopped. Once you decide to stop you may kill the program with Ctrl + C. Now, you may run the python program **process_dirs.py** with **python3 process_dirs.py** which will process the directories so that the audio is automatically trimmed to be a uniform length and processed with an audio filter and noise reduction algorithm to isolate the window strike sound.

Retraining the Model

Still on the Sensor Pi, there is a folder in **/home/pi/Engr-2022-Capstone** called **svmthings** which stores the already trained models, and code to retrain the models using the training data in the **data** folder. In this folder, you should run the python program **train_best_svm.py** which will use all of the training data, including the new data recorded, and train several different models using different characteristics of the training audio for each model. When each of these models are trained, their performance is evaluated using the **test_positives** and **test_negatives** data to see if they produce accurate predictions. The models that achieve a high accuracy score are saved to the **trained_models** directory, which is where **superscript.py** gets the trained classifier from to make real-time predictions.