System Design Document

For

TurtleTech

Team members: Christopher Angland, Laurel Dodson, Parker Brown, RoseEllen Hoke

Version/Author	Date
Version 1/ TurtleTech Team	7 October 2022
Version 2/ TurtleTech Team	8 November 2022
Version 3/ TurtleTech Team	3 December 2022
Version 4/ TurtleTech Team	10 March 2023
Version 5/ TurtleTech Team	27 April 2023

TABLE OF CONTENT

SYSTEM DESIGN DOCUMENT

1 INTRODUCTION

1.1 Purpose and Scope

The purpose of this project is to produce a payload casing for the Nvidia Jetson device and streamline the process of setting up the Jetson while also producing a functioning neural network system. The payload casing will be swappable between different drones. The payload casing will be configured in partnership with the TurtleTech Hardware team and will be tested for durability and practicality.
Currently the TurtleTech team has a functioning neural network model that identifies turtles. The goal of the senior design TurtleTech team is to build upon the current neural network model, which checks for turtles, to also be able to identify right whales. The neural networking system will only identify one specific animal at a time, it will not be able to differentiate multiple things at once. The ERAU team will produce a 3D printed payload casing and a functioning neural networking system by the end of the 2023 Spring semester.

1.2 Project Executive Summary

TurtleTech is currently a project sponsored by Northrop Grumman and the Brevard County Zoo. Embry Riddle has been given the opportunity to partner and advance the TurtleTech project. The ERAU TurtleTech team's purpose is to create a neural network system that identifies right whales while also creating a new payload case. Currently the team has produced lab procedures and tutorial videos, they have also landed on a payload casing they want to test and have a neural network system running on their machine. The procedure manual will help educate and teach anyone that joins the TurtleTech project while also streamlining the amount of time it takes to learn how to get a jetson working. The team drew multiple payloads in fusion 360 and landed on the idea of a locking mechanism. The neural network system currently identifies right whales by printing the coordinates of where they're at on the screen. By the end of the 2023 Spring semester the ERAU team will produce a 3D printed payload casing and a functioning neural network system.

1.2.1 System Overview

The neural network is built with python and deployed on a Nvidia Jetson. The Jetson board has a power supply that is connected to it to power the Jetson and USB Port that can be used to talk with the Jetson. Overall, the Jetson sits on a drone, all of this can be seen below in this figure:

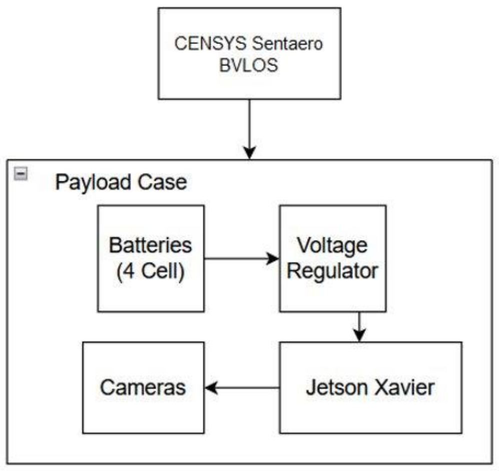


Figure 1: Basic System Overview

The mounting system has 3 main components: The mounting peg, mounting plate, and the floor mount. The mounting peg snaps into the floor mount so that it can freely rotate, but it doesn't have any other degrees of freedom. Then the mounting plate slides over the mounting pegs, then the pegs are rotated to cover the mounting plate to lock it in place.

1.2.2 Design Constraints

The design constraints to the system are limited by the Nvidia Jetson Xavier. The design constraints are also constrained by the size of the drone and how much weight can be put on the drone. The power supply creates a constraint on how much power the Jetson can pull and how much the drone needs to pull in order to be able to fly.

1.2.3 Future Contingencies

TurtleTech is a project created by Northrop Grumman and partnered with the Brevard County Zoo. This project is contingent on Northrop Grumman and the partnership with the zoo. TurtleTech relies on joint funding from the Brevard County Zoo and Northrup Grumman, therefore the existence of TurtleTech is contingent on the funding being provided. Embry Riddle has been given the opportunity to assist Northrop Grumman and the Brevard County Zoo in improving and developing the neural network system, but this is contingent on clear and concise instructions being given and also good communication skills between all of the teams. The project is also contingent on weather, turtles in the area and the functionality of the hardware and software.

1.3 Document Organization

The organization of this document is broken up into six sections. The first section is the introduction to this project, which talks about the purpose, overview, and any constraints that the project has. The second section talks about the system hardware and software architecture. The third section talks about all the inputs and outputs within the human machine interface. The fourth section is an elaborate version of the design of the software and hardware. The fifth section is about the external interfaces within TurtleTech and the sixth section is about the system integrity of TurtleTech.

1.4 Project References

Censys Technologies. (2022, October 7). *Sentaero BVLOS*. Censys Technologies Corporation. Retrieved November 8, 2022, from https://censystech.com/sentaerobvlos/

Imaging, L. (2019, January 1). *LI-XNX-CB-6CAM-FP*. Leopard Imaging. Retrieved November 8, 2022, from https://www.leopardimaging.com/product/nvidia-jetson-cameras/nvidia-nx-mipi-camera-kits/li-xnx-cb-6cam-fp-poe/

TurtleTech-erau. GitHub. (n.d.). Retrieved October 7, 2022, from https://github.com/TurtleTech-ERAU

Turtle Tech. Northrop Grumman. (2022, June 29). Retrieved October 7, 2022, from https://www.northropgrumman.com/sustainability/turtle-tech/

Build from source on Windows : tensorflow. TensorFlow. (n.d.). Retrieved November 8, 2022, from https://www.tensorflow.org/install/source_windows

Gartzman, D. (2019, June 15). Installing multiple python versions on Windows using Virtualenv. freeCodeCamp.org. Retrieved November 8, 2022, from https://www.freecodecamp.org/news/installing-multiple-python-versions-on-windows-using-virtualenv/

Getting started with Jetson Nano Developer Kit. NVIDIA Developer. (2022, November 1). Retrieved November 8, 2022, from https://developer.nvidia.com/embedded/learn/get-started-jetson-nano-devkit

How to downgrade python from 3.7 to 3.6. Stack Overflow. (1965, December 1). Retrieved November 8, 2022, from https://stackoverflow.com/questions/52584907/how-to-downgrade-python-from-3-7-to-3-6

Nicknochnack. (n.d.). Nicknochnack/TFODCOURSE. GitHub. Retrieved November 8, 2022, from https://github.com/nicknochnack/TFODCourse

Rosebrock, A. (2020, April 18). How to configure your nvidia jetson nano for computer vision and Deep Learning. PyImageSearch. Retrieved November 8, 2022, from https://pyimagesearch.com/2020/03/25/how-to-configure-your-nvidia-jetson-nano-for-computer-vision-and-deep-learning/

Tensorflow. (2021, May 7). Models/tf2_detection_zoo.MD at master · Tensorflow/models. GitHub. Retrieved November 8, 2022, from https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/tf2_detection_zoo.md

Tensorflow. (n.d.). Models/model_builder_tf2_test.py at master · Tensorflow/models. GitHub. Retrieved November 8, 2022, from https://github.com/tensorflow/models/blob/master/research/object_detection/builders/model_builder_tf2_test.py

Tensorflow. (n.d.). Models/research at master · Tensorflow/models. GitHub. Retrieved November 8, 2022, from https://github.com/tensorflow/models/tree/master/research

1.5 Glossary

- TurtleTech: sensor data collection technology for sea turtle conservation along the Florida coast.
- Nvidia Jetson Xavier: AI computer for autonomous machines, delivering the performance of a GPU workstation in an embedded module under 30W. Jetson AGX Xavier is designed for robots, drones and other autonomous machines.
- Nvidia Jetson Nano: Test env.
- Python: high-level, general-purpose programming language.
- NIST: National Institute of standards and technology
- Tensorflow:

2 SYSTEM ARCHITECTURE

2.1 System Hardware Architecture

The systems listed here are the ones that directly affect the performance and operation of the product:

- Jetson Nano:
    1. This graphics processor is the main computer on which the product will run during flight. This also will be the computer that stores any image(s) saved by the product.
    2. The onboard memory is 4 Gb, and it supports 4k footage at 30 fps.

- Rotating Peg Mounting System
    1. This mounting system will lock the payload in place. The lock will be sturdy and not allow vibrations.
- Pivoting Battery Case Holder
    1. The battery case holder will rotate with a 90-degree allowing the port to be more accessible.

This system architecture is shown in Figure 3, located in section 4.1 of this document. To overview the hardware architecture, the payload components are electrically isolated from the aircraft. For the payload, a 12v battery goes through a regulator to power the Jetson Xavier and payload camera. For the aircraft, a 24V battery powers the motors, the Orange Cube flight controller, and other avionics systems.

2.2 System Software Architecture

To modfiy the neural network we have sourced from the Kaggle competition to our parameters, we will again be using a python 3.8 environment. This environment will contain the following pacakges, along with others we may introduce as testing continues; opencv-python, numpy, Image, and others. While not currently using a virtual environment, it is currently deemed a potentially useful tool, so virtualenv may also be brought into our program as well via Anaconda. The usage of OpenCV allows for many helpful methods with opening and processing images, and Theano allows for easier handling of mathematical expressions in arrays, both of which are vital to our image recognition neural network. The current neural network we are using is called aerial-right-whale-detection/2 based on a YOLOv8 model, and it is specialized for identifying right whales, by taking in annotated files of compiled images, loading model weights from a server, checking locations within the image to return a list of coordinates where whales are detecting, then sending the resulting prediction file to a python program which puts bounding boxes on the image to easily check if detection was accurate or not. Some of the dependencies, as well as a diagram of the system, are listed and visible below.

Dependencies:

- Python 3.8 (for creating model)
- Python 3.6+ (for testing)
- Python modules:
    - Theano
    - OpenCV
    - tabulate
    - Termcolor
    - matplotlib
    - MarkupSafe
    - scikit-image
    - setproctitle
    - Pandas
    - numpy

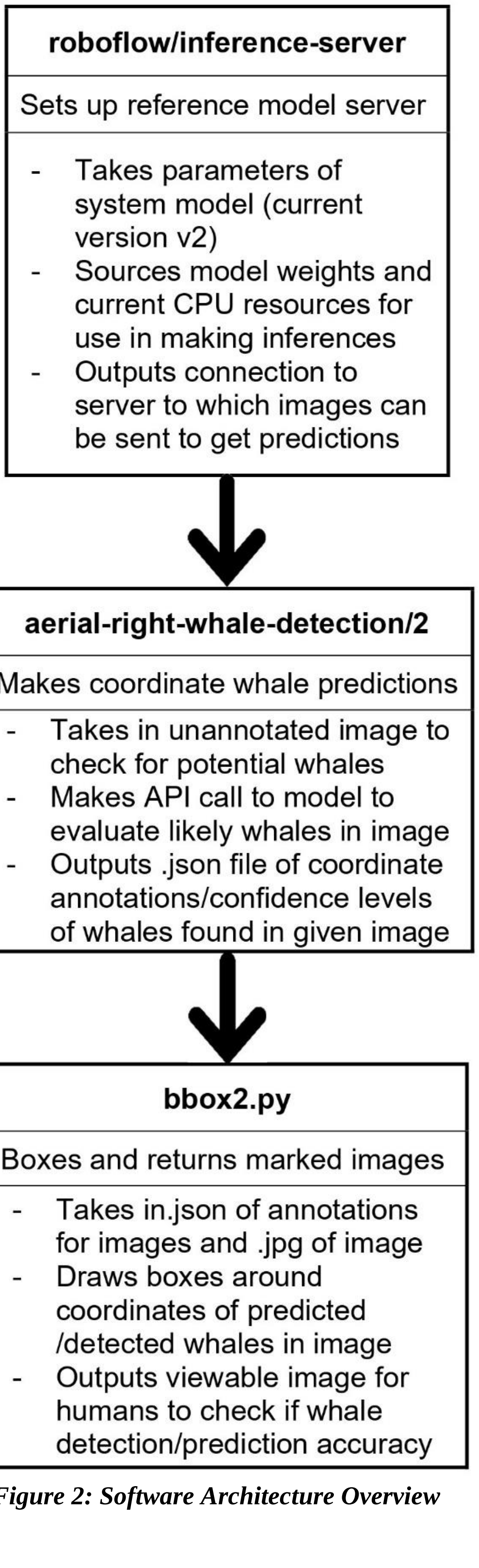

roboflow/inference-server
Sets up reference model server
- Takes parameters of system model (current version v2)
- Sources model weights and current CPU resources for use in making inferences
- Outputs connection to server to which images can be sent to get predictions
aerial-right-whale-detection/2
Makes coordinate whale predictions
- Takes in unannotated image to check for potential whales
- Makes API call to model to evaluate likely whales in image
- Outputs .json file of coordinate annotations/confidence levels of whales found in given image
bbox2.py
Boxes and returns marked images
- Takes in.json of annotations for images and .jpg of image
- Draws boxes around coordinates of predicted /detected whales in image
- Outputs viewable image for humans to check if whale detection/prediction accuracy

Figure 2: Software Architecture Overview

2.3 Internal Communications Architecture

The architecture of the system involves connecting to the Jetson Xavier via ethernet cable from a host laptop. This laptop will run a script, which will interact with the camera and signal to take pictures. The neural network will also be deployed to the target system in the same manner. Beyond that, the architecture is still being understood. The camera connects to the Jetson using a ribbon cable and must be mounted at specific angles.

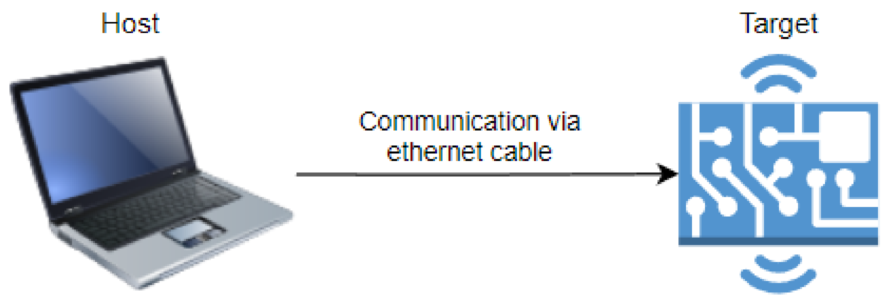


Figure 3: Communication Architecture Diagram

3 HUMAN-MACHINE INTERFACE

Humans interact with the Jetson and payload setup in various ways. Some interfacing is on the level of software, entering data to train and test the neural network, as well as inputting images from flights which are then relayed to .csv file and used by the various scripts of the image processing and whale identification programs. Currently, our interface is the linux terminal, where the input entered by humans is a direction to files of whale images that the network then processes, and the output is sorted data with coordinates of boxes containing whales and confidence percentages of sorted data. The typical process of interaction with the neural network is human sorting and labeling of training data, training the models on said refined data, then inputting new, unsorted data for testing, and checking the resulting output for accuracy before repeating the process.

The connection between human and machine will take place mostly in terms of training the neural network, as this involves both entering data into, (in the form of images and parameters) and receiving results to review from, the neural network. Currently, our interface system used for turtles is the basic Label Image image-entering system, and our setup for whales uses the same program, which allows for the adjustment of parameters and nodes of the program to fine-tune results and attempt to improve accuracy. The systems we have set up for each category respectively return images with boxed selections of the animals that have been ‘identified’; in our case, previous work was training the neural network on turtles and our new area of focus is training a new model on whales, so there is a flow to the typical process. The algorithm receives training data of specified images and attempts to find shared characteristics within that data, then is tested on a new, not-before-seen set of data so that the identification accuracy can be appraised. Then, the process repeats as the neural network algorithm is refined.

The system is intended to operate autonomously while the drone is in flight, but still receives inputs in the form of camera-taken images that will be of use to the operator when the flight has concluded and the captured data is reviewed. We are intending to port the neural network to the Jetson board, in which case the captured images will be used to generate analyzed output images before the flight has concluded, though as of now, the neural network does not run completely offline so it cannot yet be transitioned to the board. The inputs provided to the neural network during testing directly via the operator are the reference images, contained within a dataset, which train the neural network to find the expected criteria (in this case, identifying whales). However, the datasets provided by the operator in testing will imitate, but are not exactly, the actual inputs that will be received by the system.

Instead, the main point of input capture is the onboard camera mounted on the chassis of the drone (model characteristics of the camera currently unspecified, to be clarified in the future) and the processing of said input occurs within the neural network algorithm, right now on PC, but intended to happen on the Jetson chipboard, which is carried in the cargo compartment of the drone. The images captured by the camera are transferred to the Jetson via wire connection, and post-processing, the data received is a very reduced set of photos containing potential whales (for our latest trials), with measures of certainty of analysis contained along with the corresponding photos. Thus, the images (dataset and captured) sent to the Jetson board are the inputs for this system, and the categorized and analyzed images areas containing whales are the outputs of the system.

For the Payload Mounting system: the interface consists of the team interacting with the hardware mount and the pegs. The hardware mount consists of a single platform that holds the jetson processor and camera rig. The pegs are the means to which the mount is fastened to the floor. The operators installing the payload simply have to lay the mount on-top of the pegs, and then twist each peg a specified distance (90 degrees preferable).

3.1 Inputs

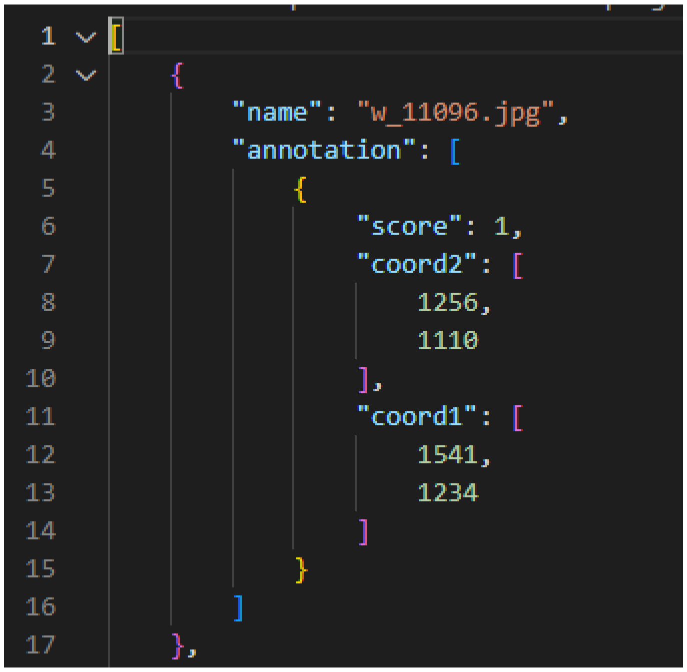
The input media provided by the operator is training data for identifying specific attributes of a target (currently a whale) within a larger picture. This is done by providing a dataset of images known to contain whales in Pascal VOC format with the aspects of the visible whale emphasized (boxed in an annotation file) for the network. The neural network breaks the entire image down into smaller sections, those containing relevant-to-the-target portions and those portions of the image that do not contain the target information, and records data about what aspects of a photo containing a whale can be used for identifying the whale. As the algorithm is fed more images in its dataset, it 'learns' criteria that are shared across varied whale images, so that when it is given a non-dataset image, it can break it down and look for those same criteria in order to identify if a shale is contained within the image or not. The essential inputs for the system function, thus, are an extensive database of images with a selected target present in each, and input images that may contain the target whales from the in-flight camera captures.

For the payload mounting system, the main input is a mount that has the proper dimensions to fit into the pegs fixated on the floor of the payload bay. For now this will remain as the

Jetson processor, but in the future other payload types (or backup copies of the payload) could be mounted.

3.2 Outputs

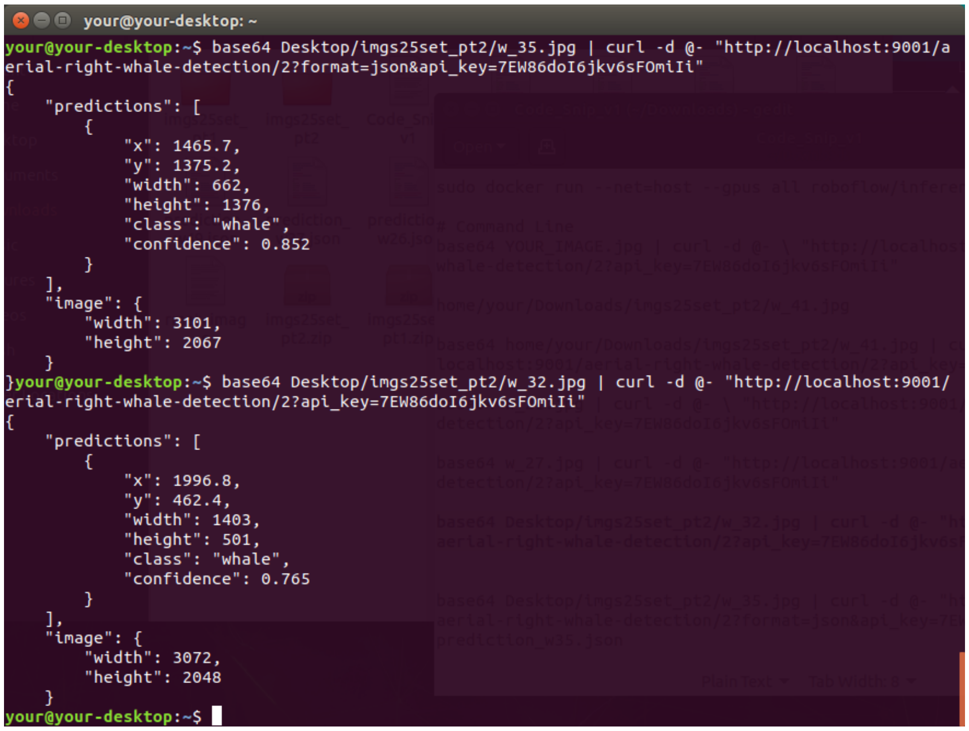
The output of the system we are creating will be identification of whales from given images. The drone camera will take images as it travels in flight above the ocean and beach, and the images of the beach and sea will be transmitted over wire to the Jetson board for processing. After processing has occurred (intended to happen in-flight, which will require optimizing the neural network to run on less memory and processor capabilities) the outputs will be a much smaller set of sorted images in a separate file, each with a selected (coordinates) section in a .json file showing a high certainty of a whale identified within the image. Depending upon how the team designs the system, some non-whale images will also be saved for comparison purposes, but many of the images without the targets will be discarded for the purpose of saving space. The most critical aspect of the outputs generated from the system, however, is the correct identification of whales from the drone-captured in-flight images showcased in the output images. Below is an image showing how a training image may be annotated for the neural network to learn from:



```
[
    {
        "name": "w_11096.jpg",
        "annotation": [
            {
                "score": 1,
                "coord2": [
                    1256,
                    1110
                ],
                "coord1": [
                    1541,
                    1234
                ]
            }
        ]
    },
```

Figure 4: Sample image with weighting and annotation, used for training model

Below is output from the neural network running, outputting prediction files on whale images:



```
your@your-desktop:~$ base64 Desktop/imgs25set_pt2/w_35.jpg | curl -d @- "http://localhost:9001/a
erial-right-whale-detection/2?format=json&api_key=7EW86doI6jkv6sFOmiIi"
{
    "predictions": [
        {
            "x": 1465.7,
            "y": 1375.2,
            "width": 662,
            "height": 1376,
            "class": "whale",
            "confidence": 0.852
        }
    ],
    "image": {
        "width": 3101,
        "height": 2067
    }
}your@your-desktop:~$ base64 Desktop/imgs25set_pt2/w_32.jpg | curl -d @- "http://localhost:9001/
erial-right-whale-detection/2?api_key=7EW86doI6jkv6sFOmiIi"
{
    "predictions": [
        {
            "x": 1996.8,
            "y": 462.4,
            "width": 1403,
            "height": 501,
            "class": "whale",
            "confidence": 0.765
        }
    ],
    "image": {
        "width": 3072,
        "height": 2048
    }
your@your-desktop:~$ 
```

Figure 5: Sample output after system processing, showing coordinates and certainty of predictions within .json files generated by neural network, showing coordinates out bounding boxes for various images.

This shows the results of those predictions overlaid as a bounding box on a whale image:

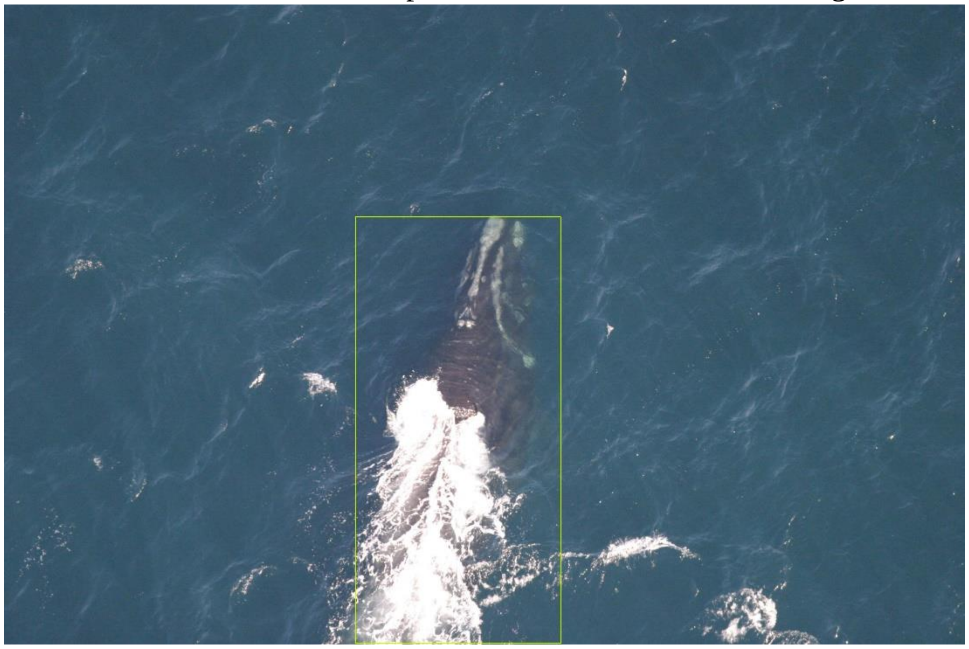
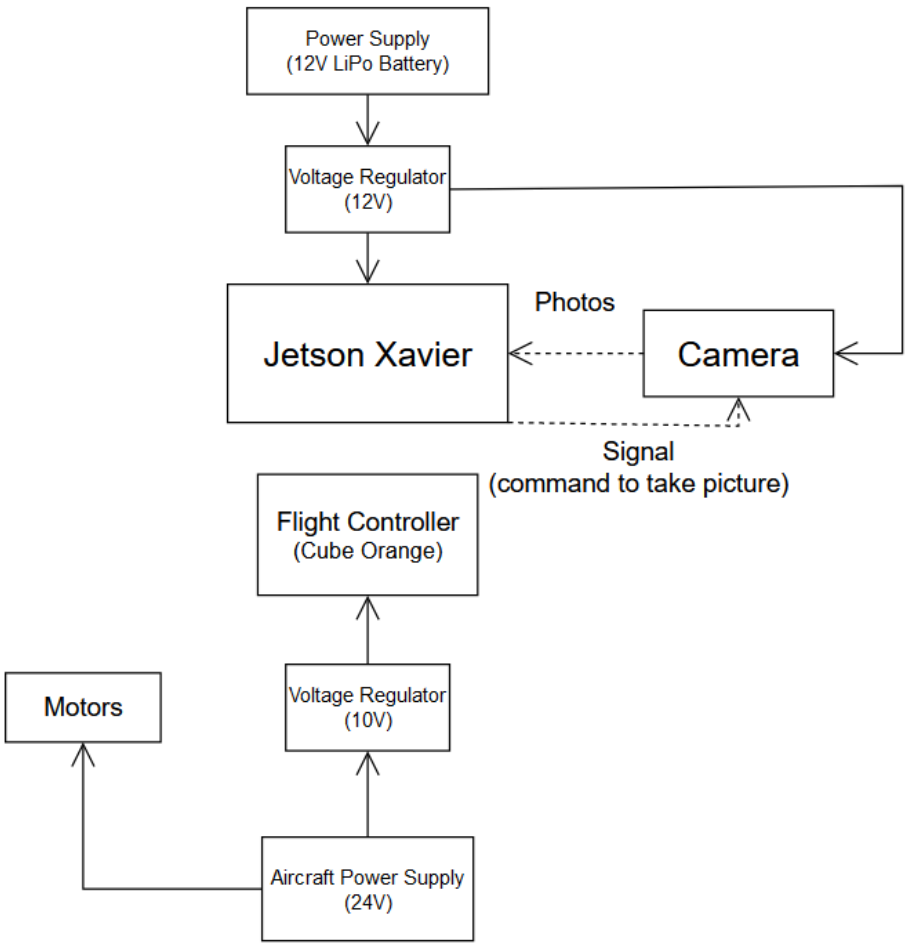


Figure 6: Sample Output image after neural-network produced prediction file is used to draw bounding box on original image. Note boxed whale.

4 DETAILED DESIGN

4.1 Hardware Detailed Design

The hardware design of the system is close to fully implemented, with only a few issues to sort out. The main components as stated before are the power supply, voltage regulator, Jetson Nano, and camera.

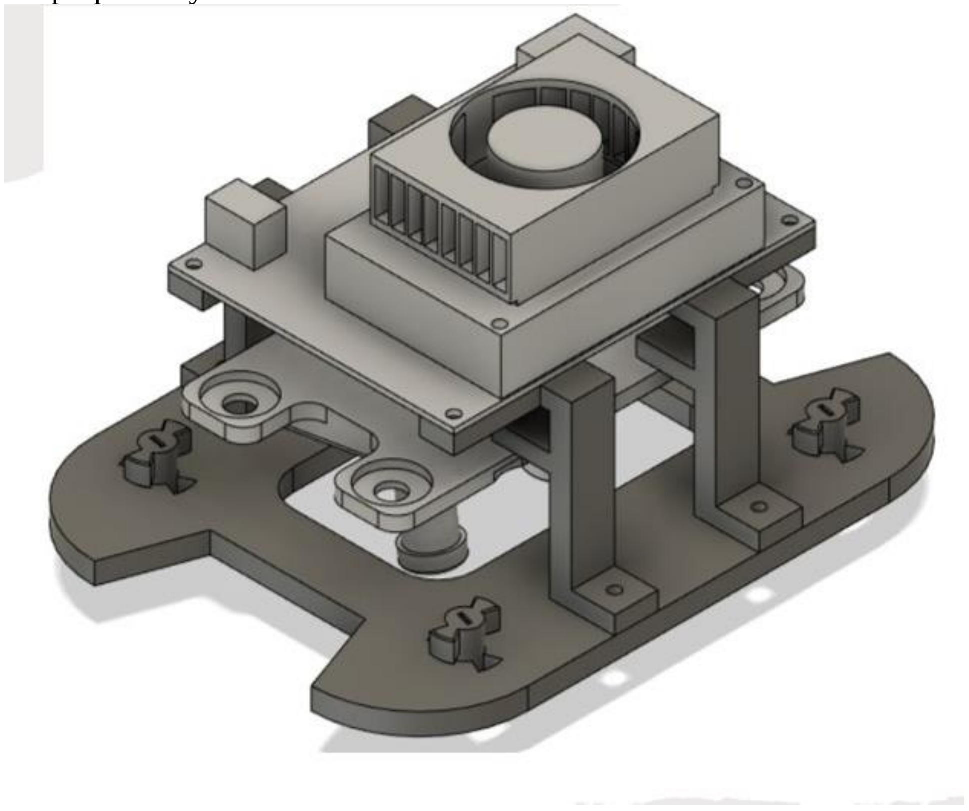


4.2 Figure 7: Hardware System Overview

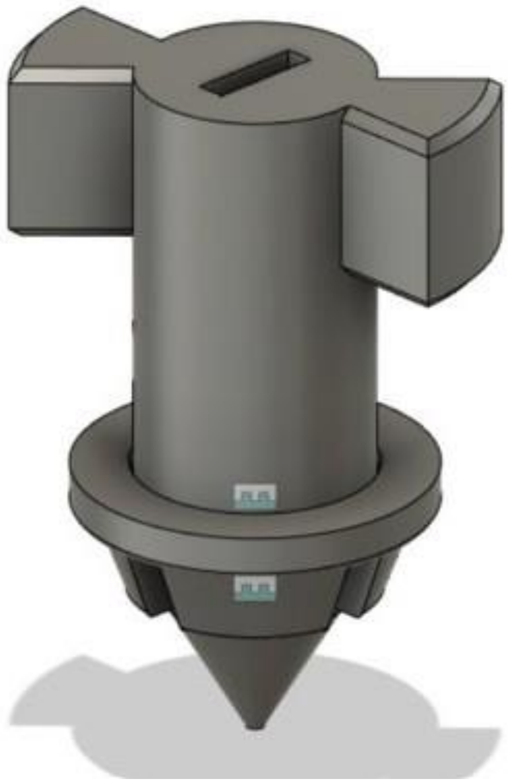
For this diagram, the flight controller includes all the avionics for the aircraft, including other components like the GPS and receiver. The dotted arrows are data connections, with this new model demonstrating how the new hardware infrastructure prevents the Jetson Xavier from shutting down due to a brown out from the aircraft power supply.

For the specifics of the mounting system, it consists of 3 main parts: The mount, peg, and fastener.

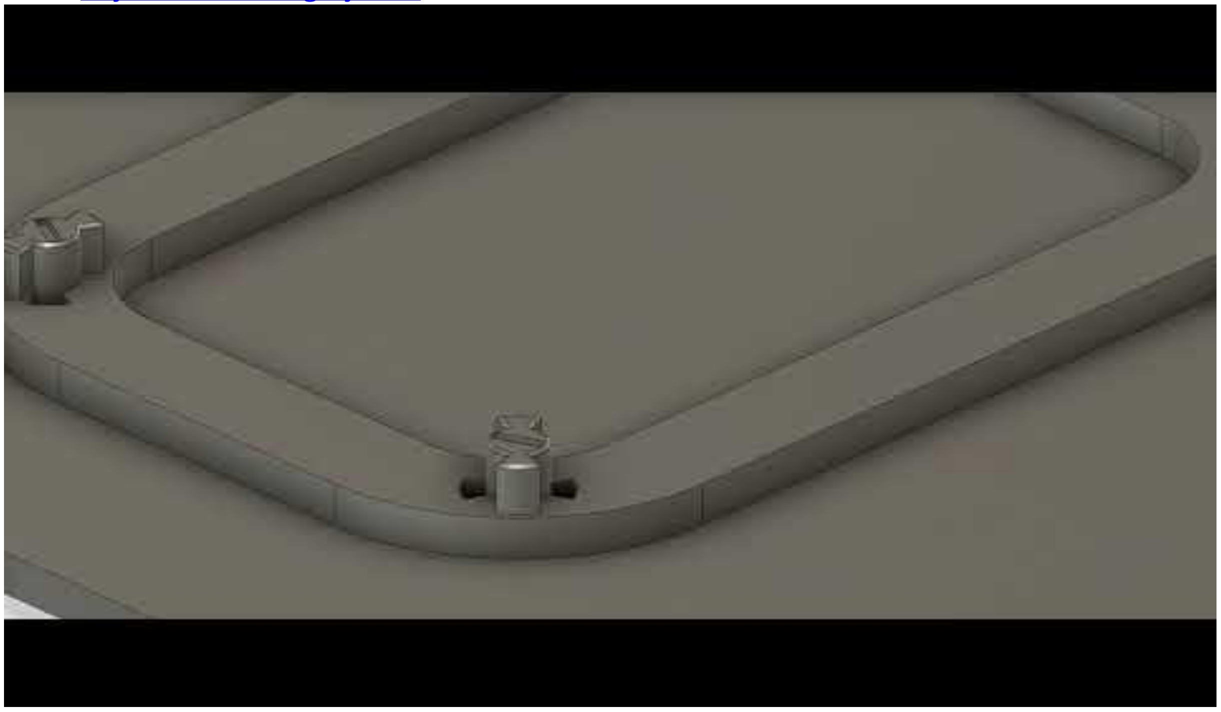
The mount itself can consist of many different components and features, as long as the floor plan of the mount has the proper size and position of peg holes. A full assembly of the proposed system is shown below.



The main component of the mounting system is the peg and latch system. When this assembly is fixated into the floor of the payload, it allows any plate with the proper hole size and thickness to be locked into place and removed with ease. The peg system would allow for any future components to be integrated into the payload bay with ease, while the pegs themselves would be a permeant feature of the payload bay.



A perviously made video showcasing how the system works is shown here: [TurtleTech Payload Mounting System]



4.3 Software Detailed Design

To process the images, we are using an inference server making API calls to our current model, hosted on Roboflow. This provides the functionality to retrain the model and easily swap in the updated version, provided the Jetson Nano has an internet connection. Additionally, to train this model, we are utilizing Label Image to generate annotation files for given whale images, which allow the model to adapt to various types orientations and quality levels of whale images. Lastly, we are using a python file to analyze the prediction data created by the neural network and add boxes to the images where predictions are made. This allows us to quickly evaluate select images to see if the neural network is making detections accurately.

The program that calls the model is a docker mount called roboflow/inference-server. The model is trained through roboflow, taking in annotated data labeled by the user in files, and utilizing default parameters of epochs and train time to categorize the specific data to search for within the entered images. The model has several dependencies which must be within the system for it to function properly, so when the model is called, it performs a check for these before continuing. The inference server calls upon the latest model weights stored for the neural network to determine the criteria upon which to evaluate new images to detect and predict whale locations within them. This data can then be used to actually test new image sets and output .json files of processed data.

Currently to test the model, we obtain the latest model weights from the previous trained session via starting a new session of the inference server. We then leverage those points via running a new instance of the model, entering into it a random image and having it call its prediction function in order to test that particular image. We then grab the new predictions in the outputted .json file from the test image, and run an additional python program on in to label the image with a bounding box and output it to display to the user.

In future iterations we hope to move this entire system onto the Jetson Xavier, and potentially find a way to move away from the reliance on Docker and having any part of the model online. That setup of model would use a script to point to a directory of captured images from the Jetson Xavier, identifying and labeling the image if a whale is present. This script would then sort the images in another separate directory, with the associated labeling.

4.4 Internal Communications Detailed Design

The package, which includes the Nvidia Jetson Xavier and the camera used, is a system that has already been built. The communication systems for those devices have been created and we will not be designing anything additional for communications at this time.

However, the software components will need to be able to speak with each other. The current state of the software runs a complete script, which will activate the components needed. But ideally, we wish to create a Python program, which will contain some system() commands, that will be able to communicate to each component effectively. This is the ideal form, as the former feels very segmented.

5 EXTERNAL INTERFACES

5.1 Interface Architecture

The Jetson is set within the payload, and connected via ribbon cable to the cameras mounted within the same payload. This allows for the cameras to take an image and send it to the Jetson directly, upon which its data is then saved to a USB inserted into the Jetson. The USB is removed after the flight, and its captured images transferred to a PC; then they can be sent to the neural network for processing.

5.2 Interface Detailed Design

Images which are captured through the drone will be processed through the neural network which uses Python to implement the image recognition program. Python will use OpenCV and Label_Image for real time image recognition and YOLOv8 (previously TensorFlow) for machine learning. A neural network processes the image data, noting down coordinates around the selected whale. The coordinate writing will also include information about the confidence level of the recognition in the images. The confidence level will mark how the estimation's accuracy. The prediction data of coordinates will then be processed through additional python files to display a bounding box around the detected whale in the image.

6 SYSTEM INTEGRITY CONTROLS

The Nvidia Jetson and the neural network system are the 2 main objects the ERAU TurtleTech team is working. It is the responsibility and job of the ERAU team to keep the system as secure as possible. The team will ensure to have a secure password and will keep all vital information about the system secure and can only be accessed by the people that need to. Below are the main system integrity controls that the ERAU TurtleTech team will comply with.

- The internal security of the device is restricted for outside users while the team members have full access to the device. The device is stored in a closed lab that requires special entry and will not be left in the open, to reduce the risk of unauthorized individuals accessing it; however, team members have permission to get into the lab, in order to do testing and refining of the program on the Jetson.
- The audit procedure will be conducted to ensure there are no risks that can be exploited and if there is a risk then an assessment of the risk will be conducted. This will be in collaboration with the other teams on the TurtleTech project – potential vulnerabilities the Jetson may introduce to the rest of the system (and fixes for any) will be discussed, as it is mainly the drone itself that could be a threat or danger.

- The assessment will list the risk, the level of severity, and how the risk will or will not be fixed, which will be decided upon the team, and if action needs to be taken such actions to repair vulnerabilities will be documented
- The assessment will follow NIST 171-800 standards
- Once the neural network program is successfully stored within the Jetson, a simple password protection wrapper program will be set up for the network, securing the neural network on the Jetson with a username and password
- To ensure that the workings of the system (and thus, potential vulnerabilities of the system) are not readily available, the documentation of said system present in the Microsoft teams and GitHub can only be viewed by people with access links, and access links will only be distributed to team members, TAs, and the professor.