System Design Document

For

TurtleTech

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

Version/Author	Date
Version 1/ TurtleTech Team	7 October 2022
Version 2/ TurtleTech Team	8 November 2022
Version 3/ TurtleTech Team	3 December 2022

TABLE OF CONTENT

SYSTEM DESIGN DOCUMENT

1 INTRODUCTION

1.1 Purpose and Scope

The purpose of the TurtleTech project is to produce a functioning neural network system on a Nvidia Jetson device that identifies aerial turtle images against non-turtle images in real-time. 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 functioning neural networking system by the end of the Fall 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 turtles in real time while also helping troubleshoot the Jetson overheating problem. Currently the team is producing a lab procedure walkthrough on how to power on a jetson. 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 ERAU TurtleTech contributions are valuable to the project as the current state of the neural network is analyzing turtles after the fight is over. With a real time, neural network system, the scope of turtle hunting can be broadened to eventually even tracking turtle trails or even specific species of turtles. By the end of the Fall 2022 semester the ERAU team will produce 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 has power supply that is connected to it to power the Jetson and USB Port that can be used to talk with Jetson. Overall, the Jetson sits on a drone, all of this can be seen below in figure

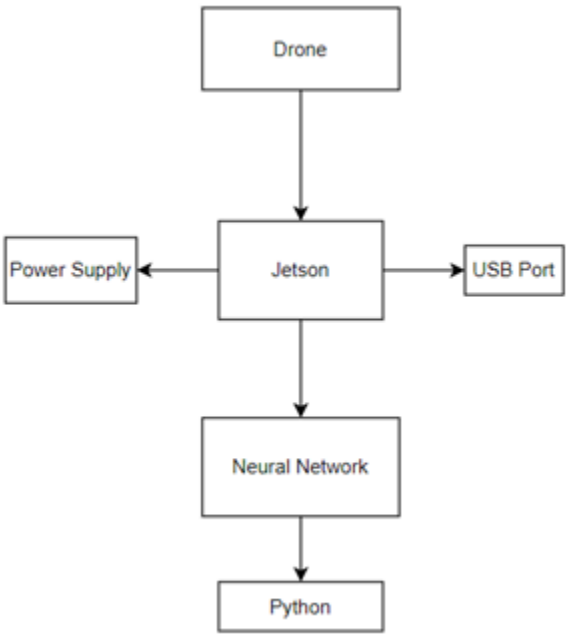

Figure 1: Basic System Overview

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:
  - 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.
  - The onboard memory is 4 Gb, and it supports 4k footage at 30 fps.
- Battery power supply
  - Due to issues running the Jetson Xavier off of the aircraft power supply, a 12V LiPo battery has been added to the payload bay specifically to power the processor and camera.
- Camera
  - We currently do not know what make or model the camera is, however, it is a crucial aspect of the product since it is the primary input.

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

For the neural network to recognize the turles, we will use a python 3.8 environment. In this environment we will have pip installed the following packages: virtualenv, opencv-python, more to come. The "virtualenv" library is used to enter in and out of the virtual environment. We created a standalone python environment which the software will depend on, to avoid any additional complexity or issues related to running different python versions. OpenCV will be used in conjunction with Tensorflow. This allows us to add components of the OpenCV computer vision library. The "labelimg" software is a python program which is used to label images for the purpose of training and testing a neural network.

- Python 3.10 (for creating model)
- Python 3.6+ (for testing)
- LabelImg
- Python modules:
    - virtualenv – creating older python version environments
    - tensorflow 2.5+
    - tensorflow-gpu (for training on gpu)
    - opencv-python
    - matplotlib
    - scipy
    - Pillow
    - pyyaml
    - tensorflow_io
    - tf-models-official
    - object_detection
    - numpy
    - Random

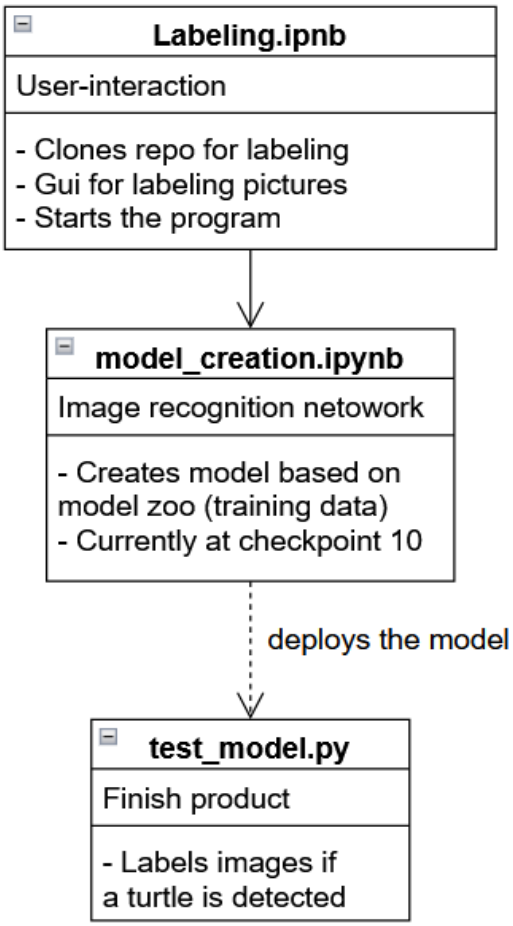

Figure 2: Software Architecture Overview

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 some sort of cable. More to come

Figure 3: Communication Architecture Diagram

3 HUMAN-MACHINE INTERFACE

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 is the basic TensorFlow image-entering system, which allows for the adjustment of parameters and nodes of the program to fine-tune results and attempt to

improve accuracy. The system we have set up also returns images with boxed selections of the images that have been 'identified'; in our case, we are training the neural network on turtles, 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 and generates outputs that will be of use to the operator when the flight has concluded and the captured data is reviewed. The inputs provided to the drone 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 turtles). 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 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 on the Jetson is a very reduced set of photos containing potential turtles, 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 showing turtles are the outputs of the system. This is described in greater detail below.

3.1 Inputs

The input media provided by the operator is training data for identifying specific attributes of a target (currently a turtle) within a larger picture. This is done by providing a dataset of images known to contain a turtle, with the aspects of the visible turtle emphasized 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 turtle can be used for identifying the turtle. As the algorithm is fed more images in its dataset, it 'learns' criteria that are shared across varied turtle 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 turtle 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 turtles from the in-flight camera captures.

3.2 Outputs

The output of the system we are creating will be identification of turtles 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, each with a selected (highlighted or boxed) section showing a high certainty of a turtle identified within the image. Depending upon how the team designs the system, some non-turtle 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. In addition, the outputs may be marked with timestamps and location data if that information is available to do so. The most critical aspect of the outputs generated from the system, however, is the correct identification of turtles from the drone-captured in-flight images showcased in the output images.

Figure 4: Sample image after processing by the system, with boxed and labeled turtle and track portions of the image

Below is an example of the same image, but with a sample time stamp:

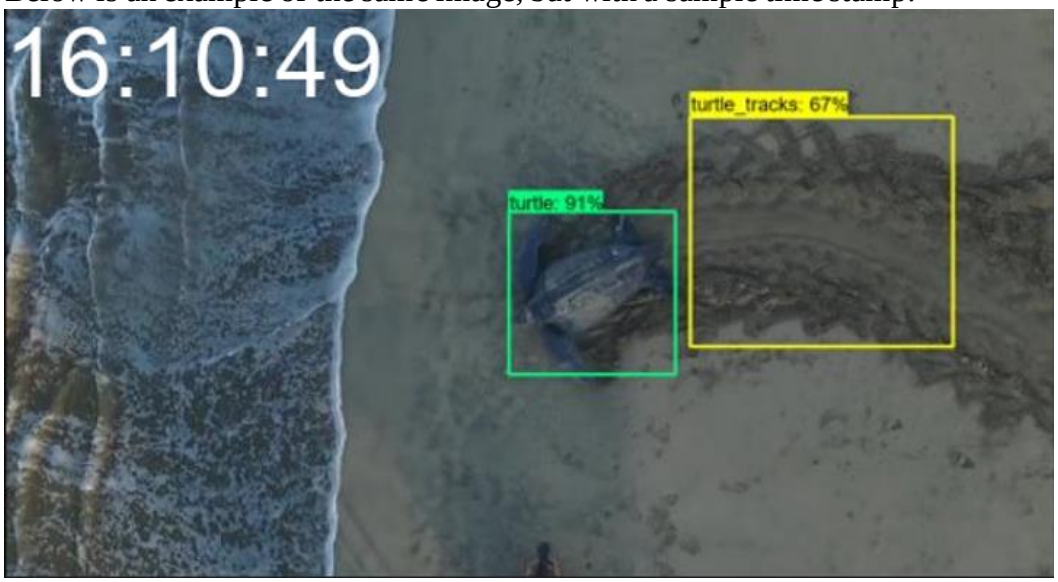

Figure 5: Sample image after system processing and timestamping – both identified turtle/tracks and timestamp

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.

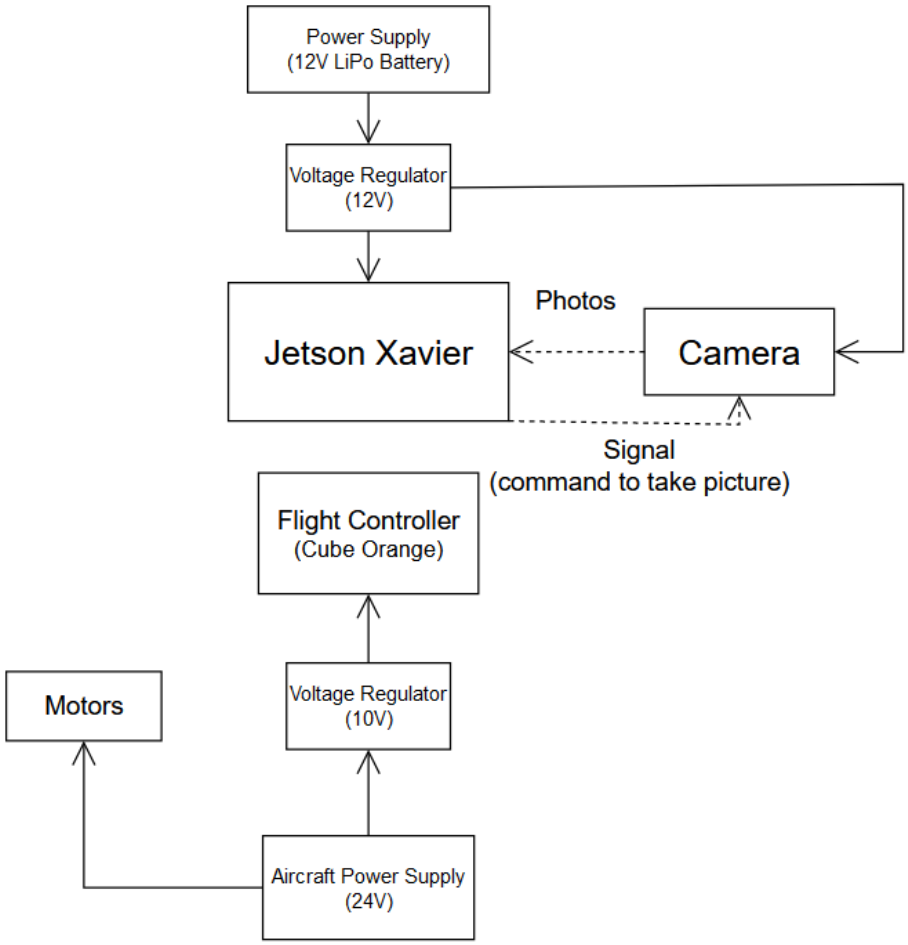

Figure 6: Hardware System Overview

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.

4.2 Software Detailed Design

Starting from labeling the image, we are using a GitHub repository called labelImg. This repository is free and provides a simple GUI that allows you to draw boxes around specific portion of an image, assign a label, then produce an .xml file, that allows us to leverage the information in our model.

The program that creates the model for the NN is called model_creation. The program will start with the user creating the specific paths for use later in the code; we are also defining the prebuilt model and obtaining the URL to later clone it to one of our paths. Then we download the pretrained model we specified earlier and install the dependencies our model will require. Here, we also run a test to make sure what is required is installed. The next step is to create our label map (turtle, turtle_tracks), that matches the labels which we created using labelImg. Then, we create something called tensorflow records, this creates a file format which our training model will use to save the checkpoints of the model. The program will then get a predefined pipeline config file associated with the pretrained model. We then alter the pipeline to alter the epochs and batch size. Finally, we train and evaluate the model.

Currently to test the model, we obtain the latest checkpoint from the previous trained session. We then leverage that checkpoint using a series of commands in "test_model". We then grab a random image using a function, "chooseRandomImage()", and use the call "object_detection.utils.visualization_utils" to label the model and visualize the results.

In future iterations we hope to use a script that loads the previous checkpoint. This script will use a path to a directory of captures images from the Jetson Xavier, identifying the image if a label is present. This script would then sort the images in another separate directory, with the associated label, with a watermark of when it was captured.

4.3 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 connected to the hardware of the drone through a USB port. The jetson will be powered by a battery installed into the drone. Images with embedded text are saved to the SD card in the Jetson. An ethernet cable is used to interact with the jetson through a PC. An HDMI cable is used to display the images captured.

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 for real time image recognition and TensorFlow for machine learning. A neural network processes the image data creating boxes around the recognized turtle. This box will be marked with a name differentiating between a turtle and turtle tracks. A confidence level will mark how accurate the estimation is.

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.