University of Arkansas – CSCE Department
Capstone II – Preliminary Report – Spring 2023

IEEE Robotics Competition Design

Nicholas Brown, Callum Brutton, Jase Cornett, Austin Flynn, Stephanie Stock

Abstract

Every year, the IEEE holds a regional robotics competition for colleges and universities. For us here at the University of Arkansas in region 5, the upcoming 2023 competition will be held in Denver. Our capstone team will be competing in this competition with the Robotics Interdisciplinary Organization of Teams (RIOT), a Registered Student Organization at the University of Arkansas. Our problem is that RIOT wants to compete in the IEEE Regional Competition and needs help designing the software aspects of the project. Our objective will be to help RIOT create a functioning robot and drone pair that will be able to compete in the IEEE competition in April 2023. We will approach the solution to this problem by working cohesively as a Capstone group and cooperating with RIOT to determine the best solutions to our problems. Our goal is to make a significant impact on the engineering community at the University of Arkansas by collaborating with engineers of different disciplines and showcasing the ability of its students to design and implement a functional autonomous system.

1.0 Problem

The challenge of the 2023 Region 5 IEEE Robotics Competition focuses on the collaboration of two robots that will implement an autonomous system. The pair will communicate together for environment mapping as necessary for critical pathfinding. One of our jobs will require us to establish communication between the drone and ground robot so that they can effectively work together to complete the competition course. Another job we will do is to take this data collected by the drone and ground robot sensors and process it, onboard the ground robot's computer. We will then use this processed data to move the ground robot depending on where it needs to go next. Furthermore, we will also use the data from both to have the drone land and take off from the platform on the ground robot throughout the duration of the competition.

The importance of this design is its ability to translate to real-world designs. Several proposed autonomous designs, such as autonomous drone deliveries that stem from a delivery truck in urban areas or a ship that uses drones to monitor the status of stormwater outflows, are simply larger-scale implementations of the IEEE competition. This competition provides a great deal of experience for several students to begin implementing autonomous robotics in our society. Autonomous robotics lessen several margins of error that humans introduce. With the introduction of robotics, the productivity of nearly anything can be improved.

2.0 Objective

The objective of this project is to construct and program an autonomous duo that can be entered into the IEEE Robotics competition and hopefully win the entire competition. These two distinct types of robots will have separate functionalities that allow them to collaborate on a set of assigned tasks. The duo will consist of a ground robot designed similarly to an autonomous car. It will be able to navigate a course of boxes with a set order and finish the course within a given time limit of 10 minutes [5]. The drone will be responsible for providing terrain mapping data and scanning QR codes on top of the boxes that will aid the ground robot in calculating the path to the next objective. It needs to function as a normal flying drone while having constant communication with the ground robot, as well as the ability to take off and land from a platform on top of the ground robot. To complete the course, the drone is required to land on the ground robot once all objectives have been completed [5].

There are virtually no physical aspects of the ground robot that need to be completed for the competition which means our biggest responsibility is working on the software of the ground robot. The only hardware part that our CSCE Capstone team will have to work on is the onboard computer for the ground bot, as well as the sensors that will collect data during the competition. We will also be sending information and data for the computer to calculate how the wheels should move so the ground bot can move throughout the competition. Since the drone comes pre-built, our role calls us to program its functionality to complete the given tasks. This will encompass programming it to scan the competition field and QR codes, as well as taking off, flying, and landing.

3.0 Background

Robotics encompass a wide variety of ideas and uses. These can range from little autonomous robots that vacuum your floor to the NASA Mars robots that roamed and map the planet [6]. The design of our robot and drone will not be as complicated as what NASA can do, but it will still come with obstacles and problems we will need to complete. To understand the scope of our project, there is background information that needs to be understood.

3.1 Key Concepts

For our ground robot to be able to move with directions, the course will need to be mapped. This will be done for us by using street view mapping. Our drone will be flying above the competition field with a camera pointing down at the field. This will allow us to create software to process these images into a 3D map of the terrain [1]. This data of the terrain will then be sent down to the ground robot. The ground robot will then be able to process this data to generate directions to the objectives.

Our ground robot and the drone will need to be able to collect data, so they know where they are at any given time. This will be done by using various sensors to gather information and data. This data will be gathered via a stereo camera, a normal camera, a LIDAR sensor, and odometers on the wheels. The platform that the drone will be landing and taking off from will also use contact switches to know when the drone has landed. The drone will use its built-in camera to scan and record data about the competition course from above the ground robot. This camera will also need to have the ability to scan and read QR codes and relay that data to the ground robot, to help decide what box is next.

Data processing will need to be done during the competition as well. The data that these various sensors collect will be processed by the ground robot using the onboard computer. The computer will then send back any needed information to the drone so that it knows its location and the ground bot's location. This data will determine where and how the ground robot and drone will move to their next location.

With a scanned map of a given area and the ability to move around, the robots need to know where they need to go next. This can be done using fiducial markers that give the robot or drone a location that they need to reach [4]. A marker is placed in the "image" of the field, giving the ground robot a distance that needs to be covered to make it to that point.

3.2 Related Work

Much background work has been done on the techniques and algorithms we will be working with to complete our project. We will be implementing the discoveries of documented research in robotic navigation using fiducial markings, sensor data, and terrain mapping via a 3D camera. These papers show problems that apply to autonomous robotics and how researchers in the field have solved them.

Dr. Karen Bradshaw and Luke Ross from Rhodes University have a research paper related to fiducial markers [2]. This paper shows the implementation of using robotic navigation and fiducial markers to find the most optimal route for a robot to move. The authors give insight into their design using algorithms, vision systems, and pattern recognition. Our group will aim to improve their work by testing and implementing new algorithms to find a more efficient path for movement.

Another research paper about sensor data and processing were done by Rajalakshmi Krishnamurthi and Dhanalekshmi Gopinathan from the Jaypee Institute of Information Technology, Adarsh Kumar from the University of Petroleum and Energy Studies, Anand Nayyar from Duy Tan University, and Basit Qureshi from Prince Sultan University [4]. Their paper dives into the problems of using sensors to collect data. They discuss how they use sensor data along with other data for rapid decision-making. Our project will differ from this slightly as it will be necessary to make rapid decisions, but communication needs to happen between the drone and robot, with the shared data they collected from their sensors.

Research has also been done in mapping a terrain using a 3D camera. This research was done by Justinas Miseikis, Kyrre Glette, and Jim Torresen from the University of Oslo, as well as Ole Jakob Elle from Oslo University Hospital [3]. Their research shows how 3D camera mapping could give a trajectory for a robot to travel. We will use the research of processing 3D images of a terrain for a map of the playing field. We will improve their designs by using the information from the images of the drone and data on board the ground robot to determine the best trajectory for our ground robot.

4.0 Approach and Design

4.1 Requirements

- Ground Robot Specifications
  - Only one ground robot will be allowed to compete (per team) in each round. This ground unit must have a fixed landing pad located on the highest point of the robot. The landing pad must be a minimum of 20 x 20 centimeters. This unit must also be equipped with a master ON/OFF switch visible to a Field Judge [5].
- Ground robot sensors
  - Several factors will influence the navigation of the ground unit. We will be utilizing a multitude of sensors, including an Intel RealSense D415, Pi Camera module V2, ultrasonic sensor, and encoders. These sensors will have the ability to dictate the ground robot's next actions.
- Drone Specifications
  - Only one drone will be allowed to compete (per team) in each round. Acceptable units include any commercially available (Ryze) DJI Tello model. No modifications to the factory propellers, propeller guards, motors, motor drive systems, or batteries will be allowed [5].
- Communication between Ground and Aerial Units
  - Our units will need to be able to communicate with each other autonomously. The Jetson Nano will act as a router to connect to the drone. A UDP connection will be established between the two to send Python commands.
- QR Reading
  - Both units will need a method to read QR codes, as each of our objectives will have a different code on top and bottom. The inside of each box contains the ID number needed for the next objective, while the outside contains the box's ID number [5]. The ground robot will do this using the Pi Camera, which will be mounted facing upwards [7]. The drone is equipped with a camera that will allow us to scan the outside of the box.
- Secondary "Poison" Objective
  - During the second round of the competition, two teams will be in the arena at the same time [5]. During this round, our ground and aerial units will need to complete the same tasks as before while also avoiding the opposing team. This round also adds an optional additional objective to the team that completes the course first. After completing the course, this team's aerial drone has the option to become "Poison." If the drone takes this objective, it will take off and search for the opposing team's ground unit. Landing on the ground unit's empty landing pad will end the round and cause the "Poisoned" team to lose all points collected in the round. If the total time for the round expires before either team is poisoned, there is no penalty awarded to either team.

4.2 Detailed Architecture

4.2.1 Aerial Drone

- For this competition, we had to use a specified drone per the competition rules. This drone was the RYZE DJI Tello [Fig. 1], which is a basic education drone for learning to code on drones. The drone is a relatively small practice drone that comes equipped with four 3-inch propellers, a range finder, a barometer, LEDs, 2.4 GHz Wi-Fi capability, and a camera that can record 720p video as an MP4. Onboard the drone there is also an Intel image processor allowing us to run tasks for the camera. The drone has a max flight time of around 30 minutes using a detachable battery that can be charged through a Micro USB. While flying, the drone has a maximum distance of 100 meters it can travel, a maximum speed of 8 m/s, and a maximum flying height of 30 meters.

Figure 1.

- We needed to implement a variety of functionalities for the drone. Unlike most uses for drones, we couldn't control it using a remote controller or from an app, you could download on your phone. This drone had to function autonomously while being controlled through a UDP from a Python file. To connect to the drone through a UDP connection, the drone used its own Wi-Fi signal that a computer could connect to. This would allow us to communicate with the drone to control it in a variety of ways for the competition.
- To have the ground robot navigate through the course correctly, we needed to have the drone fly around the competition field to get an image of what it looks like. To fly the drone, Tello has a pre-made SDK for all their drones that allows one to program the drones. This SDK came in Python, allowing us to control the drone exclusively using Python. The SDK contained a variety of functions that would allow us to fly the drone and access its camera.
- For us to be able to run this Python code, we needed to have a Python Interpreter on our computer using, in this case, Python 3.6. Then, after importing the Tello package in our Python, we could create a variable that would be our Tello class and let us reference the functions in its SDK.
- For the drone to move autonomously, we wrote Python code for a finite-state machine. This FSM will have a variety of states for what the drone needs to do at a given moment during the competition. The states will also have the needed functions, like "takeoff," "land," "steamon," "streamoff," and "move_forward."
- The computer that the drone will be connected to will be an Nvidia Jetson Nano (mentioned later). The Nvidia Jetson has Wi-Fi capabilities that allow us to use the UDP connection to send commands. Using the Nvidia Jetson as the computer controlling allowed us to have one central unit for both our ground robot and the drone.
- The object detection used to find the boxes on the course was used with the drone's camera via OpenCV. The objection detection would be used for the drone state while it is looking for a box. The drone would then rotate until it was able to center the box in the camera's view using the detection model.

- The drone also needs to be able to read QR codes, so the ground robot knows where to go next. The drone can do this using OpenCV that has a pre-built QR reader named "QRCodeDetector". When reading the code, a letter A-F or "Done" would be read and relayed down to the Nvidia Jetson on the ground robot.

4.2.2 Ground Unit

- The generalized design of the ground unit was created by RIOT members with some insight from our Capstone group to factor in where certain components should be placed.

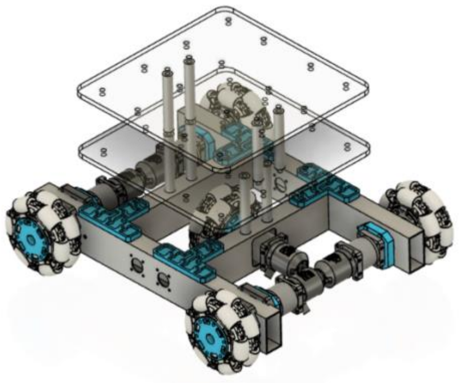


Figure 2.1

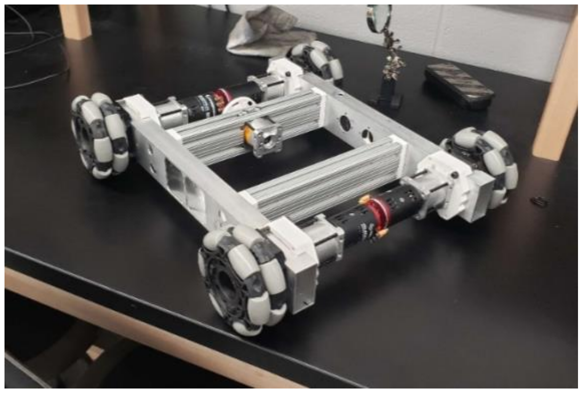


Figure 2.2

- The frame of the unit was created with a CNC machine that cut the aluminum to spec. The lower section of this frame creates housing for the five wheels as well as the motors associated with each.
- The upper sections of the unit are made of two acrylic plates that are supported by multiple aluminum rods. This allows for the Raspberry Pi Camera Module to be placed clear of the drone landing pad without causing any obstruction to the functionality of the QR code reader.
- The wheels that we decided to use consist of five Rev Robotics Omni Wheels that are four inches in diameter. Four of these wheels are placed in a standard orientation while the fifth wheel is placed in the center of the unit, perpendicular to the rest. This allows the ground unit to traverse the arena without the need to rotate, helping to eliminate drifts in angle compared to the obstacles in the course.
- Connected to each of the wheels will be a QuicRun 3650 G2 Sensored Brushless Motor. The sub-model used is the 25.5T Designed for 1/10 to 1/12 scale RC cars, these provide more than enough torque to power our machine. While not finalized, the motors are currently expected to run on a 5.2:1 gear ratio, the manufacturer suggested a 4WD Buggy with similar specs to our unit.

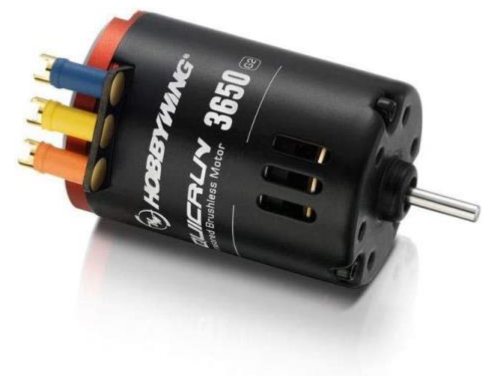


Figure 3.

- The hub for the motors consists of an ODrive V3.6. This hub is connected to the Jetson Nano through a USB 3.0 port and allows the Nano to offset some of its operating costs to this device. The ODrive also features a 12V to 56V DC power input that is capable of supplying sufficient power to each of the motors.

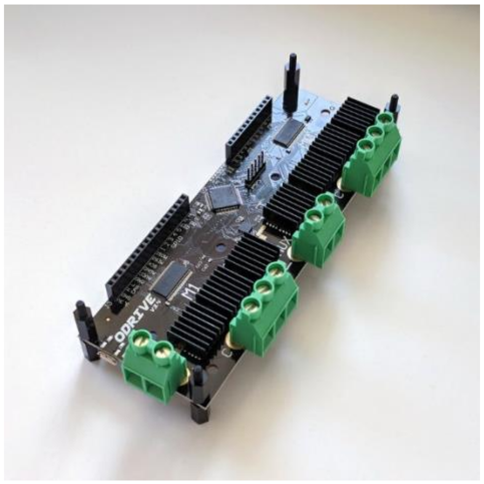


Figure 4.

- Although computer vision will be our main avenue for controlling the ground robot, we found it important to utilize an ultrasonic sensor to be our emergency stop if all else fails. The ultrasonic sensor that we are utilizing for this project is the RCWL – 1601 Ultrasonic Distance Sensor. The key aspect that had us decide to utilize this sensor, in particular, is its low-power nature and ease of use. The three 3V this sensor uses means that we won't have to worry about this sensor detracting from anything else that happens to be important to the key operation of the ground bot. Another reason we chose this sensor was that we could have the GPIO pins being utilized allowing us to save the USBs for the RealSense camera and ODrive.
- The ground bot will have a camera facing upwards to facilitate scanning the QR codes and looking for the poison drone above the ground bot. For this camera, we decided to use the Raspberry Pi Camera Module V2. The camera attaches directly to the Jetson Nano through the use of the Camera Serial Interface connector. This camera will utilize the same programs for object detection and QR code scanning the drone uses to utilize the limited memory we have available.
- On the ground robot, we will also have an Intel RealSense camera. The Intel RealSense D415 camera acts as another one of our sensors. With its depth tracking ability, it is vital to the function of the ground robot. The program that runs it is written in Python. Object

detection was added to its functionality so it can recognize the boxes within the course to successfully navigate to a box. Using depth detection, the program will be able to differentiate the sides of the box. The program can also generate an ASCII representation of the image, which can be used to center the ground unit with the box without using too much memory.

- The central program controlling the autonomy of the ground unit is ROS2 (Robot Operating System 2). This program was specifically designed as a software development kit for robotics. Building on top of ROS1, this open-source SDK has a wide variety of libraries available for use, including many pre-existing ones for components involved in our design, such as the Intel RealSense Camera and the ODrive. As the Jetson Nano is running on Ubuntu 18.04, we decided to use the Eloquent Elusor Distribution. Although this distribution is no longer receiving updates and reached the end of life in November 2020, it was the latest version with Ubuntu 18.04 support.

4.2.3 Nvidia Jetson Nano

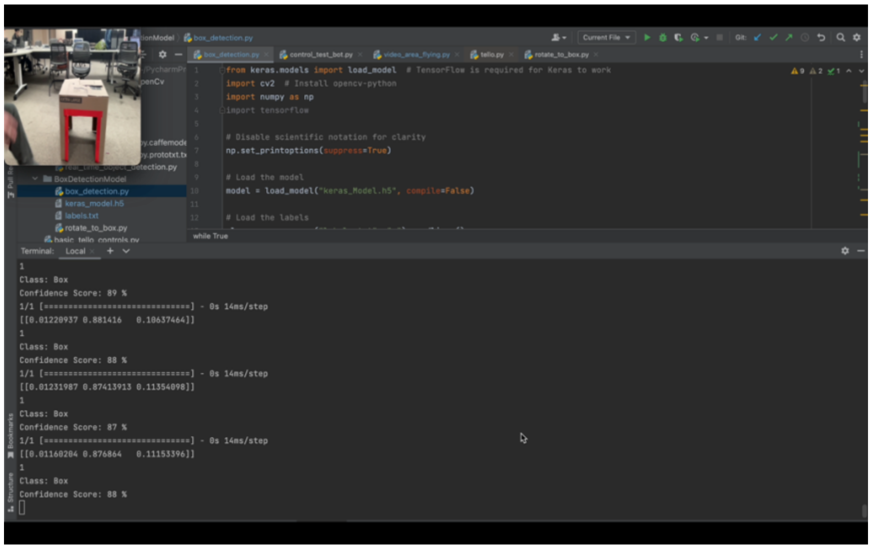
- The computing power for the ground bot and drone will be delivered by the Nvidia Jetson Nano. Responsibilities of the Jetson Nano include running our object detection, scanning QR codes found within boxes, and controlling the movement of the ground bot and drone. Object detection and QR code scanning will happen over the Trello drone and the Raspberry Pi Camera V2. The drone will communicate with the Jetson through a 2.4 gigahertz Wi-Fi connection. While the Raspberry Pi Camera will utilize one of the two Camera Serial Interface connectors found on the development board. The GPIO found on the Jetson Nano will be used to collect data from the ultrasonic sensor. Due to how the operating system works on the Jetson Nano timely updates are proving to not be possible. The current plan is to have the ultrasonic connected to an Orange Pi GIPO pin and then connect the Jetson to the Orange Pi through Inter-Integrated Circuit to have the data transferred to the Jetson in real-time.
- The operating system being used is Nvidia's distribution called Linux for Tegra. This distribution includes all of the software that is part of the Nvidia Jetpack software suit along with the necessary kernel modifications for their line of single-board computers. The latest available distribution available for the Jetson Nano is based on Ubuntu Linux 18.04 thus this limited the available software packages we could install. The initial configuration of the Jetson had the Desktop Environment change from Gnome to LXQT to have more memory available. OpenCV was updated from the base install to OpenCV 4.6.0. We had to build from source to allow us to take advantage of the CUDA acceleration. PyTorch and TensorFlow were installed through the guides available on Nvidia's forms. Due to our TensorFlow install, we stuck to Python version 3.6.9. Due to being stuck on that version of Python our team must be considerate of what packages we are installing and verify that they work with that specific version. A single member of our team was put in to verify that a requested package was compatible with that version of Python. Once a software milestone was met, the demo was uploaded to GitHub and then downloaded to the Jetson for verification and additional debugging. If any changes were required those changes would be sent back to GitHub and in the practice of important statements, issues would be directly followed. A case where import states affected one of our demos was importing TensorFlow before OpenCV. If imported in this order the operating system would have issues establishing memory to use. The correct order we

found is having OpenCV import first and then TensorFlow. Lastly, the first-time configuration of the GPIO pins on the Jetson Nano required us to make the user group called GPIO and add our user to the said group to utilize them. The GPIO pins on the Jetson Nano have several different modes. The mode we picked is to use the Raspberry Pi standard along with the flag to follow the numbering printed out on the PCB of the development board.

4.2.4 Object Detection

- To navigate through the course for both the drone and the ground robot, we needed their cameras to have object detection capabilities. This was so the cameras we were using on both would be able to see the boxes on the course. The boxes that were used in the course were specifically sized moving boxes from Home Depot, that had two open cuts on opposite sides, as well as red reflective tape around these openings. With these attributes for the boxes in the course, we decided we needed to make a custom object detection model to allow us to only know when we are looking at the specific boxes that will be used in the competition. We also needed our object detection to know what the ground robot looked like. This was so when the drone is flying around the course and needs to land back on the ground robot. The object detection model also allows the drone to recognize itself so that during later rounds when there was a "poison drone" it could avoid the opposing team's drone.
- Throughout the development of our custom object detection model, we explored various methods before landing on one we feel will be most effective. The original implementation of our object detection model relied on pre-trained data from a variety of household objects. But due to the unique appearance of the competition box mentioned above it wasn't viable for our implementation, but a good resource for our design. We were using OpenCV to process video data captured from all the cameras we were working with, so the object detection model had to be compatible with that. First, we attempted to use the YOLO algorithm to design our model. During this, we had an issue compiling the software needed to train the object detection model using YOLO. After some research, we then found TensorFlow would be a viable alternative. To train our model for TensorFlow we used Teachable Machines, which is a machine learning tool provided by Google. Using this we could create different classes that were used for different objects.
- The first iteration of our object detection was created to tell the difference between the boxes in the competition and everything else. Two classes were created to label the objects detected by our model. The box objects class was called "Box" and everything else was labeled under "Neutral". The training was run locally on our computer even though it looked like it was on the hosted website. For this first iteration of training, we used a small sample size of photos for the box and everything else, about 100 for each class. Once the training was done, we were then able to export the model to run in a program. We were given a variety of options to export the model and chose to use TensorFlow and OpenCV with the Keras option. This gave us two files and Python code to use outside of the Teachable Machines website. The two files given to us were the labels used for the classes in a .txt file and a Keras model for object detection as a .h5 file. The Python code given was to run object detection using OpenCV.

- Once we exported the object detection model from Teachable Machines, we were then able to run the code in Python. We first tested this using the cameras on our computer and accessing them using OpenCV. We then could run the code, and in the terminal, the program would print out what object it was seeing and the confidence interval it was predicting it at. Due to there only being two things to differentiate (neutral or box), we could only test the model by putting the box for the competition in the camera's frame. When running the first iteration of this model we were getting high confidence intervals of about 70% when the box was in the frame at a distance. However, the closer the box got to the camera the higher the confidence interval went increasing to around 95% [Fig. 5].
- After we were able to verify that this method would be viable and sufficient for our project, we decided to move forward with implementing it for our object detection model. We verified that the object detection model works with both cameras that we want to have object detection capabilities (the camera on the DJI Ryze Tello Drone and the Intel RealSense Camera). We are also able to run them from the Nvidia Jetson Nano on both cameras as well. When the first model of object detection was trained and created, we were able to export a .tm file, which was the project of trained images. This allowed us to continue using the previous iteration of object training to build a stronger and stronger model. For the next three iterations to give us a total of four iterations of training, we continued to use a variety of photos for the box from different angles. After these iterations, we had a total of around 400 photos for the box and 200 for the neutral class.
- We have also started to train a separate object detection model for the poison drone. As of right now, we want to keep them separated so we can fine-tune each individually before we combine them. The poison drone model was and is being trained the same way as the box model. This is using Teachable Machines and iterations adding more photos. Once the ground bot is complete, we will then train a model for that as well. This will be so the drone can find the ground bot to land on.
- Following the completion of both the box object detection and poison drone and successfully running independently, we were then able to merge them into one model. This was straightforward using Teachable Machines. We took all the photos we had used so far (saved every iteration of photos used) and trained one main model. Once these were all combined, we were able to export the trained model the same way as before and run it in Python. When testing the first iteration of the combined model detection, we ran it on our personal computer and were getting high confidence intervals of 70% or higher when we placed the drone or box in the camera's view [Fig. 5].



```
from keras.models import load_model  # TensorFlow is required for Keras to work
import cv2  # Install opencv-python
import numpy as np
import tensorflow

# Disable scientific notation for clarity
np.set_printoptions(suppress=True)

# Load the model
model = load_model("keras_Model.h5", compile=False)

# Load the labels
```

```
1
Class: Box
Confidence Score: 89 %
1/1 [==============================] - 0s 14ms/step
[[0.01220937 0.881416   0.10637464]]
1
Class: Box
Confidence Score: 88 %
1/1 [==============================] - 0s 14ms/step
[[0.01231987 0.87413913 0.11354098]]
1
Class: Box
Confidence Score: 87 %
1/1 [==============================] - 0s 14ms/step
[[0.01160284 0.876864   0.11153396]]
1
Class: Box
Confidence Score: 88 %
```

Figure 5.

- We will end up having two different models of object detection for this project. One will be designed for the drone, while the other will be used for the ground robot. Though they will be very similar as both are trained for the box in the competition. The object detection model trained for the poison drone will be used for the ground because the ground robot will have to know what the drone looks like to avoid it. This model will also have the capability to detect the ground robot, as mentioned earlier. Whereas the model for the ground robot will only have the ability to know what the box is.

4.2.5 Future Work

- Connect the ODrive to the GPIO pins of the Nvidia Jetson to control motors and receive data on how far the ground robot has moved.
- Create a finite-state machine for the ground robot to operate autonomously. This will have a variety of states depending on what the ground robot will need to accomplish at a given moment during the competition.
- Due to the issue, we have had with the angle of the camera on the drone and not being able to point directly down we will need to transpose a picture of the top of the box. This will give us a better angle for reading the QR code.
- The program will need to know how far the ground robot and drone will have to travel after locating the box. This will be done using depth detection on the Intel RealSense camera.
- When the ground robot is built, add the sensors and the Nvidia Jetson to their correct slot onboard. We will then be able to integrate the sensors with the motors to allow the ground robot to move based on the inputs from the sensors.
- Set up the Orange-Pi so we can use the GPIO pins on it to interact with the ultrasonic sensor. This will allow us to allocate GPIO pins on the Nvidia Jetson to be detected to control the motors through the ODrive. It will also give us more computation power because we will have two different systems running.

4.3 Risks

Risk	Risk Reduction
Python being inefficient	Loops within the Python code will be limited to make the program as fast as possible.
Angle of the drone’s camera	We are going to minimize the risk by transposing images.
Using computer vision on a variety of components	Will not run multiple components for the object detection at once and instead run them sequentially.
Running Python 3.6 for our system	Going to verify that any packages we wish to install is fully compatible with that version of python.
Intel® RealSense™ Camera have blind spots for closeup objects.	Going to use the ultrasonic sensor to detect anything that falls within the blind spots.
The robot will have limited computing power.	Well optimized code will be written for anything running on the robot.
Battery reliability	Extra batteries will be brought for both the robot and drone.
The robot and drone have a break in communication	The robot will finish its last given command. Once that is complete, the robot will stop and attempt to reconnect. If the robot fails to connect after several retries it will have an LED to alert the operator.
Mapping the field might not be possible	Planning to build a test course and devising a different way to accomplish navigation if mapping is not possible.
The robot hitting the entrance of the boxes	The Intel RealSense will help us line up the ground bot to the box entrance and the ultrasonic will stop the bot if seems like impact is going to happen.
The drone landing on the robot incorrectly	The robot, per the prototype design, will take a slight pause before the drone attempts a landing.

4.4 Tasks

1. Send intent to compete to IEEE.
2. Understand the rules of the competition and brainstorm possible solutions.
3. Create an initial design to pick and order the parts necessary.
4. Gain an understanding of drone operation, such as how to use the SDK to control its movements.
5. Learn about physical robot designs.
6. Learn ROS for robotic programming.
7. Test stereo camera being used for the robot.
8. Read QR codes from the drone.
9. Control the drone from a Python file.
10. Look into interfacing with the ultrasonic sensor.
11. Detect objects from the ground robot.
12. Operate the stereo camera in Linux.

13. Create an FSM for the drone.
14. Get a version of object detection working on the drone.
15. Migrate all progress onto the Nvidia Jetson.
16. Have the Nvidia Jetson running Python.
17. The drone flies around the room and can read a QR code.
18. Center objects in the camera’s frame using the object detection and rotation function.
19. Connect the drone and the ground robot using the Nvidia Jetson.
20. Connect the ultrasonic sensor to the Nvidia Jetson.
21. Find the openings of the box using the stereo camera.
22. Connect the stereo camera to the Nvidia Jetson.
23. Save the vector location of a box and QR code.
24. Install all Python packages and standardize a version of Python on Jetson.
25. Have the drone fly around the room and record a video.
26. Fine-tune the object detection model for the box.
27. Connect the Jetson to the Pi Camera.
28. Run the object detection model on the stereo camera.
29. Read a QR code from the ground robot.
30. Send data from the drone camera to the Nvidia Jetson.
31. Practice controlling motors from a robot supplied by our sponsor.
32. Measure the side of a box using the stereo camera.
33. Create an object detection model for the poison drone.
34. Have the drone flying and doing flips with the camera on.
35. Get the distance of an object that is in front of the stereo camera.
36. Assemble the robot.
37. Use GPIOs on the practice robot to read data from the sensors.
38. Transpose of a picture of the top of the box.
39. Connect ODrive to the Jetson.
40. Integrate the sensors onto the ground robot.
41. Create a top-level program to run everything.
42. Control motors through the ODrive.
43. Map all boxes in the room using the drone.
44. Do final testing and debugging.
45. Compete

4.5 Schedule

Tasks	Dates
1. Send intent to compete to IEEE	10/2
2. Understand the rules of the competition and brainstorm possible solutions.	10/2-10/15
3. Create an initial design to pick and order the parts necessary.	10/16-10/29

4. Gain an understanding of drone operation, such as how to use the SDK to control its movements	10/31-11/6
5. Learn about physical robot designs.	11/14-11/18
6. Learn ROS for robotic programming.	11/21-12/2
7. Test the stereo camera being used for the robot.	12/5-12/9
8. Read QR codes from the drone.	1/26-1/29
9. Control the drone from a Python file.	1/29
10. Look into interfacing with the ultrasonic sensor.	1/27-2/3
11. Detect objects from the ground robot.	1/28-2/3
12. Operate the stereo camera in Linux.	1/29-2/3
13. Create an FSM for the drone.	1/29-2/3
14. Get a version of object detection working on the drone.	1/31-2/6
15. Migrate all progress onto the Nvidia Jetson.	2/7
16. Have the Nvidia Jetson running Python.	2/3-2/10
17. The drone flies around the room and can read a QR code.	1/29-2/12
18. Center objects in the camera's frame using object detection and rotation.	2/3-2/15
19. Connect the drone and the ground robot using the Nvidia Jetson.	2/6-2/17
20. Connect the ultrasonic sensor to the Nvidia Jetson.	2/8-2/22
21. Find the openings of the box using the stereo camera.	2/14-2/22
22. Connect the stereo camera to the Nvidia Jetson.	2/9-2/23
23. Save the vector location of a box and QR code.	2/10-2/24
24. Install all Python packages and standardize a version of Python on Jetson.	2/12-2/24

25. Have the drone fly around the room and record a video.	2/15-2/24
26. Fine-tune the object detection model for the box.	2/16-3/2
27. Connect the Jetson to the Pi Camera.	2/16-3/3
28. Run the object detection model on the stereo camera.	2/17-3/3
29. Read a QR code from the ground robot.	2/17-3/3
30. Send data from the drone camera to the Nvidia Jetson.	2/20-3/3
31. Practice controlling motors from a robot supplied by our sponsor.	2/24-3/3
32. Measure the side of a box using the stereo camera.	2/27-3/10
33. Create an object detection model for the poison drone.	2/28-3/14
34. Have the drone flying and doing flips with the camera on.	3/1-3/15
35. Get the distance of an object that is in front of the stereo camera.	3/2-3/16
36. Assemble the robot.	3/3-3/17
37. Use GPIOs on the practice robot to read data from the sensors.	3/5-3/17
38. Transpose of a picture of the top of the box.	3/6-3/17
39. Connect ODrive to the Jetson.	3/17-3/31
40. Integrate the sensors onto the ground robot.	3/27-4/3
41. Create a top-level program to run everything.	3/27-4/10
42. Control motors through the ODrive.	3/29-4/12
43. Map all boxes in the room using the drone.	3/29-4/12
44. Do final testing and debugging.	4/10-4/21
45. Compete	4/22

4.6 Deliverables

- Design Document: Contains a sketch of what the overall robot design will look like. Key design details with the reasoning behind them, a list of hardware that will make up the robot itself. Lastly, this will include information on what software and programming languages we are planning to use.
- Website: Giving an overview of the entire project. Links to our GitHub and Trello board to give a casual observer an idea of the scope of the project. The site should feature images of the project in action and explanations of key parts.
- Code: There will be Python code for anything the drone is doing due to the availability of an official SDK. The robot's main logic will mostly be in Python. The robot will have code for movement, collision detection, and object detection. Movement commands will be figured out from information given by the drone. Each waypoint will have a value that can direct the robot to the next segment.
- Project Reports: Giving an overview of our planning and thought process when it came to the pre-production concerning the project shown in the earlier reports from Capstone I. The latter half of the reports shows what ideas changed along with solutions we needed to implement to ship on time from Capstone II.

4.7 Use Cases

- Due to our project being designed for a specific competition, the overall project does not have many use cases. Though certain parts could be applicable in other cases.
- Autonomous robotics have become more prevalent as time goes on. These can be used for a variety of tasks ranging from factory and warehouse operations to space exploration.
- Drone mapping used in our project can be used to map a variety of locations like national parks, the Moon, and Mars.
- Another use case would be used in disaster relief efforts. An autonomous robot and drone could work in tandem for mapping out sites and searching for survivors. The data that is collected in operation could also be used in later investigations.

5.0 Key Personnel

Nicholas Brown – Brown is a senior Computer Engineering major in the Computer Science and Computer Engineering Department at the University of Arkansas. He has completed relevant courses such as Digital Design, Computer Organization, System Synthesis and Modeling, Software Engineering, and Programming Paradigms. He has experience before his college career in robotics and computer system design as a research assistant. Brown will be responsible for the architecture of the computer on board the ground robot, as well as the communication between the drone and the ground robot.

Callum Bruton – Bruton is a senior Computer Engineering major in the Computer Science and Computer Engineering Department at the University of Arkansas. He has completed relevant courses such as Computer Architecture, System Synthesis and Modeling, Software Engineering, Programming Paradigms, and Computer Organization. Bruton will be responsible for communication between the ground robot and the drone, as well as working on the computer processing on board the ground robot.

Jase Cornett – Cornett is a senior Computer Engineering major in the Computer Science and Computer Engineering Department at the University of Arkansas. He has completed relevant courses such as Computer Architecture, Embedded Systems, GPU Programming, Programming Paradigms, and Software Engineering. He has experience with an Automation Engineering internship at Texas Instruments. Cornett will be responsible for sensor data processing and the movement of the ground robot.

Austin Flynn – Flynn is a senior Computer Science major in the Computer Science and Computer Engineering Department at the University of Arkansas. He has completed relevant courses such as Programming Paradigms, Software Engineering, and Computer Networks. He will be responsible for programming the autonomous movement of the drone and robot using computer vision and various sensors and the communication between them.

Stephanie Stock – Stock is a senior Computer Engineering major in the Computer Science and Computer Engineering department at the University of Arkansas. She has completed relevant courses such as System Synthesis and Modeling, Software Engineering, Computer Architecture, Programming Paradigms, and Embedded Systems. She has experience as an undergraduate research assistant. Stock will be responsible for drone communications with the ground bot and computer visions.

Dr. Jeff Dix – Dix is an Assistant Professor in the Department of Electrical Engineering at the University of Arkansas. He is a co-advisor of the Robotics Interdisciplinary Organization of Teams, and he will be overseeing the progress made by the RIOT participants involved.

Mr. Robert Saunders – Saunders is the Assistant Department Head of the Department of Electrical Engineering at the University of Arkansas. He is a co-advisor of the Robotics Interdisciplinary Organization of Teams and the sponsor for this Capstone project. We will be reporting our progress and justification for design choices to him.

6.0 Facilities and Equipment

The rules of the competition require the use of a Ryze Tello model drone [5]. The official specification of the drone states its detachable battery can power the drone for approximately 13 minutes of flight time and it can take 720p video at 30 frames per second with the equipped camera. It also has a built-in barometer, range finder, and 2.4 GHZ 802.11n Wi-Fi [7].

Through RIOT, we will have access to a variety of equipment tools, that will allow us to design the parts of the robot. This includes, but is not limited to a drill press, drill, and 3D printer.

Facilities that we will be utilizing for the project will be across the University of Arkansas Campus. These rooms, the Senior Design Lab, where we will design parts of the robots, the ELEG Lounge in Bell, where we meet with RIOT, and the Acxiom Lab where we have our Capstone group meetings, are where we will spend the most time working on our project.

7.0 References

[1] A Beginner’s Guide to Drone Mapping Software, https://www.dronepilotgroundschool.com/drone-mapping-software/

[2] Bradshaw K., Ross L., “Fiducial Marker Navigation for Mobile Robots”, 2012

[3] J. Mišeikis, K. Glette, O. J. Elle and J. Torresen, "Multi 3D camera mapping for predictive and reflexive robot manipulator trajectory estimation", 2016 IEEE Symposium Series on Computational Intelligence (SSCI), 2016

[4] Krishnamurthi R, Kumar A, Gopinathan D, Nayyar A, Qureshi B. “An Overview of IoT Sensor Data Processing, Fusion, and Analysis Techniques”, MDPI, 2020

[5] IEEE, “2023 IEEE Region 5 Annual Conference Rules for a Student Robotics Competition”, 2022

[6] Map a Mars Rover Driving Route, https://www.jpl.nasa.gov/edu/learn/project/map-a-mars-rover-driving-route/

[7] Tello *Specs*, https://www.ryzerobotics.com/tello/specs