

Vision Navigation Drone

Group 3:

Cannen Carpenter (CPE)

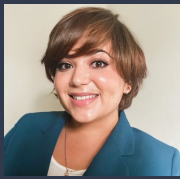
Derek Murdza (CPE)

Jazmine Roman (EE)

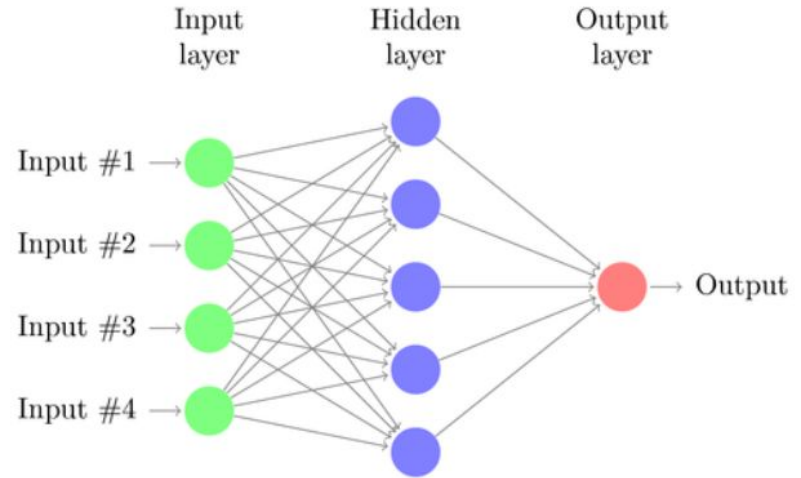
Kevin Nilsen (PSE)



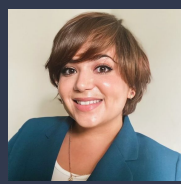
Introduction



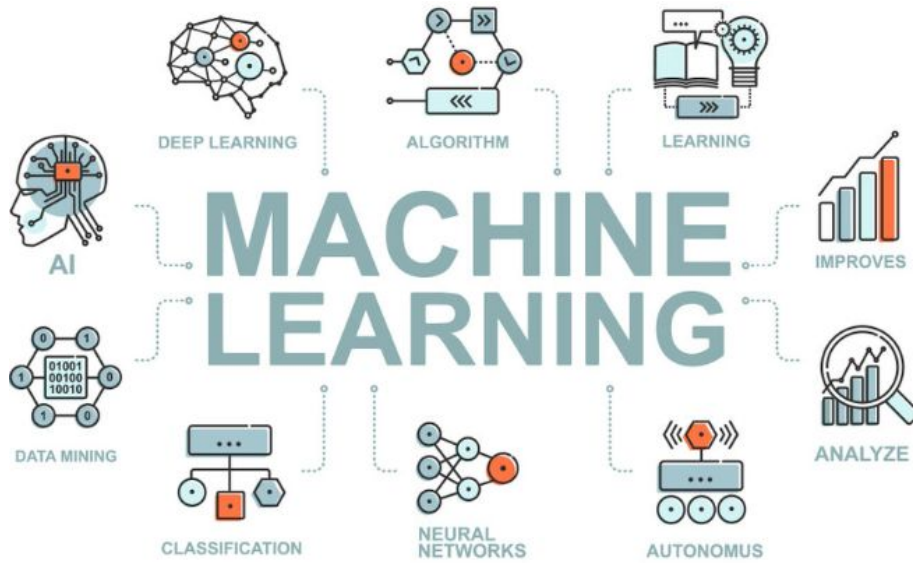
Jazmine Roman (EE)



Motivation



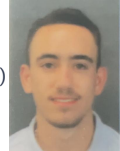
Jazmine Roman (EE)



Goals and Objectives



Derek Murdza (CpE)



Basic Goals:

- The drone must recognize three predetermined objects
- The drone will be able to navigate around these object according to the operator's instructions
- The drone must be able to detect objects directly next to the drone up to 0.25 m

Advanced Goals:

- The drone should complete its instructions within one minute
- The drone's environment detection system will be able to detect exact distances up to 0.25 m

Stretch Goals:

- The drone can complete its instructions within 30 seconds

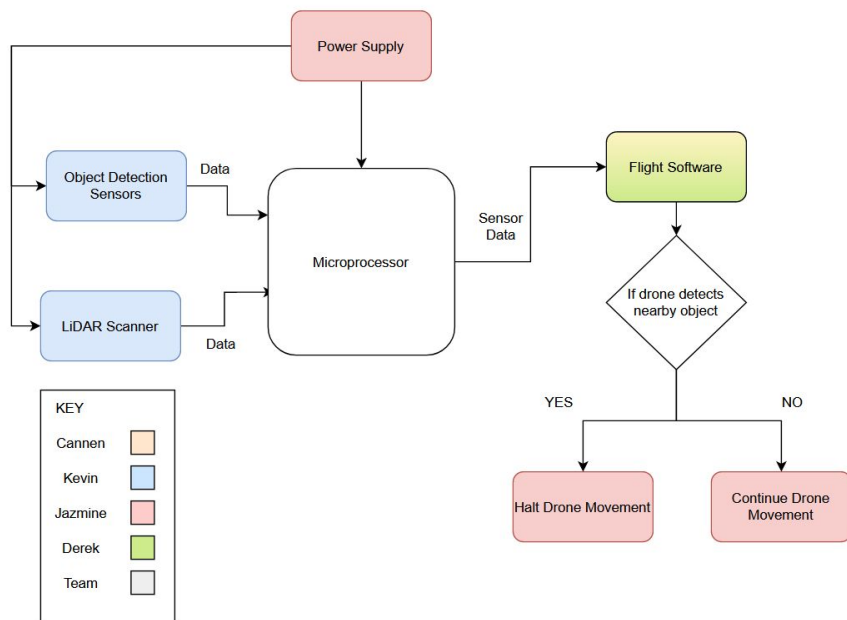


Block Diagrams

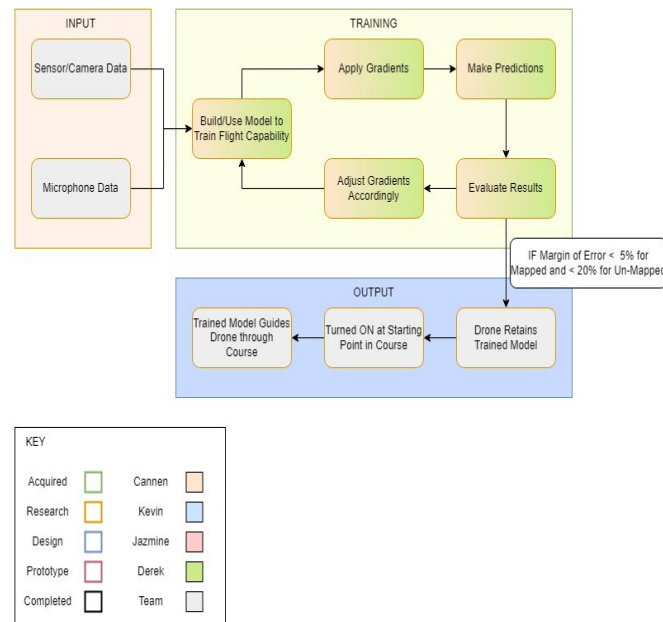


Kevin Nilsen (PSE)

Hardware Diagram



Software Training Diagram





Requirements and Specifications



Kevin Nilsen (PSE)

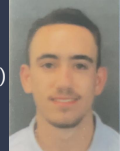
Component	Parameter	Specification
Drone	Drone flight	Should be able to hover and move in all 3 dimensions
Drone	Drone light	Should withstand small wind gusts
Drone	Drone Safety	Drone should prioritize safety of itself and others (i.e. have propeller guards, stop when object comes too close, etc)
Environment Sensors	Obstacle avoidance	Should detect objects in environment at least 0.3 m away
Object Detection	Vision Accuracy	Should detect target objects with >80% accuracy
Object Detection	Vision Distance	Should identify objects that are within a range of 8 m

Training Reward System



- Positive and negative outcomes are expected as this influences efficient training as software is implemented and modified
- With more successful tests, the drone will be more capable of producing positive results, and poor results will show the need for a change in whatever specific parameters are referenced:
 1. **Collision Avoidance**
 2. **Time Constraints**
 3. **Overall Success Rate**





Module Comparison

- **Raspberry Pi 3B/4B** was selected for use on the drone
 - 64-bit vs 8-bit
 - Higher RAM and Clock Speed
- Selected **PiCam** due to selection of Raspberry Pi module

Specification	Arduino Uno	Raspberry Pi 3B
CPU Type	8-bit Microcontroller	64-bit Microprocessor
Storage	32 KB Flash Drive	SD Card Dependent
Memory	2 KB RAM	1 GB RAM
Speed	16 MHz	1.2 GHz
USB Ports	1	4



Drone Flight



Cannen Carpenter (CpE)

- Flight is handled through the flight controller and is built with the PX4 Autopilot software
- The software for this portion will be processed on our Raspberry Pi 4B
 - Sensor data will be sent to this application, from which the drone will be able to account for collisions

Autopilot System	PX4	ArduPilot
Code	Open-Source	Open-Source
Quadcopter Support	TRUE	TRUE
Complexity	Medium	High
Specialization	Low	High
Pro	Simpler than ArduPilot	Many more possibilities with the added control benefits
Con	Less room for detailed control over the navigation	More control over the system may cause complexity to increase rapidly

Machine Learning



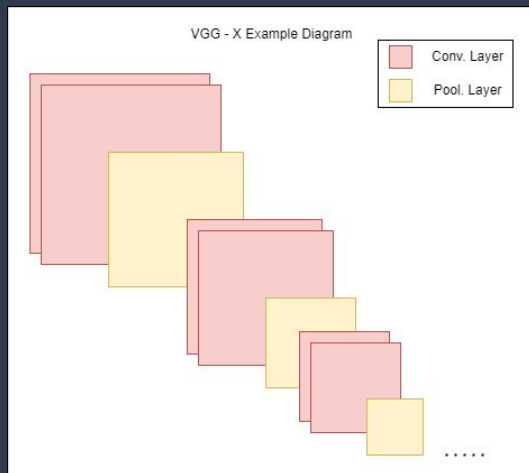
Cannen Carpenter (CpE)

Training Models	VGG-11	VGG-13	VGG-16	VGG-19
Convolutional Layers	11	13	16	19
Activation Function (most used)	ReLU	ReLU	ReLU	ReLU
Properties	Simple, low training time, unable to handle fairly complex analysis	Handles higher complexity than VGG-11, but not good enough for our needs	Still fairly simple, and can handle object recognition at the level we require	Unnecessary for our needs, longer training times would not benefit our case enough to justify its use

- For the machine learning aspect of our project, we have an open source Visual Geometry Group CNN (VGG)
- We will be making alterations to the code to allow it to make predictions on live video footage
- Predictions are then sent to the navigation loop



Object Recognition



Cannen Carpenter (CpE)

- Our trained VGG model will be utilized in unison with the BBT0 Mini Camera Module for our Raspberry Pi
- The drone camera will feed high resolution video through our model, a prediction will be made, fed to the navigation loop to be processed
- We will start with images taken of our objects and move to pre-recorded footage, and then finally to live video





Choosing a Programming Language

Framework	PyTorch	TensorFlow
Language	Python	Python
Founder	Facebook	Google
Method of Computation	Dynamic - computational graphs are created when necessary	Static - computational graphs are statically defined when compiled
Distributed Training	Built-in	Manual Set-Up
Pro	Stronger Community Presence and is more Python-like, making it easier to learn	Better Visualization of Training and Larger Libraries
Con	Little to No Visualization of Data Training, can be more difficult to identify issues	Does NOT FEEL like Python, so it takes more time to learn

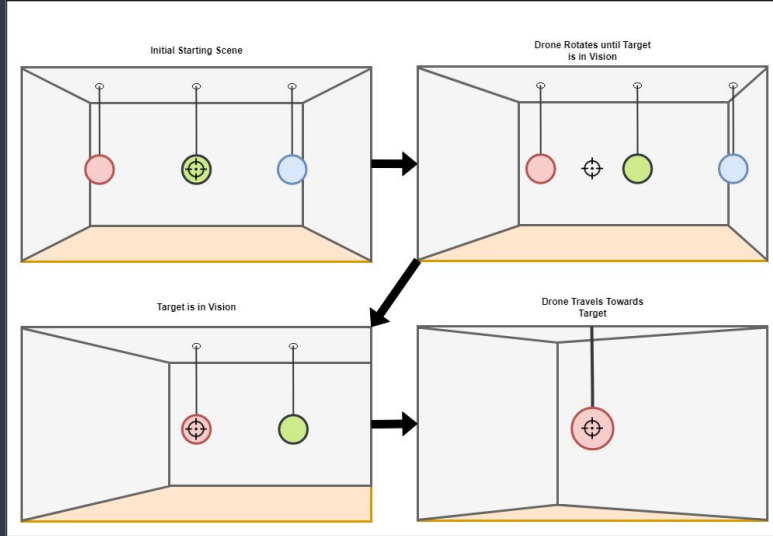


Cannen Carpenter (CpE)

- Our CNN was created in Python by Ben Trevett, utilizing the Pytorch framework
- Latest build of Pytorch will be used (1.13.1), paired with Conda, and CUDA version 11.7
- Building the navigation software will require coding in C
- Data transfer between the training model and the drone components will most likely be done by reading/writing output files



Navigation



Cannan Carpenter (CpE)

- Drone navigation is processed through the flight controller included with the drone kit
- After our model makes a prediction and sensor data is collected, navigation route is decided
- Drone will return to starting point and descend to the floor after travelling to its last destination



Environment Detection



Kevin Nilsen (PSE)

Method	Pros	Cons	Comments
Object Detection	<ul style="list-style-type: none">• Cheap• Scalable	<ul style="list-style-type: none">• Short range• Imprecise• Noise sensitive	Simple and easy to implement. Multiple sensors can be used
Distance Detection	<ul style="list-style-type: none">• More accurate	<ul style="list-style-type: none">• Short range• More complex	Likely provides little benefit over object detection
2D LiDAR Scanning	<ul style="list-style-type: none">• Highly Accurate• Long Range	<ul style="list-style-type: none">• Inaccurate for short distances• Complex driving circuitry	Worth exploring due to far more accurate environment data

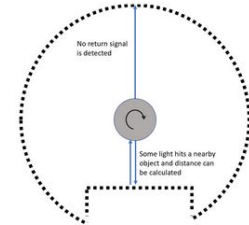
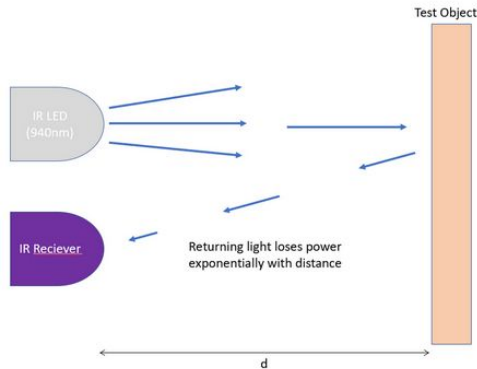


Environment Detection



Kevin Nilsen (PSE)

- The group decided to do a mixture of these methods
- Object detection sensors will be set around the drone
- A simple two dimensional scanner will be mapping the local environment continuously



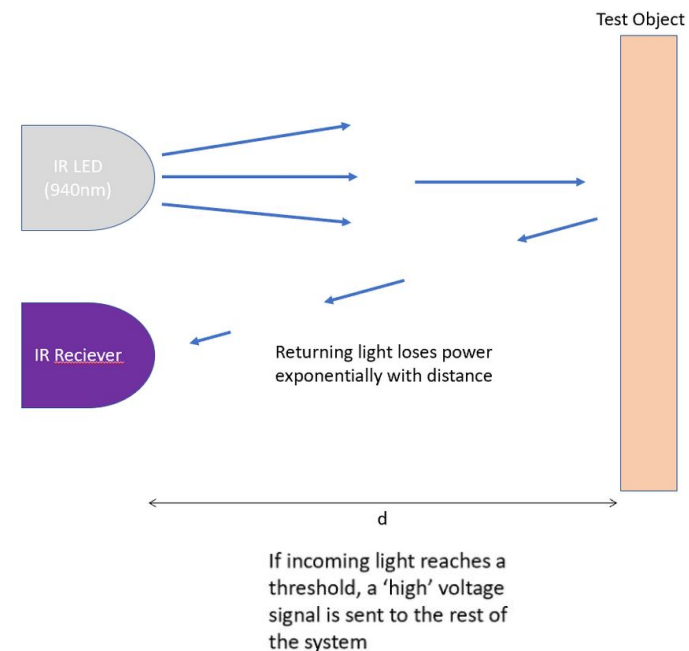


Object Detection



Kevin Nilsen (PSE)

- Whenever the receiver detects a certain amount of light returning, a HIGH signal is sent to the microcontroller, alerting the drone of a nearby hazard
- This sensor's small size allows them to be more flexible in their integration
- Four of these will be placed near the drone's propellers
- Once the Drone gets this signal, it will cease movement



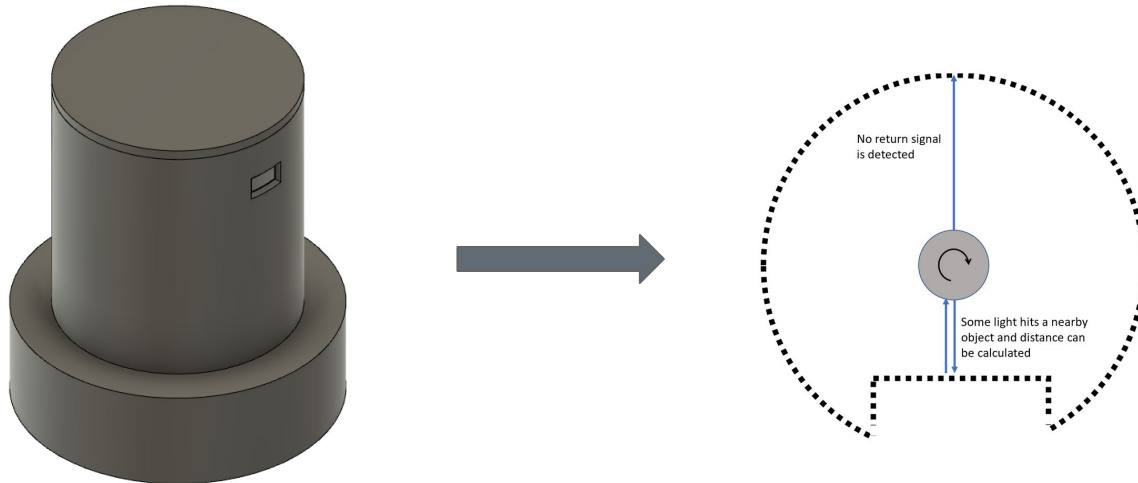


LiDAR



Kevin Nilsen (PSE)

- This scanning system will be mounted on a rotating stage on the bottom of the drone
- The distance sensor would fire continuously, creating a point cloud of the area directly around the drone
- By having two sensors on either side of the rotating module, we can have a continuously updating map for the drone to navigate with



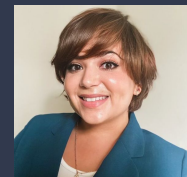
LiDAR



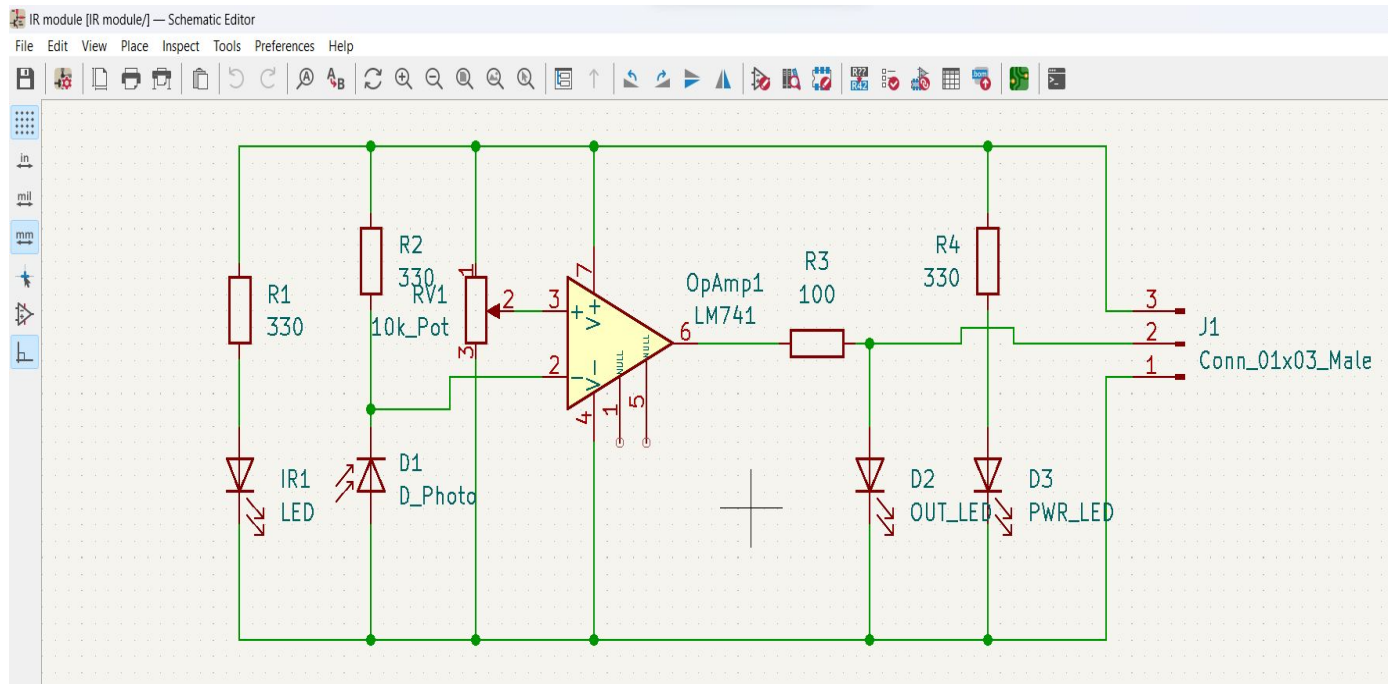
Kevin Nilsen (PSE)

- The direct time-of-flight sensing for LiDAR in this application requires a circuit that may need to drive over 1 GHz
- There has been difficulty driving our circuits a circuit this fast
 - This often requires solid state electronics on a PCB, so it is difficult to prototype efficiently
- It is not a core goal of the projects, but it would still be a useful addition that could improve location accuracy and safety
- A higher quality function generator IC as well along with an efficient peak detector circuit are the planned methods of creation

IR Sensor – PCB design



Jazmine Roman (EE)

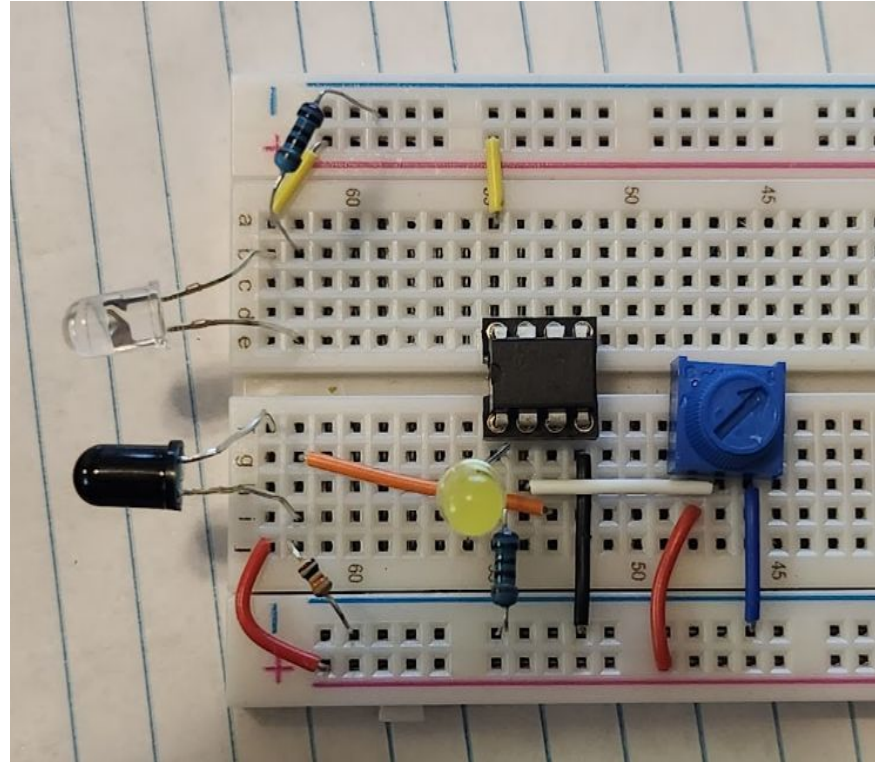
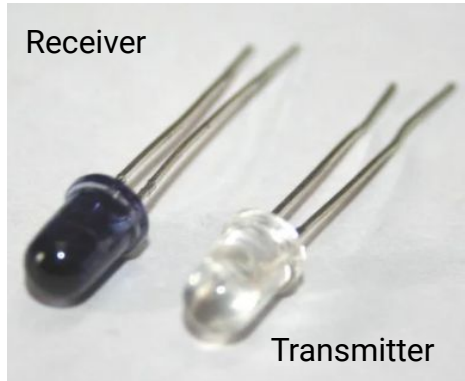




IR Sensor – PCB design



Jazmine Roman (EE)

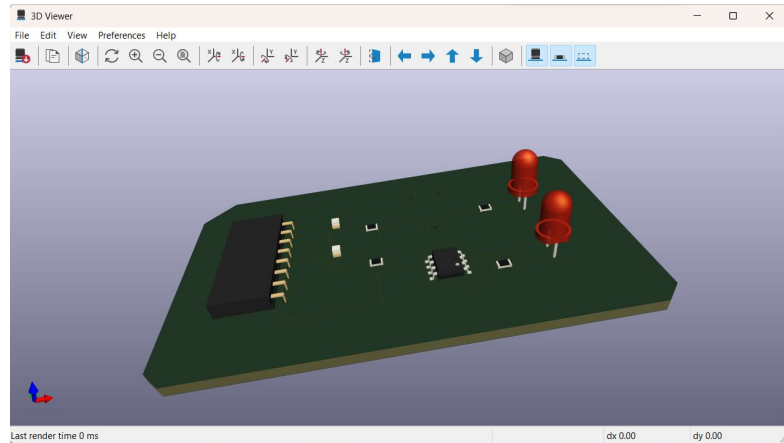
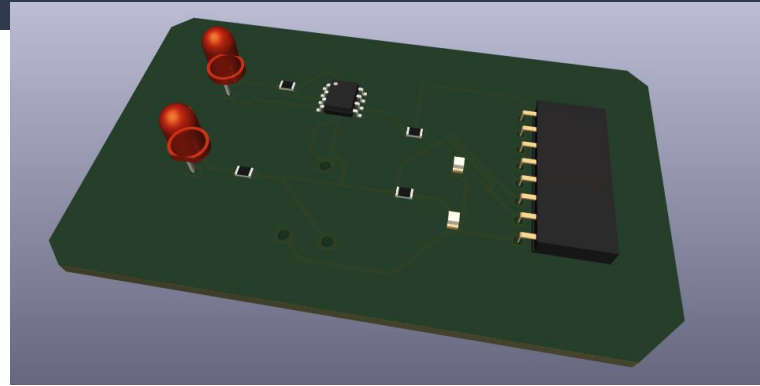
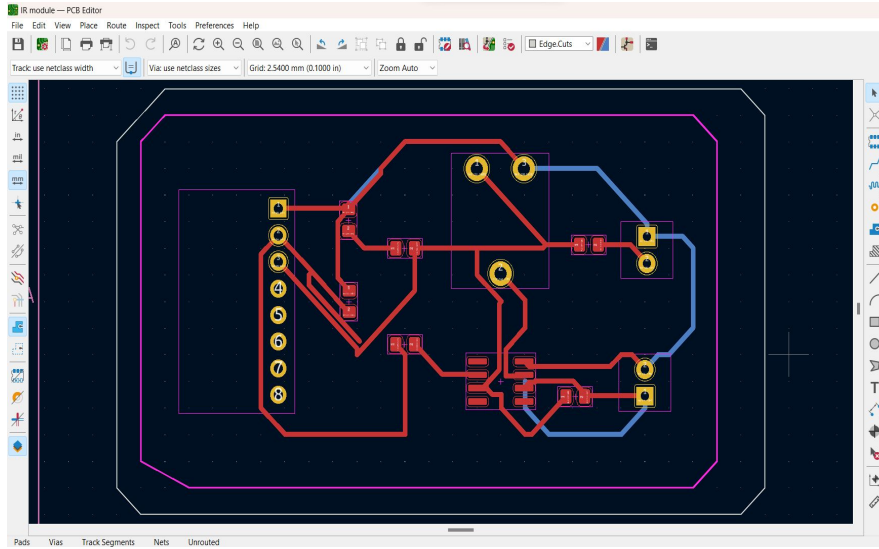


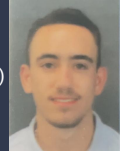


IR Sensor – PCB design



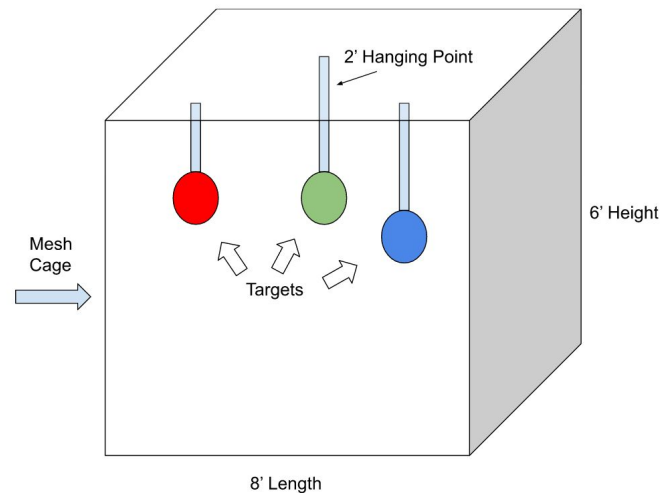
Jazmine Roman (EE)





Testing Environment

- The drone will be tested using a pre-made environment comprised of a mesh netting and hanging objects to be used as targets
- The netting is utilized as a safety precaution for both conductors and the drone itself
- The hanging objects will feature distinct colors to be used for image detection



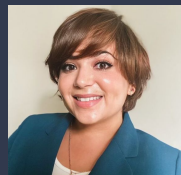
Technology Comparison – Training Models



Canner Carpenter (CpE)



Training Models	Microsoft Sample	GoogLeNet	LeNet	AlexNet	Pre-trained VGG	ResNet
Type of Network	CNN	CNN	CNN	CNN	CNN	CNN
Activation Function most commonly used	ReLU	ReLU	ReLU	ReLU	ReLU	ReLU
Loss Function most commonly used	Cross Entropy	Cross Entropy	Cross Entropy	Cross Entropy	Cross Entropy	Cross Entropy
DataSet Trained (source-results)	CIFAR10, 50,000 training images and 10,000 testing images	ImageNet, 150,000+ images from 1000 categories	MNIST, set of handwritten letters and numbers	ImageNet & CIFAR10	ImageNet & CIFAR10	CUB 200-2011, very complex set of birds
Accuracy (source-results)	70% (Top-1, short training period)	89%(Top-1) 93% (Top-5)	Lowest (85%) highest (99%)	76% (Top-1)	94%(Top-1)	80% (Top-1) 95% (Top-5)
Implementation Difficulty	Low - tutorial	Medium - only code	Low - tutorial	Low- tutorial	Low - tutorial	Medium - tutorial





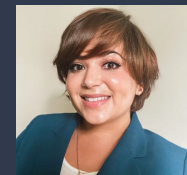
Jazmine Roman (EE)

Major Components Comparison

Motors	Brushless	-Very durable	- Complex and driven by a specialized circuit	Brushed	-2 leads only + & -	-Not durable
Flight controller	PiHawk4	-High performance -Customizable -Stable -User-friendly	-High cost -Complex	Ardupilot	-Open-source -Versatile -Cost-effective	-Constant software updates -Limited documentation
Companion Computer	Single-Board Computer	-High process power -Customizable -Cost-effective	-Power consumer -Large	FPGA Board	-High Performance -Low latency -Power efficient	-High cost -Difficult integration with flight controller systems
Microcontroller	RaspberryPi Pico	-Low cost -High Performance -Versatile	-Limited memory	ARM Cortex-M series	-Low power consumption -Easy to program	-Limited memory and storage -Low processing power
Quadcopter frame	X Frame		-Aerodynamic -Lightest frame	H Frame		-Simple to build -Heaviest frame



Bill of Material



Jazmine Roman (EE)

Part	Market Price
Frame legs	\$25
High Landing Gear legs	\$20
Propellers w/fastener	\$23
Propeller guards	\$20
Electric speed controller	\$40
Power Distribution board	\$52
UBEC power board module	\$16

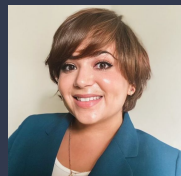
Part	Market Price
PiHawk flight controller	\$189
Raspberry Pi4	\$235
Picam	\$10
4G TF Memory Card w/wires	\$12
16G VHS-1 w/reader	\$35
Brushless motors	\$62
Vibration damper	\$8

Part	Market Price
GPS module	\$38
GPS mount	\$9
Pinspice wires	\$10
AC/DC 12V adapter battery charger	\$12
LiPro Balance Charger	\$32
Power 3000mAh Lipo3S 11.1V 50C	\$40
XT60 Protect	\$8

Part	Market Price
Screws and bolts	\$7
Power module v1.0	\$17
Radio Telemetry	\$84
Zip ties	\$5
Sticky fasteners	\$19
USB to Type C	\$8
USB to Micro Connector	\$8
AA battery pack	\$15

Part	Market Price
Velcro fasteners	\$9
Fireproof case	\$30
RC controller	\$53
Pico microcontroller	\$22
IR PCB board	TBD
LIDAR PCB board	TBD
Total without kit	\$1,116
Total with kit	\$849

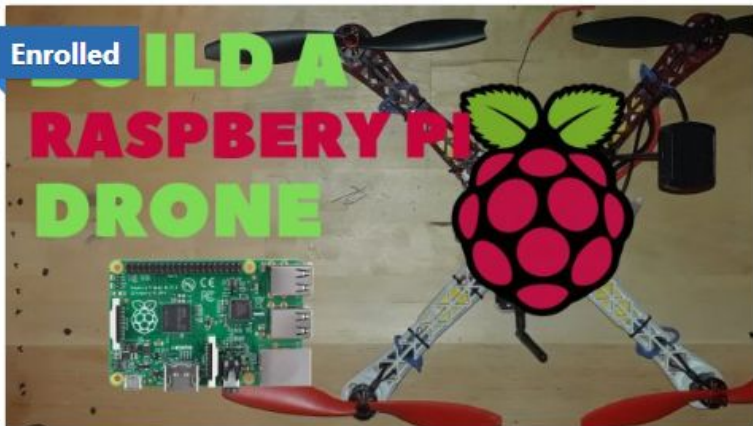
Building the Drone



Jazmine Roman (EE)

Enrolled

**BUILD A
RASPBERRY PI
DRONE**



Drone Engineering 101 Camp

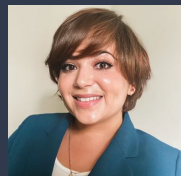
Enrolled



**21 Part Drone Building Video
Manual**



Work Distribution



Jazmine Roman (EE)

Software Team		Hardware Team	
Cannen Carpenter	<ul style="list-style-type: none">• Machine Learning implementation• Embedded systems• Flight navigation	Jazmine Roman	<ul style="list-style-type: none">• PCB KiCAD design• Parts• Drone building
Derek Murdza	<ul style="list-style-type: none">• PiHawk4 (PX4) flight simulator• Programming research	Kevin Nilsen	<ul style="list-style-type: none">• IR module design• LiDAR design

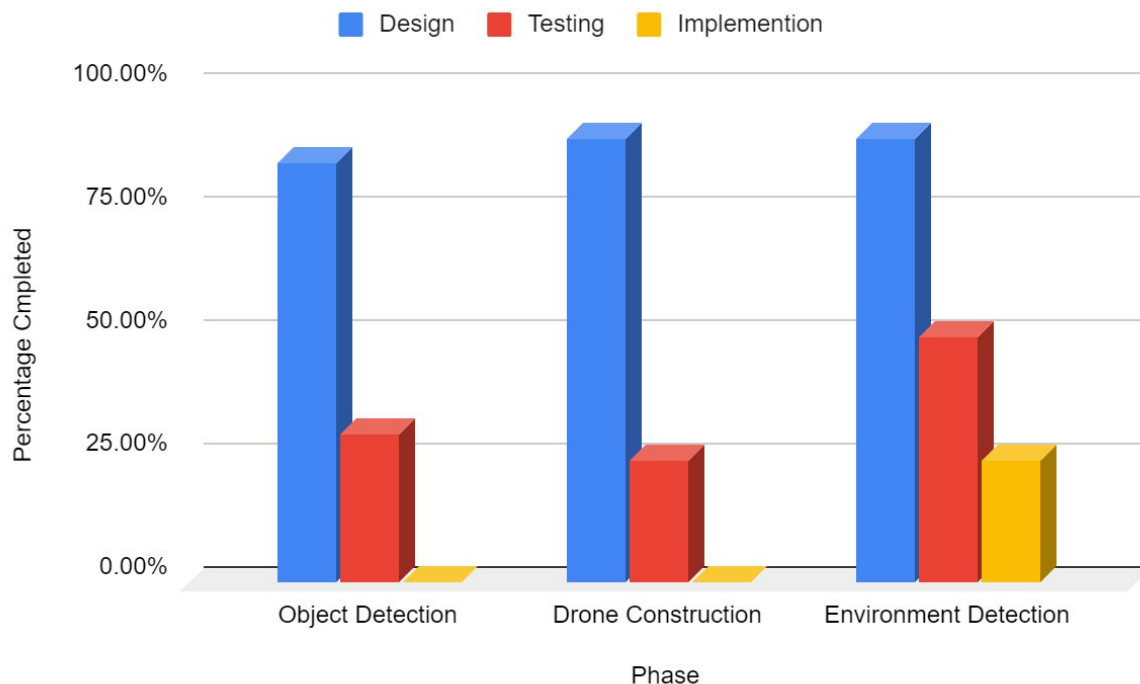




Progress



Cannen Carpenter (CpE)



What's Left

Model predictions on self-taken images, video footage, and live video

Drone parts have all arrived
Need to finish enrolled drone boot camps and begin constructing the drone

Needs to be integrated with the PCB and fine tuned
Implementing LiDAR to PCB

Finish PCB, construct drone, continue programming AI

Identified Issues

- The original sponsor for this project retired from UCF mid-semester which set some drawbacks in terms of design, implementation, and funding
- Most of the group is new to implementing software to an electronic device, such as a drone
- Driving speed of current prototypes are too low to create a full LiDAR scanner



Proposed / Implemented Solutions



- Received sponsorship from the ECE department
- Modified project goals from Visual Language to Visual Navigation
 - Simpler, yet equally efficient design
- Excessive investigation was conducted for VN software integration
 - Identifying functions of the drone and utilizing ideas from recently produced projects helped make integration much smoother and efficient



Conclusion



Cannen Carpenter (CpE)

- Machine Learning model is being altered and implemented
 - Testing with still images is to be done, then we are to move to the next step
- Drone construction is to begin soon, as parts have been received
 - PCB has not been received yet, but object recognition can still be tested without sensor data
- Once drone is constructed, the navigation software can be built and used in potential simulations
 - PX4 software application
- LiDAR/IR implementation will allow object detection for collision/navigation purposes
 - Will feed data to the PX4 software application, allowing the drone to react to objects in its path
- Overall, communication between the systems within this project will prove to be the most significant part, as if one falters, the rest may suffer great consequences

