DRONE PACKAGE DELIVERY PROJECT

Mid-way presentation

Gal Malka
Gal Peretz

ON-THE-JOB TRAINING

- We had two sessions with the previous students of this project (Sergey and Alex):
  - Learning about the drone, its parts, how to work with it and what we should be aware of.
  - Going over the code itself and understanding its design.

- Installing the framework and simulators, and testing the system E2E on the simulator.

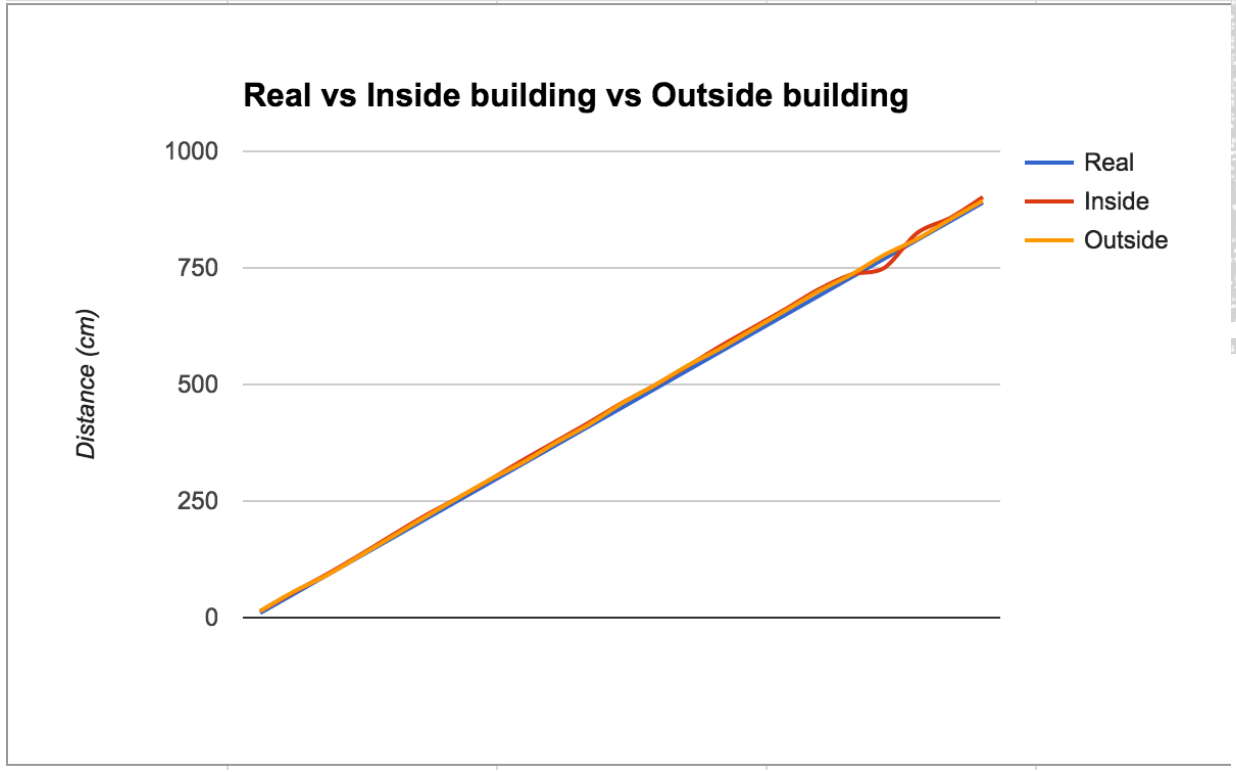
UNDERSTAND THE GOAL AND CREATE LONG TERM TASKS

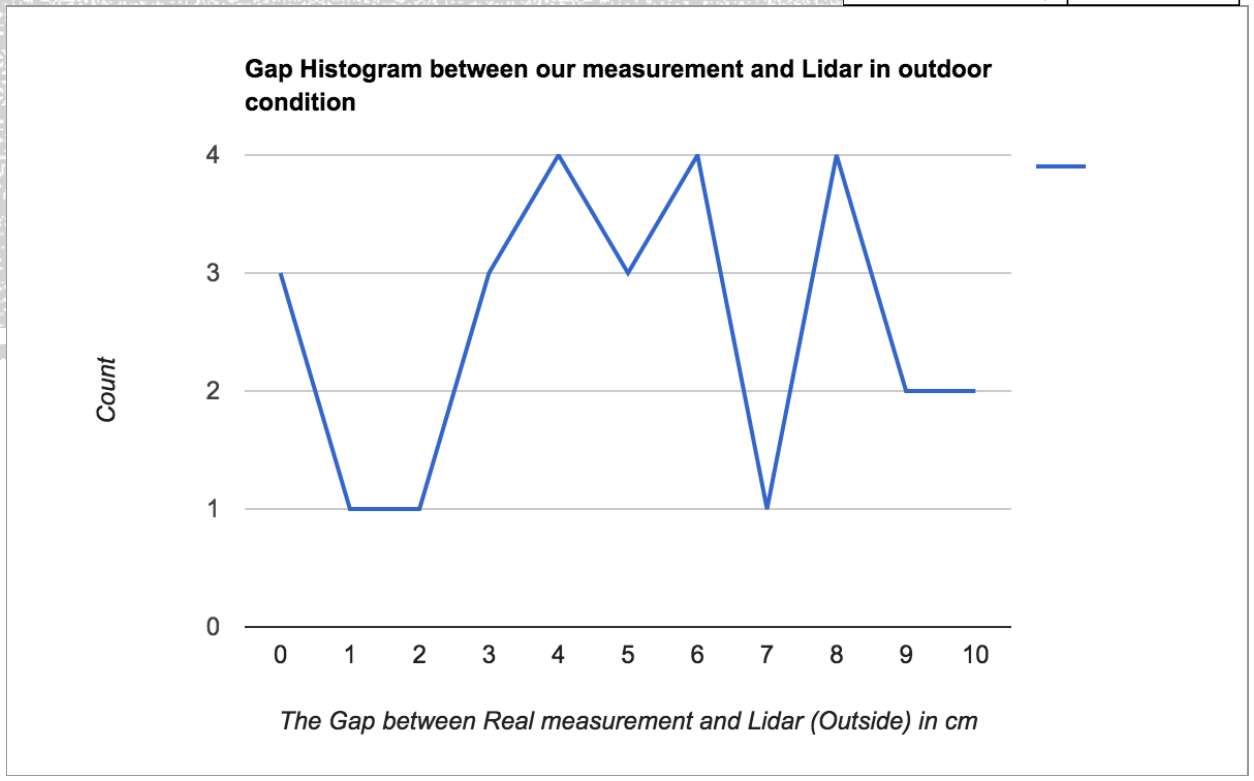
- The Goal of the project is to create a program that will control the drone autonomously – The mission of the drone is to fly to a specific location, pick up a package, deliver it to a new location, and return to home.
- After talking with Ohad, we decided together that Python is not suitable for controlling a drone, and running image processing algorithms. We decided to allocate some of the time of the project to create an infrastructure in C++ that will serve us, and the future algorithms \ missions of the next projects.
- The next step is to implement the mission itself.

Lidar`s measurement

- We use the Lidar to calculate the height of the drone because the pixels the target takes in a frame is a function of height and the FPV of the drone.
- We tested the Lidar`s accuracy in the lab and outside lab vs real measurement of the distance
- After the test we concluded that we can use the Lidar output in from 10cm to 1000cm

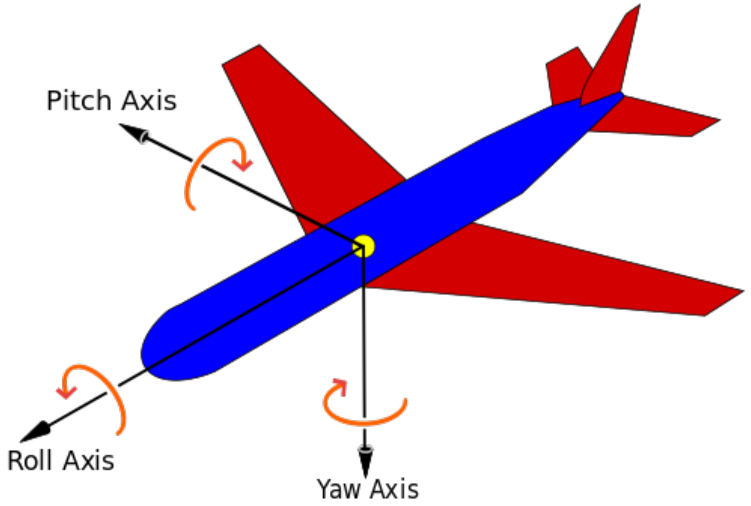
Gap Real vs Outside	
AVG(mean) (cm)	5.142857143
Variance (cm)	8.941798942
Standard deviation (cr	2.99028409



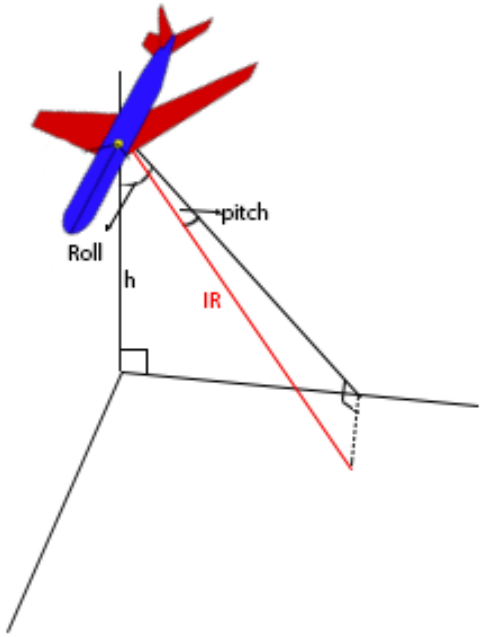


LIDAR AS A HEIGHT MEASUREMENT

- We use the Lidar to get an accurate estimation of the drone's height .
- We needed to find a way to estimate the drone height with the Lidar during the flight while the drone turns.



ESTIMATE THE HEIGHT BASE ON THE LIDAR



- So the equation is:
- Height = (cos(Pitch) * IR_distance) * cos(Roll)

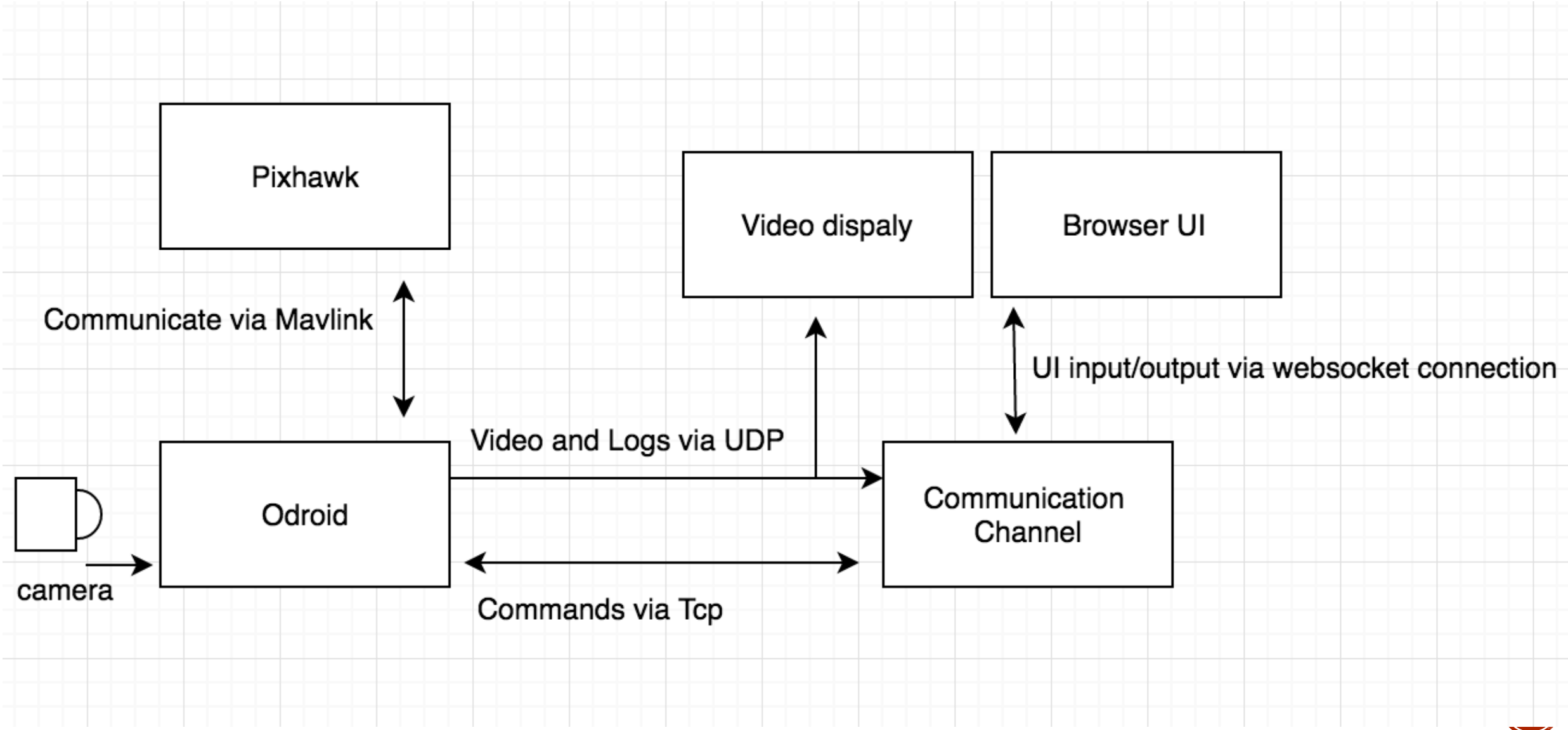
THE INFRASTRUCTURE - GENERAL

- Our system is written mainly in C++, and only specific components are written in python. We are combining the two languages because the drone SDK is written in python.
- The system is generic. Adding new missions should be simple.
- The system has wide variety of options for debugging the mission in real time and after the run.
- The system is multi-processor and utilizing all of the Odroid resources to reach maximum performance.

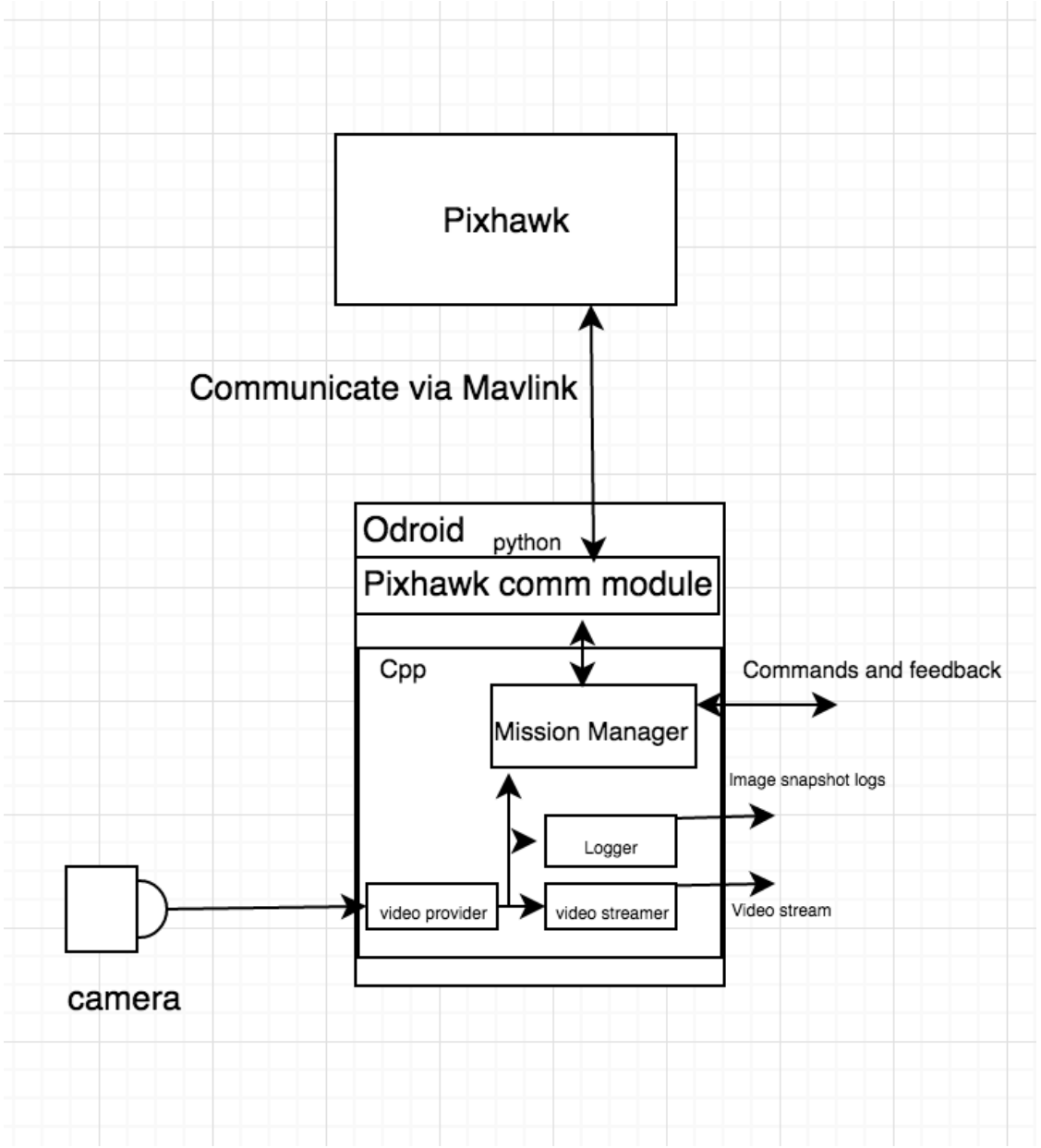
THE INFRASTRUCTURE – THE DIFFICULTIES

- Linkage C++ with Python – we had to find a way to run C++ code from Python, as the drone SDK is written in Python. The solution we chose is a library called Boost::Python which enables to run C++ and Python code together.
- Python is designed to be single-threaded. In order to run our C++ code on multi threads, we had to design a sub system that is responsible for that.
- The previous students had difficulties to stream video from the Odroid to the ground control stations. We wrote UDP endpoints in C++ and succeeded to stream the video in good quality.

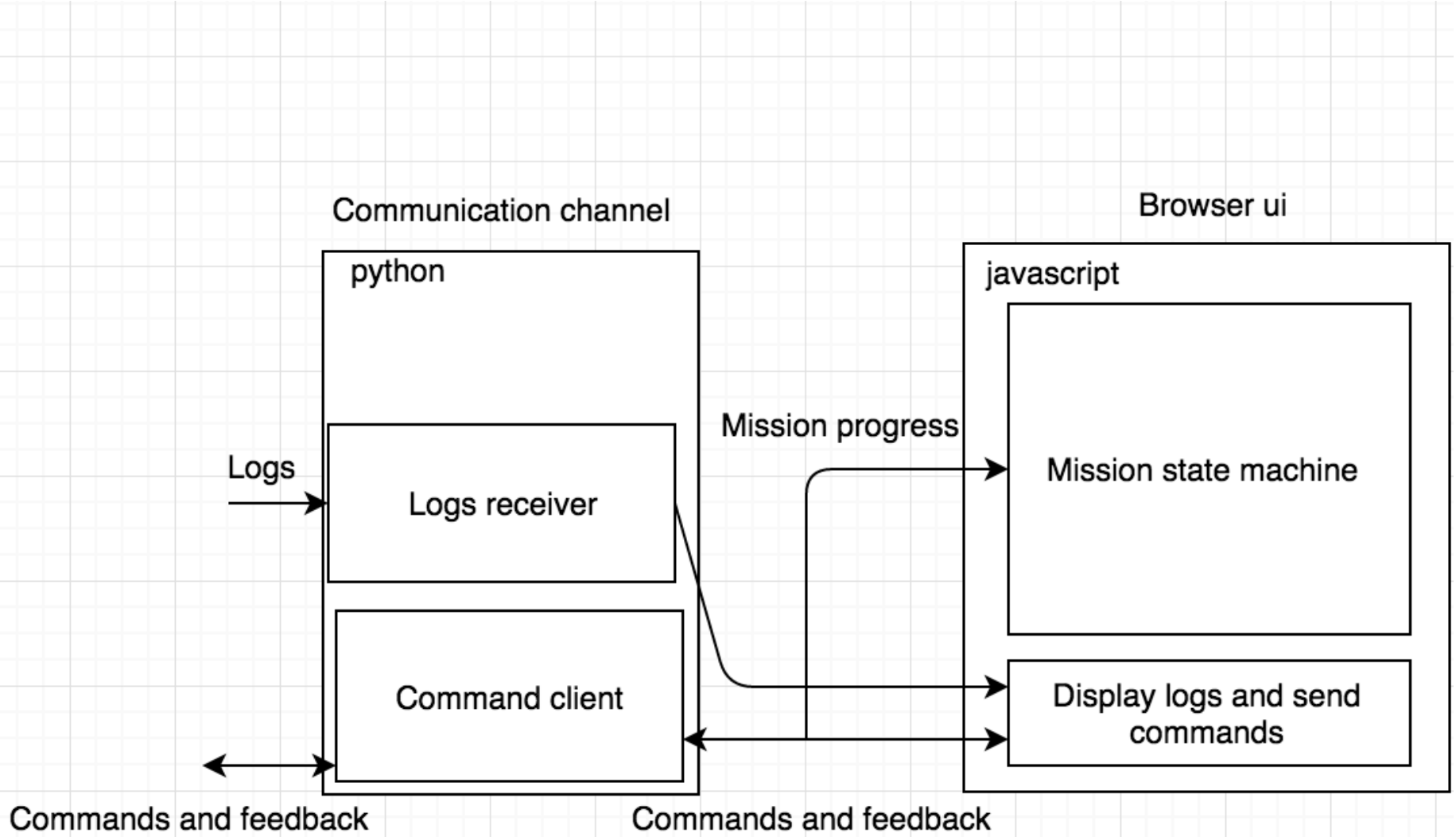
HIGH LEVEL FLOW CHART



ODROID COMPONENTS

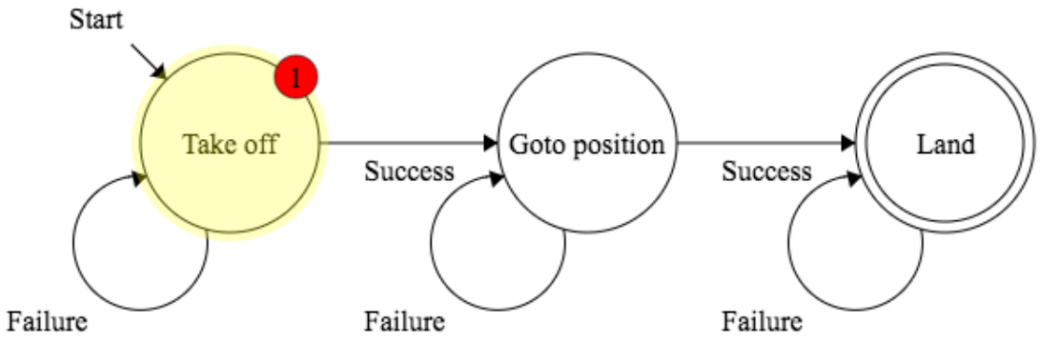


COMMUNICATION CHANNEL AND BROWSER



BROWSER UI FOR DEMO MISSION

Demo Mission



THE INFRASTRUCTURE – ODROID COMPONENTS

Mission Manager:

This is the 'brain' of the system. After sending a command to start a mission, it is taking control until completion, or until getting an abort signal from the user.

The mission is a state machine. Each state has a specific functionality (e.g. moving the drone, analyzing an image), and there is a function for every edge, which decides whether or not to move to the state on the end of the edge.

This component is designed to be very generic. In order to create a new mission, one will need to determine the state machine, write the state functionalities and the edge functions.

THE INFRASTRUCTURE – ODROID COMPONENTS (CONT.)

Video Provider:

This component is in charge of getting the frames from the camera.

Each component in the system can get the latest frame without concern for synchronization.

reads from the DEFAULT channel

Video Streamer:

Stream the video to the GCS over UDP connection read from the DEBUG channel so you can apply modification on the frame before sending it .

Video Recorder:

Record the video to a file reads from the DEFAULT channel

VIDEO CHANNELS

- To better understand the channels concept imagine one component that provides the original video stream from the camera(the video provider) that stream then split to different channels for now we have only 2 channels the DEFAULT channel and the DEBUG channel .
- The DEFAULT channel provides the original video . In the other hand you have the DEBUG channel every component in the system can create collection of modifiers you can imagine it as a pipeline of filter masks and transition that applied to the frame before you can read it . A
- good example that make the concept more clear : when before we runs the algorithm to find the center of the target in the image we prepare a collection that include gray scale filter and target_center_mask the filter transform the image from color to gray and the mask paint the center of the image in white then we load that collection in the DEBUG channel and then the video streamer reads the frame from the DEBUG channel and send it to the ground control and you can see what the drone finds in real time .
- Meanwhile the video recorder for example reads from DEFAULT channel so it doesn't read the modified frames

THE INFRASTRUCTURE – ODROID COMPONENTS (CONT.)

Logger:

The logger has several parts:

- ITraces is logged to file, to screen and to the GCS over UDP connection.
- State machine updates is logged and sent to the GCS over TCP connection. We chose TCP connection for these kind of logs because the important of them is greater than other logs.

Pixhawk communication module:

This component is written in Python because it uses the DroneKit SDK. The component has implementations for atomic commands to control the drone (e.g. 'takeoff').

THE INFRASTRUCTURE – GCS COMPONENTS

Communication Channel:

Combines the messages we send and receive over the connections we have with the Odroid. Sending commands to the drone is done over TCP connection.

After receiving the data, we can pass it to the following modules and display it to the user.

THE INFRASTRUCTURE – GCS COMPONENTS (CONT.)

Video display:

Displays the video stream we get from the Odroid.

Web User Interface:

This component displays the traces received from the drone, and displays the current state of the drone in a graphic view, so the operator can easily understand what's the drone is currently doing (for example see if it's stuck). Sending commands to the drone is also done from this module.

COARSE TARGET SEARCH ALGORITHM

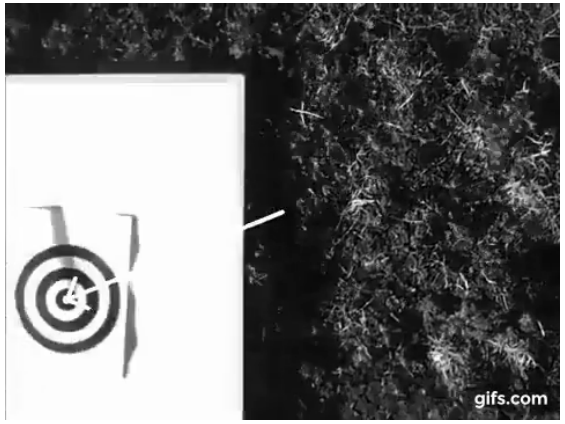
1. Run Haim Krietman's algorithm to find the polygons in the frame
2. Filter small polygons
3. Find the most circle-like polygons:
    1. Area <- polygon area
    2. Perimeter <– polygon perimeter
    3. Polygon-rank <- perimeter^2 / area – 4*PI
4. For each polygon calculate the center of mass and radius
5. Find the group of points (3-5 points) from the circles centers scatter with minimum diameter, and diameter is small enough (<= threshold)
6. Check if the group of polygons with minimum diameter for their centers are contained one inside the other
7. If we have such group return the center of the points in the group
8. Otherwise return false.

FINE TARGET SEARCH ALGORITHM

- Polylines <- find all polylines with Haim Krietman's algo
- Foreach polyline in polylines do
  - Foreach point in polyline do
    - A_matrix<– point.x^2 + point.y^2
    - B_matrix<- 2 * point.x , 2*point.y , 1
  - We want to find matrix c when C_matrix = (B_trans*B)^-1*(B_trans*A)
  - C_matrix = [r^2-a^2-b^2] so if we will find this we will have the center of the circle
    - C_matrix <- (B_trans*B)^-1*(B_trans*A)
  - Rank all suspects and find the top 20
  - Run find suspects from polygons algo
- Check if the group of polygons and polylines with minimum diameter for their centers are contained one inside the other
- If we have such group return the center of the points in the group
- Otherwise return false.

PID CONTROLLER

- We used PID controller in the fine phase when the drone need to land on the center of the target
- Our system contain error in a form of distance between the center of the frame and the center of the target .
- The output that changes the system state and minimize the error is changing the velocity vector base on the Proportional Integral Derivative methodology .



FINAL VIDEOS