SOFTWARE REQUIREMENTS SPECIFICATION for FOLLOW ME DRONES

COS301, University of Pretoria, South Africa

Five Guys, One Branch

Francois Venter (u14177553)
Brendon van Biljoen (u17011842)
Devon Petrie (u17019266)
Len Bekker (u11026953)
Gilad Tabul (u16154658)

July 18, 2019

Architectural Design

1.1 Architectural Overview

1.1.1 Interactive subsystem (Type)

The mobile app allows the ranger to perform specific actions, such as:

- Connecting to the drone
- Giving basic commands to the drone
- Viewing the detections

This is done because the ranger may be required to connect to multiple drones and have the ability to communicate with each indepenently.

1.1.2 Event driven system (Type & Style)

- Object recognition notifies the server based on object detections made in real time. The server will process this information and decide whether or not an event has been triggered. If it has, then it will emit a signal to the mobile app notifying it of a detection.
- The server will be notified by the app of any ranger movements. The server will then issue commands to the drone which will cause the drone to follow the ranger.

This was done because the drone needs to change its actions based on actions. Our systems need to change their behaviour based on specific inputs and adjust accordingly.

1.1.3 Database system (Type) & persistence (Style)

- Dropped pins generated from server snapshots will contain animal snapshots, location data and drone information. These pins will be stored in a database for future use.

This is done in order to record data for any future use that may be required.

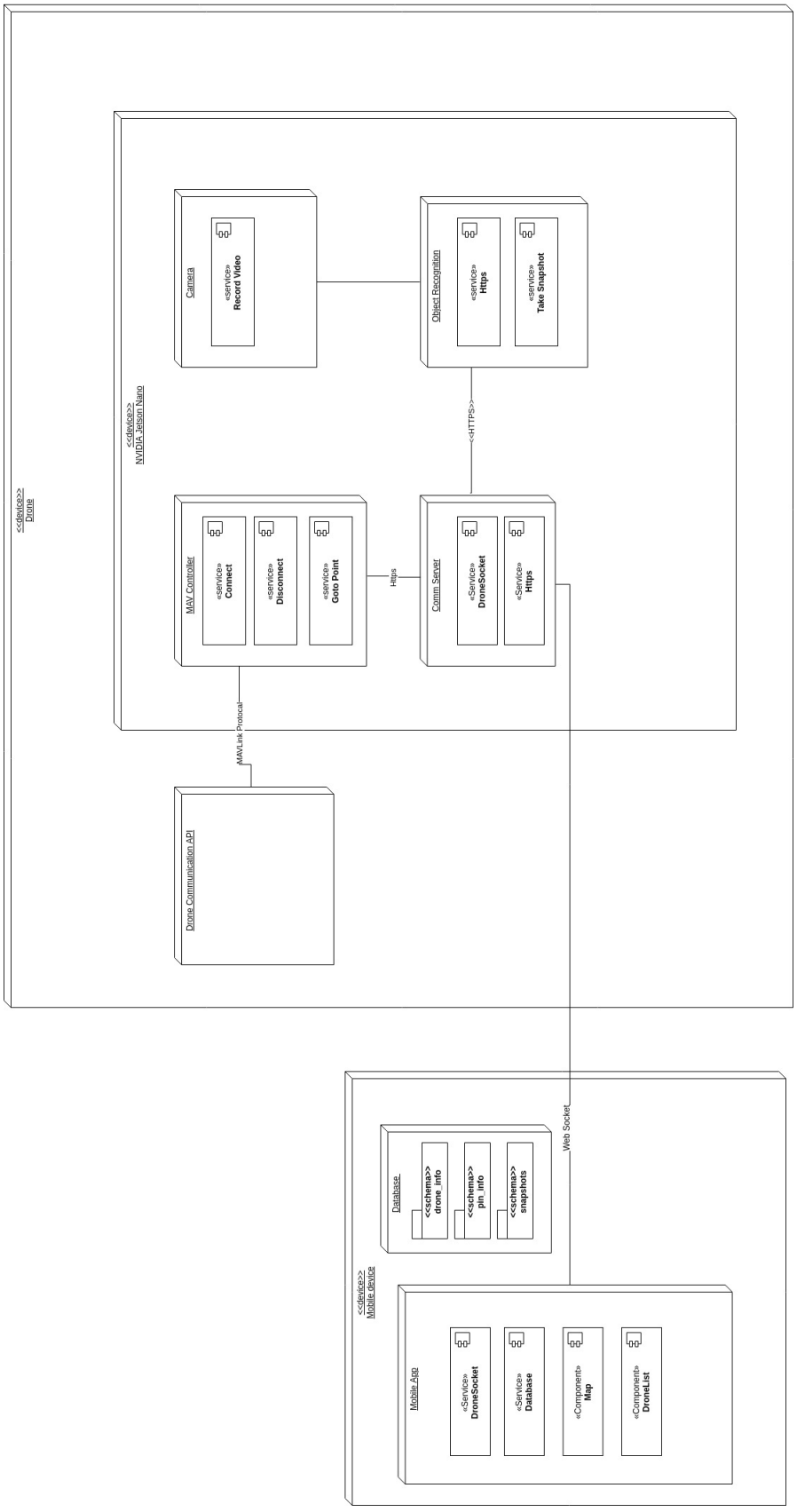
1.1.4 Client-server

- At the centre of our system we have a server that acts as a mediator. The clients, object recognition and the mobile app, communicate back-and-forth with the server, relaying information as events are triggered.

We have multiple subsystems that need to communicate with each other, thus a central server with multiple clients was the ideal solution. The server also allows us to open a socket between it and the app, allowing data to be pushed instead of needing to poll.

1.2 Design Principles

- Due to separation of concerns, high cohesion and modular design, our system is implemented in such a way as to be easily changed in the future. Thus implying a design for changeability, maintainability and upgradeability.
- The subsystems are designed to be general purpose, to allow for repurposing if need be.


<<device>>
Drone
<<device>>
NVIDIA Jetson Nano
Camera
«service»
Record Video
Object Recognition
«service»
Https
«service»
Take Snapshot
<<HTTPS>>
MAV Controller
«service»
Connect
«service»
Disconnect
«service»
Goto Point
Https
Comm Server
«Service»
DroneSocket
«Service»
Https
MAVLink Protocol
Drone Communication API
Web Socket
Database
<<schema>>
drone_info
<<schema>>
pin_info
<<schema>>
snapshots
<<device>>
Mobile device
Mobile App
«Service»
DroneSocket
«Service»
Database
«Component»
Map
«Component»
DroneList

Subsystems

2.1 Drone Navigation

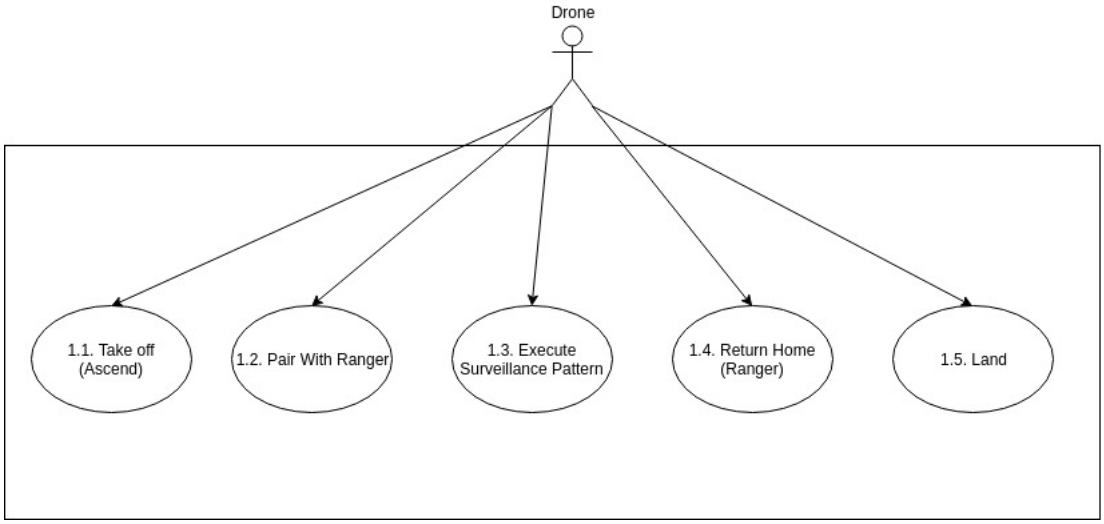
2.1.1 Functional Requirements

- **R1:** FMDS will **follow** the ranger
  - **R1.1** FMDS will allow the ranger to perform **initialisation** for the drone.
    - **R1.1.1** FMDS will allow the ranger to set himself as the validated object.
      - The validated object will not be treated as a normal object when searching the field of view of the drone. Notifications are sent to the ranger when objects are detected, however notifications will not be sent every time the ranger is identified.
    - **R1.1.2** FMDS will allow the ranger to initialise his beacon.
      - The beacon will allow the drone to know where the ranger is at any moment in time, which will assist in expanding the drone's field of view since the drone will not need to be facing the ranger.
  - **R1.2** FMDS will allow the ranger to specify a **flight pattern** for the drone
    - **R1.2.1** FMDS will detect objects while on its flight path.
      - Object recognition (as a service) will be running from the moment the ranger initialises the drone until the moment it lands. Rectangles will be drawn around identified objects along with the label for each identified object.
    - **R1.2.2** FMDS will record live video while on its flight path.
      - Stored video data will be used as further training data for the object recognition model.

- **R2:** FMDS will **patrol** the area around the ranger
  - **R2.1** FMDS will patrol by flying in the designated **flight pattern** around the ranger.
    - The desired flight pattern will be chosen upon initialisation and will remain as the default flight pattern unless updated during the drone's flight.
  - **R2.2** FMDS will perform **object detection**.
    - **R2.2.1** FMDS will drop a pin of the object detected on the 2D map.
      - The coordinates of the drone's current location will be sent to the server with a command to drop a pin at that location. In addition to the coordinates, the object's classification (elephant, rhino, person, etc) will will be sent to the server. the server will then attempt to assess the identified object's threat level. The server will then send an update to the ranger's app on their phone which will generate a pin. The pin will be coloured based on the perceived threat level.
  - **R2.3** FMDS will **record** video footage during its surveillance.
    - Stored video data will be used as further training data for the object recognition model.
  - **R2.4** FMDS will take **snapshots** of the reserve during its surveillance.

    - **R2.4.1** FMDS will build a 2D map of the reserve.
        - Snapshots will be periodically taken and stitched together at a later stage to create a virtual 2D map which is regularly updated. The updated 2D map will be useful to the rangers when wildlife, plantlife and the general health of the reserve is monitored.

2.1.2 Use Case Diagram



Drone Navigation Use Case Diagram

2.1.3 Technical Requirements

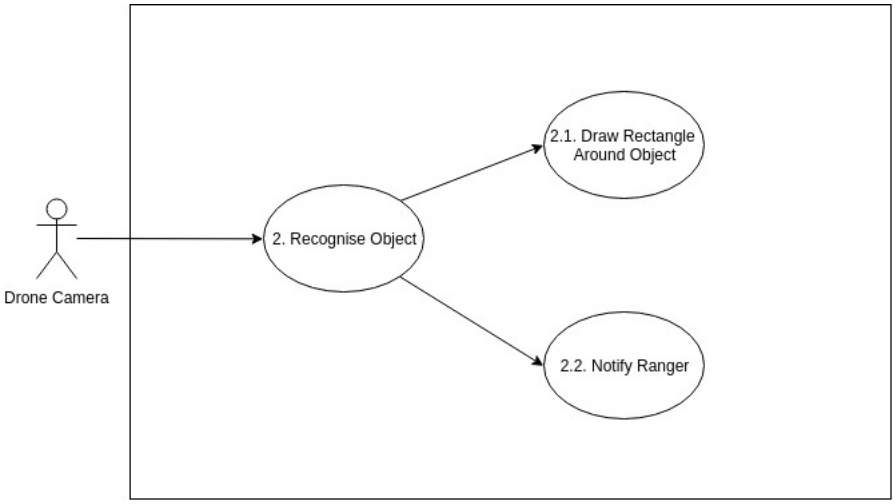
- Python v2.7

2.2 Object Recognition

2.2.1 Functional Requirements

- **R3:** FMDS will **detect objects** in the reserve
    - **R3.1** FMDS will allow the ranger to see a list of **objects** and their **locations**.
        - If an object is selected, the ranger will be pointed towards the detected object. Individual estimated threat levels will be displayed per object along with the object's classification.
    - **R3.2** FMDS will draw a rectangle around the objects that it detects.
        - Rectangles will be coloured and a label will be placed above the rectangle with the object's classification.
    - **R3.3** FMDS will notify the ranger of the objects that it detects.
        - The ranger will be notified of newly detected objects, their estimated threat level, classification and relative direction from the ranger.

2.2.2 Use Case Diagram



Object Recognition Use Case Diagram

2.2.3 Technical Requirements

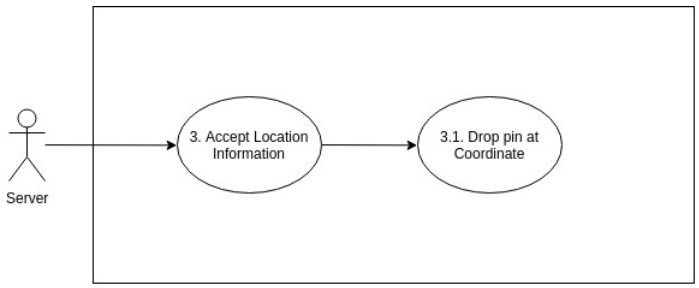
- gcc v8.0+
- cuda v10.0+
- opencv v3.0+
- darknet Yolo v3 Framework

2.3 Mapping

2.3.1 Functional Requirements

- **R4:** FMDS will build a **2D map** of the reserve
  - **R4.1** FMDS will use the snapshots taken to build a 2D map of where it has flown.
    - Snapshots will be periodically taken and stitched together at a later stage to create a virtual 2D map which is regularly updated. The updated 2D map will be useful to the rangers when wildlife, plantlife and the general health of the reserve is monitored.

2.3.2 Use Case Diagram



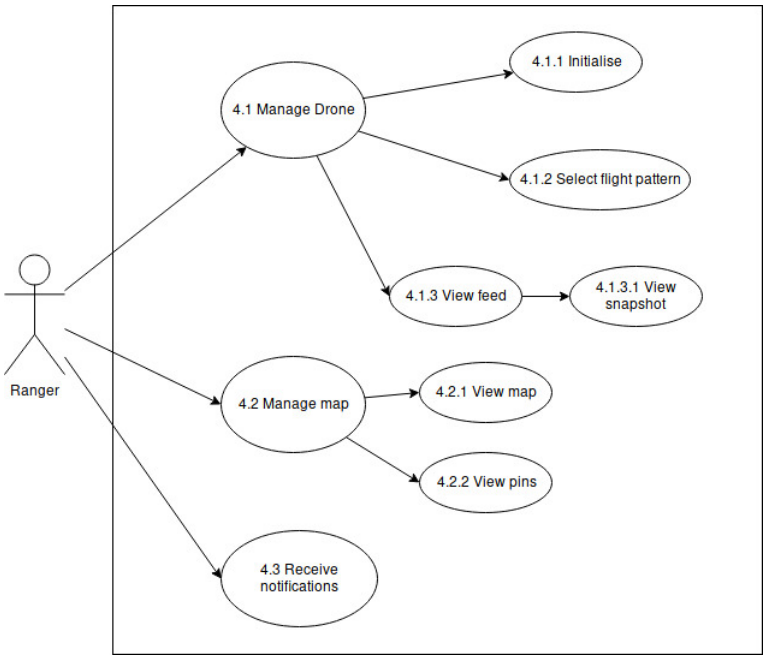
Mapping Use Case Diagram

2.4 User Application

2.4.1 Functional Requirements

- **R5:** FMDS will provide an interface for the ranger to initialise the drone.
  - Discussed in previous functional requirement (R1.1.2).
- **R6:** FMDS will provide an interface to display snapshots of the video feed.
  - The periodic snapshots (which will be used to generate the virtual 2D map) will be accessible via the application.
- **R7:** FMDS will provide an interface to display the live video feed.
  - The live video feed coming from the drone will be accessible via the application.
- **R8:** FMDS will retrieve map and pin data from the server.
  - Discussed in previous functional requirement (R2.2.1).
- **R9:** FMDS will provide an interface for the ranger to select a predefined flight pattern.
  - Discussed in previous functional requirement (R2.1).
- **R10:** FMDS will provide a user interface for the notification received from the other subsystems.
  - The notification received will include the relative position to the detected object from the drone's location.

2.4.2 Use Case Diagram



User Application Use Case Diagram

2.4.3 Technical Requirements

- Ionic cross platform framework.
- Cordova compilation suite.
- Angular Framework.