CyDrone

Kenneth Lange - Team Leader
Alain Njipwo - Chief Hardware Developer
Daniil Olshanskyi - Chief Software Developer
Luke Bell - Chief Interface Developer
Max Medberry - Chief Backend Developer

Advised by Dr. Ali Jannesari

Sdmay20_47
https://sdmay20-47.sd.ece.iastate.edu/

Project overview

- Build a custom drone
  - ZED Camera
  - Nvidia Jetson
  - Pixhawk
- Simulation Environment
  - Unreal Engine
  - AirSim

Problem statement

- Real-time Processing
- Custom Drone
- Expandability
  - Modular Code
  - Documentation

Conceptual/Visual Sketch

Figure 1: Volumetric Analysis

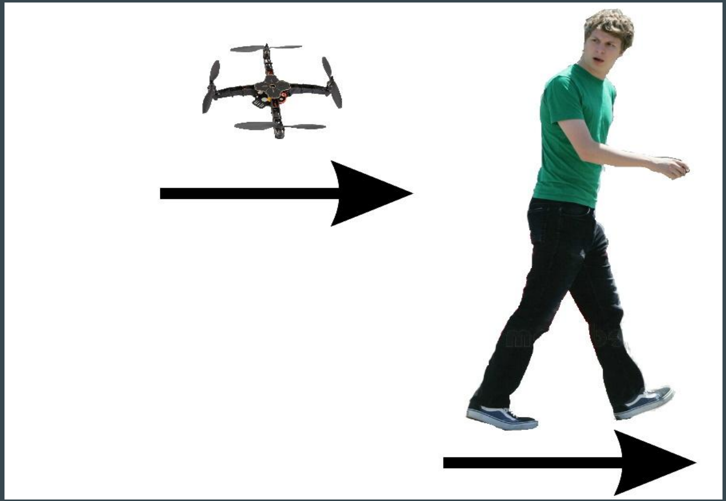


Figure 2: Autonomously Follow Target

Functional requirements

- Set up simulation environments with various features
- Simulation must be able to be used for machine learning
- Assemble the drone itself
- Drone must be stable when flying
- Drone must be controllable using RC
- Drone must be able to carry Nvidia Jetson SoC computer
- Drone must be controllable programmably

Non-functional (technical, other constraints, considerations)

- Non-functional
  - Software will be modular and well-structured
  - Code will be well documented, both via comments and a wiki
  - Hardware assembly instructions will be documented
- Technical and/or other constraints
  - Need to do research/testing to learn how to build the drone and calibrate it for stable flight
  - Need to learn new software technologies (Unreal Engine, AirSim, ROS, PX4)

Potential risks and mitigations

- Custom drone
  - Complex, expensive parts with long waits on ordering
  - Mitigation
    - Thoroughly research required parts, use care when handling and testing drone
- System integration
  - Project relies on multiple third party software systems to interface reliably
  - Mitigation
    - Use one LTS version of each software, where compatibility is guaranteed
- Simulation to Real World
  - Drone hardware and sensors may behave differently in real world compared to simulation
  - Mitigation
    - Begin testing in real world with safe environments, empty room, padding/nets, etc.

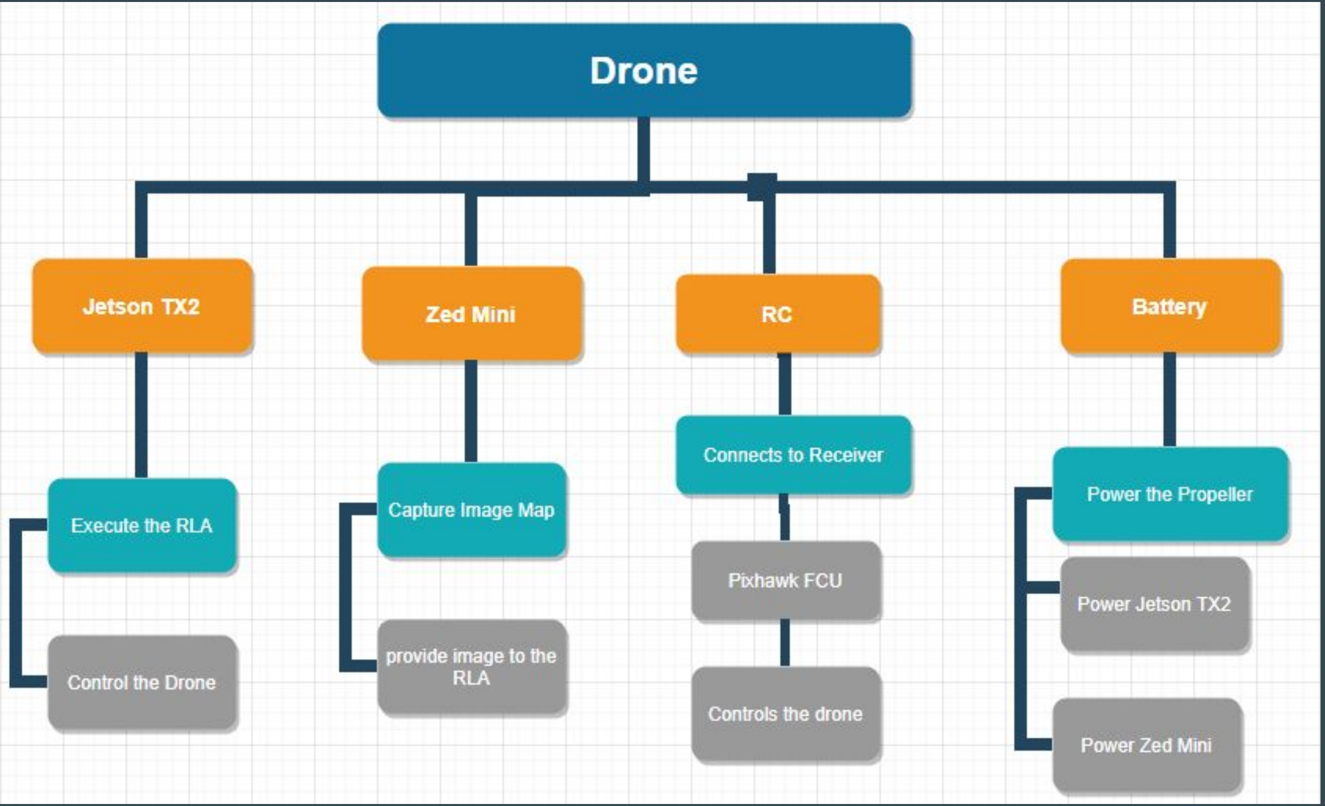
Resource/Cost estimation

- Software
  - All software used in the project is free and open source
    - Ubunutu 16.04/18.04
    - AirSim
    - ROS
- Hardware
  - The client is covering hardware costs for the drone
    - Nvidia Jetson TX2 - $450
    - UAV Chassis - $190
    - Large, powerful LiPo battery - $110
    - ESCs / Rotors / Blades x 4 - $230
    - ZED Camera - $450
    - PC with powerful CPU and GPU to quickly train RLA

Project milestones and schedule

- Researching hardware/Drone Assembly
  - Weeks 1-15
- Researching simulation/AirSim Environments
  - Weeks 1-15
- Training RLA in AirSim
  - Weeks 16-18
- RC drone flight
  - Weeks 16-28
- Jetson drone flight
  - TBD
- Autonomous drone flight
  - TBD

Functional decomposition



Detailed design - Parts & Frame assembly

Source: http://www.helipal.com/tarot-x4-quadcopter-frame-set.html

Detailed Hardware design - 1:Power Components

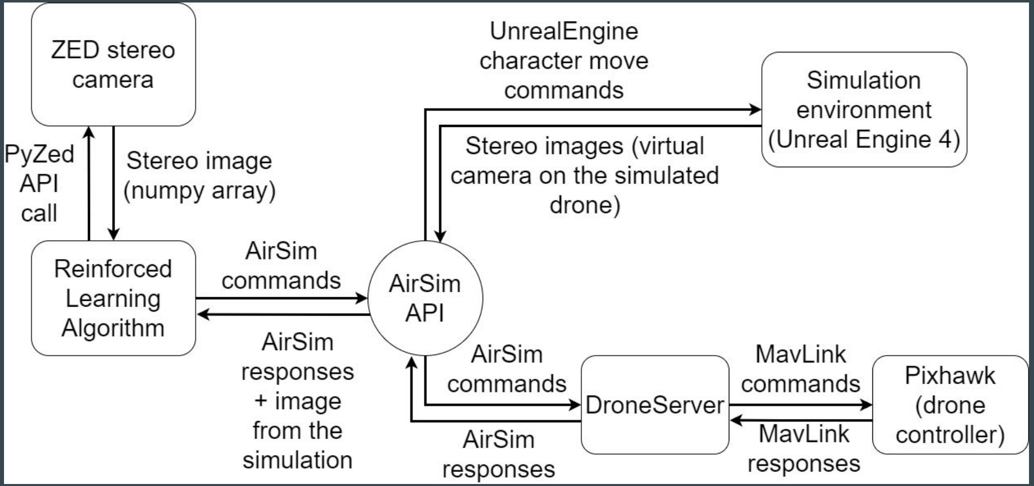
- Carbon Fiber Propellers x 4
- 4s 4006 DC Brushless motors x 4
- FlyColor 35 A ESC boards x 4
- FS-IA10B Receiver x 1
- PixHawk x 1
- Battery: Venom 4 Cell 5000mah 14.8 V

Source: http://www.helipal.com/tarot-x4-quadcopter-frame-set.html

Detailed Hardware Design - 2 : Power Components

Detailed design - Software

- Based around working with RLA
- AirSim Python API to send movement commands to drone (real-world & simulation)
- AirSim retrieves images & depth from UE to send to RLA
- DroneServer converts AirSim commands to MavLink to send to PX4 (real-world)
- ZED Python API to interface with camera inside RLA (real-world)



HW, SF, Technology platforms used

- Hardware
  - Tarot 650 Drone Platform
  - PX4 Pixhawk Flight Control Platform
  - Nvidia Jetson Mobile Computing Platform
- Software
  - Ubuntu (16.04 & 18.04)
  - AirSim & Unreal Engine
  - PX4 Firmware & QGroundControl
  - Robot Operating System (ROS)

Test plan

- Hardware Testing
  - Sensor calibration testing
  - Drone RC controls test (no blades)
  - RC-controlled test flight
  - ZED camera diagnostic test
- Software Testing
  - Simulation AI script testing
  - Controlling simulated drone with ROS-AirSim
  - Controlling drone with software (no blades)

Prototype Implementations

- Initial Prototype: ROS + Gazebo & small drone with Raspberry Pi
  - Identified issues with Gazebo
  - Client required larger drone & more powerful computing platform

- Second Prototype: Unreal Engine + AirSim & custom Tarot quadcopter chassis with Nvidia Jetson TX2
  - UE + AirSim worked better for us
  - New chassis & onboard computer capable enough for requirements of client

Engineering Standards and Design Practices

Hardware Standards:

- PX4 flight stack and ArduPilot
- CAN
- MavLink protocol

Software Practices/Standards:

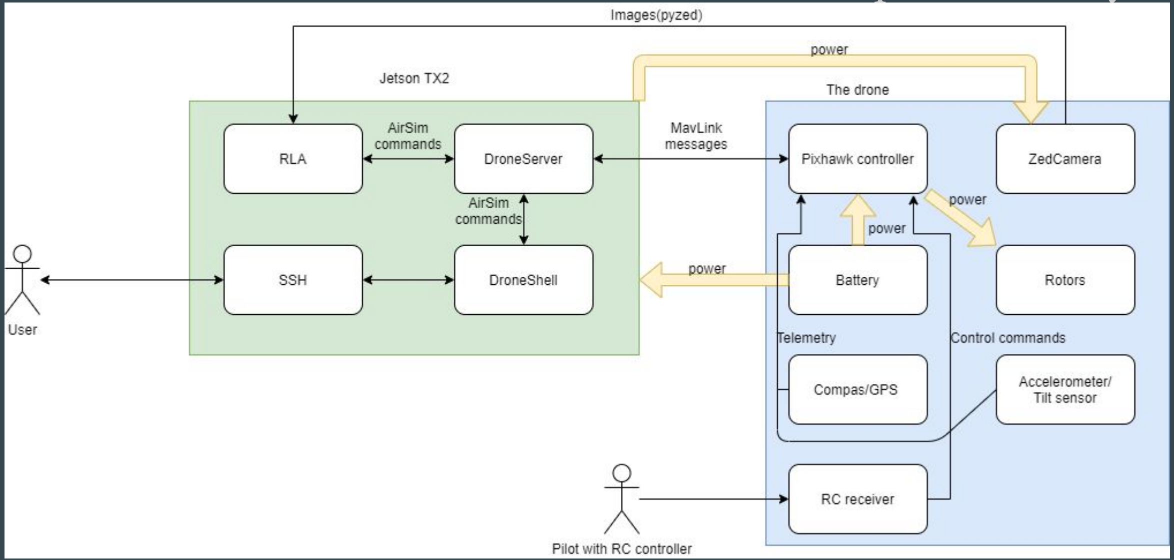
- GIT
- AirSim API
- Client-server
- ROS

Team contributions

- Kenneth Lange - Simulation setup, simulation environments, simulation maintenance
- Alain Njipwo - Drone assembly, hardware and hardware connectivity, Pixhawk controller
- Daniil Olshanskyi - Simulation setup, simulation human AI, Pixhawk controller, Jetson-Pixhawk interaction
- Luke Bell - Stereo camera, ROS, ROS-AirSim interaction
- Max Medberry - wind simulation, ROS, ROS-Airsim interaction

Current status

- Simulation and simulation environments are operational and RLA is trained
- Drone is fully flight-capable and stable
- Drone can be controlled from RC and from Jetson (possibly at the same time)
- Images from the stereo camera can be fetched to be reframed and processed by the control algorithm



Future of the project - Dr. Ali Jannesari's project

- Test programmable drone controls (safe environment)
- Test drone fully assembled
- Test drone autonomous flight
- Apply and test volumetric analysis
- Develop different control algorithms

Thank you for your attention!