# Project Pegasus

Package Delivery System

**Team A**

Tushar Agrawal

Sean Bryan

Pratik Chatrath

and Adam Yabroudi

# Teaser

# What we’re doing

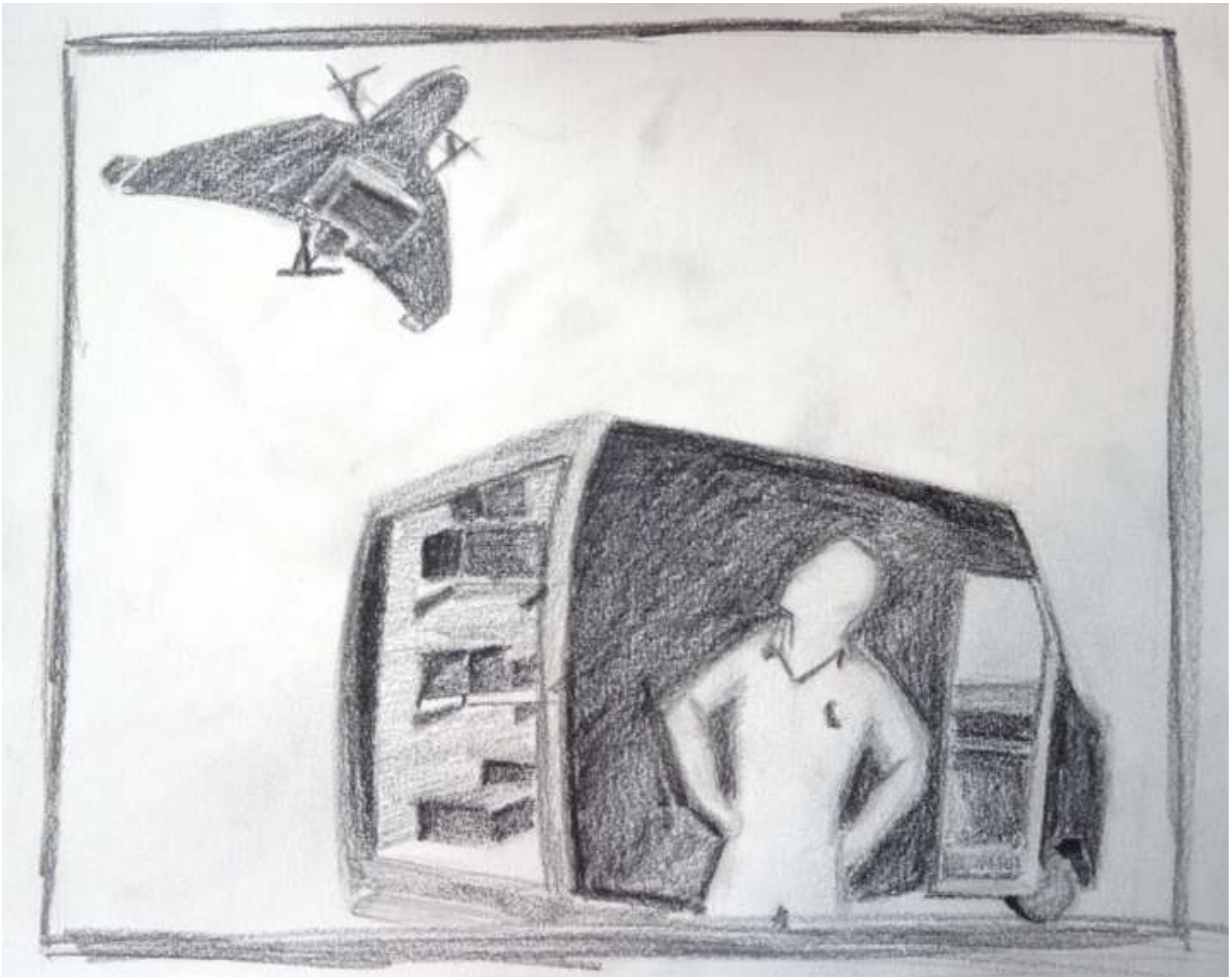
Problem

- Delivering packages to a house using UAVs

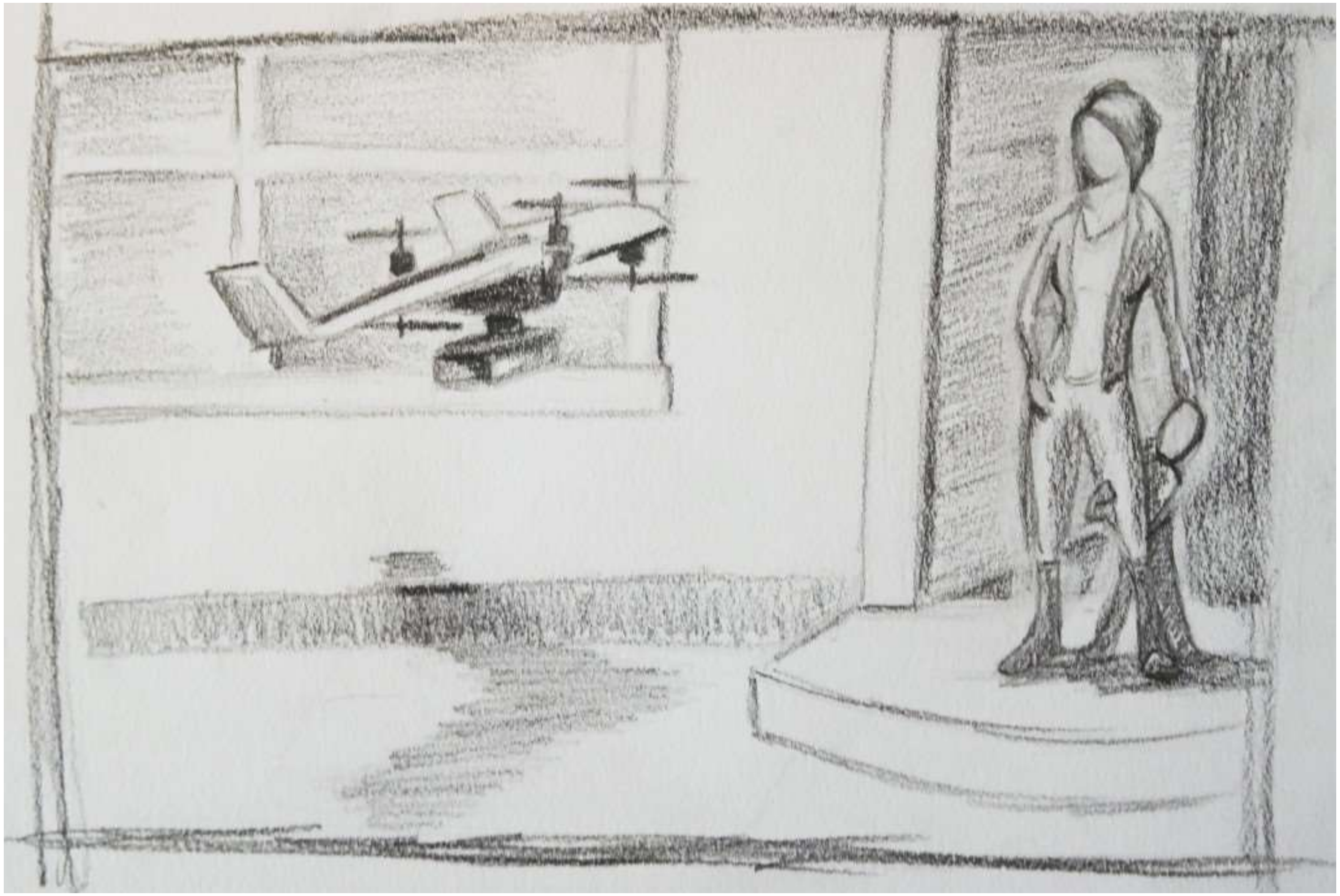
Problem Description

- Given the coordinates of the house, a UAV with a package takes off from point A
- Autonomously reaches close to the house
- Scans the outside of the house for a visually marked drop point, lands, drops off the package,
- Takes off again to land on another platform at point B.

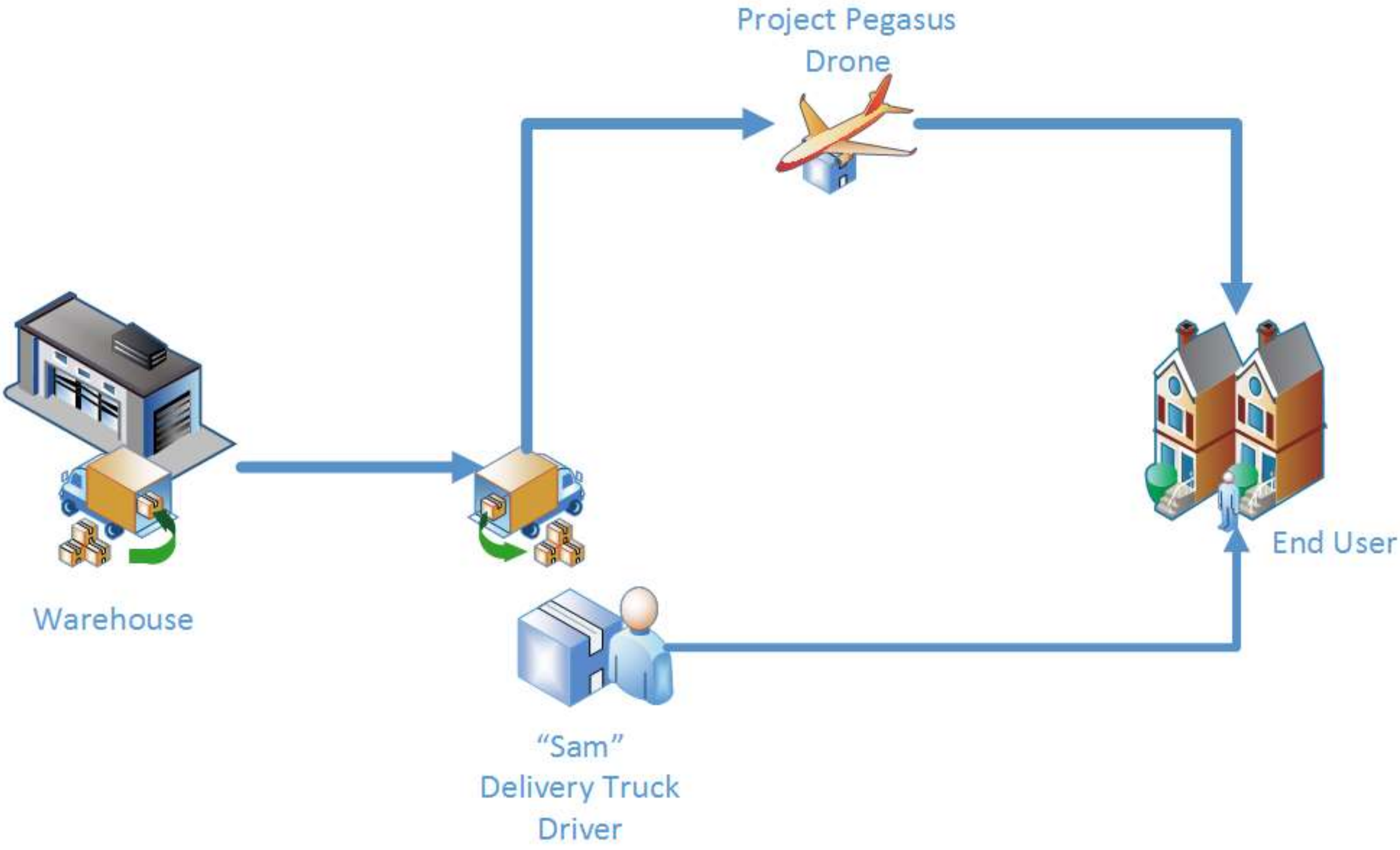
# Use Case



# Use Case



# Use Case



Project Pegasus
Drone
Warehouse
End User
"Sam"
Delivery Truck
Driver

# System Requirements

- Mandatory Functional Requirements
- Mandatory Non-Functional Requirements
- Desired Requirements

# Mandatory Functional Requirements

- M.F.1 - Hold and carry packages with a maximum size of 30cm x 30cm x 20cm, weighing up to 400g.
- M.F.2 - Autonomously take off from a visually marked platform.
- M.F.3 - Navigate to a known position close to the house.
- M.F.4 - Detect and navigate to the drop point at the house.
- M.F.5 - Land at visually marked drop point (with an open landing column of 2m radius).
- M.F.6 - Drop package within 2m of the drop point.
- M.F.7 - Take off, fly back to and land at another visually marked platform.
- M.F.8 - Takes coordinates as input from the user.
- M.F.9 - Communicates with platform to receive GPS updates (intermittently).

Targeted in Fall

# Mandatory Non-Functional Requirements

- M.N.1 - Operates in an outdoor environment.
- M.N.2 - Operates in a semi-known map. The GPS position of the house is known, but the exact location of the visual marker is unknown and is detected on the fly.
- M.N.3 - Avoids static obstacles with a minimum cross-sectional dimensions of 1.5m x 0.5m.
- M.N.4 - Not reliant on GPS
- M.N.5 - Sub-systems should be well documented and scalable.
- M.N.6 - UAV should be small enough to operate in residential environments.
- M.N.7 - Able to carry packages.
- M.N.8 - Recognizes visual markers that are located at least 10m apart.

Targeted in Fall

# Desired Requirements

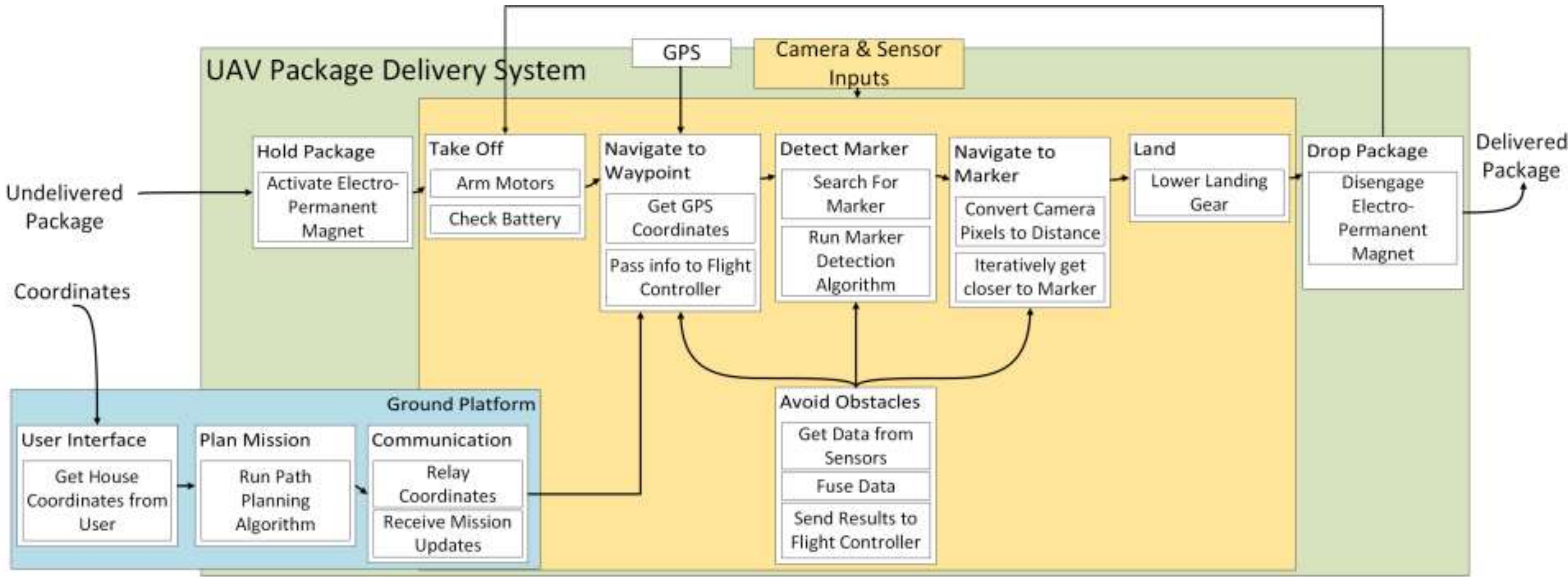
3.2.1 Functional

- D.F.1 Pick up packages.
- D.F.2 Simulation with multiple UAVs and ground vehicles.
- D.F.3 Ground vehicle drives autonomously.
- D.F.4 UAV and ground vehicle communicate continuously.
- D.F.5 UAV confirms the identity of the house before dropping the package (RFID Tags).

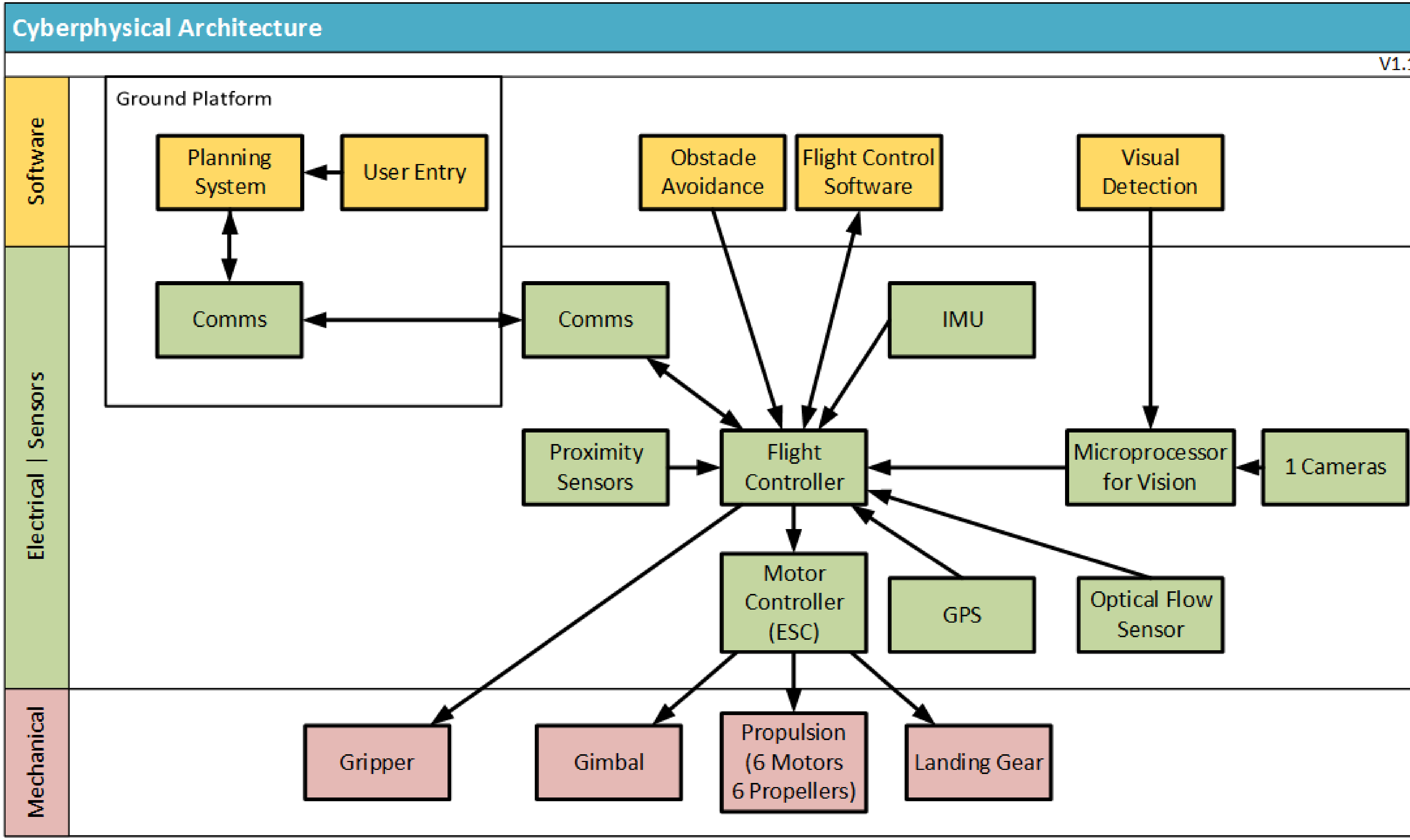
3.2.2 Non-Functional

- D.N.1 Operates in rains and snow.
- D.N.2 Avoids dynamic obstacles
- D.N.3 Operates without a GPS system.
- D.N.4 Has multiple UAVs to demonstrate efficiency and scalability.
- D.N.5 Compatible with higher weights of packages and greater variations in sizes.

# Functional Architecture



# Cyberphysical Architecture



# Current System Status

Obstacle Detection & Master-Slave Sensor Board

Vision Subsystem

Flight Control

# Targeted Fall Requirement

Vision Subsystem

MF4: Detect and navigate to marker
MN1: Operate in outdoor environment
MN2: Operate in semi-known map

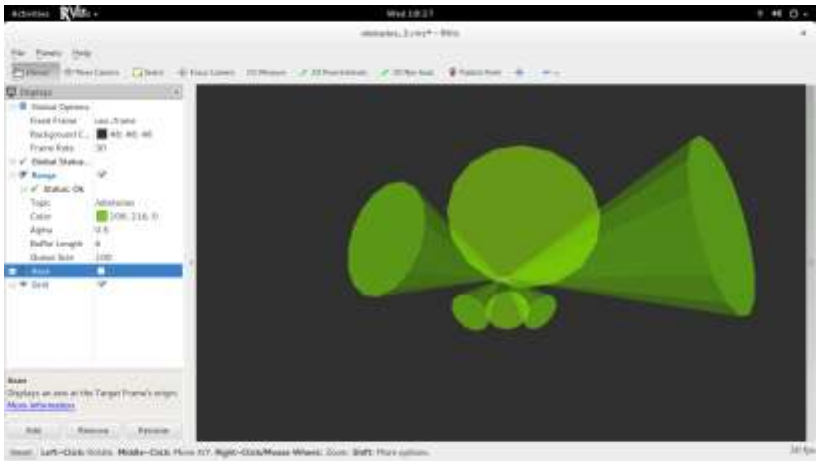
Obstacle Detection

MN1: Operate in outdoor environment
MN3: Avoid static obstacles

Flight Control

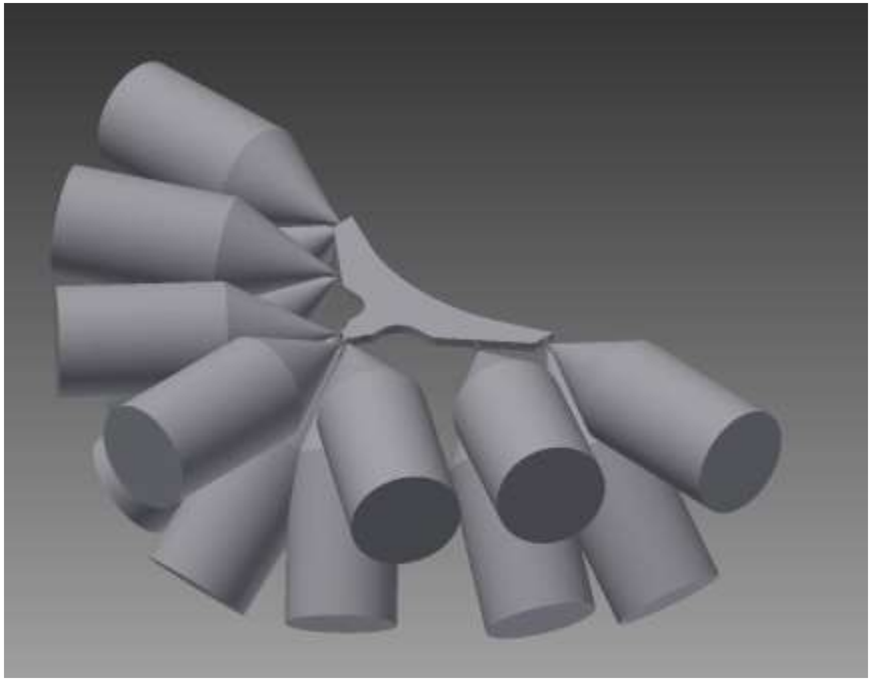
MF2: Autonomous Take Off
MF3: Navigate to known position
MF8: Take coordinate as input from user
MF9: Communicate with platform to receive GPS updates
MN1: Operate in outdoor environment

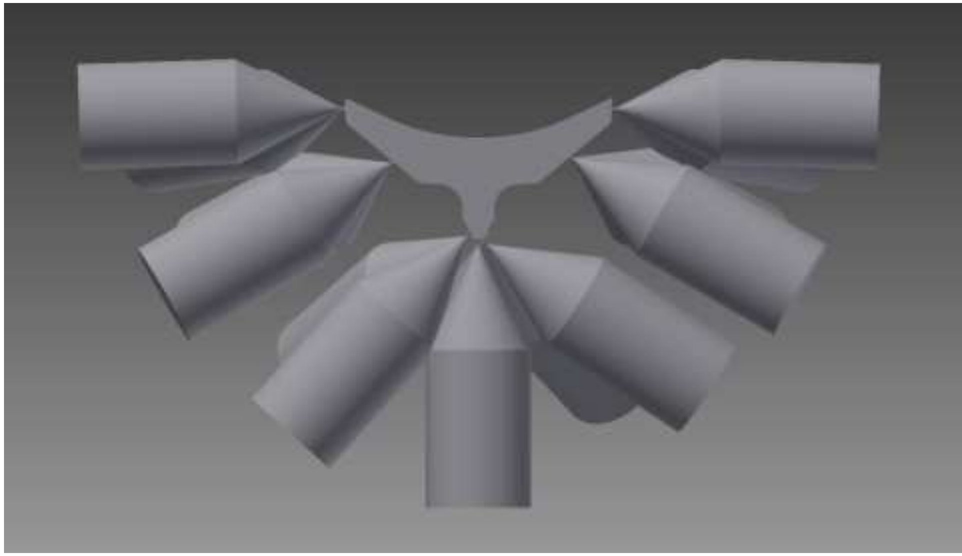
# Obstacle Detection – Subsystem Description

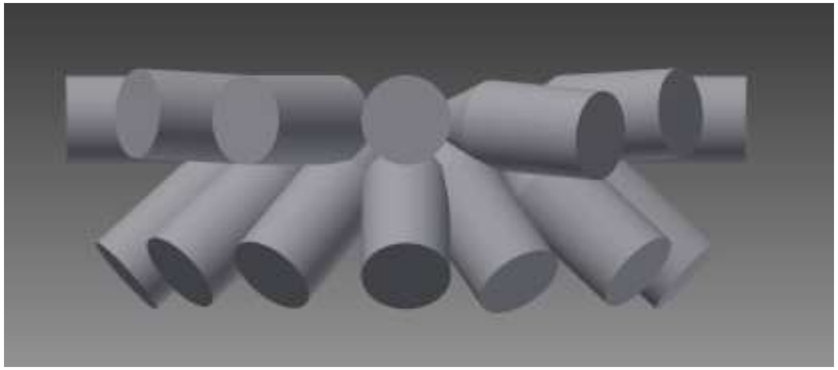
# Obstacle Detection - Video

# Obstacle Detection - Modeling



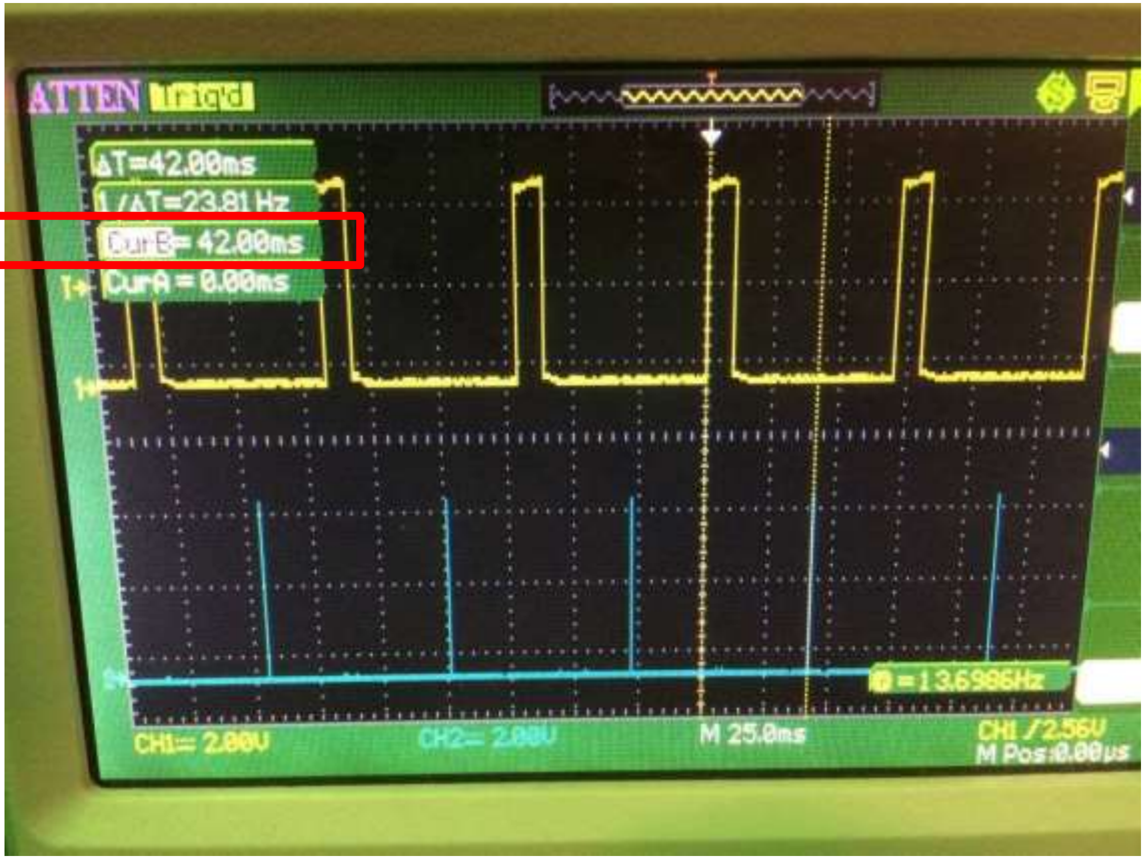


- CAD design of sensor arrangement
- 14 Ultrasonic Sensors required to cover the UAV

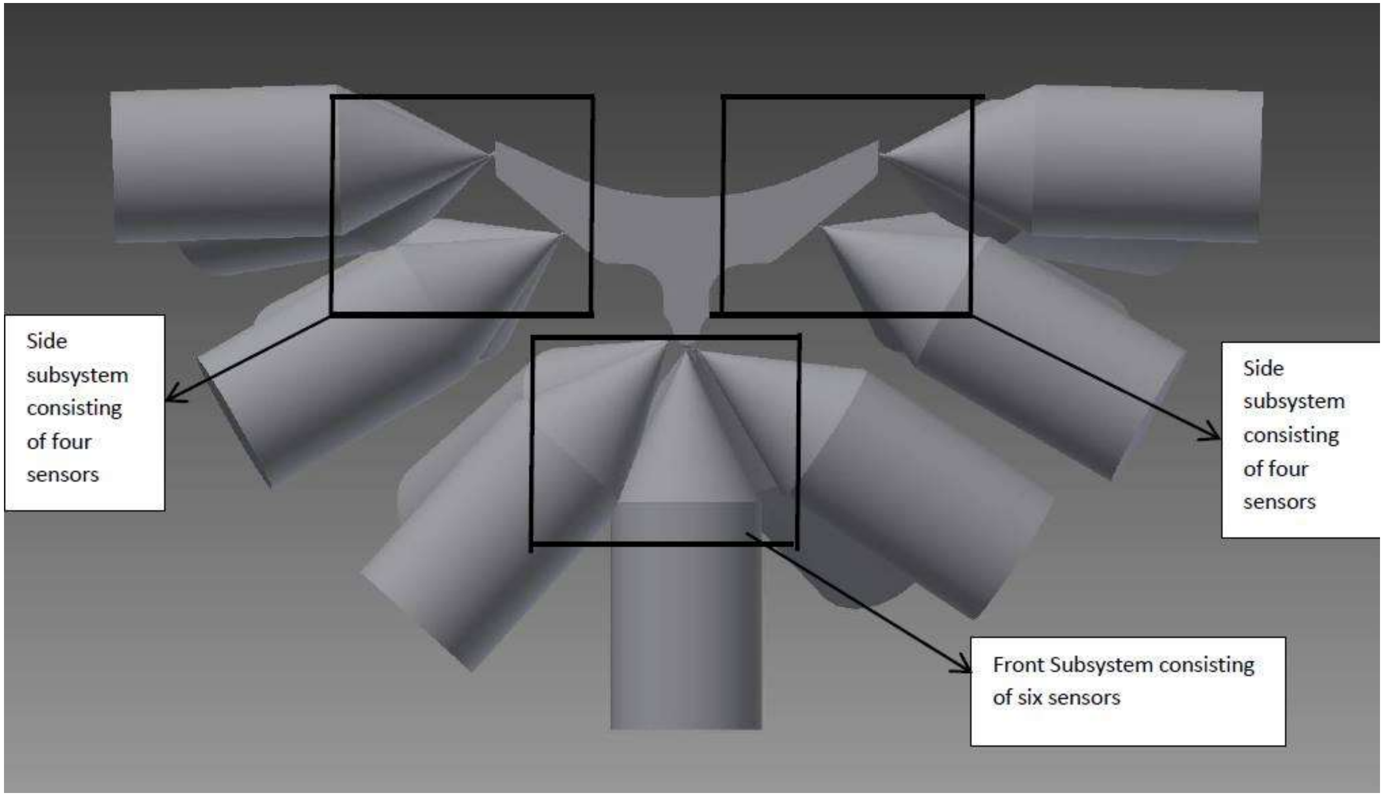


# Obstacle Detection - Test

- Time taken for 1 sensor to take reading = 42 ms



# Obstacle Detection - Test

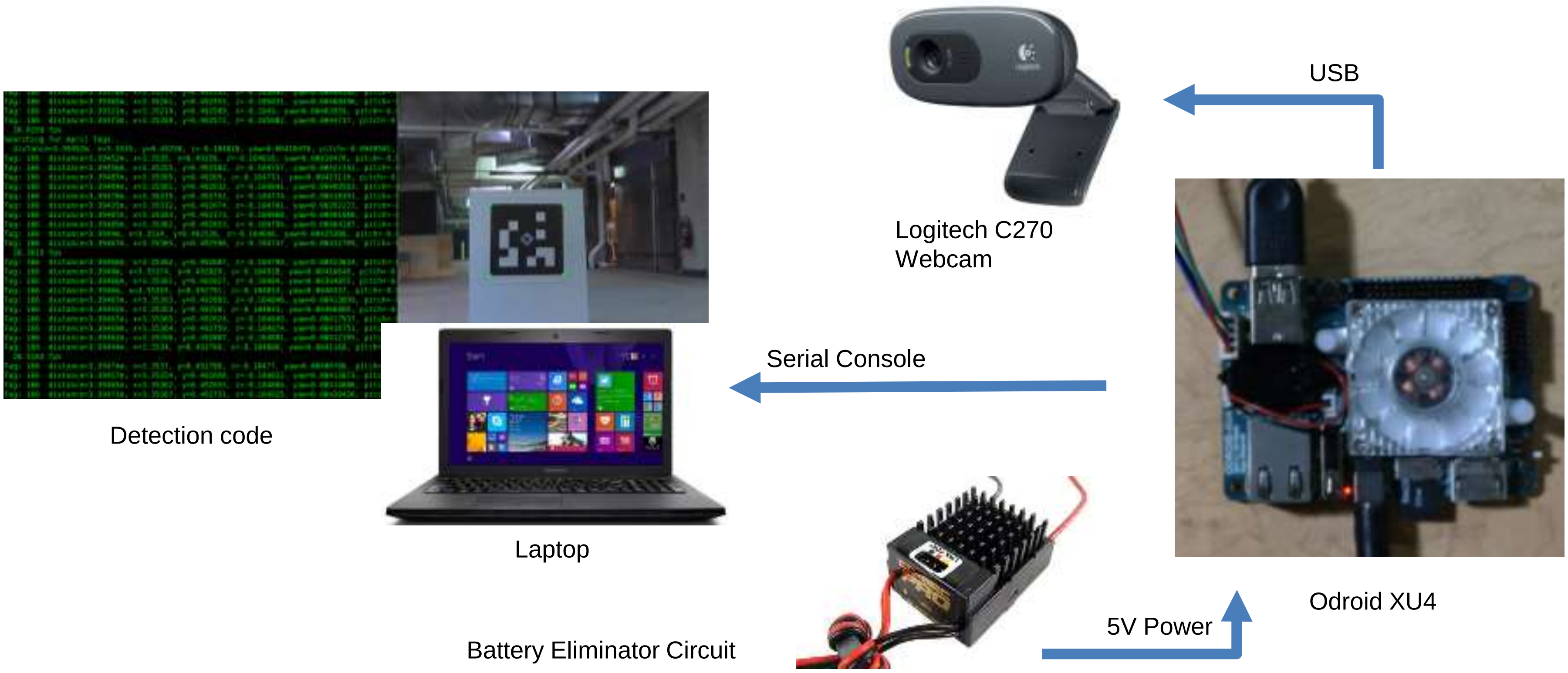



Dividing sensor subsystem into further 3 subsystems to reduce overall system sensing time

# Obstacle Detection - Analysis

- Serially pinging sensors helps to get rid of interference
- Pinging sensors facing in different direction simultaneously helps reduce update rate
- Using pinging pattern a subsystem of 6 sensors achieves 300 ms update rate

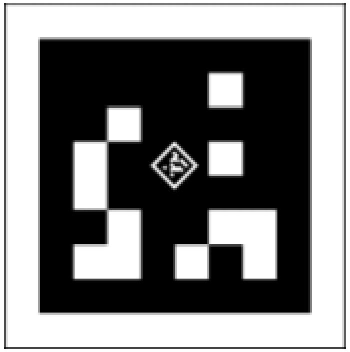
# Vision subsystem



# Vision subsystem

# Vision subsystem - Analysis

- Size - 57.5 cm square
- Range – 16 cm to 30m (Requirement for FVE: 20cm to 20m)



Nested AprilTag marker

| S.No. | Detection distances for different nested apriltag markers | | | |
|---|---|---|---|---|
| | **Outer AprilTag** | | **Inner AprilTag** | |
| 1 | 4.5cm Outer | | 0.45cm Inner | |
| | Max: 1.8m | Min: 8 cm | Not detected | |
| 2 | 18cm Outer | | 1.8cm Inner | |
| | Max: 7.2m | Min: 40cm | Max: 50cm | Min: 4cm |
| 3 | 57.5cm Outer | | 5.75cm Inner | |
| | Max: 30m | Min: 1.6m | Max: 2m | Min: 16cm |

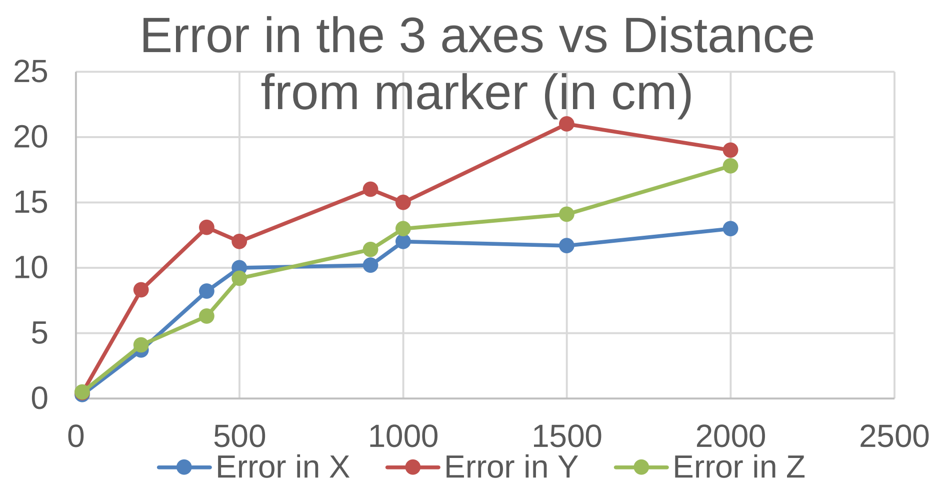
# Vision subsystem - Analysis

- Detection Speed
  - Framewise AprilTag detection slow
  - Lucas Kanade tracking speeds up continuous tracking
  - Refresh once every few frames (30) or when none found

| **Algorithm** | **FPS on Laptop (i3 4th gen)** | **FPS on Odroid (Quad core ARM)** |
|---|---|---|
| AprilTag detection | 14 | **8** |
| Lucas Kanade Tracking | 30 | 29 |
| Merged (LK + AprilTag detection) | 29 | **28** |

# Vision subsystem - Analysis

- Accuracy
  - –Camera axis calibration
  - –Measurement accuracy
  - –Attains accuracy of <5% (FVE required: 10%)





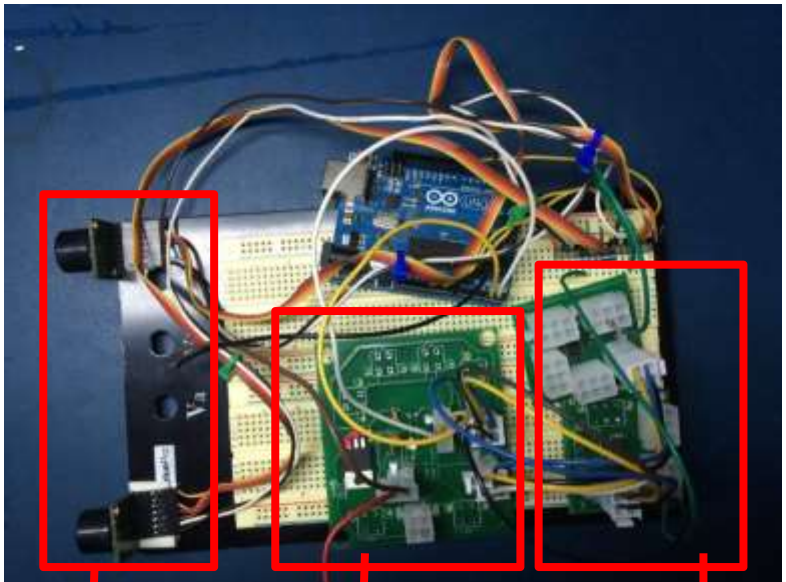
Camera mount for calibration


Marker setup

# Current Status - UAV

- Vehicle is assembled mechanically and integrated electronically
- Has flown a few times via RC controller
- Still having issues with prearm sequence due to a faulty compass. Still debugging the issue with BirdsEyeView Aerobotics

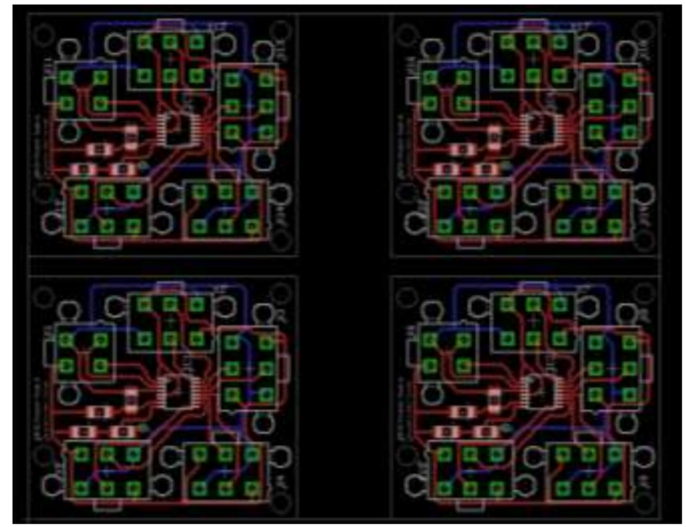
# PCB Successfully Implemented

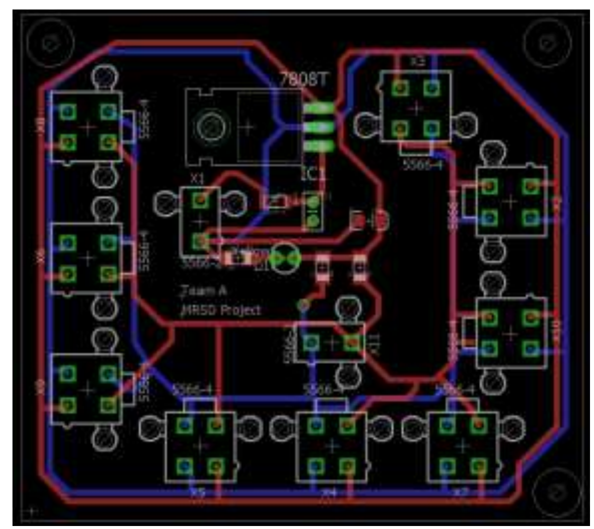


2 Ultrasonic sensor connected to 2 slave board

Power Board

2 Slave Board





# Performance Evaluation against FVE

| | Requirement | Subsystem | Performance |
|---|---|---|---|
| MN3 | Detect static obstacle of minimum size<br>1.5 m X 0.5 m & 2m X 2m | Obstacle detection | Successful within error margin of 20cm |
| | Detect obstacles of minimum size 1.5 m X 0.5 m in natural environment | Obstacle detection | Successful within error margin of 20cm |
| MN4 | Marker should be detected in 20cm to 20m range | Vision | Successful |
| | Manual flight control | Flight Control | Successful initially<br>Later compass problem |
| MF8 | Take coordinate as input from user | Flight Control | Successful initially<br>Later compass problem |
| MF9 | Communicate with ground platform to receive GPS updates | Flight Control | Successful initially<br>Later compass problem |
| MF3 | Waypoint Navigation | Flight Control | Compass problem |
| MN1 | Operate in outdoor environment | Obstacle Detection & Vision<br>Flight Control | Successful<br>Compass issue |

# Conclusion

| **Sub - System** | **Performance Evaluation** |
|---|---|
| Vision | Strong<br>Detect marker from far distance (20m) to very close distance (20cm)<br>No effect of lighting variations<br>Speed is high |
| Obstacle Detection | Neutral<br>Sensors give fluctuating readings at times<br>Not very precise |
| Flight Control | Weak<br>Need to resolve compass issue<br>And then achieve waypoint navigation |

# Work Breakdown Structure

| 1.2 | **Drone** |
|---|---|
| 1.2.1 | Choose Drone |
| 1.2.2 | Design Drone Underbelly |
| 2.1.1 | Procure Drone |
| 2.1.4 | Modify UAV for obstacle sensors |
| 2.1.5 | Fabricate underbelly |
| 3.2.2 | Test Drone R/C-only Control |
| 3.2.3 | Tune and test forward flight |
| 3.2.5 | Waypoint using hover |
| 3.2.6 | Waypoint using FF |
| 3.2.8 | Test Visual Landing of Drone |

| 1.4 | **Vision System** |
|---|---|
| 1.4.1 | Design Vision System |
| 2.4.1 | Procure Camera |
| 2.4.2 | Procure Vision Board |
| 3.3.1 | Test Camera and Board |
| 3.3.2 | Integrate and test Visual system on board |
| 3.3.3 | Test Visual Markers with Vision System |
| 3.3.4 | Integrate vision info into control |

| 1.5 | **Obstacle Avoidance** |
|---|---|
| 1.5.1 | Analyze and Layout Obstacle Avoidance Sensors |
| 1.5.2 | Design Obstacle Avoidance |
| 2.5.2 | Procure Obstacle Avoidance Sensors |
| 2.5.3 | Procure Optical Flow |
| 2.5.4 | PCB iterations |
| 3.4.1 | Integrate Optical Flow |
| 3.4.2 | Integrate and test obstacle avoidance system with drone |
| 3.4.3 | Table Test Obstacle Avoidance Sensors |
| 3.4.4 | Test Waypoint Following with Obstacle Avoidance |
| 3.4.5 | Test Visual Landing with Obstacle Avoidance |

| 1.7 | **User Interface** |
|---|---|
| 1.7.1 | Determine User Inputs/Outputs |
| 1.7.2 | Design User Interface |
| 2.7.1 | Build User Interface |
| 3.6.1 | Test User Interface |

| 1.8 | **Package Handling** |
|---|---|
| 1.8.1 | Design Gripper System |
| 1.8.2 | Design Package Modifications |
| 2.8.1 | Build/Procure Gripper |
| 3.7.1 | Test Gripper Electronics |
| 3.7.2 | Test Gripper and Package Modifications |
| 3.7.3 | Integrate and test gripper with drone |

# Schedule

- Major Milestones
  - Fix current UAV problems
  - Get AvA code communicating with peripherals
  - Get obstacle avoidance code initiated
  - Integrate all subsystems fully into the UAV
- We are slightly behind our original anticipated schedule but will still make the SVE

# Milestones

**#7 End January**
- WP Navigation (hover)
- PCB+ sensor integration
- NicaDrone bench top test

**#8 Mid February**
- Odroid+AvA code integration
- Forward flight tests

**#9 End February**
- Underbelly complete + sensors all mounted
- Optical flow code complete
- Obstacle avoidance code demonstrated on laptop

**#10 Mid March**
- Obstacle avoidance complete
- Ground control station interface complete
- Landing code for marker complete

**#11,12, SVE April**
- Testing of complete system before SVE

# Test Plan – Spring Validation Experiment

- Test B – Package carrying test
  - Validates packages can be carried and dropped by the UAV
  - Steps
    - UAV with package attached
    - Take off and hover for a minute.
    - UAV descends and lands
    - Drop package on the ground
  - Package should remain attached and released after landing

# Test Plan – Spring Validation Experiment

- Test E – Obstacle-less package delivery
  - Validates packages can be delivered without obstacles around the house
  - Steps
    - UAV + Package on a visual marker
    - System initiated by entering GPS coordinates of house
    - UAV takes off autonomously
    - Reaches waypoint near the house
    - Identify and navigate to visual marker
    - Land and drop package
    - Return back to another platform
  - Should be delivered within 2m of marker

# Test Plan – Spring Validation Experiment

- Test F – Package delivery with obstacles (uses E)
  - Validates packages can be delivered with static obstacles around the path
  - Steps
    - UAV + Package reaches near the house as before
    - Identify and plan path to visual marker
    - Place 2m x 2m obstacle in the path and beside the UAV
    - Repeat with 1.5m x 0.5 obstacle
    - Land and drop package and return
  - The UAV does not hit any obstacles

# Budget

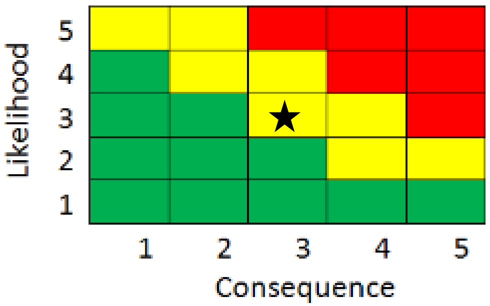
| Item | Projected budget |
| --- | --- |
| UAV + spares | $1288 |
| Vision system + optical flow | $575 |
| Obstacle Avoidance System | $670 |
| | $2533 |

- Spent $2,248 out of the projected $2,533
- Spent 56% of our allocated $4,000 thus far

# Risks Mitigated Already

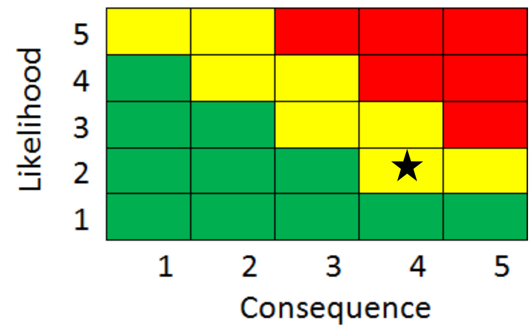
- Ordered the wrong batteries
- Odroid
- Analog sensors with long wires are noisy
- Number of sensors in obstacle detection too high

# Risk 1: Flight Dynamics



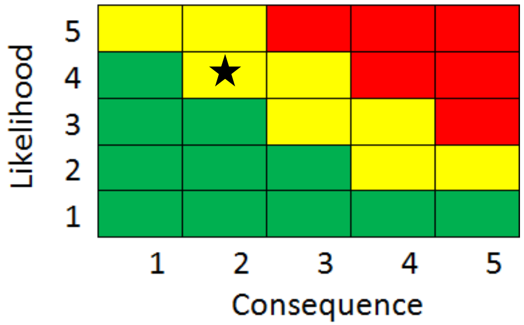

- Type: Technical
- Description:
  - Adding a payload and all the components will change forward flight dynamics.
- Consequence:
  - Forward flight might be inefficient and harder to control
- Mitigation:
  - Could add fairings (structure to increase aerodynamics) around our modifications to help with airflow.
  - Worst case we reduce maximum forward flight speed

# Risk 2: Styrofoam Structure




- Type: Technical
- Description:
  - –Vehicle is made of Styrofoam and modifications can't be undone
- Consequence:
  - –Structural support might be removed unnecessarily affecting flight dynamics and making the vehicle weaker
- Mitigation:
  - –Sensors placement is being optimized on a wood mockup of the vehicle
  - –Initial modifications will occur on the Styrofoam nose (costs $7 and modular)
  - –We have two frames (one is a spare)

# Risk 3: FireFly6 Firmware




- Type: Technical
- Description:
  - –Using proprietary firmware which isn't well tested but provides high value to project
- Consequence:
  - –Might create unforeseen coding and debugging issues
- Mitigation:
  - –Working with head firmware engineer of FireFly6 as second sponsor (in addition to UTRC).
  - –Test code immediately (already procured Pixhawk in anticipation of this problem)

# Conclusion - Fall Semester Lessons Learned

- Iterate Fast
- Be efficient.
  - Work to achieve good results and not best results
- Get spares for everything

# Conclusion - Key Spring Semester Activities

1. Finish critical components of the project
2. Understand AVA code
3. Integrate all subsystem – Vision, Obstacle detection into flight controller as soon as possible
4. Develop obstacle avoidance algorithm

# Questions?

# Thank You