

FlySense

Preliminary Design Review

Team C

November 2nd 2017

Shivang Baveja

Harikrishnan Suresh

Nick Crispie

Joao Fonseca

Sai Nihar Tadichetty

Augmented reality based pilot assistance system



- **Difficult to fly helicopter below 200m altitude, especially takeoff and landing**
 - Pilots rely mainly on their judgement while maneuvering in a tight space with obstacles
 - Standard instrument panel diverts their attention
- **Pilot assistance system which keeps pilot close to the reality by giving all the necessary information right in front of their eyes**
 - Display surrounding obstacles
 - Warn pilots about possibility of collision (escalating warning)

Four scenarios identified talking with NEA stakeholders

Take-off: Obstacle saturated environment



Pilot backs helicopter, yaws and climbs up at 6° ascend path angle (Vertical climb consumes a lot of fuel)

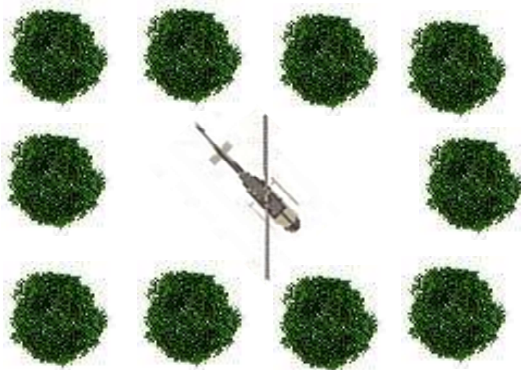
Landing: Isolated obstacle



While descending at a steep descent angle, Pilot cant see the obstacle as view blocked by fuselage

FlySense: gives visual and audio warnings about surrounding obstacles
gives sensor information right in front of pilot's eyes

Landing: Obstacle saturated environment



Pilot needs to steer the rotor away from obstacles while performing an almost vertical landing

Take-off: Isolated obstacle



While taking off initial climb angle cant be too large so pilots have to suddenly pull-up sometimes

Updated Functional and Performance Requirements

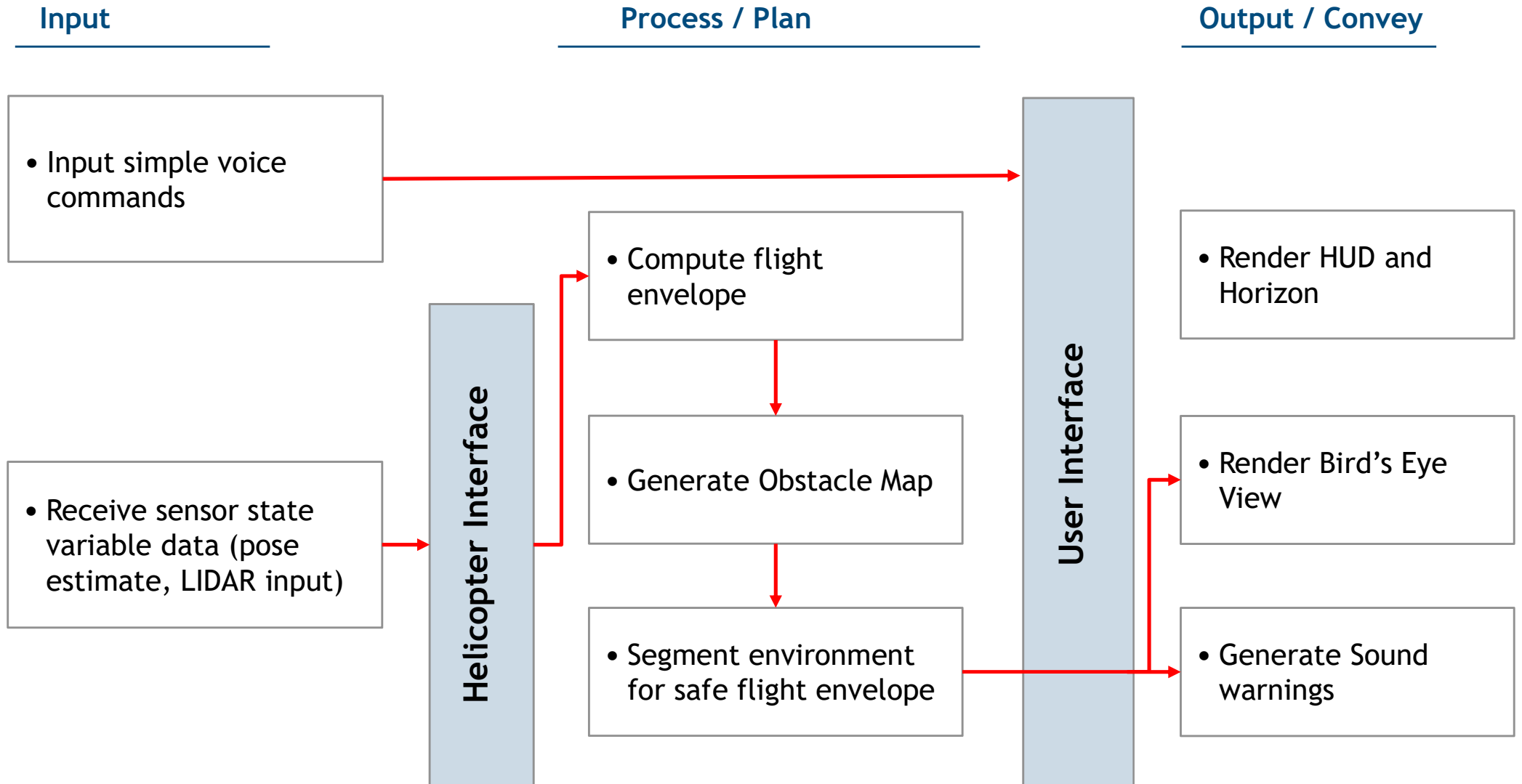
Feature	The system SHALL	Target Performance
Input	<ul style="list-style-type: none"> Receive sensor state variable data (pose estimate, LIDAR input) 	<ul style="list-style-type: none"> Process 1 Velodyne input real time
	<ul style="list-style-type: none"> Input simple voice commands 	<ul style="list-style-type: none"> 5 commands 80% recognition Works offline
Process / Plan	<ul style="list-style-type: none"> Compute flight envelope 	<ul style="list-style-type: none"> Project up to 5 seconds into future
	<ul style="list-style-type: none"> Generate Obstacle Map 	<ul style="list-style-type: none"> Latency <1s Objects of 2 m size, in distances < 20 m
	<ul style="list-style-type: none"> Segment environment for safe flight envelope 	<ul style="list-style-type: none"> Latency <1s
Output / Convey	<ul style="list-style-type: none"> Render HUD and Horizon 	<ul style="list-style-type: none"> >25 Hz refresh rate
	<ul style="list-style-type: none"> Render Bird's Eye View 	<ul style="list-style-type: none"> >25 Hz refresh rate
	<ul style="list-style-type: none"> Generate Sound warnings 	<ul style="list-style-type: none"> Closest object Different from helicopter warnings

In Green: Modified or Added

Updated Non-Functional and Performance Requirements

Segmentation	The system WILL	Target Performance
Installation	<ul style="list-style-type: none"> • Easily be integrated with flight interfaces • Leverage pre-existing flight sensors • Be easy to setup (hardware and software) 	<ul style="list-style-type: none"> • Works with NEA flight system (SVE)
Interaction with Pilot	<ul style="list-style-type: none"> • Feel natural to the pilot • Be easy to put/remove headwear • Be comfortable to wear headwear for long periods of time 	<ul style="list-style-type: none"> • Focal distance up to 20 meters • Wearable like normal glasses • Weights less than 1 pound
Information Displayed	<ul style="list-style-type: none"> • Be clear and simple • Be non intrusive to the pilot • Be non distracting for the pilot 	<ul style="list-style-type: none"> • Focus group with 3 pilots using solution
Other criteria	<ul style="list-style-type: none"> • Be substantially more affordable than available solutions (e.g. fighter jet pilot helmets) 	<ul style="list-style-type: none"> • Solution hardware cost below USD 5,000

In Green: Modified or Added

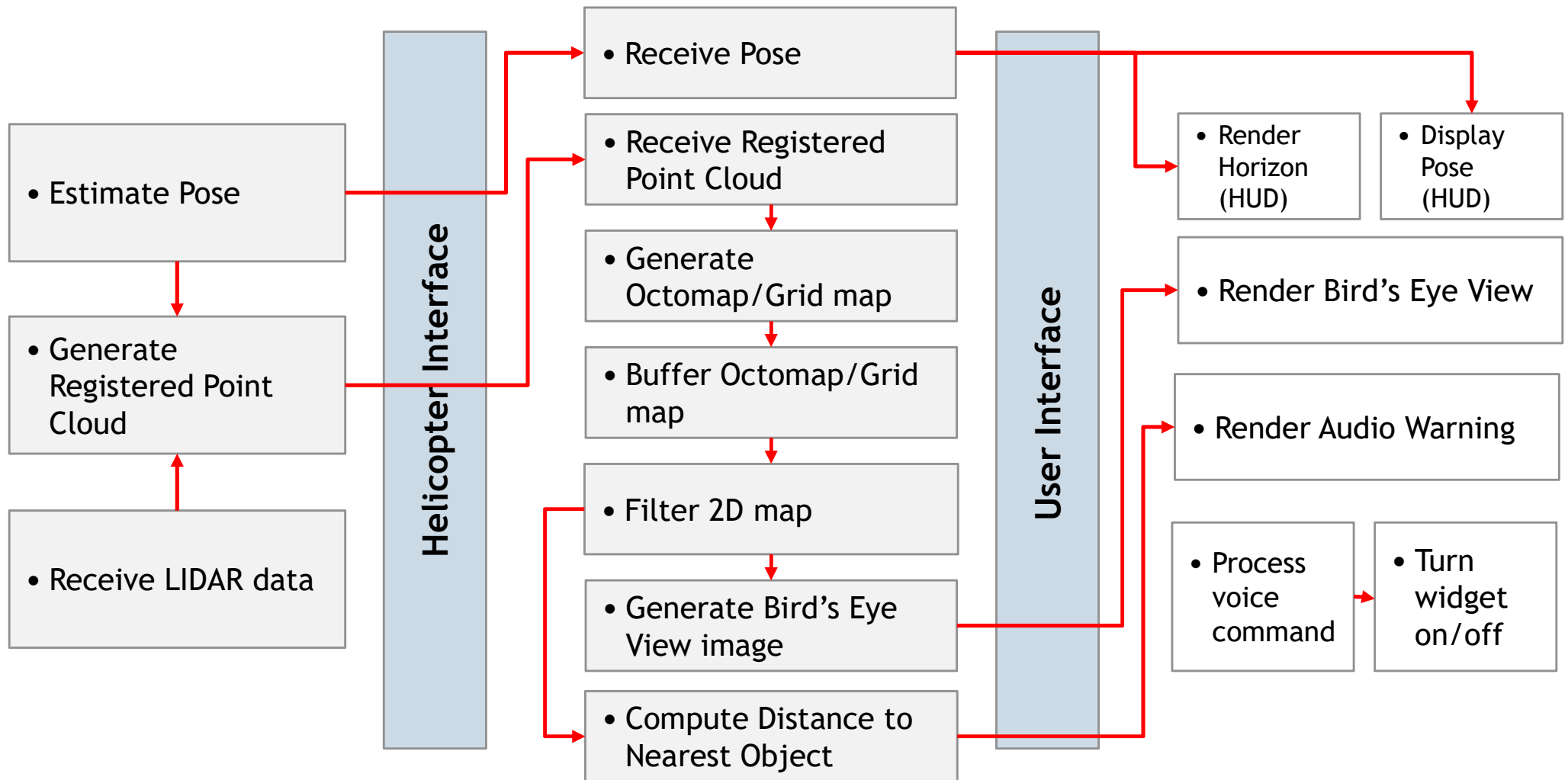


→ ROS □ Android □ Linux

NEA Flight System

Jetson TX2 (Sensing)

Epson BT-300 (UI/UX)



The FlySense system is divided into Aerial and User Systems

Aerial System: Mapping + Onboard computer + PDB



- DJI Matrice 100 mounted with Jetson TX-2 and Velodyne VLP16 Puck, Power Distribution Board
- NEA Helicopter with Jetson TX-2 connect through LAN cable (uses ROS)

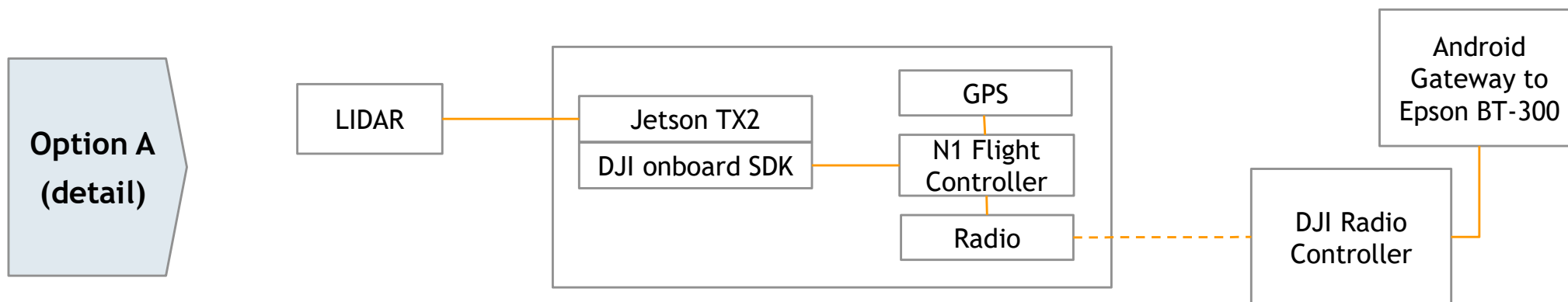
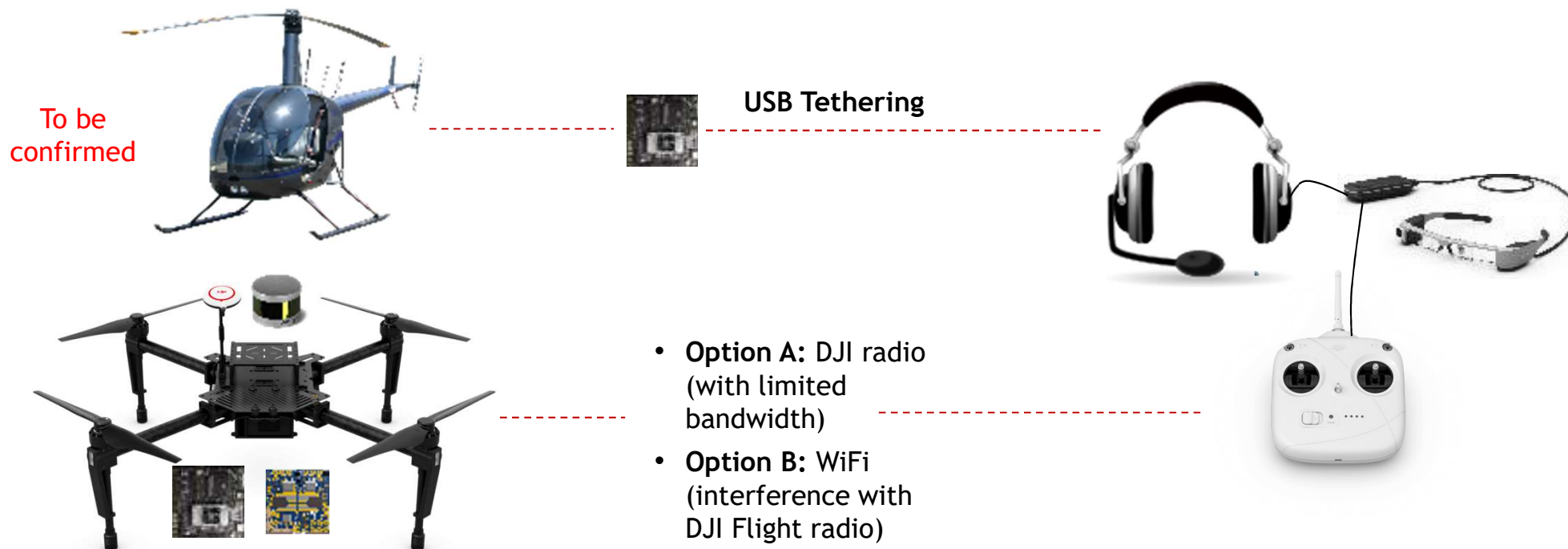
User System: Augmented Reality interface (UI/UX)

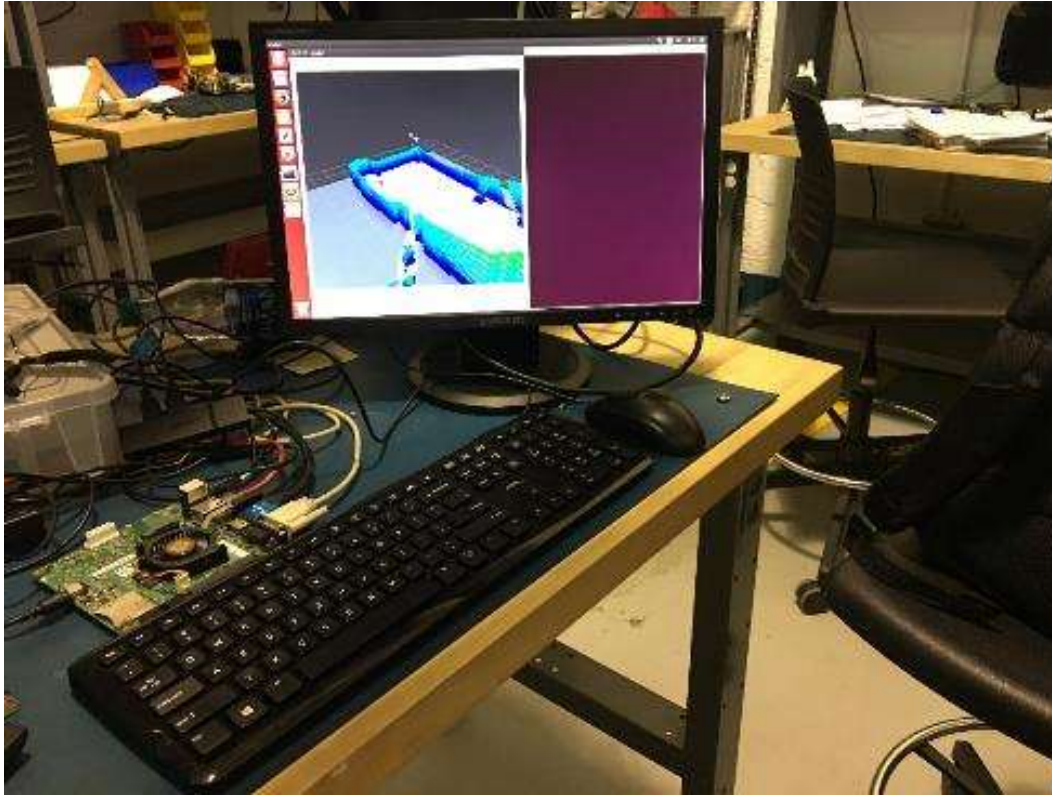


- Epson BT 300 AR headset communicates to onboard computer
- Headphones for audio warnings and voice command recognition

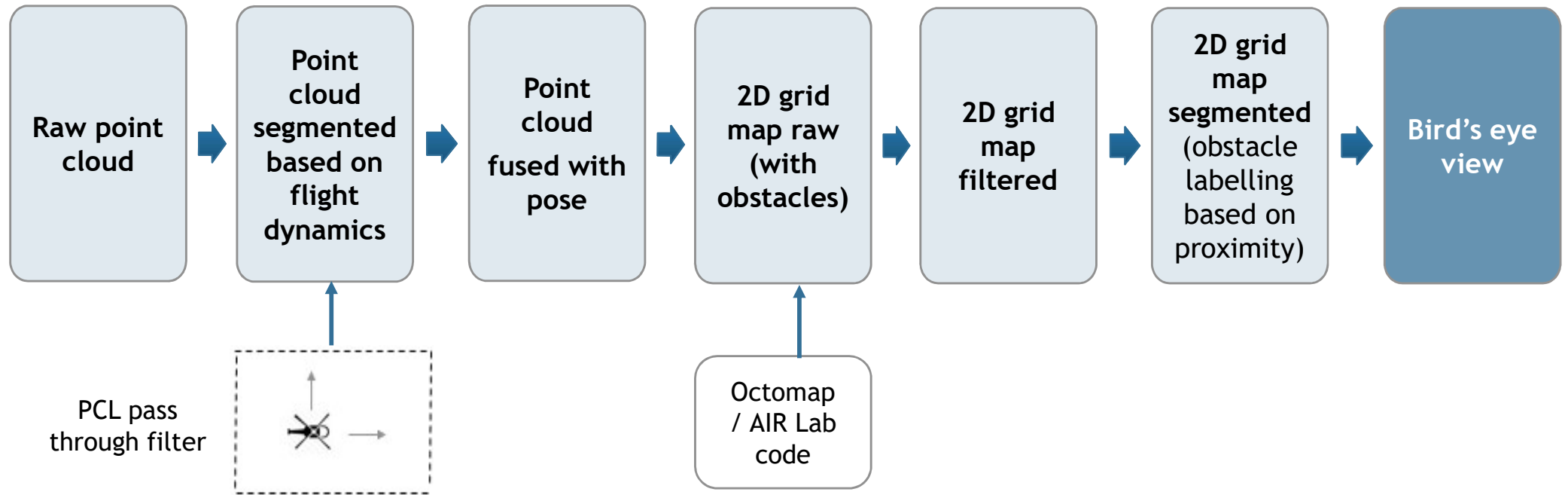
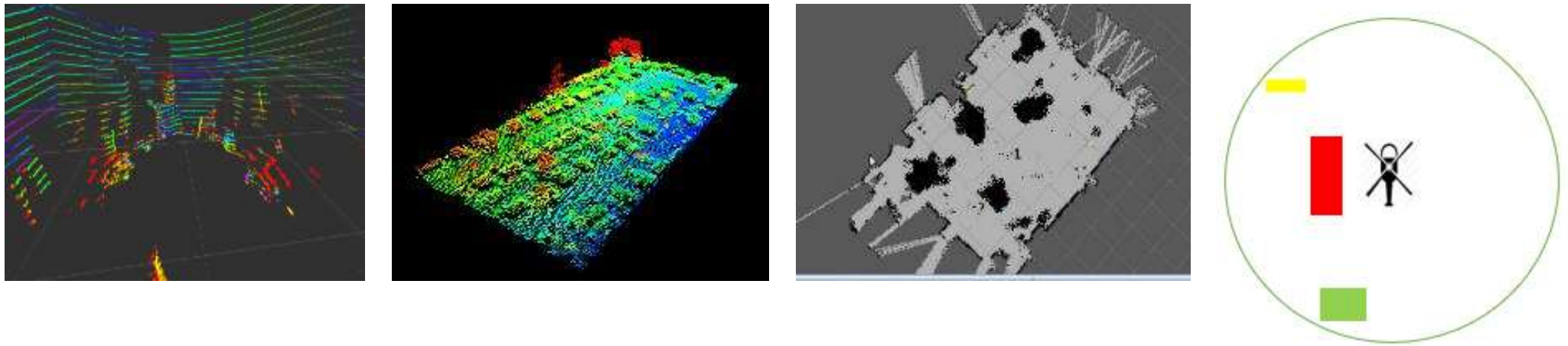
Aerial System

User System





- All the components have been procured except Jetson TX2 (working with Jetson TK-1 for now)
- Mapping v1 is working on Jetson
- Communication interface specification completed
- Work going on to interface Jetson to the Epson AR using DJI SDKs



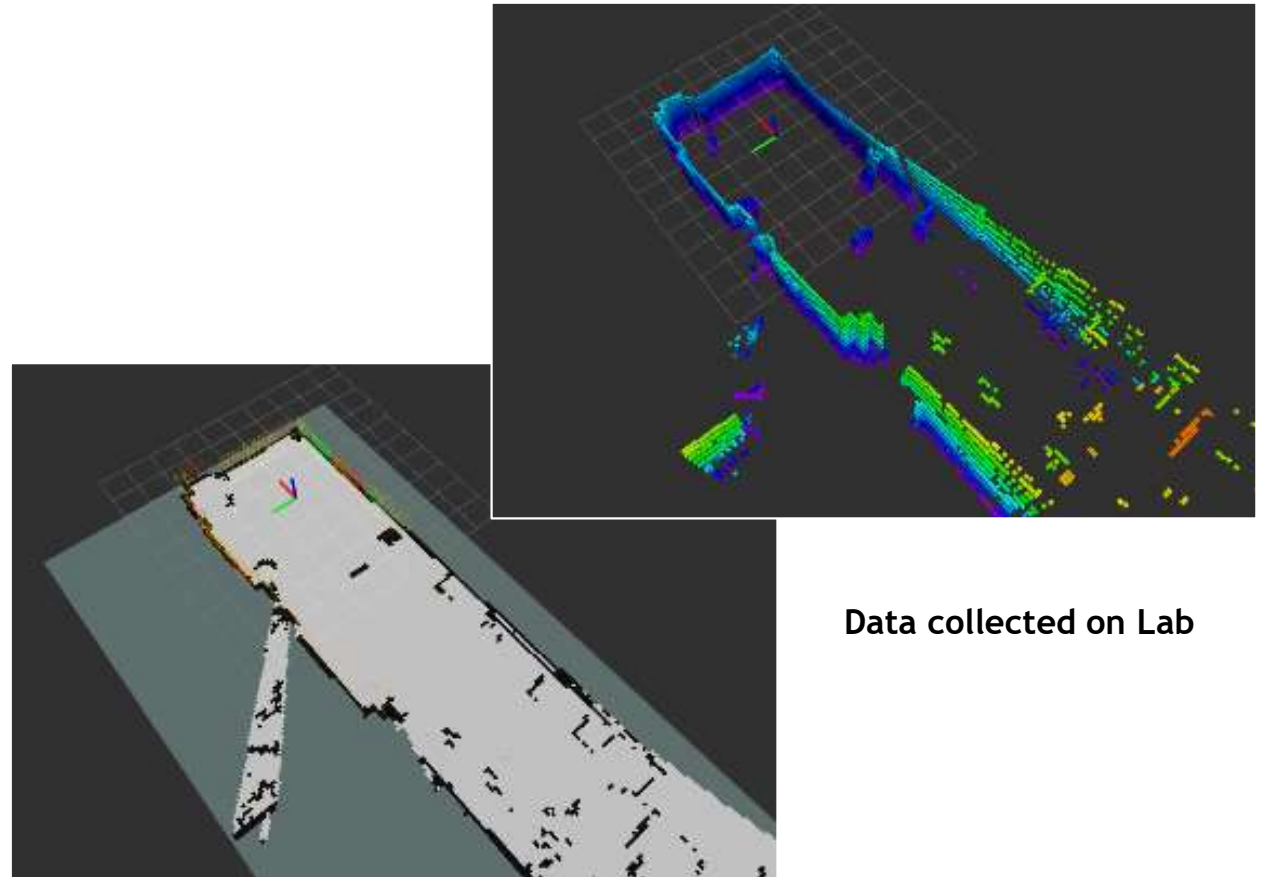
Note: Proximity labelling is made using time for impact (not distance)

Completed work:

- 3D mapping using the Octomap package in ROS
- Optimization of Octomap to improve update frequency of Octomap, increase sensor range and remove outliers on the ground

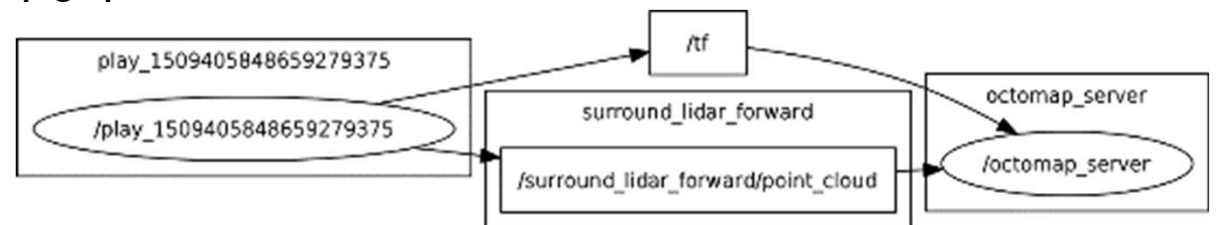
Future work:

- Segmentation of raw LIDAR data based on flight dynamics using PCL pass through filter
- Generate 2D obstacle occupancy map - raw, filtered and segmented
- Generate bird's eye view (ROS output* - 2D image / matrix with 0s,1s,2s and 3s)



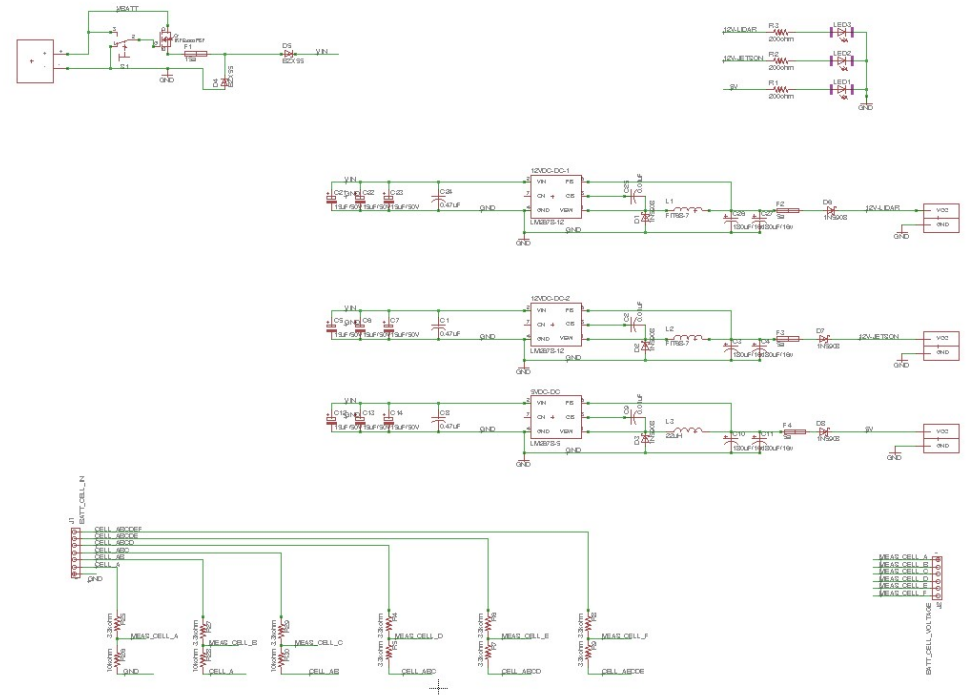
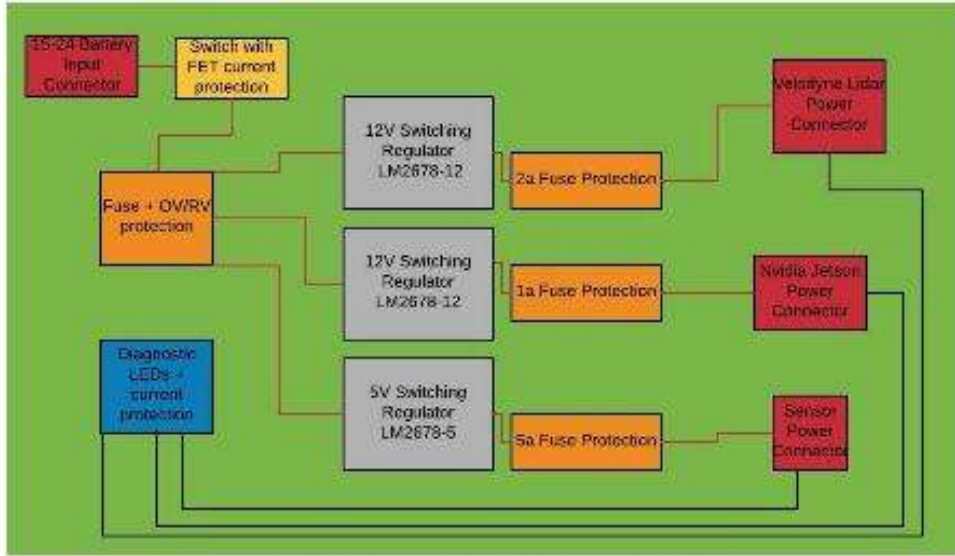
Data collected on Lab

Rqt graph for NEA offline data



* To be decided

PCB Concept and Schematic for Power Distribution Board

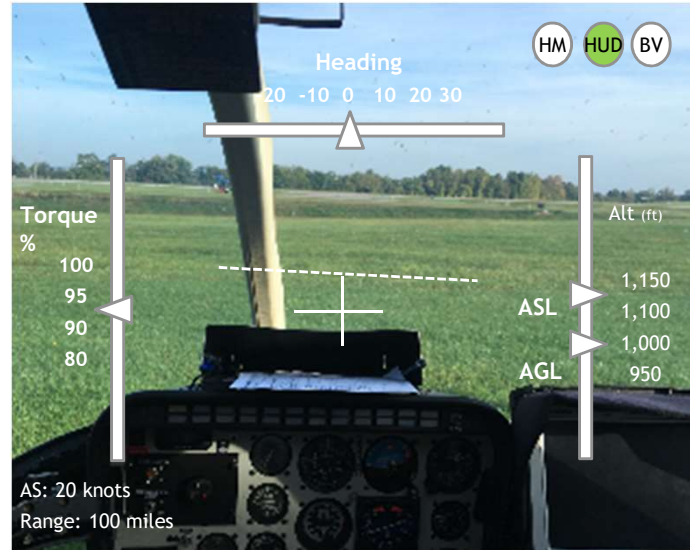


- 3 power lines
 - LIDAR (12V)
 - Jetson (12V)
 - IMU/GPS sensor (5V)
- Input is 6 cell Li-ion Battery (approx. 24V)



www.futurlec.com

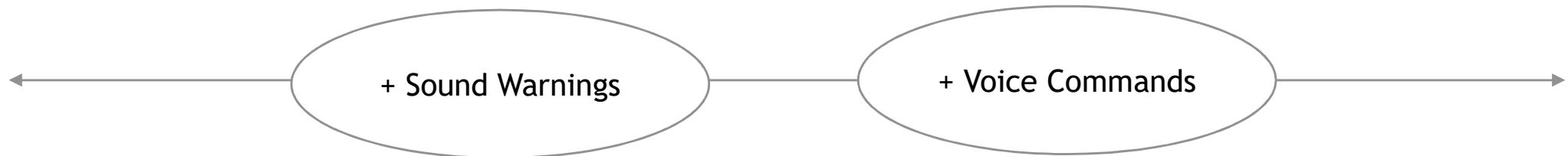
Requirements downsized to three basic features: “User interface (including voice and sound)”, “standard instruments” and “birds’ eye view”



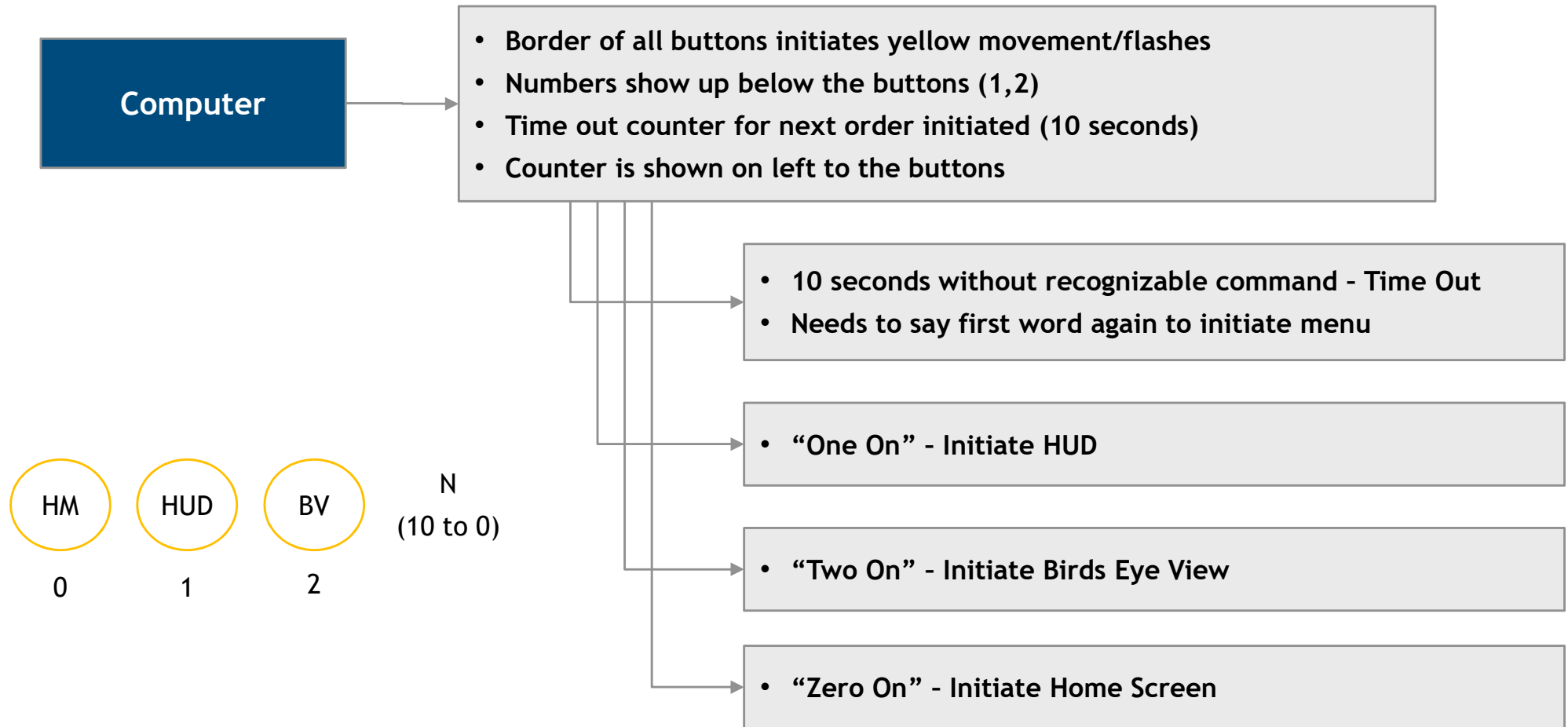
- Default is no AR display on to prevent cluttering pilot’s view

- Information from sensors (heading, altitude, avg altitude above ground, Torque, avg speed)

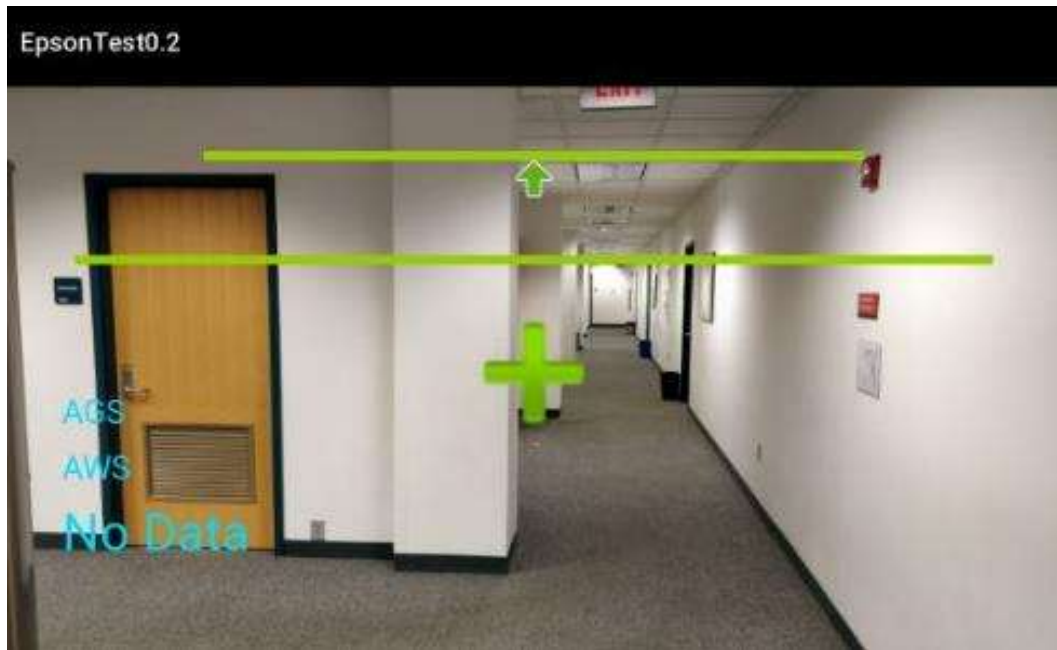
- Obstacle inside widget with scale of the circle indicated on the side



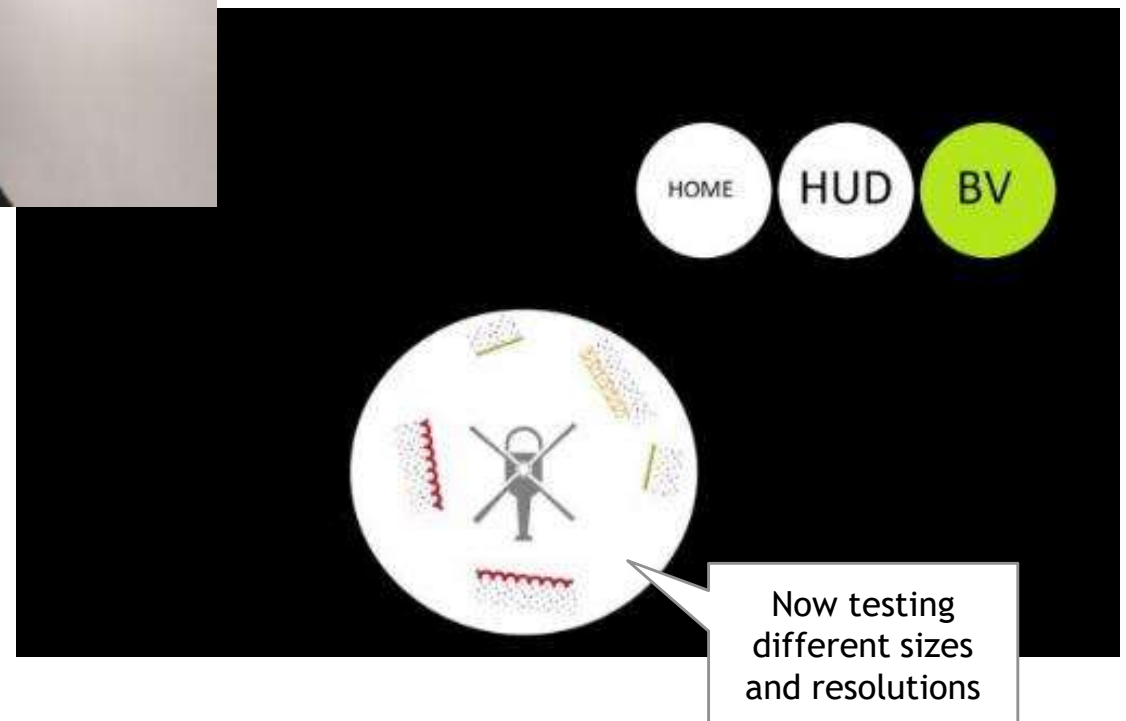
	Description	Current Status
Voice Commands	<ul style="list-style-type: none"> • Voice operated menu tree for pilots to toggle through different modules 	<ul style="list-style-type: none"> • Command tree logic ready • Initial research done (identified Pocket Sphinx library to interpret voice commands in noisy environments)
Visuals	<ul style="list-style-type: none"> • Home screen with three buttons • HUD with standard instruments • Bird's Eye view with 2D obstacle map, coloured as function of time to impact 	<ul style="list-style-type: none"> • Mock-up of home screen ready • Mock-up HUD ready with roll, pitch, yaw • Bird's Eye view - 2D obstacle map in progress
Sound Warnings	<ul style="list-style-type: none"> • Sound warnings with escalating square wave sound as nearest object becomes nearer 	<ul style="list-style-type: none"> • Initial research done (identified Android libraries to push differentiated sound to left and right channels, and to choose a frequency of signal)



HUD view overlaid on reality



Bird's Eye View in the developer emulator





Note: Black in the developer emulator is transparent in the HUD.

PROJECT MANAGEMENT: WORK BREAKDOWN STRUCTURE

	Aerial Subsystem	User Subsystem	FlySense Validation
Algorithm	<ul style="list-style-type: none"> • 3D mapping • Flight envelope • 2D obstacle map 	<ul style="list-style-type: none"> • Sound warnings • Speech recognition (with noise cancellation) 	<ul style="list-style-type: none"> • N/A
Software	<ul style="list-style-type: none"> • Interface to AR • Interfacing with sensors • Mapping implementation • Bird's eye view image generation 	<ul style="list-style-type: none"> • Generate Sound Warnings • Render Bird's Eye view • Render Std. Instruments • User Interface (Buttons, pop-up, speech) 	<ul style="list-style-type: none"> • N/A
Hardware (procure, setup, test)	<ul style="list-style-type: none"> • PDB • LIDAR • Jetson • INS-GPS 	<ul style="list-style-type: none"> • Augmented Reality Headset up and running (Epson) 	<ul style="list-style-type: none"> • Flying Quadcopter with sensors • Jetson to Helicopter computer
Integration	<ul style="list-style-type: none"> • Jetson -> AR • Jetson+LIDAR • Jetson+GPS-INS 	<ul style="list-style-type: none"> • Jetson communication protocol • AR -> Jetson 	<ul style="list-style-type: none"> • Integrate with NEA LIDAR Datasets • Integrate with Quadcopter • Integrate with NEA Flight System
Testing	<ul style="list-style-type: none"> • LIDAR Static • LIDAR Moving • GPS/INS test • PDB test 	<ul style="list-style-type: none"> • AR connected with PC • AR connected with Jetson 	<ul style="list-style-type: none"> • Test with NEA LIDAR dataset • Test with Quadcopter • Test with NEA Flight System

Subsystem-level schedule for Fall semester

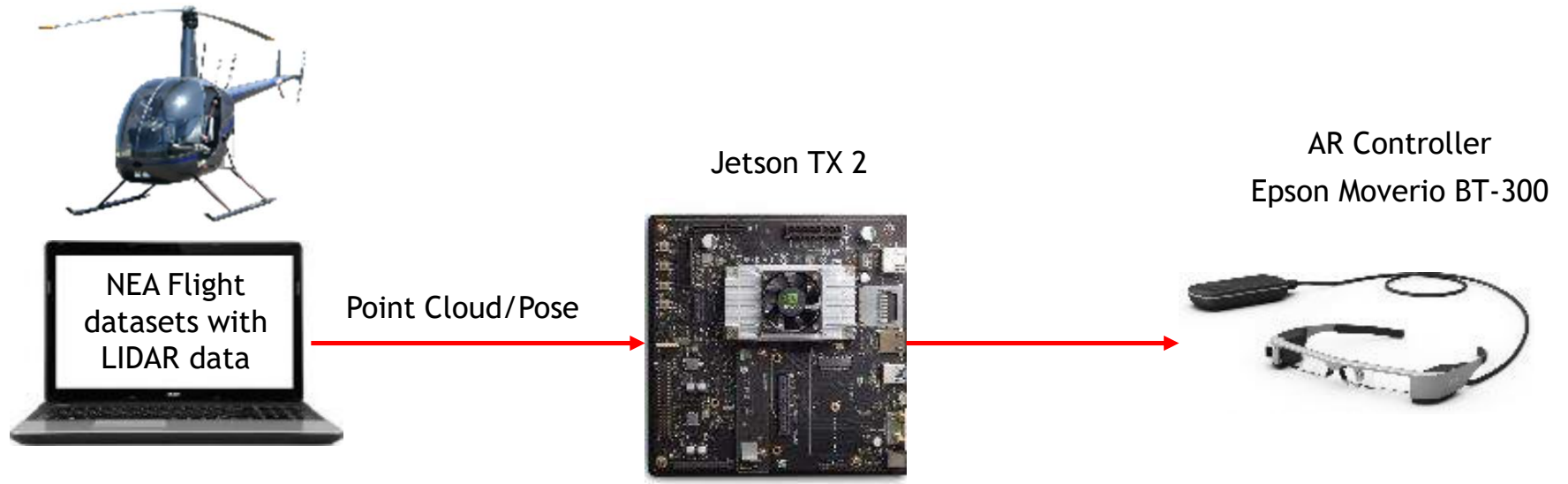
Tracks	PR 1 20 Oct 2017	PR 2 27 Oct 2017	PDR 31 Oct	PR3 Nov 10	PR4 Nov 22	PR5 Nov 30	PR6 Dec 7	Critical Review Dec 14
User Sub-system	First visualization of sensor suite ✓ Specification to onboard computer ✓	Mock-up of HUD ✓	Mock-up of Bird's Eye View 	Mock-up of sound	Head tracking for artificial horizon	HUD and Bird's Eye View with NEA dataset	Additional testing and refinement	Planning for next phase
Aerial Sub-system	Point cloud of Lidar Data set ✓	Onboard obstacle map generation ✓	Obstacle map with NEA flight dataset 	Obstacle map with benchtop hardware	Interface to AR			
System Int. & Testing	Procurement of Epson BT-300, Velodyne and Jetson ✓	Design of Fall verification experiment ✓	Integration and testing of HUD		Integration and testing for Fall Validation Experiment (HUD, BDV, Sound)			

 Currently on track, but potentially could run into delays as dependencies start to become a larger factor

Milestone	Desired Functionality	Test Method
Progress Review #3	<ul style="list-style-type: none"> • 3D to 2D map on Jetson • Birds Eye View on AR • HUD on AR • First version of audio warnings • First version of Speech recognition 	<ul style="list-style-type: none"> • Visuals generated from previously collected data • Use data to render Bird's eye view on AR head set
Progress Review #4	<ul style="list-style-type: none"> • Mapping performed on Jetson with data being passed in real time • Jetson interface to User Interface complete 	<ul style="list-style-type: none"> • Generate map from obstacles placed in open area
Progress Review #5 /#6	<ul style="list-style-type: none"> • Bird's Eye view with simulation data with binary LR sound and visual warnings synchronized 	<ul style="list-style-type: none"> • NEA datasets + benchtop hardware • Benchtop hardware on Quadcopter manually moved around local obstacles

Test 1

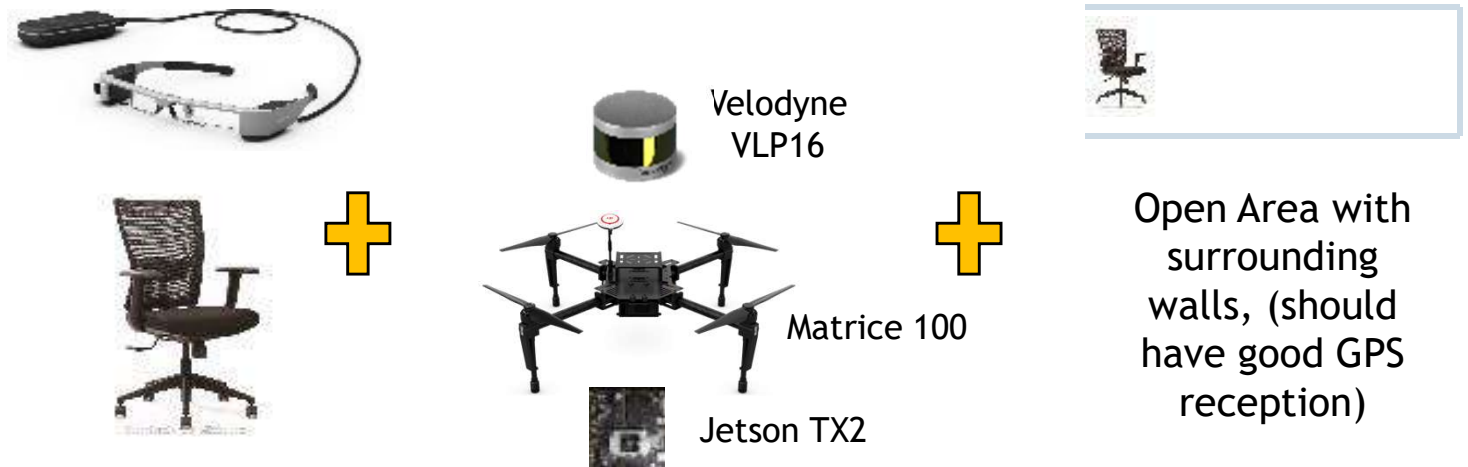
Location:
Inside
Robo-
lounge



Test 2

Location:
outside
Robo-
lounge

A person
wearing AR
headset, while
sitting in a chair
(quadcopter
mounted on it)



Test 1 Procedure	Performance Evaluation
1. Pilot puts on the Headset and checks widgets (HUD, BV) are working by giving voice commands	<ul style="list-style-type: none"> • 5 commands • 80% recognition • Works offline
2. Pilot enables HUD by voice command. The HUD shows Std. instrumentation	<ul style="list-style-type: none"> • Latency <1s • >25 Hz refresh rate
3. Start simulation, aircraft takes-off (shown in RVIZ as a frame)	<ul style="list-style-type: none"> • Objects of 2 m size, in distances < 20 m
4. Aircraft flies near wall, Bird's Eye pops-up in AR headset	<ul style="list-style-type: none"> • Project up to 5 seconds into future
5. As aircraft goes closer to wall, color of wall changes from yellow to Red. Pilot hears audio warning in L and/or R ears.	<ul style="list-style-type: none"> • Sound warning (different from helicopter warnings)
6. Aircraft moves away from the wall. Color label of the wall is changed from Red to Yellow. Audio warnings go away.	
7. While flying in direction parallel to wall, wall is labelled yellow (aircraft is not going towards the wall)	

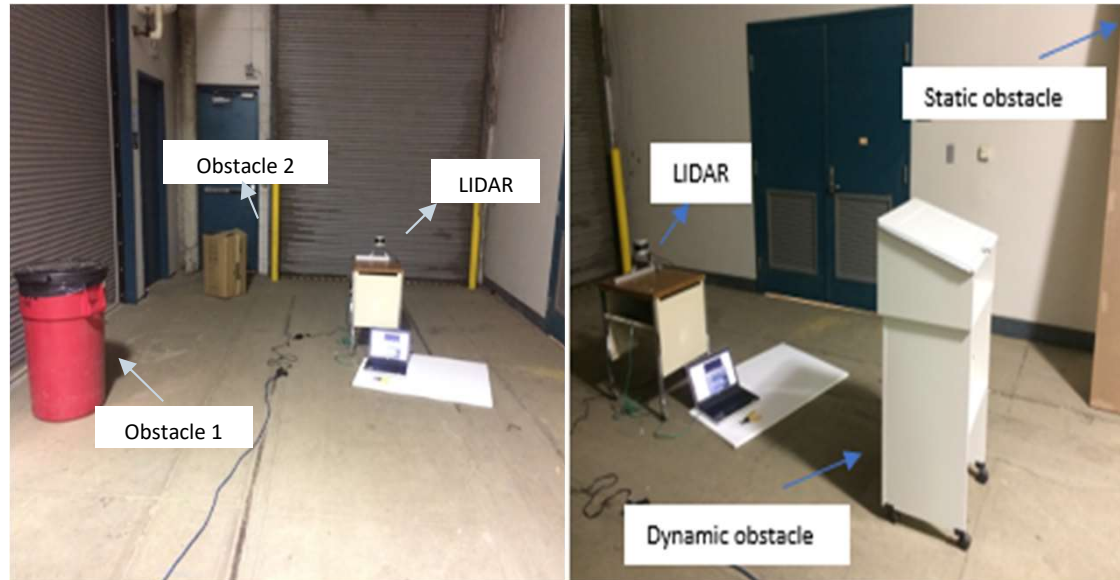
Test 2 Procedure	Performance Evaluation
1. Pilot sits on the chair and puts on the Headset	
2. Pilot enables HUD by voice command. The HUD shows Std. instrumentation	<ul style="list-style-type: none"> • 5 commands • 80% recognition • Works offline
3. As pilot moves the chair near the left wall and bird's eye pops in AR headset	<ul style="list-style-type: none"> • Process 1 Velodyne input real time • Latency <1s • Objects of 2 m size, in distances < 20 m • Closest object
4. As pilot moves the chair closer to wall, color of wall changes from yellow to Red. Pilot hears audio warning in L ear.	<ul style="list-style-type: none"> • Project up to 5 seconds into future
5. As pilot moves the chair away from the wall. Color label of the wall is changed from Red to Yellow. Audio warnings go away.	
6. As pilot moves the chair parallel to wall, wall is labelled yellow	
7. Pilot moves away from walls and switches off Bird's Eye View by giving audio command. Pilot enables HUD again	
8. Pilot comes back to the home location and removes the headset.	

Completed work:

- Tests outside MRSD Lab to evaluate the performance of Velodyne PUCK.
- Scenarios: Environment with only static obstacles, static obstacles and one moving obstacle
- Procured Platform DJI Matrice 100 and associated components from inventory

Future work:

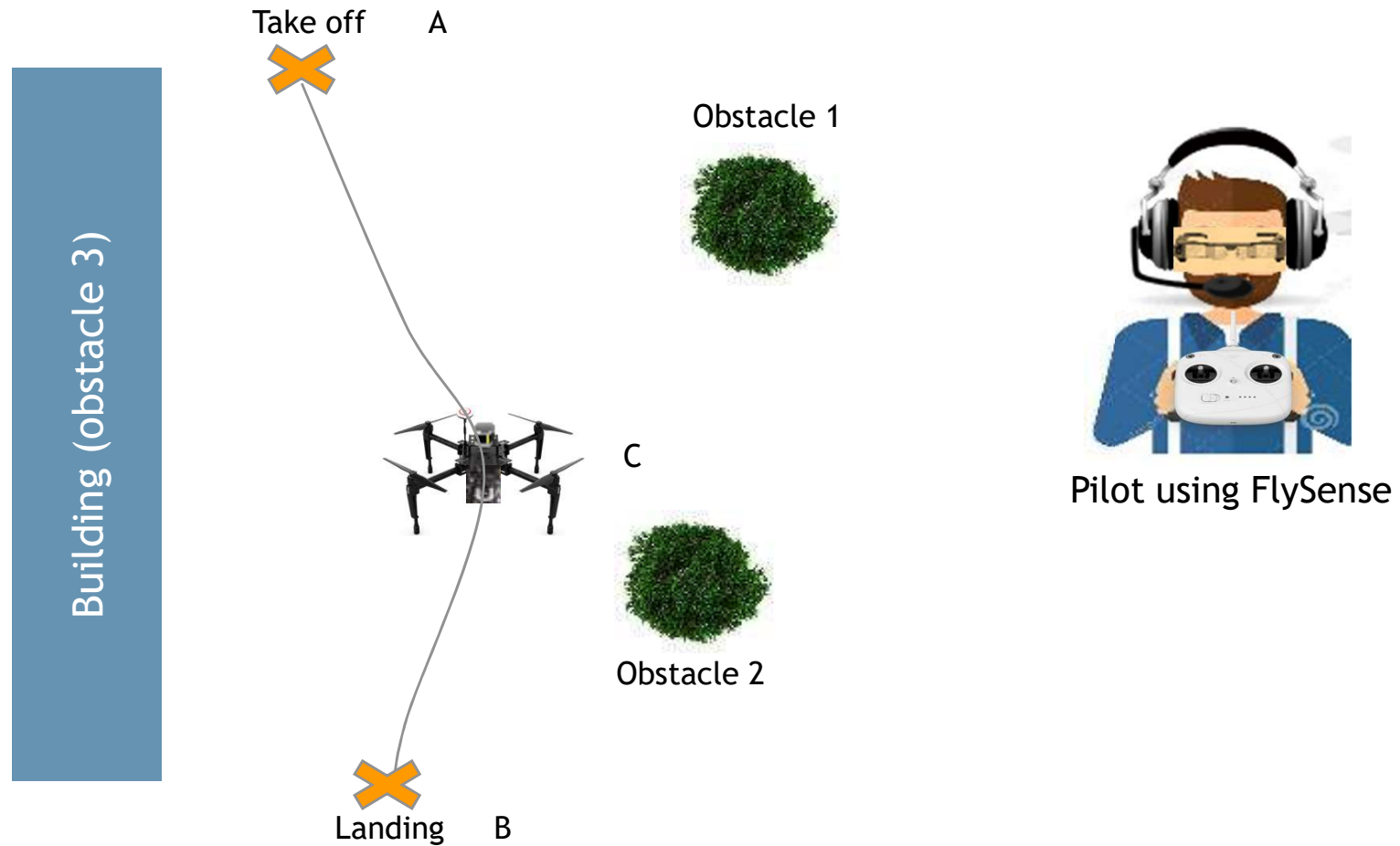
- Order final On-board computer (Jetson TX2)
- Design and fabricate mounts for LIDAR, On-board computer
- Final integration of components on DJI Matrice 100



Milestone	Desired Functionality	Test Method
January	<ul style="list-style-type: none"> • Quad running with remote commands • Standard Suite of Instruments in AR 	<ul style="list-style-type: none"> • Fly quad in open outside area • Live data transmitted to AR glasses
February	<ul style="list-style-type: none"> • Real time LIDAR processing from Quad • Mapping refined to accommodate live data • User interface shows full capability 	<ul style="list-style-type: none"> • Fly Quad with LIDAR, display live obstacle mapping on computer (rViz) • Round one of user feedback from focus group
March	<ul style="list-style-type: none"> • Switching of modes with voice commands with background noise • Smaller obstacles detected and mapped at faster speed 	<ul style="list-style-type: none"> • User test with AR glasses and helicopter background noise • Round two of user feedback from focus group
April	<ul style="list-style-type: none"> • Full Flight demonstration with Quadcopter and ground based pilot with AR headset 	

Test location

Open area at CMU with good GPS signal, normal wind, and few obstacles
(Tentative: Outside Robo Lounge)



Test Procedure	Performance Evaluation
1. Pilot wears the FlySense system, in default Home screen / Mode 1	<ul style="list-style-type: none"> Light weight and comfortable to wear
2. Gives voice commands “Computer”, “One on”, “Two on” and “Zero on” to check before flight	<ul style="list-style-type: none"> Recognize 5 voice commands with 80% accuracy
3. Gives command in RC for Quadcopter to take off (point A)	
4. Gives voice command “One on” to get HUD	<ul style="list-style-type: none"> Render HUD with refresh rate > 25Hz
5. Pilot flies quadcopter to desired altitude with payload and hovers	<ul style="list-style-type: none"> Carry the payload to a height of 2m
6. Gives voice command “Two on” to get Bird’s eye view	<ul style="list-style-type: none"> Render Bird’s eye view with refresh rate > 25Hz
7. Pilot flies Quadcopter towards landing zone (point B)	

Test 1 Procedure	Performance Evaluation
8. Obstacles 1 and 3 appear in green in AR interface (check slide 24)	<ul style="list-style-type: none"> Show obstacles of 2m size in distance < 20m with latency <1s
9. On reaching point C, obstacle 2 appears in Red in AR interface and loud beeps are heard in the left ear	<ul style="list-style-type: none"> Compute flight envelope 5s into future Segment environment with latency <1s Generate sound based on obstacle location
10. Gives command in RC to steer the quadcopter away from obstacle 2	
11. Obstacle 2 now turns yellow and beeps stop	<ul style="list-style-type: none"> Compute flight envelope 5s into future Segment environment with latency <1s Stop sound based on obstacle location
12. On reaching point B, Pilot gives command in RC to land the quadcopter	
13. Gives voice command “Zero on” to switch back to home screen and removes FlySense	

PROJECT MANAGEMENT: BUDGET

Type of Budget item	Supplier	Description	Model no.	Unit cost	Units to purchase
Borrowed Equipment	NEA	LIDAR	Velodyne Puck VLP-16	8,000	0
Borrowed Equipment	MRSD Lab	Quadcopter ("old" frame)	DJI matrice 100	2,847	0
Confirmed Budget	Amazon	AR Headset	Epson BT-300	799	1
Confirmed Budget	Amazon	FlySense On-Board Computer	Jetson TX2	559	1
Confirmed Budget	DJI	Flight Controller (IMU + GPS)	DJI N3	319	1
Borrowed Equipment	MRSD Lab	FlySense On-Board Computer	Jetson TK1	200	0
Confirmed Budget	MRSD Lab	Magnetometer	SparkFun HMC5883L	2	0
Projected Budget	DigiKey	Electronics Components	Various	50	N/A

TOTAL Cost \$12,816

Spent/Earmarked Amount \$1,767

Remaining Funds \$3,233

In Red: To be ordered this week.

In Green: Pending confirmation whether or not it will be needed

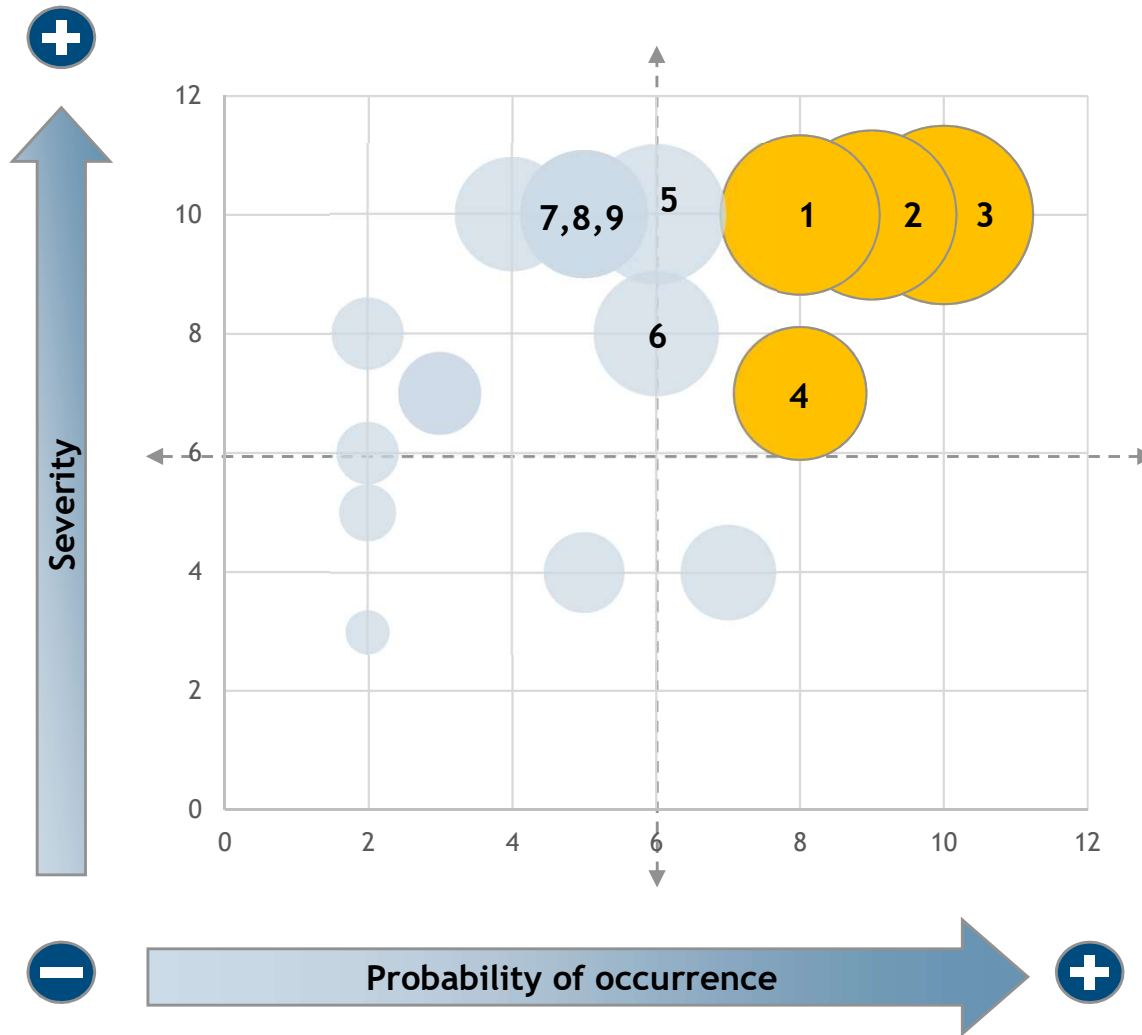
PROJECT MANAGEMENT: DETAILED RISKS (1/2)

Risk item	Description	Consequences	Causes	Mitigation Strategies	Occ	Sev	RPN	Risk Owner
CPU performance	CPU cannot handle LIDAR data	Cannot meet requirements	Processor speed/RAM	Purchase Jetson TX2	10	10	100	Nick
Octomap	Octomap lags when processing LiDAR data in real time	Cannot meet latency functional requirement	Octomap is too slow	Decrease resolution of the occupancy grid, prioritize processing in the region of interest	9	10	90	Hari
LIDAR	LIDAR data is not processed correctly	2D maps generated are not adequate to be shown to the pilot		Test different processing techniques to generate the 2D maps	8	10	80	Hari
LIDAR	LIDAR data is too coarse to do effective obstacle mapping	Cannot generate effective obstacle map, impacting requirements		Buffer data in the obstacle map to be able to capture smaller relevant obstacles	8	7	56	Hari
UI/UX voice commands	Voice commands do not work in varying degrees for giving information to user interface	Cannot change modes in flight, do not meet some functional requirements	Voice command detection not robust to helicopter noise	Use only a few words, multiple mics, use Android system to avoid rewriting code	5	10	50	Joao
Team overworked in other classes	Have too many requirements to meet for FVE and SVE	Objective of project not met	Overambitious targets, lack of proper planning, underestimation of work	Limit mandatory requirements, prioritize or consolidate requirements	6	10	60	Nick
AR Headset	AR headset not robust to changes in light	Cannot meet key functional requirements	Not designed for helicopter lighting conditions	Use Epson shades, use Epson camera to dynamically manage color scheme	5	10	50	Nihar
AR development	Can't get full user interface working on headset	Cannot meet key functional requirements	Difficult API, scheduling	Start with a limited mock-up, get buy in on new additions, gradually deploy them	5	10	50	Nihar
AR Headset	AR headset gives pilots headaches	Can't meet key non-functional requirements	Humans are not evolved enough	Simplify interface to avoid nausea, check if the feeling disappears with training	5	10	50	Nihar
Limits on real-flight testing	Weather prevents flight testing	Unable to perform validation experiments	Pressure differences, rain, God, etc.	Plan for videotaped FVE	6	8	48	Joao
UI testing	limits on human ui testing	Delay in development schedule, incomplete solution impacts non-functional requirements	Human testing regulations	Test on ourselves, don't collect personal data, prepare test procedure and submit it to the university	2	3	6	Joao

PROJECT MANAGEMENT: DETAILED RISKS (2/2)

Risk item	Description	Consequences	Causes	Mitigation Strategies	Occ	Sev	RPN	Risk Owner
Integration	Project does not work when we put it all together	Cannot meet key functional requirements	Lack of planning, non-robust component parts	Unit testing, start early	4	10	40	Nick
NEA	NEA data not useful	Unable to build accurate simulation/ model to base user interface off of	Available dataset does not include tail rotor data, data is too coarse	Include datasets from both Helicopters and Quadcopters	7	4	28	Hari
CPU development	Difficulty in working with CPU SDK	Delay in development, potentially can't meet requirements in FVE/SVE	Lack of OEM support	Work with other teams, pair up on development, recruit help from NEA/other CMU	3	7	21	Shivang
Testing	Flight hardware breaks during testing	Can't test or perform FVE/SVE	User error, weather	Do not test in bad weather, use NEA quadcopter and record videos for FVE/SVE.	3	7	21	Shivang
LIDAR	LiDAR data takes up too much space	Cannot process too much data because of storage limits	Not enough disk space on a computer	Buy an external drive for storing data for offline processing	5	4	20	Hari
3D sound development	Cannot map sound to 3D effectively	Does not meet non-functional requirements		Binary LR sound	2	8	16	Joao
Lose a team member sickness/personal	Team member is unavailable due to injury, sickness, personal matter, etc	Reduced work force	Sickness, team difficulties	Team up on many tasks, have a lead and second for each functional area	2	6	12	Nick
LIDAR	Velodyne Lidar is too heavy to put on a drone	Can't test or can't test as long as we would like	Velodyne VLP-16 is too heavy	Increase motor torque (and batteries?)	2	5	10	Shivang
UI testing	Limits on human ui testing	Delay in development schedule, incomplete solution impacts non-functional requirements	Human testing regulations	test on ourselves, don't collect personal data, prepare test procedure and submit it to the university	2	3	6	Joao

We are currently deploying mitigation actions with owners to address the key risks identified



Risk mitigation strategies

Obstacle Map generation (Shivang/Hari)

1. Purchase Jetson TX2
2. Decrease resolution of the occupancy grid, prioritize processing in the region of interest
3. Test different processing techniques to generate the 2D maps
4. Buffer data in the obstacle map to be able to capture smaller relevant obstacles

Team overworked in other classes (Nick)

5. Limit mandatory requirements, prioritize or consolidate requirements

UI/UX voice commands (Joao/Nihar)

6. Bird's eye pops-up automatically, 2 microphones for noise cancellation + Android offline library

AR Headset (Nihar)

7. Use Epson shades, use Epson camera to dynamically manage color scheme
8. Start with a limited mock-up, get buy in on new additions, gradually deploy them
9. Simplify interface to avoid nausea, check if the feeling disappears with training

Note: 19 risk identified so far

Thank You!

Backup

Data struct used for communication with the Jetson

```
struct DataPacket
{
    time timestamp;           // Hour, minutes seconds, milliseconds
    string messageType;      // Type of data packet (e.g. "HandShake")
    uint8 header;            // should always be 0xFE
    uint8 length;            // set to sizeof this struct
    uint8 deviceId;          // who is talking
    int8 StateID;            // what state are you talking about
    int8 StateSource;        // who changed the state (multiple possibilities)
    blob Image;              // Image for bird's eye view (if applicable)
    int8 ImageSize;          // Number of pixels in image
    float engineTorque;      // Engine Torque (0 to 100)
    float AGL;               // Average Ground Level (feet)
    float ASL;               // Average Sea Level (feet)
    float AS;                // Average Speed (knots)
    float range;             // Range left (feet)
    float HeliPose[3];       // Degrees (3D vector)
    int 3DSoundFreq;         // SoundFreq for 3D sound
    int 3DSoundAmpLeft;      // Volume level on left headset
    int 3DSoundAmpRight;     // Volume level on right headset
    uint8 checksum;         // probably nothing got mangled
    uint8 footer;           // should always be 0xFF
};
```

State flow while interacting with the home screen (buttons)

N	Timeline	Last step of process	Detailed steps
1	System booting is done	Internal	
2	FlySense application is selected	Internal	
3	Set default StateID = "0" (Default screen)	Internal	
4	Handshake sent to FlySense Onboard Computer	Output	Check if there is any un-replied "HandShake" message topic <ul style="list-style-type: none"> ○ Yes: proceed to confirmation step ○ No: Message published on ROS topic "HandShake" (every second until connection is established)
5	Handshake confirmation received from FlySense Onboard Computer	Input	Listen for reply message on ROS topic "HandShake" <ul style="list-style-type: none"> ○ Success: Clear message queue
6A	User selects view using device touchpad	Output	Publish message with "StateID=x" with StateSource="User"
6B	User selects view using voice commands	Output	Publish message with "StateID=x" with StateSource="User"
6C	FlySense automatically turns on Birds' Eye view (nearby obstacle)	Input	Listen for message with "StateID=2" with StateSource="Computer"
7	Handshake with FlySense time out	Output	After 2 seconds without HandShake from FlySense: <ul style="list-style-type: none"> ○ Inform user "No Data Received!" ○ Go to step "handshake sent to FlySense"

State flow while interacting with the HUD

N	Timeline	Last step of process	Detailed steps
1	Head pose is measured	Internal	
2	Data received from onboard computer	Input	Listen to messages in topic "SensorData"
3	Reset time out counter	Internal	
4	Process Data Packet and check for check sum consistency	Input / Output	
4A	Confirm StateID = "2" (Standard instruments)	Internal	<ul style="list-style-type: none"> • State ID different from 2 • If StateSource in DataPacket = "Computer" update StateID on the side of the AR • If StateSource in DataPacket = "User" initiate handshake to update state on the side of the FlySense computer
4B	If Consistent: Display to data feeds to user	Internal	<ul style="list-style-type: none"> • If checksum is not correct for less than 1 second: no action • If checksum is not correct for more than 1 second: display message to the user in AR headset "Corrupt Data Received!"
4C	If Consistent: Update virtual horizon using heads' relative pose	Internal	
4D	If not Consistent: Show latest valid data / wait for next valid Data Packet	Internal	
4E	User is informed corrupted data is being received	Internal / Output	


State flow while interacting with the HUD

N	Timeline	Last step of process	Detailed steps
1	Allow user to change image display preferences (size and location)	Internal	
2	Load latest user preferences for image display (size and location)	Internal	
3	Data received from onboard computer	Input	Listen to messages in topic "SensorData"
4	Reset time out counter	Internal	
5	Confirm StateID = "3" (Bird's Eye View)	Input / Output	<ul style="list-style-type: none"> • StateID different from 3 • If StateSource in DataPacket = "Computer" update StateID on the side of the AR • If StateSource in DataPacket = "User" initiate handshake to update state on the side of the FlySense computer
6	Process Data Packet and check for check sum consistency	Internal	<ul style="list-style-type: none"> • If checksum is not correct for less than 1 second: no action • If checksum is not correct for more than 1 second: display message to the user in AR headset "Corrupt Data Received!"
6A	If Consistent: Display data image to user	Internal	
6B	If not Consistent: Show latest valid data / wait for next valid Data Packet	Internal	
6C	User is informed corrupted data is being received	Internal / Output	

The work streams have been reorganized to minimize rework

Initial Approach


A



FlySense

Sensors in a box

B



Quadcopter

C



Simulation (online)

D



Helicopter


Current Approach

A



Simulation (offline flight dataset)


B



FlySense


Sensors in a box

C



Quadcopter

D



Helicopter



