Quadcopter Docking on a Moving Platform

Conceptual Design Review

Team E – Dock-in-Piece

Paul M. Calhoun

Rushat Gupta Chadha

Keerthana Subramanian Manivannan

Aishanou Osha Rait

Bishwamoy Sinha Roy

MRSD Project – Dock-in-Piece

October 2, 2015

Table of Contents

MRSD Project – Dock-in-Piece

October 2, 2015

1. Problem Description

1.1. Project Motivation

Our problem statement is inspired by the challenges faced by FMC Technologies Schilling Robotics personnel while docking their Remote Operated Vehicle (ROV) to the Tether Management System (TMS). The ROV detaches and deploys from the bottom of the TMS when the system is at depth. The TMS is negatively buoyant and is suspended from a ship. As the ship heaves on the surface of the water, the TMS heaves up and down with a slight lateral motion. ROV Operators must dock and latch the ROV to the underside of the moving TMS prior to resurfacing. This can be very challenging for even experienced operators. Collisions frequently damage the ROV and TMS. The tether is sometimes squeezed between the ROV and the TMS, which degrades the communication and power supply between the TMS and the ROV. At times, the tether breaks and the ROV falls to the bottom of the seabed resulting in the need for another ROV to be deployed to bring it back.

1.2. Project Goal

Through this project we will demonstrate the autonomous docking and undocking of a quadcopter from the underside of a suspended moving platform. This model will approximate the subsea system of ROV and TMS, complete with determining the safe conditions to dock and providing mechanical latching system that minimizes the forces between the quadcopter and the platform. The project concentrates on an aerial counterpart as water testing and water-proofing an electric system provides challenges that the sponsor isn't interested in.

2. Use Case

A developer at Schilling Robotics is looking through a hobbyist drone site and sees a retrofit kit that adds a minimal payload, and the capability of autonomous docking to a platform moving in a single axis. Having several customers of his unmanned undersea vehicle branch who want a method of navigating to a tether management system with their underwater remotely operated vehicle, he purchases the retrofit. He reasons that it will be fun, and possibly get him a pay point on his next performance cycle if he can demonstrate its usefulness to his supervisor. He purchases the retrofit and declines to fill out a customer survey asking him what further features he wants to see in the next version since this one has all the features he wants already.

After waiting several weeks and visiting FedEx four times to find out why his package has ended up in a facility in Delaware when he lives in California, the developer finally receives the kit and spends a weekend setting up a dock. The addition of the software changes to his Phantom 2 takes a few minutes and the hardware install is just as swift. It's a windy day and the tree he'd tied his platform too was swaying quite a bit, and after his initial disappointment at the app telling him it was impossible to dock in those conditions, repeatedly mashing the 'dock' button finally proved effective and the drone successfully attaches itself to the dock without running into the tree. It even weaves around his bird feeder and succeeds in avoiding a starling that appeared intent on driving the drone out of the air. Satisfied, he calls for the drone to undock. Unfortunately, the wind has picked up and he is forced to use a ladder to retrieve it, the drone wisely refusing to detach itself when it was likely to be blown into the tree before it can get up the thrust to avoid it. He is pleased that the retrofit is light and not very cumbersome, making it somewhat easier to bring the drone down the ladder.

The developer secures funding from his supervisor and contacts the student team who launched the retrofit into a full product. Though hesitant at first, they engage an attorney and draw up a limited use contract for the TDP of the docking kit. The developer is happy, his boss less so when he sees what kind of royalties the developer had agreed to, and the developer realizes he's going to have to work very hard for that pay point. He gets going and succeeds in adapting the code for his customers' ROV and TMS. On its first test, the ROV collides with an undersea vent, but the entire test is invalidated when they discover an octopus had attached itself to the ROV camera and that a warning had been displayed by the adapted software, but not where the ROV operator is used to viewing warnings and cautions.

Finally, launch day arrives and the customer is pleased with the results. The ROV docks without needing the use of a heave-compensated winch. The ROV smoothly detaches from the TMS, goes about its mission, and returns to be hauled up on the TMS without incident. The customer is also very happy with the user interface, removing the need for lengthy training and decreasing the costs of using the ROV since the operators don't have to be as skilled any more. The developer gets a bonus from his supervisor, an angry letter from the sailors' union, and a bill from the quadcopter kit developers after an independent audit.

Future deployments of ROV systems aboard ships include the changes and a program to make the necessary changes is implemented on legacy ROV carriers as they are brought in for routine maintenance. Costs across the fleet decrease and A_O increases significantly, drawing the attention of the US Navy. They approach Schilling, who directs them to the original retrofit team. After a three-way negotiation between Schilling, the retrofit team, and NAVSEA's PEO-IWS contracts personnel, a government-only use contract is signed. None of the project team has need-to-know and apart from the regular checks, they know nothing more about the system as being used by the Navy. One does read in defense news that shortly after the contract is signed that more undersea robots were being deployed by the Navy, but few details are forthcoming.

3. System level requirements

3.1. Functional Requirements

3.1.1. The system shall

F1. Have two major components: quadcopter and moving docking platform
F2. Detect and communicate when docking and undocking is not possible

3.1.2. The docking platform shall -

F1.1 Be moving
F1.2 Hold the quadcopter in place once it has docked

3.1.3. The quadcopter shall -
F2.1 Localize itself w.r.t. the platform
F2.2 Plan a path to the docking platform
F2.3 Navigate to the platform
F2.4 Dock/undock to/from the platform without any collision

3.2. Non-Functional Requirements

3.2.1. The system shall:
NF1. Function in a GPS degraded environment
NF2. Be easy to operate, maintain, and repair
NF3. Provide a user interface with DOCK and UNDOCK options and provide status
NF4. Cost less than $3,000 to own over its life cycle

3.2.2. The quadcopter shall:
NF2.1 Have a payload capacity of > 500g

3.3. Performance Requirements

3.3.1. Mandatory Requirements

The docking platform will-
MP1.1. Have 1 degree of freedom along Z-direction
MP1.2. Oscillate in harmonic motion with dominant frequency < 0.5Hz
MP1.3. Have oscillations' span ±200mm
MP1.4. Have a locking mechanism which supports weight of 5kg

The quadcopter will –

MP2.1. Localize w.r.t. platform within 50mm accuracy
MP2.2. Navigate to the platform within 10 minutes
MP2.3. Dock to the platform autonomously and without colliding within 10 minutes

3.4. Desirable Requirements

The docking platform will-
DP1.1. Have 3 degrees of freedom along X, Y and Z-direction
DP1.2. Have random movements in 3D space

The quadcopter will -

DP2.1. Localize w.r.t. platform within 30mm accuracy
DP2.2. Navigate to the platform within 5 minutes

4. Functional Architecture

4.1. Docking

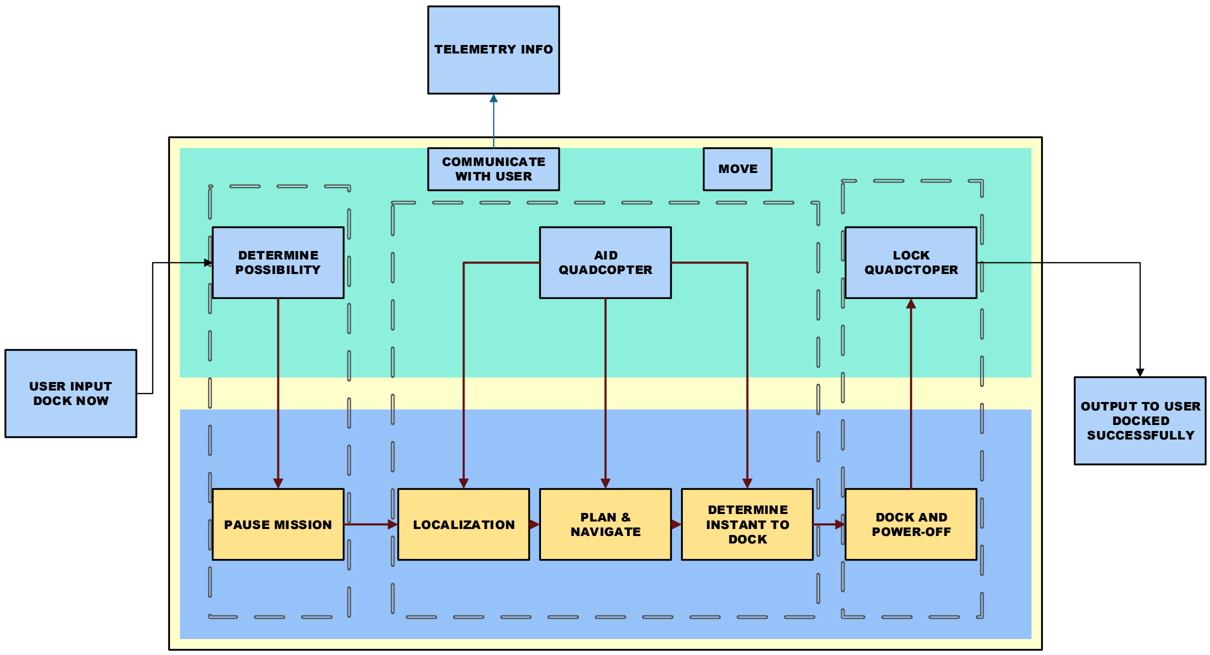


Figure 1 Physical Architecture of the Docking process

The system's input is the user's decision to dock and the output is the successful dock. The architecture is divided into two parts - the quadcopter and the platform. The quadcopter and platform are working together to plan the approach to the docking platform and dock at an opportune moment. There are three phases in the whole docking process. The platform and the quadcopter relay periodic updates to the user about their individual statuses and poses. (Figure 1)

4.1.1. Decision Phase

Once the user requests the docking to initiate, the platform receives the request, analyzes its movement, and sends the request to dock to the quadcopter. If possible, the quadcopter and platform move on in the docking procedure. If docking isn't possible (maybe the quadcopter is too far away or the platform's movement is too erratic), then the user is notified aptly.

4.1.2. Navigation Phase

In the navigation phase, the quadcopter plans a path to the platform using onboard sensor information fused with sensor data gathered from the platform and abides by the calculated waypoints. Once the quadcopter is within observation range (set later in design process) the system enters the last phase.

4.1.3. Docking Phase

In this phase, the quadcopter keeps a safe distance from the platform and analyzes the platform's heave. It uses the observation to understand the movement of the dock. If the quadcopter deems the platform's movement is reasonable, it proceeds to move within latching range. If docking is deemed impossible the quadcopter notifies the user. Lastly, this check is redundant as the decision is done in the decision phase. However, it is needed to re-analyze the situation to assure the decision made still holds. The initial dock is to assure that a preliminary decision can be made early on in the docking process. That way time and energy need not be wasted in approaching an undockable platform. The final output is sent to the user that the quadcopter has been docked successfully.

4.2. Undocking

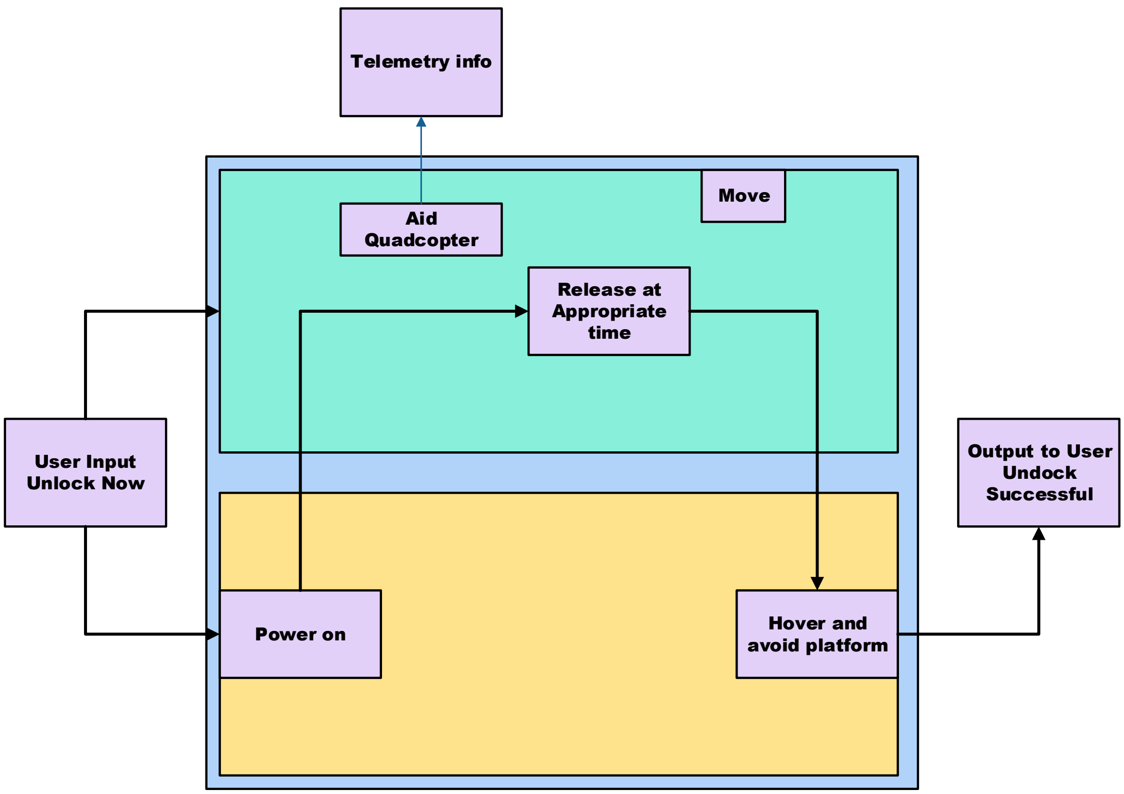


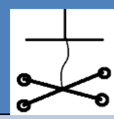
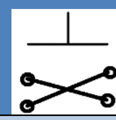
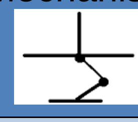
Figure 2 Functional Architecture of the Undocking mechanism

In the undocking process (Figure 2), the user inputs the command to undock. Both the platform and the quadcopter receive the command. The docked quadcopter powers on, the docking platform decides the right time to release and unlocks the quadcopter. The quadcopter hovers on that place and avoids collision with any object and flies away. The user gets output that the undocking is successful. The docking platform keeps moving in the z-axis all this while. The platform and the quadcopter give periodic updates of their status to the user.

5. SYSTEM LEVEL TRADE STUDIES -

5.1. Overall System Design -

Table 1 Overall System Trade Study

Category	Weightage (100%)	Tethered 	Non-Tethered 	Heave Compensating Mechanism 
Ease of Quadcopter Maneuverability	20	7	10	10
Customer Requirements	15	9	9	5
Mechanical Complexity	15	8	8	4
Scalability	15	8	8	3
Control Complexity	15	7	8	7
Cost	10	7	9	4
Chance of Failure	10	7	9	5
Total	**10**	**7.6**	**8.75**	**5.75**

An important thing to keep in mind while deciding various system level designs is the fact that this project is a simulation of a real life problem. To deliver what our client needs, it is very important that our project is in line with their expectations, problems faced with the existing hardware.

Through discussion during the sponsor meetings, team has come up 3 system level solutions that are being judged by the 7 criteria shown in Table 1.

Of the 3 potential solutions, 2 were purely quadcopter control based, while the other was a mechanical solution to be installed on the moving platform. While the sponsor gave the team a lot of flexibility on the range of possible solutions, a mechanical solution was strongly discouraged, primarily because of the scalability issue and the fact that mechanical appendages has lots of on-site maintenance problems. Since the team shared common interest in making the final product as implementable as possible, the heave compensating mechanical appendage was ruled out. A major decision that the team had to make was whether to use a tether on the quadcopter or not. While using a tethered quadcopter would be the closest simulation of the actual system, the low score in ease of quadcopter maneuverability, which had the maximum weight, automatically ruled the tethered solution out. With a score of 8.75 on a scale of 10, a Non-Tethered quadcopter using a purely control based approach was the most scalable and desired solution to the problem.

5.2. Subsystem Level Trade Study

5.2.1. Quadcopter Selection -

Table 2 Quadcopter selection trade study

Category	Weightage (100%)	DJI Matrice 100 [1]	TurboAce Matrix [2]	3DR solo [3]	3DR X8+ [4]
Payload Capacity	20	8	9	4	7
Customizability of processor	15	8	1	7	7
Availability of an SDK	20	9	0	8	8
Documentation of SDK	20	9	0	8	8
Position of on Board Camera	10	8	9	4	4
Battery Life	5	8	8	6	4
Spares / availability	5	8	8	8	8
Cost	5	3	6	7	8
Total	10	8.35	4.35	6.9	7.45

The quadcopter is the most important acquisition of this system as the whole project revolves around its control. A carefully discussed set of criteria for the selection of a quadcopter with their respective weights are illustrated in Table 2 .Three of these criteria played a significant role in the decision process, Payload capacity, availability of an SDK (Software Development Kit), and its documentation. After filtering out hobby quadcopters, the team narrowed down to 4 quadcopters, suppliers of which are famous among the aerial vehicle community for various reasons like payload capacity and the SDK. However, 2 of these (TurboAce Matrix and 3DR solo), got ruled out because of lack of an SDK and low payload capacity.

A very lengthy analysis based on reviews from users and developers led to the conclusion that even though the DJI Matrice 100 was more expensive, its add-ons, preloaded flight algorithms, filtered sensor outputs, battery life, and excellent reviews would overall be a very big advantage for the team while troubleshooting sensor and hardware related problems. Since this was the most important part of our project, cost wasn't given a high weight and hence the DJI Matrice 100 won the trade study with a score of 8.35 on a scale of 10.

6. Cyberphysical Architecture

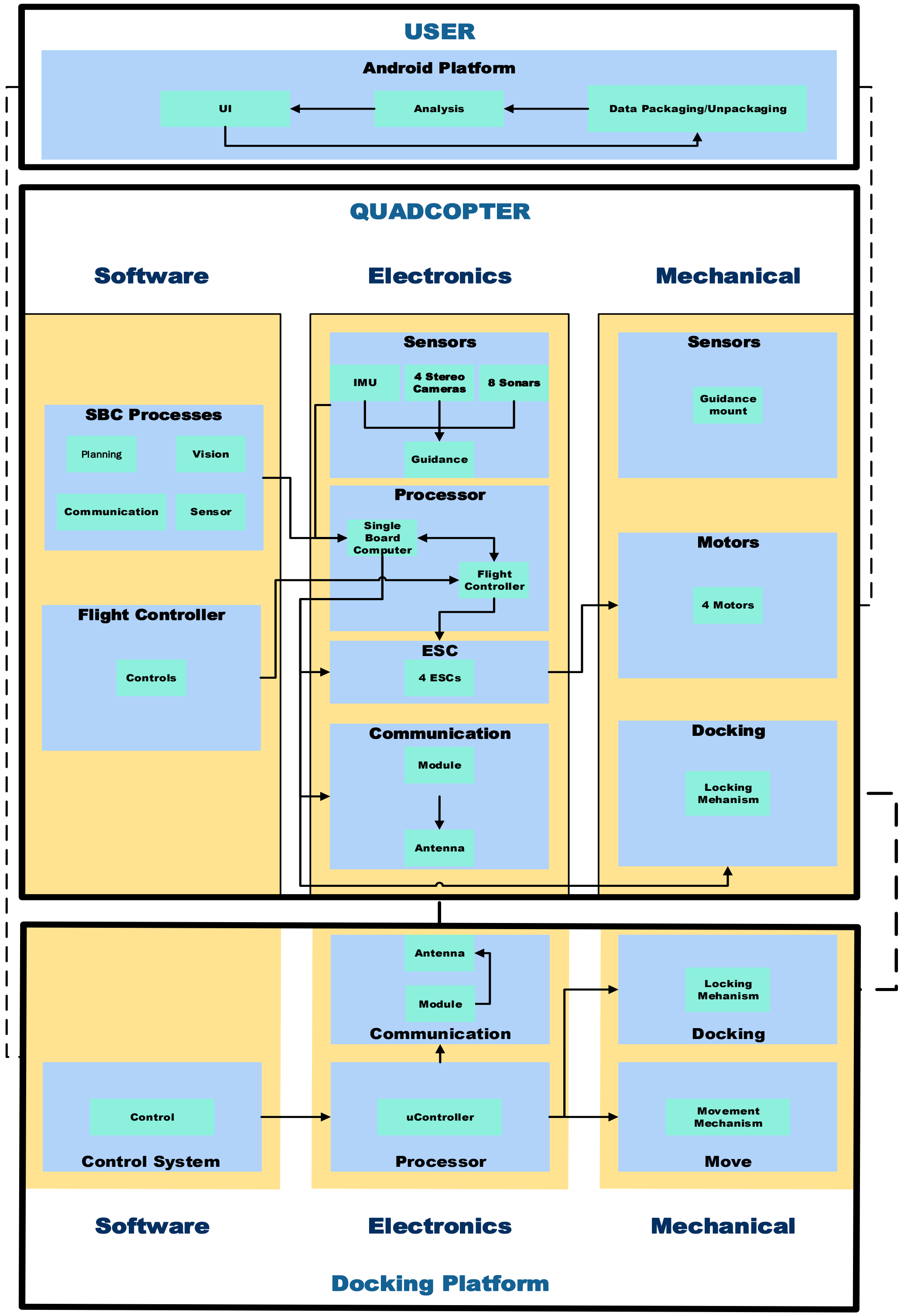


Figure 3 Cyberphysical Architecture

The architecture, shown in Figure 3, is divided into three abstractions for the docking platform and the quadcopter. The software abstraction encompasses the algorithms used to accomplish the functions depicted in the functional architecture. The electronic abstraction shows the different electrical equipment and their connection to run the algorithms from the software abstraction. The lines from the software abstraction show which processor runs the processes. Lastly, the mechanical abstraction holds the mechanisms that allow the software algorithms to manifest into the physical realm. The most substantial mechanical feature is the docking mechanism that will lock the quadcopter onto the platform.

6.1. DJI Matrice 100 (M100)

The M100 comes with a flight controller (N1), 4 ESCs, 4 motors, and the Guidance package (Figure 4). The quadcopter provides two CAN ports and two UART ports to connect to third party devices, Single board computer (SBC). Also, a 1.2 mile range is supported by the onboard antenna. Using this communication link, HD videos can be transmitted. As such, this link will be used to download vital status information onto a developed android application, which will act as the UI. The quadcopter is normally controlled by a remote controller. The best way to approach the autonomy problem is to make the SBC mimic the remote controller's commands.

The Guidance[5] package provides 4 direction stereo cameras with 8 sonars. The system can provide velocity, obstacle distance, IMU (Inertial Measurement Unit) data, ultrasonic data, grayscale image, and depth image information. The guidance also provides two communication interfaces, UART and USB. Video transmission can only be done via USB. As such, the guidance can do some preprocessing on the grey-scale images and stream them to the SBC, which can do more complicated processing. Lastly, the guidance system's sensors provides obstacle avoidance to the quadcopter. A safe zone is established by the code and the quadcopter avoids any collision with an obstacle within the safe zone. This is big safety feature that will be used to assure collision free docking.

There are three SDKs provided by DJI: Mobile SDK[6], Onboard SDK[7], and Guidance SDK[7]. The Guidance SDK allows us to build applications using the compiled sensor data. Additionally, another SDK allows us to develop application on a Linux board connected to the N1 flight controller via UART. This communication can be achieved via ROS, abstracting away the low level communication details. The highest level software development (also the most complicated algorithms) will be run on a SBC using the provided Onboard SDK. Lastly, an Android mobile app developed on top of the provided Mobile SDK will receive status information from the platform and the quadcopter and present the information to the user.

Figure 4 Guidance System

7. Subsystem Descriptions

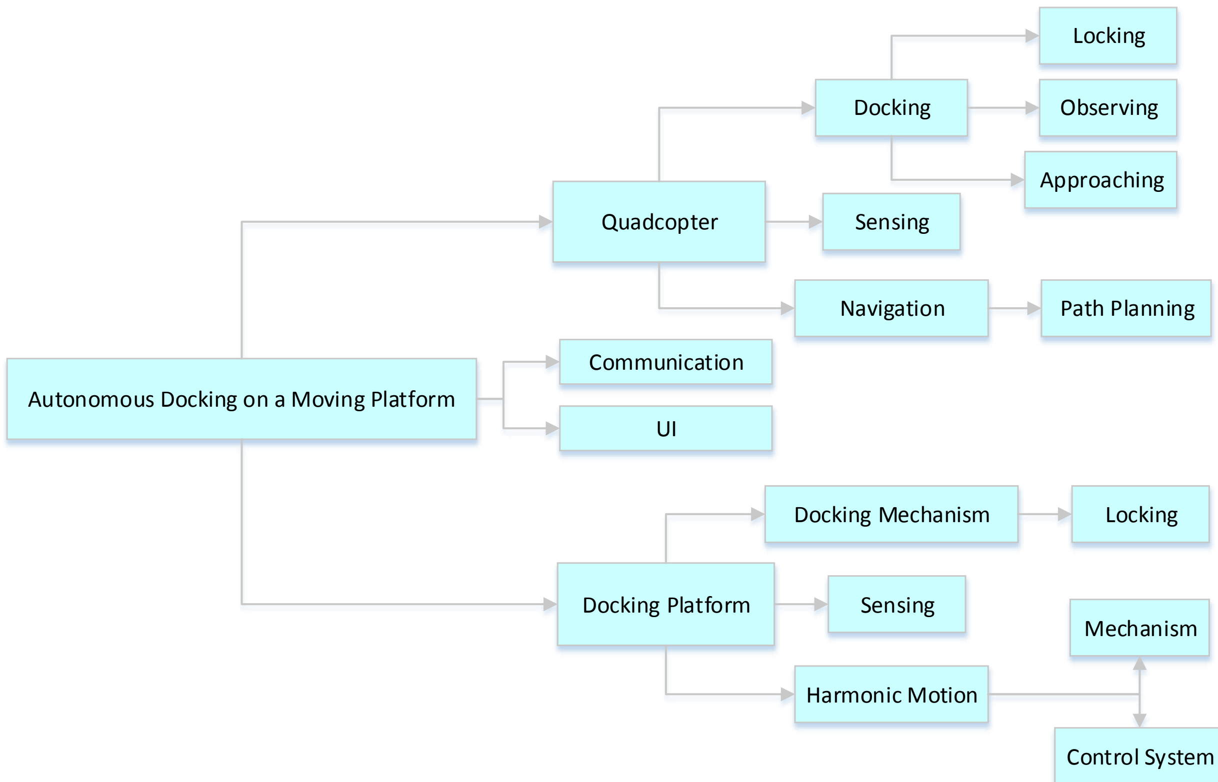


Figure 5 Subsystem Breakdown

Figure 5 shows the subsystem break down. The following sections explain the subsystem breakdown.

7.1. Docking Platform

7.1.1. Sensor Package

The sensor package will contain an IMU to measure the acceleration and orientation. This information will need to be fused from multiple sensors to provide waypoints to the quad.

7.1.2. Mechanisms

The docking platform holds a grappler which is used to latch the quadcopter onto it.

7.1.3. Controls

The docking platform is constantly moving in the z-direction with a mixture of several simple harmonic motions. A trade study regarding platforms and its control is still pending.

7.2. Quadcopter

7.2.1. Sensor

The sensor subsystem is working with the DJI Guidance system. Visual tracking of the platform and relative localization will be the main tasks.

7.2.2. Navigation

The navigation subsystem will take the vector from the quadcopter to the platform and plan a smooth path and send waypoints to the flight controller. It will also need adapt the path to the motion of the platform and errors in the quadcopter's performance. The subsystem will work with the Onboard SDK on the SBC.

7.2.3. Mechanism

The mechanism subsystem will engineer a mechanical latch that will lock the quad onto the docking platform. The main concerns of the mechanism subsystem is to hold the weight of the quadcopter under the moving platform.

7.2.4. Docking

The docking subsystem will design a method of learning the pseudo-random motion of the platform and determine if docking is possible and then plan an opportune moment for approaching the platform, prioritizing a collision free docking. This subsystem will work with the Onboard and Guidance SDKs.

7.3. Communication

The communication subsystem is responsible for designing a method of communicating a protocol to transmit the information between the platform, the quadcopter, and the mobile application. This information will be used by the quadcopter to localize itself into the frame of reference of the docking platform. This subsystem will work with Onboard SDK. Most probably, a new protocol will need to be designed to enable the communication.

7.4. User Interface

An android app will provide capabilities that will allow the user to control and monitor the docking/undocking operations. Two activities will be developed:

1. The first activity will expose three buttons. One button that will pop-up a dialogue box to connect to the network. A second button to initiate docking/undocking process. Lastly, a third button will be used to switch to the telemetry activity. Presence or absence of a successful connection will be reflected by the changing the background color. (Figure 6)

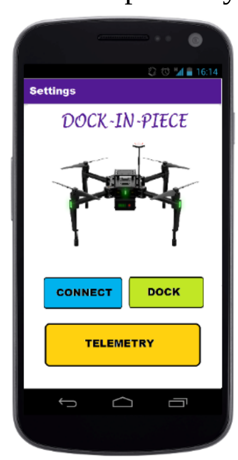


Figure 6 Decision activity

MRSD Project – Dock-in-Piece

October 2, 2015

2. The second activity will consist of two fragments and provide the status of the operation through live video feed and display of data statistics. The data will include distance between the quadcopter and platform, estimated time left to dock, the maximum frequency of oscillation of the platform and the percentage of the battery left. The user can swipe the display between the video feed and the data feed.

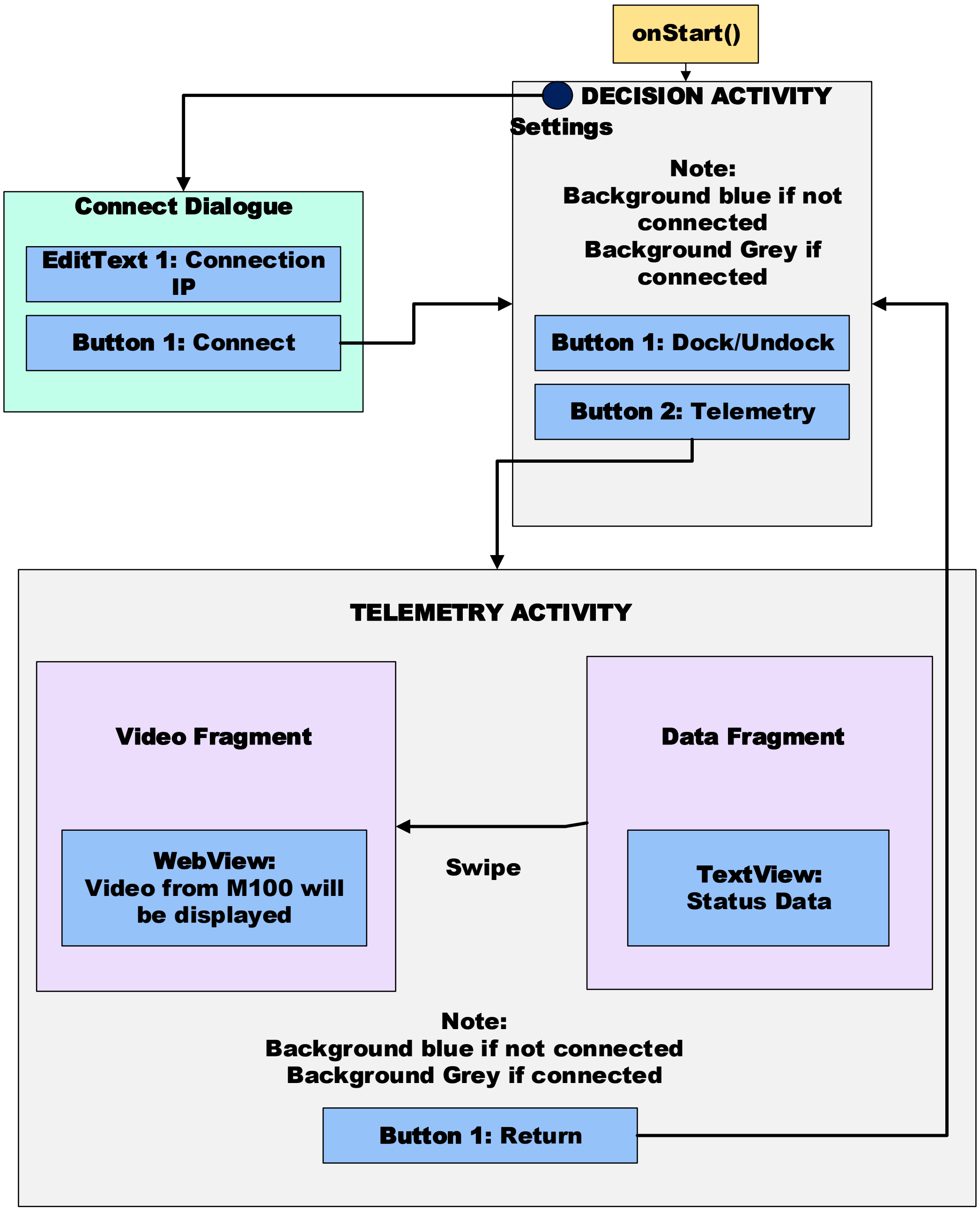


Figure 7 Telemetry Application Wireframe

8. Project Management

Our project management strategy is one of proactive action to produce consistent results and continuous improvement. By tracking our progress against a plan, we will make sure that we continue to have realistic goals and benchmarks, adjusting our expectations and effort to produce subsystems on schedule and with solid functionality. The following cover our tasking and tracking methodology.

8.1. System Schedule and Validation

Our project can be broken down into the following milestones, which will demonstrate the validity of the subsystems and concepts as they are produced:

Table 3 System Schedule

Date	Concepts Demonstrated	Methodology / Documentation
OCT 28	Physical design	Physical architecture fully populated, System/Subsystem Specification fully populated
NOV 10	Platform design and control system complete	Simulation (shows in sim that the dock moves correctly)
NOV 23	Platform constructed, dock designed	Physical testing of platform mounting (demonstrates that it can the mounted), simulation
DEC 10	Dock functional, quadcopter navigates to safe distance from stationary dock autonomously	Physical testing
DEC 14	CDR	
JAN	Path planning can avoid dock moving with single harmonic motion	Physical testing
FEB	quadcopter can dock with single harmonic, avoid dual harmonic	Physical testing
MAR	quadcopter can dock with dual harmonic, avoid triple harmonic	Physical testing
APR	quadcopter can dock with triple harmonic	Physical testing
MAY	Final review	

8.2. Work-Breakdown Structure

The project can be broken into three major subsystems and then further into seven intermediate subsystems and fifteen basic subsystems (see Fig.8).

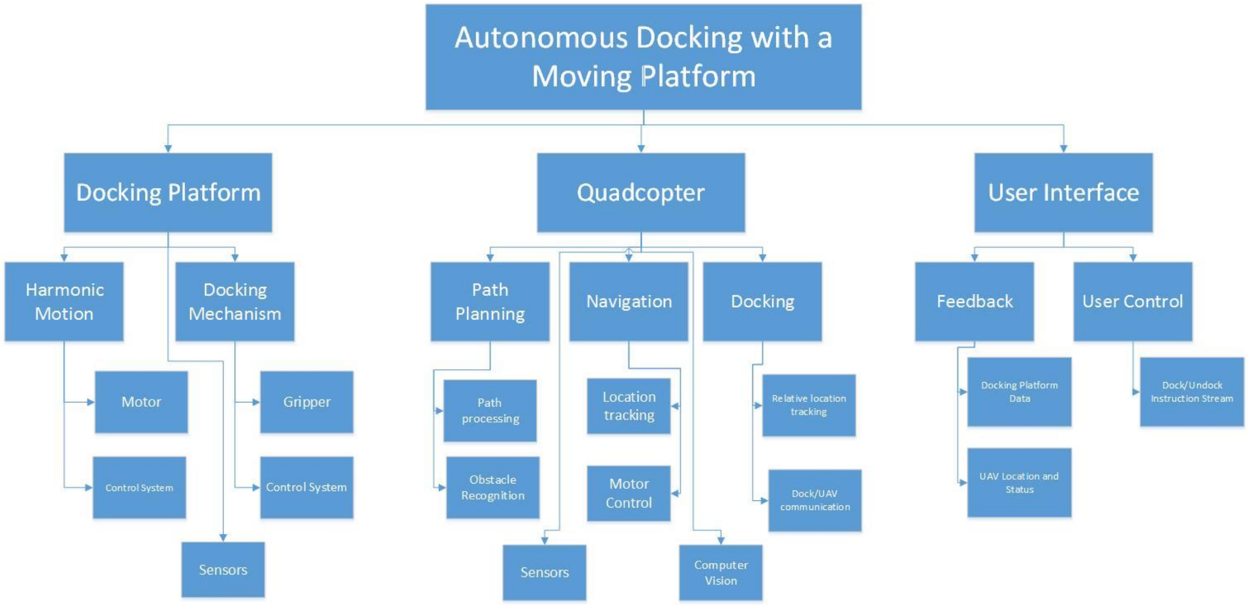


Figure 8 Work Breakdown System

8.3. Subsystem Gantt Chart

Our subsystems will be developed concurrently as much as possible, as represented in the following Gantt Chart:

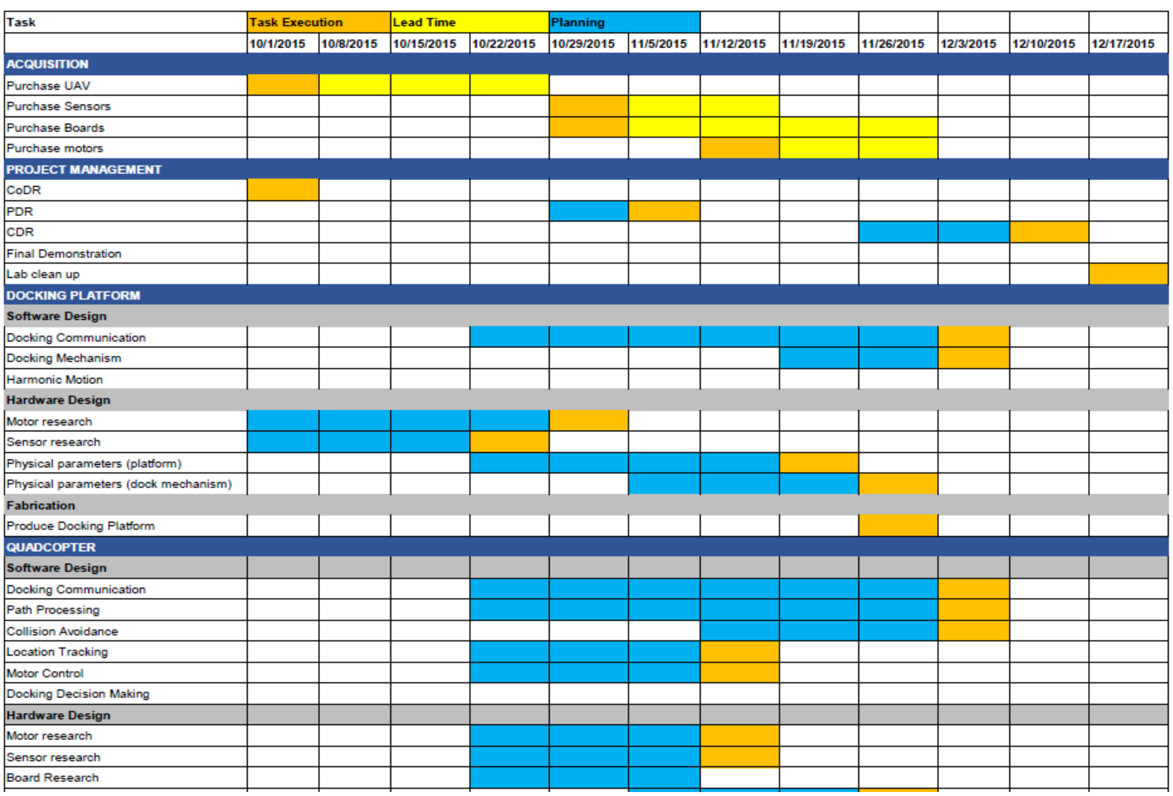


Figure 9 Gantt chart for Fall '15

MRSD Project – Dock-in-Piece

October 2, 2015

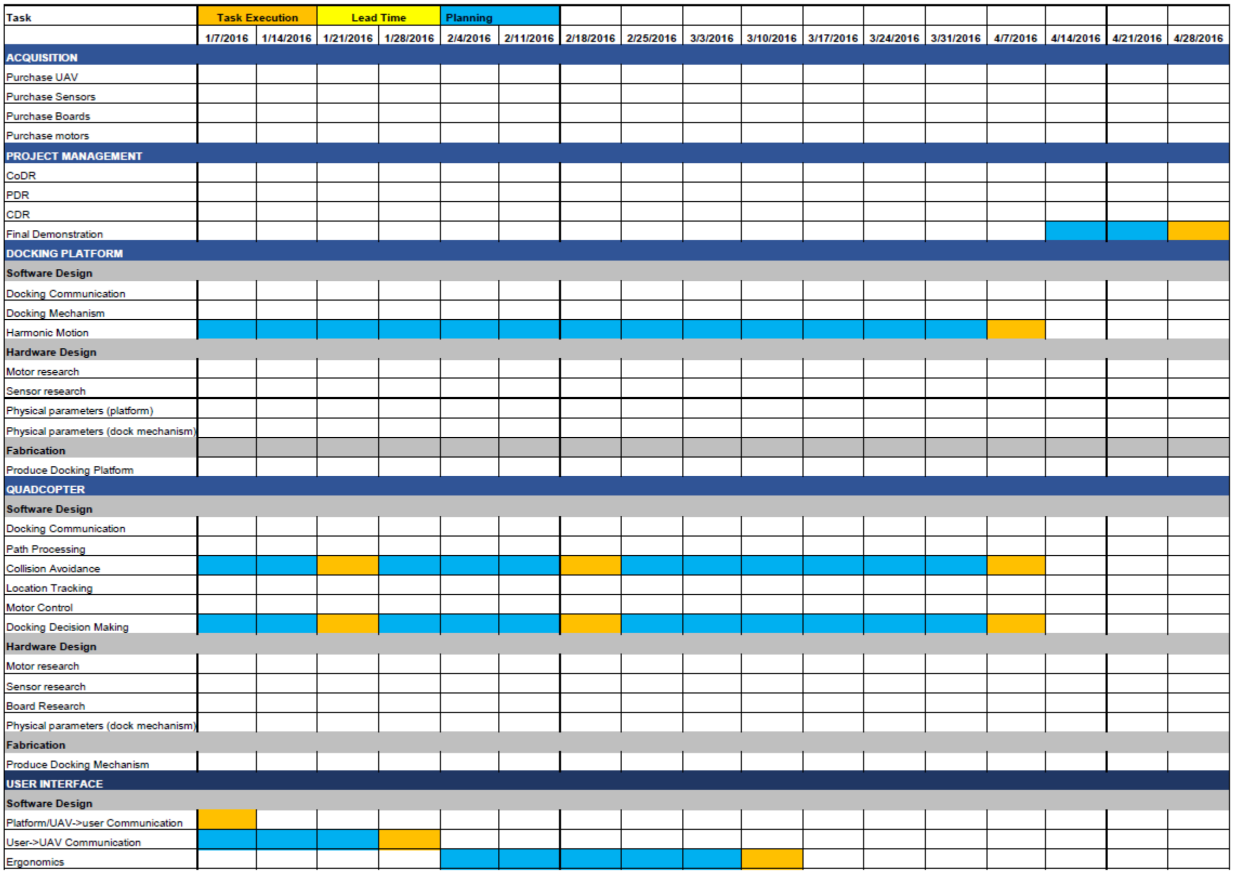


Figure 10 Gantt Chart for Spring'16

8.4. Verification and Validation

V&V testing will be performed as a series of internal (verification) and external (validation) milestones. The majority of verification tests will be done as part of the continuous progress and improvement initiative, with milestones only serving to produce documents for official use to record already proven subsystem functionality and system integration. These tests will occur within one week of declared conclusion of development (see Gantt Chart). All testing will be performed in the MRSD lab spaces in the B level of Newell-Simon Hall (NSH) unless otherwise specified. Any test that shows multiple methods first refer to continuous improvement test and then to final acceptance test.

Validation will be done either at the request of the project sponsor (Schilling) or concurrently with documented verification as part of regular status updates. Validation milestones will occur only at the completion of first system integration and then at final system testing. (See Table 4)

Table 4 Verification & Validation table

Test Item	Test Method	Metric of Success
Docking Platform Communication	Passive receiver	Docking platform status messages are 100% correct format and arrive 95% of the time
Docking Platform Physical Parameters	Perturbation	Docking platform remains stable in x and y axis when subjected to perturbation in excess of 140% of vibrations expected from z-axis motion.
Docking mechanism physical parameters	Direct use	Docking mechanism closes and holds on quadcopter, maintaining hold and position on mount without dropping the quadcopter or sustaining damage.
Quadcopter Docking Communication	Passive receiver	Quadcopter status messages are 100% correct format and arrive 95% of the time
Quadcopter/Docking Platform collaboration	Both items communicating with each other while quadcopter is in flight	Quadcopter responds to docking platform 99% of the time with correct response, and docking platform localizes quadcopter to within performance parameter threshold
Path Processing	Simulation / quadcopter in flight	Simulated quadcopter plans path 95% of the time within reasonable parameters of best path, real quadcopter satisfies performance threshold
Collision Avoidance	Simulation / quadcopter in flight	Simulated quadcopter avoids collision 100% of the time without unnecessary maneuvers, real quadcopter satisfies performance threshold
Location Tracking	Docking platform receiver	quadcopter localizes self within performance threshold
Motor control	Simulation / quadcopter in flight	Simulated and real quadcopter control flight to within performance threshold
Docking decision making	Quadcopter in flight with platform present	quadcopter satisfies performance threshold
Platform/quadcopter-> User communication	Simulation/quadcopter in flight with app	Simulation and quadcopter provide readable and correct data to app with 95% accuracy
User->quadcopter communication	Simulation/quadcopter in flight	quadcopter responds to user with feedback or action 99% of the time
Ergonomics	IV&V testing	75% of test subjects find the interface easy to use.

8.5. Team Responsibilities

Table 5 Team Responsibilities

Role	Primary	Secondary
Quadcopter path planning	Bishwamoy SR	Aishanou OR
Quadcopter navigation	Rushat GC	Bishwamoy SR
Docking mechanism	Keerthana SM	Rushat GC
User Interface	Aishanou OR	Bishwamoy SR
Platform Design	Rushat GC	Keerthana SM
Platform Fabrication	Paul C	Rushat GC
Platform motion controls	Aishanou OR	Paul C
Communication	Keerthana SM	Paul C
Project Management	Paul C	Keerthana SM
Website	Rushat GC	Aishanou OR

8.6. Parts and Budget

Although our part list is preliminary, our budget strategy is reasonably mature.

Budget (Table 6):

Table 6 Budget

Source	Use	Amount total
Schilling	Capital Investment	$5000
MRSD	Consumables	$4000 (shared)
MRSD	Parts	$4000 (shared)
MRSD	Equipment	$4000 (shared)

Parts (Table 7):
Note: These costs are approximate

Table 7 Parts Table

S.No.	Part	Subsystem	Cost
1	DJI Matrice 100	quadcopter	$3300
2	Guidance System	quadcopter	$1000

8.7. Risk Management

We are using an ongoing risk tracking and mitigation system to proactively prepare for and minimize risks as can be seen by the following risk overview chart and risk list. (Table 8 & 9)

Table 8 Risk Matrix

Probability	Severity	A Negligible	B Low	C Moderate	D Severe	E Catastrophic
5	Nearly Certain	0	0	0	0	0
4	Likely	0	1	0	1	0
3	Possible	0	0	0	0	0
2	Unlikely	0	0	2	1	0
1	Rare	0	1	1	4	1
Immediate Action						
Urgent Action						
Action						
Monitor						
No Action						

Table 9 Risk Overview Chart

Risk	Probability	Severity	Date Identified	Mitigation	Notes	Action to Take
DJI SDK is an unsuitable development platform	1	C	10/2/2015		Cost risk	No Action
Matrice cannot support our needs	1	D	10/2/2015	Research prior to purchase	Cost risk	Monitor
Guidance sensors unsuitable to	1	D	10/2/2015		Schedu le/cost risk	Monitor

our requirements						
Simulations diverge significantly from reality	2	C	10/2/2015	Careful simulation creation	Schedule risk	Monitor
Docking mechanism fails while Quadcopter is docking or docked	4	D	10/2/2015	Place net under platform	Physical risk	Immediate Action
Quadcopter collision avoidance fails in flight	2	D	10/2/2015	Keep Guidance On	Physical risk	Action
Quadcopter attempts to shut down engines after a false positive dock	1	D	10/2/2015	Place net under platform	Physical risk	Monitor
Delays in shipping	4	B	10/2/2015	Order in advance	Schedule risk	Action
NSH lab not big enough for testing	1	D	10/2/2015	Find that out early and reserve Rangos	Schedule Risk	Monitor
Electrical failures	2	C	10/2/2015	Wire safety / fuses	Schedule/cost risk	Monitor
Platform fails mechanical requirements	1	B		Make another one	Schedule risk	No Action
A developer becomes unavailable	1	E			Schedule Risk	Monitor

References

[1] DJI Matrice 100: https://developer.dji.com/matrice-100/
[2] TurboAce Matrix: http://www.turboace.com/matrix_drone.aspx
[3]3DR Solo: https://3drobotics.com/evolve/
[4] 3DR X8+: https://3drobotics.com/x8/
[5] DJI Guidance: https://developer.dji.com/guidance/
[6] Mobile SDK : https://developer.dji.com/mobile-sdk/
[7] On-Board SDK: https://developer.dji.com/onboard-sdk/features/
[8] Guidance SDK: https://developer.dji.com/guidance-sdk/