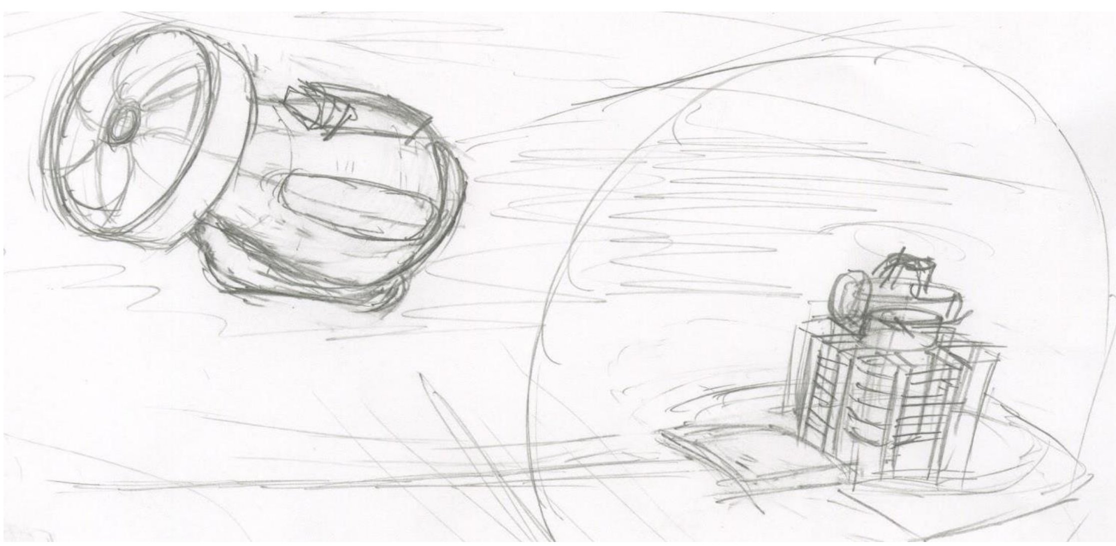
Column Robotics: Team C
Critical Design Review
December 18, 2015

Job Bedford

Cole Gulino

Erik Sjoberg

Rohan Thakker



1. Abstract

This report summarizes our progress on the development of a terrestrial analogue to an autonomous underwater vehicle capable of searching for and docking with deep-sea wellheads.

We start-off by presenting an overall description of the project and the use case followed by the System-level requirements. We then show the functional and cyber physical architectures that describes how our system meets the requirements.

Next, we show the current status of our implementation of the system along with the project management tools that we used for planning and tracking our progress. The last part of the report consists of the conclusions, references and appendices.

2. Project Description

Wellheads are infrastructures for pumping oil and gas on the ocean floor. They are responsible for a large portion of the world's oil consumption. When one of these system breaks down it can assume billions of dollars in damages. A prime example is the BP oil spill which had catastrophic effects on the BP Company and the Gulf of Mexico as a whole.

Unfortunately, current maintenance and monitoring of these wellheads is expensive costing hundreds of thousands dollars per intervention. At pressures too deep for human to useful intervene, oil companies are often require a specialized ship, with a highly trained crew to deploy a manual ROV (remotely operated underwater vehicle) to perform a simple checkup or turn a valve. Due to this cost, oil companies often choose to leave well-head unmonitored until a problem arises, and by then it can already be too late.

Seeing this pain, our team proposes an Autonomous Robotic Solution to reduce cost, resources, and human intervention. We will demonstrate a terrestrial analog to an underwater vehicle capable of autonomously searching for, identifying and docking with undersea wellheads. Due test resources and pool time constraint, a terrestrial analog was chosen over an actual AUV (Autonomous Underwater Vehicle). This terrestrial analogue will be a Quadrotor Drone capable of 'swimming' through air.

AUVs (Autonomous Underwater Vehicle) exist that can search and identify undersea wellheads, but none we have seen that can autonomously dock or intervene at a wellhead. AUV with this capability will allow for cost effective, regular maintenance and monitoring of this wellhead which will reduce avoidable damages and loss of resources. Figure 1 shows a pictorial description of the problem statement.

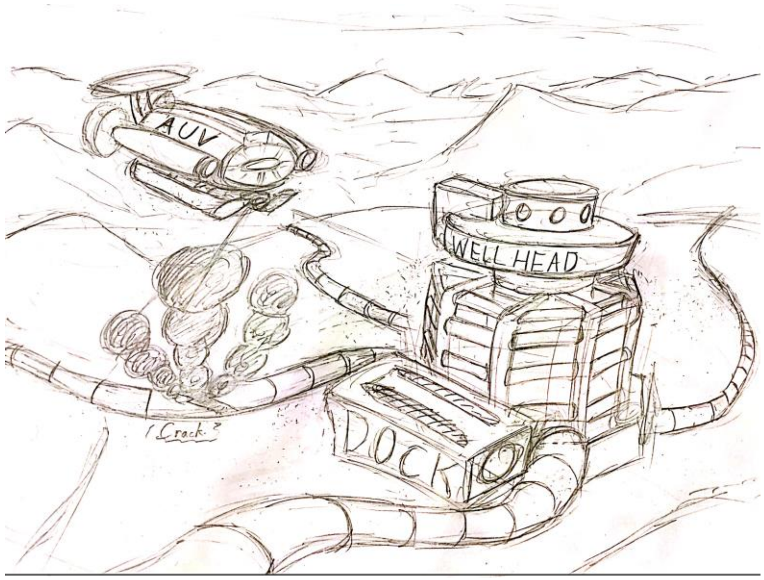


Figure 1) Visual Description of Autonomous Underwater Exploration of a Wellhead

3. Use case

The depths of the ocean floor are home to an enormous plethora of flora and fauna. In our times, however, manmade obstacles have joined the ranks of deep sea denizens. There may be no more important man made sea inhabitant than the deep sea wellhead. These objects facilitate the distribution of our widest used fuel source, fossil fuels.

A wellhead just like any other lies at the bottom of the sea near the gulf coast. The life of the undersea wellhead is one of isolation and duty. Years ago he was lovingly designed and built by a team of engineers. Those engineers however lost touch with the wellhead as soon they placed him underneath the ocean surface. It has been years since the wellhead has seen another metal denizen or human face. The wellhead still must do his job valiantly day in and day out, because the fossil fuels he carries and protects would create a catastrophe if they ever seeped into the ocean waters.

To most everyone else, today was like any other day, but for the wellhead, today was a day of tragedy. His structure has grown weak with time. The rust around his pipes is growing slowly, getting worse every day. He sees oil leaking from the cracks in his body, more each day.

The wellhead is afraid. He knows that the ROVs necessary to go underwater and interact with him are prohibitively expensive. He knows that they'll never check on him until it is too late.

The wellhead waits and waits and waits. He does not know this, but help is on the way. Suddenly one morning, an autonomous underwater vehicle comes into his vicinity. There was no tether connecting him to an expensive ROV ship. There was no skilled laborer operating him from afar. The vehicle notices the wellhead, surveys every inch, and notices the leak. The next day, a large team comes and saves the lonely wellhead.

The wellhead cannot believe that he and the other water denizens were saved that day. He believes that this is a miracle. What he does not realize is that the oil company that bought his new autonomous friend, bought him with the specific purpose of doing routine checks on the wellheads. Now the company can do routine checks in order to protect the environment and their legal interests. Every month the lonely wellhead receives a visit from his friend the autonomous underwater vehicle.

Our terrestrial analog, the drone, will start somewhere in the vicinity of the wellhead, and lift off to begin its search. It will perform a searching strategy until it comes across the wellhead as shown in Figure 2.

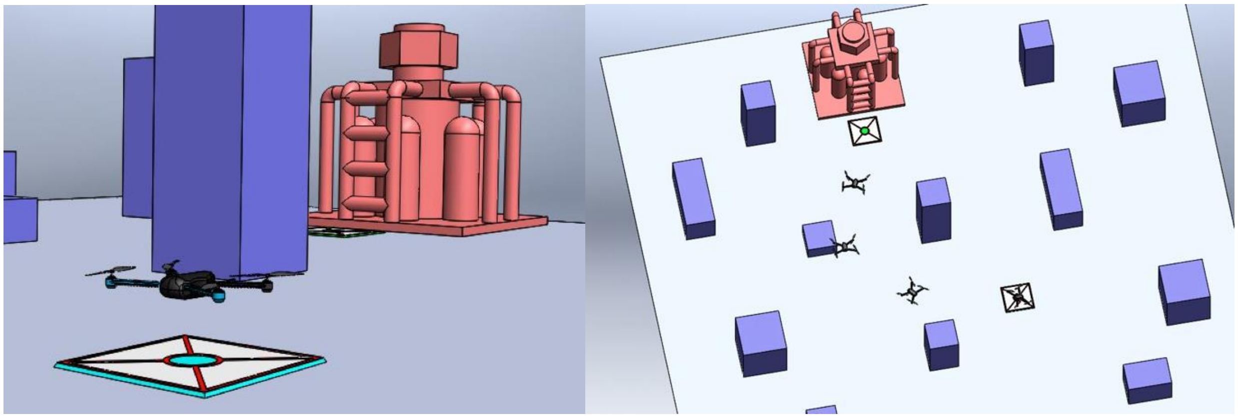


Figure 2) Autonomous Searching for Wellhead

It will perform a searching strategy until it comes across the wellhead. Once the drone thinks it has found the wellhead it will identify via a specialized tag or feature. The drone will then initiate its pre docking orientation and positioning as shown in Figure 3.

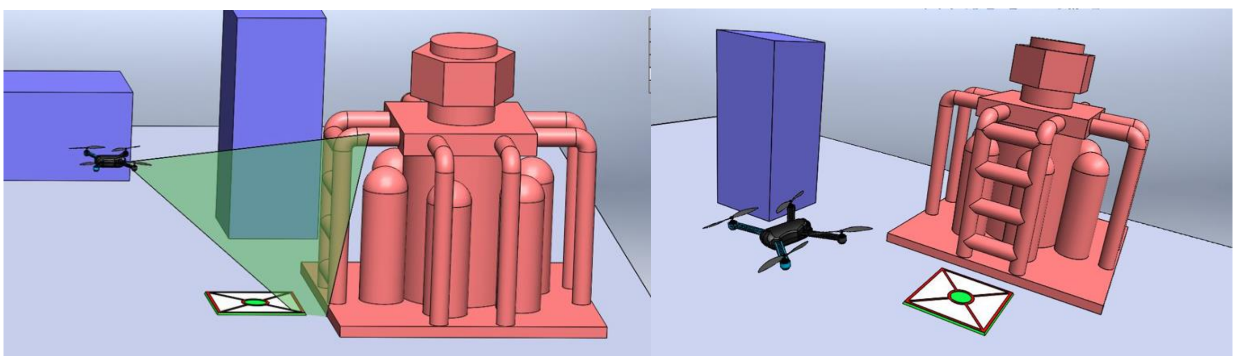


Figure 3) Wellhead Recognition and Initiating Pre-Dock Position

The drone will then proceed to dock accordingly as shown in Figure 4, and the system will be successfully complete.

Figure 4) Drone in the Process of Docking

4. System-Level Requirements

4.1 Mandatory Functional Requirements:

- **MF1:** Locate Oil/Gas wellhead infrastructure with known heading in $25m^2$ area
- **MF2:** Autonomously maneuver to wellhead within one hour
- **MF3:** Positively ID as correct wellhead with 90% confidence
- **MF4:** Maintain hover position over dock within +/- 1m of dock position continuously
- **MF5:** Rigidly dock in five degrees of freedom
- **MF6:** Provide status feedback to user of current state at 0.1Hz

Changes in mandatory functional requirements since PDR:

The search area for MF1 was changed from $50m^2$ to $25m^2$ in order to accommodate the area in the B-level. This is the same area that we used for the Fall Validation Experiments. This change will allow us to test in an area that we have constant access to. It is an area that is familiar to us and that we have some control over. The area will also be the same one that we use to test whenever the net gets moved to, so we have some consistency in our test patterns. This will also ensure that we have ample opportunity for tests throughout the semester.

A performance metric was added to MF4 to specify that we want to maintain a hover position over the dock within +/- 1m of dock position continuously. We had received some feedback from the PDR that not all of our requirements had performance metrics, so we made sure that we put realistic and important requirements that pushed our design forward. If the drone can hover within +/- 1m of the dock position continuously once it has discovered the wellhead and reached the dock position, the drone will be able to make a descent that will allow it to dock successfully. This performance metric is what we will use to determine when we have moved from the “get into docking position” state to the “docking” state. This metric will allow the drone to be able to quantitatively determine when it is ready to dock. It will also ensure that the tag always remains within the field of view of the camera.

These changes will allow us to be more effective in the spring. The changes will improve the team’s test effectiveness.

4.2 Desired Functional Requirements:

- **DF1:** Locate Oil/Gas wellhead infrastructure in low visibility with unknown heading in $25m^2$ area
- **DF2:** Positively ID as correct wellhead from visual object recognition with 90% confidence
- **DF3:** Align with dock located at known radius but unknown angle from wellhead within +/- 1m

Changes in desired functional requirements since PDR:

There were many changes to the desired functional requirements since the PDR. Many of these changes were to add performance metrics in order to flesh out the requirements and give the team something to test against for the Spring Validation Experiment.

The same performance metric for MF1 was given to DF1 which defines the search are to be $25m^2$. The main difference between MF1 and DF1 is that the searching is done in low visibility with an unknown heading. This requirement has been more fleshed out since the PDR. The team has decided that the maximum scope that we want to accomplish for the searching subsystem is that the robot can be placed into the search area without knowing its initial state and while having degraded vision. We believe that this will help us mimic a situation that we might have in an underwater environment.

The team also added a performance metric to DF2 so that we require visual object recognition with 90% confidence. This requirement is part of our maximum scope where we do not want to have to rely on specific tags placed on the wellhead in order to get accurate object recognition. This performance metric is the same as the performance metric for MF3, which will ensure that we will be able to recognize the dock with a high level of accuracy.

A performance metric was also added to DF3 which is the same as the performance metric for MF4. This performance metric deals with determining the pre-docking position. The explanation of this performance metric was described above in the mandatory functional requirements section. DF3 represents the maximum scope of the project. In the minimum scope, the dock will be placed at a known radius and angle from the wellhead. The wellhead will be placed in the corner of the search area, so that the robot will always be able to see it from the front. This will allow the robot to not have to rotate during flight. Rotation during flight can throw off optical flow in a way that may render the control of the robot ineffective. In our maximum scope, the dock can be located within any angle from the wellhead in a known radius. This will mean that the quadcopter will have to rotate and search all around the wellhead in order to locate the dock. This kind of robustness to the system is a stretch goal that we have.

4.3 Mandatory Non-Functional Requirements:

- **MNF1:** Operable with simple graphical user interface
- **MNF2:** Provides emergency stop for system with less than one second lag
- **MNF3:** Operable by a single person

Changes in the mandatory non-functional requirements since the PDR:

There have not been many changes to the mandatory non-functional requirements since the PDR. Most of the mandatory non-functional requirements deal with operation. The team has a strong desire to make operation more efficient in our solution compared to the current solution which requires a large ship and skilled operators to run the robot. The team still desires that the robot be operable by a single person with a simple graphical interface. This interface should be able to provide very minimal input from the operator while the intelligence of the drone takes over the rest of the functionality.

The major change in the mandatory non-functional requirements is that a performance metric was added to MNF2 which was missing before. The impetus for this change was specifically provided by feedback from our PDR. The main reason that this requirement is so important to the team is that it is a very slow process to go from automated to manual control in the Iris+. It is also a very slow process to get the drone to unarm. It takes over three seconds to get the Iris+ to disarm and go to manual control. This is too slow for our needs. In those three seconds, the drone will surely fall and possibly endanger a person, the robot, or another's personal project. By providing a solution to disarm the drone with an emergency stop that has less than one second of lag, the team will drastically improve the safety of the drone during testing and other operation.

4.4 Desired Non-Functional Requirements:

- **DNF1:** Reduce operator cost by at least one-half
- **DNF2:** Simulate low-visibility: Unable to get visual feed beyond 3m from camera/quadrotor

Changes in the desired non-functional requirements since the PDR:

There have been more changes in the desired non-functional requirements than the mandatory non-functional requirements since the PDR. Most of the changes have been to flesh out what the team would like to accomplish in our maximum scope during the spring semester.

The first change is that the team added a performance metric to DNF1. This metric fleshed out that we would like to reduce operator cost by at least one-half. This performance metric is admittedly arbitrary. Our solution will not provide everything that an Oil/Gas company would need to replace its current system with an autonomous one. We are here acknowledging that the overall motivation for our project is to create a system that could replace the current systems for wellhead intervention while drastically reducing the cost of said system. We believe that reducing it by half is a fairly obvious arbitrary requirement. It is one, however, that we may not be able to quantify exactly for our system, which is why it is a desired and not a mandatory functional requirement. Anything that we build will of course meet this requirement, but the R&D required to get a complete system is an unknown to us.

The more useful performance metric for desired non-functional requirements is for DNF2. This metric quantifies what low-visibility will mean for our system. In an underwater environment, low-visibility is something that any robot will face. In our maximum scope, we would like to tackle this challenge. For the team this means that the robot will not recognize visual information that the robot receives beyond 3m from the camera or quadrotor. There are a few ways that this can be accomplished. One way would be to have a light on the quadrotor near the camera while the room it searches is in darkness. This is not ideal, because it will render the optical flow ineffective. A more elegant solution can be accomplished in software. We can use the depth information from the RGB-D sensor on the front of the quadrotor to determine the distance of objects in front of the quadrotor. The robot can then apply a Gaussian blur on the images or just ignore them completely. This is can all be done in software. The team believes that 3m will provide what we need to simulate low visibility.

5. Functional Architecture

Figure 5 shows the reduced functional architecture for the team's project. The functional architecture is broken down into three major sub-functions: "Locate and Identify Desired Wellhead", "Move to Pre-Docking Position", and "Dock on Wellhead".

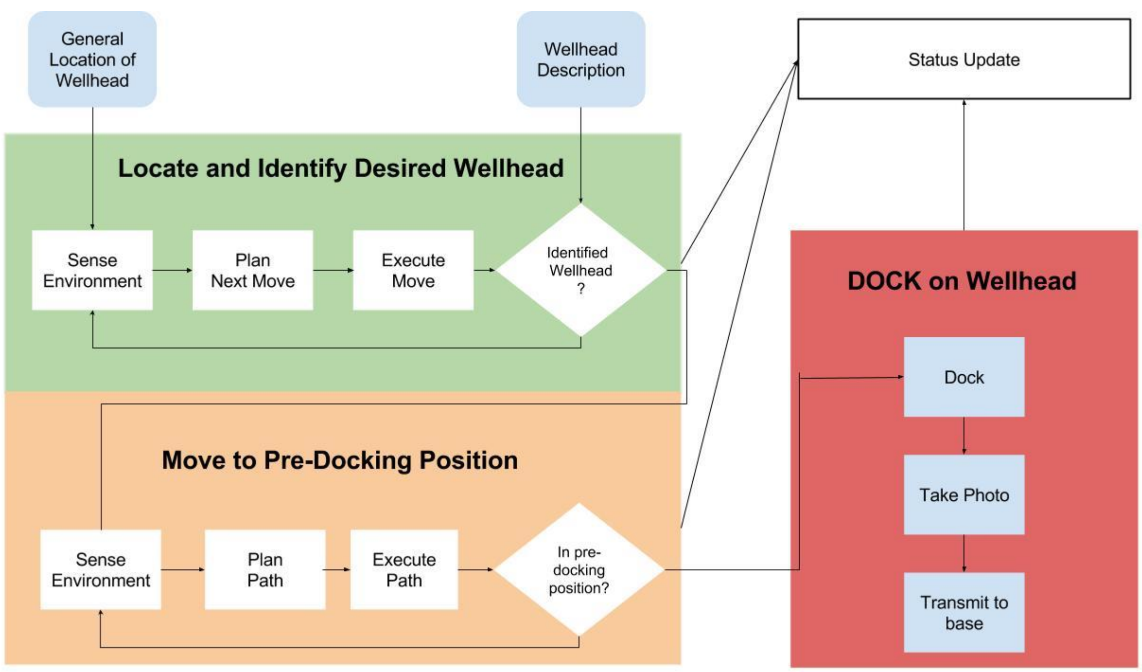


Figure 5) Simplified Functional Architecture

Figure 6 shows an expanded version of the "Locate and Identify Desired Wellhead" sub-function.

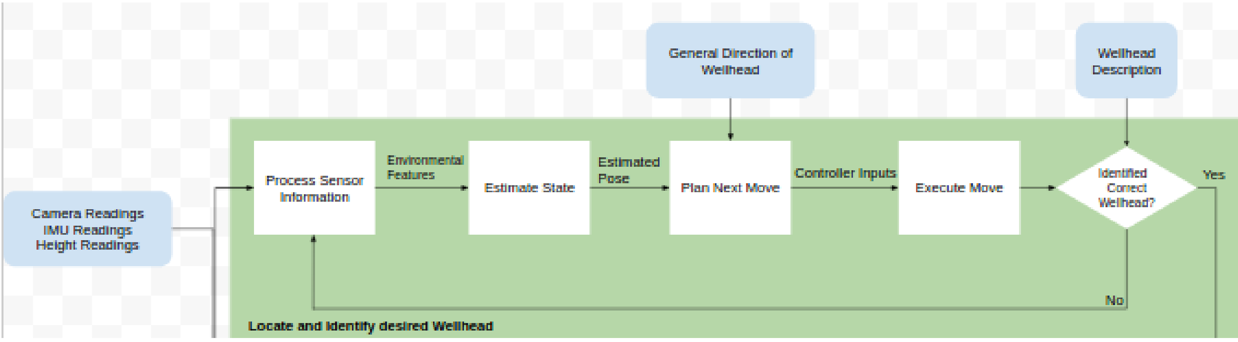


Figure 6) Locate and Identify Desired Wellhead Subfunction

Figure 6 clearly shows the flow of information into and throughout the sub-function. The main inputs to the system are: "Camera Readings, IMU Readings, and Height Readings", "General Direction of Wellhead", and "Wellhead Description". Internally information is passed between each block in the fashion of: sense, plan, and act. This block is executed on a loop until the robot has identified the correct wellhead. Once it has identified the wellhead, the system changes to the "Move to Pre-Docking Position" state as shown in the figure below.

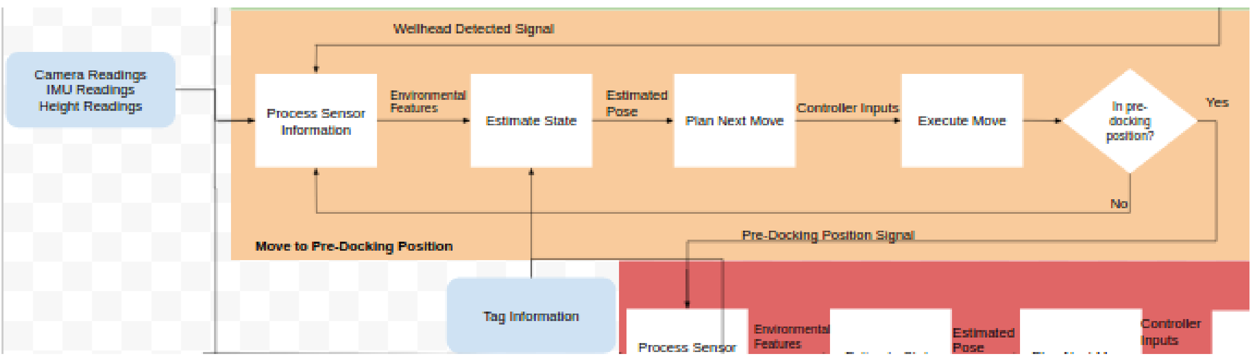


Figure 7) Move to Pre-Docking Position Subfunction

In Figure 7, the flow of information for the "Move to Pre-Docking Position" sub-function can be clearly seen. The inputs to this sub-function are: "Camera Readings, IMU Readings, and Height Readings" and "Tag Information". This tag information is for the dock. The internal flow of information is the same loop as the "Locate and Identify Desired Wellhead" sub-function, except for the stopping criteria. The stopping criteria is "in pre-docking position" which is determined by mandatory functional requirement 4: Maintain hover position over dock within +/-1m of dock position continuously. Once the robot has reached the stopping criteria it moves into the "Dock on Wellhead" state as shown in the figure below.

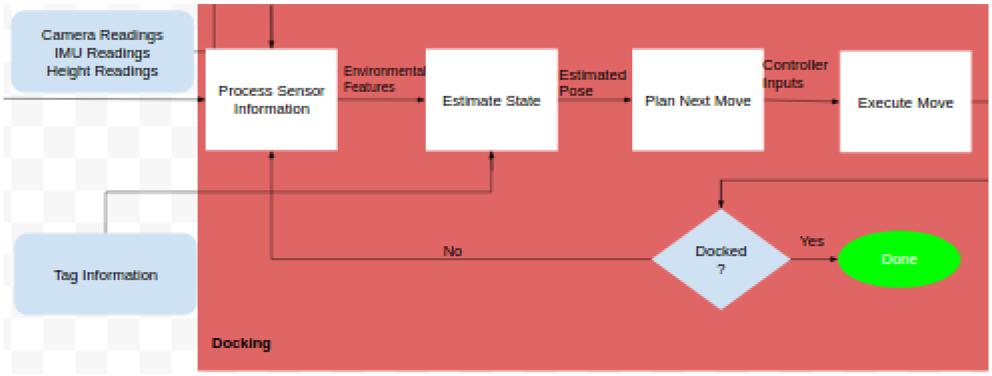


Figure 8) Docking Subfunction

Figure 8, above, shows the final sub-function and state of the system, docking. Once the robot has reached the pre-docking position it will make its docking descent and complete its task of docking. The main inputs to the system are: “Camera Readings, IMU Readings, and Height Readings” and “Tag Information”. Again this tag information is provided for the dock and can be in the form of an AR tag, LED or IR lights, etc. This state will continuously check to see if the drone has docked. The internal flow of information again is passed back and forth in the plan, sense, and act paradigm. Once it has effectively docked, it will enter a done state. The done state will capture an image of the wellhead and transmit it back to the user. This image will be displayed on the simple graphical interface that corresponds to manual non-functional requirement 1.

6. Cyberphysical Architecture

The cyberphysical architecture, shown in Figure 9, has been broken down into five main parts: Infrastructures, sensors, single board computer, motor control & UAV, and user interface. We have organized our cyberphysical architecture based on how the systems are physically organized and interact.

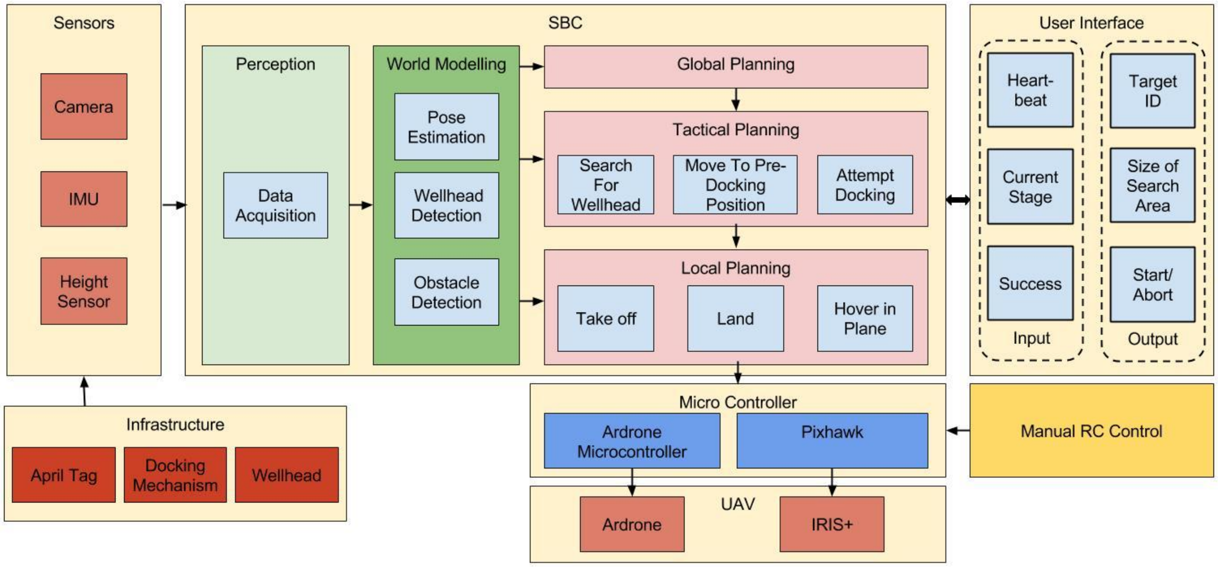


Figure 9) Cyberphysical Architecture

The Infrastructure consists of the main none onboard robotics systems that exist to support the UAV. The infrastructure comprises of the april tags, the docking mechanism, and the wellhead. The aprils tags signify then coordinate the beacon based navigation towards the wellhead and the dock.

The sensors consists of the camera, IMU, and height and optical-flow sensor. The downward facing camera allows the drone to view the dock and ground april tags. The IMU is used for the drones state-estimation. Height and follow sensor is also used for the state estimation, but also use for the localization and height stabilization.

For the single board computer we have an underlying software architecture that implement the ‘Toaster-Wedding Cake’ model. The ‘Toaster-Wedding Cake’ model constitutes the flow of data and information in a sense-plan-act format. The toaster is the vertical blocks of perception and world mapping. The systems perceives the environment through the sensors, then develops a model of that environment. The wedding the flow of data through the high level global plan to the low level local planning. This planning structure dictates the actuation the system will have on the environment.

The motor control and the UAV are the drones itself and its control. This section are broken into two parts. The AR.Drone2 is the drone we used for testing of high level searching algorithms and exists as a backup if Iris+ cannot perform the necessary tasks.

The high level software will be run on the single-board computer with information being passed to it from the wireless communication and low level microcontroller. The User Interface also connects to the wireless communicator and goes from the user to the single board computer to be used for the high level software. The single board computer through the wireless communication module also sends data to the user interface.

7. Current System Status:

7.1 Targeted System Requirements:

We decided to drive our work based on completing subsystem functionalities rather than targeting specific requirements. Based on the targeted subsystems, Table 1 shows the status of the functional and nonfunctional requirements that we met after the fall validation experiment. The details of the subsystems targeted are covered in the next section. We have not met many of our requirements because we have focused on completing the hardware setup and implementing the subsystem functionalities. Each subsystem functionality is an independent state and achieving all the requirements will involve running a state machine on these independent states.

Table 1) Targeted Systems Requirements

MF1. Locate Oil/Gas wellhead infrastructure with known heading in 25 m^2 area
MF2. Autonomously maneuver to wellhead within 1 hour
MF3. Positively ID as correct wellhead with 90% confidence
MF4. Maintain hover position over dock within +/- 1m
MF5. Rigidly dock in 5 DOF
MF6. Provide status feedback to user of current state at 0.1 Hz
MNF1. Operable with simple graphical user interface
MNF2. Provides emergency stop for system with less than 1 second lag
MNF3. Operable by a single person

Completed	In-Progress	Not Started

7.2 Subsystem Descriptions/Depictions:

Figure 10 shows the subsystem status in our cyberphysical architecture.

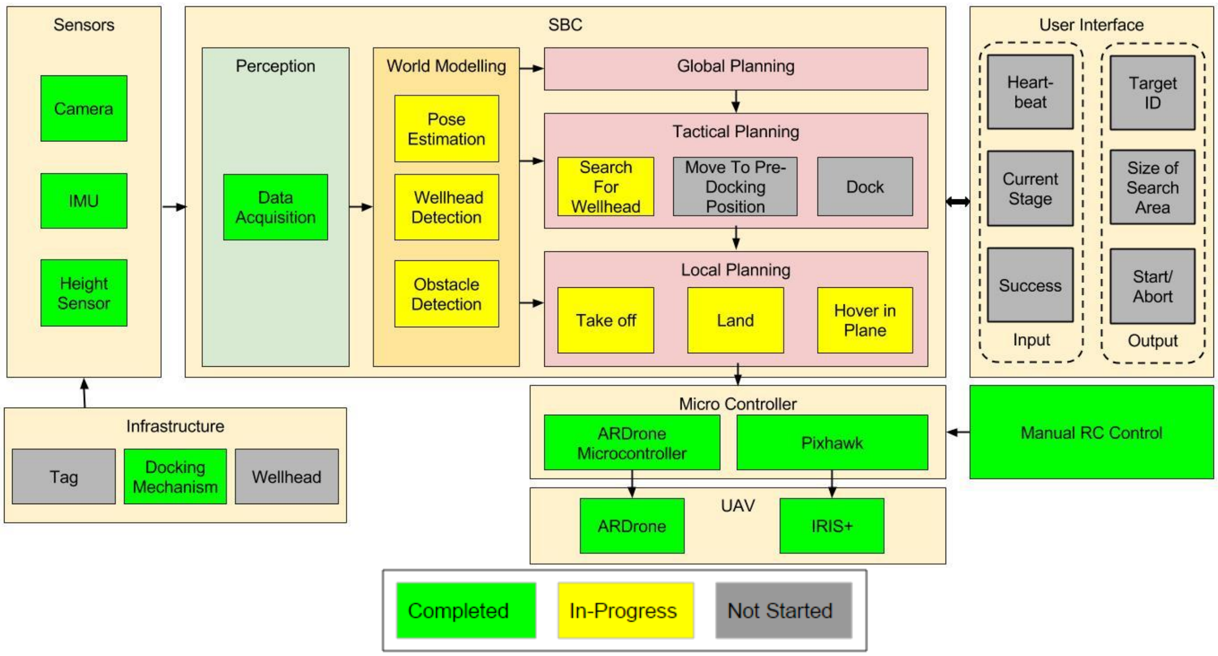


Figure 10) Current Subsystem Status

7.2.1. Infrastructure Subsystem

Landing a quadrotor at desired a location is a hard problem because of the turbulence in the airflow of the thrusters when the quadrotor is close to the ground. Hence, one of main design criterion was to be able to tolerate large variance in pose at which the quadrotor can approach the

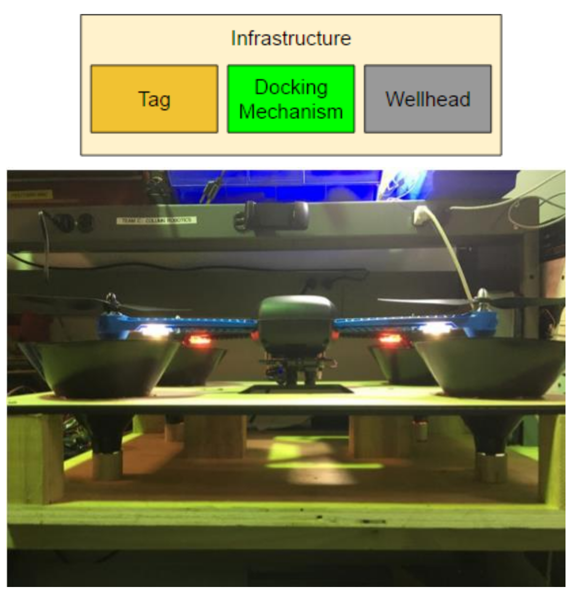


Figure 11) Infrastructure Subsystem Status

dock. To meet this requirement for the docking mechanism, we are using four cones to funnel the quadrotor down to the desired location, as shown in Figure 11. Using this strategy we can tolerate larger tracking errors in our control algorithm during landing. We will be manufacturing a mock-up of the wellhead infrastructure in the next semester. The details of the tag are covered in the perception subsystem.

7.2.2 Sensor Subsystem

Table 2 shows the description of the components of the sensor subsystem, and Figure 12 shows the components of the sensor subsystem mounted on the Iris+.

Table 2) Sensor Subsystem Description

Sensor	Sony Playstation Eye	PIXHAWK	PX4FLOW KIT
Function	Downward camera whose feed is used to detect the APRIL Tags	Flight controller to run the attitude control loop of the quadrotor	Sensor to provide visual odometry estimates
Features	Supports a framerate of 120hz at 320x240 resolution.	ST Micro L3GD20 3-axis 16-bit gyroscope ST Micro LSM303D 3-axis 14-bit accelerometer / magnetometer Invensense MPU 6000 3-axis accelerometer/gyroscope MEAS MS5611 barometer	PX4FLOW V1.3.1 optical flow sensor smart camera compatible with PX4 PIXHAWK flight controller. Used to obtain visual odometry updates
Image	 *source: http://amazon.com*	 *source: https://pixhawk.org*	 *source: https://pixhawk.org*

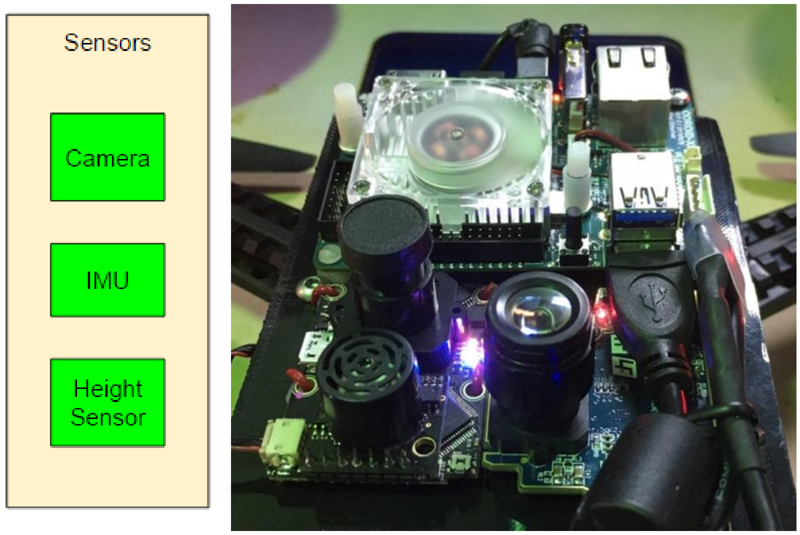


Figure 12) Sensor Subsystem

7.2.3 World Modeling Subsystem

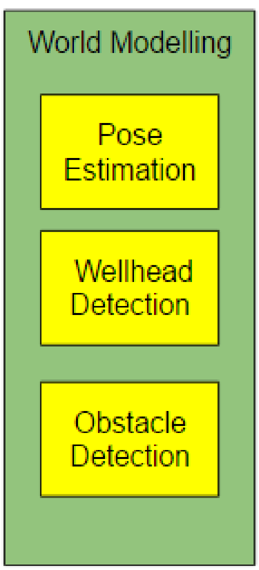


Figure 13) World Modelling Subsystem

As shown in Figure 13, the world modelling subsystem consists of the following three nodes:

1. Pose Estimation: This node will estimate the pose of the quadrotor in the world frame.
2. Wellhead Detection: This node will estimate the position of the wellhead in the quadrotor frame.
3. Obstacle Avoidance: This node will update the occupancy grid with the obstacles, once they are detected.

We did not focus on implementing these systems during the fall semester, however, we have experimented with some algorithms that will help us implement this system. The following are the algorithms that we explored:

1. APRIL tag detection

Reference [2] shows a library by Mike Kaess, written in C++ that detects APRIL tags and estimates the pose of the robot. We can use this to detect the wellhead and the docking mechanism

2. Lucas-Kanade based optical flow

We can use this algorithm to estimate the velocity of the quadrotor using the camera feed. Scale estimation is one of the major problems with this algorithms. We are using the PX4Flow sensor that implements this algorithm and estimates the scale using an integrated ultrasonic sensor which measures the distance to the ground. After consulting last year's MRSD teams, we are confident that this solution works.

3. LSD-SLAM (Large Scale Semi-direct Simultaneous Localization and Mapping)

This algorithm extends the idea of PTAM (Parallel Tracking and Mapping) but instead of using features like SIFT/SURF which are computationally intensive, it directly uses intensity values of the images. Hence, many research groups have been able to run this algorithm in real time on quadrotors. We can use this algorithm to detect obstacles and correct for the drift in position estimates that occurs due to integration of velocity estimates from optical flow.

7.2.4 Global Planning Subsystem

As shown in Figure 14, we are using a 3 layered architecture for the planning. Each layer acts like a state machine for the layer below it. For example, the global planning starts with "Search For Wellhead", on finding the wellhead, it transitions to the "Move To Pre-Docking Position". On reaching pre-docking position, it transitions to the "Attempt Docking" state. Similarly, "Search For Wellhead" is a state machine that uses "Take off" and "Hover in Plane" states. For this semester we have implemented the entire local planning and hence, most of tactical planning on the AR.Drone. We demonstrated this functionality in FVE by doing a lawn mower search using the AR.Drone. The details of this are covered in the next section. Our focus for the next semester is going to be to implement this on the IRIS+.

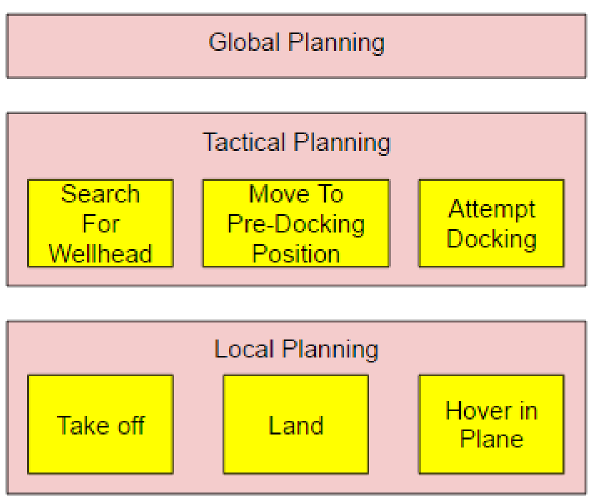


Figure 14) Planning Subsystem

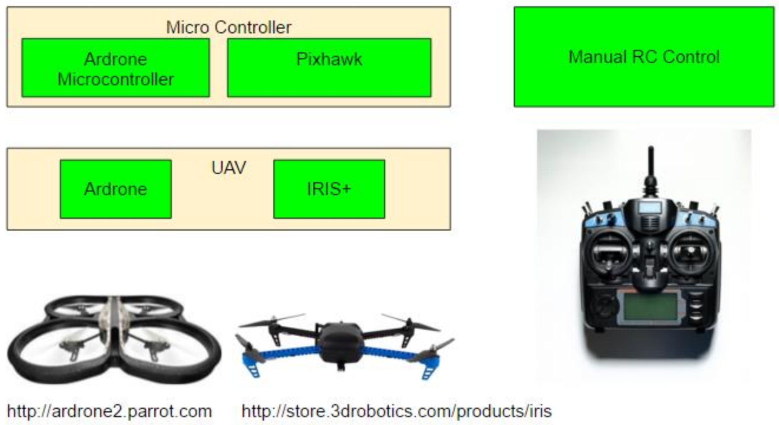


Figure 15) Hardware Subystem

7.2.5 Hardware Subsystem

The figure 15 shows the components of hardware subsystem. The AR.Drone is reliable quadrotor system that we obtain from the MRSD storage at no cost to us. The AR.Drone acted as our initial test bed to run our high level search algorithms and code. The AR.Drone is also our fall back and risk mitigations if the Iris+ drone cannot perform our desired tasks. The drone does not require any extra hardware and is controlled via wifi from a host computer. It has a forward facing and downward facing cameras, and the downward facing camera doubles as an optical flow sensor.

The Iris+ drone is a commercially bought quadrotor that we are modifying to with sensors and a SBC. The Iris+ drone's motors' low level controls are commanded via Pixhawk, which also has a compilation of various sensors, such as 9 axis IMU, and barometers. It also handles our communication to the RC controller. The SBC will be communicating to the Pixhawk via UART to control the drone's movements.

7.3 Modeling, Analysis and Testing

Initially we were trying to track a trajectory by doing closed loop control by using feedback from visual odometry based on optical flow. To evaluate the performance of our algorithm we moved the quadrotor in a 3x3m square, 2 times. The Figure 16 shows the result of our experiment. It can be inferred from the graph that we have a drift of 1m for a displacement of 1m. Clearly, we cannot implement our lawn mower search with such a large magnitude of drift.

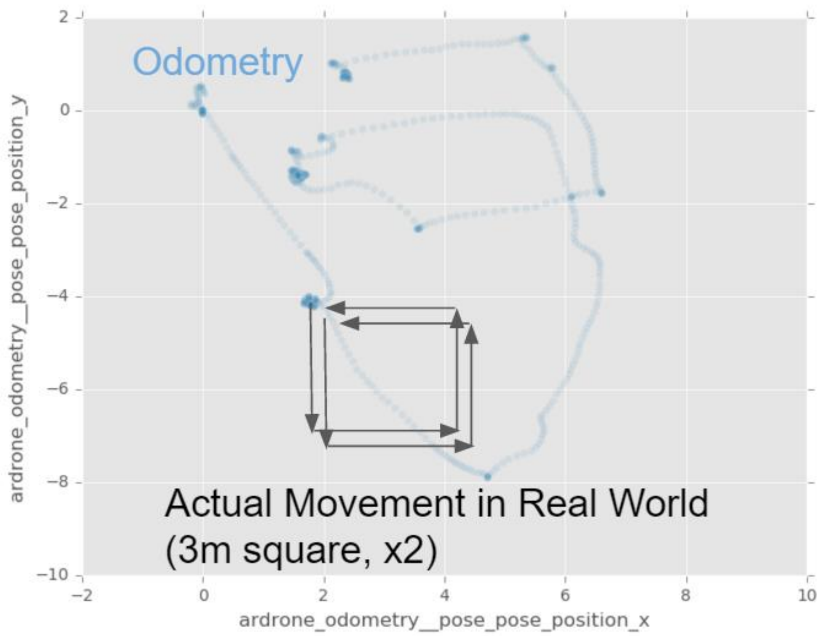


Figure 16) X vs Y Odometry Readings From Flight Test

We solved this issue by using extended kalman filters to fuse the odometry estimates with the motion model of the quadrotor. The kalman filter equations used by the algorithm are shown below:

Notation:

A_t : *Motion Model*
B_t : *Control Input Model*
μ_t : *State Mean*
Σ_t : *State Variance*
Q_t : *Motion Model Noise*
C_t : *Observation Model*
R : *Observation Noise*
K_t : *Kalman Gain*
$z_t - C_t\bar{\mu}_t$: *Innovation*

$$\mathrm{KalmanFilter}(\mu_{t-1}, \Sigma_{t-1}, u_t, z_t)$$

Prediction

$$\bar{\mu}_t = A_t \mu_{t-1} + B u_t$$

$$\bar{\Sigma}_t = A_t \Sigma_{t-1} A_t^\top + Q_t$$

Gain

$$K_t = \bar{\Sigma}_t C_t^\top (C_t \bar{\Sigma}_t C_t^\top + R)^{-1}$$

Update

$$\mu_t = \bar{\mu}_t + K_t (z_t - C_t \bar{\mu}_t)$$

$$\Sigma_t = (I - K_t C_t)\bar{\Sigma}_t$$

Slide courtesy Kris Kitani

The A and B matrices for the motion model and the control input model were obtained by linearizing the quadrotor dynamics about the hover position using Taylor's expansion. This was implemented using the tum_ardrone [3] API. The final result of the tracking algorithm running with the EKF can be seen in the video on our website.

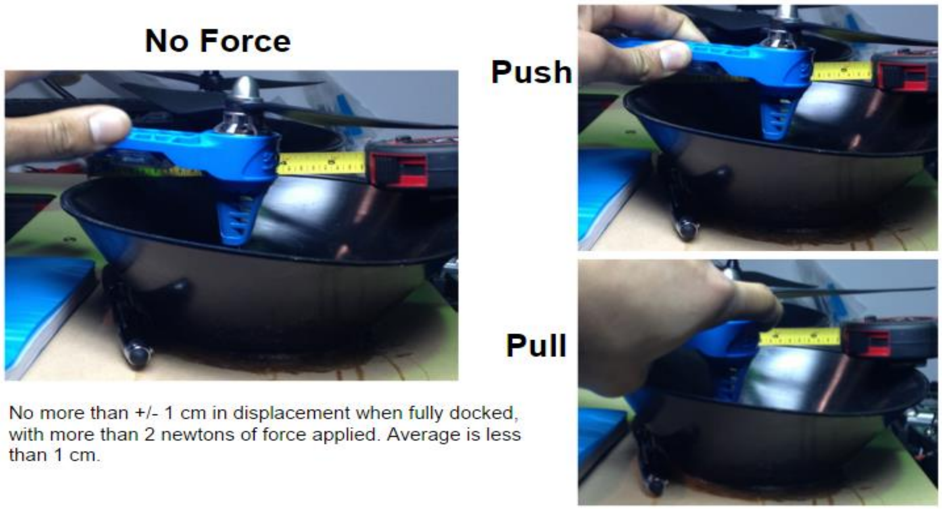


Figure 17) Docking Mechanism Compliance Test

Figure 17 shows the results of our compliance test performed to validate that we meet our functional requirement of the docking subsystem. Figure 18 shows the images of the drop test performed using IRIS+ quadrotor. As shown in the figure, the docking mechanism was successfully able to funnel the quadrotor to the center of mechanism.

Figure 18) Docking Mechanism Drop Test

7.4 FVE Performance Evaluation:

For the Fall Validation Experiment, we broke it up into two parts; the Autonomous searching demo with the AR.Drone, and the hardware setup on the Iris+ drone along with the dock prototype. For the demo with the AR.Drone, the drone needed to takeoff perform a lawn mower search where it snaked through a cordoned off portion of the B-level bay area, and then returned to its original take off position. For this demo the AR.Drone needed to show reliable state estimation and localization to determine its position in space and the desired path it must take.

During the FVE, the success criteria was defined in terms of speed, repeatability, robustness, and accuracy. The necessary success conditions, as defined in our evaluation, were:

1. **Successful takeoff** and hover of drone under manual control.
2. Drone autonomously **completes 4 search sweeps** of length > 4m each.
3. Drone path during search sweeps **does not overlap with itself**.
4. Drone successfully **avoided contact** with walls of hallway.
5. Clear downward-facing **video feed displayed** during entire search process.
6. Full search process **succeeded within 10 minutes** of drone takeoff.

The demo met all the conditions successfully. During the demo, the drone took off under manual control. The user brought the drone to the proper height. The search sequence is initiated. The drone begins to sweep in its lawn mower search. It avoids walls and obstacles. In each sweep it takes a unique air route avoiding overlapping itself. Throughout the whole sweep process the drone streams video for the downward facing camera to a host computer. The full search without setup takes an average of two minutes. The drone sweeps four times through the length of the quarantined area. A video of the drone completing this half of the FVE can be seen on the MRSD Team C webpage.

For the prototype dock demo and the hardware-setup on Iris+, the demo needed to showcase the key functionalities of the sensors, SBC and the host computer. The drone video orientation estimation must show valid orientation in roll pitch and yaw. The dock prototype needed to constrain the drone by its landing gear in 5 degrees of freedom. The necessary success condition for the hardware-setup and dock prototype, as defined in our evaluation, were met:

1. Iris+ constrained within **+/- 1cm** in all directions by dock.
   Tighter than required +/-2 cm in dock (5 DOF).
2. Valid **orientation estimate** and image (taken from the camera on the drone) is displayed on the PC
   Showcased the valid orientations: (Roll, Pitch, Yaw) = (90,0,0) and (0,90,0).
3. ‘rostopic hz’ command shows **1.09Hz** for downward facing camera feed:
   Faster than required 0.1Hz on relevant topic on PC as specified in the requirements.

The dock also withstood a meter high drop test with the drone. The dock cones funneled the drones landing to the desired position. The landing gear combined with the dock cones absorbed a great amount the impact energy. This allowed for the drone to safely land with little abrupt impulse, protecting the sensitive components on its hardware.

7.5 Strong and Weak Points

7.5.1 System Strengths:

- **Robust lawnmower search with AR.Drone**
  As could be seen in our FVE and promotional video, we were successful in achieving a robust “lawnmower” search pattern with our AR.Drone platform. With this critical requirement met, we have a high level of confidence that our fallback system will be able to meet all “must” requirements for our system.

- **Shock absorbtion quality of the dock**
  Our dock design has met its design criteria well, and in fact performed beyond our expectations. One example of this is the significant shock-absorption capabilities of the dock, which reduces the forces experienced during a hard landing.

- **Well integrated power system**
  Our power system fits entirely inside the original chassis of the drone itself, reducing the need for delicate external components.

- **Compact design for sensor and SBC mounting**
  A 3d-printed mounting plate and careful component selection has enabled a compact, sleek solution for our add-on sensors and single-board computer

7.5.2 System Weaknesses:

- **Automated Iris+ control untested**
  Although our choice of a well-tested, widely used open source drone platform reduces our risk exposure significantly, the fact that we have not yet demonstrated automatic control of our Iris+ platform is a significant weakness of our current system. Unexpected events or problems can happen, and we need to demonstrate this capability as soon as possible.

- **Small backwards drift of AR.Drone**
  Our AR.Drone platform shows a consistent, systematic backwards drift of its onboard odometry during flight, which complicates the path planning code. Due to the consistent nature of the problem this can be compensated for, however it is definitely sub-optimal.

- **Jerky waypoint navigation**
  Our use of an overly-simple PID + waypoint-based path planning algorithm has resulted in relatively jerky movement of the AR.Drone during its path planning. We intend to improve the smoothness of the trajectory during subsequent quarters.

8. Project Management

8.1 Work Breakdown Structure

Figure 19 shows the work breakdown structure for Fall, and Figure 20 shows the work breakdown structure for Spring.

WBS Summary

	WBS for Fall Validation Experiment	Status:	**PDR**	**Now**
1	**Open-loop ARDrone Control: takeoff, land, move from PC**			
1.1	Low-level open-loop control / takeoff in ROS	Erik/Job/R		
1.2	Display ROS node graph	Erik		
2	**Fall AR.Drone Position X,Y Movement Demo**			
2.1	AR.Drone relative odometry working in ROS	Erik (Cole)		
2.2	Closed loop on absolute position control	Cole (Erik)		
2.3	Integrate AR.Drone demo subsystems	Erik		
3	**Hardware and ROS Setup on Iris+**			
3.1	Pixhawk -> SBC -> PC ROS setup + sensor display	Rohan (Job)		
3.2	Completed Iris+ Hardware Setup	Job (Rohan)		
4	**Dock Prototype**			
4.1	Formulate Dock Design Criteria	Job/Team	N/A	
4.2	Dock Internal CODR	Team	N/A	
4.3	Manufacturable CAD Model of Dock	Job		
4.4	Tested, working physical prototype	Job		
5	**Non-Demo Focus Areas:**			
5.1	Iris+ Relative Odometry	Rohan		
5.2	Stable Open-Loop Control of Iris+			
5.3	Integrated closed-loop position control of Iris+			
5.4	Searching for tag on ground with AR.Drone	Cole	N/A	

Figure 19) Work Breakdown Structure for Fall

	WBS for Spring Validation Experiment		Status
1	**Hardware and ROS Setup on Iris+**		
1.1	Iris+ Hardware Setup	Rohan	
1.2	Completed Iris+ Interface	Job	
1.3	Build backup Iris+	Erik	
2	**Low Level Control of Iris+**		
2.1	Iris+ Relative Odometry	Cole	
2.2	Open-Loop Velocity Control + Hover	Rohan	
2.3	Closed-Loop Position Control (PID)	Job	
2.4	Advanced Trajectory Control (Lattice Planner)	Cole	
3	**Simple Cone Search (Dock/Wellhead)**		
3.1	Search for wellhead + hold position (front facing camer	Cole	
3.2	Search for dock + hold position (bottom facing camera	Rohan	
4	**Autonomous Docking**		
4.1	Pose estimation during docking phase	Rohan	
4.2	Complete automated docking sequence	Erik	
5	**Smart Cone Search**		
5.1	Avoid walls during search	Erik	
5.2	Integrate wellhead search -> dock lock-on	Job	
6	**Integrated Search and Dock**		
6.1	Search for dock + hold position (bottom facing camera	Rohan	
6.2	Polish and test final demo	Erik	

Figure 20) Work Breakdown Structure for Spring

8.2 Schedule Status

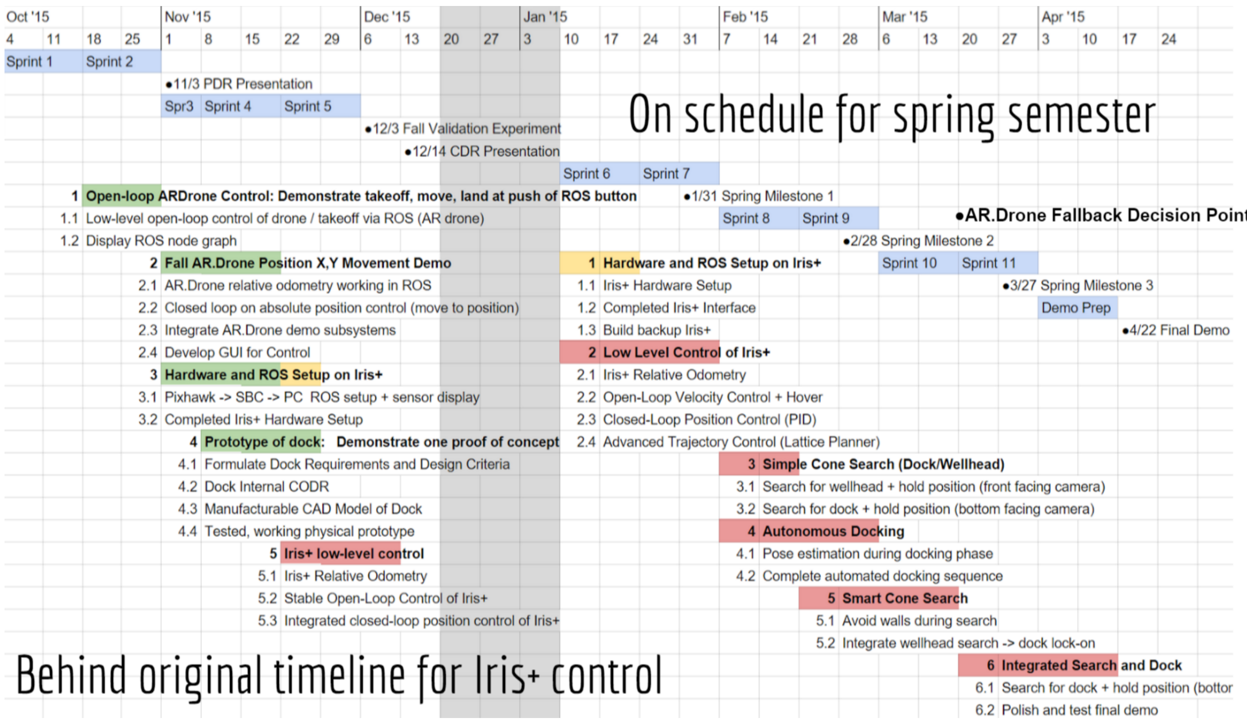


Figure 21) Fall and Spring Schedule

As shown in Figure 21, we have split our work time into 2-week sprints. We have a total of 6 of these sprints between January and the start of April, with an extra two-weeks is also set aside for final demo preparations.

Although we were successful in completing our demonstration and system requirements for the fall quarter, we are behind our original timeline for working automatic control of the Iris+. As a result, this work has shifted to the beginning of the spring semester.

One significant date is March 20th, at which point we will decide whether or not to abandon the Iris+ and focus our efforts to the backup AR.Drone platform. If we do not have autonomous flight demonstrated and a high-confidence path forwards for autonomous docking by this point, we will focus all work on our backup platform.

The test plan section below breaks out our intended activities during each of these sprints.

8.3 Test Plan

Table 3 identifies key capability milestones for the spring-semester Progress Reviews.

Table 3) Test Plan for Spring

Deadline	**Deliverable Functionality**	**Method to Test**
Late January Progress Review 7	Low level control of Iris+. Backup Iris+ hardware completed	Stable, teleoperated control of iris+ via ROS. Demonstrate in net.
Mid-February Progress review 8	Simple cone search with Iris+	Cone-shaped search pattern approaching wellhead; stop when wellhead tag identified.
Late February Progress Review 9	Autonomous docking of Iris+	Autonomously recognize dock from above, approach and land on dock, confirm rigidity in 5 DOF
Mid-March Progress Review 10	Smart cone search with Iris+ **(AR.Drone Fallback Decision Point)**	Iris+ searches for wellhead and locks on dock position while avoiding hitting walls.
Early April Progress Review 11	Integrated Search and Dock	Iris+ searches for wellhead, locks on dock position, and autonomously docks.
Mid April Progress Review 12	Integrated working system	Full demo: Take off, Search for wellhead, Orient to dock, land on dock, send signal.
April 22 and April 29 Spring Validation Experiment	Demonstration of integrated system	Same as above, but better!

8.3.1 Spring Validation Experiment

Needed Equipment:

1. Iris+ with mounted sensors and computer hardware
2. wellhead
3. dock
4. caution tape

Operational Area:

25m^2 in B - Level Basement

Test Process:

1. Cordon off section of hallway
2. Place wellhead at one corner of search area and dock 1m in front of the wellhead
3. Place Iris+ on ground at opposite corner of search area facing wellhead within +/- 5 degrees
4. Hit START button on PC to initiate sequence
5. Confirm Iris+ lifts off and begins searching for wellhead (marker)
6. Confirm Iris+ arrives within 3 meter radius of wellhead
7. Confirm Iris+ orients above dock in pre-docking position (within 1 meter of dock)
8. Confirm Iris+ successfully lands in dock, constrained in 5 DOF

Success Conditions / Metrics:

Mandatory:

1. Iris+ autonomously takes off from ground
2. Iris+ arrives within 3 meter radius of wellhead
3. Dock with docking station, constrained in 5 DOF

Desired:

1. Dock constraints 6 DOF
2. Successfully avoid obstacles

Figure 22 shows a schematic view of our spring semester validation experiment.

In its initial condition, the drone will not be able to detect the wellhead due to reduced visibility, simulating the underwater environment of a real wellhead. It will be placed with an initial +/- 5 degrees of accuracy.

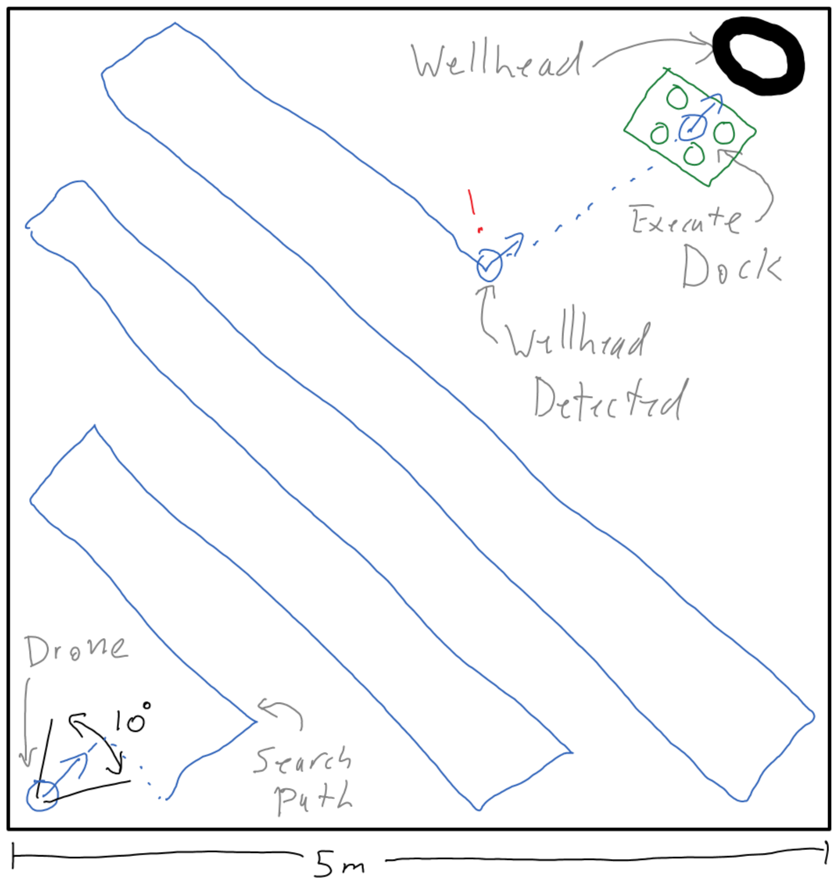


Figure 22) Sketch of Spring Validation Experiment

The drone will follow a search path similar to the one sketched in blue above, and once it detects the wellhead (under 3 meters from wellhead) it will approach the dock, which is located in front of the wellhead. Once the drone has a visual lock on the dock with it's downward-facing camera, it will execute an automatic docking sequence and arrive in its final docked position.

8.4 Budget Status

Total budget: $4000
Total spent to date: $2807.06, as shown in Table 4.
Big ticket items:

- 3DR Iris+ Drone x2: $1200
- Minnowboard Max x86 SBC: $150
- Odroid XU-4 Arm SBC x2: $166
- NicaDrone Magnet: $90
- PX4 Flow Optical Flow Camera x2: $300

Table 4) Full Budget

Quantity	Part Name	Unit Price	Total Price
1	3DR IRIS+ Quadcopter	$599.99	$599.99
4	3DR IRIS+ Propellers	$9.99	$39.96
1	MINNOWBOARD-MAX-DUAL	$145.95	$145.95
1	Odroid XU-4 Board	$83.00	$83.00
1	Edimax EW-7811Un	$9.99	$9.99
1	Playstation Eye	$8.95	$8.95
1	micro HDMI cable (	$6.49	$6.49 + shipping
8	3DR IRIS+ Propellers	$9.99	$79.92
1	PX4Flow	$149.00	$149.00
2	Iris+ Battery	$40.00	$80.00
1	Power Module	$25.00	$25.00
1	Battery Charger	$25.00	$25.00
1	Guard Props	$25.00	$25.00
1	Small soccer cones	$10.00	$10.00
2	NicaDrone Perment Magnet	$45.00	$90.00
1	shallow soccer cones	$20.00	$20.00
1	Bowl	$16.69	$16.69
12	10uF Tantalum Capacitor	$0.99	$11.88
3	16V TVS Diode	$0.45	$1.35
3	7.5V TVS Diode	$0.47	$1.41
3	8A Fuse	$0.89	$2.67
3	5A Fuse	$0.70	$2.10
3	*XT60 Connector*	$1.50	$4.50
3	CGRID SL Connector - Male	$0.26	$0.78
3	CGRID SL Connector - Female	$0.88	$2.64
3	Voltage Regulator	$3.70	$11.10
3	5.5mmx2.1mm 5V DC Barrel Connector	$3.95	$11.85
1	High-Strength Multipurpose Neoprene Rubber	$10.42	$10.42
4	Electrically Conductive ABS/PVC	$13.57	$54.28
5	DF13 6 Position Connector 25 cm	$0.55	$2.75
3	3DR Cable Pack	$16.99	$50.97
2	3DR Cable Pack	$16.99	$33.98
1	Playstation Eye	$8.95	$8.95
1	3DR IRIS+ Quadcopter	$599.99	$599.99
1	PX4Flow	$149.00	$149.00
1	Asus Xtion Pro Live	$329.99	$329.99
1	Power Module	$25.00	$25.00
1	Odroid XU-4 Board	$83.00	$83.00
	Total		**$2,807.06**

8.5 Risk management

Table 5 shows our Risk Management Table, where we have tracked all of the major risks to our system.

Table 5) Risk Management Table

RIsk #	RIsk	Requiren	Type	Description	Owner	Consequence	Likelihood	Severity	Risk Reduction Plan
6	Cannot get the UAV to dock successfully	3.1.5	Programmatic	Dock design and manufacturing does not have the properties needed to successfully dock, or the quadcopters dynamics or structural properties stop the quad from successfully docking.	Job	4	4	16	Prototype multiple dock designs early and often Maintain existing AR.Drone system as fallback Focus on precision landing ASAP
14	Quadcopter goes wild during run	3.1	Technical	Quadcopter has unexpected motion that can be damaging to the quad or others around it	Job	4	3	12	Create an ABORT button on the computer to take control of quad and land it if it has unsafe motion.
8	Drone is damaged	3.1	Schedule + Cost	Drone is damaged while testing and operation	Cole	3	4	12	Use a net while testing Buy multiple backup parts Save budget for a replacement drone ($600)
2	Extra payload on UAV throws off dynamics	3.1.2	Technical	The extra payload on the quadcopter might change the dynamics of the system and will require modification of controller	Rohan	3	4	12	Test the manual dynamics with weights as soon as possible Test integrated control systems as soon as possible Keep AR.Drone as backup
15	Not enough battery life	3.3	Technical	Quadcopter does not have enough power to successfully meet requirements	Job	2	5	10	Keep extra batteries on hand for hot swap and possibly add extra battery power to payload
13	System error while in flight	3.1	Programmatic	There is some system error that occurs while Quad is in flight, resulting in a loss of control	Cole	2	5	10	Every exception must be handled correctly. Develop E-Stop / abort system.
1	Cannot get accurate localization of system	3.1.2	Technical	We cannot get accurate localization from our sensors	Rohan	3	3	9	Don't rely on accurate global positioning
9	Accurate sensing requires expensive sensor	3.1.3 3.1.4	Technical	The inexpensive sensors we have in the lab or get early cannot detect dock and/or obstacles	Erik	3	3	9	Save money on the budget for expensive sensors
4	Not able to detect obstacles and other objects	3.1.2 3.1.3	Technical	There is not enough processing or payload capacity to be able to have a good enough system to detect obstacles and other objects	Erik	3	3	9	Buy multiple processors and test them for speed and low weight. Use methods of visual recognition which require less processing and memory: like tags.
10	Dock does not rigidly connect with UAV	3.1.5	Technical	Dock that is designed does not rigidly dock	Job	3	3	9	Design a mechanism to attach rigidly to quad externally
12	Software packages do not work on ARM architecture	3.1	Technical	Software packages that we need to do certain tasks do not work on our ISA or our operating system. Reduces effectiveness and creates extra work	Erik	3	3	9	Buy an extra x86 based OBC and test for compatibility with needed packages
7	High and low level software system dependencies	3.1.1 3.1.2 3.1.3 3.1.5	Schedule	There are high level dependencies on high level and low level designs which is hard to work in parallel	Cole	2	4	8	Use the AR.Drone2 to work on the high level software design, and use the Iris+ for low level software design

The major risk that we identified and added after the PDR is shown in Figure 23.

Added Risk Mitigation Strategies

Risk ID:	Risk Title:	Risk Owner:	Date Submitted:	Date Updated:
16	AR.Drone breaks during testing	Cole	11/15/2015	11/25/2015
Description:				
AR.Drone breaks or is damaged during a test run before the FVE				
Consequences:		**Risk Type:**		**Risk Level:**
Team will not be able to complete the FVE challenge		- Schedule - Programmatic		YELLOW 9 / 25
Risk Reduction Plan		**Expected Outcome:**		**Comments**
1. Take out a second AR.Drone from inventory		AR.Drone is available in inventory, so this will be no problem		MITIGATED

Figure 23) Risk 16 Mitigation Strategy

This risk has been tracked and mitigated since the PDR. We were able to get a second AR.Drone from inventory. We are also going to be tracking this risk for the Iris+ during the Spring semester. During the early parts of the Spring Semester, we are going to be building a second Iris+ with the full electrical hardware the same as what we have shown for the FVE. This will ensure that we have mitigated this risk for both the AR.Drone and the Iris+.

The main risk that we have been tracking during the spring semester is shown in Figure 24.

Risk Mitigation Strategies

Risk ID:	Risk Title:	Risk Owner:	Date Submitted:	Date Updated:
6	Cannot get UAV to successfully dock	Job	10/21/2015	12/13/2015
Description:				
Dock design and manufacturing does not have the properties needed to successfully dock, or the quadcopters dynamics or structural properties stop the quad from successfully docking.				
Consequences:		Risk Type:		Risk Level:
The quadcopter will not be able to dock, and a major performance requirement will not be able to be accomplished		- Technical - Programmatic		16
Risk Reduction Plan		Expected Outcome:		Comments
1. Prototype multiple dock designs early and often 2. Focus resources on precision landing **3. March 9th as decision date to switch from Iris+ to AR.Drone**		Majority of work time spent on developing controls and hardware of dock		

Figure 24) Risk 6 Mitigation Strategy

This risk has been given a firmer mitigation strategy. We have set a hard deadline of March 9th to switch development from the Iris+ to the AR.Drone. This March 9th date will allow us enough time to complete our required tasks for the SVE with the AR.Drone. By getting position control working on the AR.Drone during the Fall, we have mitigated the risk that we cannot get enough control over the Iris+ to successfully dock. With this hard deadline, we have fully mitigated this risk.

Figure 25 shows the updated Risk Likelihood-Consequence Table.

Likelihood					
5		15, 13			
4		10, 12	2, 8	6	
3			1, 9, 4 10,12, 16, 17	14	
2					
1					
	1	2	3	4	5
	Consequence				

Figure 25) Risk Likelihood-Consequence Table

9. Conclusions

9.1 Lessons Learned

The major lessons learned during the spring semester by Team C can be summarized below:

- It is difficult to communicate and get everyone on the same page
- One person’s plan may not meet what the others in the group feel it should be
- It is easy to get busy with other things and not deliver what you need to deliver every sprint.

Team C found that it was often difficult to get everyone on the same page during meetings. Oftentimes, one person would say one thing and it would mean something completely different to another person. This would range from technical definitions to emotions to scheduling conflicts. The most detrimental miscommunications happened when technical definitions were not properly communicated. Often, two people would be arguing about a technical question without realizing that they were both on the same side. Other times, a person would get hung up on a simple aspect of a technical question because they were not understanding the definitions another team member was using.

Another form of miscommunication was during the planning stage. This would happen whenever the team would delegate tasks. Oftentimes, one team member's idea of what they are working on should look like. This can often lead to problems whenever the sprint is over. Work that the rest of the team felt should have been done will not get done because of this miscommunication. This leads to a lot of wasted effort on the part of every team member and could be detrimental at the later parts of the semester, whenever things get into crunch time. Luckily this miscommunication did not the outcome of our fall deliverables, but this miscommunication lead us to be less productive each sprint than we would have liked.

One of the major lessons that we learned is that the MRSD program can be very time consuming. Our technical classes require a lot of our time, and that time can often be miscalculated by the team. Often, work during the sprint would either go undone or half done, because other work cropped up for the team. If work was not essential to completing our FVE goals, it would always go undone. This was because we did not have a proper accountability system in place during the semester.

9.2 Key Spring Activities

Due to the lessons the team has learned during the Fall semester, the team has come up with a few key activities to increase productivity during the Spring semester:

- Go back to requirements during a miscommunication
- Communicate and record all decisions made
- Get specific demonstrable deliverables for each sprint for each person
- Show demos at the sprint kick-offs

The first task will help with miscommunication. By explaining what one is talking about in the language of requirements, the team be able to ensure that we are on the same page. This is because we have already talked extensively about requirements. The second task will help ensure that one is held accountable. By recording every decision that we make, it will not be possible to claim ignorance about tasks that have been delegated to each member.

The main spring activity that we have come up with to increase productivity are numbers three and four on the list. By having specific demonstrable deliverables and demos for each team member during the sprint kick-offs, the team will be able to hold each other accountable for the work done during each sprint. It will also allow the team to get on the same page with what everyone is working on. These will act as internal progress reviews that will cover some of the incomplete things that the team is not ready to show in the progress review to Professor Dolan and the TAs.

10. References

[1] http://charliedeets.com/3d-robotics-x8-multicopter/
[2] http://people.csail.mit.edu/kaess/apriltags/
[3] http://wiki.ros.org/tum_ardrone

Appendix A: Detailed WBS

Detailed Fall WBS eye-chart

	Demo Functionality	Task Description	Sprint	Owner	Not Started	In Prog	Compl	Est.	Left	Done	Dependencies				
1	**Open-loop ARDrone Control: Demonstrate takeoff, move, land at push of ROS button**														
1.1	**Low-level open-loop control of drone / takeoff via ROS (AR drone)**			**Erik/Job/R**											
		Set up ROS framework / GIT repo	1	Erik			.	4	0	3	e				
		Write READER node to acquire AR.Drone data	2	Rohan			.	8	0	8		e			
		Design and write MOVER node to issue commands	2	Job			.	8	0	8		e			
		Document and share READER interfaces	2	Rohan			.	1	0	0			e		
		Document and share MOVER interfaces	2	Job			.	1	1				e		
		Integrate and test control script (implement move-while-pressed)	2	Cole/Job			.	8	8				e		
		Add takeoff / land / abort features to MOVER	2	Job			.	4	4				e		
1.2	**Display ROS node graph**			**Erik**											
		Setup launch script for nodes and topics	2				.	2	1						
2	**Fall AR.Drone Position X,Y Movement Demo**														
2.1	**AR.Drone relative odometry working in ROS**		3	**Erik (Cole)**											
		Test if x,y odometry is working when AR.Drone in flight	3				.	1	0	1	a				
		Evaluate / test accuracy and drift of built-in odometry	3				.	2	0	2		a			
		Get it all working / documented					.	4	0	2			a		
2.2	**Closed loop on absolute position control (move to position)**		3	**Cole (Erik)**											
		Research + document robot_localization package	1				.	2	0	2					
		Implement 2d x,y map display in ROS	2				.	4	0	4					
		Test current implementation with simple waypoint	3				.	2		1		a			
		Get closed loop system working / documented	3				.	8	0				a		
2.3	**Integrate AR.Drone demo subsystems**		4	**Erik**											
		Acquire second AR.Drone2	4	Erik			.								
		Thoroughly test actual demo process (with adversrial)	4	Erik			.	4	1	3					
2.4	**Develop GUI for Control**						.	8	0						
3	**Hardware and ROS Setup on Iris+**														
3.1	**Pixhawk -> SBC -> PC ROS setup + sensor display (Demo Focus)**		3	**Rohan (Job)**											
		Understand what we know and don't know	3	Rohan			.	2	0	2					
		Set up SBC (ssh connection, wireless config, ROS install)	3	Job			.	6	0	6	d				
		Set up communication protocols between pixhawk and SBC(Receive)	3	Rohan			.	8	0	4		d			
		Setup communication Pixhawk -> PC (via RF)	3	Rohan			.	8	0	4					
		Set up communication between SBC and PC	3	Job			.								
		Set up downward facing camera	3	Job			.	4	0	3					
		Order PX4, Battery, Charger, Spare props	3	Rohan			.	1	0	1					
		Integrate Everything required for the data display demo	5	Rohan/ Job		.		4	2	2					
3.2	**Iris+ Hardware Setup (Demo)**		3	**Job (Rohan)**											
		Evaluate onboard pixhawk power supply	2				Cole	2	0	1	b				
		Evaluate power protection needed	2				Cole	2	0	5					
		Decide on exact sensor hardware configuration					Job/Rohan	2	2		b				
		Design + simple CAD layout of RGBD / SBC / PX4FLOW and mounting	4	Rohan (Erik)			.	8	0	8		b			
		Finalize Routing for cables	4	Rohan (Erik)		.		2	1	2		b			
		Fabricate mounting hardware	5					8	8				b		
		Assemble parts and route wire	5					2	2					b	
		Order parts for PCB	4				Cole	2	0	1					
		Connect the power distribution wire to pixhawk	5					8	8						
		Oder I2C for Pixhawk					.	2	0	1					
		Order spare UART for PIXHAWK					.	2	0	1					
		Order UART connector for odroid					.	2	0	1					
		Build power cable for Odroid		Erik			.	2	0	2					
		PX4->Odroid UART Interface hardware			.			2	2						
		Shorten Sony camera cable					.	2	0	2					
		find M2 bolts for Sony camera and finish mounting				.		2	2						
		Test UART connection for pixhawk->odroid				.		2	2						
		Interface PX Flow (check power and software interface)				.		2	2						
		Email New FVE		Erik			.	2	0	2					
		Communication: PIXHAWK -> ODROID -> PC			.			2	2						
3.3	**Completed Iris+ Interface**														
		Connect serial 4/5 on Pixhawk to UART on Odroid (Interface)		rohan		.		4	2	2	d				
		Setup high speed communication between SBC and Pixhawk		Rohan		.		8	4	4		d			
		Integrate PX4Flow						16	16						
4	**Prototype of dock: Demonstrate one proof of concept, one actual prototype**														
4.1	**Formulate Dock Requirements and Design Criteria**			**Job**											
		Iterate dock concepts / design (get to 30% done by PDR)	3				Job	4	0	5	c				
4.2	**Dock Internal CODR**		4				Job	4	0	4			c		
4.3	**Manufacturable CAD Model of Dock**		4	Job		.		8	6	2		c			
4.4	**Tested, working physical prototype**														
		Design, fabricate, build, develop rough prototype.	4	Job		.	Job	8	4	4			c		
		Order parts, build dock, etc. (enumerate me)	4	Job		.	Job	8	6	2				c	
		Test manual docking	5		.		Job	2	2						c
5	**Non-Demo Focus Areas: (Dependent on Iris+ hardware setup)**										3.1, 3.2				
5.1	**Iris+ Relative Odometry**														
		Install and configure flow camera module			.			4	4						
		Write node to publish Iris data			.			8	8		f				
		Test SVO Algorithm	3	Rohan		.		8	3	6					
5.4	**Searching for tag on ground with AR.Drone**			**Cole**											
		Get Webcam to be able to recognize tag	4	Cole		.		2	2	2					
		Research high frequency tags for visual servoing	4	Cole	.			8	8						

	Demo Functionality	Task Description	Sprint	Owner	Not Started	In Prog	Comple	Est.	Left	Done
6	**Other Deliverables**									
	Project management tools development		2	Erik			D	8	0	8
	Revise WBS tasks		1	Cole			D	1	0	1
	Risk management plan		1	Cole			D	2	0	3
	Revise risk plan + as	Hold meeting to update + allocate ownership of risk in management plan	2	All			D	4	0	3
	22-Oct Progress Rev	Prepare: a. Progress b. Challenges c. Future work		Erik			D	1	0	4
	23-Oct Website Check 1 A19.1: Website Check 1									
		Post round-robin presenter order to website	1	Cole			D	1	0	1
		Verify all requirements met + delegate/organize tasks	2	Erik			D	1	0	1
		1. System summary - Cole 2. System design - Cleanup Rohan 3. System implementation 4. System performance 5. Project management - Erik 6. Media (videos, pictures, poster) - Job 7. Team page - 8. Documents Finish by Thursday after class	2	Erik, Cole, Job, Rohan			D	4	0	12
	27-Oct Systems presentation:									
		Practice and assemble	2	All			D	4	0	8
		1. Requirements to be verified at the Fall Validation Experiment (FVE)	2	Job			D	1	0	1
		2. Significant changes to the architectures	2	Rohan			D	1	0	1
		3. Work Breakdown Structure (colorized)	2	Erik			D	1	0	1
		4. Critical path on your schedule	2	Erik			D	1	0	1
		5. Top 3 risks and planned mitigation actions	2	Cole			D	1	0	1
	27-Oct Task 12: Power PCB Conceptual Design									
		3. Describe design requirements for power for each subsystem a. Number of connectors and current capacity b. Monitoring input voltage, if you require a manual switch; main overvoltage/reverse voltage protection c. Voltage regulating circuit (efficiency/cost, output voltage, and peak output current) d. FET or other advanced controls e. Reverse Voltage/over voltage protection f. Need to be able to control power? g. Monitor voltage (could just be an LED)	2	Cole, Erik, Job, Rohan			D	4	0	8
		1. List sources of electrical power (battery chemistry, number of cells, voltage range, and continuous/peak current output)					D	2	0	1
		2. Subsystems to be powered (required voltage range, whether regulation is required, continuous and peak current draw)					D	2	0	1
	29-Oct Progress Review 2 in lab + ILR3									
		Prepare Progress Reivew 2: a. Progress b. Challenges c. Future work	2	Cole			D	4	0	2
	1-nov Task 12: Power PCB draft schematic		2	**Cole**				4	0	2
	3-Nov Preliminary Design Review I (midsemester presentations)									
		Outline / record tasks + owners for PDR	2	Cole			D	1	0	2
		Complete PDR tasks, present in class Tuesday C 4:06 - 4:28 p.m.	2				D	16	0	32
		Use Case		Job			D	2	0	0
		System-level requirements		Cole			D	2	0	0
		Functional Architecture		Cole			D	2	0	0
		System/Subsystem Descriptions		Rohan			D	2	0	0
		Current system status		Cole			D	2	0	0
		Cyberphysical Architecture		Rohan			D	2	0	0
		Project Management		Erik			D	4	0	0
		Risk Management		Erik			D	2	0	0
		Project Description		Job			D	2	0	0
	10-Nov A12: Final Power PCB schematic and layout			**Cole**			Cole	2	0	8
	12-Nov Progress Review 3 + ILR4			**Rohan**						
		Prepare: a. Progress b. Challenges c. Future work					.	4	0	4
	19 - Nov Systems Engineering Presentation # 3		4	**Erik/Cole**						
		Project status and action: Discuss tracking of WBS and schedule, discuss what you are doing to address unforeseen technical challenges, delays, and/or lack of progress	4				Erik	4	0	2
		Progress towards the fall validation experiment: Successes, issues, change of requirements, etc.	4				Erik	4	0	2
		Update on risk mitigation (risks mitigated, risks added)	4				Cole	4	0	2
	20-Nov A19.2: Website Check 2							4	0	4
		Update project management + test process					Erik			
		Initial + additional system implementation details (3 - System Implementation)								
		Add PCB Information					Cole			
		Post ILRs					Job			
		Add PDR					Job			
		Fix Problem Description					Job			
		Fix use case					Rohan	2	0	2
	24-Nov A14: Progress review 4 in lab + ILR5									
		Prepare: a. Progress b. Challenges c. Future work					.	4	0	0
	3-Dec Progress Review 5 (includes Power Distribution System PCB) (Fall Validation Experiment)									
		Prepare: a. Progress b. Challenges c. Future work d. PCB					.	4	0	0
	10-Dec Progress Review 6 (Fall Validation Experiment encore) (with PDS PCB)						.	8	0	0
	14-Dec Critical Design Review Presentation I A17: Critical Design Review							16	16	
		Tweak Use Case pictures and info (Didn't do....)		Job			.			
		WBS for spring + schedule status		Erik			.			
		Spring Milestones		Cole			.			
		Spring Validation Experiment		Cole			.			
		Fall Video		Job			.			
		Spring Risks		Cole			.			
		Update Current System Status and how it performed against the FVE (performance evaluation)		Rohan/Job			.			
		Functional Archetecture, Bigger font (Change it back to old one)		Rohan			.			
		Add performance metrics to all of requirements		Cole			.			
		Conclusion		Cole			.			
	18-Dec Website Check 3 A19.3: Website Check 3						.	4	4	