Column Robotics: Team C

Conceptual Design Review

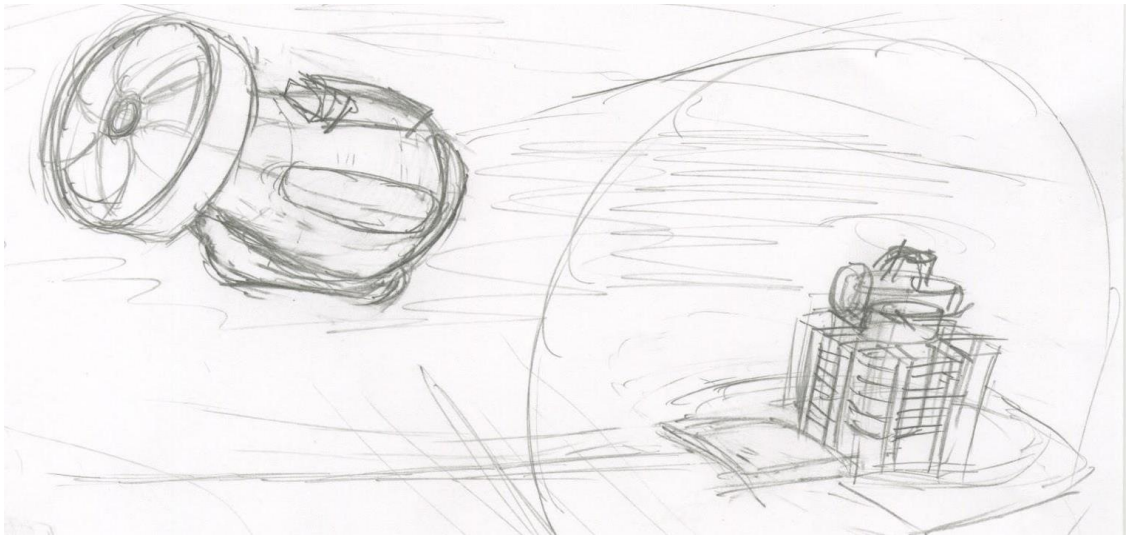
Job Bedford

Cole Gulino

Erik Sjoberg

Rohan Thakker

Table of Contents



1. Project Description

1.1 Autonomous Exploration and Docking

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.

2. 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.

3. System-Level Requirements

3.1. Functional Requirement

- Locate Oil/Gas Wellhead
  - Min scope: fully visibility, known heading pipe leads to it
  - Max scope: low visibility, heading unknown
- Autonomously Maneuver to Wellhead
  - Avoid obstacles
- Positively ID as correct wellhead
  - Min scope: recognize ID tag/color on wellhead
  - Max scope: Scan structure of wellhead, compare to CAD model
  - Align with wellhead dock
- Rigidly dock to wellhead
  - Min scope: Rigidly attached in 6 DOF
  - Max scope: Rigidly attached in 6 DOF with electrical connection

3.2. Non-Functional Requirement

- Reduce cost of wellhead intervention
- Ease to operate
- Simulate underwater environment

3.3. Performance Requirements

- Sequence completed within 4 hours
- Locate specified oil/gas wellhead
  - Within area: 200 m^2
  - Get within visual range
  - Positive ID of specific wellhead (90% confidence)
- Autonomously maneuver to desired wellhead
  - 90% success rate
  - Aligns with dock
  - Within a 1 meter radius
- Docks with wellhead
  - No damage to infrastructure or robot
  - >50% docking success rate per attempt
- Provide feedback
  - Current state at 0.1 Hz
- Operable by <=2 People

3.4. Non-Functional Requirement

Provide status feedback to user
Sense environment
Avoid obstacles

4. Functional Architecture

The functional architecture (shown in figure 1) is designed to be generic such that it is consistent irrespective of application to underwater robot or aerial analogue. Using the general location and wellhead description (i.e. input from the user) the robot locates and identifies the wellhead. It does this by sensing the environment, planning and then executing the next move. If the wellhead is successfully identified then it moves to the pre-docking position, else it repeats the process. The robot uses a similar process to reach the docking position, after which it senses the environment, plans a path to the pre-docking position and executes it. If the robot has reached the pre-docking position, then it begins docking, else it repeats move to pre-docking position. Then the robot tries to dock to the docking station until it succeeds. On completion the robot takes a photograph of the wellhead and sends it to the base. Throughout the process the robot gives a heartbeat status update along with updates at every change in state.

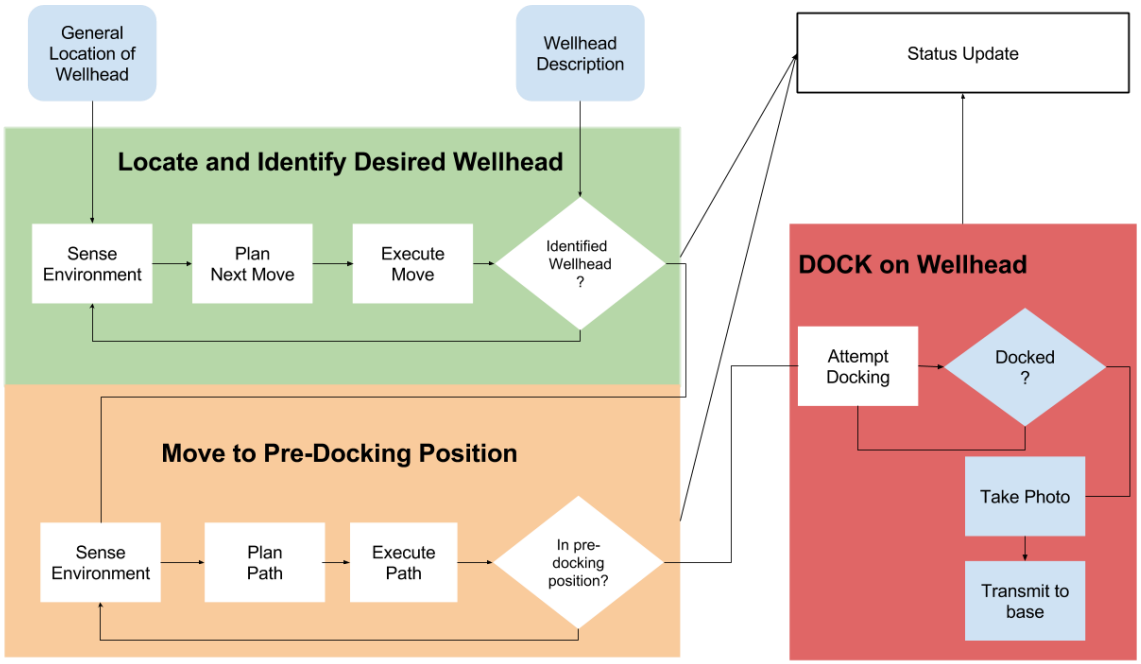


Figure 1) Functional Architecture

5. System-level Trade Studies

5.1 Quadcopter Platform

Table 1) Quadcopter Platform Trade Study

Parameter Name	Weight (1,3,,9)	Parrot AR Drone 2	3DR Iris+	3DR X8+
Flight Time/Payload	9	2	3	5
Existing Sensor Package	3	5	2	2
API Quality/Documentation	9	4	4	4
Wingspan	1	4	4	3
Cost	1	5	3	1
Hardware Expansibility (max processing)	9	1	3	5
Community	3	5	4	4
Hardware Expansibility (sensing options)	3	1	3	5
	Total:	105	124	**163**

The three most important factors in choosing our quadcopter platform were hardware expansibility for max processing power, flight time/payload capacity, and quality documentation and API. The three quadcopter platforms we analyzed were the Parrot AR Drone 2.0, the 3DR Iris+, and the 3DR X8+.

The quadcopter platform is integral to the success of our project. A readymade platform that contains all of the essential hardware will allow us to focus on the higher level algorithms that we want to implement.

In order to have a structured search, the UAV will need to be able to run on battery for an extended period of time while it completes the task at hand. This will require a quadcopter that has enough basic flight time. We also will need to add extra sensors and processors to run our algorithms on board and simulate an underwater environment. This will require a platform that is hardware expansible. These two features are intertwined in the payload parameter. We need to be able to carry an extended payload of electronics, sensors, and the docking mating device. This requires that we have a quadcopter that can carry this load.

Finally, the API documentation and quality is incredibly important. In order to have the time to implement our higher level algorithms, we need to have an API for the system that reduces the complexity of aspects of the project that are not our focus.

In looking at our top three choices, the API quality is top notch on all three platforms. 3DR and Parrot are industry leaders because of their quality API system. Where the 3DR X8+ distinguishes itself from the pack is in the flight time/payload and hardware expansibility

parameters. The 3DR X8+ is vastly superior to the other two platforms and as such wins out in our quadcopter platform system trade study.

5.2 Docking Mechanism

Table 2) Docking Mechanism Trade Study

Parameter Name	Weight (1,3,9)	4x Funnel Dock	Sliding Mesh	Decapitated Pyramid	C-leg on Bars
Docking Approach Slop	9	3	5	2	2
Post-docking tolerance	3	4	2	3	4
Mechanical Robustness (of dock)	3	4	3	3	3
Cost	1	5	2	3	3
Size of mating device on docking vehicle	9	4	4	3	2
Weight of mating device on docking vehicle	9	4	4	3	4
size/weight of device	3	3	2	3	3
Complexity (Meche & Electrical)	3	5	2	4	2
	Total:	**152**	146	114	111

We brainstormed initial ideas (shown in figures 2, 3, 4, and 5 to come up with four basic mechanical structures for our docking mechanism. Every design we chose is passive besides the sliding mesh. In analyzing our weights, we came up with three aspects that are above the rest in importance.

We felt that the docking approach slop was very important in order to make the precision needed to dock successfully much easier to obtain. The size and weight of the mating device on the docking vehicle must be kept small in order to meet the physical and payloads limitations of the chosen UAV.

Other important considerations were robustness of dock to reduce breakage and complexity in order to reduce scope on our project.

5.2.1 4x Funnel Dock

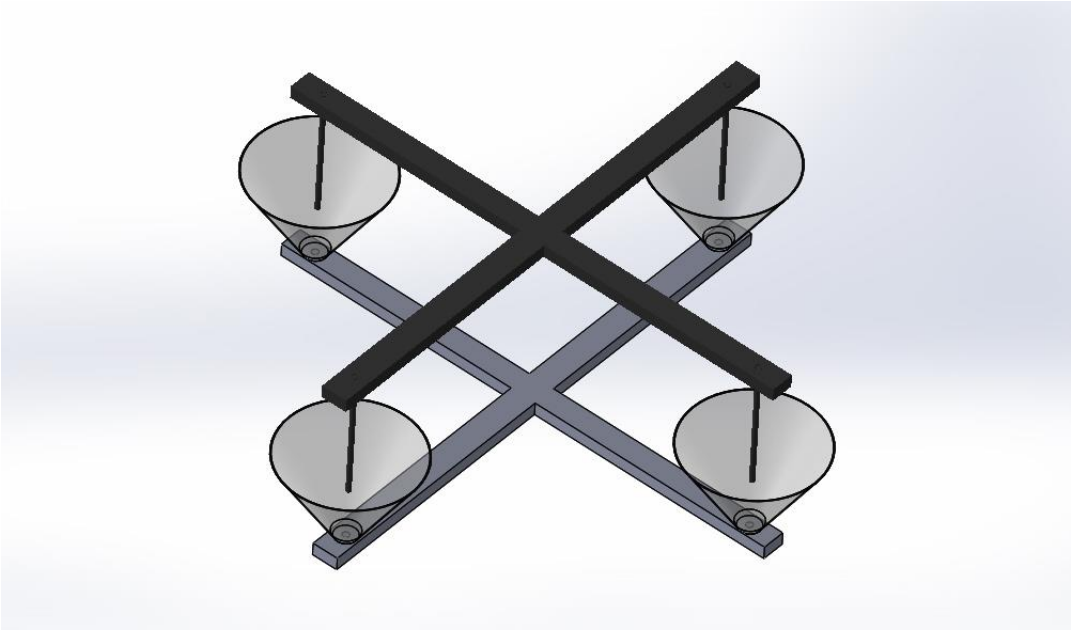


Figure 2) Funnel Dock Mechanism Initial CAD Model

5.2.2 C-leg on Bar

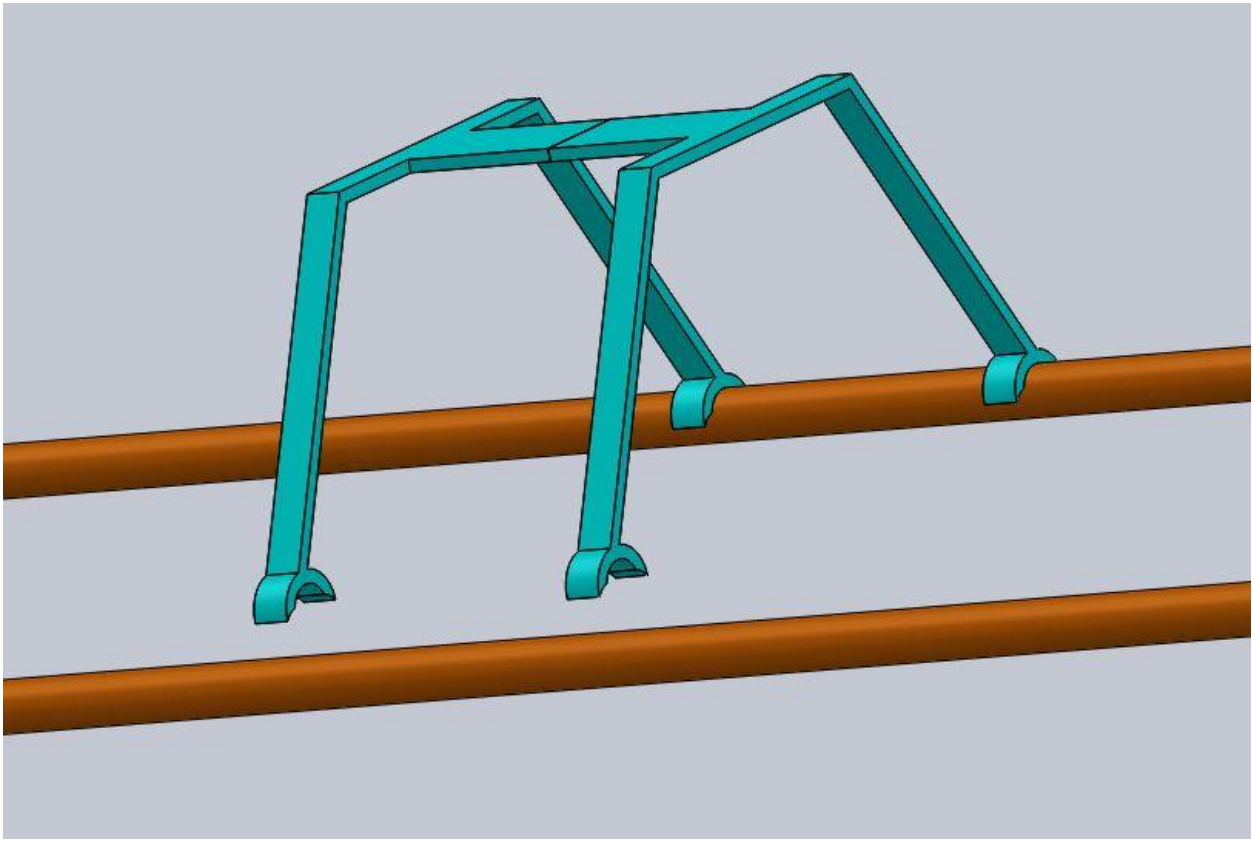


Figure 3) C-Leg on Bar Dock Mechanism Initial CAD Model

5.2.3 Decapitated Pyramid

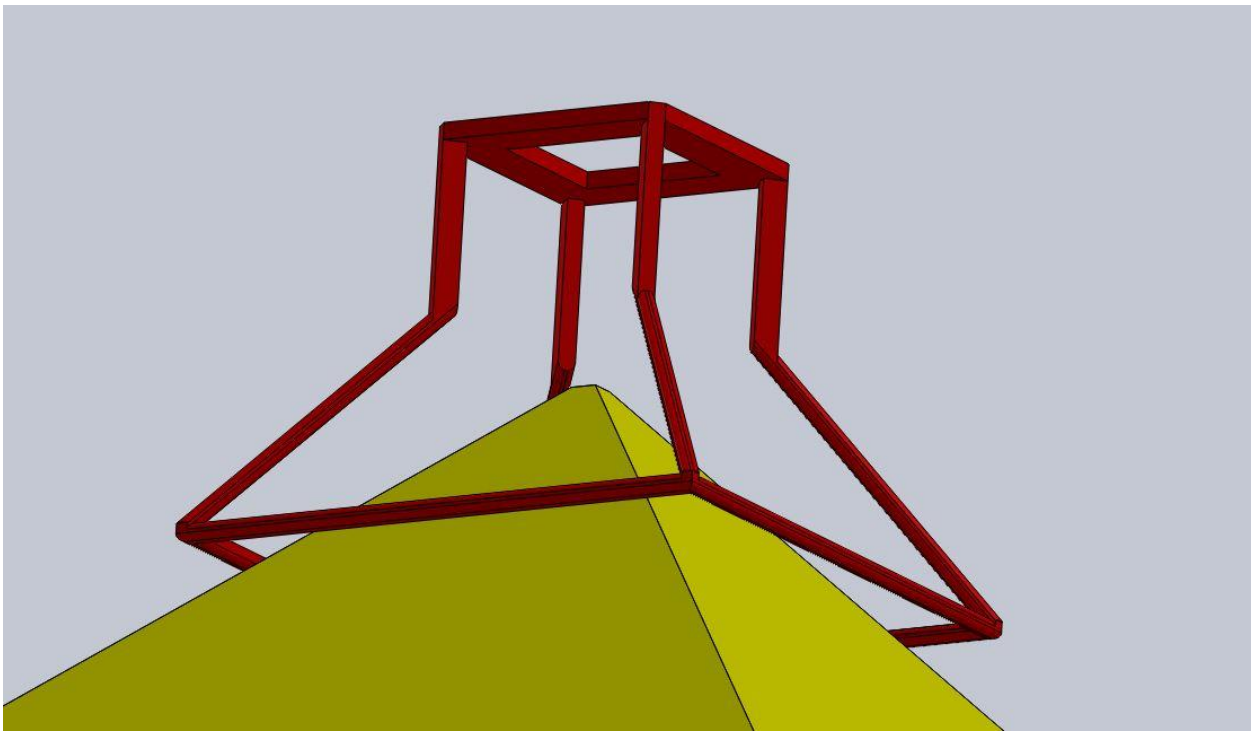


Figure 4) Decapitated Pyramid Docking Mechanism Initial CAD Model

5.2.4 Mesh Dock

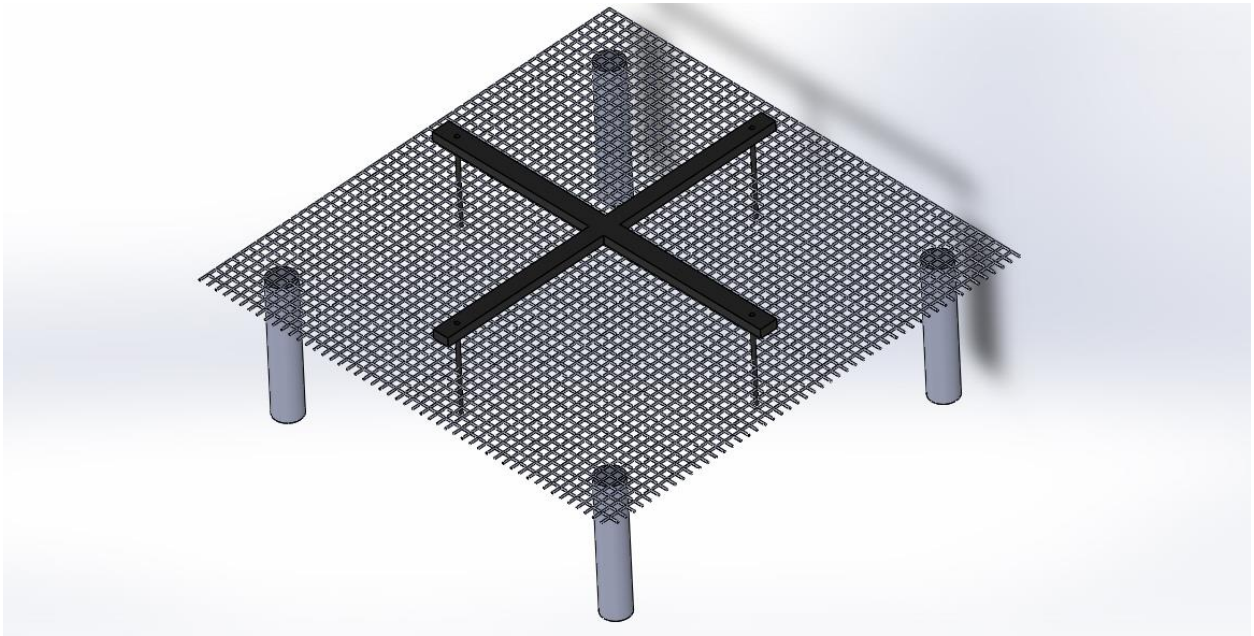


Figure 5) Mesh Docking Mechanism Initial CAD Drawing

6. Cyberphysical Architecture

The cyberphysical architecture (shown in figure 6) has been broken down into five main parts: electro-mechanical, sensors, software, user interface, and infrastructure. We have organized our cyberphysical architecture based on how the systems are physically organized.

The electro mechanical system comprises the mechanical and electrical hardware which is mostly on the quadcopter platform. Some things like the single-board computer and wireless communication module will be added.

The sensors connect to two main apparatuses. The camera, which will be doing high level perception, is connected to the single board computer. The pose and depth/height sensors are lower level and connect to the microcontroller which will be doing low level software.

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.

Finally, the infrastructure is physically separated from the UAV. It is the docking station and wellhead identifiers that will be build, but connect to the UAV through the docking mating device connected to it.

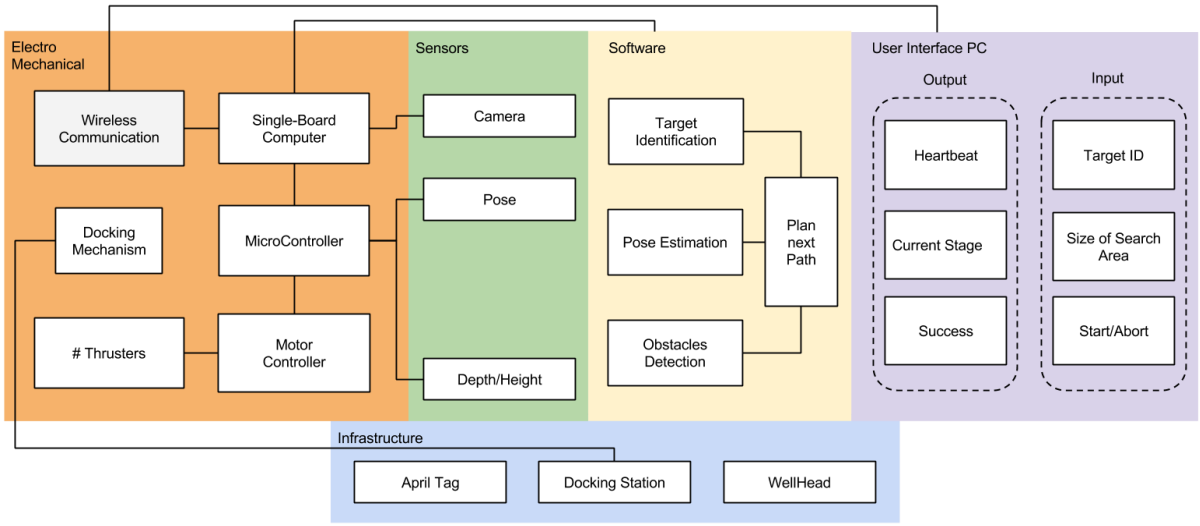


Figure 6) Cyberphysical Architecture

7. Subsystem Descriptions

Based on the functional architecture the systems can be divided into the following 3 major sub-systems:

7.1 Locate and Identify Desired Wellhead

This subsystem gets the approximate location and description (identifier) of the Wellhead to estimate its state. Using this it computes the required action to search for the wellhead and then executes the action. On locating the wellhead, the sub-system will assert that it has found the desired Wellhead by comparing the features of the located wellhead to the description of the wellhead that was received as an input. If the located wellhead is identified as the desired wellhead than the system will proceed to align to wellhead, else repeat the process.

The exploration subsystem refers to the system of algorithms that leads the robot: (1) from its starting position to within visual range of desired wellhead and (2) from within visual range of the wellhead to its pre-docking position. The function of the exploration subsystem is to define the ways in which the UAV will search for and get in range of the desired wellhead and docking system. It also functions to get the UAV from the general vicinity of the wellhead to its optimal pre-docking position.

When getting the UAV from its starting position to within visual range of the wellhead, the algorithms can take advantage of the search area input given by the user interface. The search area will allow the UAV to plan an optimal path so that every inch of the search area has come within the visual range of the UAV. This algorithm will need to be modified when low lighting conditions are tested. An example of such optimized search path in the circle (shown in figure 7):

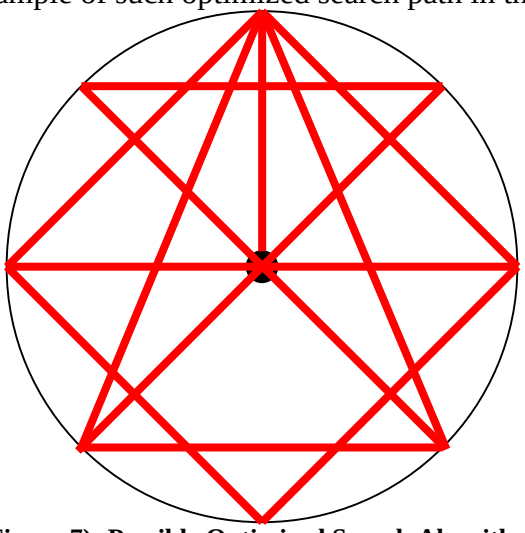


Figure 7): Possible Optimized Search Algorithm

These kind of algorithms can be quite complicated with parameters constantly needing to be changed for the type of platform, visual sensor suit, and search area. There can also be a few areas of blackout. This can be adjusted for in the algorithm. There is always tradeoffs which could cause undesired functionality in certain situations. This algorithm also requires good information about the pose of the UAV.

An alternative method is to use a lawn mower search method (shown in figure 8):

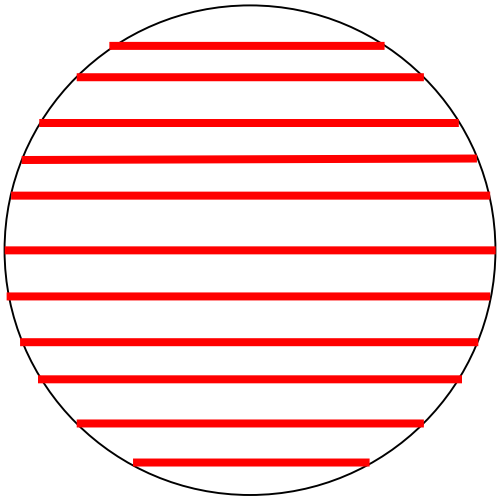


Figure 8): Lawnmower method

This lawnmower method is much simpler and avoids blind spots. It, however, is not as quick in its search path, especially if the wellhead is at the top or the bottom of the search area.

7.2 Align to Wellhead

Once the system has located and identified the wellhead, this sub-system will move the robot to pre-docking position. Since the robot can locate the wellhead from any side, it needs to align itself in the direction of the docking station. It starts by sensing the environment and estimating the robot's pose w.r.t. to the docking station. Due to budget constraints, we cannot afford to have high resolution inertial measurement units used by commercial underwater robots, hence we plan to simulate equivalent performance using visual odometry. To localize the UAV with respect to the docking mechanism and wellhead, we plan to design special features on the docking mechanism which can be used by the vision systems for pose estimation.

Once the UAV is localized with the wellhead, it will use its prior knowledge of the CAD model of the docking mechanism to know which way to approach in docking. Then it generates a path from its current position to pre-docking position and executes it. If the robot successfully reaches the docking position then it moves to the next step of docking to the wellhead, else it repeats the process.

These methods are incredibly computationally intensive and could be out of the scope of our project. The alternative is to use some tagging system (like APRIL tags) to localize oneself to the docking mechanism for docking.

7.3 Dock on Wellhead

Upon reaching the docking position the robot, this subsystem attempts to dock until it succeeds. Once it's successfully docked to the docking station it captures an image of the wellhead and transmits it to the base.

Once the UAV has correctly identified Dock and has properly orientated itself for the landing approach, the UAV will then commence its final task, and dock with the well head. This subsystem, will utilize the rover's on-board vision sensors to determine the dock features for position and re-correction during landing. This subsystem will be a two part mechanism: (1)

features on the robot's landing gear that interlock/attach to (2) specialized on-dock mechanism. The robot will be outfitted with 3D print or fabricated landing gear specially design to (1) create a stable cushion for landing and (2) mesh with docking interface. The docking interface is designed to provide a sizable area for docking and actuates to provide a rigid connection that constrains the robot in 6 DOFs.

A desirable functional requirement is to have the dock create an electrical connection with the dock in order to provide signal of a successful dock, or recharge the robot. The docking station is primarily a mechanical interaction. The docking station must be able to handle variability in the robot's approach and misalignment. It would also be advantageous to funnel the robot's movement to a constraint footprint.

Passive slopes will funnel the UAV's landing gear into a correct foothold with little disturbance to the UAV's flight path. Once UAV's landing gear are in the desired foothold location, a latching mechanism will activate, securing the UAV to the platform. At this time an electrical connection will be made in tandem with the latch to recharge or add a communication pathway.

7.4 User Interface

The user interface subsystem's function is to serve as the link between the on-board computer of the UAV and the user. The UI is responsible giving the UAV critical information about its search area and path. It must also be responsible for alerting the UAV when to start its mission and when to abort if there is danger. The UI must also be responsible for handling the outputs of the UAV. It must transmit data to the user about the UAV's current status, stage in execution, and other collected data at the Wellhead. In simulating an underwater environment, the UAV will only be sending data at a rate of around 0.1 Hz.

The user interface will be responsible for three major inputs to the UAV system: (1) Target ID, (2) Size of Search Area, and (3) Start/Abort. The UI must provide the UAV with some general information about its search.

The first is the Target ID. The UI must provide the UAV with some sort of identifying information about the desired object it must search for. We are aiming for the user to be able to send a CAD drawing of the docking station to the UAV as a way of identifying it. The drawback to this method is that the possibly large CAD files could be complicated to reason and may not provide enough information in an unstructured search. An alternative is to provide a color, tag identifier, or some other easily identifiable marker.

The UI must also provide the UAV with a general search area. This will allow the UAV to have bounds on its search and optimize its search path. One approach for this information is to set some radius around the UAV that it must search. The problem with this is that poor localization will render this information useless. An alternative is to use a physical boundary. This approach is not as versatile.

Finally, the user must be able to send a start/abort signal to the UAV in order to ensure safe operation.

The user interface must also be responsible for three major outputs from the UAV system: (1) Heartbeat, (2) Current Stage, and (3) Success. This information gives the user valuable information about the system.

The heartbeat is a simple status update at 0.1Hz in order to ensure the user that the UAV is operating normally. This update would be especially critical for an undersea environment where the autonomous system may not be seen by the user.

The current stage status update gives the user the stage that the UAV is in: locating and identifying desired wellhead, moving to pre-docking position, and docking on wellhead. This information will be provided to the user as the UAV changes to each of these states. An alternative is to provide this information with the heartbeat.

The success status update lets the user know that the UAV has successfully docked and is in position. We plan to have this information come in the form of an image of the desired wellhead infrastructure. This will again be valuable information for the undersea analogue to our UAV. In an undersea environment, an image of a deep wellhead would be invaluable information. The problem is that, by running on the low bandwidth underwater model, an image may not be feasible to send in an undersea environment. An alternative is to send a simple success signal with the heartbeat.

8. Project Management

8.1 Responsibilities

Rohan: **Visual Odometry-SLAM, Controls, Design Docking station,** ARDrone setup, Software infrastructure / dev processes

Erik: **visual obstacle detecting / SLAM, Planning Architecture, Sensor integration / evaluation**, Sensor fusion, Software infrastructure / dev processes, Interfacing high/low-level processing, Control for docking, Build / design wellhead,

Job: **Software Quad exploration (paths), Docking station design**, Object recognition, Software Communication, High level controls, sensor/hardware evaluation,

Cole: **Object recognition, Path planning**, embedded software, software development, controls, sensor integration / evaluation, communication and UI, docking electronics

8.2 Work Breakdown Structure

Table 3) Work Breakdown Structure

Work Description	Customer Need	Work	% Complete
Epic: Fully integrated system	**High**	**382 hrs**	**0%**
Epic: Search for and Identify Wellhead	**High**	**207 hrs**	**0%**
Display drone heartbeat signal	Medium	5 hrs	0%
Safe takeoff and land (abort)	Medium	15 hrs	0%
Detect + avoid walls / stationary obstacles	Medium	25 hrs	0%
Detect obstacles	Medium	15 hrs	0%
Plan path around obstacles	Medium	10 hrs	0%
Control (maintain) pose automatically	High	65 hrs	0%
Change pose (x,y, theta, height)	High	15 hrs	0%
Track changes in pose	High	30 hrs	0%

Detect distance from floor	High	10 hrs	0%
Detect rotation/translation	High	20 hrs	0%
Track changes in pose w/ low visibility	Low	20 hrs	0%
Estimate current position in environment	Medium	20 hrs	0%
Detect environment features	Medium	5 hrs	0%
Estimate current position w/ low visibility	Low	20 hrs	0%
Plan next movement	Medium	30 hrs	0%
Identify wellhead	High	22 hrs	0%
Detect wellhead identifier	High	5 hrs	0%
Detect raw wellhead	Low	10 hrs	0%
task: build wellhead model	Low	10 hrs	0%
Send wellhead identifier to drone	Medium	4 hrs	0%
Receive + store wellhead identifier	Medium	2 hrs	0%
Compare detected identifier to stored id	Medium	1 hr	0%
Epic: Autonomously maneuver to pre-dock pos.	**High**	**42 hrs**	**0%**
Status update to user for "At Wellhead"	Medium	2 hrs	0%
Avoid contact with wellhead	Medium	20 hrs	0%
Detect wellhead structure	Medium	10 hrs	0%
Plan path around wellhead to find dock	Low	10 hrs	0%
Detect dock itself	Medium	10 hrs	0%
Steadily hold position above dock	Medium	5 hrs	0%
Orient appropriately for docking	Medium	5 hrs	0%
Epic: Dock at wellhead	**High**	**133 hrs**	**0%**
Status update to user for "Docking"	Medium	2 hrs	0%
Manual docking at wellhead (prototype)	Medium	30 hrs	0%
Automated docking at wellhead	High	60 hrs	0%
Controlled approach to dock	High	30 hrs	0%
Detect abnormal/failed docking attempt	Medium	10 hrs	0%
Return to pre-dock position	Low	20 hrs	0%

Detect successful docking	Medium	5 hrs	0%
Rigidly lock to dock	Medium	20 hrs	0%
Make electrical connection with dock	Low	10 hrs	0%
Take image (post-docking)	Low	4 hrs	0%
Transmit image of dock	Low	2 hrs	0%

8.3 Risk management

8.3.1 Design / Technical Risk

a. Give enough weight/processing capacity to enable Kinect-like sensor if needed
    i. Risk: Not able to detect obstacles
b. Design fallback global positioning
    i. Risk: Visual Odometry does not work
c. Props and Net (also around dock)
    i. Risk: Can't get enough precision in docking descent
    ii. Risk: Damage to drone
d. Design dock for low landing precision
    i. Risk: Can't get enough precision in docking descent
e. Prototype multiple designs for docking early
    i. Risk: Can't get the quad to dock

8.3.2 Scheduling / Budget Risk

f. Testing fundamental capabilities with AR.Drone2
g. Purchase parts kit for entire second drone
h. Maintain Multiple charged batteries
i. Complete MVP by December

8.4 Budget

Table 4) Preliminary Budget

Part Item		Description		Cost
Quad 3DR X8+ assembly		[2]		$1350.00
Quad 3DR X8+ Backup Parts		[3]		$550.00
Single Board Computer		Tronsmart Ara X5		$200.00
Depth Sensor		e.g. RealSense, XtionPro		$100.00 - $300.00
Camera				$100.00
Extra Batteries				$149.99
Battery Charger				
Optical flow kit				$149.99
Total				$2799.98

8.5 Schedule

8.5.1 Schedule: Fall Semester

Figure 9) Fall Semester Schedule

8.5.2 Schedule Continued: Spring Semester

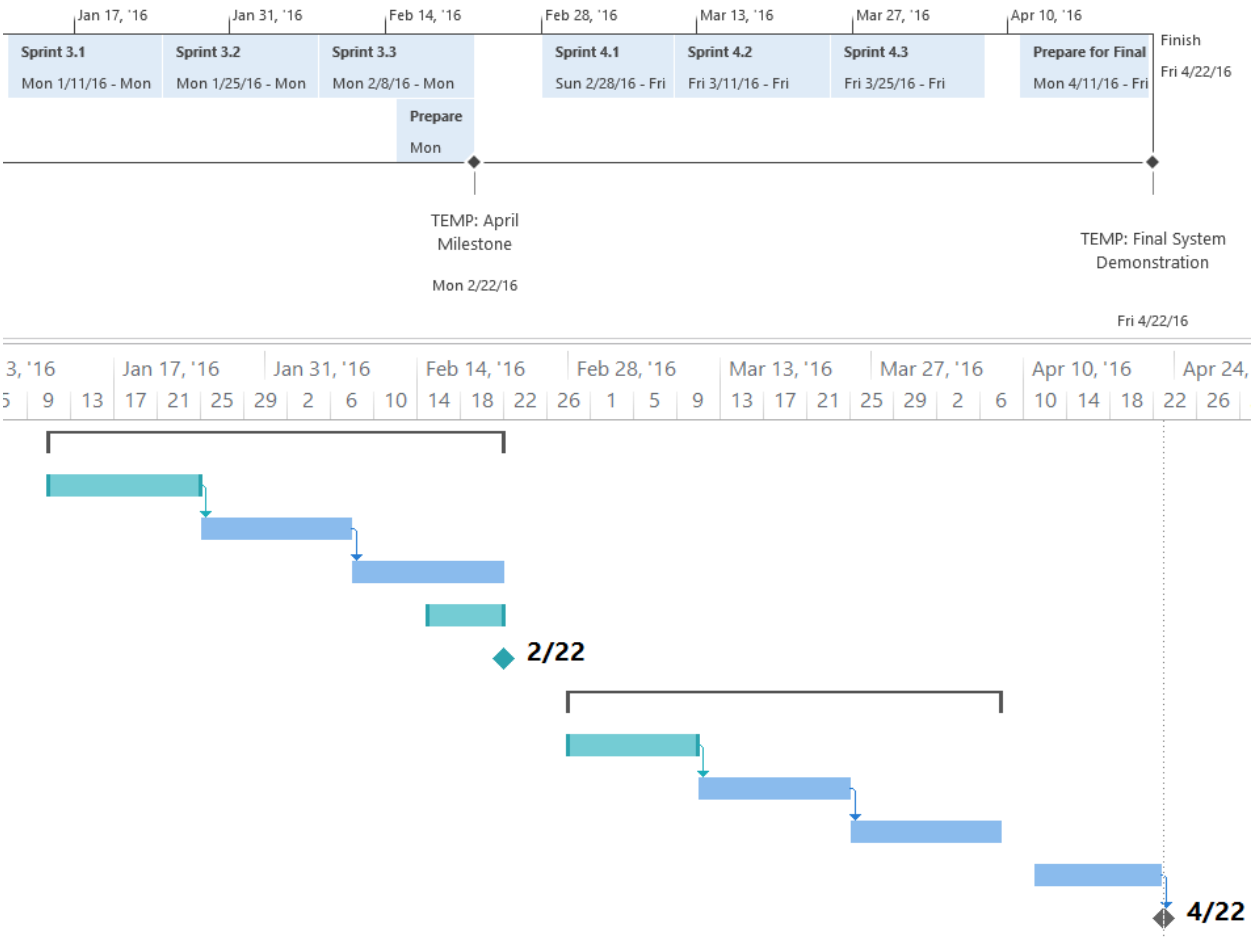


Figure 10) Spring Semester Schedule

References

[1] http://www.pddnet.com/news/2013/11/photos-day-constructing-subsea-wellhead-trees
[2] https://store.3drobotics.com/products/x8-plus
[3] https://store.3drobotics.com/products/diy-quad-kit