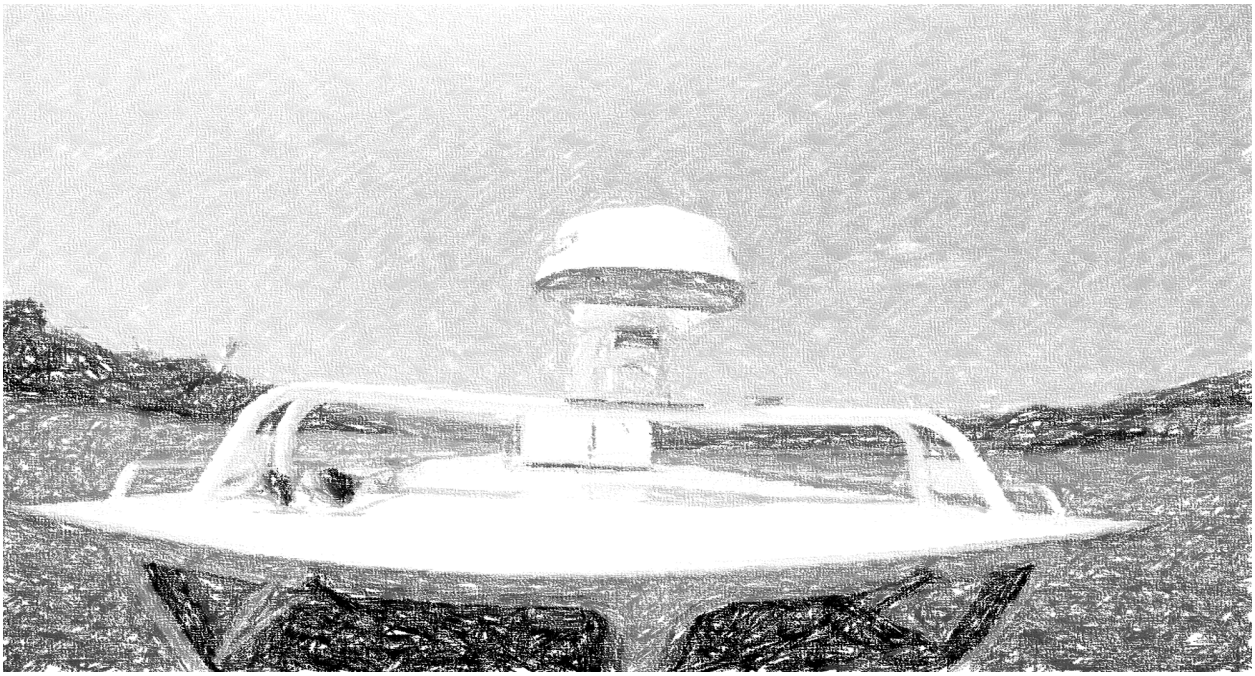
Concept Design review

Autonomous Water Taxi



Team Members (Team B):
Tushar Chugh
Shiyu Dong
Bikramjot Hanzra
Tae-Hyung Kim
William Seto

Mentors:
John Dolan
Dimitrios Apostolopoulos

Sponsor:
Jeremy Searock

Carnegie Mellon University
The Robotics Institute

Contents

List of Figures

List of Tables

1. Project Description

Figure 1.1: 27 ft SeaHawk Aluminum Welded Boat

The National Robotics Engineering Center, the applied research lab of the Robotics Institute, has recently acquired a 27 ft SeaHawk aluminum welded boat (Figure 1.1). The boat has been converted to a test bed for maritime autonomy research, with drive by wire controls and a defined interface. The boat also has a cabin and heating/air-conditioning to facilitate year-round use for developers onboard. It is also outfitted with autonomy sensors and a positioning system.

Currently, the boat has an on-board computer that has implemented a low-level controller. This includes closed-loop heading control, open-loop throttle control, and the ability to specify waypoints. The vessel is ready for further autonomy development. In this project, we will develop algorithms and tools to enable autonomous navigation of the boat to defined destinations along the river.

2. Use Case

James would like to spend a nice day with his family, beginning with some shopping at Walmart, then followed by an exciting game at PNC Park, and finally a nice dinner at the Cheesecake Factory in Southside Works. Unfortunately, as these locations are located in completely opposite directions, he has been unable to make this perfect day materialize.

However, one day James hears about a new company called Pittsburgh Auto-Pirates. It is a new service that allows anyone access to their own private water taxi ride. All James needs to do is install the smartphone app, head over to the closest dock, and request a water taxi. After boarding, he chooses one of the predefined destinations and the boat will autonomously navigate to the specified location. Now James can enjoy a peaceful and intimate ride with his wife, along with the beautiful views on the Monongahela River, all while getting from Downtown to Southside Works in a matter of 15 minutes.

Most importantly, the water taxi system will link many of Pittsburghs attractions and important locations that are located on the riverfronts. If one is looking for entertainment, Heinz Field, PNC Park, and the Rivers Casino are all conveniently accessible by boat. If one needs to go shopping, Southside Works, Walmart, and Costco are all options as well. Finally, the ability to utilize the water taxi as a mode of commuting will be a boon for everyone in the city. Rather than being cramped in a stuffy subway, James will be able to boast to his friends in New York about having the river as part of his daily commute. Citizens and the city of Pittsburgh will both benefit greatly from the water taxi operation.

3. System-level Requirements

3.1 Functional Requirements

Autonomy

Autonomy of the boat shall -

FR.1 Provide control for throttling and steering

- Interface with low level controls which are already retrofitted in the boat

FR.2 Navigate boat to destination given no obstacles

- Navigate boat from one waypoint to another
- Map the path from a given map and the data of the Radar
- Navigation through multiple waypoints using graph search algorithms

FR.3 Detect and avoid Static obstacles

- Detect static obstacles using the data from the Radar
- Generate a new path (that avoids obstacles) and navigates through it

FR.4 Drive through bridges

- Radar would be detecting bridges but we need to go under them

Stretch

FR.5 Detect Dynamic Obstacles

FR.6 Estimate path of dynamic obstacles

Log Playback

Log playback feature shall -

FR.7 Record data from sensors

- Store the data from RADAR and IMU in files to be used for testing

FR.8 Play-back data to boat/simulator

- Replay the data to the simulator for testing

FR.9 Find and Integrate simulator

- Integrate simulator like Gazebo

Operator Control Unit

Operator Control Unit (OCU) shall -

FR.10 Provide interface to enter destination

- Develop GUI to provide an interface to user to give destination as the input

FR.11 Display current location on map (display)

- GUI should also display our current location on a map (Google Map)

3.2 Non-Functional Requirements

NFR.1 Follow right alignment on path

- Maintain the alignment of the boat to right side of the river

NFR.2 Not confuse other manual boats

- While avoiding dynamic obstacles our boat should not confuse other boat drivers

NFR.3 Keep 25-30 trials for testing (Budget constraint)

- We have a limited budget with us which puts a constraint of 25-30 trains on river

NFR.4 Maintain Code Quality

- Our code will be reused by other teams in future so it is critical for us to maintain code quality.

NFR.5 Video of working boat

- Creat video of working autonomous boat which can be used to pitch potential sponsors for developing this technology further.

3.3 Performance Requirements

PR.1 Maximum Speed: 15 MPH

- This is the maximum speed before which the boat uplifts from the water

PR.2 Testing distance: 2 miles in 15 minutes (5 static obstacles)

- During the test run we will have to complete 2 miles in 15 minutes

PR.3 Obstacle size: Proportional to the resolution of sensors

- The minimum size of the obstacle detected would be proportional to the resolution of Radar sensor. For now, we would be taking this value as $30cmx30cmx1mt$(height)

PR.4 Types of obstacles: Shore, buoys, other boats

- We have to detect shores, buoys and other boats but as docking is not a part of our requirements we will not have algorithms to navigate obstacles near docks

---

4. Functional Architecture

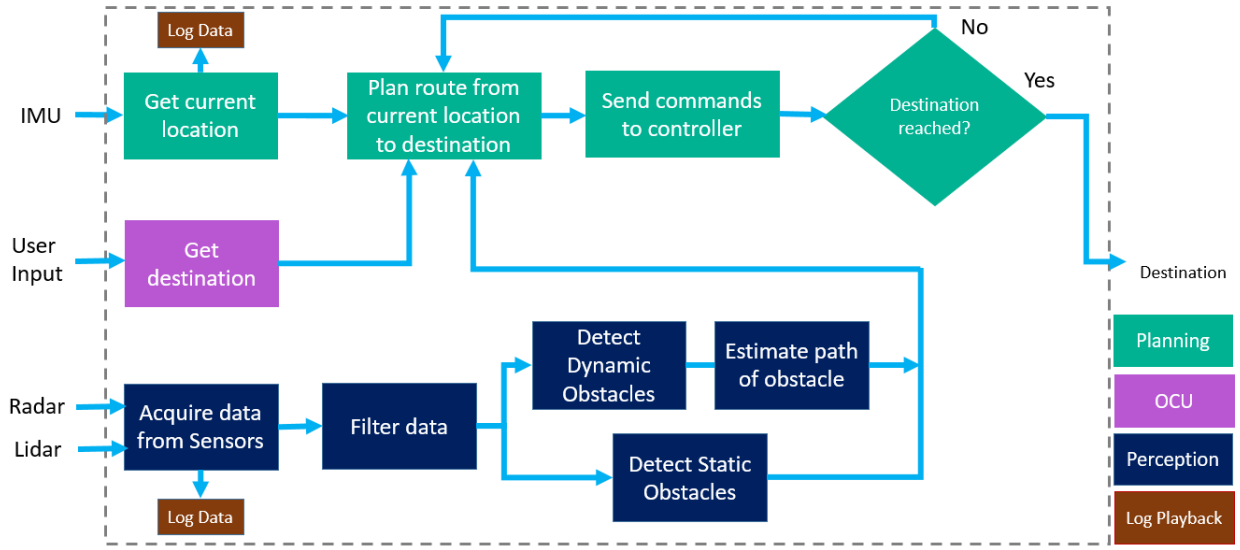


Figure 4.1: Functional Architecture

-

Figure 4.1 describes the functional architecture of the Autonomous Water Taxi. The functional architecture can be viewed as the building blocks of elements spanning across perception, planning, OCU and Log Playback. The desired output of the system is that the autonomous boat reaches the final destination by avoiding obstacles. The input of the system is user input, which specifies the way points and destination, and the data acquired by sensors, which includes IMU input that gives data of current position(GPS coordinates, heading), and Radar and Lidar which detect the objects.

First part of the functional architecture is obstacle detection based on Radar and Lidar. The system acquires data from Radar and Lidar sensor, which we can log and play back that data for further analysis purposes. Since it will consist of a lot of noise in the data got from the sensors, especially for the radar, the next step will be filtering the data to get the useful information that indicates the size and location of the obstacles. For the obstacles, we need to divide the obstacles into two categories, static obstacles and dynamic obstacles and then design different strategies for them, because for dynamic obstacles we need to estimate path of it and then decide to follow its path or just avoid it. And we will get obstacle avoidance algorithm for this part finally.

The second part of the system is Log Playback. Log Playback allows us to log the data from the sensors(RADAR, IMU) and then analyze and implement algorithms based on the data. It also provides required information for the OCU module, and finally shows in the user interface.

The third part is Operator Control Unit (OCU), which allows user to specify the geometry location, usually the longitude and latitude, of waypoints and destination. The idea of the water taxi is that we implement an autonomous driving boat and users are able to use the mobile app to order a taxi service, and the water taxi will pick up passengers at each point and plan a path.

The final part is path planning which gathers data from IMU sensor to get the current position and location of the autonomous boat. The IMU sensor has an embedded high-accuracy GPS and digital compass to get an accurate location and gives to the path planning module as the input. The path planning function gets the input of current location, destination and the information of the dynamic and static obstacles. Then it plans a path for the autonomous boat to reach each way point without hitting any obstacles and sends accordingly give commands to the low-level controller. Then the following function will estimate the current location and check if the autonomous boat reaches each way point and finally arrives the final destination successfully. If the autonomous boat has not reached the destination, then it will go back to the path planning function and design a new strategy for path planning with updated input from sensors.

---

5. Cyber-physical Architecture

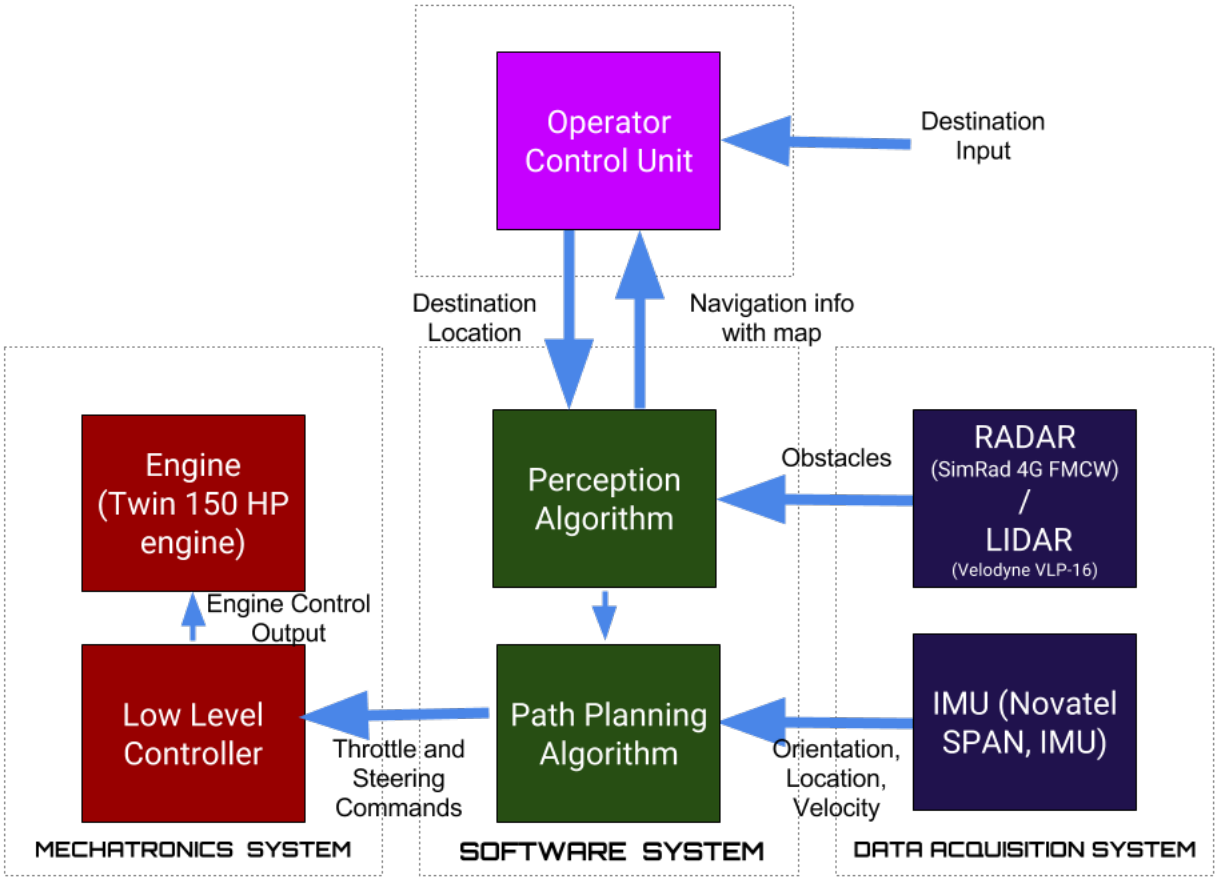


Figure 5.1: Cyber-physical Architecture

As shown in the Figure 5.1, the cyber physical architecture is divided into 3 systems namely -

- Data Acquisition System
- Software System
- Mechatronics System

5.1 Data Acquisition System

As the name suggests, the primary function of the data acquisition is to acquire data from the various sensors that are outfitted on the boat. Currently, there are 3 sensors that are installed on the boat -

- RADAR – SimRad 4G FMCW Radar [1]

Figure 5.2: RADAR – SimRad 4G FMCW Radar

- IMU – Novatel SPAN [2]

Figure 5.3: IMU – Novatel SPAN

- LIDAR – Velodyne VLP-16 [3]

Figure 5.4: LIDAR – Velodyne VLP-16

RADAR data will be used to detect obstacles like other ships, bridges and shores. The IMU which is GPS enabled will be used to acquire GPS data, Velocity, Orientation from the IMU. As of now, as per the requirements of the sponsor we are not using the LIDAR sensor but we might use it if we do not get the desired results by using only the RADAR.

Once, we have the data from the sensors(RADAR and IMU), the next step is to use this data to find the GPS location of the boat using IMU data, detect obstacles from the RADAR data. Once, we have the location of the boat and the positions of the obstacles are in the environment, we need to plan how the boat navigates to the navigation. All these jobs are done by the software system. Since this is a software intensive project, this system is the heart of the project.

5.2 Software System

In the software system, we have 2 subsystems -

- Perception
- Path Planning

The perception algorithm uses the RADAR data to detect obstacles like boats, shore, bridges etc and feeds this data to the path planning algorithm. The path planning uses this higher level data along with the IMU data to plan the path. As of now, we have not decided which frameworks and algorithms we'll be using in the software system.

5.3 Mechatronics System

The last system is the Mechatronics system. Most of the low level control has already been done by engineers at NREC. We only need to send high level commands to the engine controller to change the speed and direction of the boat.

6. Subsystem Description

6.1 Perception

The basic function of the perception subsystem is to utilize our sensors to detect static and dynamic obstacles and generate a map of the environment. In our design, we will choose radar as our first choice for the perception system. With the radar, we will be able to obtain raw data that indicates the range and relative velocity of the obstacles. After filtering the noise in the data, we can estimate the size and distance of the obstacles. For the obstacles, we will need to avoid static obstacles such as bridges and buoys and then path plan around these objects.

In more technical detail, the radar we will be using for perception is the Simrad Broadband 4G FMCW radar. We will use an open source software called Navico Broadband Radar Plugin[4] to get the data from the radar. Once we are able to get the data, we will experiment with different types of filters to do mapping and obstacle. One promising option we found is the Gaussian Mixture Probability Hypothesis Density Filter (GM-PHD).[5] It is expected to give better results than common methods such as the particle and Kalman filters.

6.2 Planning

In planning, we will solve the problem of path planning of our autonomous boat in an unknown environment. Here, we assume that we would be getting the obstacle data and parameters from perception and IMU data (Heading, GPS coordinates) from the low level controller in the boat. We would be solving the problems considering the path in 2D. Here our subsystem is divided into three major tasks:

- Commanding low level controller to control speed and heading.
- Commanding low level controller to navigate from one GPS waypoint to another.
- Developing an algorithm based on an existing graph search path planning like A to navigate the boat through multiple waypoints considering shortest path. This would also include localizing the vehicle and mapping it on existing map through APIs like Google Maps.
- Plan the new path in case we receive obstacle data/map from the perception sub-system.

After literature review [6], [7], [8], the possible graph search algorithms are:

- Forward search in continuous coordinates, e.g., using rapidly exploring random trees (RRTs)
- Directly formulate the path-planning problem as a non-linear optimization problem in the space of controls or parametrized curves

6.3 Data Logging & Playback

The data logging subsystem will aggregate and save the raw data from the sensors (IMU, RADAR) so that we can do post processing and analysis after our field tests. The logged data will serve as the basis for our playback system which will interface with our simulator in order to replay our sensor data so that we may experiment with our perception and path planning algorithms when were not in the field.

Currently, we plan to utilize ROSs built in functionality and API for logging and playing back data. The built-in functionality is very easy to get started with and consists of simple terminal commands to set up subscription to nodes which are publishing data. However, this may be insufficient since this functionality may not be very tolerant to time constraints and requirements. In which case, we can develop a more robust framework that is fine tuned to our application by using the core API that provides the logging/playback features.

6.4 OCU Subsystem

Operator Controller Unit (OCU) is a user interface device interacting with the controller of autonomous water taxi. OCU provides the visual information and takes commands. OCU shows map with location and trajectory of the water taxi and other boats sailing around it. An operator can input destination information which corresponds to existing ports.

6.5 Simulation

The simulator enables us test the perception and path planning algorithms with a well-built model of boat and the environment. The simulator is a requirement for the autonomous boat development for two main reasons:

- Since we have limited field testing opportunities with the boat due to budgeting constraints, we need other ways of testing for our algorithm. The simulator proves to be a good choice for improving the perception and path planning algorithm with advantages of time and budget saved.
- The simulator provides a way to visualize the algorithms and implementing the simulator successfully will be viewed as a big milestone in our project.

The preliminary choice of our simulator is gazebo. Gazebo is a well-designed simulator platform for robots to test algorithms effectively and efficiently. Gazebo is ROS based and many gazebo models has been developed that can be used in our simulation.

For the simulator, we need to build a model of the boat that includes the accurate size and dynamic nature of the boat. Then we should be able to import an external map like Google maps and data that indicates the position and size of the obstacles to the simulator. Then we can implement the path planning algorithms to get best performance for both dynamic and static obstacles.

7. Trade Studies

7.1 Simulator

Simulation is an essential part in robot design, and the simulator provides an easy and effective way to test and optimize the control algorithms. A good robotics simulator usually gives a 3D model of the robot and environment along with sensors for the robot. We selected simulators with a 3D rendering engine and 3D modeler from the most commonly used simulation software for robotics development and finally chose Gazebo in our design.

We weigh the simulator from five categories:

- Programming language
- Extensibility
- User interface
- Support/tutorial,
- Supported robots and sensors.

First of all, support for programming languages, robots and sensors serve the highest priority. Considering programming language, the given platform of the autonomous boat is ROS-based, therefore a simulator that is based on ROS will be preferred. Moreover, since our team members are more familiar with Python and C++, using a simulator for Python and C++ language will increase development speed. For sensors and robots, the selected simulator Gazebo has support for most commonly used robots, including UAV, UGV, underwater robots, robot arms and humanoids, and most commonly used sensors, including IMU, GPS, camera and laser scanners.

Extensibility and support/tutorials are considered as secondary priorities. Extensibility will enable us to use external APIs. Plugins and tutorials will help us learn the basics of the simulator and get problems quickly solved with resources on forums and wikis. Luckily, for the simulator Gazebo, we are able to find and use the resources including API documentation, public forum, user manual and wiki.

Finally a good user interface is also important for a simulator. Most of the simulators have a user interface to make development easier, but some simulators are still command line based like MORSE, in which case would take more time to get used to compared to other simulators.

Table 7.1: Simulator – Trade Studies

	Weight	Gazebo	MORSE	Webots	V-Rep	Sim Spark
Programming Language	0.25	9	9	8	6	5
Language	0.2	5	6	7	8	5
Extensibility	0.1	7	6	8	8	7
User Interface	0.1	5	3	5	5	5
Support/Tutorial	0.2	9	6	7	6	8
Supported Robots & Sensors	0.25	8	5	6	7	6
Total	1	7.6	6.4	7.2	7	6.2

7.2 Selectiing Map API's for OCU

Operator Controller Unit (OCU) is a user interface device interacting with the controller of autonomous water taxi. OCU provides the visual information and takes commands. OCU shows map and location and trajectory of water taxi and other boats sailing around it, A controller can input waypoints information which is correspond to the location of port.

Table 7.2: OCU Map API Trade Study

	Weight	**OS Map**	**Google Map**	**QGIS+NE**	**G-Earth**	**Bing Map**
Offline usage	0.2	9	0	6	9	0
User Interface	0.2	6	9	6	9	8
Support/Tutorial	0.2	9	9	8	6	8
Extensibility	0.2	8	8	4	6	6
Coding Language	0.2	9	6	6	6	4
Total	1	8.2	6.4	6.0	7.2	5.2

7.3 Sensor Trade Study

Even though our requirement is to focus on radar only, the boat has a mount ready for lidar installation, so we decided to perform a trade study on possible configurations we could have with the sensor. The results indicate that we should stick to radar only for now.

Table 7.3: Sensor Trade Study

	Weight	**Radar alone**	**Lidar alone**	**Radar + Lidar**
Ease to install	0.1	9	8	7
Cost	0.2	9	6	4
Code Integration	0.2	7	7	4
Sponsor's Preference	0.3	10	1	5
Output	0.2	7	6	9
Total	1	8.5	4.9	5.6

8. Project Management

8.1 Work Breakdown Structure

The work breakdown structure is shown in the Figure 8.1.

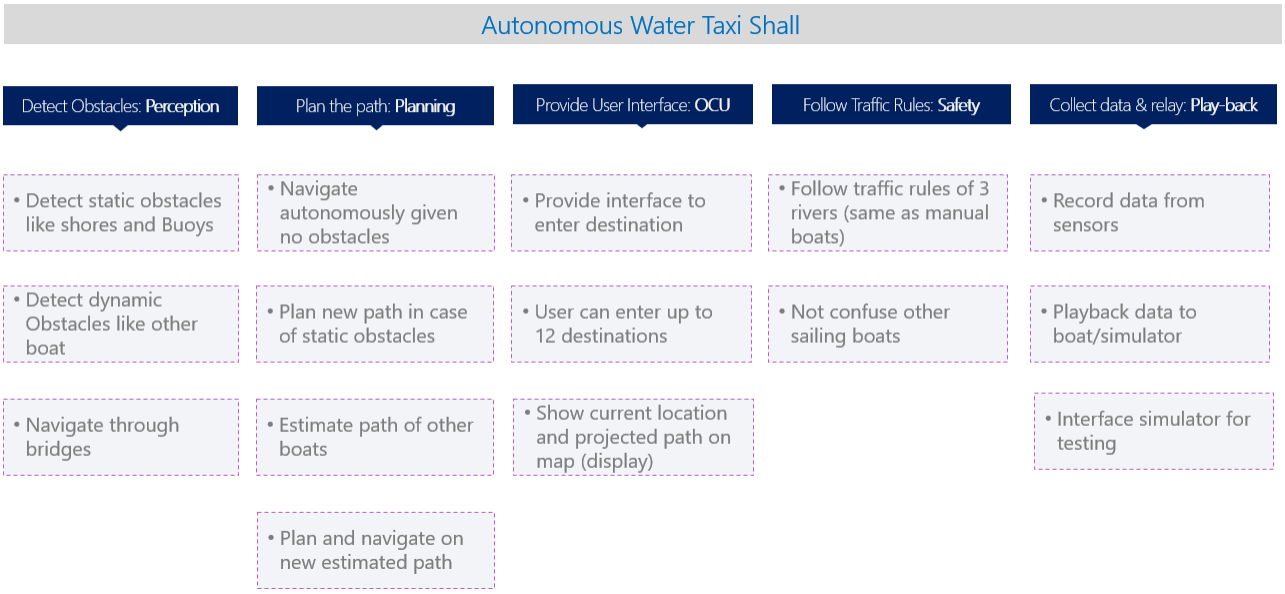


Figure 8.1: Work Breakdown Structure

8.2 Milestones

The milestones for the projects are defined in the subsequent sections.

8.2.1 Fall 2015

Table 8.1 shows the milestones for Fall 2015.

Table 8.1: Fall 2015 Milestones

DATE	TASK
10/20/2015	Test the open source library for perception
	Interfacing with IMU
	Get ROS resources for path planning
11/03/2015	Get filtered data from radar
	Path planning algorithm demonstration
11/20/2015	Get filtered data from IMU
	Select and implement path planning algorithms
12/14/2015	Test detecting the static obstacles
	Develop a simulator for path planning algorithms

8.2.2 Spring 2016

Table 8.2 shows the milestones for Spring 2016.

Table 8.2: Spring 2016 Milestones

DATE	TASK
02/01/2016	Preliminiary design OCU
03/01/2016	Integrate the perception algorithm on the boat
04/01/2016	Integrate the path planning algorithm on the boat
04/30/2016	Demonstrate autonomous navigation of the boat

8.3 Schedule

Figure 8.2 shows the Gantt Chart. Figure 8.3 shows the Calender View. Figure 8.4 shows the Grid View.

8.4 Systems Validation Experiments

As we have many unknowns in our project, so in order to have things under control we plan to operate in fail fast model. Herein, we would be working on the unknown first and create prototypes first and followed by delving deep in each of these domains to fulfil our requirements. The table below contains our major critical testing milestones of our project for the fall and spring semesters. The validation tests have been categorized based on the functions. Our focus for first semester is to get the boat navigation autonomously to destination (without obstacles in path) and also to detect static obstacles with the radar. Integration of both these would be done in the next semester.

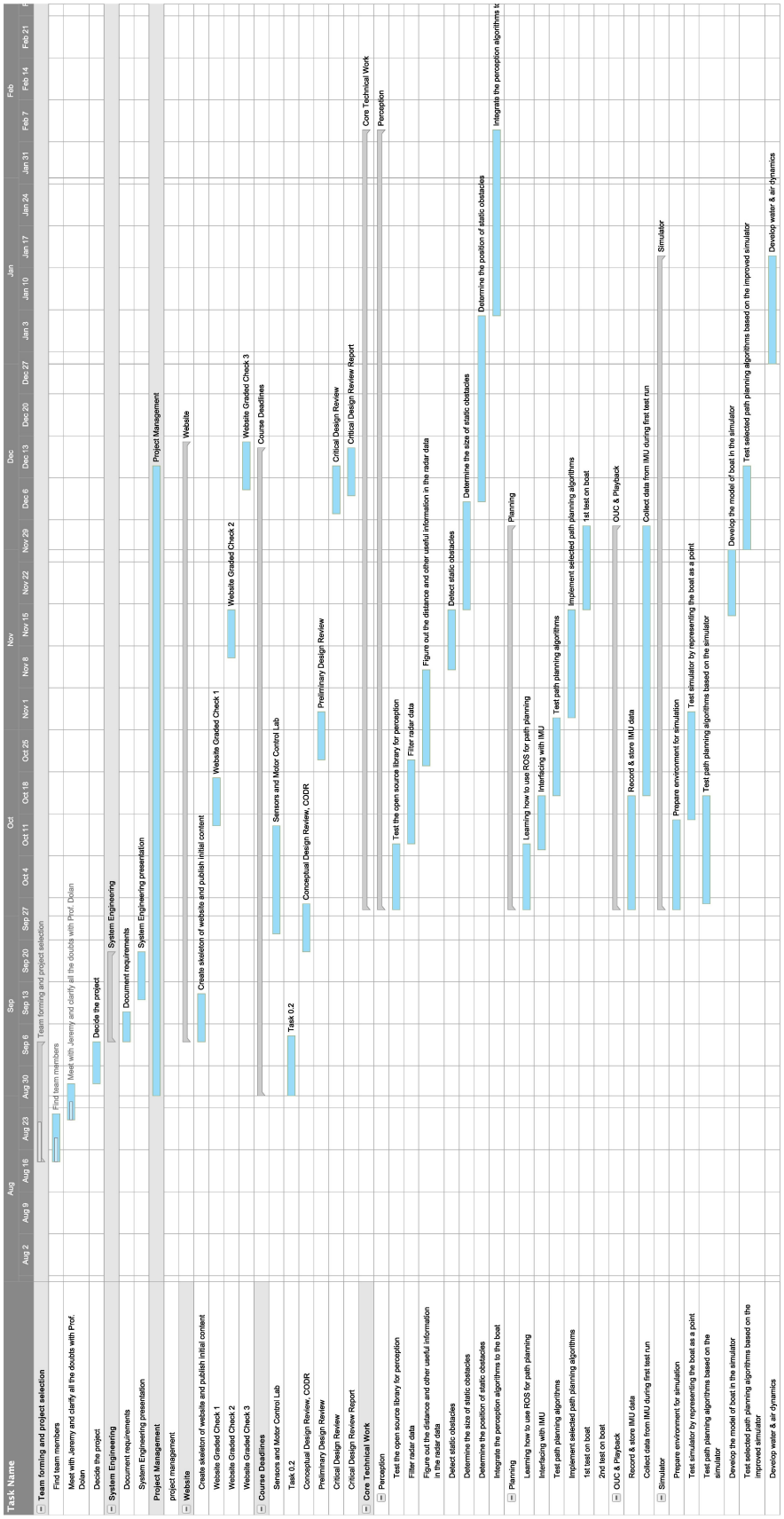


Figure 8.2: Gantt Chart

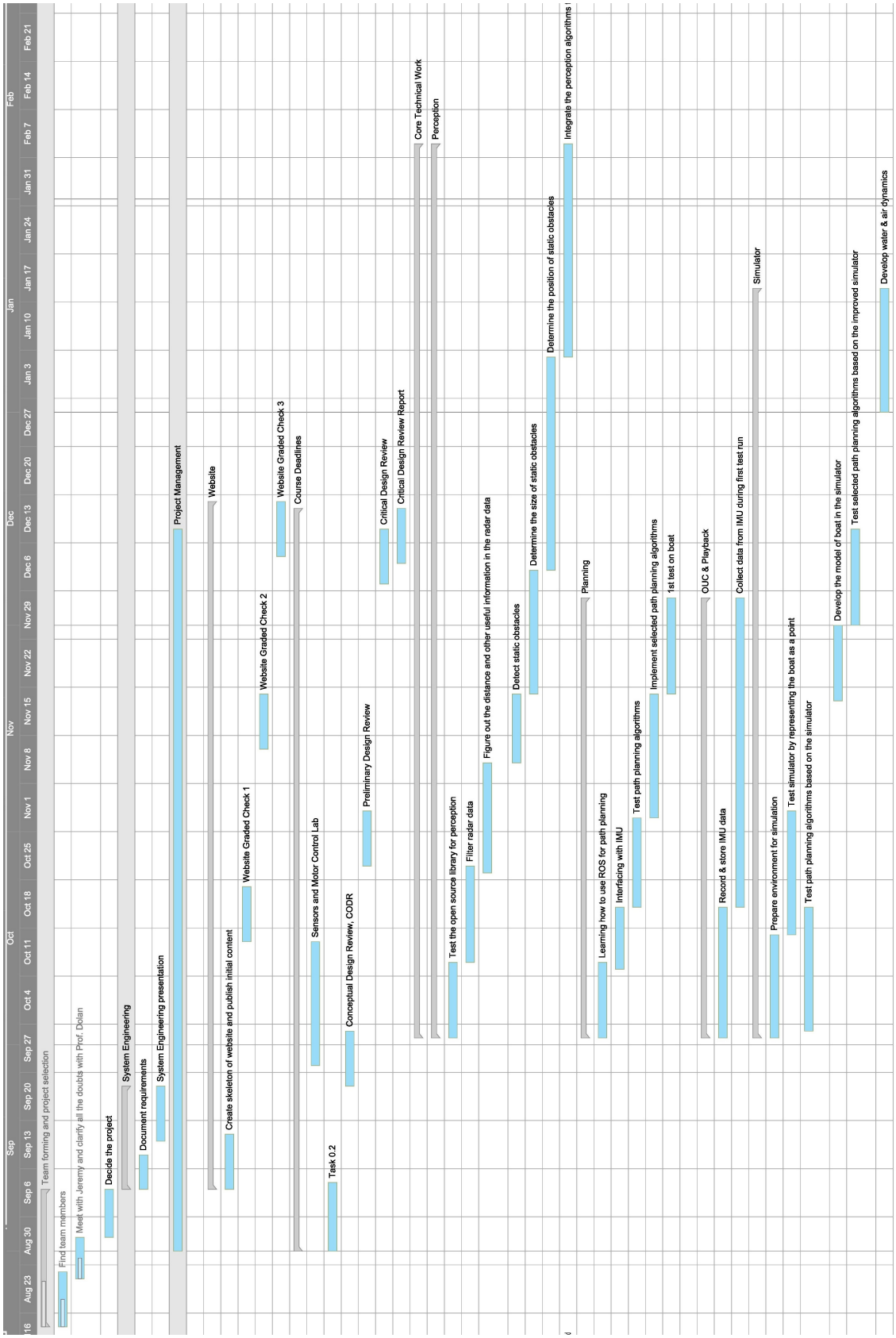


Figure 8.3: Calender View

Task Name
Team forming and project selection
Find team members
Meet with Jeremy and clarify all the doubts with Prof. Dolan
Decide the project
System Engineering
Document requirements
System Engineering presentation
Project Management
project management
Website
Create skeleton of website and publish initial content
Website Graded Check 1
Website Graded Check 2
Website Graded Check 3
Course Deadlines
Sensors and Motor Control Lab
Task 0.2
Conceptual Design Review, CODR
Preliminary Design Review
Critical Design Review
Critical Design Review Report
Core Technical Work
Perception
Test the open source library for perception
Filter radar data
Figure out the distance and other useful information in the radar data
Detect static obstacles
Determine the size of static obstacles
Determine the position of static obstacles
Integrate the perception algorithms to the boat
Planning
Learning how to use ROS for path planning
Interfacing with IMU
Test path planning algorithms
Implement selected path planning algorithms
1st test on boat
2nd test on boat
OUC & Playback
Record & store IMU data
Collect data from IMU during first test run
Simulator
Prepare environment for simulation
Test simulator by representing the boat as a point
Test path planning algorithms based on the simulator
Develop the model of boat in the simulator
Test selected path planning algorithms based on the improved simulator
Develop water & air dynamics

Figure 8.4: Schedule Grid View

8.4.1 Fall 2015 Experiments

PERCEPTION (Fall 2015)

Table 8.3: Perception System Validation (Fall 2015)

Step ID	Step Description
PE.1	Place Simrad Radar at a location wherein we have empty space nearby (50x30x30 feets)
PE.2	Power the radar (12V)
PE.3	Get the raw data from radar to show the interfacing
PE.4	Show filtered data from the Radar
PE.5	Display location and dimensions of the obstacles

Conditions

- *Location:* Outside of NSH building or at NREC
- *Min Size of Area:* 50 feet x 30 feet x 30 feet
- *Needed Equipment:* Simrad Radar, Power for Radar, Obstacles to be detected
- Keep Obstacles of size more than 1mt x 1mt x 1mt

PLANNING (Fall 2015)

Table 8.4: Planning System Validation (Fall 2015)

Step ID	Step Description
PL.1	Get the boat in the river
PL.2	Turn on engine and other controls of boat
PL.3	Command low level controls to control throttle and heading through the laptop
PL.4	Show filtered data from the Radar
PL.5	Autonomously Navigate to Destination

Conditions

- *Location:* Simulators/Rivers
- *Min Size of Area:* 1 mile (without obstacles)
- *Needed Equipment:* Boat, Radar, Fuel, IMU / Simulator (Laptop)

LOG PLAYBACK (Fall 2015)

Table 8.5: Log Playback System Validation (Fall 2015)

Step ID	Step Description
LP.1	Switch On the recording from Radar and IMU while following the above steps of Planning
LP.2	At the end of the path have some obstacles like Kayaks in front of the boat
LP.3	Show the recorded data in a log file

Conditions

- *Location:* River
- *Min Size of Area:* 2 mile
- *Needed Equipment:* Laptop

SIMULATOR (Fall 2015)

Table 8.6: Simulator System Validation (Fall 2015)

Step ID	Step Description
SI.1	Switch On the recording from Radar and IMU while following the above steps of Planning
SI.2	Give source and destination locations to simulator and it will show the planned path

Conditions

- *Location:* Classroom
- *Needed Equipment:* Laptop

8.4.2 Spring 2016 Experiments

PERCEPTION (Spring 2016)

Table 8.7: Perception System Validation (Spring 2016)

Step ID	Step Description
PE.1	Get the boat on the river
PE.2	Switch ON the engines
PE.3	Enter the Destination through OCU
PE.4	Keep 5 obstacles in the path
PE.5	Boat avoids the obstacles while moving towards destination

Conditions

- *Location:* River
- *Test Run:* 2 Miles
- *Needed Equipment:* Simrad Radar, Power for Radar, Obstacles to be detected
- Keep Obstacles of size more than 1mt x 1mt x 1mt

PLANNING (Spring 2016)

Table 8.8: Planning System Validation (Spring 2016)

Step ID	Step Description
PL.1	Get the boat in the river
PL.2	Turn on engine and other controls of boat
PL.3	Enter the Destination through OCU
PL.4	Display the path till the destination in OCU
PL.5	Autonomously Navigate to Destination

Conditions

- *Location:* Simulators/Rivers
- *Min Size of Area:* 1 mile (without obstacles)
- *Needed Equipment:* Boat, Radar, Fuel, IMU / Simulator (Laptop)

LOG PLAYBACK (Spring 2016)

Table 8.9: Log Playback System Validation (Spring 2016)

Step ID	Step Description
LP.1	Switch On the recording from Radar and IMU while following the above steps of Planning
LP.2	Show the recorded data in a log file
LP.3	Playback the data to simulator for testing

Conditions

- *Location:* River
- *Min Size of Area:* 2 mile
- *Needed Equipment:* Laptop

SIMULATOR (Spring 2016)

Table 8.10: Simulator System Validation (Spring 2016)

Step ID	Step Description
SI.1	Switch On the recording from Radar and IMU while following the above steps of Planning
SI.2	Give source and destination locations to simulator and it will show the planned path

Conditions

- *Location:* Classroom
- *Needed Equipment:* Laptop

8.5 Team Member Responsibilities

We have divided the work into -

- Non-Technical Work
- Technical Work

The technical work division is enumerated in Figure 8.5. The name is blue represents the person in charge

Technical Work

Perception	Planning	OCU	Log Play Back	Simulatior	System Integration
Bikram	Tushar	Tae-Hyung	William	Shiyu	Bikram
Shiyu	Tae-Hyung		Tushar		
William					

Figure 8.5: Techinical Team Work Division

The non-technical work division is enumerated in Figure 8.6.

Non-Technical Work

Program Management	Field Management	Web Management	Code Quality Management	Project Kick-off	Team Management
Tushar	William	Bikram	Shiyu	Team	Tae-Hyung
• Release Mgt. • Scrum Scheduling meetings • Communication • Purchasing	• Logistics for Field Trials • Data collection and management	• Publish content website • Handle Latex for documentation	• Repository • Code Reviews • Version control • Check-ins	• Structure to kick-off the project • Documentation and presentations	• Team Culture • Stress management • COI Management

Figure 8.6: Non Technical Team Work Division

8.6 Budget

Table 8.11 shows the budget breakdown of the project.

Table 8.11: Budget

S.No	Description	Cost	Subtotal	Quantity	Total
1	Field Trial				
	Safety Personal	500			
	Fuel	200			
	Boat Transportation	100	800	25	20000
2	Simrad Radar	2500	2500	1	2500
3	Travel arrangements	1000	1000	1	1000
4	Miscellaneous	500	500	1	500
	Total				24000

8.7 Risk Management

With each specific risk factor, we assign a impact level, which can be low, medium, or high risk.

Table 8.12: Risk Management

Risk Factor	**Description**	**Impact**	**Mitigating Actions**
RADAR is Insufficient	Radar can yield poor performance and be difficult to use.	Medium	Investigate other sensors as we do development.
Perception Algorithm Scope	Sponsor requires perception to be a key deliverable.	High	Main goal is to detect static obstacles. Adequate for a demo.
Path Planning Scope	It would be nice to have a sophisticated global & local planner.	Low	A nave local planner should be sufficient in the worst case.
Limited Development Time	Work limited to weekends due to travel time performance	Medium	Set aside $1500 budget for taxis, also purchase 2^{nd} radar
Limited Testing Trials	We only have 2530 field tests performance and be difficult to use.	Medium	Appoint a field manager to plan logistics of each test.
The weather	The winter will severely hinder our testing opportunities.	Low	Create a advanced simulator.

Reference

[1] Simrad. Broadband-4G-Radar. http://www.simrad-yachting.com/en-US/Products/Radar/Broadband-4G-Radar-en-us.aspx, 2015. [Online; accessed 02-October-2015].

[2] Velodyne. VLP-16 Datasheets. http://velodynelidar.com/docs/datasheet/63-9229_VLP16_Datasheet_Rev-A_Web.pdf, 2015. [Online; accessed 02-October-2015].

[3] Span. Inertial Measurement Units (IMUs) . http://www.novatel.com/products/span-gnss-inertial-systems/span-imus/, 2015. [Online; accessed 02-October-2015].

[4] canboat. Navico Broadband Radar Plugin. http://opencpn.org/ocpn/navico_radar, 2014. [Online; accessed 02-October-2015].

[5] Bryan Clarke, Stewart Worrall, Graham Brooker, and Eduardo Nebot. Towards mapping of dynamic environments with fmcw radar. In *Intelligent Vehicles Symposium (IV), 2013 IEEE*, pages 147–152. IEEE, 2013.

[6] Dmitri Dolgov, Sebastian Thrun, Michael Montemerlo, and James Diebel. Practical search techniques in path planning for autonomous driving. *Ann Arbor*, 1001:48105, 2008.

[7] Cédric Pradalier and Gion-Andri Büsser. Design and implementation of a navigation algorithm for an autonomous sailboat.

[8] Clément Pêtrés, Miguel-Angel Romero-Ramirez, and Frédéric Plumet. Reactive path planning for autonomous sailboat. In *Advanced Robotics (ICAR), 2011 15th International Conference on*, pages 112–117. IEEE, 2011.