

Software Design Document for RoboSub

Version 1 approved

Prepared by Ricardo Medina, Heriberto Gonzalez, Gabriela Cortes-Mejia, Sana
Shaikh, Albert Chew, Kevin Williams, Horacio Mondragon, Brandon Pham,
Wilson Weng

Civil Engineering Department and Mechanical Engineering Department, CSULA

May 14, 2021

Table of Contents.....	pg 2
Revision History.....	pg 5
1. Introduction.....	pg 6
1.1. Purpose.....	pg 6
1.2. Intended Audience and Reading Suggestions.....	pg 6
1.3. System Overview.....	pg 6
2. Design Considerations.....	pg 7
2.1. Assumptions and dependencies.....	pg 7
2.2. General Constraints.....	pg 7
2.3. Goals and Guidelines.....	pg7
2.4. Development Methods.....	pg 8
3. Architectural Strategies.....	pg 8
4. System Architecture.....	pg 9
4.1.	pg 9
4.2.	pg 9
5. Policies and Tactics.....	pg 10
5.1. Specific Products Used.....	pg #10
5.2. Requirements traceability.....	pg #
5.3. Testing the software.....	pg #

5.4.	Engineering trade-offs.....	pg #
5.5.	Guidelines and conventions.....	pg #
5.6.	Protocols.....	pg #
5.7.	Maintaining the software.....	pg #
5.8.	Interfaces.....	pg #
5.9.	System's deliverables.....	pg #
5.10.	Abstraction.....	pg #
6.	Detailed System Design.....	pg 13
6.1	Mission Planning.....	pg 13
6.1.1	Responsibilities.....	pg 13
6.1.2	Constraints.....	pg 14
6.1.3	Composition.....	pg 14
6.1.4	Uses/Interactions.....	pg 14
6.1.5	Resources.....	pg 14
6.1.6	Interface/Exports.....	pg 14
6.2	Navigation.....	pg 15
6.2.1	Responsibilities.....	pg 15
6.2.2	Constraints.....	pg 15
6.2.3	Composition.....	pg 15
6.2.4	Uses/Interactions.....	pg 15
6.2.5	Resources.....	pg 15
6.2.6	Interface/Exports.....	pg 16
6.3	Controls.....	pg 16

6.3.1	Responsibilities.....	pg 16
6.3.2	Constraints.....	pg 16
6.3.3	Composition.....	pg 16
6.3.4	Uses/Interactions.....	pg 16
6.3.5	Resources.....	pg 17
6.3.6	Interface/Exports.....	pg 17
6.4	Computer Vision.....	pg 17
6.4.1	Responsibilities.....	pg 17
6.4.2	Constraints.....	pg 17
6.4.3	Composition.....	pg 17
6.4.4	Uses/Interactions.....	pg 18
6.4.5	Resources.....	pg 18
6.4.6	Interface/Exports.....	pg 18
7.	Hardware	
7.1	Controls Hardware.....	pg 18
7.2	Navigation Hardware.....	pg 19
7.3	Main Computer.....	pg 19
8.	Database Design	pg 21
9.	User Interface.....	pg 21
10.	Requirements Validation and Verification.....	pg 21
11.	Glossary.....	pg 24
12.	References.....	pg 24

Revision History

Name	Date	Reason For Changes	Version
Ricardo Medina	5/14/2021		1.0

1. Introduction

1.1 Purpose

The purpose of this document is to detail the software in development for the AUV project.

The mission planning portion of the AUV being constructed by the Senior Design Engineering team is to explain the SMACH model developed to control the AUV subsystems.

The Controls subsystem purpose is to control the Robosub movement with eight thrusters using PID controllers. We used three different axes for control movement. Pitch axis to tilt the sub forward or backwards, Roll axis to move the sub side to side, and YAW to rotate the sub right and left.

The Computer vision subsystem is designed to recognize specific images in the underwater environment and send relevant data to the mission planning system.

This document shall allow future developers of the product to modify and expand the state machine task states to accomplish the specific outlined tasks for the competition.

1.2 Intended Audience and Reading Suggestions

The types of readers that the document is intended for is for developers and users that intend to continue the work done by the 2020-2021 Senior Design Team. The reason for this is due to the inability of the entire team to test the AUV within unforeseen circumstances.

1.3 System Overview

The software system is split up into component parts, each with its own specific tasks as follows:

1.3.1 Controls

Design and test a controls system to allow 6 degrees of movement with precision and accuracy.

1.3.2 Mission Planning

Design and implement the Autonomous code controlling all AUV subsystems including subsystem communication. This system is geared towards completing tasks set forth by the robosub competition.

1.3.3 Navigation

Implement sensor data acquisition and processing using arduino and serial connections.

1.3.4 Computer Vision

Design and training a Computer Vision system capable of recognizing the objects and tasks set forth by the Robosub competition.

2. Design Considerations

2.1 Assumptions and Dependencies

- Refer to section 7 Hardware for description of hardware dependencies
- Linux
 - The dependent ROS software must be installed on a linux environment.

Factors that may affect the requirements in the document are:

- The use of other Ubuntu versions
- Another version of ROS
- A programming languages other than Python and C++

2.2 General Constraints

Due to the ROS version on the platform, ROS nodes are restricted to be written in one of the following languages:

- Python 2.7
- C++

Any Code to be run on an Arduino must be written in the Arduino language.

Main safety requirements are to not damage any components of the AUV as well as any item involved in the execution of competition tasks. Additionally, we must not cause any harm to the divers observing the competition in the water.

A significant limitation that impacts the design of the system's software is the inability to properly test the software. The reason for this limitation is due to the fact that the engineering team was unable to build an AUV due to the 2020 pandemic.

2.3 Goals and Guidelines

One goal of the software is to design and build it in a way that allows another team to continue work on the AUV instead of starting over from square one.

- Mission Planning - Users will be able to build upon existing state machines as well as adding new state machines if needed. These states are independent, sharing utility functions and states

- Controls - Designing controls code to be tested when the hardware is completed in 2021
- Navigations - Test sensors independently and create packages dedicated to each individual sensor
- Computer Vision - Design the workflow for labeling and training a computer vision system for use on a yearly basis.

Each of these goals will be focused on the documentation and commenting of code with the expectation of reducing the time to onboard a new senior design team.

2.4 Development Methods

Agile Development philosophy was used because of unavailability of hardware from the Senior Engineering team. Each burst was discussed beforehand on what is able to be accomplished without having the hardware. After each burst, teams discussed what the best way to proceed, what tests can be written, and what designs can be made, but not implemented.

3. Architectural Strategies

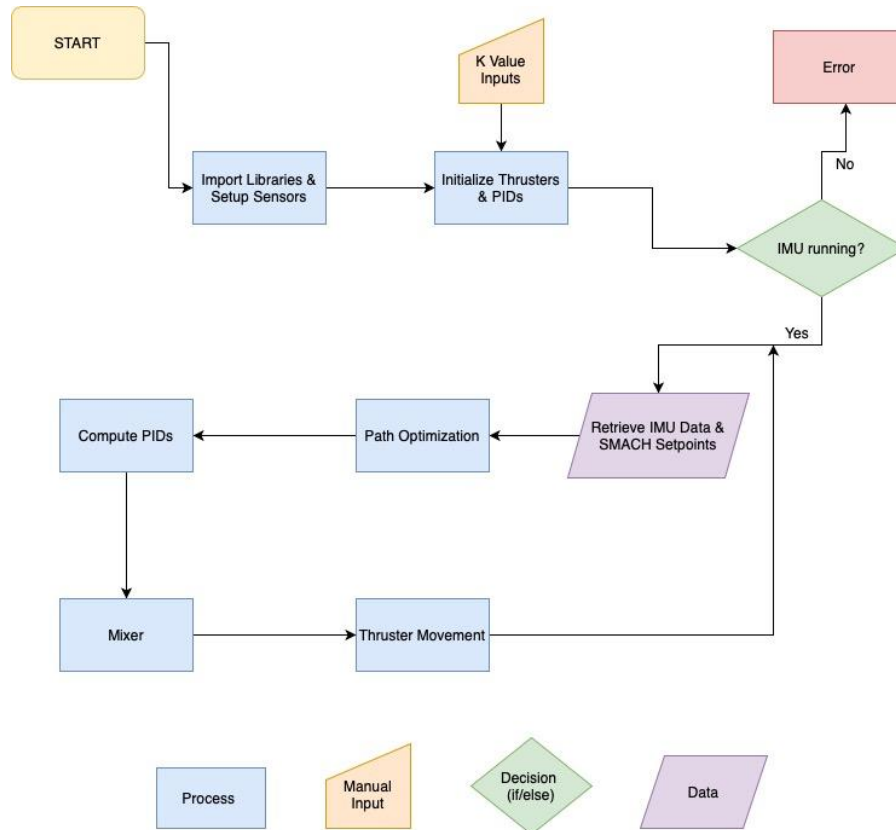
Software was split using ROS packages, stand alone software dedicated to a particular task. These were split both in terms of functionality, or sensors. This approach was used to allow for reuse of code that does not need to be changed, for example IMU and Sonar should be reusable year after year.

In addition the SMACH system was separated into various sub systems, each with the goal of completing a single tasks, allowing for faster development, debugging and testing times, as well as readability of code for similar tasks that reappear in the competition.

An error package was also implemented to simplify monitoring of the various functionality. This package can be expanded as functionality and requirements change.

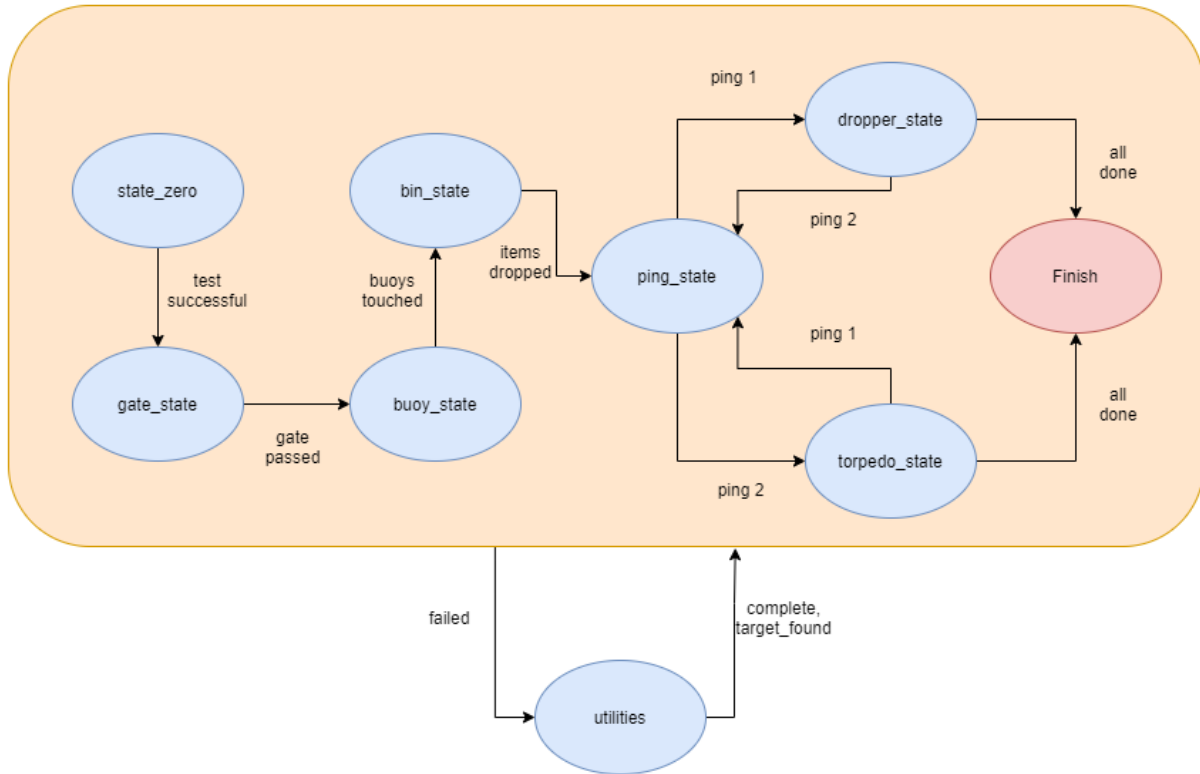
4. System Architecture

Controls



Mission Planning

Flowchart



5. Policies and Tactics

5.1 Choice of which specific products used

Arduio IDE

ROS - <http://wiki.ros.org/>

Arduino - <https://www.arduino.cc/>

Smach - <http://wiki.ros.org/smach/Documentation>

Ubuntu 18.04.5 - <https://releases.ubuntu.com/18.04.5/>

Ubuntu 20.04.2.0 - <https://releases.ubuntu.com/20.04/>

VMware Workstation -

<https://my.vmware.com/en/web/vmware/downloads/details?downloadGroup=PLAYER1556&productId=800&rPid=47861>

5.2 Plans for ensuring requirements traceability

- Traceability is enforced by requiring appropriate readme documents that explain how to utilize the different programs the AUV uses as well as the tasks that the AUV has to perform using those programs.

5.3 Hierarchical organization

The Software shall follow the following organization.

Each package will be located in its own directory and contain the following:

- Readme File containing description of package
 - Software requirements
 - Interface specification
 - Hardware description (if applicable)
 -
- Start script named start.py (or c++ file where appropriate)
- Required libraries (for hardware components)
- Scripts (or src) directory containing all software.
- Design Images of software (where appropriate)

5.4 Software Implementation Procedure

The software must be cloned from the repository in the catkin workspace folder created by ROS

Example structure:

Catkin_ws

-Software Directory

-src

The software directory folder must be renamed to src, removing the existing src directory.

To compile the following command shall be run:

Catkin_make

6. Detailed System Design

This project is split across a number of submodules each with their own environments and considerations.

6.1 Mission Planning

6.1.1 Responsibilities

- The main responsibility of Mission Planning is to utilize SMACH(state machine) to give instructions to the AUV so that it can perform the tasks provided to it.

6.1.2 Constraints

- SMACH states work within the AUV one task at a time (unless states are set to run in concurrence)
- SMACH is a task level architecture for complex instructions, so efficiency is not a strong point

6.1.3 Composition

- ROS – Robot Operating System
- SMACH – State Machine

6.1.4 Uses/Interactions

The functions of the software involve:

- Defining the current state that the AUV is in
- Taking data obtained from other components of the AUV
- Defining when the AUV transitions to another state from its current state

- Passing data between different states
- An example of the state machine is the gate task, where the AUV uses data from Computer Vision to help it recognize a gate in one state and transitions to a state where the AUV uses instructions provided by Controls to move toward the gate

6.1.5 Resources

Smach is a Python library that is independent of ROS.

6.1.6 Interface/Exports

- Mission Planning's product does not directly interface with any hardware from the AUV, only through interfacing with other systems that directly interact with hardware.
- Other software products that the AUV will utilize is Python 2.7 with APIs from ROS Melodic and Smach within the Ubuntu 18.04.5 operating system. Users may have to download VM ware Workstation player to use Ubuntu.
- The product in terms of mission planning does not have requirements that involve the use of communication functions like a network server or electronic forms. The reason for this is that all the functions that the product will perform is created in a way that allows the AUV to perform without any dependency to the internet. However, these functions are placed in the AUV by use of websites designed to store code integral to the AUV's function.

6.2 Navigation

6.2.1 Responsibilities

- The main responsibility for this component is to give the sub a sense of direction and tell it where it should go, whether hearing pingers in the water, or detecting near objects it should stray away from. This subsection of the AUV does not directly mingle with clientele and only speaks to other parts of the AUV.

6.2.2 Constraints

- The blue robotics pinger is limited in what it can detect. It may be needed to detect an object in its direct pathway but if a larger object lies directly behind a smaller object, the distances for the larger object will be returned. This effectively renders this component useless for objects on the smaller side but has an astounding effect on navigating away from potentially hazardous objects that could cause damage if crashed into.

6.2.3 Composition

- With the use of the blue robotics ping sonar, we need multiple interfaces and multiple OS to get the sonar up and running. First, we need an Arduino board and its complimentary software for output validation of the sonar. Next, the use of Robotic Operating System or “ROS” running on Ubuntu or Linux to run parallel with the Arduino sonar code for transmission of distance data throughout the AUV.

6.2.4 Uses/Interactions

- The navigation component only interacts with other parts of the AUV with the use of ROS nodes. These nodes gather information from the sonar connected Arduino and passes this information, in the case of the blue robotics ping sonar, distance information to every component of the AUV. This component has no interaction with clientele or humans as it is only used within the AUV itself to inform other AUV components of incoming objects or how far the sub is from larger submerged entities.

6.2.5 Resources

- The Arduino connected Blue Robotics Ping Sonar needs an Arduino board and its complimentary software, mainly the serial monitor. We will use the Serial ROS Arduino library which will connect the Arduino code to ROS. In addition to the arduino and complimentary ROS package, we will need Ubuntu 20.04 Focal to set up the ROS environment which runs ROS Noetic to allow the two systems, Arduino and ROS, to talk to each other. For ROS, we have set up the publisher/subscriber libraries.

6.2.6 Interface/Exports

- With ROS and the Arduino we have three different variables with one data type, float. Each variable is responsible for specific transfer of information. “Chatter” is distance in mm, “Chatte” is distance in inches, and “Chatt” is confidence level.
 - Each is set up as the following:
 - `std_msgs::Float32 data;`
 - `ros::Publisher chatter(“chatter”, &data);`
 - `std_msgs::Float32 inches;`
 - `ros::Publisher chatte(“chatte”, &inches);`
 - `std_msgs::Float32 confidence;`
 - `ros::Publisher chatt(“chatt”, &confidence);`

6.3 Controls

6.3.1 Responsibilities

- The main responsibility for the Controls subsystem is to control the AUV's movement utilizing the eight thrusters with PID controllers. This is done through a combination of publishers, subscribers, and the API.

6.3.2 Constraints

- Performing different test cases on the PID balancing code with the restrictions of the 2020-2021 school year
- Communication between the software developers and hardware developers working on the same or different parts of the AUV.

6.3.3 Composition

- ROS – Robot Operating System
- PID – Proportional, Integral, and Derivate
- SMACH – State Machine
- PWM – Pulse-Width Modulation
- IMU – Inertial Measuring Unit
- Arduino IDE - Arduino Integral Development Editor
- ESC - Electronic Speed Controller

6.3.4 Uses/Interactions

The functions of the software involved:

- Retrieving the current state as input that the AUV is in using the IMU sensors
- Obtaining the setpoint values of the desired position from the SMACH node
- Calculating error values as the difference between input and setpoint
- Outputting a PWM speed for the respective thrusters
- Combining the output values from the PID controllers
- Calculating the final PWM signals to send to the thrusters

6.3.5 Resources

- PID control with Arduino – http://www.electrooobs.com/eng_robotica_tut6.php
- Setting up a PID on a drone – https://www.technik-consulting.eu/en/optimizing/drone_PID-optimizing.html
- PID library – https://github.com/br3ttb/Arduino-PID-Library/blob/master/PID_v1.h
- IMU library - https://www.arduino.cc/reference/en/libraries/mpu6050_tockn

- Motor calibration - <https://www.youtube.com/watch?v=1EnMEOc9Psc>

6.3.6 Interface/Exports

- Through ROS we are able to communicate with the Navigation Module to collect the setpoint values from the State Machine.

6.4 Computer Vision

6.4.1 Responsibilities

- To process video input from the two cameras onboard and detect and classify the objects throughout the course.

6.4.2 Constraints

- Best way to run the Computer Vision Module is to use a capable GPU (Graphics Processing Unit)
- Storing the video output of the algorithm can take up a lot of storage
- Computer Vision Module can introduce thermal throttling onto the system
- Learning Dataset may contain many pictures or videos that take up large amounts of storage.

6.4.3 Composition

- DARKNET: Open-Source Neural Network Framework
- OPENCV: Python Computer Vision Library
- YOLO: Real-time object detection algorithm
- CUDNN: Deep Neural Network library (cuDNN) is a GPU-accelerated library of primitives for deep neural networks.

6.4.4 Uses/Interactions

- The Computer Vision Module is made up of a Convolutional Neural Network algorithm called YOLO which is used for real time object detection, the algorithm is trained using Darknet and given a dataset of images which have been labeled to distinguish the objects we want to detect. The algorithm was trained on over 4 thousand images and was trained to detect 12 unique objects. The training was done over a 50 hours period and had gone through 24 thousand iterations and had processed over a million images.

6.4.5 Resources

- DARKNET <https://github.com/AlexeyAB/darknet>
- OPENCV <https://opencv.org/>
- YOLO <https://pjreddie.com/darknet/yolo/>
- CUDNN <https://developer.nvidia.com/cudnn>

6.4.6 Interface/Exports

- The Computer Vision Module directly communicates with the Mission Planning Module to output a package of information that include:
 - Name of Object
 - Confidence Level
 - Horizontal Error
 - Vertical Error
- This communication is done by using the Robotic Operating System which allows the modules to communicate with one another.

7. Hardware

7.1 Controls Hardware

- Arduino Uno:

Arduino Uno is a microcontroller that can control motors. The Arduino is an open-source electronics platform based on easy-to-use hardware and software. The Arduino boards are able to read inputs - light on a sensor, a finger on a button, or other sensors and turn it into an output - activating a motor, sending pwm to esc and so on.

- Thrusters:

An electric motor is an electrical machine that converts electrical energy into mechanical energy. Most electric motors operate through the interaction between the motor's magnetic field and electric current in a wire winding to generate force in the form of torque applied on the motor's shaft.

- ESC (electronic speed controller):

An electronic speed control is an electronic circuit that controls and regulates the speed of an electric motor.

- Propellers:

A propeller consists of blades. Each blade is designed with a sight curve and a particular shape, which helps move the water or air down and past it. The leading edge is the part of the propeller blade that hits the water or air first. The trailing edge is the part where the water or air leave the blades.

- **Power Supply:**

A power supply is an electrical device that supplies electric power to an electrical load. All power supplies have a power input connection, which receives energy in the form of electric current from a source, and one or more power output connectors that deliver current to the load.

- **MPU6050:**

MPU6050 is basically a sensor for motion processing devices. It is capable of processing nine-axis algorithms, it captures motion into X, Y, and Z axis at the same time.

7.2 Navigation Hardware

- **Arduino Mega and Uno**
 - . Limited to Arduino Language
 - . Limit of 40mA on each pin
 - . Total limit of 200mA for all pins
 - . Recommended input of 7~12V
 - . Maximum Input of 6~20V
 - . Total output of 5.5V
 - . Current outputs USB connection 500mA - 1A
- **Blue Robotics Sonar**
 - Must be grounded
 - Input of 5.5V
 - Uses Arduino library or Python 3 with raspberri pi
 - Serial TTL to usb required for computer use
 - Output display on Arduino Serial Monitor or Blue Robotics PingViewer application
- **Vector Nav-100 Imu**
 - Attitude:
 - Yaw, Pitch, & Roll
 - Quaternions
 - Direction Cosine Matrix
 - Angular Rates:
 - Bias-Compensated
 - Calibrated X, Y, & Z Gyro Measurements
 - Acceleration:
 - Calibrated X, Y, & Z Measurements

- Magnetic:
 - Calibrated X, Y, & Z Measurements
 - Barometric Pressure
- Teledyne Pathfinder DVL
 - X - Traversal
 - Y - Traversal
 - Depth
- Blue Robotics Bar30 pressure Sensor
 - Temperature (Celsius)
 - Pressure(mBar)

7.3 Main Computer

The main computer of the AUV is a Jetson TX2 with following capabilities:

- NVIDIA Maxwell™ GPU with 256 NVIDIA® CUDA® Cores
- Quad-core ARM® Cortex®-A57 MPCore Processor
- 4 GB LPDDR4 Memory
- 16 GB eMMC 5.1 Flash Storage
- 10/100/1000BASE-T Ethernet
- USB 3.0 Type A
- USB 2.0 Micro AB (supports recovery and host mode)
- HDMI
- M.2 Key E
- PCI-E x4
- Gigabit Ethernet
- Full-Size SD
- SATA Data and Power
- GPIOs, I2C, I2S, SPI*
- TTL UART with Flow Control
- Display Expansion Header*
- Camera Expansion Header*

8. Database Design

No database was implemented for this design.

9. User Interface

No User interface was implemented at this time.

10. Requirements Validation and Verification

10.1 Mission Planning

10.1.1 Functional Requirements

Requirement No.	Requirement Description
10.1.1	The system shall transition to different states based on the current task the AUV is faced with.
10.1.2	The system shall receive data from other systems that are required to perform the current state.
10.1.3	The system shall send required inputs for other systems to perform their operations.
10.1.4	The system shall follow a predetermined plan of the competition tasks to perform.
10.1.5	The system shall consist of a series of state machines for each individual task in one larger overall state machine.

10.1.2 External Interface Requirements

Requirement No.	Requirement Description
10.1.1	The system shall handle data using ROS' publishers and subscribers.
10.1.2	The system shall interface the AUV's subsystems together to operate the AUV.
10.1.3	The system shall output various movement commands to the controls system for the AUV's mobility.

All of the requirements for Mission Planning are fulfilled from the use of the other components of the AUV.

10.2 Navigation

Requirement No.	Requirement Description
10.2.1	Shall interface with the VECTORNAV IMU and record/send pitch, yaw, roll data to the AUV
10.2.2	Shall interface with the DVL and record/send north-south, east-west translational data to the AUV
10.2.3	Shall interface with the Barometer and record/send depth data to the AUV
10.2.4	Shall interface with the Hydrophones and record/send Pinger location data to the AUV
10.2.5	Shall interface with the Sonar and record/send object detection data to the AUV
10.2.6	Shall use sensor data, and provide output to controls to navigate the AUV to its next intended task

10.3 Controls

10.3.1 Functional Requirements

Requirement No.	Requirement Description
10.3.1	The Arduino code shall output proper speed to its thrusters so it can balance the AUV.
10.3.2	The Arduino code shall subscribe from ROS topics of pitch and roll angles to determine if the Arduino can make it balance with angle 0 degree.
10.3.3	The Arduino code shall output proper speed to its motors to move forward or backward as it receives input from the ROS navigation system.

10.3.2 External Interface Requirements

Requirement No.	Requirement Description
10.3.1	The system shall handle data using ROS' publishers and subscribers.
10.3.2	The system shall interface the AUV's subsystems together to operate the AUV.
10.3.3	The system shall output various movement commands to the Controls subsystem for the AUV's mobility.

10.4 Computer Vision

10.4.1 Functional Requirements:

Requirement No.	Requirement Description
10.4.1	<p>The software shall be trained on a dataset large enough for the CNN to learn and distinguish the difference between the various objects in the course</p> <ol style="list-style-type: none"> 1. 500-1000 images per object 2. Each Image in the Dataset must be labeled using OpenLabeling
10.4.2	<p>The software shall be trained using Darknet and using the YOLOv4 algorithm</p> <ol style="list-style-type: none"> 1. Edits must be made to the algorithms variables to be applicable to the need of the dataset that it is given 2. Darknet requires a txt file listing the paths to all the images in the dataset along with the config file that would be edited for dataset 3. Training can vary in time depending on the GPU used to train the CNN

10.4.2 External Interface Requirements:

Requirement No.	Requirement Description
10.4.2	The Computer Vision software will take input from two Blue Robotic Low Light cameras where each cameras video feed will be passed into the Darknet software where it is then compiled and ran onto the YOLOv4 CNN and the output will be the name of the object of interest that is in present view of the AUV and a estimated angle at which the camera is viewing the object of interest.

11. Glossary

- AUV – Autonomous Underwater Vehicle
- ROS – Robot Operating System
- SMACH – State Machine
- DARKNET – Open Source Neural Network Framework
- OPENCV – Computer Vision Library
- YOLO – Real-time Object Detection Algorithm
- Convolutional Neural Network (CNN) – a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other.
- CUDNN - CUda Deep Neural Network

12. References

YOLOv4: Optimal Speed and Accuracy of Object Detection – <https://arxiv.org/pdf/2004.10934.pdf>

Darknet Website (no longer updated) – <https://pjreddie.com/darknet/>

Darknet GIT Repository – <https://github.com/AlexeyAB/darknet>

OpenLabeling – <https://github.com/Cartucho/OpenLabeling>