Software Requirements Specification

for

Robosub

Version 1.01 approved

Prepared by Carl Solli, Christian Castillo, Ashkan Aledavoud, Robin Romero, Alan Chan, Leslie Araujo, Edwin Tran, Bryan Sanchez, Jeanie Jeon, Daniel Valadez

May 13th 2022

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
Christian Castillo	10 Dec 2021	Starting document as team	0.01

1. Introduction

1.1 Purpose

The purpose of this document is to give a general overview of all requirements necessary for the successful operation of an Autonomous Underwater Vehicle for the express purpose of competing in the ROBONATION RoboSub Competition. This project is composed of various submodules, each with their own dedicated software requirements. This document will include the requirements for all the different submodules.

1.2 Intended Audience and Reading Suggestions

This document is intended for developers that are working on or need an insight into this project.

1.3 Product Scope

This project will be built up of different ROS packages. Here are a list of the packages that will be developed:

- User Interface
- Mission Planning
- Computer Vision
- Controls & Sensors
- Guidance Navigation Control

These packages will be run simultaneously, communicating amongst each other to successfully operate the AUV.

1.4 Definitions, Acronyms, and Abbreviations

AUV - Autonomous Underwater Vehicle

ROS - Robot Operating System

SMACH - State Machine

PID - proportional, integral, and derivative

GUI - Graphical User Interface

1.5 References

RoboSub 2022 Handbook:
https://robonation.org/app/uploads/sites/4/2022/05/2022-RoboSub_Team-Handbook_v2.0.pdf

ROS - https://www.ros.org/

SMACH - http://wiki.ros.org/smach/Documentation

Smach Viewer - http://wiki.ros.org/smach_viewer#Documentation

2. Overall Description

2.1 System Analysis

The purpose of the AUV software is to compete in a Robotic Submarine competition in which each AUV must complete a set of tasks utilizing each component feature included in the AUV. The various tasks include detecting and recognizing gates for the AUV to pass through while selecting a side to follow through for the rest of the competition, detect and swim into a buoy with an image of the side that was previously selected, shoot mini missiles through a poster's designated area, maneuver to a bin and be able to lift the bin's cover and drop items into the bins, and finally set bottles onto designated spots in an octagon.

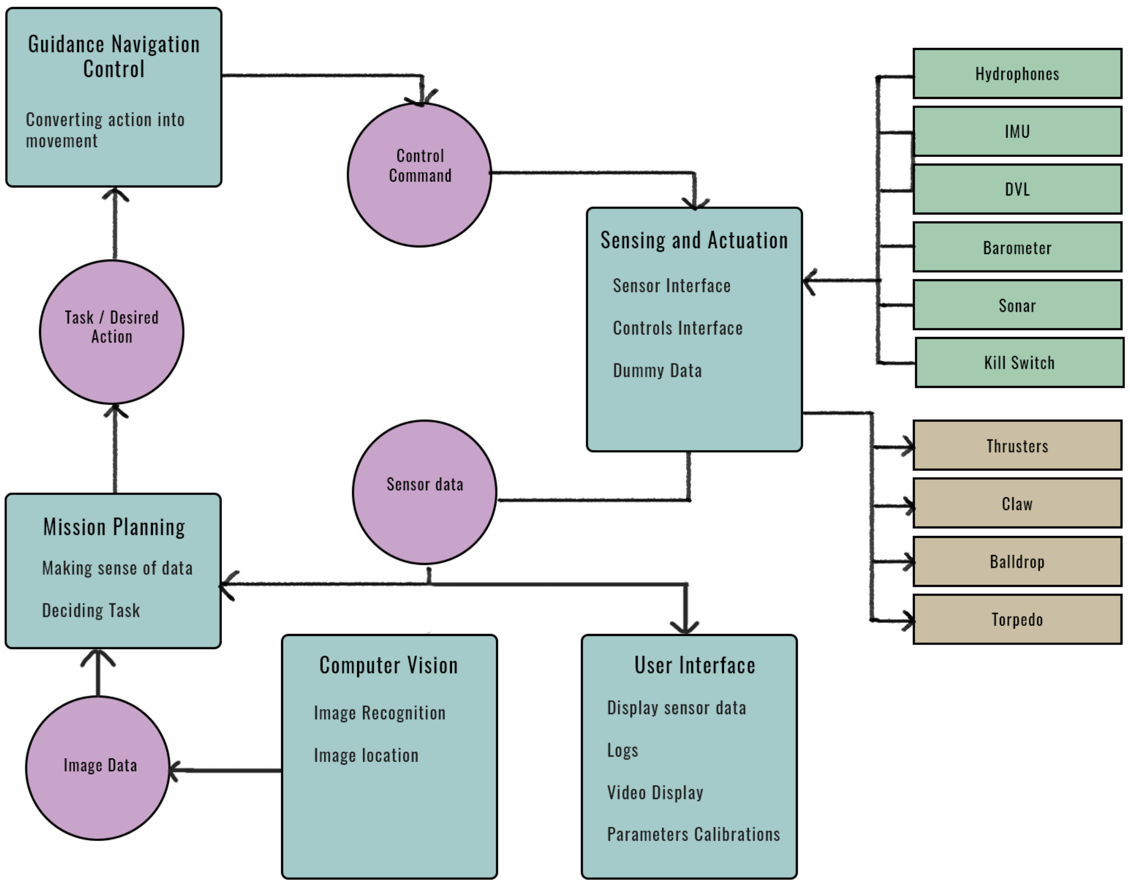
To complete these tasks the AUV will use image recognition software to detect and recognize task specific objects. AUV sensors will receive data such as pool depth and AUV's current position to update Mission Planning's software to report task distance. The data gathered through the image recognition software will dictate if the current task is completed. After receiving data from both Computer Vision and Mission Planning, the AUV needs to move onto the next task or continue with the current task.

2.2 Product Perspective

The AUV's software depends on the state machine provided by Mission Planning to determine which task to complete. Mission Planning's software also depends on Computer Vision's software which will allow the UAV to classify objects correctly for the current task and vice versa Computer Vision depends on Mission Planning to understand the current task to complete.

The software detailed in this document may work similarly to competitors due to the nature of an AUV competition where each robotic submarine must complete the same series of tasks but some AUV's may have an advantage due to their design and software efficiency. The similarities between all the robotic submarines are the ways each submarine completes their tasks, though the methods to reach each task may differ. The motivation behind our design is to complete the required tasks quickly and accurately to gain the most amount of points to maintain a point advantage.

The AUV's software is made up of different departments which include Controls, Mission Planning and Computer Vision. The diagram shown below shows how each group is a component of the overall system.



2.3 Product Functions

The functions of the AUV's software include:

- Detecting and classifying objects within the view of the AUV
  - This will be done using OpenCV for computer vision and YOLOv4 for deep learning algorithms that classify objects found using OpenCV.
- Output function returns the name of the object, bounding box surrounding the object and confidence value for the object's classification.
  - Function will be done using the Darknet framework + YOLOv4, along with mathematical calculations of bounding boxes and object classification.
- AUV sensors send current data to Mission Planning, data includes:
  - AUV's current depth inside the pool
  - AUV's current position relative to an object/task
  - AUV's current rotation and speed when traversing through the pool
  - AUV's current distance from target object for the task

2.4 User Classes and Characteristics

The user classes that are anticipated to use this product are software developers/engineers of future Senior Design Teams as well as RoboSub club members that find a necessity to understand and use Darknet and YOLOv4 to develop software with the ability to detect, recognize and classify objects. Other aspects of the AUV that other users are expected to use are the mission planning software to construct a state machine in which each component of the AUV will be able to pass data back and forth to keep the AUV up to date with its current objective.

2.5 Operating Environment

Environment:

1. Operating System:
    a. Ubuntu 18.04
2. Software:
    a. Darknet/Yolov4
    b. CUDA
    c. OpenLabeling
    d. ROS
3. Hardware: Jetson TX2
    a. GPU: 256-core NVIDIA Pascal™ GPU architecture with 256 NVIDIA CUDA cores
    b. CPU: Dual-Core NVIDIA Denver 2 64-Bit CPU
    c. Quad-Core ARM® Cortex®-A57 MPCore
    d. Memory: 8GB 128-bit LPDDR4 Memory
4. Arduino Mega

2.6 Design and Implementation Constraints

Current Limitations of the AUV include:

- Communication between the groups designing the software of the AUV and the groups designing the hardware of the AUV.
- Simulation of how the software of Computer Vision, Mission Planning and Sensor/Controls will run on the AUV
- Must be efficient as possible with the given hardware to not overwork components

2.7 User Documentation

User documentation will not be provided with the software.

2.8 Assumptions and Dependencies

AUV Competition Assumptions:

- Assumptions that the AUV Competition Rules will remain similar to previous competitions.

- If there are minor rule changes, we will need to update our software and/or hardware to complete the new objectives listed in the updated rules.

Dependencies:

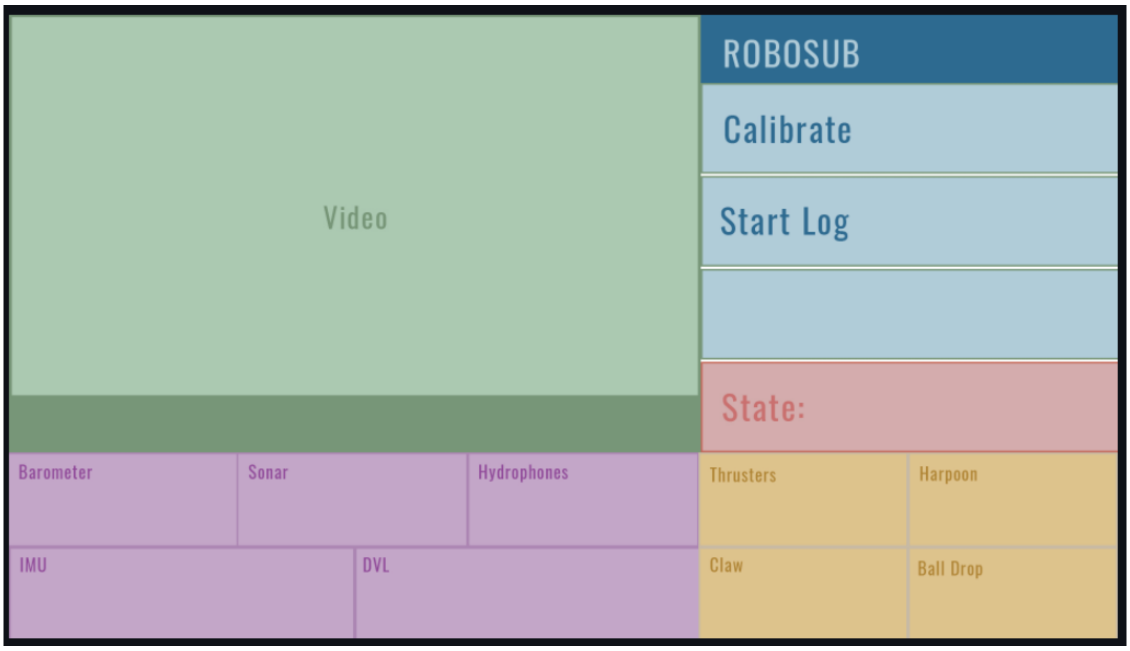
- Reusing software from the previous year as a foundation for our software.

2.9 Apportioning of Requirements

No requirement delays expected.

3. External Interface Requirements

3.1 User Interfaces



The GUI is divided into five sections and that is the general, video, sensor, control and state sections and will be used only for testing purposes. The user interface or any communication with the AUV is not to be used during the competition. The general section will display options to start creating a log of all the data that is to be received and also calibrating. The video section will be used to display what the video camera on the AUV is currently receiving. The sensor section will be used to display the data gathered from the multiple sensors on the AUV. The control section will be used to show the current status of the controls on the AUV. The state section will be used to display the current state that the AUV is currently in.

3.2 Hardware Interfaces

The AUV will have a Arduino Mega 2560 microcontroller for the sensors and actuators to connect to and a Nvidia Jetson TX2 which the camera will connect to. The AUV will also have a thruster board which all 8 thrusters on the AUV will be connected to. Below is a detailed list stating exactly what sensors and controls are connected to which piece of hardware.

- Arduino Mega 2560 Microcontroller
  - Sensors
    - VectorNav IMU (Inertial Measurement Unit)
    - Teledyne Pathfinder DVL (Doppler Velocity Log)
    - Blue Robotics Bar30 Pressure Sensor
    - Blue Robotics Ping Sonar
    - AS-1 Hydrophones
  - Actuators
    - Claw
    - Ball drop

        - Torpedo
- Nvidia Jetson TX2
    - Camera
        - Blue Robotics Low-Light Camera
- Thruster Board
    - Thrusters

3.3 Software Interfaces

For software we will be using Python 2.7 on Ubuntu version 18.04 as our operating system with ROS version Melodic. For computer graphics software, we will be using YOLO and OpenCV for image detection. For mission planning we will be using the ROS -independent Python library SMACH to build our state machines. For the control software we will be using the PID library in the Arduino IDE.

3.4 Communications Interfaces

For communication, the AUV’s sensors will be communicating with the Arduino Mega using rosserial and will also be using rosserial to communicate with the Arduino Mega to receive the data from the sensors.

4. Requirements Specification

4.1 Functional Requirements

The software is to be implemented on an AUV designed by California State University Los Angeles. The software has been split into ROS packages that each have their own functional requirements. There are also some general requirements for the system as a whole.

General

1. The software shall provide an Autonomous system to accomplish tasks set forth by the ROBOSUB competition
2. The software shall provide a controls system for AUV maneuvering
3. The software shall implement object recognition and tracking
4. The software shall interface with onboard sensors
5. The software shall navigate between tasks in a large underwater environment
6. The software shall monitor subsystems and check for possible errors

Mission Planning

1. Defining the current state the AUV is in
2. Taking in data obtained from other components of the AUV
3. Defining when the AUV transitions to another state from its current state
4. Passing data between different states

Controls & Sensors

1. The software shall connect and get data from the IMU
2. The software shall connect and get data from the DVL
3. The software shall connect and get data from the hydrophones
4. The software shall connect and get data from the sonar
5. The software shall connect and get data from the barometer
6. The software shall connect and control the 8 thrusters
7. The software shall connect and control the 2 torpedoes
8. The software shall connect and control the claw
9. The software shall connect and control the ball drop
10. The software shall publish the data from the sensors

Computer Vision

1. 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
   a. 500 - 1000 images per object
   b. Each image in the dataset must be labeled using OpenLabeling
2. The software shall be trained using Darknet and using the YOLOv4 algorithm
   a. Edits must be made to the algorithm variables to be applicable to the need of the dataset that it is given
   b. 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

4.2 External Interface Requirements

-

4.3 Logical Database Requirements

Non applicable.

4.4 Design Constraints

The AUV must have a clearly labeled kill switch that will disconnect the battery from the propulsion devices. This does not necessarily have to cut battery to the computer. When reactivated, the propellers must not spin.

Code written for the AUV must be able to run on an Nvidia Jetson TX2. To get the best performance from the TX2, we will use the latest supported versions of software. It will use Ubuntu 18.04 Bionic Beaver as the operating system. ROS Melodic will be the version of ROS used as the framework for the robotic framework. The version of Python to be used is ver. 2.7.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

The Autonomous Underwater Vehicle (AUV) must weigh less than 125lb (56.7 kg).

The AUV must pass safety inspection by the competition technical staff.

The AUV must work autonomously during the competition runs. There should be no way for communication between the AUV and any person or off-board computer during the competition. All parts of the AUV must be submerged and can not break the surface of the water.

The AUV must be battery powered using sealed batteries so as to reduce hazards from the chemicals in the batteries.

The AUV must not release any objects other than markers, torpedoes, and compressed air.

The AUV must be able to be slung in a harness or sling.

5.2 Safety Requirements

- The AUV must be safe to use and must not cause damage to the pool or divers in the pool.
- It must not hurt anyone or damage anything while on the surface during inspections.
- Projectiles should not pose a threat or cause bodily harm.
- Projectiles should not be pointed at any person.
- Projectiles should not activate unexpectedly.
- When starting up, the propellers must not spin, nor should the missiles, ball dropper, or robotic arm activate.
- After the kill switch is activated, the propellers must not spin.
- The propellers must not spin after reactivated into a safe state after kill switch press.
- The AUV must be able to be harnessed or slung for inspections.
- It must pass inspection by competition inspectors.

5.3 Security Requirements

The code must not be publicly available. It is only to be seen and worked on by authorized RoboSub team members. We must also maintain version control to be able to see how the code is changing and be able to revert to previous versions if needed. Those needs are met by GitHub as the code is stored on a private repository that can only be viewed by authorized accounts.

5.4 Software Quality Attributes

The software must be tuned to the specific attributes listed in the rules to be released by the competition organizers. It should also be well documented enough to give a good foundation for next year's RoboSub team. The software should have a degree of adaptability to adjust to fit the rules once they are officially released. The software should include a test feature to be able to diagnose and assess different components of the AUV.

5.5 Business Rules

Not applicable

6. Legal and Ethical Considerations

In order to comply with the rules of the competition, communication with the AUV during competition is not allowed. Cheating is grounds for disqualification.