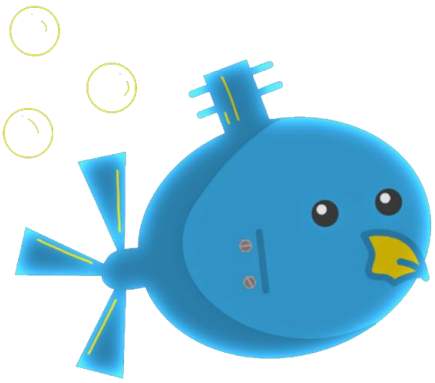
Senior Design Final Report

RoboSub



Version 1.0 - 05/12/2023

Team Members:

Victor Solis

David Camacho

Brandon Cao

Thomas Benson

Andrew Heusser

Bailey Canham

Milca Ucelo-Paiz

Bart Rando

Hector Mora-Silva

Roberto Hernandez

Faculty Advisor:

Richard Cross

Liaisons:

Mark Tufenkjian

Table of Contents

1. Introduction:

1.1. Background:

Our Robosub project is a collobartion by the senior design teams from the Mechanical, Electrical, and Computer Science departments. The objective of the project was to participate in the RoboSub competition organized by RoboNation. Our team utilized various technologies, including ROS2 , OpenCV, Vector NAV Control Center, and ORB-SLAM 3, to develop a software system for an Autonomous Underwater Vehicle (AUV).

The software system was designed to serve specific functionalities while remaining adaptable to different workflows. Its architecture comprised several modules such as sensors, actuators, a Teensy microcontroller, and a Jetson TX2. Each module played a crucial role in the overall coordination and functionality of the AUV, enabling it to execute underwater tasks autonomously.

Throughout the project, our team accomplished significant milestones in key areas such as object detection, data mapping, and autonomous task execution. However, the development of the software system is an ongoing process, and there is still room for future enhancements. We envision that future senior design RoboSub teams will continue to refine the system and expand its capabilities, ensuring its continued growth and success.

1.2. Design Principles:

The RoboSub Software Architecture is built on top of the ROS2 Framework. This is an open source community effort framework that keeps modularity and portability in mind. There is a large amount of community projects that use standards laid out by REP 115 and allow for new packages seemlessly integrate into the ROS2 echosystem. The standards allow for packages to easily interface with other packages without knowing the method of implementation that the pakcage uses. This is a principle that good software follows. ROS2 is a common solution for robotics applications and is the largest open source software that is used.

With this in mind, we have chosen to break up our software system into four main ROS2 packages: a Control package, a computer vision package, a mapping & localization package, and an autonomy package. Each package is designed to handle a desired functionality of the robot. With the help of the ROS2 Frameowrk, we develop interfaces

between the different packages that can allow for communication between them allows each package to get information that helps them accomplish their functionaity.

1.3. Design Benefits:

The benefits of the above principles are allowing our packages to be interchangable. This benefits future developers in this project because they can completely replace a package with a different solution without disrupting the system. As long as the packages use the interfaces, they can change the implementation of the module. This also allows for new packages to be developed and during their development, other modules do not have

1.4. Achievements:

The RoboSub Senior Design team has acheive advancements in the software system. New implementaiton for packages have been used to increase the usefulness of each package and increase the quality of sensor data that is used in our system. Also, a completely new software package, the mapping & localization package has been mapped out for future developers to improve.

2. Related Technologies:

2.1. Existing Solutions:

We have looked into existing tools and libraries for creating a autonomous robot software system. The software systems are designed together such that the RoboSub is able to accomplish or complete underwater tasks.

OpenCV (Open Source Computer Vision Library) is a powerful open-source computer vision and machine learning library that offers various tools and functions to help process video data. OpenCV is utilised for tasks including object detection and recognition and tracking movements enabling the sub to navigate and avoid obstacles and complete its tasks in a challenging underwater environment. LabelIMG is an open source library that was used to assist in the labeling of images. The YOLO (You Only Look Once) algorithm and a Darknet [Neural Network] were used to aid object detection.

Vector NAV Control Center was utilised to communicate with the VN-100 IMU and confirm the status of the IMU. With the Control Center we were able to test the IMU and confirm the custom made cable works as well as how to parse the data the system was reciving.

For the mapping and localization subsystems open source library ORB-SLAM 3 was used. This library had several pre-requisites. Pangolin is an open source library used to aid with mapping the environment. Eigen is an open source library used to help with the linear algebra components of positioning the robot and the environment. DBow2 is another open source library for aiding with the image data and g2o is another open source library to aid with the optimization.

Autonomy uses ROS2 and Behavioral Trees which a library designed to be flexible, easy to use and fast. It consists of Trees of nodes which are a set of building blocks and are "assembled" to build behaviors .

2.2. Reused Products:

The complete software system is developed using ROS 2 as a framework. The ROS 2 platform provides facilities such as sensor access, communication middleware, camera access, network access, and visualization tools. The ROS 2-based framework allows efficient message passing and distributed computing for various operations on data, such as filtering sensor inputs and controlling the AUV's motion.

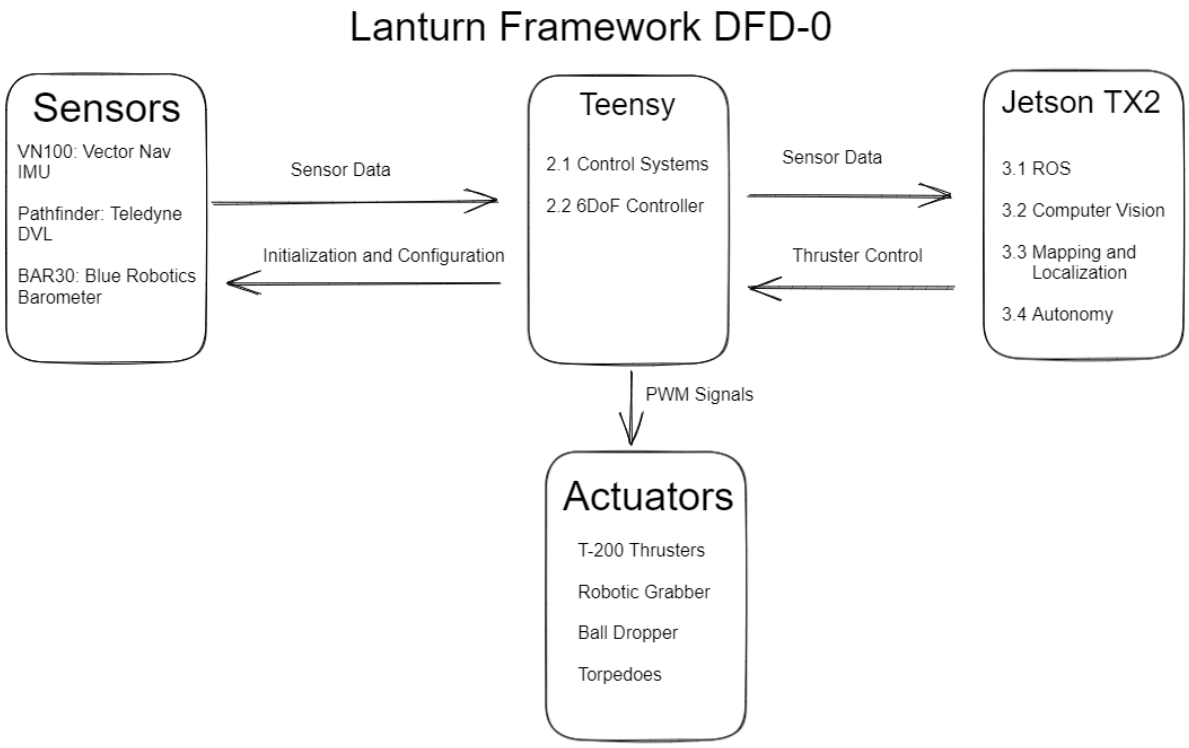
ROS 2 allows for the integration of various sensors, actuators, and control algorithms. The modular design simplifies the development process

3. System architecture:

3.1. Overview:

The architecture for the microcontrollers on the robot and all sensors and accuators, and can be broken down into four main factors: Sensors, accuators, the Teensy microcontroller and the Jetson TX2.

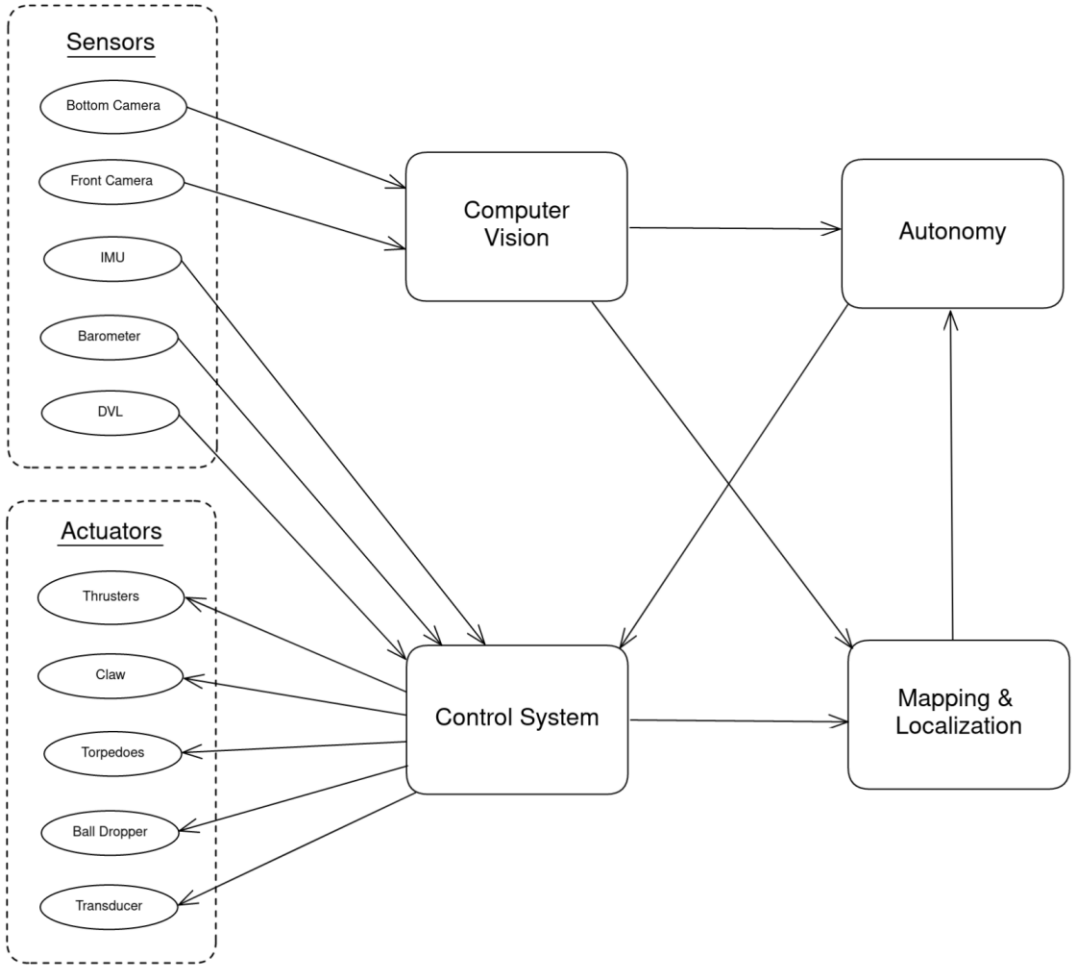
Here is a diagram (DFD level 0) that shows how this architecture works at a high level:



- **The Sensors:** The Sensors are responsible for getting all the data of the robot, using all this sensor data eventually the robot will know where its current location is as well as the loction of the tasks it wishes to complete are. Without sensors the robot has no idea of its current location
- **The Teensy Board:** A secondary microcontroller, that acts as the hardware inferface, for all of the sensors and acuators, and the ROS operating system on the Jetson TX2. The codebase for the 6DoF controller also reside on this board
- **The Actuators:** Through PWM siginals from the teensy, the thrusters, grabber, ball dropper, and torpedoes can call be controlled in ths system.
- **The Jetson TX2:** The primary microcontroller board. Has Unbutu 20 and ROS installed upon it. The codes bases for computer vision, mapping and localization, and autonomy all reside on this board

3.2. Data Flow:

Here is an overview of the Framework as a system, and how it connects to the Robot. This is also our DFD level 1:



There are six major modules in this system. They are described in more details in section 6. Here is a brief overview of them:

3.2.1. Sensors: The Sensors are responsible for getting all the data of the robot, using all this sensor data eventually the robot will know where its current location is as well as the loction of the tasks it wishes to complete are. Without sensors the robot has no idea of its current location

3.2.2. Computer Vision: Computer Vision uses the camera sensors to identify objects and determine the distance away from those indentified objects. Once an object has been identified it will be labled and sent to Autonomy to figure out what to do with the object

3.2.3. Mapping and Localization: Mapping and Localization takes the sensor data sent to it by controls and uses this data to create a map of the current location its in. Using this map it is able to determine its current location and the location of other objects in the vecinity

3.2.4. Autonomy: Autonomy is the main logic of the robot it takes all the information sent to it from Computer Vision and from Mapping and Localization, it then uses all this information to make discision about where the robot wishes to go. As well as what tasks the robot is moving to and what will need to be done to complete the task. Once it has created the task the robot wants to do it will send that information to controls.

3.2.5. Control Systems: Controls is the low level communication with the sensors, it is responsible for actualizing and initalizing all the sensors and then sending that data out to Mappign and Localization to be processed. Once all the data has been processed, Autonomy will send a comannd to controls telling what task the robot is supposed to preform next. Once controls has this task it will determine what Actuators need to be used to preform this task and will send the information to the Actuators.

3.2.6. Actuators: The Actuators are the indidual motors which allow the robot to move and preform tasks. Controls will send information to each indivdual actuator allowing the desired servo to move. Use the combination of servos controls has directed the actuators to use, the robot moves in the desired direction or preformes the disered task.

3.3. Implementation:

The project was split into four sections to allow for efficient development: Control Systems, Computer Vision, Mapping and Localization, and autonomy. Each section plays a key role in presenting the progression of the project.

3.3.1. Control Systems

The control system sub team is responsible for implementing and aquiring data from sensors, implementing PWM signal commands to control accutators, as well as to implement the six degrees of freedom controller.

3.3.2. Computer Vision

The Computer Vision sub team is to recognize specific images, such as badges or dollar signs, in underwater environments and then process them through a machine learning model and send the relevant data to Autonomy/Mission Planning.

3.3.3. Mapping and Localization

The Mapping and Localization sub team is to gather data from the sensors such as barometer to create a virutal map, then send relevant data and virtual map to Autonomy/Mission Planning.

3.3.4. Autonomy

The Autonomy/Mission Planning sub team is to cover the SMACH model developed to control the AUV subsystems through each task at the RoboNation Competition.

4. Conclusions:

4.1. Results:

Our team was able to create a practical and effective software system that can be implemented within the AUV that was built by our partner ME/EE Senior Design Team. This will allow the CSULA AUV to be able to participate in the 2023 Robonation RoboSub Competition. Our software was created to spcifically accomplish desired competition tasks to earn the most amount of points possible.

The Computer Vision sub-team was in charge of training the objects and was able to create an object detection and line detection model for two different competition tasks that are going to used for the prequalification part of the competition.

The Controls sub-team was in charge of translating data from the sensors into thruster commands, and utilized the VectorNav VN-100 and a custom library in order to do this.

The Mapping and Localization sub-team was in charge of combining camera and IMU data, and was able to use Orb Slam 3 to help the AUV develop a map of it's surroundings and save it to a atlas.

The Autonomy sub-team was in charge of implmenting and designing the software structure for the AUV, and was able to use behavioral trees to direct the AUV to the next logical competition task, while consistently performing system and safety checks.

Finally, we utilized a program called ROS2 to corrdinate and communicate between the software built by the different sub-teams.

4.2. Future:

While we have heavily expanded upon the AUV's capabilities through our software system, it is by no means perfect. There is still work to be done, and we hope that future senior design RoboSub team's will be able to build upon our code and documentation to design a better system. Additionally, the system we created can be utilized in more ways then just within a competition AUV, such as within the fields of oceanic research, artifact discovery, and military applications. However, there is still much to do before that point, and here are some following suggested improvements for future RoboSub teams:

- Test the AUV in the water to see its accuracy based on the current implementations done from each sub-team.
- Computer Vision: Train and test a new system for the actual competiiton rounds, since the current system is only trained for prequalification round objects.

- Controls System: Add sonar, hydrophone, and torpedo capabilites.
- Mapping & Localization: Further test ROS publisher and subscriber topics.
- Autonomy: Figure out a simulation solution (Ex. Unity or MATLAB) to expand goal behavior.

5. References:

1. ROS: "Wiki." *Ros.org*, http://wiki.ros.org/
2. BehaviorTree.CPP: *BehaviorTree.CPP*. www.behaviortree.dev.
3. Groot: *Groot | BehaviorTree.CPP*. www.behaviortree.dev/groot.
4. YOLOv4: "Yolov4 Tiny Object Detection Model." *YOLOv4 Tiny Object Detection Model*, Roboflow Inc. , https://roboflow.com/model/yolov4-tiny.
5. Darknet: Bochkovskiy, Alexey. "Home · Alexeyab/Darknet Wiki." *GitHub*, GitHub, Inc., https://github.com/AlexeyAB/darknet/wiki.
6. OpenCV Download: Linuxize. "How to Install Opencv on Ubuntu 20.04." *Linuxize*, Linuxize, 5 July 2020, https://linuxize.com/post/how-to-install-opencv-on-ubuntu-20-04/.
7. OpenCV: doxygen. "OpenCV Modules." *OpenCV*, OpenCV, https://docs.opencv.org/4.x/.
8. Google CoLab: Google. "Colaboratory." *Google Colab*, Google, https://research.google.com/colaboratory/faq.html.
9. LabelIMG: Heartexlabs. (n.d.). *Heartexlabs/labelimg: LabelImg is now part of the label Studio Community. the popular image annotation tool created by Tzutalin is no longer actively being developed, but you can check out label studio, the open source data labeling tool for images, text, hypertext, audio, video and time-series data.* GitHub. Retrieved December 9, 2022, from https://github.com/heartexlabs/labelImg
10. VN-100: VectorNav "VN-100 User Manual." V*ectorNav, https://www.vectornav.com/resources/user-manuals/vn-100-user-manual*