



*Cornell University
Autonomous Underwater Vehicle*

Spring 2017

Webserver

A screenshot of the CU AUV webserver interface. The top navigation bar includes 'CU AUV' with a yellow dot, and menu items 'Index', 'Drive', 'Test', and 'SHM'. The main content area is titled 'Status of artemis @ teag'. On the right, there are several data panels: a vertical list of coordinates (D, H, L, X, Y) all at 0.00; a DVL status panel with 'DVL ALTD: 0.00', 'DVL TEMP: 0.00', and four beams (1-4) with 'FWRD' and 'SWAY' values; a control panel with 'HK' (SK), 'PC', 'OT', and 'EN'; a 'roll' panel with 'DES: 0.00', 'VAL: 0.00', 'RTE: 0.00', and 'ON' (OFF); and a 'depth' panel with 'DES: 0.00', 'VAL: 0.00', 'P: 7.00', and 'I: 0.30'. On the left, there are tables for 'Depth' (0), 'Pressure' (0), and 'Thrusters' (a table with columns 'Thrusters' and 'Status' and rows for 'aft_port', 'aft_starboard', and 'fore_port').

Semester Review

Noel Picinich (nmp53)

May 13, 2017

Contents

1	Abstract	2
2	Goals	3
3	Previous Implementations	3
4	Design: Why React?	4
4.1	Speed	4
4.2	Modularity	4
4.3	Scalability	4
5	Implementation	5
5.1	Components	5
5.2	Props and State	5
5.3	Handlers	5
5.4	Style	6
5.5	Keyboard Commands	6
6	Results	7
7	Future Improvements	8
7.1	Additional Features	8
7.2	Suggested Changes from Survey Responses	8

1 Abstract

A frequent obstacle encountered by CUAUV electrical and mechanical sub-team members is the need to control the sub or gain access to its various statuses but not being able to do so without the help of a software member. Our current system for driving the sub and performing tests involves the knowledge and execution of terminal commands that are often too involved and unfamiliar for non-software members to execute. Last semester, Angela Yang (aqy2) and Danny Qiu (dq29) began the implementation of a unified webserver that will combine the features of past webserver implementations and create a lasting resource for the team as a whole.

This semester I joined Angela and Danny in further developing this new webserver by using React that will solve the aforementioned obstacles and overcome complexity with extremely coherent and user-friendly features. Due to my limited knowledge of React, the bulk of my semester was dedicated to learning React and researching its most effective styles of implementation. This documentation will focus on the understanding I have gained and the React features I was able to implement.

PORT: 0	DES HEAD: 0.00	HEAD: 0.00	DVL ALTD: 0.00	HK
STAR: 0	DES DPTH: 0.00	DPTH: 0.00	DVL TEMP: 0.00	SK
FORE: 0: 0	DES PTCH: 0.00	PTCH: 0.00	DVL BEAM 1 FWRD:	
AFT: 0: 0	DES ROLL: 0.00	ROLL: 0.00	DVL BEAM 2 0.0	PC
SFOR: 0	DES VELX: 0.00	VELX: 0.00	DVL BEAM 3 SWAY:	OT
SAFT: 0	DES VELY: 0.00	VELY: 0.00	DVL BEAM 4 0.0	EN
heading				
DES: 0.00	P: 0.70			
VAL: 0.00	I: 0.00			
OUT: 0.00	D: 0.20			
RTE: 0.00	IG: 0.00			
ON	<input type="checkbox"/> OFF	RD: 20.00		
pitch				
DES: 0.00	P: 0.50			
VAL: 0.00	I: 0.00			
OUT: 0.00	D: 0.10			
RTE: 0.00	IG: 0.00			
ON	<input type="checkbox"/> OFF	RD: 50.00		
roll				
DES: 0.00	P: 0.50			
VAL: 0.00	I: 0.00			
OUT: 0.00	D: 0.15			
RTE: 0.00	IG: 0.00			
ON	<input type="checkbox"/> OFF	RD: 50.00		
velx				
DES: 0.00	P: 5.00			
VAL: 0.00	I: 0.00			
OUT: 0.00	D: 0.50			
RTE: 0.00	IG: 0.00			
ON	<input type="checkbox"/> OFF	RD: 0.15		
vely				
DES: 0.00	P: 5.00			
VAL: 0.00	I: 0.00			
OUT: 0.00	D: 0.50			
RTE: 0.00	IG: 0.00			
ON	<input type="checkbox"/> OFF	RD: 0.50		
depth				
DES: 0.00	P: 7.00			
VAL: 0.00	I: 0.00			
OUT: 0.00	D: 3.20			
RTE: 0.00	IG: 0.00			
ON	<input type="checkbox"/> OFF	RD: 0.50		
Not on vehicle. Monitoring disabled.				

Figure 1: Current control helm that software members use to control the sub and access its various statuses.

2 Goals

The CUAUV WebsERVER is a new team-wide applicable tool that aims to fulfill the following goals:

- Be a user-friendly resource for all subteam members to perform tasks including:
 - Accessing the primary statuses of the sub, that have previously been accessed solely through the control helm (see Figure 1).
 - Driving the sub through GUI components and through keyboard commands that parallel current control helm keyboard commands (e.g. Forward, Increase Depth, Zero, Soft-kill, etc.).
 - Running tests on the sub, including Syscheck, Thruster Tests, and Actuator Tests.
 - Viewing the values of all AUV Shared Memory System (SHM) variables.
- Attain the simplicity and coherence to be fully adopted by non-software members for long-term use.
- Be a lasting resource for CUAUV that will be continually maintained and improved to grow with the increasing functionality of each year's subs and the expanding software stack. This will entail semesterly design reviews and updates that will take into account functionality input from across subteams.

3 Previous Implementations

“This web GUI has been in the works by various members over the past few years, with each iteration bringing a different flavor of a websERVER. However, with recent modifications of the code base, multiple versions of websERVERs have popped up, each with a specialized usage. A legacy websERVER still exists in the code base as well as a minimal websERVER built by Zander Bolgar (asb322) last year. There also exists a vision websERVER used to run the vision GUI. This project aims to bring all the websERVERs together, so that only a single websERVER needs to be run to provide all the necessary functionality.”
- Danny Qiu (dq29), Fall 2016 Unified WebsERVER Documentation

4 Design: Why React?

The first task we took on this semester was learning how to implement the JavaScript library `React.js`. This new library, developed by Facebook engineers, is becoming increasingly popular for its functional and reactive power. We chose to incorporate React for its speed, modularity, and scalability.

4.1 Speed

An important feature of the webserver, is its ability to update the subs' numerous statuses in real time. Without React, DOM manipulations are not very fast. The DOM, Document Object Model, is a representation of a webpage that can be changed using a scripting language; we use JavaScript. While we can't speed up DOM manipulation itself, we can use React to only update aspects of the DOM that actually change. To do this, React uses what is called the virtual DOM, a virtual representation of every DOM object that is much faster to update. Now, when React updates the DOM it first updates the entire virtual DOM, compares the virtual DOM to the previous version of itself and then updates only the changed aspects in the real DOM. This process is what makes React faster than other frameworks and is what will allow the webserver to update more quickly when there are only minimal changes in the extensive number of statuses and variables.

4.2 Modularity

React code is composed of components: small, reusable sections of code. React's component based structure makes it easy to reuse code by simply referencing a component that's already been created. Not only does this improve development efficiency, but also maintainability: updating a single root component updates every file that uses it!

4.3 Scalability

React implements a heuristic, constant time algorithm for updating the DOM and changing states that maintains its time efficiency even as an application grows. This will be even more applicable to our webserver as advancements in the sub lead to more variables and more data.

5 Implementation

This section will describe the unique process of implementing components in React. I will use the implementation of a SHM variable search bar to explain these basic steps.

5.1 Components

As mentioned previously, React is composed of components. These components are essentially JavaScript functions that take in parameters called `props` and return elements to be displayed by the `render` method. The `render` method tells React what you want rendered and React manipulates the DOM to reflect this description. In developing the websERVER we use JSX, a JavaScript syntax extension, that allows us to easily write React elements for React to render. React elements are JavaScript objects that can be stored as variables in JSX. Furthermore, any piece of JavaScript code may be inserted into JSX by simply enclosing it in curly braces. The first step in implementing the SHM search bar was to add a search bar element to the `render` function of the exported class `SHM`.

5.2 Props and State

Although React components take in `props`, these `props` should never be manipulated by the component. Since components cannot change their inputs (`props`) they use the concept of state to manipulate their output. The second step in adding the search bar was to add a `state` variable in the `SHM` class constructor to store the initial value of the search. This `state` variable is then passed, as a `prop` called `query`, into the `SHMGroupList` component that is called by the `SHM` class. The `SHMGroupList` class then accesses this value by referencing `this.props.query`. Since the render function of `SHMGroupList` generates which SHM groups are rendered, I added a conditional using JavaScript that filters out SHM group names not containing a sub-string of the query entered.

5.3 Handlers

In order for the rendered list of SHM groups to change when the user types a query, our React code must use an event handler. An event handler is often a function within a class that handles changes in the application's state. In this case, I implemented a `handleQueryUpdate` function that takes in the

parameter `e`, for event. This handler will update the `state` variable `query` to be the value inputted by the user into the search bar.

5.4 Style

This final step, unrelated to React, entails manipulating the style of the new search bar to reflect the rest of the webserver. I completed this step by adding the Bootstrap class `form-control` to the `div` search component in the `shm.jsx` file and then updating the `cuauv.css` file for additional style improvements. Figure 2 illustrates the final result.

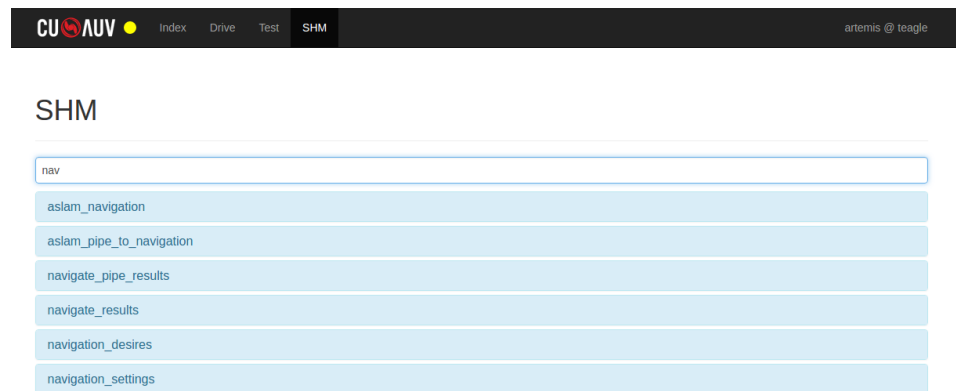


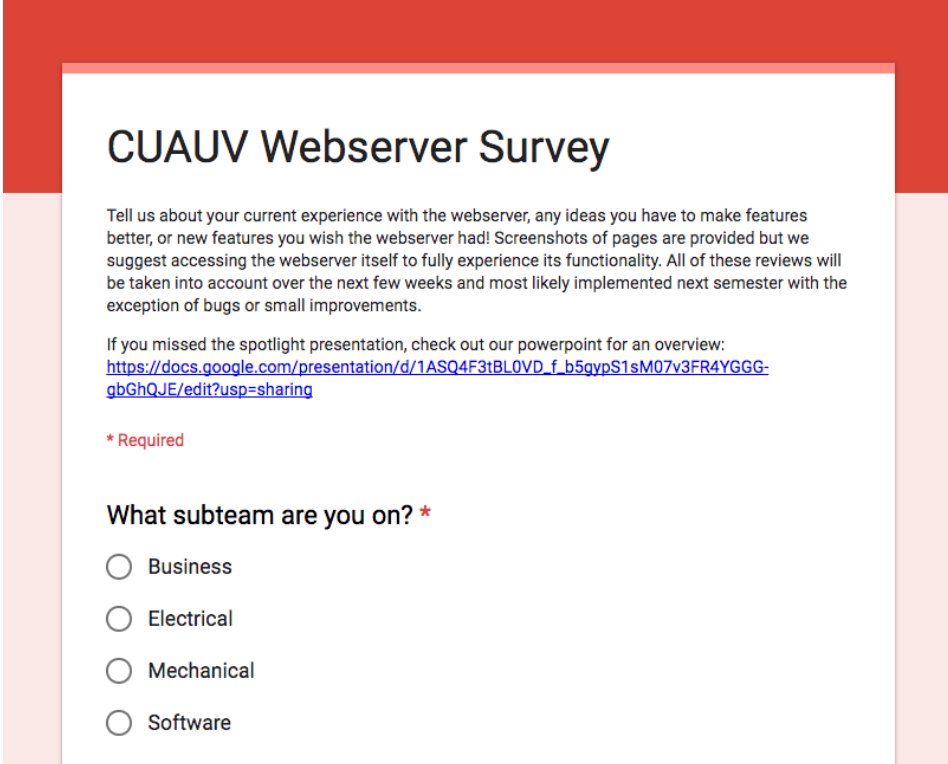
Figure 2: Screenshot of the current Search bar filtering based on the query “nav”

5.5 Keyboard Commands

Since the webserver was meant to be a tool for all team members, we wanted software members to retain the familiar `control-helm` capabilities of driving the sub through keyboard commands. This will also be a more intuitive feature for other subteam members to become accustomed to. These keyboard shortcuts are keys on the computer that mirror buttons on the webserver drive page. This feature was implemented using `keycode` values obtained from implementing the JavaScript `keydown` library in the button component classes.

6 Results

So far the webserver has been a success! In our recent presentation to the entire team we received positive feedback about the project and our execution seems to be at a sufficiently comprehensible level. Since the webserver was just released for team-wide usage, the true test of its success will be this coming summer when we hope the other subteams will fully adopt and utilize all of its features. In addition to our recent presentation where we accepted preliminary feedback, I also created a Google form, depicted in Figure 3, to act as a long term resource for all CUAUV members to give Danny, Angela, and I feedback, requests, and new ideas for future advancements in the webserver.



The image shows a screenshot of a Google Form titled "CUAUV Webserver Survey". The form is set against a white background with a red border. The title is in a large, bold, black font. Below the title, there is a paragraph of text explaining the purpose of the survey. A link is provided for those who missed a presentation. A red asterisk indicates a required question. The question is "What subteam are you on?" with four radio button options: Business, Electrical, Mechanical, and Software.

CUAUV Webserver Survey

Tell us about your current experience with the webserver, any ideas you have to make features better, or new features you wish the webserver had! Screenshots of pages are provided but we suggest accessing the webserver itself to fully experience its functionality. All of these reviews will be taken into account over the next few weeks and most likely implemented next semester with the exception of bugs or small improvements.

If you missed the spotlight presentation, check out our powerpoint for an overview:
https://docs.google.com/presentation/d/1ASQ4F3tBLQVD_f_b5gypS1sM07v3FR4YGGG-gbGhQJE/edit?usp=sharing

*** Required**

What subteam are you on? *

- Business
- Electrical
- Mechanical
- Software

Figure 3: Screenshot of the Google form

7 Future Improvements

The following improvements are focused on advancing front-end functionality. See the Spring 2017 Documentation of Danny Qiu (dq29) for additional information about future back-end improvements.

7.1 Additional Features

- Fully incorporated Vision GUI that will allow the user to view output from the subs' cameras as they drive
- Mobile touch version of the Drive page where the user can control the sub through intuitive touch manipulations.
- The ability to change SHM variables directly on the webserver SHM page: currently they are available for view only. In addition, the user should be able to search for subsections of SHM groups.

7.2 Suggested Changes from Survey Responses

- Move the **Zero Desires** button from its location between the “Left” and “Right” navigation buttons, in order to reduce risk of accidental clicking.
- A key for keyboard shortcuts, although software members are very familiar with these, other subteam members need a resource to get accustomed to driving the sub in this efficient manner.
- A status feature that shows the number of other team members that are currently accessing the webserver, specifically to avoid conflicting demands if multiple members try to drive the sub at one time.