# Gonzaga University Robotics Club: Robosub 2019

Tyler Willis, Liam Jones, Aaron Wong, Daniel Mobley, Navath Nhan, Aaron Weber,
Brandon Takahashi, Kevin Mattappally, Eric Av, Anna Smith, Ben Howard, Marissa Encarnacion,
Michael Tome, Jasmine Fischer, Blaine Atkins, Emily Ellewin,

***Abstract*—Last year, for the first time in club history, GU Robotics attended the international Robosub competition where we exceeded our own expectations and learned where to find room for improvement. This year, club activities attempted to address the system limitations we came up against last year and lay a foundation for future advancements. Our work this year adds functionality to the computer control systems, including computer vision, and creates a platform to be used in the future with minimal modifications.**

**This paper is a documentation of the engineering and technical work we accomplished on the mechanical, electrical, and computer science systems as we try to increase capability without sacrificing reliability. This paper should explain what we did and why we did it, and serve to familiarize someone with the technical and organizational aspects of GU Robotics 2019 Robosub competition entrant.**

## I. Competition Strategy

Our competition strategy is reliability through simplicity. Our team is still working on the body of knowledge necessary for object identification, vehicle tracking, and mission control, so focusing any team's effort on advanced systems like grippers or launchers would be working in vain. Our testing this year focused on getting the most valuable data from the computer vision system, integrating that with data from the IMU, and creating the autonomous mission control scheme from that. Our testing focused on the integration of the computer vision system, under the belief that reliable computer vision would be the most bang for our buck in terms of navigation. Last year, we scored our points by having our diver point the sub in the right direction and moving in a reliable curve. This year, we hope to be able to use the compass to orient ourselves, use the vision system to pass through the gate on the small side, and as a stretch goal, we'd like to bump into the buoys. Going beyond that requires either a DVL or an echolocation system, neither of which we have the financial or personal resources to acquire or integrate. Someday, we hope to add those to our platform, which would enable us to make a good faith attempt at the higher level course challenges.

## II. Vehicle Design

### A. Mechanical

Our 2019 development work was heavily influenced by lessons we learned at competition in 2018. Our previous strengths were robustness, ease of control, and in-water stability, while our weaknesses were the time it took to open the hull, difficulty in arranging and loading the electronic components, and limited room for upgraded computer systems. We introduced a new hull to optimize electronics accessibility, we created a custom electronics tray for compact arrangement, and we changed the frame design to support balance adjustments. We did all this with an eye towards modularity for future expansion.

*1) Hull:* We replaced the tube design of last year with an off-the-shelf underwater enclosure from Polycase, mounting the electronics to a custom tray and routing the wires through the lid. Initially, we used a 10x10x12 box, but testing revealed we could fit all our components in an 8x8x10 box and score bonus points by minimizing ballast weight. The box is opaque polycarbonate, but the lid is clear so we can see our indicator LEDs and inspect for water leaks. Inside the hull, we designed a custom electronics tray that holds all the electronics and the batteries, allows for easy removal and component access, and also serves as a structural brace, protecting the sides of the box against deflection from water pressure. We used an on-campus laser cutter to create the interlocking acrylic parts, and tolerances are within .05 mm of what we expected. Since we routed all wires through the lid, we designed and 3D printed custom brackets that allow the lid to clip on the side of the box while we work on the internal electronics, and version two will optimize the shape to be epoxied onto the lid. If future developments require more space, we could switch back to the larger box and re-cut the electronics tray with new proportions, and be running within a day.

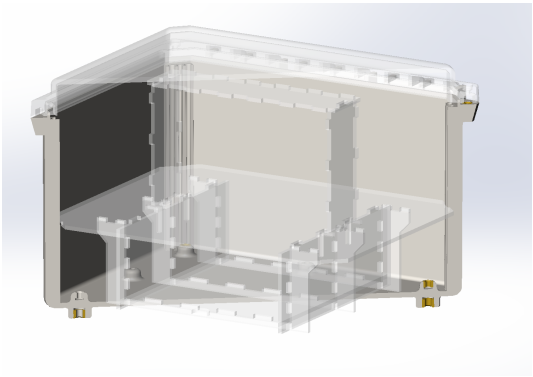

Fig. 1. Custom Electronics tray in the box

*2) Frame:* In a change from last years all t-slot aluminum construction, we used plasma cut aluminum panels on the side to allow for precise positioning of motors and the hull, as well as providing extra protection for components if the sub bumps into walls. These were modified based on test experience to minimize weight and maximize flexibility of mount locations. Standardized hole sizes work with the t-slot

mounting brackets, and custom 3D printed motor mounts allow for near-universal motor positioning.

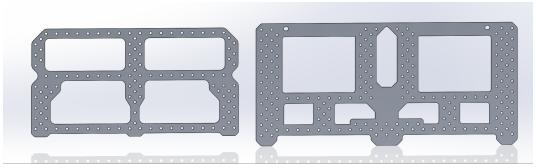

Fig. 2. Side Panels V1 (left) and V2 (right)

*3) Camera Mount:* In the creation of the computer vision system, rather than use two cameras to acheive our desired field of view (forward and down), we would use one camera on a pivot mount. Once we selected a waterproof servo, our team member Michael Tome designed brackets to attach the rotating camera box, the servo, and the frame. Designed for 3D printing, these brackets fit all the components surprisingly well, and accounted for the offset pivot point of the servo. In version two, we improved the accessibility of the mounting screws and reduced high stress points in the brackets.

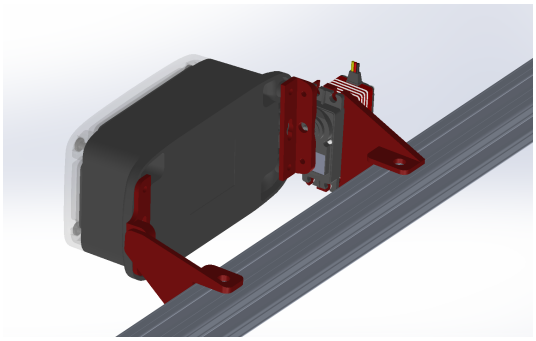

Fig. 3. Camera brackets for 3D printing

*B.*

Electronics Systems

*1) Backplane:* Most electrical components interface with the Texas Instruments microcontroller board. The 2018 submarine used a perfboard backplane to facilitate these connections. In the interest of a more robust, professional backplane for the 2019 sub, we designed a PCB using Eagle software from Autodesk and created it using the milling machine provided by Gonzaga. The microcontroller slots in to the backplane which connects it to headers for indicator lights, voltage sensing, current sensing, leak sensing, camera servo and light PWM outputs, ESC signal outputs, a ground equalizer, auxiliary 5V high-amperage input, and an I2C bus for connecting the depth sensor, IMU and further expansion.

*2) Leak Sensor:* This year we transitioned to a new hull with our batteries on the bottom of the hull. As such, it was difficult to access the state of the battery compartment and the hull's reliability was an unknown. Any leaks in the battery compartment would have been a critical issue. With the new consolidated hull design, we can more reliable verify the state

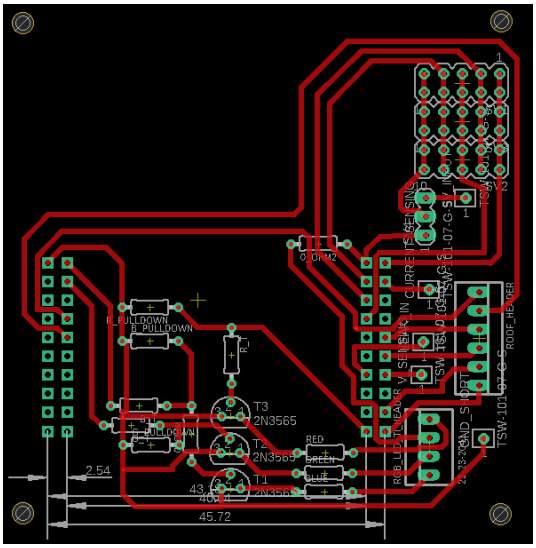


Fig. 4. Backplane wiring diagram

Fig. 5. Leak Sensor board top

Fig. 6. Leak Sensor board bottom

of the submarine. To assist in locating leaks, we designed a leak sensor board which was created on our milling machine. The board has two independent probe headers to allow for easier identification of leak location. The board has two LED indicators, as well as a signal header for communication with the microcontroller which are both activated when current is detected across probe leads, indicating the presence of water.

*3) Kill Switch:* For the ability to cut power to the submarine from outside the hull, a magnetic hall effect sensor is placed near the hull wall with a magnet on the outside. Removal of the magnet opens the hall effect sensor, which kills power to the submarine. The hall effect sensor switches a transistor which is used to switch a relay. In previous years this disconnected

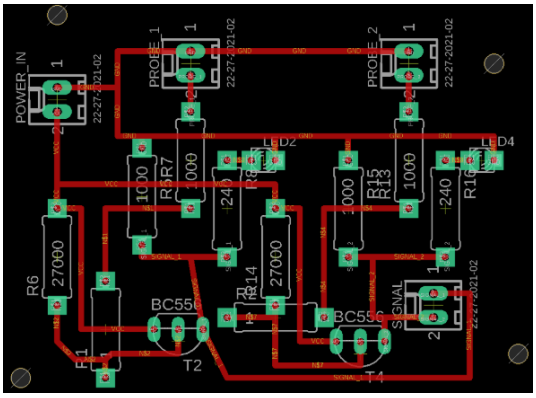


Fig. 7. Leak Sensor wiring diagram

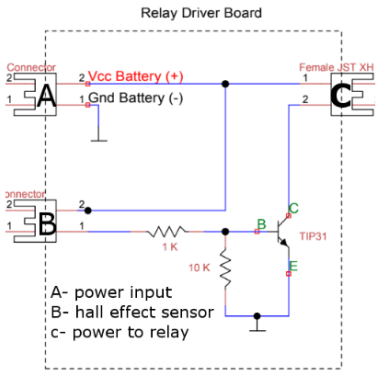


Fig. 8. Relay Driver Board Wiring Diagram

power to the whole submarine. This year with the addition of an RFID reader to reset the TI microcontroller, we have reconfigured the kill switch to only cut power to the thrusters, allowing us to cease sub operation while not hard resetting the mission computer. This gives us a quicker turn-around from resetting our system.

*4) Electronics layout:* The increase of electrical components made efficient, effective and functional layout important. The IMU (which is mainly used as a compass for now) was placed in the corner, away from noisy components to minimize interference. Positioning it in different locations and spinning the sub revealed this to be one of the few locations where it remained functional.

To save space on the electronics tray, the ESCs were attached to the underside of the lid. This also minimized the number of wires that needed to travel between the penetrators on the lid and the tray.

To minimize the number of cameras needed, the camera is mounted in an external box attached to a servo which can rotate the camera up and down. When this functionality is enabled in software, it will allow the same camera to serve the purpose of following lines on the floor, locating targets ahead, and finding an area to surface.

### C. Software

*1) Mission Computer:* The Mission Control software team focuses on controlling the actions of the sub and communication between the different systems running on the sub. The center of the mission control is a Java program that is executed on an NVIDIA Jetson TX2. This mission control program communicates to the TM4C123GH6PM microcontroller to control motors and receive sensor data and a Python program to interpret camera data using OpenCV. The microcontroller communications are done through UART, and the Python communications are done through a UDP server. In both cases, the Java program sends data through a communication protocol using a set of enums called SendTypes and ReceiveTypes. These enums are part of a system we developed that sends a specific character id for what type of data is being sent (such as a PWM value for a specific motor or a desired depth value) along with the data. Whenever a SendType is sent the data being sent and the timestamp of when it was sent are recorded, so that they can be viewed after testing to find what commands were sent at what time.The microcontroller and Python send data to the Java program in the same format, and the Java program has a list of ReceiveTypes that tell what type of data the character id corresponds to, and these are recorded in the same manner as the SendTypes.

It was determined that it would be best to have a Mission Computer running Linux on board to handle all autonomy related decisions, interface with the image recognition camera and communicate with the Microcontroller with easy-to-use functional control commands via a custom UART serial protocol. All Mission Computer code is written in Java, and all communications with the Microcontroller and camera are abstracted away for ease of use. The Computer follows sequential step by step procedures to complete each task, while making real-time decisions that allow the Computer to decide when and how to perform each tasks. There is also a graphical user interface that it is used for testing and debugging and was made with the idea of keeping debugging time to a minimum. OpenCV for python was used for interfacing with our camera.

*2) Mission Control:* The mission control program is capable of parsing and running mission scripts written in JSON. These scripts in JSON contain a series of steps and actions that the submarine will take, given the right condition is met. This script can be best thought of as a linked-list where the mission computer only traverses to the next node when all of its exit conditions have been met. This mission script allows us to quickly modify the behavior of our submarine while allowing it to autonomously execute a set of instructions. Mission scripts consist of a set of nodes, with each node having actions and exit conditions. Actions are values that are sent to the microcontroller or Python program, such as motor PWM values or setting and enabling a PID loop. Exit conditions are the conditions that must be met before the mission can move on to the next node, which can include simple conditions like a certain amount of time elapsing or more complex conditions such as holding a certain depth or

heading for a period of time.

### D. Embedded Systems

The goal of the Embedded Systems team is to provide an interface for our Mission Computer to communicate with our motors, sensors, etc. To do this, we have a Texas Instrument microcontroller that uses protocols like I2C, UART and PWM to communicate with the sensors and motors while providing feedback to the Mission Computer.

*1) Microcontroller Unit:* The microcontroller used is the TM4C123GH6PM. This unit was chosen for its widespread support as well as flexibility in functionality. These qualities made it possible to prototype and develop functionalities in a timely manner. The microcontrollers functionalities were accessed through the widely supported TivaWare drivers. The drivers made it easy to use the various peripherals provided without extensive knowledge in the microcontrollers architecture. The flexibility of the Nested Vectored Interrupt Controller allowed for a responsive system. The use of interrupts provide an illusion of concurrency which is a key component of the embedded system.

*2) Control Loops:* The interrupt service routines provide 3 main control loops. The main function loop, the UART receiving interrupt service routine, and the real time interrupt service routine. The UART receiving interrupt service routine is triggered when a character is received on the UART channel. The main function loop controls prototyping and specific function testing, while the real time interrupt service routine executes the PID control loop that alters motor values to achieve the given set point.

The UART interrupt service routine appends the received characters onto a global string. When the interrupt service routine has received our predetermined ”end of transmission” character, it will then proceed to process the string it has received. Each of these strings will be 6 bytes long, 1 byte as an identifier character, 4 bytes as a standard IEEE 752 floating-point number and the final byte as our predetermined ”end of transmission” character, the ” ”. The identifier character appended to the start of the string will signify what the subsequent floating value represents. Different identifiers have been selected to represent different values such as desired depth, desired heading or desired forward thrust. A special identifier character ”*” signifies that the subsequent characters will represent a debugging string that the main function loop will handle. Upon receiving the special identifier character, a flag will be raised to signify for the main function loop to execute its debugging scripts. Upon receiving a general identifier with a floating point value, the UART interrupt service routine will store the received float in its appropriate variables and change the appropriate flags to signal a new setpoint has been received.

The main function loop waits on the ”foundEOT” flag. This flag is raised when a special identifier character * has been received. The main function will then parse through the received string which often contains a debugging request like ”pssr” which requests for the microcontroller to test the pressure sensor, or ”mtr” which requests a motor test. This main function loop is used primarily for debugging and prototyping purposes.

The real time interrupt service routine waits its timer to expire before triggering. It is set to trigger once every 100ms. The real time interrupt service routine will run through a PID calculation, taking in the current sensor data and comparing it to the desired sensor data. It will then compute the error between the two values and the 3 corresponding proportional, integral and derivative values. These values are recombined to give a motor output value that will be used to bring the submarine closer to the set point. This loop will run continuously to bring the submarine to its set point and hold its set point. Due to the nature of the PID algorithm, the computation has to be computed at very specific intervals for the output value to be meaningful. The use of a real time interrupt service routine is crucial to maintaining a consistent sample time. The real time interrupt service routine will then execute a set of triggered actions before sending all internally stored variables to the Mission Computer to be synchronized.

*3) Sensors:* The TM4C123GH6PM also interfaces with a large portion of the sensors on the submarine. These include the depth sensor, accelerometer, gyroscope and magnetometer. The MPU9150 is used to provide acceleration, gyroscopic and magnetic heading data while the MS5837 is used to provide pressure data. The MPU9150 is housed as a sensor boosterpack for the TM4C123GH6PM while the MS5837 is housed as the Bar30 as provided by BlueRobotics. The microcontroller interfaces with the MPU9150 through I2C and transforms the data through a Madgwick filter to compute the submarines current magnetic heading in degrees.

*4) RFID Inputs:* Our team worked on a wireless method to send signals to the submarine while in untethered. We added an RFID sensor which communicates with our Mission Control software through UART. This RFID sensor was specifically chosen to for its low frequency of 125kHz, allowing it to communicate even through water. The RFID sensor actively reads RFID tags that come within 5cm. If the RFID tag’s identification digit matches the stored values within our Mission Control, it triggers a programmable action. This method is used to trigger the shut-down of the microcontroller or the start of various mission scripts. The RFID sensor provides a unique and flexible way of communicating with our submarine when untethered.

### E. Machine Learning

We began to implement Machine Learning into the submarine to better seek out competition challenges and improve object detection beyond mere color recognition. We manually collected and labelled images from previous competition videos available online to train the model to detect the gate. Using the NVIDIA Digits software, we were able to create a deep neural network for the purpose of object detection using the DetectNet framework and a KITTI dataset. From there, we used our completed model to pretrain another model, and increase the accuracy of our predictions.

*1) Labels:* Our team created a custom labelling software in C# using Windows Presentation Foundation to manually label our collected images. Several versions of this program were created, with features added to ensure that the several parameters to use the Digits software, namely uniformity in the image sizes and meeting the minimum bounding box requirement of 60 x 60 pixels. The program collected the relevant data points on the selected region, and transcribed the information into a separate label document. We collected such information as X, Y, and Z. We then distributed the images into two separate data sets, the training and the validation set. Approximately ten percent of our collected data, and their corresponding labels, were placed into our validation set, which were chosen to represent a diverse range of angles and situations, rather than selected randomly.

*2) Models:* The Digits software then trains a model through 500 epochs using our given training set and the corresponding labels, and tests for the accuracy, precision, mAP, and the loss of each value against the validation set. A graph is then created for each value before the next epoch begins. The network then continues testing the weights randomly until the end of the 500 epochs.

A pretrained model is created using the last most successful model at the epoch of the highest reported mAP value, and is then trained with the assigned weights for another 500 epochs, allowing the neural network to refine its assigned weights. The process is only repeated twice to avoid overfitting the network to the data set.

## III. Experimental Results

Early in the year, we created a small robotic car that works off of the same Nvidia Jetson and motor controllers as the sub, as a platform for testing and debugging the computer vision system without the logistical burden of a pool test. Using this platform, we were able to integrate the computer vision system with the heading control system, allowing for pool tests to focus on integration rather than debugging. We saved many man hours of prep, transport, and waiting, and the car serves as a more engaging demonstration for prospective students than the sub could.

On the mechanical side, testing proved that the original 10”x10”x12” box was not only too big, but also required an absurd amount of ballast. We switched to a smaller model of the same box, and tests showed that it is secure down to 18 feet, without the reinforcing electronics tray. We learned that the acrylic used in the electronics tray is brittle and will crack under repeated stresses, so techniques to minimize stress concentration factors were used. Ideally, the material would be switched to polycarbonate for its decreased brittleness, unfortunately it is not able to be laser cut. We also modified the design of the side panels to minimize interference with the y-axis thrusters and allow for maximum flexibility with mounting components for balance and drag reduction.

## Acknowledgments

We’d like to thank Patrick Nowacki for 4 years of being our club advisor, hearing our crazy ideas, and pointing us towards something that will actually work instead. We are grateful to Dr. Timothy Fitzgerald for taking on this role moving forward, and we’re excited for his guidance. Huge thanks to Dr. Brian and Donna Jones for allowing us to take over their home for a week during competition and for giving us food on top of that. Thanks to all the young students on the team, whose energy and passion to know more and do more drives this team, we’re excited to see what you do with the club. Thanks to the students of the other Robosub teams for your knowledge, camaraderie, and freely given help and advice.

## APPENDIX A: SCORING EXPECTATIONS .



### *Appendix A: Expectations*

Below is the scoring table showing the points associated with each task. Enter the points you **expect to score** with the vehicle(s) that you have designed and engineered. At the end of the competition, enter the points you **actually scored** in the last column.

| Subjective Measures | | | |
|---|---|---|---|
| | Maximum Points | Expected Points | Points Scored |
| Utility of team website | 50 | 35 | |
| Technical Merit (from journal paper) | 150 | 100 | |
| Written Style (from journal paper) | 50 | 40 | |
| Capability for Autonomous Behavior (static judging) | 100 | 50 | |
| Creativity in System Design (static judging) | 100 | 60 | |
| Team Uniform (static judging) | 10 | 9 | |
| Team Video | 50 | 35 | |
| Pre-Qualifying Video | 100 | 0 | |
| Discretionary points (static judging) | 40 | 20 | |
| Total | 650 | 349 | |
| | | | |
| **Performance Measures** | | | |
| | Maximum Points | | |
| Weight | See Table 1 / Vehicle | 98.5 | |
| Marker/Torpedo over weight or size by <10% | minus 500 / marker | -0 | |
| Gate: Pass through | 100 | 100 | |
| Gate: Maintain fixed heading | 150 | 150 | |
| Gate: Coin Flip | 300 | 300 | |
| Gate: Pass through 60% section | 200 | 0 | |
| Gate: Pass through 40% section | 400 | 400 | |
| Gate: Style | +100 (8x max) | 100 | |
| Collect Pickup: Crucifix, Garlic | 400 / object | | |
| Follow the "Path" (2 total) | 100 / segment | 100 | |
| Slay Vampires: Any, Called | 300, 600 | 300 | |
| Drop Garlic: Open, Closed | 700, 1000 / marker (2 + pickup) | | |
| Drop Garlic: Move Arm | 400 | | |
| Stake through Heart: Open Oval, Cover Oval, Sm Heart | 800, 1000, 1200 / torpedo (max 2) | | |
| Stake through Heart: Move lever | 400 | | |
| Stake through Heart: Bonus - Cover Oval, Sm Heart | 500 | | |
| Expose to Sunlight: Surface in Area | 1000 | | |
| Expose to Sunlight: Surface with object | 400 / object | | |
| Expose to Sunlight: Open coffin | 400 | | |
| Expose to Sunlight: Drop Pickup | 200 / object (Crucifix only) | | |
| Random Pinger first task | 500 | | |
| Random Pinger second task | 1500 | | |
| Inter-vehicle Communication | 1000 | | |
| Finish the mission with T minutes (whole + factional) | Tx100 | | |
| | | 1897.5 | |

Fig. 9. expected point values

## APPENDIX B:COMPONENT LIST

| A | B | C | D | E |
|---|---|---|---|---|
| **Component** | **Vendor** | **Model/Type** | **Specs** | **Cost (if new)** |
| Buyancy Control | N/A | | | |
| Frame | 8020 | 1x1 in. T slot | 20 feet, cut to length | $60 |
| Waterproof housing | Polycase | YQ-100806 | 10x8x8 inches | $69 |
| Thrusters | Blue Robotics | T200 | | $169x6 |
| Motor Control | Texas Instruments | TM4C123GH6PM | | $16 |
| High Level Control | Nvidia | Jetson | | |
| Actuators | N/A | | | |
| Propellers | Blue Robotics | Propellor Set | | $5 |
| Battery | Turnigy | 4S1P 14.8V 20C Hardcase Pack | | $32.80x6 |
| Converter | | | | |
| Regulator | | | | |
| CPU | Nvidia | Jetson | | $400 |
| Internal Comm Network | | | | |
| External Comm Network | | | | |
| Programming Language 1 | Java | | | |
| Programming Language 2 | C++ | | | |
| Programming Language 3 | Python | | | |
| Compass | Texas Instruments | MPU9150 (SensorHub BoosterPack) | | $40 |
| Inertial Measurment Unit | Texas Instruments | MPU9150 (SensorHub BoosterPack) | | $40 |
| Doppler Velocity Log | N/A | | | |
| Camera | ELP | ELP-USB500W02M-L36 | 3.6mm fixed lens | $52 |
| Hydrophones | N/A | | | |
| Manipulator | N/A | | | |
| Algorithms: Vision | Open-Source | OpenCV | | Free |
| Algorithms: Acustics | N/A | | | |
| Algorithm: Localization and mapping | N/A | | | |
| Algorithms: Autonomy | Texas Instruments | TivaWare | | |
| Open Soure Software | OpenCV | | | |
| Team Size | 24 | | | |
| HW/SW Expertise Ratio | 11:13 | | | |
| Testing Time: Simulation | 30 hours | | | |
| Testing Time: In Water | 5x 3 hour days | | | |

Fig. 10. Component List