Umm Al-Qura University

Collage of Computer and Information Systems

Computer Engineering Department

Graduation Project Report

BSc Project
CE Department
Project ID: UQU-CE–14-204

Red Sea Coral Reef Monitoring and Underwater Sensing using OpenROV

Abdul Rahman Jan **42906066**

Saed Al Sarhani **429014922**

Ahmed Al Zahrani **43002746**

Hilal Al Qurashi **42906013**

Department of Computer Engineering

Faculty of Computer and Information Systems

Umm Al-Qura University, KSA

Contact Information

This project report is submitted to the Department of Computer Engineering at Umm Al-Qura University in partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Engineering.

Author(s):

Name: Abdul Rahman Jan
ID: 42906066
Email: mehmoodjan89@yahoo.com

Name: Saed Al Sarhani
ID: 429014922
Email: alsarhani19827@gmail.com

Name: Ahmed Al Zahrani
ID: 43002746
Email: ahalzahrani@gmail.com

Name: Hilal Al Qurashi
ID: 42906013
Email: hilalqrji@gmail.com

University supervisor(s):

Emad Felembad

Dept. of Computer Engineering
Faculty of Computer and Information Systems
Umm Al Qura University
Internet: http:// www.emadfelemban.net/
Phone: +966 56787263
Kingdom of Saudi Arabia

Intellectual Property Right Declaration

This is to declare that the work under the supervision of _____ having title "_____" carried out in partial fulfillment of the requirements of Bachelor of Science in _____, is the sole property of the Umm Al-Qura University and the respective supervisor and is protected under the intellectual property right laws and conventions. It can only be considered/ used for purposes like extension for further enhancement, product development, adoption for commercial/organizational usage, etc., with the permission of the University and respective supervisor.

Date: _____

Author(s):

Name: Abdul Rahman Jan *Signature: _____*

Name: Saed Al Sarhani *Signature: _____*

Name: Ahmed Al Zahrani *Signature: _____*

Name: Hilal Al Qurashi *Signature: _____*

Supervisor(s):

Name: Emad Felemban *Signature: _____*

Anti-Plagiarism Declaration

This is to declare that the above publication produced under the supervision of _____ having title "_____ _____" is the sole contribution of the author(s) and no part hereof has been reproduced illegally (cut and paste) which can be considered as Plagiarism. All referenced parts have been used to argue the idea and have been cited properly. I/We will be responsible and liable for any consequence if violation of this declaration is proven.

Date: _____

Author(s):

Name: Abdul Rahman Jan *Signature:* _____

Name: Saed Al Sarhani *Signature:* _____

Name: Ahmed Al Zahrani *Signature:* _____

Name: Hilal Al Qurashi *Signature:* _____

Acknowledgement

First of all, praises for Allah, in guiding us to the right way and always blessing us with his mercy.

We owe many thanks to the people who helped and supported us throughout the course of our project. We pay our regards to our families for their prayers and great support and the push to achieve success, they were very patient while we were always busy. We are heartily thankful to our supervisor Dr. EmadFelemban for the guidance and support from initial to the final level that enabled us to develop an understanding of the project.

A special thank goes to "Engr. Mohsin Murad" for advising us in the project and for sharing useful information and experiences. Again we say Alhamdulillah, for giving us the strength and health to do this project work until it was done.

Abstract

Oceans and marine life are an important aspect of our environment as they cover 73% of earth's surface. While being a source of huge volume of food and other resources for human beings, the oceans also help maintain our eco-system. Red Sea with its vast coral reefs is a very rich environment for marine life, allowing it to be destination for many divers and tourist activities. However with the rapid changes in the climatic conditions and the increase in water pollution, these coral reefs are depleting fast. In this work we present an underwater monitoring and observation system based on an open-source underwater submarine to observe the coral reefs via a high definition camera as well as with the help of multiple sensors to observe critical underwater climatic parameters like light intensity and temperature.

Keywords: OpenROV, Coral Reef Monitoring, Single Board Computers

Table of Contents

List of Figures

List of Tables

CHAPTER 1

INTRODUCTION

1. Introduction

In this chapter we will discuss the purpose of project, the purpose of the document in hand and some literature review in the form of existing systems.

1.1. Purpose of the Project

As we are seeing in the modern age of technical developments in the field of scientific and discover modern methods to saving time and effort and enables us to get to the massive information in easy and fast ways we know the marine environment is Important source life and we should protect this environment.

Coral reef is the center of the huge amounts of biological diversity, and major player in the survival of the entire ecosystem. So it provides various marine animals with food, shelter and protection, which keeps generations of kinds alive

Moreover, coral reefs are considered an integral part of maintaining human life by being a source of food (such as fish and shellfish, etc.), as well as being a space marine eco-tourism, which provides economic benefits

Coral reefs face increasing risks due to environmental pollution and diseases that afflict and destructive fishing practices and warming of the oceans. We create a thought to use robot it can helps researchers in marine environment to overcome the challenges they may face during their work with help of using OpenROV robot and some of extra development. The extra sensing capabilities that we add includes the addition of sensing capabilities by integrating temperature, depth and light sensors with the OpenROV (an open-source remotely operated vehicle).

Possible applications of our sensing capable underwater robot are:

- Measure temperature, depth, infrared and light.
- Monitor marine protected areas
- Detect new life under sea.
- Detect unauthorized Navy submarines and sea mines
- Monitoring of marine reserves.

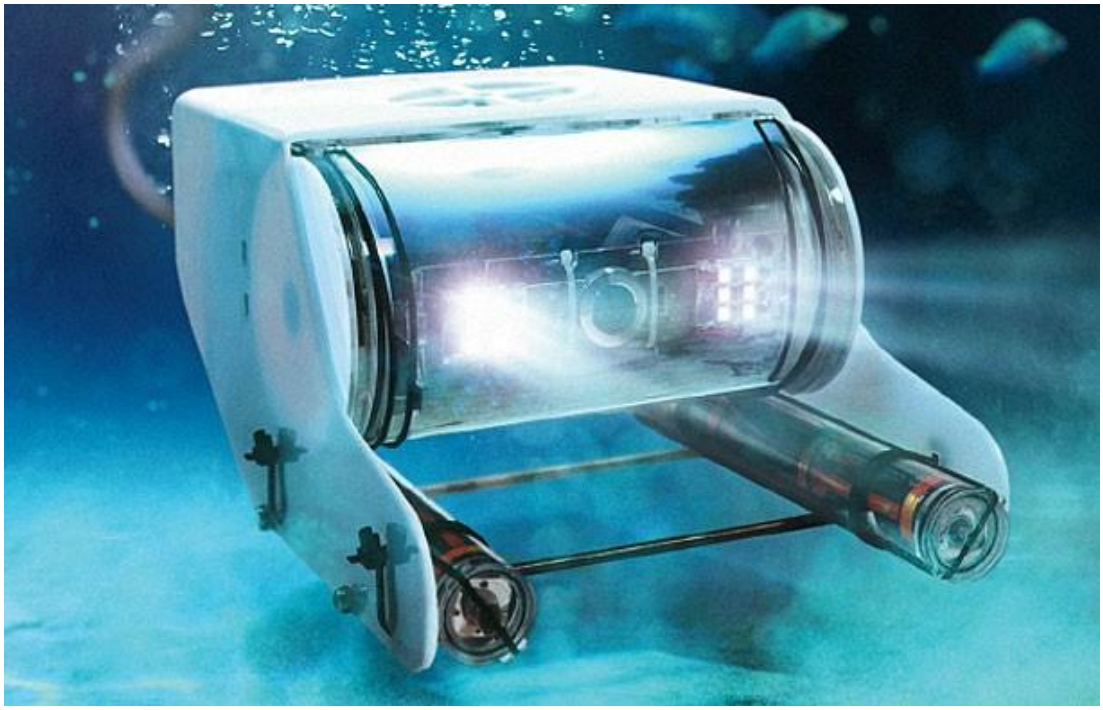

FIGURE 1.1 (OPENROV SUBMARINE)

1.2. Purpose and Overview of This Document

In this document we propose a system based on an open-source underwater submarine to sense some vital environmental parameters to observe the red sea's rich coral reef environment. The document will explain in detail the steps required in setting up the ROV and integrating a sensor board that will enable it to measure various climatic variables.

The document will also take the reader through programming the BeagleBone Black single board computer that is the heart of our ROV and to program the Arduino based Controller Board. Then we will modify the robot to include a temperature sensor to measure water temperature, light intensity, depth and infra-red radiations around the coral reefs.

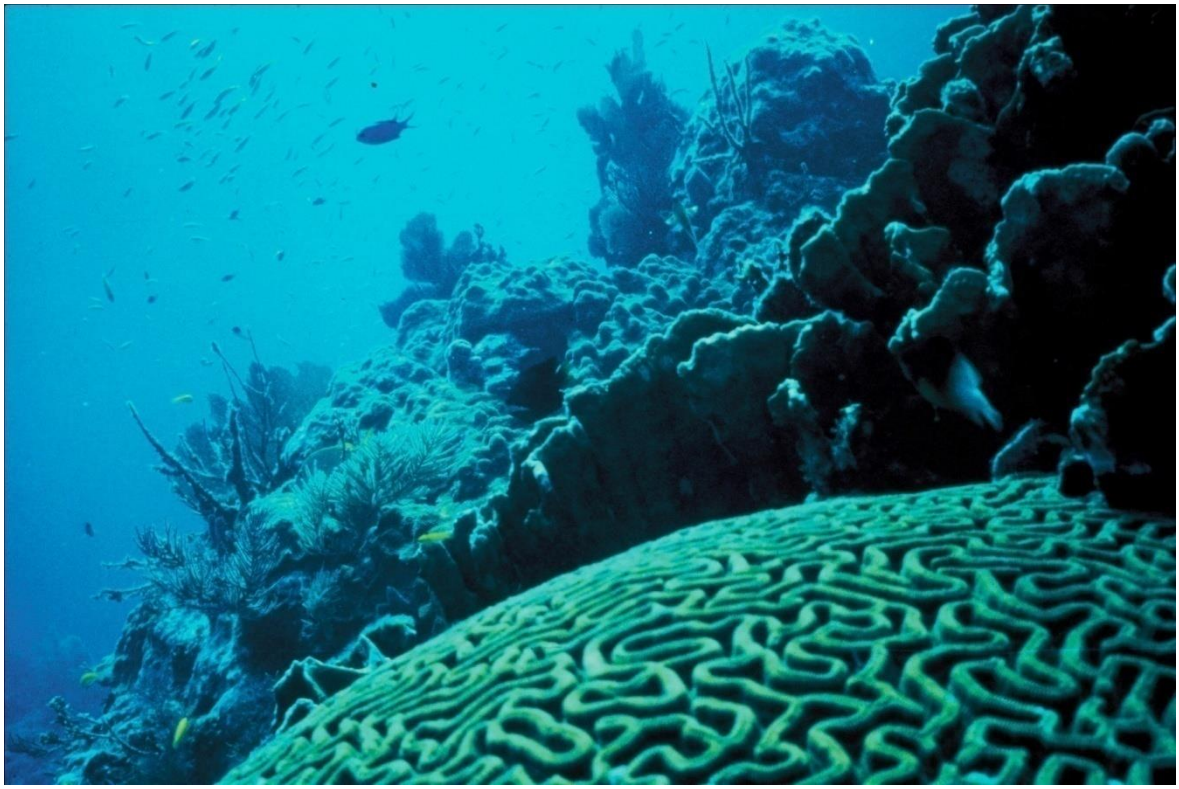

FIGURE 1.2 (A LOOK AT RED SEA'S RICH CORAL REEF)

OpenROV is a OpenSource underwater vehicle, remotely operated from a standard browser via a Node.js application running on an embedded Linux board (the BeagleBone) and connected to a webcam and an Arduino. In this write up we are going to tell you some insights on how the ROV is put together, mainly focusing on the software and electronic components, followed by deployment in Red Sea and small pools for proof of concept of the system that is designed.

1.3. Existing Systems

1.3.1. Existing System Descriptions

Traditional methods using divers to monitor coral reefs and risk variables and record and the circumstances surrounding marine life.

But with the passage of time and the evolution of technology in our modern age has become such a solution not feasible for the following reasons:

1. Exposure to risks divers in the sea such as drowning and attacking predatory fish and sudden change in depth.
2. Lack of effectiveness of the registration data and access to researchers and specialists.

FIGURE 1.3 (A DIVER IS OBSERVING MARINE LIFE AT CORAL REEF)

1.3.2. Problems in the Existing Systems

The current systems in place pose great threat to human lives and the limitation in diving capabilities make it difficult to stay submerged for a longer period of time. Remote monitoring via ROVs is the only viable option to monitor the mysterious life underwater but the huge costs associated with such systems makes them harder for the academia and common man to acquire.

FIGURE 1.4 (SHARK ATTACKS ARE COMMON PROBLEM IN MANNED OBSERVATORY MISSIONS)

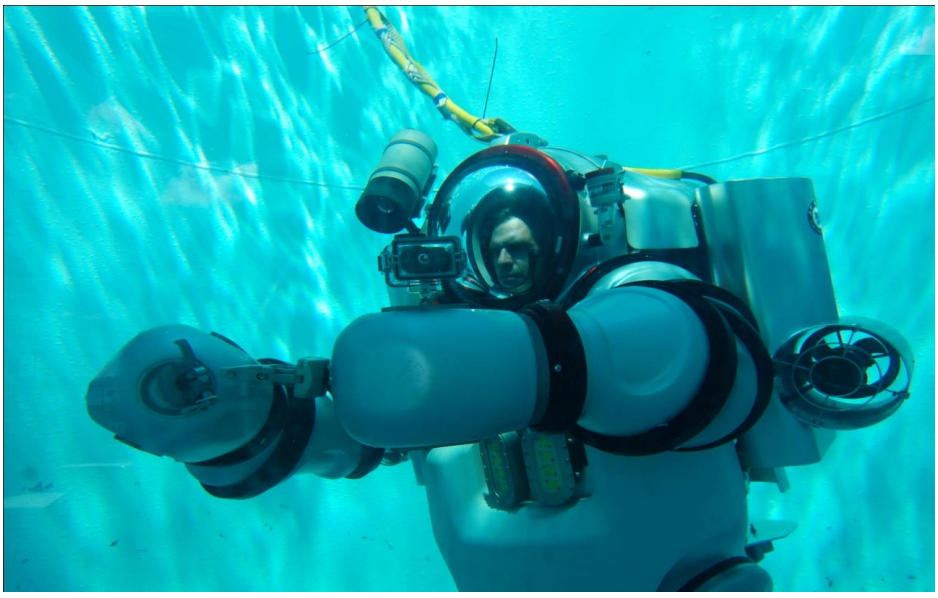

FIGURE 1.5 (DEEP SEA UNIFORM MAKES IT HARD FOR PEOPLE TO NAVIGATE AND HAS A LIMITED OXYGEN SUPPLY)

CHAPTER 2

SYSTEM ANALYSIS

2. System Analysis

In this chapter, the document explains in detail all the data analysis and pre-project data and requirement gathering techniques and tools used.

2.1. Data Analysis

A diagram of the data flow is called tools, a style of modeling Important analysis and building information can be noted that a major user who logs on to the system and displays the main picture and temperatures and audible beeps and and lighting visible and invisible, and is considered the .... heart structure and ..... Control between the network and the Internet and the computer connects to the web cam to record the sights and connects to sensors also arrive energy batteries are the first and the second is the power supply

2.1.1. Data Flow Diagrams

The ROV side of the vehicle consists of a BeagleBone Black single board computer running a flavor of embedded linux stripped of several modules for efficient functioning over a small embedded computer. The beaglebone black is hosting a Node.Js server allowing the user various controls of the ROV via access through an IP address. An Arduino Mega 2560 based controller board provides connectivity to various ESCs running the Camera Tilt Servo, LEDs and the Thurster Motors. Beaglebone Black is directly connected to the controller board and relays information to the motors and servos via arduino's interfaces.

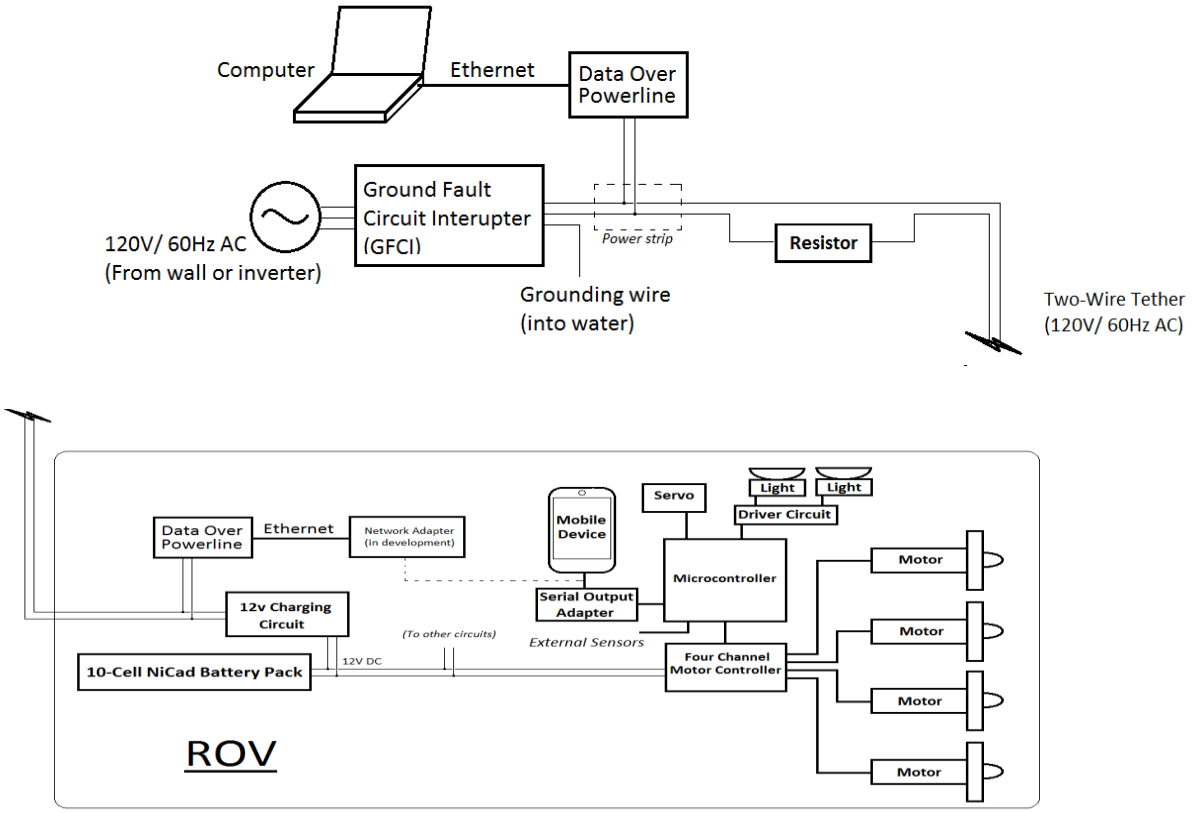

FIGURE 2.1 (SYSTEM FLOW DIAGRAM)

The data is received from a user side PC via a two wire Power Over Ethernet adapter. It converts the data coming from the PC's RJ45 ethernet module into two wire interface and at the ROV side converts it back into Ethernet signals to be supplied to the Beaglebone Black's ethernet controller.

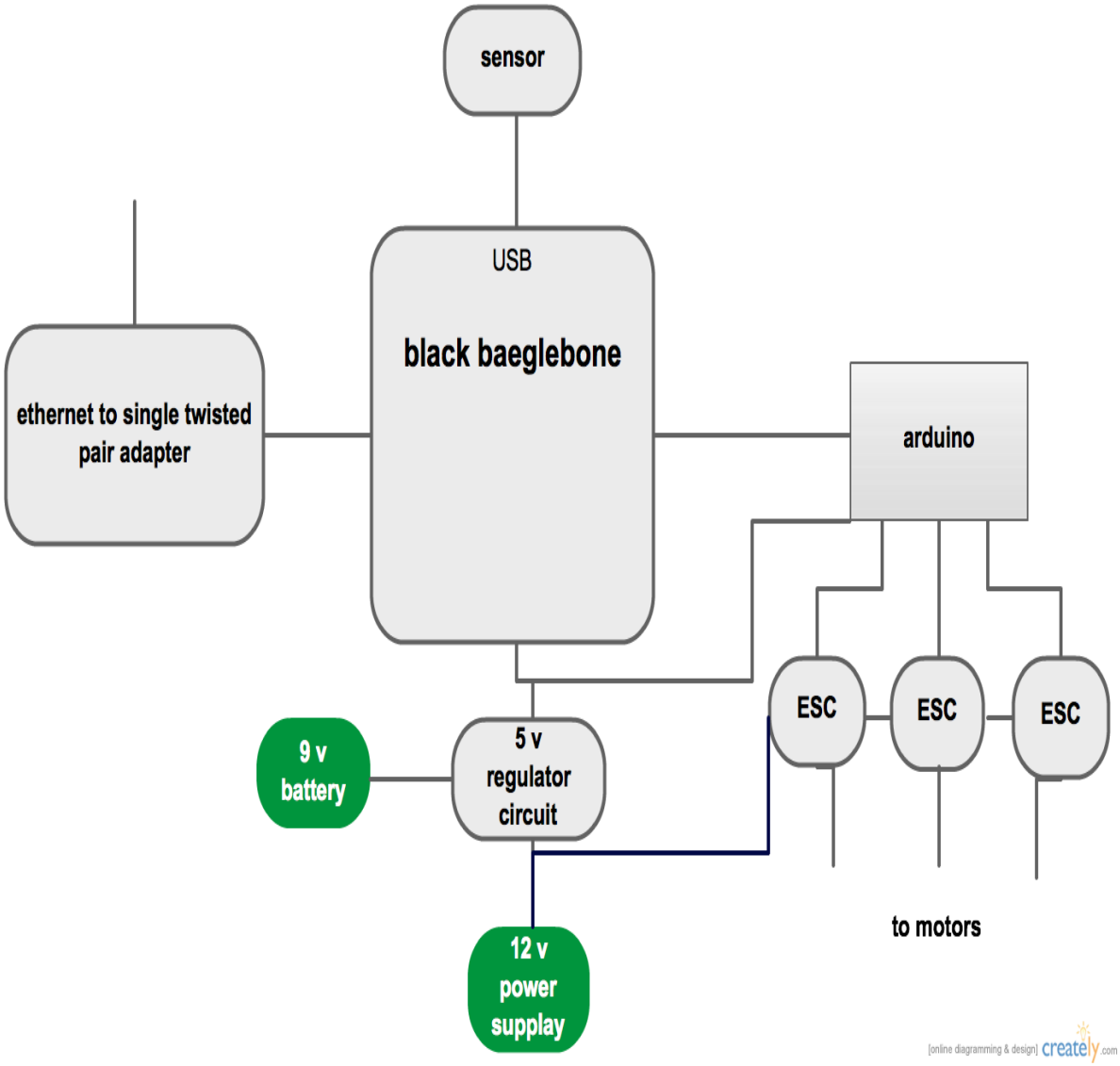

FIGURE 2.2 (BLOCK DIAGRAM OF OPENROV SUBSYSTEMS)

A USB based HD Web-camera is connected to the Beagle Bone Black USB controller and allows the user to take real time video and audio streams for recording as well as navigational purposes. A sensor board is mounted on top of the ROV that is capable of measuring real time temperature, light intensity and infra-red values.

At the PC side the system is connected via an Ethernet cable after the power over Ethernet module converts the two wire data to Ethernet. The server hosted on BeagleBone Black is accessed via its IP Address though a browser can commands can be passed to control various functionalities of the ROV and real time data can be visualized.

2.1.2. System Requirements

2.1.2.1. Clients, Customers and Users

- Operators: Responsible for operating the navigational software to drive the OpenROV. They will also be responsible to extracting the sensory data.
- Students/Scientists/Researchers: Analyses the sensory data as well as the Video stream for scientific evaluation.
- Divers: Uses the data from the real time imagery to know if it is safe for diving at that particular place or not.
- Developers: Responsible for development and porting software firmware and future modifications to the already developed system.

CHAPTER 3

DESIGN CONSTRAINTS

3.1. Design Constraints

In this chapter we discuss the basic design constraints, the hardware and software environments and the initial prototyping.

3.1.1. Hardware Environment

OpenROV is a remotely operated mini-submarine that weighs ~2.5 kg and has dimensions 15 cm x 20 cm x 30 cm. This submarine is powered by C batteries and can be assembled from common materials, with the most expensive piece being the BeagleBone Linux computer. The submarine is controlled from a laptop computer connected to the submarine via a tether and is equipped with on-board LEDs and a camera. OpenROV is an open-source hardware project. By providing the list of the submarine parts and instructions on how to assemble them, the developers aim to democratize underwater exploration.

3.1.1.1. OpenROV Controller

The version 2.5OpenROV Controller Board complements the Beaglebone Black as the brains of the OpenROV. It allows the high-level computing power of the Beaglebone Black to control the sensors and actuators of the ROV, such as the motors, servos, and lights. This Controller Board performs the same functions as the earlier OpenROV Cape, while adding a spectrum of new functionality.

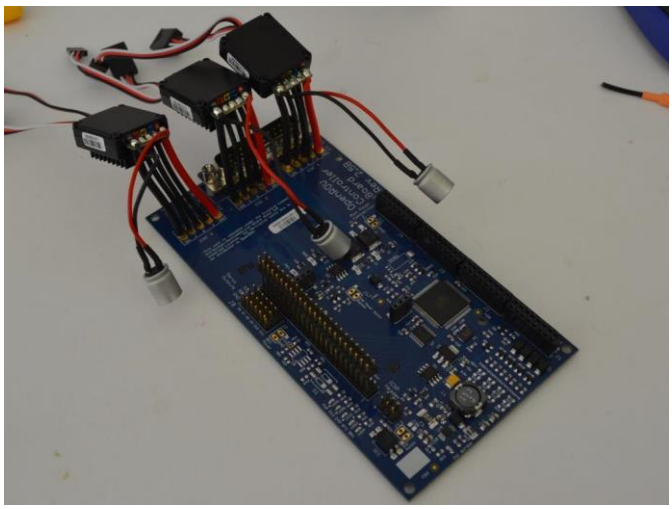

FIGURE 3.1 (OPENROV CONTROLLER BOARD)

Some features of the OpenROV main controller board include an Embedded Arduino Mega with plenty of available IO for external devices and add-ons. It comes with an Embedded Power Switch - that Allows ROV to be turned on and off remotely (through tether) by just plugging it into the USB port of the PC that is used to operate the ROV. There is a space for ESC mounting pads to connect the ESC that will later on drive the motors for thrusters and the camera tilt servo.

Easy plug mounting of Homeplug adapter and Beaglebone Black without requiring any additional wiring. On-Board DB25 Connector for removable tether connection to the main board. 4 Power PWM channels for driving lights, scaling lasers, and other power devices. 6 Servo output channels for driving ESCs and servos. Embedded measurement of voltages, currents, temperature, and (optionally) humidity. Built-in 5v and 3v3 I2C buses for allowing fast connectivity with the IMU and Depth sensor.

3.1.1.2. Beaglebone Black

The Beaglebone Black has a 1GHz ARM-based CPU, 512MB of RAM and 2GB of onboard storage, expandable with a MicroSD slot. In practical terms, this is enough to run a Linux OS, along with a web browser and other desktop applications, though with limited performance. Don't think that it will replace your primary PC, but it can be a powerful tool for sophisticated projects, and a good way to learn about Linux-based operating systems. You can use the Beaglebone Black as nothing more than a small, standalone Linux computer, but the hardware is designed for use as an embedded system—a computer installed inside of a larger electronics project. The main evidence of this is in the two rows of GPIO (general purpose Input/Output) pins mounted along either side of the board. These pins allow the Beaglebone Black to communicate with a wide range of sensors, servos, outputs and other hardware, letting it act as the brain of a large, complex project.

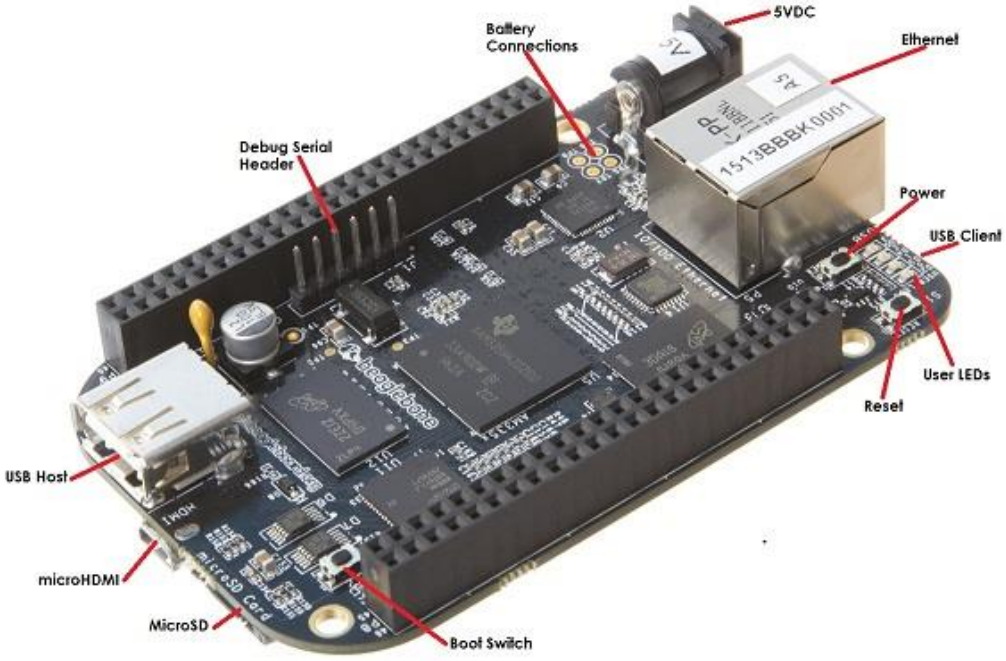

FIGURE 3.2 (BEAGLEBONE BLACK SBC)

It ships with the Debian GNU/Linux in onboard FLASH to start evaluation and development. Many other Linux distributions and operating systems are also supported on BeagleBone Black including:

- Ubuntu
- Android
- Fedora

3.1.1.3. OpenROV Structural Characteristics

The new structural design of the sub is intended to be easier and faster to build, as well as easier to modify and debug. Structural changes and new propellers also make it faster, more buoyant, and more durable, says the project.

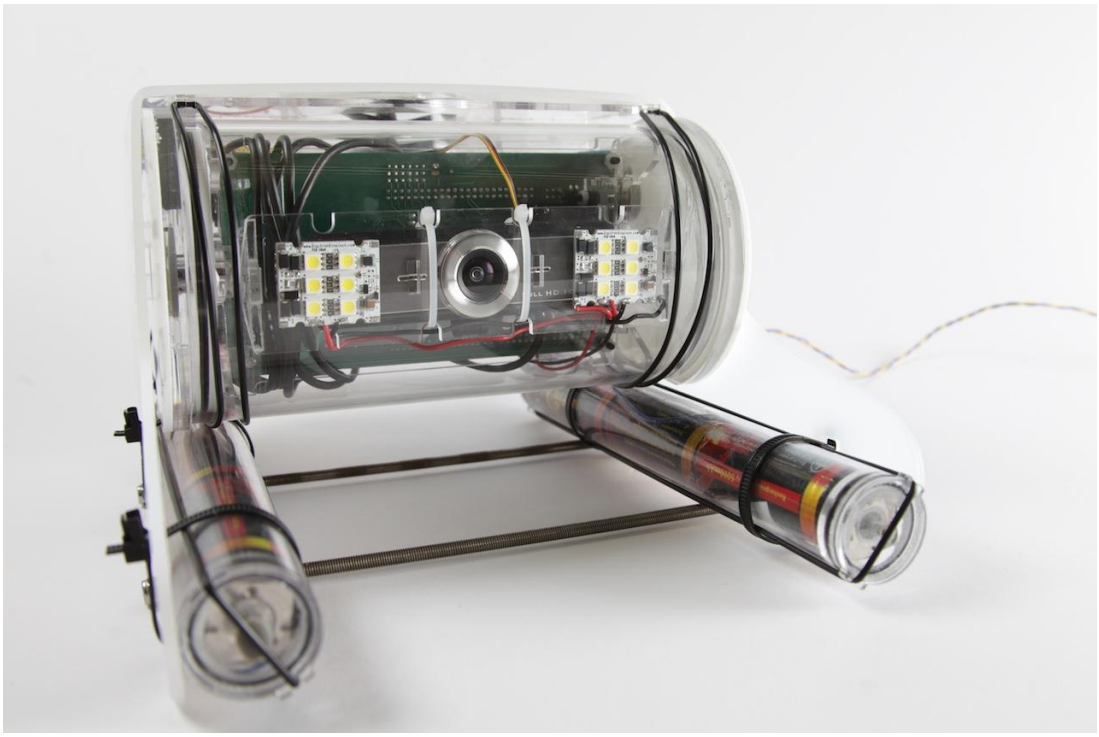

FIGURE 3.3 OPENROV ROBOT

Some of the main structural features are:

- Polypropylene shell — compared to earlier acrylic, it's more durable and buoyant, with tighter tolerances; semi-translucent
- Graupner propellers — 200 percent more efficient for faster speed; twice the battery life
- Battery tube design — clear PETG tube construction with O-ring-based endcaps and potting-sealed rear endcaps for greater leak resistance at depth
- Other structural — double tabs for shell mounting, for greater durability; stronger endcap flange; wider structure with more internal space for adding components including USB cabling direct to BeagleBone
- Camera — independently rotating camera and light platform with HD video, wide-angle lens, and tilt function
- Laser rangefinders — 2x laser emitters for distance and size calculation.

The current version of OpenROV still runs on three FalconSEKIDO brushless motors, with two horizontal and one vertical thruster. It's unclear whether it can descend past the earlier record of 25 meters in depth. The goal of the project is 100 meters, and a 100-meter tether with 10/100 Ethernet is

provided in the kit. (WiFi, sadly, is not usable underwater). As before, users control the device via a connected laptop using a web browser UI.

3.1.2. Software Environment

3.1.2.1. Angstrom Linux

The Angstrom Distribution releases demo images that ship with the with the boards and are used in the process of validating the boards and diagnostics for users. Of course, they have many uses beyond validation and diagnostics, but there are many other distributions of Linux and other non-Linux software systems you might want to run.

Ångström was started by a small group of people who worked on the Open Embedded, Open Zaurus and Open Simpad projects to unify their effort to make a stable and user friendly distribution for embedded devices like handhelds, set top boxes and network-attached storage devices and more. Angstrom is versatile; it scales down to devices with 4MB of flash to devices with terabytes of RAID storage.

3.1.2.2. Node.Js Server

Node.js is a platform built on Chrome's JavaScript runtime for easily building fast, scalable network applications. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient, perfect for data-intensive real-time applications that run across distributed devices.One goal of OpenROV is to have onboard video for live viewing as the user operates the ROV.

Combining Node.Js with MPEG_Streamer and Sockects provides a lot of power and room for future growth. But is also provides well documented means to extend OpenROV. With Node.js and Socket.io, not only are we able to stream video to a web browser by updating an image, but we are also able to control the ROV and view valuable sensor information.

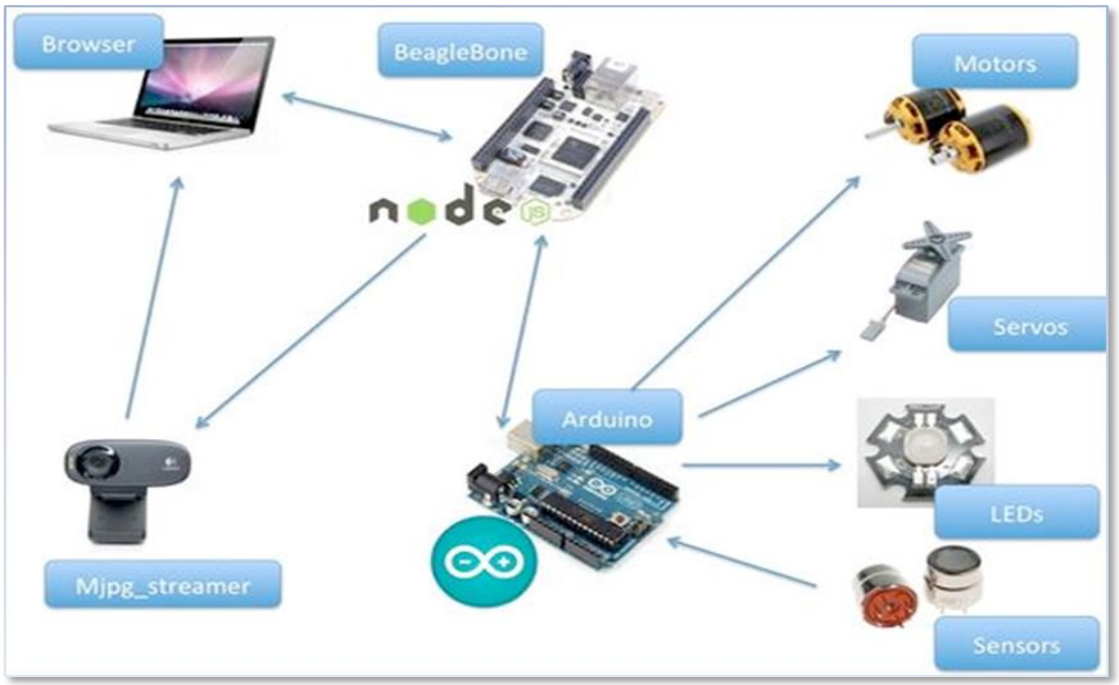

FIGURE 3.4 (NODE.JS CONNECTIVITY)

3.2. Architectural Strategies

3.2.1. Project Management Strategies

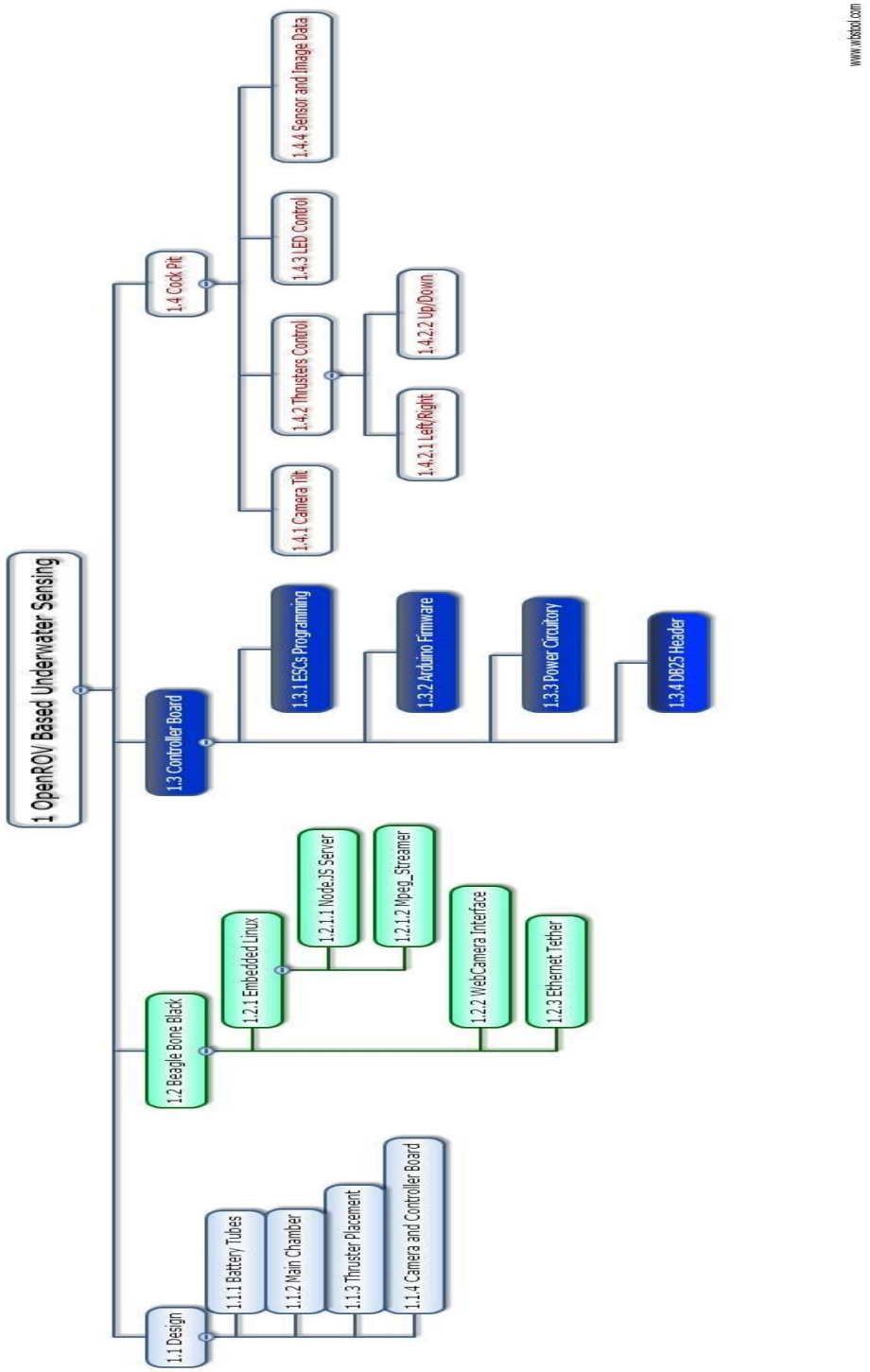

FIGURE 3.5 (PROJECT'S WORK BREAKDOWN STRUCTURE)

3.2.2. Development Method

Building the OpenROV doesn't require a master in robotics or engineering, but it's a fun project that can be completed in a weekend. We have laid down all the steps in an easy to follow guide. When you are done building and have used your ROV a few times, if you feel like it, you should start contributing, either by contributing to the core system (body, electronics or code) or by building extensions (payloads) that plug in into the ROV.

3.2.2.1. Tools Required For Assembly

- Applicator syringe for Acrylic Cement
- Applicator gun for Epoxy
- Hot Glue Gun
- Soldering Iron
- Heat Gun for shrink tubing (but kitchen stove will work)
- Sandpaper
- Alan wrench for M2 socket head screws
- Hacksaw (or other small saw) for cutting plastic syringe

3.2.2.2. Materials Required For Assembly

- Acrylic Cement for gluing laser cut parts
- Epoxy for potting End Caps
- Liquid Electrical Tape for waterproofing motor lead connection
- Heat Shrink Tubing for motor lead connection
- Solder
- Silicon Grease or Petroleum Jelly for O-Rings
- Silicone Spray for prevention of motor oxidation

3.2.2.3. Electronic Parts

- OpenROV controller with ATMEGA2560 (Arduino Mega) on-board
- OpenROV interface board to connect your computer
- BeagleBone Black
- Genius HD Webcam

- Falcon SEKIDO brushless motor ESCs (3)
- LED light array (2)
- Laser emitters for distance and size calculation (2)
- Voltage regulator
- Tenda Homeplug adapter for ethernet-over-2wire
- HiTec micro servo

3.2.2.4. OpenROV Equipment

- All laser-cut acrylic parts needed to assemble OpenROV
- Laser-cut polypropylene shell with unique serial number
- Clear cast acrylic Electronics Tube
- Extruded acrylic battery tubes (2)
- Extruded acrylic propeller ducts (3)
- Battery terminals (4)
- O-rings for endcaps, battery tubes, and electronics tube
- Stainless steel and nylon metric fasteners needed for assembly
- Stainless steel M5 payload rods (2)
- 1mL syringes for venting (2)
- 100 meter twisted pair tether
- 15 meter 20ga. hook-up wire
- DB-25 connector
- RJ45 male/male connector
- Brushless DC motors (3)
- Graupner high-efficiency marine propellers (3)
- Reusable zip-ties for battery tube mounting
- Small zip-ties for camera mounting
- 3M adhesive-lined heatshrink for marine environment (3)
- Cable wrap and heatshrink for cable management
- OpenROV vinyl decals (2)

3.2.2.5. Openrov Cockpit Installation

Step 1

Follow the instructions from our image project to get a starting image: https://github.com/ OpenROV/ openrov-image. The default user is rov and password is OpenROV

Step 2

Go ahead and start up the image in the beaglebone. If you connect the rov to a router it will use DHCP to get an address. You many need to login to your router and examine the dhcp logs to figure out what

IP was assigned. If you connect directly to a laptop, a static IP of 192.168.254.1 is used. Go ahead and SSH on to you rova) sshrov@

Step 3

Customize the name of the image to match your rov#.Change from "openrov" (without quotes) to "openrov-XXXX" (without quotes) - where XXXX is your serial number.vi /etc/hosts exit

Step 4

Upload the right firmware on to the arduino that is driving the motors and sensors. The source code for the arduino is actually installed in the git repo on the beaglebone. The beaglebone has a full arduino development environment and the ability to upload the firmware to the arduino.Since there are multiple versions of the ROV with multiple possible configurations you have to first tailor the options for the firmware to match your ROV. It defaults to the stock installation for the current shipping ROV kit.

a) from the ssh session: sudo pico/opt/ openrov/ arduino/OpenROV/Aconfig.hb) put a 1 for the options you have, a 0 if you don't have the option and save. For most folks you don't need to do anything because the type of board will be automatically detected. In case you couldc) Login to the web session for the rov, choose settings, and select the upload firmware to arduino option.
b) The easiest way to upgrade your installation is to ssh on to
   1) ssh on to your rov
   2) cd /opt/openrov
   3) sudo ./update.sh

   This will go to the github repository and pull the latest code. You may need to reboot after the update. There is a known issue where the serial.io project sometimes fails to compile. You can ignore that, but you may have to try again if it aborts the update.

3.2.3. Future Enhancements

Some of the future enhancements can be extracted from the experiences we had with OpenROV throughout our course of project. They include:

- Adding charging circuitry to auto charge the batteries without the need for removing them out of the battery tubes.
- Water proofing with a bigger size rubber insulation so the ROV can with stand dives of more than 30ft depths.
- The OpenROV cockpit software lack a UI element for the sensors data display that is coming directly from the integrated IMU Depth and Environmental sensors.
- Object detection and tracking algorithms can be easily integrated with the real time video streams onboard Beaglebone Black SBC.
- Underwater explorations require portable batteries to be kept on ship so they can power the ROVs for longer periods of time.

CHAPTER 4

SYSTEM ARCHITECTURE

4.1. System Architecture

4.1.1. Pc Software

Connecting to the OpenROV

Method 1:

- Connect the tether directly to your computer's ethernet port
- Power up the OpenROV
- Wait about 1 minute for the system to boot up (a red light on the cape should light up indicating that the Arduino is reset and powered on by the OpenROV software)
- Direct your web browser to "192.168.254.1:8080"
- Make sure that your Ethernet adapter is set to a static IP of 192.168.254.2
- To set your static IP, you can follow these instructions for Windows, or OSX
- Your subnet mask should be 255.255.255.0, and you can leave the default gateway empty

Method 2:

- Connect the tether to a router that your computer is connected to
- Power up the OpenROV
- Wait about 1 minute for the system to boot up (a red light on the cape should light up indicating that the Arduino is reset and powered on by the OpenROV software)
- Direct your web browser to 192.168.1.1 and log into the router
- Pull up the list of connected devices, typically this menu is called something like "List of Clients" or "DHCP reservation". Google around to find specific instructions for your router model
- Find the IP of the OpenROV. The OpenROV client name will be listed as "OpenROV" (no surprise there)
- Direct your browser to "TheOpenROVipYouFound:8080"

4.1.1.1. The Cockpit

The cockpit software frontend allows for easier control over the ROV functionalities. It allows for sending and receiving commands and data directly from the ROV through a browser.

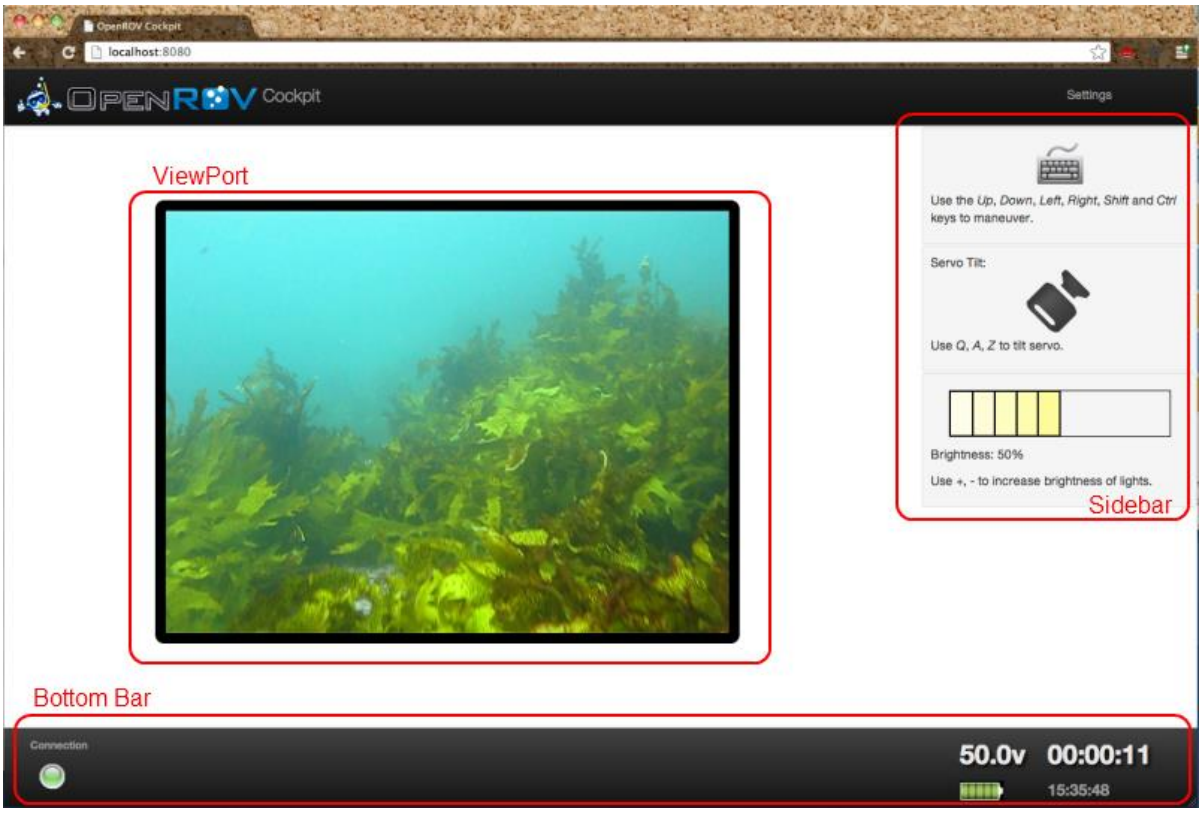

FIGURE 4.1 (OPENROV COCKPIT FRONTEND)

The view port

FIGURE 4.2 (THE VIEWPORT)

This portion of the UI shows the live video feed from the ROV. You cannot do anything with it :). In case the webcam is not detected you will see the following "no-camera" image.

FIGURE 4.3 (NO CAMERA SIGN)

If instead you see a "broken source" image it means the camera has been detected, but it's not sending video. Sometimes this happens the first time you connect after powering up the ROV: just refresh the page in the browser and it should go away.

Bottom bar

On the bottom bar the status of the ROV is displayed.

On the far left you have an indicator of whether the connection between the board and physical actuators (if you know about the architecture of the system, it checks the connection between the beaglebone board and the arduino on the cape).

On the other side you have the battery level and the current value of the voltage entering the cape and more right the time since the board has been started and the time of the day.

Sidebar

On the sidebar you have boxes with the devices you can control on the ROV, and their current status.

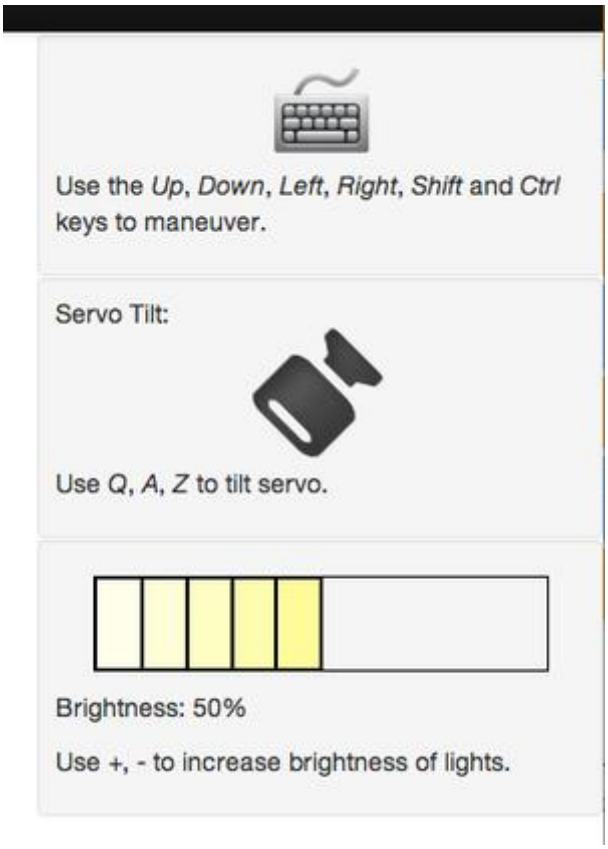

FIGURE 4.4 (COCKPIT SIDEBAR)

Controller detected

On the top you have a box that tells you if controller has been detected, and if not, it informs you that you are using the keyboard.

FIGURE 4.5 (KEYBOARD SELECTED)

Camera inclination

It shows the current angle of inclination of the servo that rotates the webcam.

Light brightness

It shows the brightness of the front-facing light.

Settings page

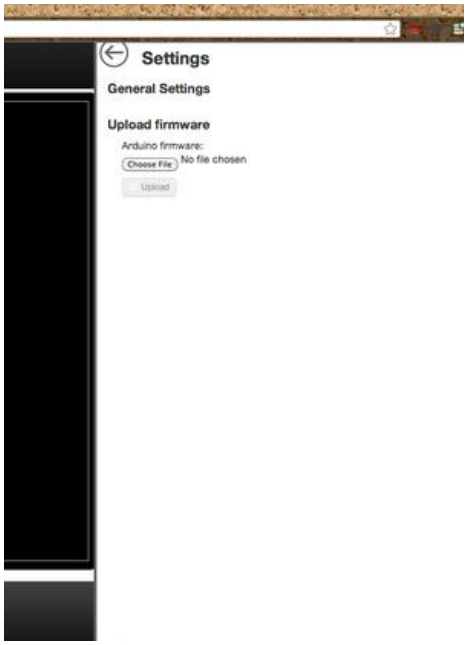

FIGURE 4.6 (THE SETTINGS PAGE)

Via the Settings page you can now upload a new firmware for the Arduino and in future set some system-wide settings and preferences.

4.1.1.2. CONTROLLING THE ROV

You can control the ROV either via a game controller, or via a normal keyboard.

4.1.1.2.1 Keyboard

Throttle and dive

You can control the forward / backward motion, the rotation and dive / surface using the arrow keys and the control keys:

Key	Effect
Arrow Up	Go Forward
Arrow Down	Go Backward
Arrow Right	Rotate clockwise
Arrow Right	Rotate counter-clockwise
Control	Go down (dive)
Shift	Go up (surface)

TABLE 4.1 NAVIGATIONAL CONTROLS

Set the speed of the motors in 5-step increments, using keys from 1 to 5:

Key	Effect
1	Set speed 20%
2	Set speed 40%
3	Set speed 60%

4	Set speed 80%
5	Set speed 100%

TABLE 4.1 SPEED CONTROLS

You can also trim the forward / backward and the dive / surface motion:

Key	Effect
7	Trim Lift down (dive)
8	Trim Lift up (surface)
9	Trim Backward
0	Rotate Forward
Space Bar	Reset all trim settings

TABLE 4.2 (THRUSTOR CONTROLS)

Web cam rotation

The rotation of the camera is controlled via the Q, A, Z keys:

Key	Effect
Q	Point webcam up
A	Point webcam forward
Z	Point webcam down

TABLE 3.4 WEBCAMERA CONTROLS

Light brightness

You can turn on the front-facing lights and control their brightness using the + and - keys. The brightness can be controlled in 10 steps.

Key	Effect
p	Increase brightness
o	Decrease brightness

TABLE 4.4 LED BUTTONS

4.1.1.3. PORTING THE LINUX CODE AND ARDUINO FIRMWARE

To install the OpenROV image onto the BeagleBone you will need:

- The BeagleBone
- A micro-SD card with 4gb or more
- A SD card reader
- A computer

Applying the image to your micro SD card is an easy seven step procedure. It installs the linux code over the SD Card containing the programming files and the Node.Js Server Software.

Step 1: Download the pre-made compressed image file.

Step 2: (For windows) Download SDFormatter and Win32DiskImager to your computer

Step 3: Format a microSD card using SDFormatter. "Quick Format" will work fine

Step 4: Unzip the file you've downloaded (OpenROV-2.5_01.img.7z)

Step 5: Run Win32DiskImager and select the image file (ending in *.img) you just unzipped. Make sure that the drive you select under "Device" is the SD card you've inserted.

Step 6: Write the image file to your formatted SD card by pressing the "Write" button in Win32DiskImager. This process will take around 5 or 10 minutes.

FIGURE 4.7 (WIN32DISKIMAGER EXAMPLE)

Step 7: Once the image has been successfully applied to the microSD card, eject the card and put in in your BeagleBone.

Once the ROV has restarted with the new linux image, we can proceed to programming the OpenROV Controller Board.

Step 1: In Cockpit, press the "Settings" button in the upper right-hand side of the screen, and press "Upload firmware from SD card to Arduino"

Step 2: A window called "Upload Arduino Firmware" will pop up. Press the blue "Apply New Firmware" button in the lower right side of the window. This will upload the Arduino Code necessary for the Controller board to work.

Step 3: Once the Arduino Firmware has been uploaded (the window will stay open, but the green progress bar will have made it all the way to the right) you can close the window.

Step 4: Restart the ROV

Step 5: Once the ROV has restarted, verify that you can read telemetry data, control the lights, and control the thrusters. You may need to program and calibrate the thrusters before they respond properly to commands.

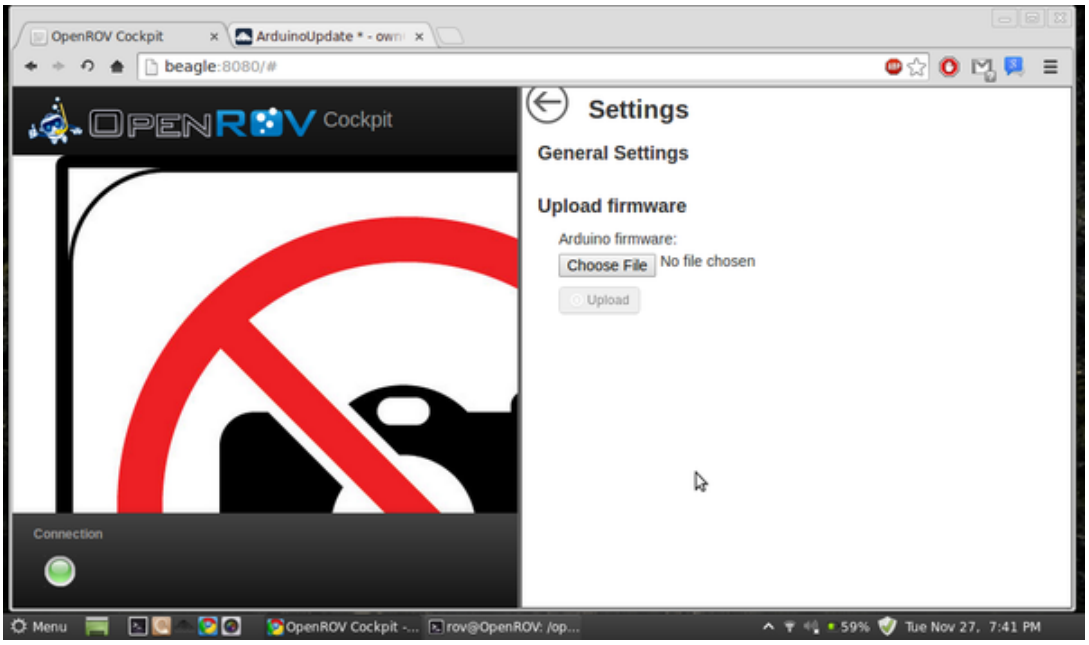

FIGURE 4.8 (SETTINGS)

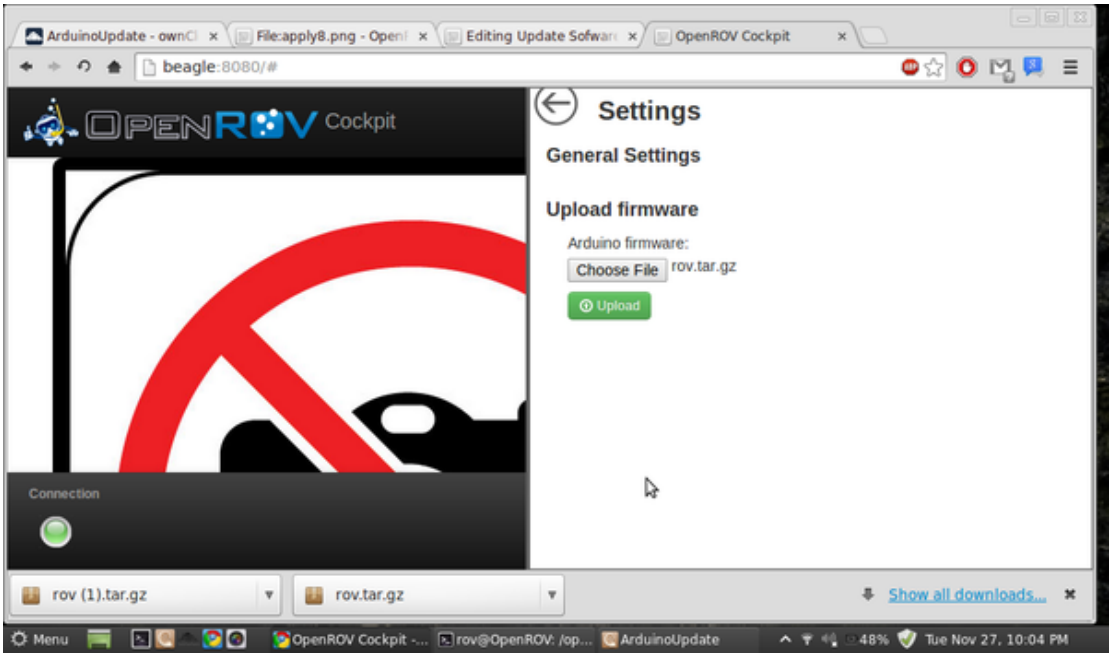

FIGURE 4.9 (UPLOADING FIRMWARE)

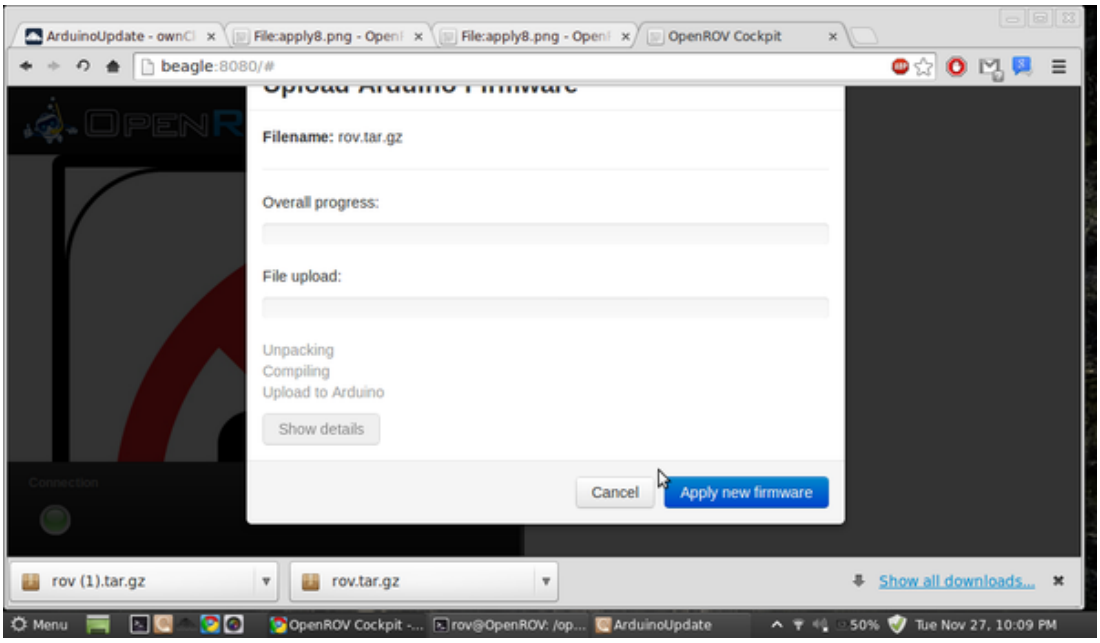

FIGURE 4.10 (APPLYING ARDUINO FIRMWARE)

4.1.2. OPENROV HARDWARE ASSEMBLY

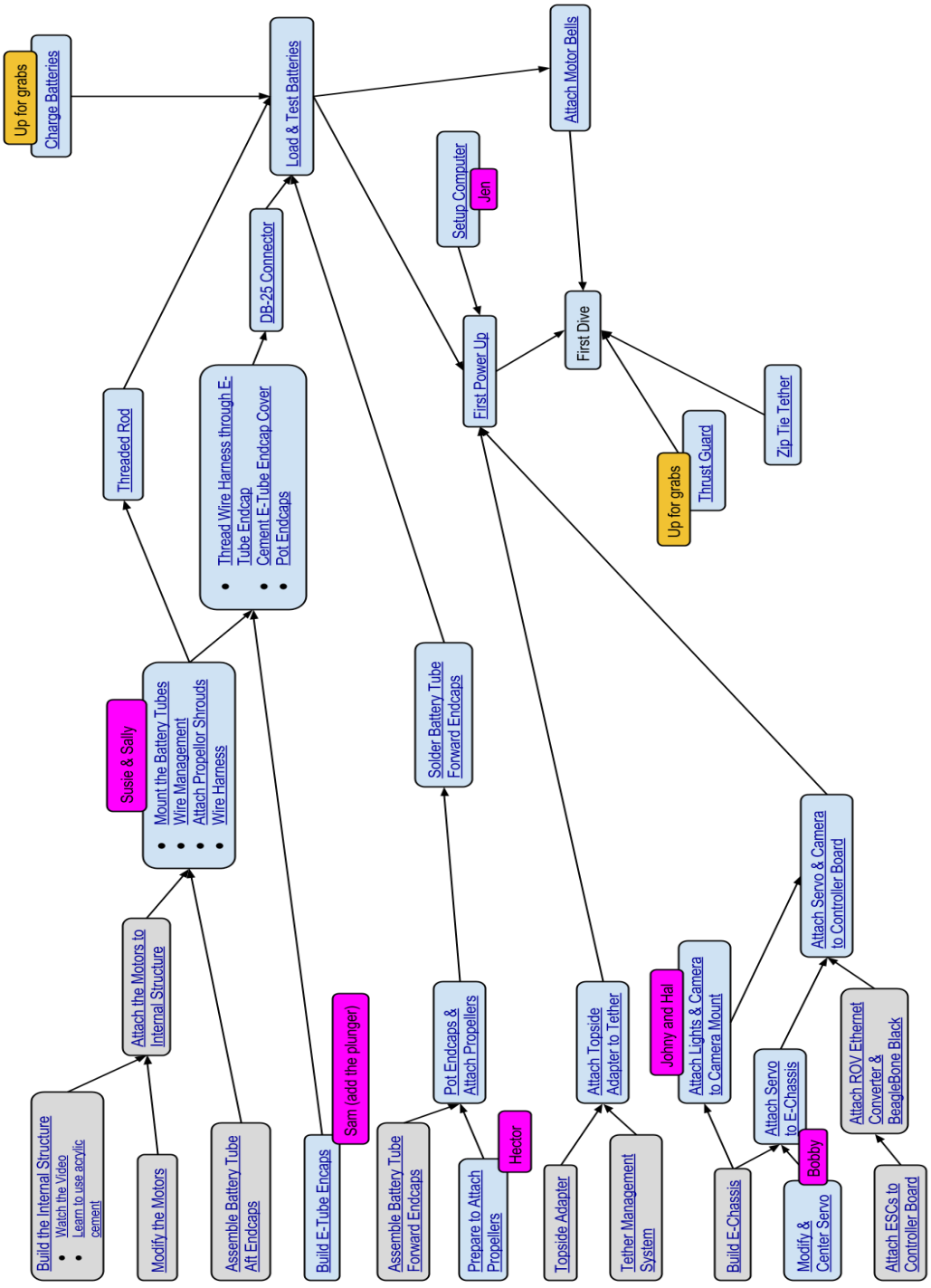

FIGURE 4.11 (OPENROV ASSEMBLY FLOWCHART)

In the following set of instructions, the document will walk you through all the steps of building OpenROV v2.5. Although these steps apply to anyone building an OpenROV - whether that be from scratch or from a kit- this instruction set assumes that you already have the parts included with each kit. We hope you'll enjoy this... now let's get started! Check to make sure that you have all the necessary parts to build your OpenROV v2.5! If you want to be thorough, check the Bill of materials available online to make sure you have all the parts. The first thing we're going to build is the internal structure of the ROV which fits inside the ROV Shell.

FIGURE 4.12 (ASSEMBLY PARTS)

To build the internal structure, we'll need several tools and materials:

Internal structure parts (15 pieces) (There are other acrylic pieces included in the kit, but those are for other parts and are not shown here)

- Acrylic cement and applicator syringe

- Sand paper (around 150 grit)
- Paper towels

Using the right technique with the applicator syringe will make assembling the internal structure a lot easier and cleaner. Before applying cement to a joint, be sure to create suction inside the syringe so cement only comes out when you want.

The pieces you got likely came with a protective plastic backing on them. You'll need to peel this off of each piece before cementing it to anything

Cement two of the three handle pieces (shown in the photos) together to make one thick stack.

It's best to initially stagger the pieces to help guide the applicator syringe to the interface between them. The cement sets quickly though, so be sure to align the pieces with each other while they can still move.

Once the first two handle pieces are attached, add the third. This is a good moment to fillet the handle (now comprized of three pieces) using sand paper so it will be nicer to hold on to once the ROV is built.

Now, we'll attach the lateral shroud support to the handle and attach that assembly to the top piece of the internal structure. After assembling the pieces, add cement to any areas that have not yet been bonded. Be sure to create a good bond between the top plate and forward part of the handle as a lot of the strain from picking the ROV up will be concentrated here.

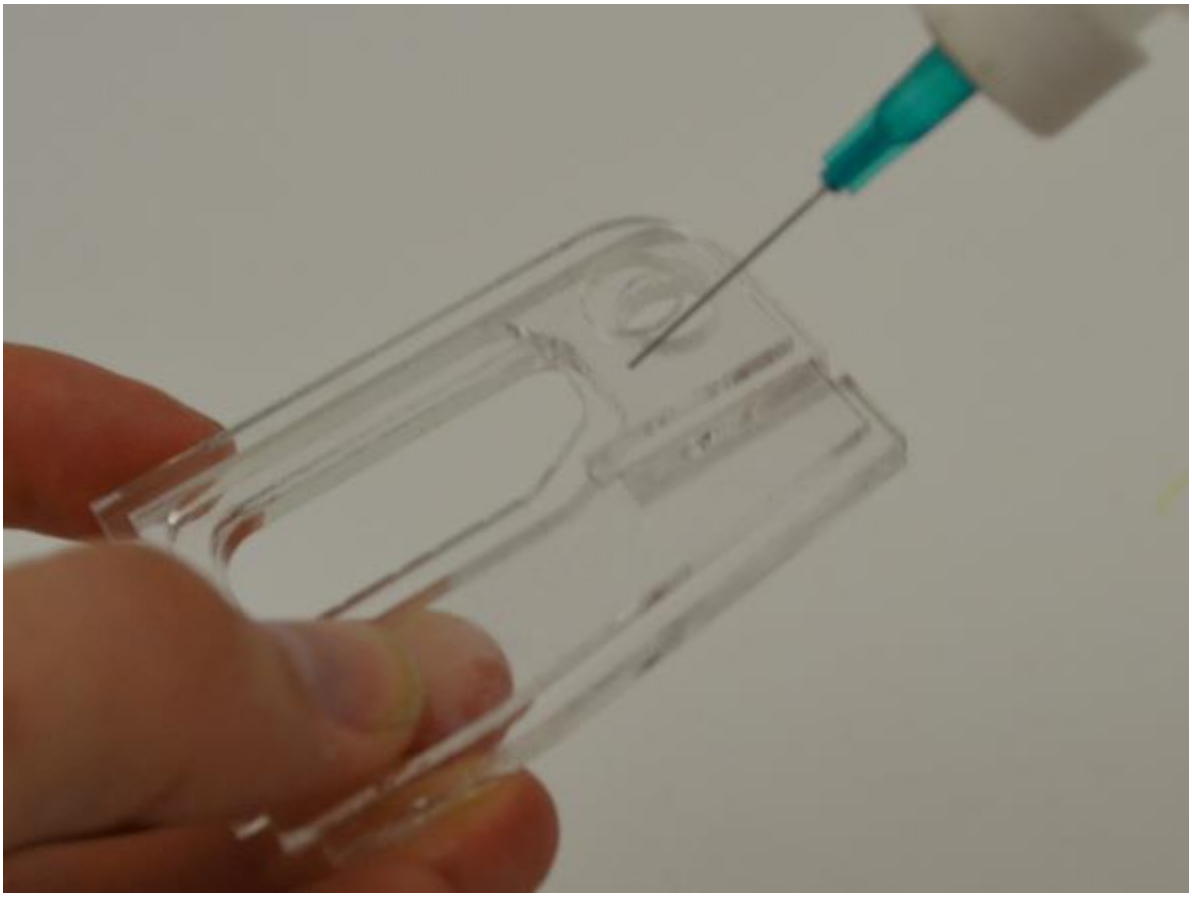

FIGURE 4.13 (APPLYING ACRYLIC CEMENT THROUGH SRYNGE)

The Bulkhead and Motor Mount assembly can now be placed down onto the Top Plate of the ROV. Once again, be sure to bond any interfaces that have been created. The Tube Cradle pieces can now be built. Two pieces are used for each of the two cradles. Just as you did with the handle pieces, it is recommended that you stagger the pieces in order to guide the syringe and cement, but be sure to align the pieces before the cement sets. Place the Tube Cradle Assemblies into place and cement them to the rest of the internal structure. Be sure there is a good cement bond near the ends of the cradles as these areas will be under significant strain when securing the main tube later on.

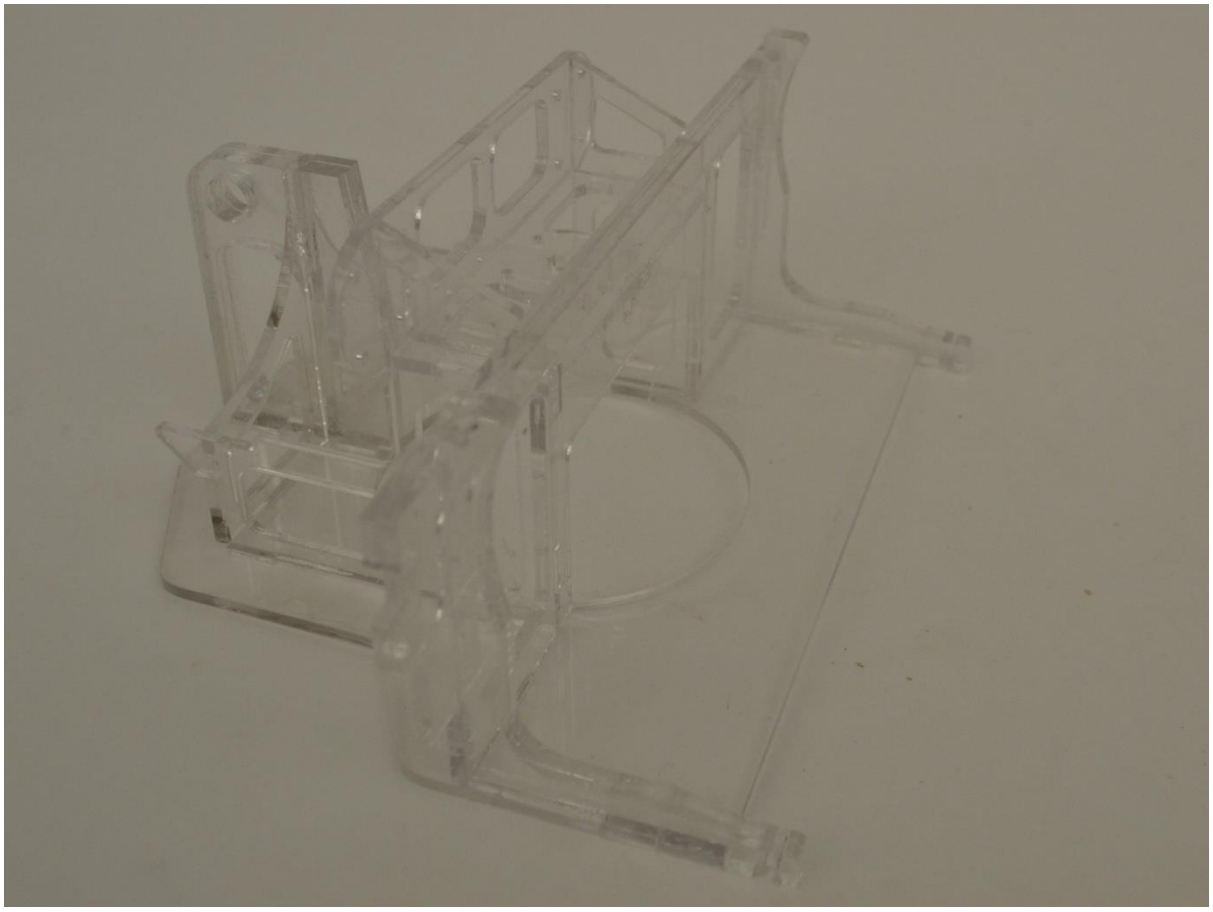

FIGURE 4.14 (MAIN ACRYLIC ASSEMBLY)

Insert the needle of the acrylic cement syringe between the two endcap disks with the needle facing downward and the bottom of the disks squeezed together. Remember to first create a suction in through the needle of the syringe so you can control the flow of cement. Inject acrylic cement between the two disks until about half of the disk area is wet . Press the disks together against a flat surface on top of a paper towel.If needed, the disks can be rotated against each other about the syringe to spread out the cement. Make sure the glue does not take hold while the pass through holes are not aligned. Use a paper towel to remove any excessive glue that beads up against the edges. The clear disks provide a visual indication of how/where the disks have adhered. If there are spots that did not get bonded with glue, applying glue to the perimeter of the disk in that area may help strengthen the attachment.

After removing plastic backing, add another 94mm disk (bigger one) to the syringe on the other side of the 90mm disk. So, it goes big circle -> little circle -> now big circle again. Push the large-hole-side of the syringe into the center hole of the disk so that the two large disks sandwich the small disk.

Repeat the preceding steps to cement the disk in place. Remove backing and place the 108mm diameter (thick) endcap flange over the most recently added disk (where the large-hole-side of the syringe sticks out)make sure that the hole in the center is concentric with the syringe, and the cutout slot is also aligned with the wire harness hole. Inject acrylic cement between the flange disk and the outer 94mm disk in a similar way to how other disks have been attached. Clamp or press the parts together until the cement sets. The final white flange disk will be left off for the time being. Repeat the previous procedure for the other endcap set BUT MIRROR THE DIRECTION OF THE 108MM THICK FLANGE DISK. The endcaps should look like mirror images of each other when finished so that they will fit appropriately on either side of the electronics tube.Now we're going to start building the battery tubes. Our first procedure will be to assemble the aft battery tube endcap. You'll need the parts boxed in red.Roughen a corner of the spring terminal adjacent to the smaller hole (the one that doesn't have a lip). Put a blob of solder over the roughened corner. Cut four 1m lengths of 20awg stranded wire (two will be used now for the first battery tube and two will be used for the other battery tube). Strip about 1cm of insulation and tin one end of each of the four wire lengths. Put one of the tinned wire ends through the smaller hole (without a lip) on the base of the spring terminal and solder it to the base (basically bend over the solder blob and heat to attach). Thread a second 1m long wire through the bigger hole on the spring terminal (the one with the lip).

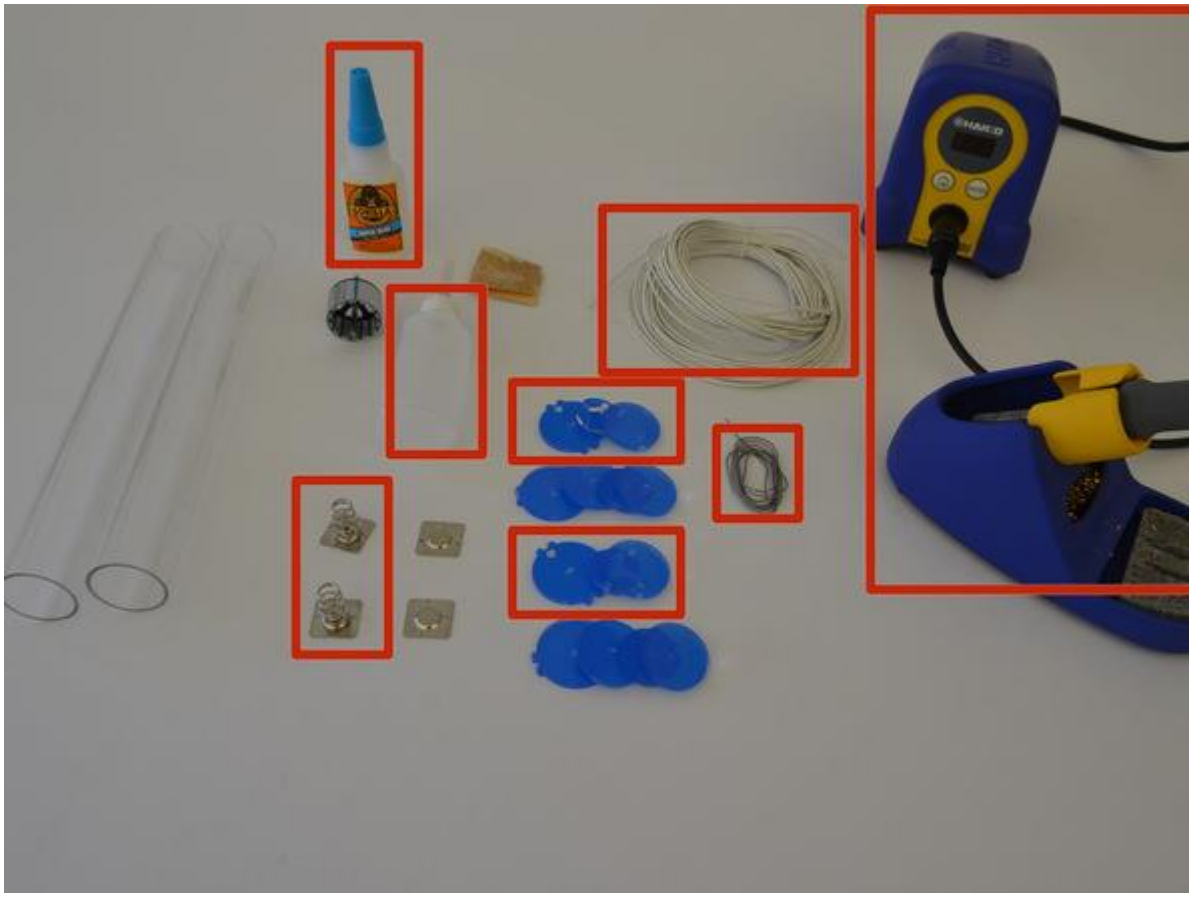

FIGURE 4.15 (BATTERY TUBES)

Run the lead wires through the aft battery endcap inner disk (it's the one with two holes that looks like a button). Super Glue(not acrylic cement) the spring terminal to the inner aft battery endcap disk.Align and glue outer and middle disks of aft endcap. It's important that it lines up correctly - the nub of the middle disk accenting the hole on the outer disk. Measure out 25cm of wire length passing through the spring terminal endcap. (Good idea to make a reference mark on the wire near the base of the terminal.Slip the middle and outer disks over the longer length of wire. Both wires through the middle hole.Double check that the wire is 25cm (once the disks are cemented together, it's very hard to pull the wire in or out). Make sure the inner, middle, and outer disks are concentric (visual). Then glue them together.

Before mounting the motors, use the 1.5mm hex wrench to tighten the set screws on each motor. (We've found that these are often a little loose out of the box, and it's much easier to tighten them now then after the motors have been mounted)

Enough screws, lock washers, and nuts were included to fasten each motor in four places, however, it is easier and just as effective to only use two sets per motor.

Place the M3 screws in through the top of the motor mount, then, holding the screw in place with the hex key, place the motor on the internal structure so that the screw also fits through the appropriate mounting hole. With the internal structure up-side-down and the motors facing toward you, the wire harness should go around the left side of the main bulkhead.

To fasten the motor, place a lock washer on the shaft of the screw as shown, then place a nut on top of that and hold the nut steady with your thumb. Tighten the screw with the hex wrench until tight. Pliers should not be necessary. The motor with the shortest wire lead should go on the left with the structure upside-down (which will be the starboard side of the ROV), the middle-length wire lead motor should go on in the middle (it will be the vertical motor) and the motor with the longest leads should go on the right. It will be the port motor when the ROV is right side up. Glue the two acrylic rings (motor shrouds) on to the structure.

The wire harness can now be inserted into the port endcap. Insert the wire harness from the flange end inward as shown.

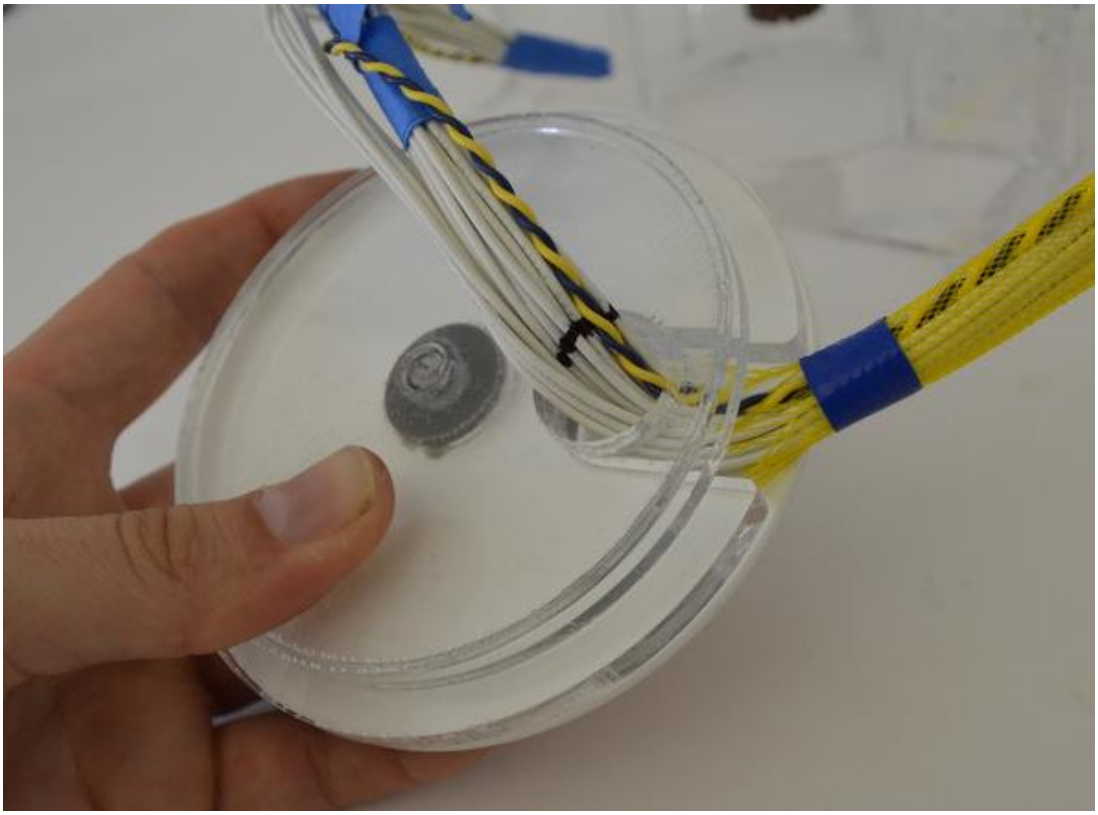

FIGURE 4.16 (WIRE HARNESS THROUGH THE END CAP)

Pull the wire harness through the end cap until the marks on the wires made earlier are flush with the inner plate of the end cap.

To verify that the wire markings are in the right place (so that the harness will not be too long or too short), position the endcap in its cradle and check that the mesh-covered part is slightly slack. If the harness is not the right tightness, it can be adjusted. If this is done, you should make a new mark on the wire to indicate the appropriate place it should pass the inner face of the endcap. In this step, we'll add the white flange cover to the theendcap. Position the flange cover so that its flat edge is aligned with that of the main flange plate. Be sure the spread out the wires in the harness so that they do not push the flange cover out.

Inject cement between the two plates in a similar way to how the other parts of the endcap were assembled. After sufficient cement has been injected between the plates, push the assembly down against a table to spread the cement across the interface.

Be sure the flange cover and main flange plate stay concentric and aligned as the cement sets. At this point, the flange cover for the starboard endcap can also be added in a similar way to how the port

endcap flange cover was added. Use a hot glue gun to create a barrier that will prevent potting compound from leaking out the channel when the endcap is potted.

We're now going to get ready for one of the most important steps of the entire kit-building process: potting the port endcap. To start out, you'll want to make it so the wire harness coming through the endcap is pulled stright upward so that it doesn't touch the edges of the hole it is going through.

FIGURE 4.17 (MOTORS)

For the next steps, you'll want to find a place where you can clamp or otherwise hold the endcap down against upward tension on the wire harness. You'll also need something above the wire harness that you can attach a rubber band too. A desk with a hutch, the space below a table, or an empty shelf on a bookcase are all good candidates.

Wrap some tape around the wire harness to make its cross-section somewhat circular just above where the wire comes out of the endcap hole.

Tie a thread or piece of fishing line to the wire harness a few cm above where it comes through the endcap. A knot that involves several wraps around the harness will work better.

Secure the other end of the line to a rubber band that is positioned directly above the hole in the endcap. Before tying the line off, pull it tight so that the rubber band is stretching downward from the tension in the line.

Use hot glue to block the outlet of the endcap pass-through so that potting compound will not leak out around the sides of the harness. Try not to get too much glue on the disks that will go inside the tube, and also make sure that the wire harness itself does not bulge up above the height of the large flange disk.

Keep the hot glue gun on and ready to use so that any hole is discovered once the endcap has been potted can be plugged. (This can be done by simply wiping off any escaping potting then filing that area with hot glue)

Start potting the endcap by inserting the mixing tube as far down into the pass-through as possible. You'll want to fill starting from the furthest-back point to avoid bubbles. Also, insert the mixing nozzle between the wires in the bundle to make sure there are no voids that are not getting filled.

Fill the pass-through hole to the rim (if a little spills over, that's no big deal- it can be wiped off or left as is)

Within the first 20 mins of potting, move the wire harness back and forth as well as up and down longitudinally (like a toilet plunger) to make sure potting compound is filling every void.

E-90FL starts to set up after about 90 mins, but it will take a day for it to fully harden.

Insert wisdom here.

While the E-90 is still fluid, you'll want to attach propellers to the motor bells for each thuster. Start by sanding the shaft of each motor bell using a medium-grit (we used 150-grit) sand paper.

Cut off three 4mm long pieces of the adhesive-lined shrink tubing that was also used on the motor leads. These will be used to keep the shaft concentric with the mounting hole in the propeller.

Slide one of the pieces of shrink tubing down to the root of each shaft and shrink it down until it is secure using a heat gun.

Place the tip of the mixing nozzle firmly into the hole of a propeller and the hole with potting compound until it overflows. There may be some air that needs to escape for the potting to fully fill the hole. It's okay for some potting to overflow onto the surface above the hole.

Using a rotating motion, press the propeller down against the motor shaft. You'll need to sort of screw the threading onto the shrink tubing on the shaft for it to go on all the way without scrunching the tubing.

Do these steps for all three motor bells and place them somewhere where they can point directly upward. A wire shelf or a crack made between two books can allow the bottom side of the shaft to be out of the way.

We're now going to build the Electronics Chassis (also known as the E-Chassis). For this, you'll need the 13 parts shown here (as well as an additional two blinder pieces that are not shown here)

Start by cementing the Endplate pieces to the Main Platform. Orientation is importent here: the large hole on the Main Platform should be in the upper right corner with center stand-off on Endplate pieces pointing toward you (as shown). This will allow the OpenROV Controller Board to fit on the right way. Now attach the four Braces (two for each Endplate) to the E-Chassis. They should be able to fit in if inserted at an angle. Next, cement the Servo Mount to the same side of the Main Chassis the Braces are on.

Now we'll start making the camera platform. Start by cementing the two pieces for the Camera Mount arms together. The round bases of each piece should end up being concentric, but to aid with cementing, it may be easier to stagger the pieces initially. Of course, be sure to align them before the cement sets. Next, you'll want to cement the Camera Mount arms to the Camera Platform with the smaller of the two pieces on each arm facing outward.

Finally, attach the white blinders to either side of the camera hole. Now it's time to attach the ESCs to the OpenROV Controller Board. In each ESC box, you'll find one ESC, two zip ties, and a bag with shrink tubing and double-sided adhesive. You'll want to keep the zip ties and bag, so put them aside for now.

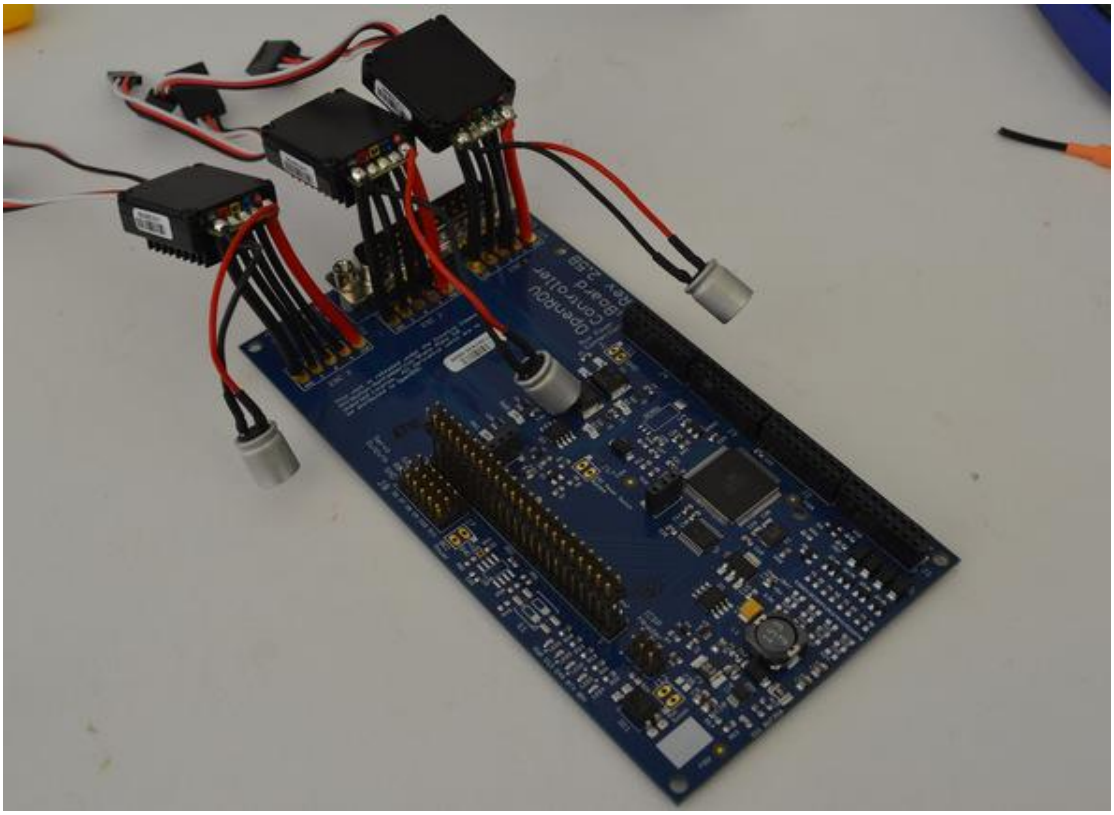

FIGURE 4.18 (ESC MOUNTING)

We'll want to cut the thick lead wires from the ESC down to a length of 4cm (Don't cut the thin capacitor wires). To keep everything looking tidy when mounted to the Controller Board, it's best to start by cutting one of the lead wires to the right length, then use that as a guide to cut the others afterward.

Next, we'll want to roll each wire around between our fingers to make the cross section of the wires circular. This will be important later on (especially with the two power wires) for getting them to fit into the holes on the Controller Board.

Because each wire has a very high strand count (and thus the strands are very fine), we'll do a trick to prevent them from fraying. Using a soldering iron, heat the tips of each wire (which is not yet stripped) and apply a bead of solder. The silicon insulation on the lead wires should be able to take the heat.

You should end up with five wires that have small beads of solder covering their strands as shown. You can now strip each wire. Strip off 4mm of insulation from each wire using a sharp razor (auto-

strippers will not work well with the flexible silicon insulation and the short cut length). The ESCs should now be ready to be mounted to the Controller Board.

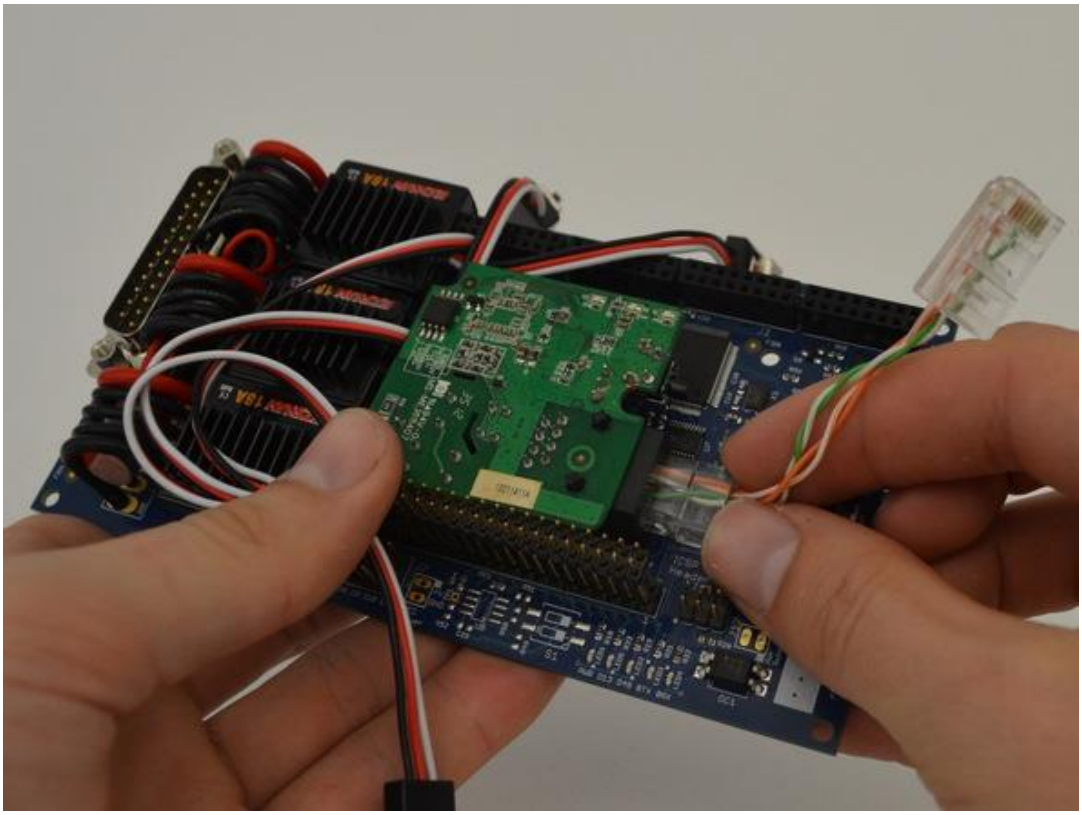

FIGURE 4.19 (MOUNTING POWER OVER ETHERNET ADAPTER)

To keep the USB cable for the webcam out of the way, we'll wrap it around the main platform of the E-Chassis. Because the Electronics Tube will have to fit over the entire assembly, it's important that the wraps of cable are side-by-side and do not stack on top of each-other. Depending on where the end of the doubled-up cable ends up (that seems to vary every time we build an E-Chassis) you can use the velcro strip that came with it, some tape, or a small zip tie to hold it in place.The resulting bundle should lay flat against the E-Chassis platform, but be pushed to the side enough so that when the camera platform is pointing all the way up or down, the camera doesn't collide with the cable.

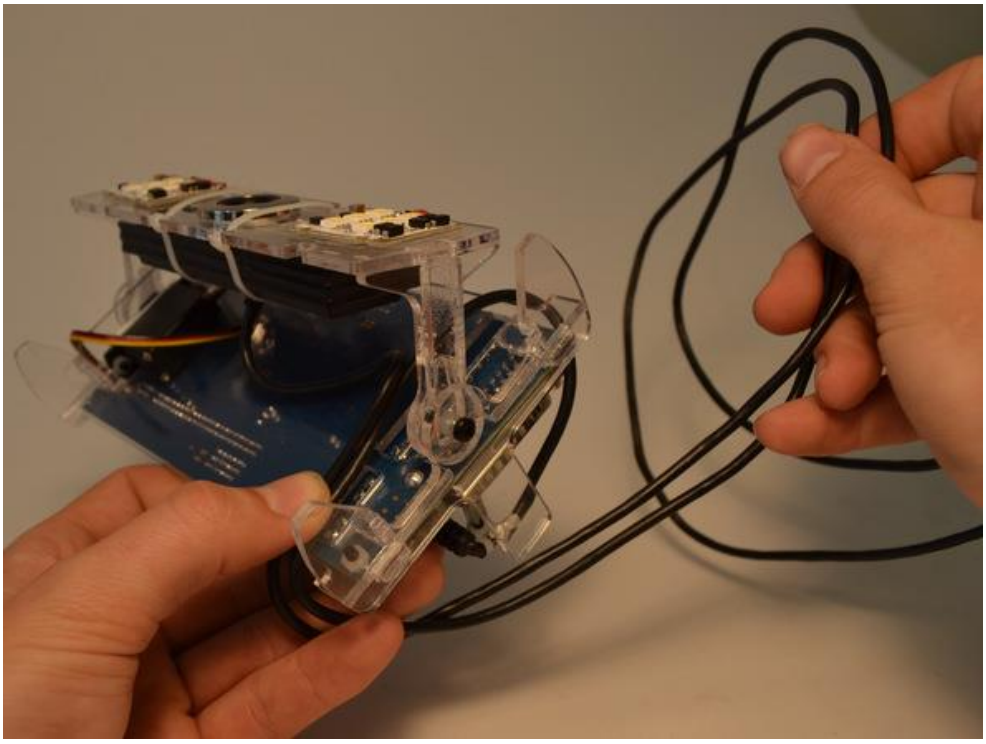

FIGURE 4.20 (CAMERA ASSEMBLY)

Once the potting in the main endcaps has cured, you can attach the DB-25 connector to the end of the wire harness. Start by removing the tape that was used to hold the bundle together. Next, measure a distance of 17cm from the inside surface of the endcap, and cut one of the wires from the bundle at that length. Measure twice, cut once- you don't want the harness to be too long or two short by more then about 1cm. Use the one wire that you've cut as a guide to cut all the remaining wires in the bundle to the same length.

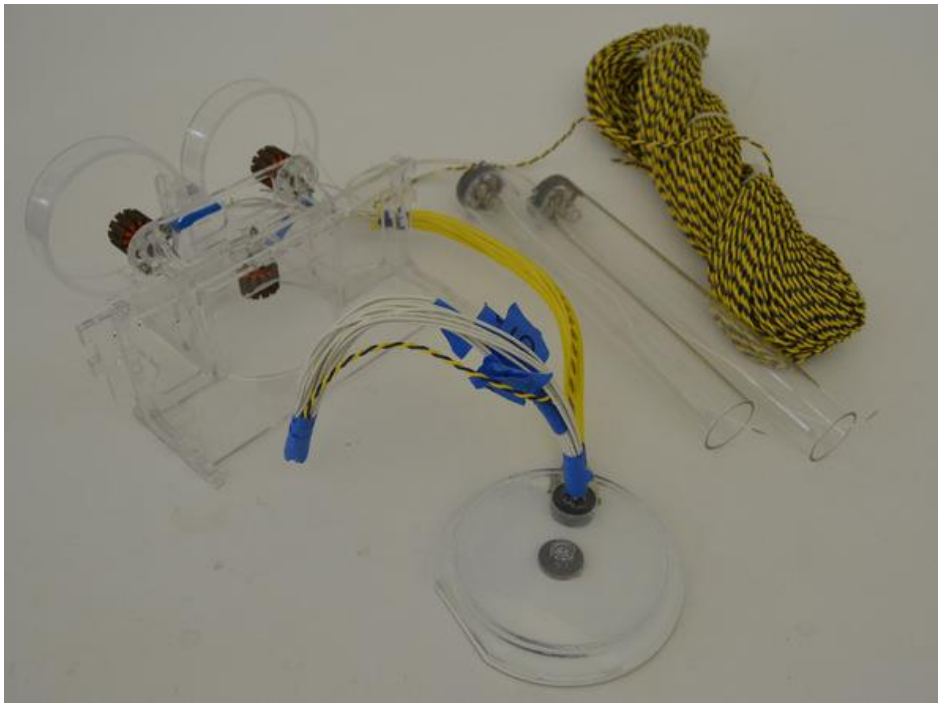

FIGURE 4.21 (WIRING)

Now you can get ready to solder the DB-25 connector to the end of the newly trimmed wire harness. Start by stripping off about 3mm of insulation from the end of each wire. Because each stripped wire will just barely fit in the solder cups for the DB-25 connector, it's easiest not to tin the wires before inserting them into their respective positions. Look on the back of the DB-25 connector (the end with the solder cups) and identify pin 1 (labeled with a faint imprint in the plastic for the connector). Use the guide shown to attach wires to the DB-25 connector. Be sure that battery polarity is correct by measuring for continuity between the forward (positive) terminal in each pack and the other end of the wire it goes too. The four I2C wires and the two AUX wires can be any of the extra six that were potted into the endcap- there is no order that need be put to them at this time.

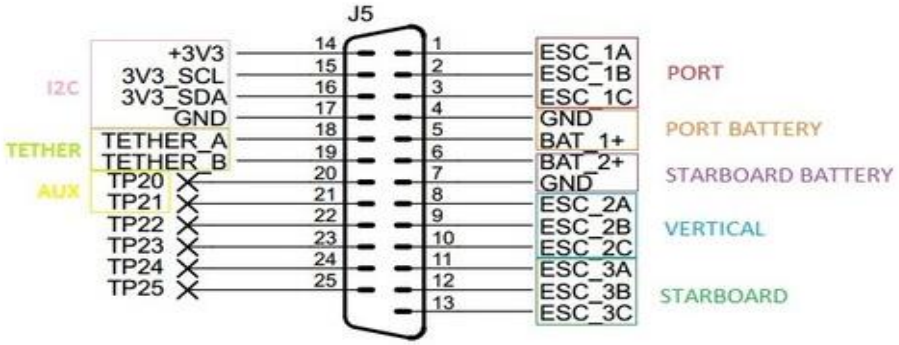

FIGURE 4.22 (DB25 LAYOUT)

You're now almost ready to do your first power-up on the ROV! Insert the DB-25 connector attached to the wire harness into the reciprocal connector on the Controller Board. You may have to wiggle it around a bit to fit under the USB plug for the webcam, but it should go in without needing to unplug anything.

Plug the one of the Ethernet patch cables that came in the Homeplug box into the Topside Adapter and into the Ethernet port on your laptop.

Plug the USB cable that came with your BeagleBone into your computer and get ready to plug the other end of the cable into the port on your Homeplug adapter.

Connecting USB power to the Topside Adapter will cause the ROV to turn on. Since the ESCs may not be calibrated initially, it is a good idea to make sure the propellers on the ROV are clear, and that the camera platform can move smoothly if the servo moves.

Plug the USB cable into your topside adapter and look for lights on the Homeplug board in the Topside adapter to turn on. They should go through a startup sequence but then all be lit after several seconds (but the middle light may blink rapidly)

Look at the controller board and verify that the green "PWR" LED is on. You should also see one sold red LED and one blinking LED which were used to test the board during the QA process before being shipped.

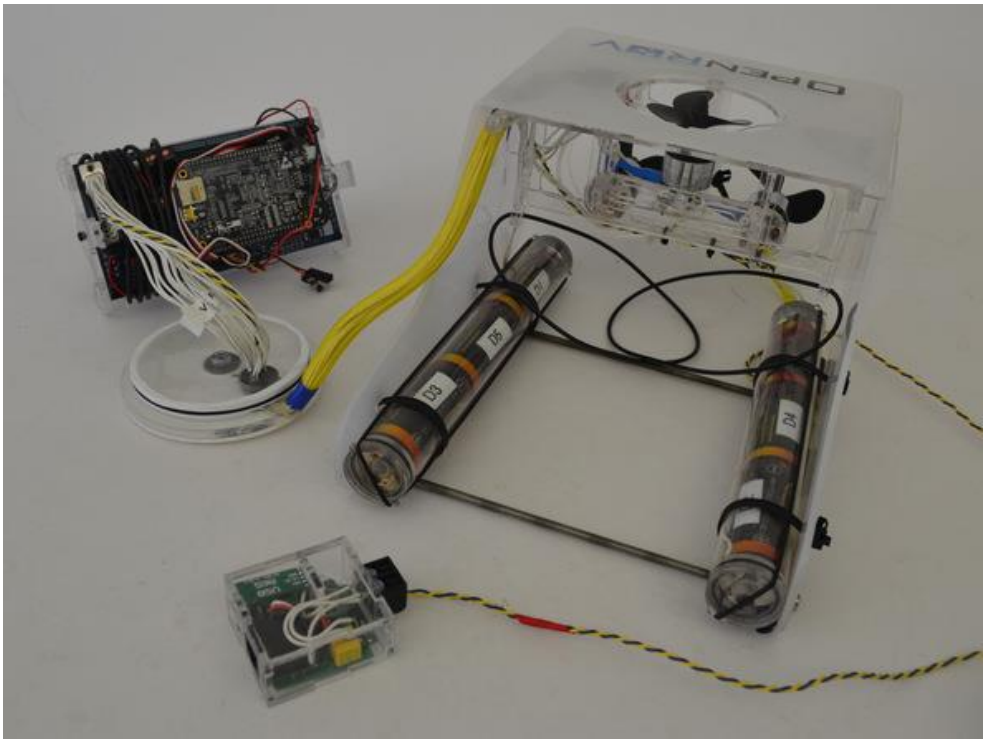

FIGURE 4.23 (COMPLETED CIRCUIT)

To prevent your ROV tether from getting twisted in the motors add a zip tie to the circular hole in the handle at the back of the ROV. Do not cut the zip tie. Wrap the ROV tether around the zip tie. Wrap the zip tie and tether with electrical tape.

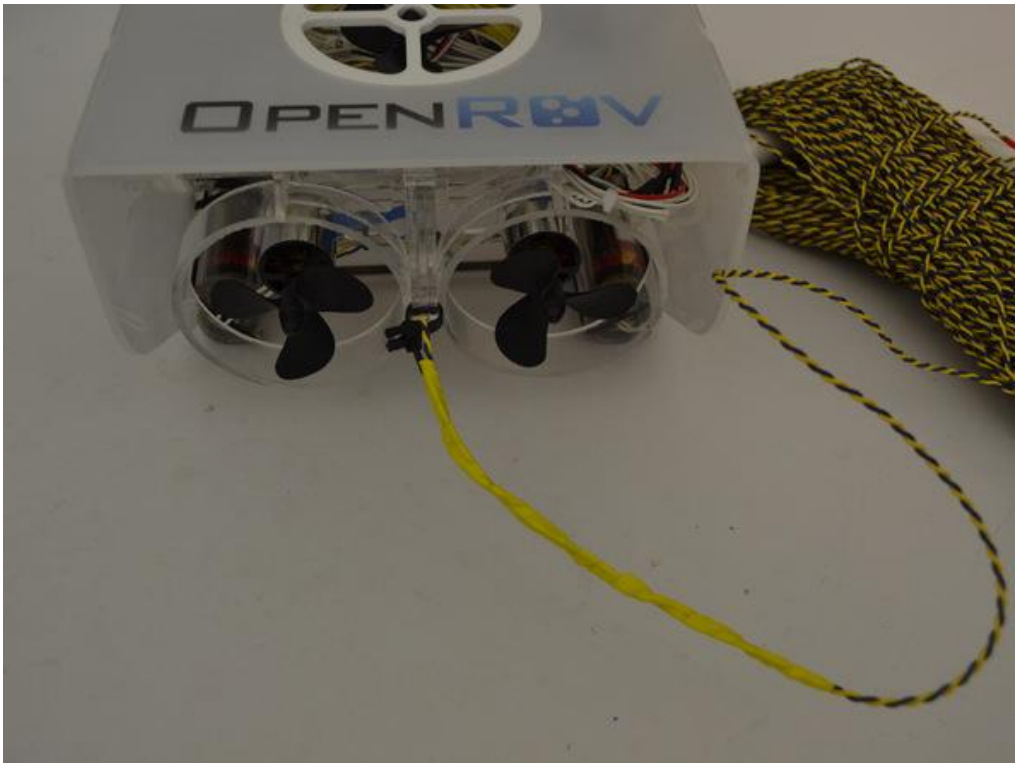

FIGURE 4.24 (THETHERING)

4.1.3. SENSOR SYSTEM

4.1.3.1. SBT 80 SENSOR BOARD

The SBT80 is a low-power, multi-channel sensor board designed for use with IEEE 802.15.4 compliant and TelosB wireless sensor network platforms. The SBT80 features visual light, infrared, acoustic, temperature, dual-axis magnetometer and dual-axis accelerometer sensors. It can be directly connected to the expansion connectors of the TelosB platform using an IDC header.

The SBT80 is a low power, flexible sensor board with multiple sensor modalities (visual light, infrared, acoustic, temperature, dual-axis magnetometer and dual-axis accelerometer) that is specifically designed to be plugged into TelosB wireless motes for use in sensor networks, data fusion, rapid application prototyping and monitoring applications. A front layout of SBT80 is shown in Figure 44.

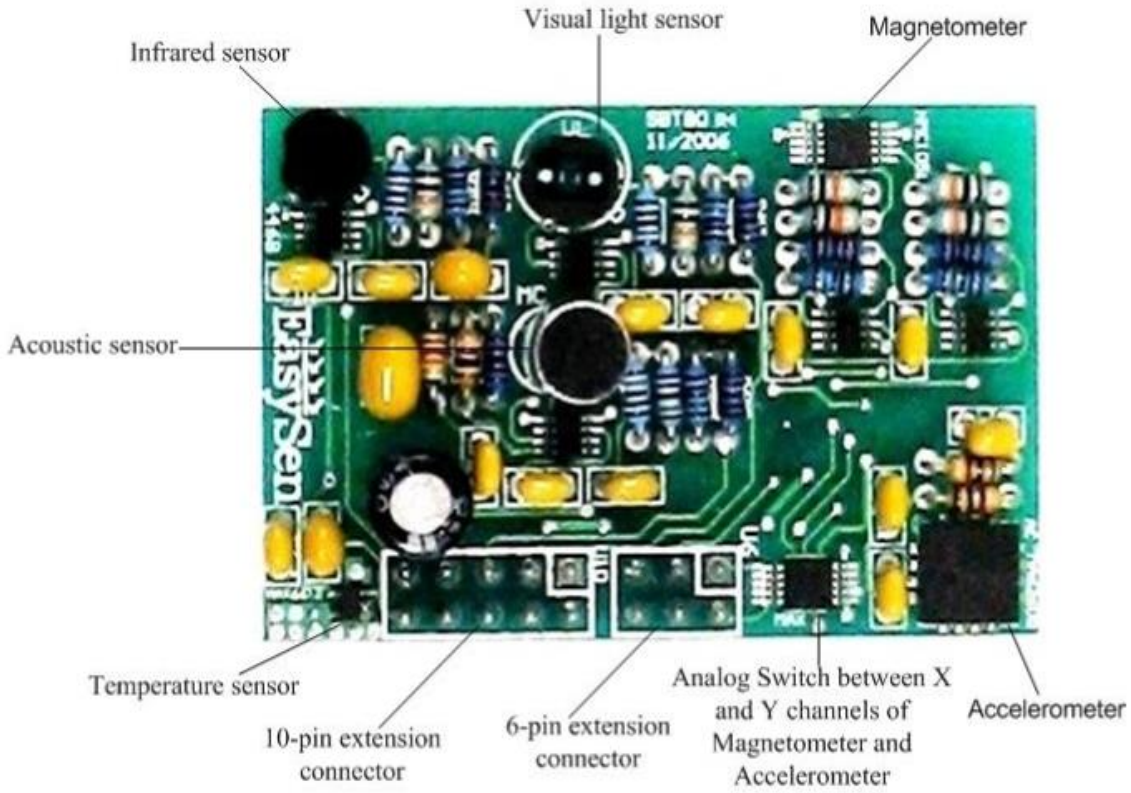

FIGURE 4.26 (SBT80 SENSOR BOARD)

The SBT80 is accompanied by tested TinyOS and Java code that can be used to gather data over 8 different sensor channels, display sensor readings on a computer, and to switch the SBT80 into a power saving "sleep" mode. The SBT80 has a smaller form-factor than TelosB platforms. By leveraging industry standard wireless sensor platforms, the SBT80 enables a wide variety of multi-modal sensor applications for security, surveillance using wireless sensor networks.

4.1.3.2. TELOSB

Each such sensor network node has typically several parts; a radio transceiver with an internal antenna or connection to an external antenna, a microcontroller, an electronic circuit for interfacing with the sensors and an energy source in the form of a battery or an embedded from of energy harvesting. A sensor node might vary in size from that of a shoebox down to the size of a grain. The mote function within the network and typically full-fill one of the two purposes. Either data-logging, processing sensor information from the environment or acting as a gateway in the adhoc wireless network formed by all the sensors to pass data back to a collection point.

The TelosB mote platform is an open source, low-power wireless sensor module designed to enable cutting-edge experimentation for the research community. The TelosB bundles all the essentials for lab studies into a single platform including: USB programming capability, an IEEE 802.15.4 compliant, high data rate radio with integrated antenna, a low-power MCU with extended memory and an optional sensor suite.

The IEEE 802.15.4 protocol has been adopted as a communication standard for the low data rate, low power consumption and low cost Wireless Sensor Networks. This protocol is quite flexible for a wide range of applications if appropriate tuning of its parameters is carried out. Most of the mote platforms use this standard for communication between the motes. TelosB mote is IEEE 802.15.4 compliant and use 2405 to 2480 Mhz band for communication. TelosB mote boards are typically powered from an external battery pack containing two AA batteries.

FIGURE 4.27 (TELOSB INTEGRATED WITH SBT80)

TinyOS is a free and open source software component-based operating system and platform targeting wireless sensor networks (WSNs). TinyOS is an embedded operating system written in the nesC programming language as a set of cooperating tasks and processes. It is intended to be incorporated into smartdust. TinyOS started as a collaboration between the University of California, Berkeley in co-operation with Intel Research and Crossbow Technology, and has since grown to be an international consortium, the TinyOS Alliance.TinyOS applications are written in nesC, a dialect of the C language optimized for the memory limits of sensor networks. Its supplementary tools are mainly in the form of Java and shell script front-ends. Associated libraries and tools, such as the nesC compiler and Atmel AVR binutilstoolchains, are mostly written in C.

TinyOS programs are built out of software components, some of which present hardware abstractions. Components are connected to each other using interfaces. TinyOS provides interfaces and components for common abstractions such as packet communication, routing, sensing, actuation and storage.

TinyOS is completely non-blocking: it has one stack. Therefore, all I/O operations that last longer than a few hundred microseconds are asynchronous and have a callback. To enable the native compiler to better optimize across call boundaries, TinyOS uses nesC's features to link these callbacks, called events, statically. While being non-blocking enables TinyOS to maintain high concurrency with one stack, it forces programmers to write complex logic by stitching together many small event handlers. To support larger computations, TinyOS provides tasks, which are similar to a deferred procedure call and interrupt handler bottom halves. A TinyOS component can post a task, which the OS will schedule to run later. Tasks are non-preemptive and run in FIFO order. This simple concurrency model is typically sufficient for I/O centric applications, but its difficulty with CPU-heavy applications has led to the development of a thread library for the OS, named TOS Threads.

TinyOS code is statically linked with program code and is compiled into a small binary, using a custom GNU toolchain. Associated utilities are provided to complete a development platform for working with TinyOS.

CHAPTER 5

TESTING AND RESULTS

5.1. TESTING AND RESULTS

After solving the problems that we faced during the installation of the project and prepare the submarine for practical experience. We connect the submarine of cable network by ip address. Special For the device then we dive the submarine in the pool to test the functions and recording the information required.

We faced some problems during the experiment, including water leaks and the difficulty of controlling the submarine and an imbalance and problems in the distribution of weight. It was resolved modification place and add some weights to better control and balance. As well as we did better insulation. the experience was succeeded so we tested the functions of sensors and read variables and recorded.

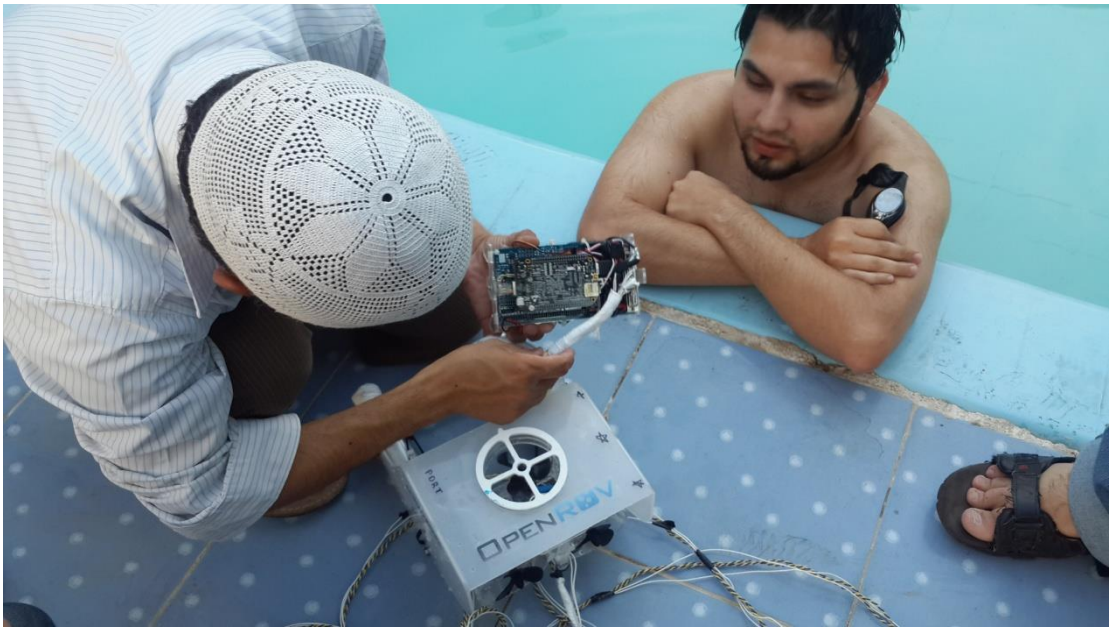

FIGURE 5.1 (TESTING IN A POOL)

The results from the sensor system were obtained by using a long USB extension wire the size of the tether cable. The steps followed were:

Program one TelosB mote with the SenseToRadio application given here.

(a) Copy and unzip the folder SenseToRadio into the /opt/tinyos2.x/apps folder in your TinyOS installation (above files will be inside /opt/tinyos2.x/apps/SenseToRadio folder)

(b) Connect one TelosB mote to the PC USB connector.

(c) Open a cygwin window, go inside the folder /opt/tinyos2.x/apps/SenseToRadio in the cygwin prompt

(d) Execute the command

make telosB install.1

This will install the application code into the TelosB mote. Here the number "1" represents the address assigned to this particular mote. Users are

free to assign any address for this purpose

Connect the sensor board SBT80 to the expansion connector of the TelosB mote

For Displaying the sensor readings on PC:

(e) Ensure that the newly programmed TelosB mote is attached to the PC via the USB port

(f) Open a cygwin window (or by reusing the same cygwin prompt above)

(g) Execute the command

motelist

This will display the serial port number the telosB mote is connected to the PC

Example : COM6

(h) Go inside the folder /opt/tinyos2.x/apps/SenseToRadio in the cygwin prompt

(i) Using the above COM port number, execute the following command at the cygwin prompt

java net.tinyos.tools.PrintfClient -comm serial@COM6:telosb

(j) This should create a display the sensor data at the cygwin window as shown in Figure

47 (sensor readings are shown with decimal values in the range of 0-4095).

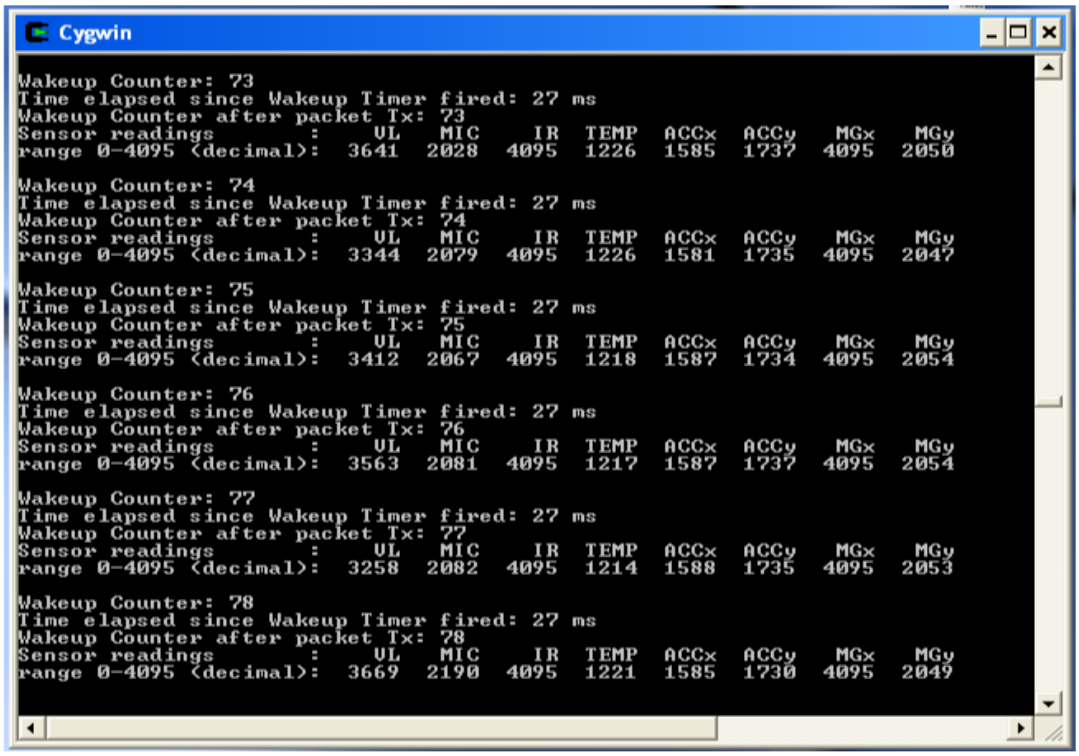

FIGURE 5.2 (SENSOR RESULTS)

FIGURE 5.3 (OPENROV WITH SENSOR SUITE)

CHAPTER 6

CONCLUSION

6.1 Conclusion:

Marine life is so important in human life and has a direct impact on the life cycle. As already mentioned in this report that the marine life and coral reefs facing threats and risks that may lead only natural imbalance.

From this threats, rising of ocean temperatures and lack of sunlight to access the coral reefs. And With the development of modern technologies in our time so as to provide researchers time and effort and Maintain on the lives of divers from the dangers of the sea

We have created an idea to monitoring the factors affecting the life of coral reefs and marine life By using the submarine OPENROV. And the addition of heat and light sensors to record the variables which enables scientists and Researchers find ways to maintain these coral reefs and the marine environment

References:

[1] Assembly instruction in Arabic. [online].http://www.isnaha.com/index.php?option=com_k2&view=item&id=708-الروبوت-الغواص:openrov

[2] Software manual of the board with details on how to install the firmware and softwares related to the board.[online].http://wiki.openrov.com/index.php/Software_Manual

[3]Website for the graduation project.[online].https://sites.google.com/site/uqugraduation/

[4] How to build OpenROV.[online] http://openrov.dozuki.com/Guide/How+to+Build+OpenROV+v2.5/2

[5] Datasheet for SBT80 tiny sensor suite.[online].http://www.alldatasheet.com/view.jsp?Searchword=Sbt80

[6] IEI remote intelligent systems.[online].http://www.ieiworld.com/product_groups/industrial/detail_list.aspx?gid=09049539805266347587&cid=08141364396101700680

Appendix A CODE

SBT Sensor Application code for the module (tinyos compiler):

```
#include "sbt_serial_msg.h"
#include "radio_msg.h"

configuration SbtSensorApp{}
implementation {
  components SbtSensor;

  components MainC;
  SbtSensor.Boot -> MainC.Boot;

  components new TimerMilliC() as Timer;
  SbtSensor.Timer -> Timer;

  components SerialActiveMessageC as Serial;
  SbtSensor.SerialSend -> Serial.AMSend[AM_SBT_SERIAL_MSG];
  SbtSensor.SerialReceive -> Serial.Receive[AM_SBT_SERIAL_MSG];
  SbtSensor.AMControl -> Serial;
  SbtSensor.SerialPacket -> Serial;

  components ActiveMessageC as Radio;
  SbtSensor.RadioSend -> Radio.AMSend[AM_RADIO_MSG];
  SbtSensor.RadioReceive -> Radio.Receive[AM_RADIO_MSG];
  SbtSensor.RadioControl -> Radio;
  SbtSensor.RadioPacket -> Radio;

  components new SBT80_ADCconfigC() as Light;
  components new SBT80_ADCconfigC() as Temp;
  SbtSensor.Light -> Light.ReadADC0;
  SbtSensor.Temp -> Temp.ReadADC3;
}
```

SBT Java Code for Serial Display (Java Application for PC):

```
/**
 * This class is automatically generated by mig. DO NOT EDIT THIS FILE.
 * This class implements a Java interface to the 'SenseToRadioMsg'
 * message type.
 */

public class SenseToRadioMsg extends net.tinyos.message.Message {

    /** The default size of this message type in bytes. */
    public static final int DEFAULT_MESSAGE_SIZE = 20;

    /** The Active Message type associated with this message. */
    public static final int AM_TYPE = 6;

    /** Create a new SenseToRadioMsg of size 20. */
    public SenseToRadioMsg() {
```

```
        super(DEFAULT_MESSAGE_SIZE);
        amTypeSet(AM_TYPE);
    }

    /** Create a new SenseToRadioMsg of the given data_length. */
    public SenseToRadioMsg(int data_length) {
        super(data_length);
        amTypeSet(AM_TYPE);
    }

    /**
     * Create a new SenseToRadioMsg with the given data_length
     * and base offset.
     */
    public SenseToRadioMsg(int data_length, int base_offset) {
        super(data_length, base_offset);
        amTypeSet(AM_TYPE);
    }

    /**
     * Create a new SenseToRadioMsg using the given byte array
     * as backing store.
     */
    public SenseToRadioMsg(byte[] data) {
        super(data);
        amTypeSet(AM_TYPE);
    }

    /**
     * Create a new SenseToRadioMsg using the given byte array
     * as backing store, with the given base offset.
     */
    public SenseToRadioMsg(byte[] data, int base_offset) {
        super(data, base_offset);
        amTypeSet(AM_TYPE);
    }

    /**
     * Create a new SenseToRadioMsg using the given byte array
     * as backing store, with the given base offset and data length.
     */
    public SenseToRadioMsg(byte[] data, int base_offset, int data_length) {
        super(data, base_offset, data_length);
        amTypeSet(AM_TYPE);
    }

    /**
     * Create a new SenseToRadioMsg embedded in the given message
     * at the given base offset.
     */
    public SenseToRadioMsg(net.tinyos.message.Message msg, int base_offset)
{
        super(msg, base_offset, DEFAULT_MESSAGE_SIZE);
        amTypeSet(AM_TYPE);
    }

    /**
```

```
     * Create a new SenseToRadioMsg embedded in the given message
     * at the given base offset and length.
     */
    public SenseToRadioMsg(net.tinyos.message.Message msg, int base_offset,
int data_length) {
        super(msg, base_offset, data_length);
        amTypeSet(AM_TYPE);
    }

    /**
    /* Return a String representation of this message. Includes the
     * message type name and the non-indexed field values.
     */
    public String toString() {
      String s = "Message <SenseToRadioMsg> \n";
      try {
        s += "  [nodeid=0x"+Long.toHexString(get_nodeid())+"]\n";
      } catch (ArrayIndexOutOfBoundsException aioobe) { /* Skip field */ }
      try {
        s += "  [counter=0x"+Long.toHexString(get_counter())+"]\n";
      } catch (ArrayIndexOutOfBoundsException aioobe) { /* Skip field */ }
      try {
        s += "  [data=";
        for (int i = 0; i < 8; i++) {
          s += "0x"+Long.toHexString(getElement_data(i) & 0xffff)+" ";
        }
        s += "]\n";
      } catch (ArrayIndexOutOfBoundsException aioobe) { /* Skip field */ }
      return s;
    }

    // Message-type-specific access methods appear below.

    ////////////////////////////////////////////////////
    // Accessor methods for field: nodeid
    //   Field type: int, unsigned
    //   Offset (bits): 0
    //   Size (bits): 16
    ////////////////////////////////////////////////////

    /**
     * Return whether the field 'nodeid' is signed (false).
     */
    public static boolean isSigned_nodeid() {
        return false;
    }

    /**
     * Return whether the field 'nodeid' is an array (false).
     */
    public static boolean isArray_nodeid() {
        return false;
    }

    /**
     * Return the offset (in bytes) of the field 'nodeid'
     */
```

```
public static int offset_nodeid() {
    return (0 / 8);
}

/**
 * Return the offset (in bits) of the field 'nodeid'
 */
public static int offsetBits_nodeid() {
    return 0;
}

/**
 * Return the value (as a int) of the field 'nodeid'
 */
public int get_nodeid() {
    return (int)getUIntBEElement(offsetBits_nodeid(), 16);
}

/**
 * Set the value of the field 'nodeid'
 */
public void set_nodeid(int value) {
    setUIntBEElement(offsetBits_nodeid(), 16, value);
}

/**
 * Return the size, in bytes, of the field 'nodeid'
 */
public static int size_nodeid() {
    return (16 / 8);
}

/**
 * Return the size, in bits, of the field 'nodeid'
 */
public static int sizeBits_nodeid() {
    return 16;
}

////////////////////////////////////////////////////////
// Accessor methods for field: counter
//   Field type: int, unsigned
//   Offset (bits): 16
//   Size (bits): 16
////////////////////////////////////////////////////////

/**
 * Return whether the field 'counter' is signed (false).
 */
public static boolean isSigned_counter() {
    return false;
}

/**
 * Return whether the field 'counter' is an array (false).
 */
public static boolean isArray_counter() {
```

```
    return false;
}

/**
 * Return the offset (in bytes) of the field 'counter'
 */
public static int offset_counter() {
    return (16 / 8);
}

/**
 * Return the offset (in bits) of the field 'counter'
 */
public static int offsetBits_counter() {
    return 16;
}

/**
 * Return the value (as a int) of the field 'counter'
 */
public int get_counter() {
    return (int)getUIntBEElement(offsetBits_counter(), 16);
}

/**
 * Set the value of the field 'counter'
 */
public void set_counter(int value) {
    setUIntBEElement(offsetBits_counter(), 16, value);
}

/**
 * Return the size, in bytes, of the field 'counter'
 */
public static int size_counter() {
    return (16 / 8);
}

/**
 * Return the size, in bits, of the field 'counter'
 */
public static int sizeBits_counter() {
    return 16;
}

////////////////////////////////////////////////////////
// Accessor methods for field: data
//   Field type: int[], unsigned
//   Offset (bits): 32
//   Size of each element (bits): 16
////////////////////////////////////////////////////////

/**
 * Return whether the field 'data' is signed (false).
 */
public static boolean isSigned_data() {
    return false;
```

```
    }

    /**
     * Return whether the field 'data' is an array (true).
     */
    public static boolean isArray_data() {
        return true;
    }

    /**
     * Return the offset (in bytes) of the field 'data'
     */
    public static int offset_data(int index1) {
        int offset = 32;
        if (index1 < 0 || index1 >= 8) throw new
ArrayIndexOutOfBoundsException();
        offset += 0 + index1 * 16;
        return (offset / 8);
    }

    /**
     * Return the offset (in bits) of the field 'data'
     */
    public static int offsetBits_data(int index1) {
        int offset = 32;
        if (index1 < 0 || index1 >= 8) throw new
ArrayIndexOutOfBoundsException();
        offset += 0 + index1 * 16;
        return offset;
    }

    /**
     * Return the entire array 'data' as a int[]
     */
    public int[] get_data() {
        int[] tmp = new int[8];
        for (int index0 = 0; index0 < numElements_data(0); index0++) {
            tmp[index0] = getElement_data(index0);
        }
        return tmp;
    }

    /**
     * Set the contents of the array 'data' from the given int[]
     */
    public void set_data(int[] value) {
        for (int index0 = 0; index0 < value.length; index0++) {
            setElement_data(index0, value[index0]);
        }
    }

    /**
     * Return an element (as a int) of the array 'data'
     */
    public int getElement_data(int index1) {
        return (int)getUIntBEElement(offsetBits_data(index1), 16);
    }
```

```
/**
 * Set an element of the array 'data'
 */
public void setElement_data(int index1, int value) {
    setUIntBEElement(offsetBits_data(index1), 16, value);
}

/**
 * Return the total size, in bytes, of the array 'data'
 */
public static int totalSize_data() {
    return (128 / 8);
}

/**
 * Return the total size, in bits, of the array 'data'
 */
public static int totalSizeBits_data() {
    return 128;
}

/**
 * Return the size, in bytes, of each element of the array 'data'
 */
public static int elementSize_data() {
    return (16 / 8);
}

/**
 * Return the size, in bits, of each element of the array 'data'
 */
public static int elementSizeBits_data() {
    return 16;
}

/**
 * Return the number of dimensions in the array 'data'
 */
public static int numDimensions_data() {
    return 1;
}

/**
 * Return the number of elements in the array 'data'
 */
public static int numElements_data() {
    return 8;
}

/**
 * Return the number of elements in the array 'data'
 * for the given dimension.
 */
public static int numElements_data(int dimension) {
  int array_dims[] = { 8,  };
```

```
        if (dimension < 0 || dimension >= 1) throw new
ArrayIndexOutOfBoundsException();
        if (array_dims[dimension] == 0) throw new
IllegalArgumentException("Array dimension "+dimension+" has unknown size");
        return array_dims[dimension];
    }

}
```

Code for Makefile (Program to Build the whole project):

```
COMPONENT=SenseToRadioAppC
CFLAGS += -I$(TOSDIR)/lib/printf

BUILD_EXTRA_DEPS = SenseToRadioMsg.class
CLEAN_EXTRA = *.class SenseToRadioMsg.java

SenseToRadioMsg.class: SenseToRadioMsg.java
      javac SenseToRadioMsg.java

SenseToRadioMsg.java:
      mig java -target=null $(CFLAGS) -java-
classname=SenseToRadioMsg SenseToRadio.h SenseToRadioMsg -o $@


include $(MAKERULES)
```