Smart Integrated Tile

Final Report for CSE 237D

Team Member

Liren Chen, lic002@eng.ucsd.edu, A53288236

Xuanyi Yu, xuyu@eng.ucsd.edu, A53267999

Acknowledgment

Prof. Ryan Kastner

Dr. Brian Zgliczynski

Eric Lo

1. Abstract

In this project, we explore different hardware choices in order to develop a cheap, reliable depth sensor device for Scripps Institute of Oceanography. The sensor will log and display depth readings in relatively shallow, high-illumination conditions in coral reef communities. The depth information will then be used in conjunction with photographs to create 3D models of the coral reefs. The current method for collecting this depth information is by manually measuring the depth just above the coral reefs, which is strenuous for the divers and prone to errors. We provide a reliable alternative that automates this process and increases the precision of the measurements. Major design decisions include choosing a method of displaying depth data and how to quickly and easily transfer the collected information while maintaining the integrity of the waterproofing at an affordable cost.

2. Introduction

Background - The 100 Island Challenge is a collaborative effort based at Scripps Institution of Oceanography to describe the variation of coral reefs from across the globe. Scientists combine classical field surveys with innovative imaging and data technologies to archive reefs digitally and watch how populations change through time[1]. In order to achieve this goal of 100 Island Challenge, they are monitoring coral reef communities by scanning and constructing 3D models in long time series. To calibrate these models, the depth information is a must. However, manually measuring the depth information just above the coral reefs is strenuous for the divers and prone to errors. A device that can automate the depth measurement process will greatly facilitate the efficiency of collecting data.

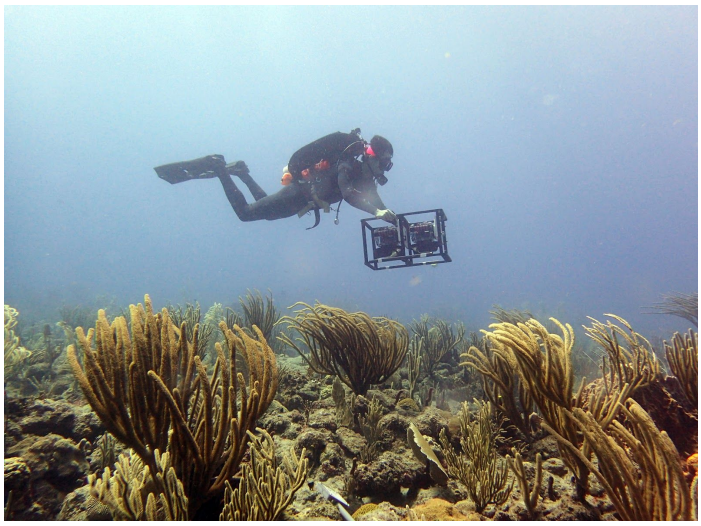


Figure 1: Divers were scanning coral reefs underwater.

Requirements - Researchers from Scripps Institution of Oceanography wish to work with this device under the following conditions: Bring the device on a boat and turn it on before going diving with the corals. Then they dive down and place the devices on the coral reef, and the device starts collecting depth information and displaying it on the screen. The divers swim back and forth to do the photomosaic survey, capturing images of the devices and the data shown on the screen. After that, they collect the devices and return to the boat. They read the data from the devices and download it for backup. Finally, they turn off the device and charge the device overnight.

Objectives - Based on the requirements from Scripps Institution of Oceanography, we are building a device that can measure and display depth information when it's placed on coral reefs. Then the depth information could be captured by cameras and retrieved from images. The other detailed functions are described as follows.

- Switch: Since researchers want this device to work only underwater to save the battery as much as possible, a switch is needed to make sure the device can turn on before diving and turn off when it's back to land.

- Waterproof: Since the device needs to be working under 10-15 meters of salt water for around 4 hours, it should be strictly waterproof to avoid erosion. This is the most pivotal part of this project. Since without practical waterproof design, the device cannot work stably for a long period. Waterproof also means that there may not be any buttons or plugs on this device.

- Clear display: The device works in relatively shallow, high-illumination conditions in coral reef communities. There will be sunlight, wind, and wave changing over time. The display module should display the depth information as clear as possible under different conditions to make sure the information can be retrieved from the images accurately.

- Rechargeable: The device needs to be reusable and rechargeable. Since the device is strictly waterproof, wireless charging is a better option compared to using plug and cable. The battery needs to be working for around 4 hours and can be charged fully over one night. In addition, there should be a strategy such as LED to indicate how much battery left.

- Data storage: There may be cases that researchers fail to retrieve accurate depth information from unclear images. For backup consideration, this device should be integrated with a data storage module such as a SD-card.

- Data Transmission: In order to download the depth data, a reliable wireless data transmission module is needed. The device should be able to transfer depth data to computer, and the computer should be able to download data from multiple devices.

3. Design Decisions

Previously, Dan Sturm and Robert Barlow built the very first prototype (namely Ver.0) for Reef Pin project during summer 2018. They worked for 10 weeks full-time for this project. They are pioneers and did an amazing job. To continue with the project, we did borrow some design ideas from them. Instead of simply replicating their work, we explore a wide range of hardware choices and develop the software from scratch.

To speed up the prototyping process. We are building prototypes with breakout components as possible. Since our design decisions might change rapidly, we want to keep components modularized for unit testing and fast assembling.

Price, power consumption, waterproof, and reliability are the most important factors during making design decisions. Normally, it should be a trade-off between these factors. Our goal is to find a sweet spot that could meet all of the requirements.

MCU - We choose STM32F103C8T6 as our microcontroller. The other two choices are MSP430 and Arduino. MSP430 has the lowest power consumption. Arduino the easiest one for software development. We choose STM32 because it is the most powerful one. Since we might need I2C bus, SPI bus, and multiple USART ports. STM32 has enough pins to connect to a wide range of peripherals. The price for each STM32F103C8T6 breakout board is about 5$. The power consumption during runtime is about 50mA, which is acceptable.

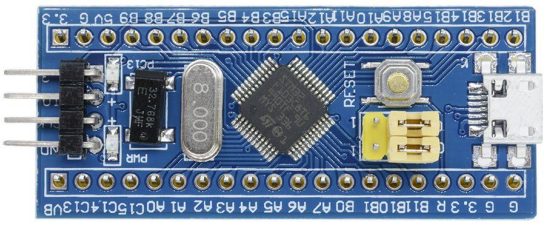


Figure 2: STM32f103c8t6 breakout

Display - In Ver.0, Dan and Robert use 7-segment LED to display depth and heading information. However, since there is no test record for Ver.0, we didn't know how the LED perform underwater. In our first prototype (namely Ver.1) we want to integrate a LED display and test it underwater to see if it's bright enough for data reading. We choose Adafruit 1.2" 4-Digit 7-Segment Display since it's the largest modular display with I2C bus which we could find in Amazon. We also explore other kinds of displays. E-ink display has ultra-low power consumption and it can passively reflect sunlight. We also integrate a 2.9 inch E-ink module in Ver.1.

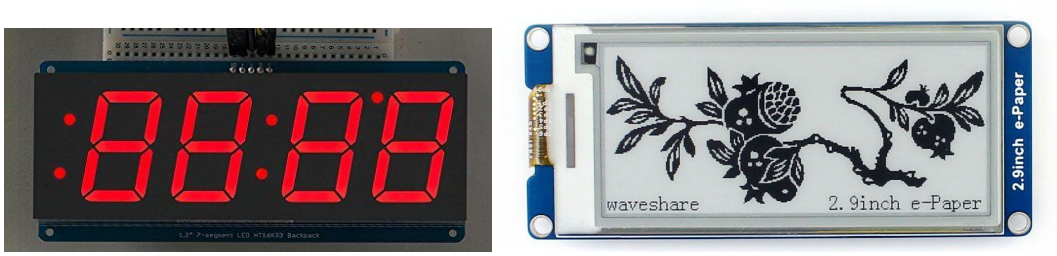


Figure 3: LED and E-ink display module

Depth Sensor - We use MS5803-14BA pressure sensor. This sensor is accurate and durable. Theoretically, it could withstand about 130 meters of water, which is sufficient in coral reef areas. In Ver.1, we use a breakout bought from Amazon. Because the breakout part 20$ more expensive than the raw sensor, we assembled our own breakout parts with PCBs since our second iteration. The red one above is bought from Sparkfun, the other two are assembled by us.

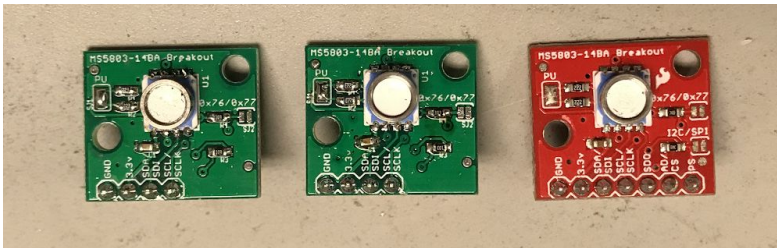
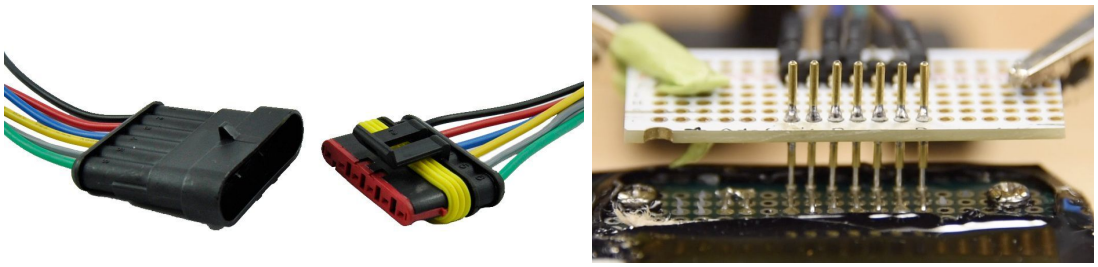


Figure 4: Pressure sensor breakouts

Waterproof Connector - In Ver.0, they didn't expose any pins. To make the device reprogrammable, we need to figure out a solution for a waterproof connection. In Ver.1, we tried waterproof connectors for automobiles. These connectors have rubber seals to prevent water, but they are not rated for diving. This is a temporal solution for our first several underwater tests. In Ver.2, we tried to use pogo pins to reprogram the device. Several copper pads are exposed on the surface of the device. Pogo pins have springs inside, and they can tap on the pads on the device. For underwater usage, copper pads are fully covered by hot glue to prevent erosion.

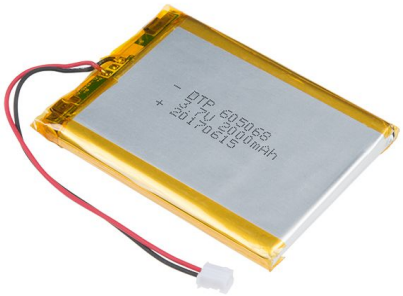


Ver.1 Connector *Ver.2 Pogo pins*

Figure 5: Potential choices of connectors

Battery - In Ver.0, they are using 2Ah Li-ion batteries. Those batteries are very compact and nicely built. However, they are around 13$ each. We choose 18650 Li-ion batteries instead. It's relatively bulky but is only about 5$ each. The capacity is about 2-3Ah depending on different

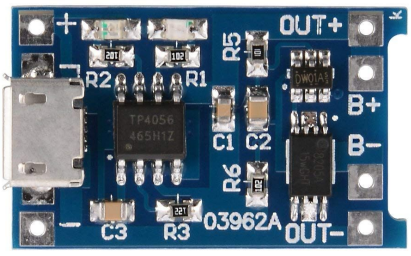
brands. Generally, it could power the device for running 10-20 hours. The batteries we are actually using are bought by Eric 5 years ago, which is still functioning.



2Ah Li-ion *3.4Ah 18650 Li-ion*

Figure 6: Potential choices of battery

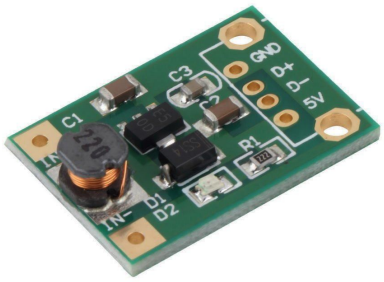
Charging - We use TP4056 breakout to charge and protect the battery. The input of TP4056 should be 5V. In Ver.1, we wired out the charging port through the waterproof connector. In Ver.2, we use the Qi receiver to power the charging chip. The Qi an open interface standard that defines wireless power transfer using inductive charging. The receivers are bought during last summer. They are about 15$. Potentially, they could be replaced by more inexpensive alternatives in the future.

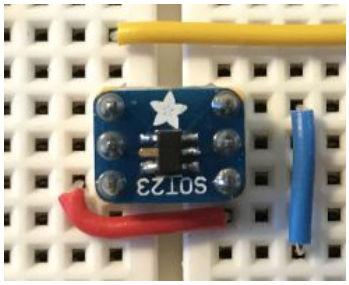


TP4056 breakout *Qi receiver*

Figure 7: Parts in charging module

Power Design - The MCU breakout is powered by 5V or 3.3V. We use a step-up regulator to raise the battery voltage to 5V. When the device is idle, we want to use as low power as possible. The MCU could turn into standby mode, in which the current is around uA. During building Ver.1 and Ver.2, we haven't figure out the cut-off circuit, the sensors and displays are always powered. On the breadboard, we tested linear dropout regulator, which is controlled by enable pin. Then, the MCU could use GPIO to control the power for sensors.





Step-up regulator *Linear dropout regulator*

Figure 8: Parts in power module

4. Implementations

a. Hardware Assembly

Generally, we built two potted prototypes and a test prototype on a breadboard.

First Prototype (Ver.1) - This prototype includes MCU, pressure sensor, LED display, E-ink display, RTC module, battery, step-up regulator, and reed switch. First, the components are assembled on a breadboard to test the integrity. The power consumption during runtime is 114mA, which is measured by 3.7V power supply. On the breadboard, each module is tested to make sure they are functioning. Then, all parts are soldered on a prototyping board. On this board, all wires including power, I2C, SPI, are soldered with wires. There are 6 wires connect to the outside: GND, VCC, SCLK, SDIO, TX, RX. VCC is using for charging. SCLK and SDIO are used for reprogramming the MCU. TX and RX are for data transmission. Soldering all of these wires requires decent soldering techniques and tons of time. Since space is very tight, all components need to be carefully placed to make sure there are no conflicts. With the prototyping board, there is much flexibility, but any wrong manipulation may lead to serious consequences. If time permits, it's better to make a PCB instead.

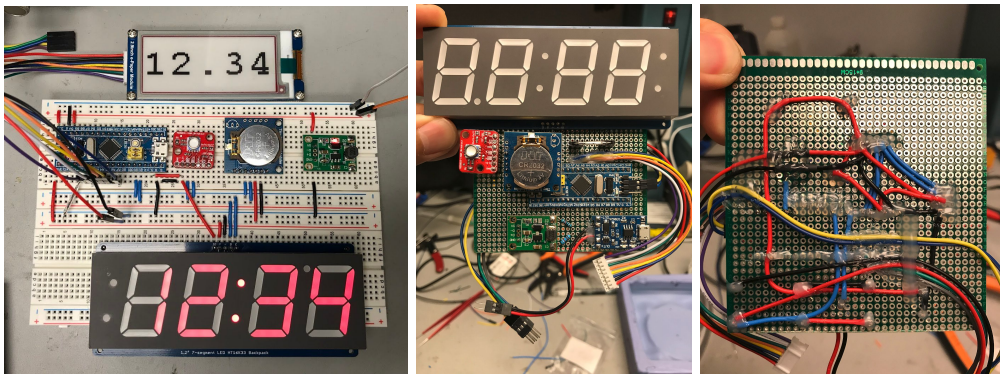


Figure 9: Test with the breadboard, assemble with prototyping board.

After soldering, the device is put in the mold. The mold is made by the previous group. We really like the design for the bottom of the device. Before potting, to make LEDs inside visible, we covered LEDs with hot glue. We use plastic tapes to make a cylinder around the depth sensor. LED display and E-ink display is also surrounded by tapes and hot glues to prevent covered by

epoxy. The above techniques completely covered the displays and depth sensor. They also enable us to pour more epoxy above the height of displays and sensor. Actually, we made a mistake here. The RTC module is installed higher than displays. We have to pour more epoxy than expected to fully cover the battery for RTC.

Figure 10: Potting process

After the epoxy is cured, redundant tapes and epoxy around displays and sensor are removed. The overlook of the device is as follows.

Figure 11: Overlook of 1st iteration

Second Prototype (Ver.2) - This prototype is built after the failure of the first prototype. Compared with the first one, we removed the LED display, RTC module, and waterproof plug. We use the RTC module on STM32 chip and pogo pins instead. A Qi receiver is integrated into the device. We didn't put the Zigbee module inside because we hadn't figured out the LDO at that moment. If the power of Zigbee is not cut off, the standby power consumption will be around 20mA, which is too high. The main purpose of this prototype is to verify the pogo pin connection and further test E-ink display. If it works reliably, it could be delivered to divers for deeper underwater tests.

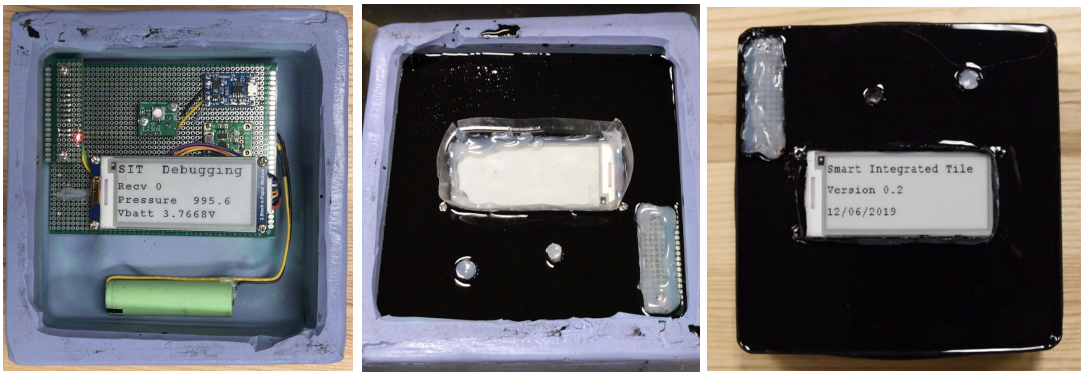


Figure 12: Potting process of the 2nd iteration

Breadboard Prototype - This is our debugging and testing platform, integrated with SD card, Zigbee module, and LDO module. This prototype is closest to our final design with maximum functionality.

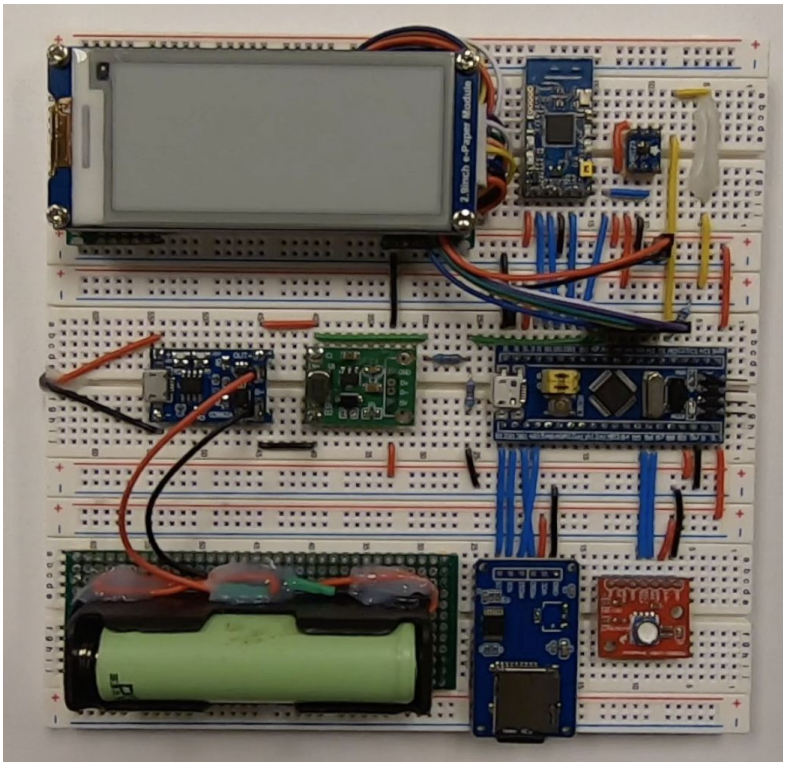


Figure 13: Breadboard Prototype

b. Software Development

Overview - The software development consists of two parts, one is STM32 embedded software, another is a GUI running on PC for data transmission purpose. Our embedded software is located under ./stm32 folder. Since the schematic for two prototypes is different, the code for each prototype is located in folders ending with different version numbers. The code for GUI is located in ./GUI folder.

IDE & Toolchain - To develop STM32 software, the most popular IDE is Keil. Since it's a very expensive commercial software, we use SW4STM32 IDE instead. SW4STM32 is an official free software produced by STM32. It is compatible with both Windows and Mac. We use ST-Link to download software into the embedded chip. With ST-Link and SW4, we can use step-by-step debugging.

Library - Our embedded software is fully dependant on STM32 Standard Peripheral Libraries. This library is older than HAL library. Basically, these two libraries has the same functionality, we choose Std Peripheral libraries simply because we are more familiar with them.

Basic Components - In general, we use GPIO, USART, I2C, SPI, ADC, EXTI, PWR, and RTC in STM32 Std Peripheral libraries.

Structure - The driver code for each module is developed separately. The code for depth sensor, SD card, and E-ink module are developed based on their Arduino libraries. Most of the breakout sensors are provided with Arduino libraries that could instantly plugin. Basically, we are rewriting these libraries with STM32 backend. Most of the user logic is defined in main.c file. The embedded software is modularized and extendable for future teams.

GUI - The GUI is based on Tkinter. Tkinter is the standard GUI library for python. With the GUI, users can connect to the device, change device settings and download depth data. We want to make the software as lightweight as possible. The program use pySerial to talk to the serial port. Data are transmitted through the serial port, Zigbee module, and finally received by the other Zigbee module in the device. We use multi-threading to listen to receiving buffer for pySerial. For now, the GUI is not fully developed. We've finished the first layer of the data transmission protocol. With this protocol, data could be sent by frames. Noise information from other Zigbee devices is eliminated. The command set needs to be further developed based on users' demand.

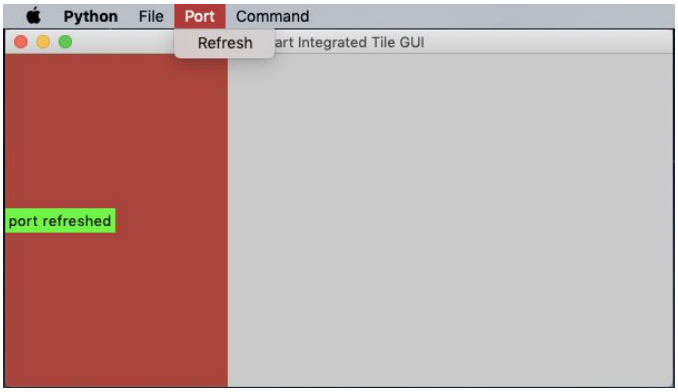


Figure 14: Simple GUI based on Tkinter

5. Milestones

We anticipate having two iterations during this quarter. In each iteration, we are building one deliverable and testable prototype.

First Iteration - During the first 4 weeks, we've built the first fully integrated prototype. It has a depth sensor, LED display, E-ink display, real-time clock, and battery monitor system. The device is reprogrammable and rechargeable by using sealed connectors. We've tested it in a swimming pool, recorded videos to compare the performance of two different kinds of displays.

Milestones Status and Tasks Accomplished:

Deliverable	TimeLine	Status
Design whole schematic, including battery management module, switch module and display module.	05/07	Done
Build an experimental prototype on a breadboard and test it	05/08	Done
Writing code for STM32 to communicate with peripherals: SPI/I2C Interfaces for depth sensor and display, standby mode logic	05/07	Done
Build prototype with soldering board and pot it	05/13	Done
Underwater test	05/16	Done

Problems - After running for about 2 weeks, our first prototype died suddenly. Unfortunately, this bug is not diagnosable and reproducible. Since the device is totally potted with epoxy, we cannot reopen it to debug the hardware. This accident greatly changed our plan for the rest of the time in this quarter. We have added some tasks related to diagnosis for the first iteration.

Second Iteration - We finished the second prototype during the rest of the quarter. Generally, we have made several improvements compared to the first iteration, integrating wireless charging module, wireless data transmission, and data storage module additionally. The design decisions for the second prototype largely depend on the test results of the first one, we chose to use E-ink module as our display. Because of more complexity, more uncertainty is inherited in the second iteration. We have changed the plan during our development.

Milestones Status and Tasks Accomplished:

Deliverable	TimeLine	Status
Design additional schematic, including updated battery management module, wireless charging module, pogo-pin reprogramming strategy, and wireless data transmission.	05/30	Done
Build an experimental prototype on a breadboard and test it for a long time	05/30	Done. Since our 1st prototype only survived for 2 weeks, we want to test the 2nd prototype on the breadboard longer for durability test.

Build a prototype with soldering board and pot it	06/03	Done
PCB design for depth sensor breakout and main board	06/06	We have finished the PCB design for depth sensor breakout and handed it over to manufacture, it works well. As for the main board of device, we have finished a version of PCB design. But since we have tested the 2nd for only limited times, this version of PCB design may need further revision after long period underwater tests.
Design data transmission protocol and the interface between computer and device	06/07	We have finished the protocol for transferring a frame of data. For now, the protocol doesn't support transferring large files.
Underwater test	06/06	Done

6. Conclusion

In this project, we built a waterproof device that can measure the depth and display it on the screen clearly based on the requirements from Scripps of Institute Oceanography, used for achieving the goal of 100 Island Challenge. In order to design a reliable device, we have compared multiple choices for each requirement and finalized our design based on the test result. In this quarter, we have built two iterations combining different hardware choices. The 1st iteration focused on comparing two display module as well as integrating the core depth measurement module. The 2nd iteration has additionally integrated with wireless charging and wireless data transmission revised the previous design of some modules based on the test result of 1st iteration. We have tested both prototypes in the lab and underwater, tried to mimic the real working condition of the device. In the future, this project will continue to build a more robust and more durable version, and the device will be handed over to researchers to have it tested in the ocean.

7. References

[1] Understanding the World's Coral Reefs http://100islandchallenge.org/