Continuing Improvements of the Sharkduino Animal Tag System

A thesis submitted in partial fulfillment of the requirement
for the degree of Bachelor of Science with Honors in
Physics from the College of William and Mary in Virginia

by

William E. Laney

Accepted for Honors

Advisor: Prof. Wouter Deconinck

Prof. Kevin C. Weng

Prof. William E. Cooke

Prof. Gina L. Hoatson

Williamsburg, Virginia
December 2017

Contents

Acknowledgments

First and foremost, I would like to express my sincere gratitude to my major advisor Prof. Wouter Deconinck and my VIMS advisor Kevin Weng for their continuous support of my research, and for their patience, motivation, enthusiasm, and immense knowledge. Their guidance helped me in all aspects of research and writing of this thesis.

I would like to thank my labmates Abby Bilenkin, Ben Powell, Dara Kharabi, Hanqiu Peng, and Seanna Nam for all of their help and insight, and for tolerating my meetings.

Finally I would like to thank my family: my parents Judy and Hyrum, and my brother Joshua for supporting me in every aspect of my life.

List of Figures

List of Tables

Abstract

Much about the movement and habits of marine animals is still unknown. This information can be found for terrestrial animals through direct observation or video surveillance; however, these optical methods are not effective in aquatic environments. To circumvent this problem, researchers "tag" aquatic animals with small sensor systems that record data about the animals movements. The researchers then recover the tag and use signal analysis to draw conclusions about the habits of the animal. Unfortunately, commercially available tags are expensive and most can only be used once. We created an inexpensive and reusable tag to be used on sandbar sharks held in captivity at the Virginia Institute of Marine Science (VIMS) and VIMS Eastern Shore Lab (ESL). The tag consists of a three-axis accelerometer and a three-axis gyroscope for movement information as well as a real time clock (RTC) for timing. The data is written to microSD card. This is all run through a 3.3V Arduino Pro Mini microcontroller and powered by a lithium ion polymer (LiPo) battery. Initial prototypes were built using commercially available components connected through point-to-point wired connections. While creating the prototype, size, form factor, and power consumption were serious concerns. Once it was confirmed that these prototypes functioned as expected, we created custom printed circuit boards (PCBs) in order to condense the tag into a smaller package for deployment. Work then proceeded to refine and improve the PCBs both in terms of form factor and power efficiency.

Chapter 1

Introduction

The purpose of this project was to create a cheap and versatile underwater animal tag. Originally the tag was intended to act primarily as an accelerometer with water pressure and temperature reading capabilities. We later decided to expand the system to incorporate a gyroscope. This gives more detailed information about the movement of the animal. The tag was designed for use primarily on sandbar sharks (*Carcharhinus plumbeus*) in the Chesapeake Bay; however we attempted to design it in such a way to make it widely applicable to many different types of aquatic animals in varied environments. In addition to needing to function underwater, the tag has been designed to minimize power usage, memory usage, size, and weight. This was done both to maximize the deployment time of the tag and to minimize its interference with the animal's behavior and habits.

Tags such as these already exist commercially; however they are expensive, costing hundreds to thousands of dollars. Additionally, most are single-use because they do not have rechargeable batteries. This project hopes to overcome both of these limitations by creating a tag that can be reused and is available at a lower price point than that of commercially available tags.

Work has additionally been done to devise procedures for rapid waterproofing of the tags for use on live animals in captivity. It is important to have a reliable way

to test new tag hardware on animals without expending the time and effort that would be required to waterproof a tag in a way that would be adequate for long term deployments in the wild.

Chapter 2

Previous Work

2.1 Sharkduino V1.0

During the summer of 2016 work was done to move the Sharkduino system from prototypes on solderable breadboards onto a custom printed circuit board (PCB) designed in house. This was successfully completed. This first PCB design was dubbed Sharkduino V1.0. It consisted of a 3.3V Arduino Pro Mini microcontroller stacked on top of two PCB boards. The top board, designated Layer 1, had a MM8452 3-axis accelerometer, a L3GD20H 3-axis gyroscope, and a push-push type microSD card slot [1], [2]. The second PCB board (Layer 2) had a DS3234 real time-clock (RTC), a MCP7381 lithium polymer (LiPo) battery charger, a microUSB port, and a JST connector [3], [4]. The JST connector was used to connect a LiPo battery to the board and the microUSB charged the LiPo without the need to unplug it. Layer 2 was designed to stack underneath both Layer 1 and the Arduino. Finally a Layer 3 board was built, but it did not follow the stacking design of the other two layers. It consisted of an MS5803-14BA pressure sensor and a TMP102 temperature sensor [5], [6]. Since these sensors would be exposed directly to the water, they were put on their own small board, which could be physically isolated from the rest of the system. This simplified the construction of waterproofing for the system.

This stacking design gave the system a cube-like form factor which helped to minimize the size of it in all three dimensions. The size of the device was approximately 50x18x22mm, not including the battery (Figure 2.1). The size of the battery is chosen to be the largest possible without the device becoming too large or heavy, which could interfere with the animal's behavior [7].

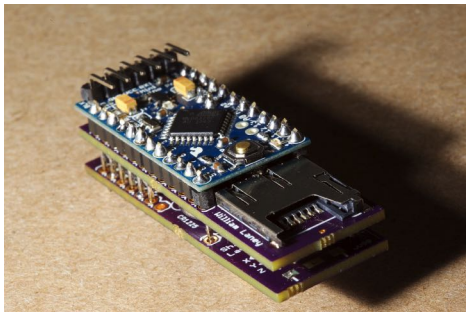

Figure 2.1: Sharkduino V1.0

After the design work was completed, we built two Sharkduino V1.0s. We deployed them on sandbar sharks at the Virginia Institute of Marine Science (VIMS) Eastern Shore Lab (ESL) in Wachapreague, Virginia. These animals were caught in the wild and held in captivity in a large tank at the ESL. An image of the tag on an animal at the ESL can be seen in Figure 2.2.

Through the assembly and subsequent deployment of the systems, we identified both flaws and desired improvements in the design. First we found that the charging circuit for the LiPo battery did not work properly. The primary cause of this problem was that the microUSB port was not firmly attached to the board by the solder alone. This caused it to become detached from the board due to the force used to remove the plug from the port. We mitigated this issue on the second Sharkduino V1.0 that we assembled by placing hot glue around the connector, and attempting to be more gentle when plugging and unplugging from the port. This did not fix the charging circuit problems, however, as we found that the indicator light emitting diode (LED)

attached to the charging circuit would not turn on. This made it impossible to determine when the battery was fully charged without disconnecting it from the system. We believed that the problem was caused by the LED itself and not by the circuit, as the circuit was adapted from a known good design. We were not able, however, to continue to troubleshoot the charging circuitry at that time. The reason for this was that we were under significant time constraints due to the release schedule of the animals, and the charging circuitry was not integral to the data recording capabilities of the device. It was possible to charge the battery off board and plug it in, so we used that workaround. We did examine the charging circuit more closely later; a more in-depth analysis of the problems related to it is presented in Chapter 5.

Figure 2.2: Sharkduino V1.0 Attached to a Sandbar Shark
Animal research was conducted under IACUC-2016-03-21-11000-kcweng

Another problem that we knew about before deployment was battery life. From our preliminary tests and calculations on data sheets it appeared that we would only be able to record data for 2-3 days. This was less time than what is required for deployment in the wild, since the animals behave atypically for 2-3 days after being tagged. The possible solutions to this, without new design work, would be use a larger battery or to program the device to not turn on until a few days after it was deployed.

The remainder of the issues we discovered with the device were found during the deployments on animals. The first and most serious of these was that the microSD

card could easily be ejected from the slot while the tag was attached to the animal. Once this happened no more data would be collected because there was nowhere to store the data. This prevented us from acquiring long data sets of animal movement. The longer the device was on the animal, the higher the chance that the card would become dislodged. This issue was caused by the type of microSD card slot we were using, and the placement of the slot on the board.

The next issue we found had to do with the solder holding the two layers and the Arduino together joints on the board. The joints presented sharp edges that could puncture the heat shrink tubing that we were using as waterproofing. We mitigated this issue by covering the device in electrical tape. The waterproofing procedure will be discussed in greater depth in Chapter 4.

Finally we found that the form factor of the device was not ideal. The device was tall and thin. This made it stick far off the fin of the animal, interfering too much with the animals swimming pattern. We determined that it would be better for the device to shorter and wider [8].

2.2 Sharkduino V2.0 Design

Based on what we found over the summer, work began in Fall 2016 to improve the design. We identified a new gyroscope, the FXAS21002 3-axis gyroscope, that would cut the power consumption of the system by nearly 50%, and created prototype designs for it [9]. We also purchased a wide variety of new microSD card slots to test. We then determined which ones would hold the microSD card most firmly while still giving good access to the card, even after possible waterproofing. Additionally we identified and created designs for a new RTC (the DS1339B) that was significantly smaller than our previous one, and used the I^2C communication protocol [10]. This put all the components, except the microSD card (which uses SPI) onto the same

protocol. This simplified our code base. These changes culminated in creating the design for a new Sharkduino system with a new form factor. The PCB boards for this new system were ordered at the end of the Fall 2016 semester.

Chapter 3

Design Revisions and Sharkduino V2.0 to V2.1

3.1 Sharkduino V2.0 Assembly

At the beginning of the Spring 2017 semester the first task was to assemble and test the new Sharkduino V2.0 boards. These boards contained three new integrated chips (ICs), and their surrounding circuitry. We had worked with all these chips previously on test boards for the independent chips but this larger board was significantly more complicated. We ordered solder stencils to help with applying the solder paste. The solder stencil is a stencil that lays on top of the PCB and solder paste is applied to it. The solder paste is then spread over the stencil, applying it only to the copper pads that need it. After this the components are properly placed onto the board and the board is then placed in the solder reflow oven to harden.

While using the solder stencil was easier than applying all of the solder paste by hand, it did present its own set of challenges. When using the stencil, the PCB, stencil, and work space all have to remain steady. The solder pads are very small and close together, especially for the integrated chips, so even small shifts would bring the alignment of the board and the stencil out of acceptable tolerances. This problem was solved by using a jig around the board, taping the solder stencil to

the jig, and then taping everything to a sturdy table. Even with these precautions the stencil could still become missaligned due to slack in the taping. Additionally the stencil was made out of polyimide film so it could easily become crumpled or bent. Due to these problems, we instituted a procedure where we used the stencil to apply paste to the most difficult integrated chip pads first and then worked outwards to the less delicate pieces. The difficulty of placing solder paste for the integrated chips is determined by the spacing of the pads in the chip package, and the distance between the chip and the nearest component. When we noticed that the stencil was coming out of alignment, we would simply remove it and apply the solder paste to the remaining, easier components by hand (for an example of the use of a jig and stencil see Figure 6.1). A fully assembled Sharkduino V2.0 can be seen in Figure 3.1.

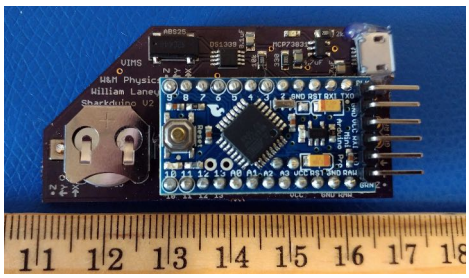

Figure 3.1: Sharkduino V2.0

3.2 Revising Sharkduino V2.0 into V2.1

After assembling three Sharkduino V2.0s we identified that we could upgrade the accelerometer on the board from the MM8452 to MM8451. Both of these accelerometers are in the same family and have the same electrical characteristics. There are two main differences between them. The first difference is that the MM8451 has a first in, first out (FIFO) buffer [11]. This allows us to store accelerometer reads on the accelerometer chip, and helps to increase the time between writes to the SD

card. Writing to the SD card is the most power intensive process on the board, so increasing time between writes increases the battery life. The other difference between accelerometers is price: the MM8451 is $1 more expensive than the MM8452. We decided that the added power efficiency from the FIFO buffer was worth the price increase, especially considering that no new engineering work needed to be done to switch to the MM8451.

We then assembled two Sharkduino V2.0 boards with the MM8451 accelerometer put on it. We called this Sharkduino V2.1. Even though it was the same PCB, it had a different sensor on it than the V2.0 and thus necessitated a version increase. We tested these new V2.1s and found that they worked as expected.

Chapter 4

Rapid waterproofing

One of the problems we faced over summer 2016 was waterproofing tags in a quick and easily reversible way. Most electronics that are used in marine science research are waterproofed with a technique called potting. This is when the electronics are fully encased in epoxy to waterproof them. This is a difficult and time consuming process, especially for devices with complex topographies, because no air bubbles can be left in the potting. If air bubbles are present, as the animal goes deeper, the pressure differential between the enclosed air and the ocean will cause the potting to fail.

Potting is additionally difficult or impossible to undo, and makes it difficult to access ports on the electronics without significant pre-planning during the potting procedure. When working with short deployments of a device in a tank, potting is not necessary. The tank is not deep enough to lead to serious pressure concerns, and since the deployment is short there is less time for wear on the waterproofing to accumulate and cause failure. Finally, potting is a time consuming process and it is not desirable to have to spend so much time waterproofing for a short deployments.

For these reasons we began to develop a technique to rapidly waterproof tags for short term (less than one week) deployments in the tank during summer 2016. The goal of this waterproofing was to be fast, cheap, and easy. This was achievable

because we did not have many of the concerns associated with longer deployments, including the possibly of the tag going into deep water. For these reasons we began to develop a procedure in which we put the tag into heat shrink tubing and sealed both ends with a heat source. Over the summer we used this technique; however we found that it was prone to error. During the Spring 2017 semester we conducted research into refining our heat shrink tube sealing techniques to be more reliable and controlled.

We determined that in order to create a good seal on heat shrink tubing, the tubing needed to be clamped between two flat heated surfaces. This would create a large smooth sealed area on the tubing. Heat shrink normally begins to contract at around 120°C, so the heated surfaces would need to be heated to that temperature. For an example of a good seal see Figure 4.3.

4.1 Impulse Sealer

For our first attempt at a new sealing procedure we experimented with using an impulse sealer. An impulse sealer is a device used to create airtight seals on cellophane bags. It consists of an arm that swings down onto a heated bar (Figure 4.1). The idea is to put the cellophane bag on the heated bar and then push the arm down on top of it to form the seal. When trying to apply this to the heat shrink tubing, however, we ran into various problems.

The first problem was that the heated area on the impulse sealer was smaller than desired. We wanted the sealed part of the tubing to be 2-3cm in length. The heated bar was only a few millimeters long. We tried to develop a procedure where we slowly sealed down the required length, but a second problem made that impossible.

This second problem that we faced was that the impulse sealer would automatically turn off after about one second. This is a safety measure built into the sealer.

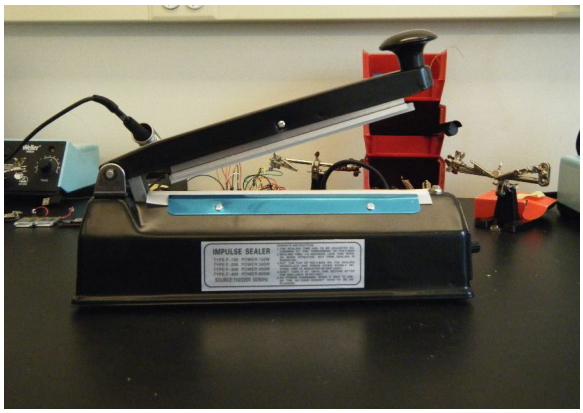

Figure 4.1: Impulse Sealer

The time of one second was chosen because that is about how long it takes to seal a cellophane bag. Sealing the heat shrink tubing takes much longer, possibly more than a minute. It may have been possible to alter the sealer and disable the timing circuitry but this, coupled with the first problem, and the fact that only one side of the tubing was heated by the sealer, led us to abandon the impulse sealer.

4.2 Flat Iron

After more brainstorming we realized that there is a common household item that is capable of creating the type of seal we needed. A flat iron, traditionally used for straightening hair, consists of two flat heated plates that come together with each other. Furthermore a standard flat iron width is about 2.5cm, and they are often temperature adjustable up 235°C. This width of heating surface and temperature control are ideal for creating the type of seal needed on the tubing.

We created a procedure where we set up the flat iron in a clamp, put the heat shrink tubing into the iron, and then closed the clamp on the tubing for around 30 seconds. We then disconnected the flat iron from power and let everything cool with the clamp shut for about 30 minutes. A picture of this set up can be seen in

Figure 4.2. We needed to let the tubing cool while still clamped because when we removed the tube from the clamp while still hot the seal pulled apart during the cooling process. We did try to develop a system where the hot tubing was removed from the clamp and dipped into an ice bath. This led to waterproof seals, but the seals did not appear as full so we decided this faster sealing time was not worth the increased risk. A picture of these seals can be seen in Figure 4.3.

Figure 4.2: Flat Iron Heat Shrink Tube Setup

We tested multiple tags sealed with this technique in water. The first test was to simply put running water over the tag and see if any leakage occurred. We then sealed paper towels in heat shrink tubing and sunk it about 15cm in Lake Matoaka, where we left it for three days. Finally we made three tubes with paper towels in them, and one tube with a functioning Sharkduino. We sunk the three paper towel tubes about 90cm in the water of the VIMS dock. We left the tube with the functioning Sharkduino floating on the surface of the water. When we retrieved these tubes after about two weeks there was no leakage in any tube. Finally during the summer of 2017 we deployed five Sharkduino tags waterproofed using this method in a tank

environment for 1-2 weeks each. We observed no failures during this time. This leads us to conclude that the our heat shrink tube sealing procedure is adequate for use in the tank environment.

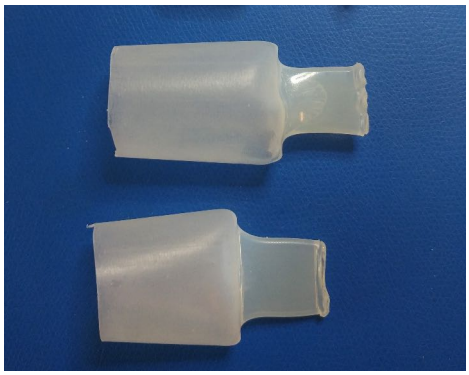

Figure 4.3: Heat Shrink Seals
Top: Seal from flat iron ice dipping procedure
Bottom: Seal from flat iron cooling procedure

Chapter 5

LiPo charging issues and load sharing

5.1 Identification of the Problem

A long term issue on the tag was that the microUSB port and MCP7381 LiPo battery charger circuit did not function properly for various reasons. In the early versions of the tag this was due to assembly errors. These errors included things such as the LED indicator being installed backwards or the microUSB port not being firmly attached to the board. However, even after these issues were worked through, the charging circuity still seemed to behave erratically.

The major sign of the LiPo charger performance issues was the indicator LED. It did not turn on or off in a predictable manner. After assembly errors, the most obvious possible reason for this was that the diode drop voltage on the LED was too low. This would have led to the LED being on even when it should not have been, because the MCP7381 was still outputting a small amount of current from the charge indicator pin. The MCP7381 data sheet does not specify an expected diode drop from the connected LED [4]. It also does not clearly indicate what the expected output voltage should be in various states. We were able to rule out this possibility by experimentation and through consulting other designs.

After ruling out issues with the LED, we determined that the battery was never truly finishing a charge cycle. The MCP7381 was not fully charging the LiPo due to a load sharing issue between the LiPo and the rest of the system. When the microUSB cable is attached to the system, the power from the cable goes into the MCP7381, and then the regulated power exits from the MCP7381 to both the LiPo battery and the power regulator on the Arduino Pro Mini. The Arduino power regulator in turn outputs regulated power to the rest of the system. This sharing of the regulated power from the MCP7381 between the Arduino and the LiPo is called load sharing and was the cause of the charging issues.

LiPo batteries use a very specific charging curve that is regulated and outputted by the MCP7381 [4]. The MCP7381 knows which step in the charging curve to use by looking at small fluctuations in the current being drawn from it by the battery. When the Arduino was drawing current in addition to the battery, these current readings were wrong. This prevented the MCP7381 from fully charging the battery.

5.2 A Stop Gap Solution: Sharkduino V2.2

Since there was no time to resolve this issue before summer 2017 a new version of the tag was made that did not include any LiPo charging circuity. We wanted to have a large number of tags ready for the summer because that is the only season when we have access to sandbar sharks. This version of the tag was known as Sharkduino V2.2. It removed the MCP7381 and the microUSB port along with their associated circuitry. It also had the JST two-prong connector on the top of the board where the microUSB port formerly was. Finally it had a hardware interrupt broken out from the gyroscope. This version of the tag was not to be developed further, because future versions of the tag would require the LiPo charging capabilities. An assembled Sharkduino V2.2 can be seen in Figure 5.1. Further discussion on the new techniques

used to assemble the Sharkduino V2.2s can found in Chapter 6.

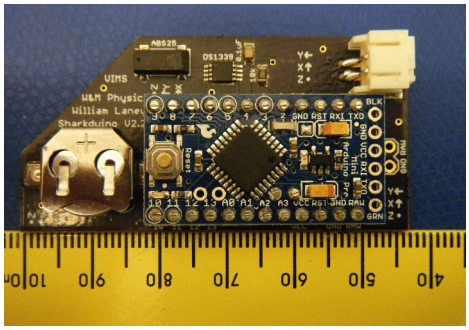

Figure 5.1: Sharkduino V2.2

5.3 A More Permanent Solution

With the identification of load sharing being the root cause of the LiPo battery charging issues, we began work to address the problem. The need to charge a battery embedded in a system, while still having the system powered and functioning is common in consumer electronics. While this was not the exact problem that needed to be solved in the Sharkduino system, it was analogous and thus we began by looking at consumer electronics for a solution.

In consumer electronics a power path management circuit is used to solve this problem. These circuits consist of a power path lithium ion charger IC and surrounding support circuitry. Many of these ICs exist on the market at varying price points with widely varying characteristics [12]. What all these circuits had in common is that they would lead to a significant increase in complexity of the Sharkduino circuity, and would require a significant investment of resources to design, validate, and integrate into the existing system. Because of this we attempted to identify a simpler solution before beginning the design work associated with creating a power path circuit.

The problem preventing proper LiPo charging was that the Sharkduino System

pulled too much current from the LiPo while the LiPo was attempting to charge. Since we wanted the system to be fully embedded, and eventually potted in epoxy, we could not detach the battery from the system or have any sort of mechnical switch to disconnect them. The power path circuit would disconnect them electrically and automatically, but we did not want to invest in its development. We eventually realized that if we could reduce the current drawn by the Sharkduino system enough, the MCP7381 would fail to notice it and complete a charge cycle normally.

Code to put the system into an ultra low power state already existed as it was used in the normal operation of the tag for a start up delay of the tag. We took this code and created a program that only put the system into this state and did nothing else. We then wired a MCP7381 on an external board to a Sharkduino, connected a battery to it, and observed the results. We found that the MCP7381 was able to compete a charge cycle with a Sharkduino in its lowest power state attached. The time taken to complete this cycle however, was increased up to 25%. The MCP7381 was not able to complete a charge cycle in any amount of time with a normally operating Sharkduino.

We deemed this increase in battery charge time to be an acceptable tradeoff for not having to design and add a power path management circuit to the Sharkduino. The solution for charging Sharkduinos going forward will be to put them into a battery charging state, which is simply an ultra low power state, and return them to a normal state for deployment. The downside to this is that the Sharkduino is only ever fully off when the battery is depleted.

Chapter 6

Sharkduino Batch Assembly

6.1 Batch Assembly Procedure

In preparation for the Summer 2017 season we decided to try to build ten Sharkduino V2.2s. This large number of tags would enable us to collect as much data from live animals as possible. Building ten new Sharkduino systems represented a significant time investment. Prior to this point, we assembled Sharkduinos one at a time, completely independent of one another. However since we needed a larger number of Sharkduinos quickly we began to look into what sort of efficiency improvements could be achieved by developing a batch assembly process. The goal was to cut down on the assembly time per Sharkduino by assembling multiple Sharkduinos together.

Broadly speaking the steps to assemble a V2.X style Sharkduino are as follows:

1. Place the microSD card slot in solder paste on the bottom side of the board, and then reflow

2. Put solder paste for all the components on the top side of the board using a combination of the solder stencil and syringe

3. Place components on the top side of the board, starting with the small ICs and then working out to the passive components and larger ports

4. Reflow the board
5. Visually inspect the board for bad solder joints and check for short circuits with the multimeter
6. Hand solder the JST connector
7. Hand solder 0.1" header pins on the Arduino Pro Mini
8. Hand solder the Arduino Pro Mini onto the board
9. Run test code to confirm all the sensors and ports are working, and set the RTC
10. Run the Sharkduino configuration code

This whole process took roughly 2-3 hours to complete for a single Sharkduino.

The first step we took to improve the efficiency of this process was to put the microSD card slot onto multiple boards at once, and on a different day as the rest of the steps. The microSD solder paste and reflow is very straightforward so it was not difficult to do on four or five boards at once. Doing this on a separate day as the rest of assembly gave two distinct benefits. First it decreased the continuous working time for the full assembly. Second it allowed the solder reflow oven time to completely cool before using it to reflow the top side of the board. When we were doing all the steps on the same day the top side of the board often came out of the oven looking burnt. We believed this was because the solder reflow oven did not have time to fully cool after reflowing the microSD slot on the bottom side. This increased starting temperature and distorted the heating curve programed into the oven, leading to the top side solder joints being slightly burnt.

The next step we took was to try to assemble two boards at once. To do this we first took a board with a microSD slot previously attached and used the stencil

and syringe to apply solder. A Sharkduino V2.2 in with a stencil can be seen in Figure 6.1. We then took a second board and applied solder in a similar way. Next we placed all the needed components on the first board, and then the second. Finally we placed both boards into the reflow oven. When they came out of the reflow oven we conducted the rest of the steps in tandem for both boards. This decreased the time per board considerably, from 2-3hr per board to slightly under 2hr per board.

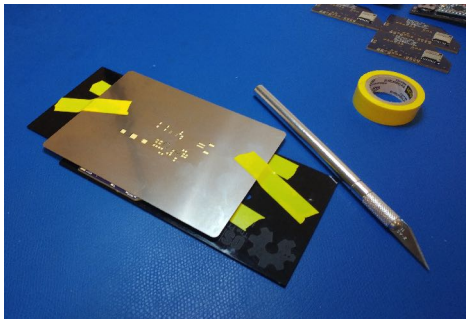

Figure 6.1: Solder Stencil in Use

Next we tried to streamline the assembly even further to do three boards at the same time. For this batch we did the solder stencil for all three boards, and then the solder syringe for all three boards. This is opposed to doing all the solder paste on one board and then starting over on the next board as we did in the previous batch. We then placed components on all three boards at the same time. This means that, for example, we placed the gyroscopes on all three boards, then the accelerometers on all three boards, etc. This was much more efficient than doing the boards independently (placing the gyroscope and then the accelerometer on the first board and then starting over on the second board). We finally reflowed all three boards and conducted the hand soldering and testing in tandem. This further improved the time per board, getting it down to 1-1.5 hours per board.

Overall these relatively simple and obvious batch techniques greatly reduced the

time per board. Much of the time per board was being lost switching between steps. These improved techniques however are not scalable to larger production runs. If, for example, we wanted to build 100 Sharkduinos it would become more efficient to use a pick and place machine. This is a robotic machine that picks up the components and places them onto the proper spot on board. These machines are very expensive, but for large enough production runs it becomes cost effective to have the tags manufactured by an outside company. At the level at which Sharkduino is currently operating, though, hand assembling the boards is still the most cost and time effective.

6.2 Batch Assembly Yield

The yield of successful builds needs to be monitored with this system going forward. Through ten attempted builds only four Sharkduino V2.2s came out fully functional. A fifth one appeared to be fully functional during initial assembly, but its continued use has shown that its battery life is consistently below what is expected (see Chapter 7 for more information). Two Sharkduinos were deemed functional for development purpose, but not deployment. This was due to a problem with the RTC battery backup on one, and a non functional gyroscope on the other. Three more were deemed failed builds during testing.

Of the three boards deemed failures, one had a short between power and ground, one had the gyroscope move out of place during reflow (Figure 6.2), and the last had unexpectedly low resistance between power and ground.

After the summer we were able to identify the cause of the the low resistance between power and ground on the third failed board. It was due to a short between the gyroscope interrupt, which is a pulled-up line, and ground. By cutting the interrupt line, thus removing the gyroscope input alarm functionality, we were able to remove the short and make the Sharkduino equivalent to a Sharkduino V2.1. An image of

the location of the short and of the repair can be seen in Figure 6.3.

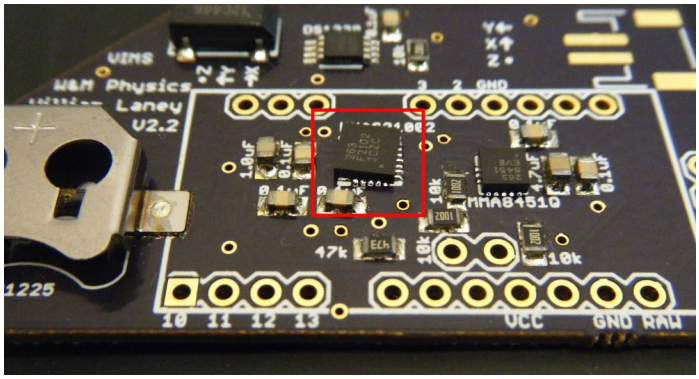

Figure 6.2: Gyroscope Rotated Out of Place

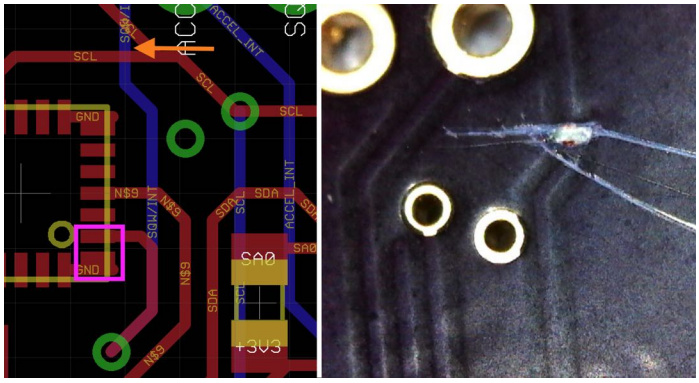

Figure 6.3: Short Repair in Failed Sharkduino V2.2
Left: The location of the short is highlighted in pink, the location of the cut in the line is pointed to by the orange arrow
Right: The bottom side of the board with the trace cut

Even with the latter repair of a failed board we achieved a 40% success rate in getting fully functional Sharkduinos and a 50% success rate in getting deployable Sharkduinos. While we do not have enough data to determine if the yield rate is significantly lower with the batch assembly than with the standard hand assembly, we do believe that the yield rate is lower than desired. Going forward work needs to be done to determine procedures that are less error prone. Additionally research needs

to be done in reworking and repairing boards so that non-deployable Sharkduinos can be made deployable. These problems would also be mitigated by production runs using a pick and place machine.

Chapter 7

Battery Life Characterisation

One of the most important characteristics of the tag is its battery life. However during the our deployments in the summer of 2017 there was uncertainty about the expected battery life of the tags, and the battery life in the field did not seem to match up with the expected values. Due to this we began work to characterize the battery life of different versions of the tag.

The most effective and accurate way to do this was also the simplest and most time consuming: plug the tag into a fully charged battery and see how long it lasted. While doing these battery life tests the tag was left to sit in a controlled environment, so that we could be sure that the results were accurate and comparable. The most important variables to control for were temperature, movement of the tag, and tag sample rate. The temperature has to be controlled because LiPo batteries demonstrate a temperature dependence [13]. The physical movement of the tag needed to be controlled to ensure that no mechanical failure, could cause the tag to shut off prematurely.

These tests have been conducted sporadically since September 2016. A list of all controlled battery tests conducted since that time is given in Table 7.1. Average constant current draw is the variable of interest in the data. It was calculated by dividing the battery capacity by the time the tag was powered. We expected that

the average constant current draw would be roughly similar between all 2.x tags, and that the 2.x tags would exhibit a significantly smaller average constant current draw than the 1.0 tag. This is because the 2.x tag used a significantly lower power sensor package then the 1.0 tags.

Table 7.1: All Sharkduino Battery Tests
All tests done at room temperature ($\sim 20^oC$) at 25Hz sample rate

Sharkduino	Battery Size (mAh)	Start Time	End Time	Duration (Hr)	Average Constant Current Draw (mA)
V2.2b	1100	11/6/17 18:44	11/15/17 7:16	204.54	5.4
V2.0a	500	11/10/17 13:58	11/14/17 11:59	94.02	5.3
V2.2c	500	11/10/17 13:43	11/14/17 11:18	93.58	5.3
V2.2b	500	10/13/17 12:57	10/17/17 20:53	103.93	4.8
V2.2f	500	10/13/17 11:57	10/13/17 23:14	11.29	44
V2.1a	500	10/13/17 13:04	10/17/17 17:38	100.57	5.0
V2.0b	500	10/8/17 19:34	10/12/17 17:06	93.54	5.3
V2.2b	500	10/9/17 12:19	10/10/17 19:38	31.33	16
V2.2f	500	10/6/17 9:27	10/6/17 18:22	8.91	56
V2.2f	500	10/5/17 17:53	10/6/17 2:42	8.80	57
V2.0b	500	1/27/17 15:39	2/2/17 1:23	129.72	3.9
V1.0a	400	9/6/16 15:26	9/7/16 0:36	9.17	44

From this table, it appears that there was a large variance in battery life, independent of time. The cause of this maybe that many of the measurements were taken on the Sharkduino V2.2f. This tag appears to show atypical battery life performance. This is clearly seen when its battery life is compared to that of the V2.2b. These tags are the same version and run the same code, so they should show identical battery life characteristics. Unexpectedly, they do not. The characteristics of V2.2b are more in line with what is expected from the similar V2.0 and V2.1 tags. This led us to conclude that there was a hardware problem with tag V2.2f. Removing that tag from the data we get the results in Table 7.2.

The difference between the average constant current draw seen in V2.2b between

Table 7.2: Sharkduino Battery Tests with V2.2f removed
All tests done at room temperature ($\sim 20^oC$) at 25Hz sample rate

Sharkduino	Battery Size (mAh)	Start Time	End Time	Duration (Hr)	Average Constant Current Draw (mA)
V2.2b	1100	11/6/17 18:44	11/15/17 7:16	204.54	5.4
V2.0a	500	11/10/17 13:58	11/14/17 11:59	94.02	5.3
V2.2c	500	11/10/17 13:43	11/14/17 11:18	93.58	5.3
V2.2b	500	10/13/17 12:57	10/17/17 20:53	103.93	4.8
V2.1a	500	10/13/17 13:04	10/17/17 17:38	100.57	5.0
V2.0b	500	10/8/17 19:34	10/12/17 17:06	93.54	5.3
V2.2b	500	10/9/17 12:19	10/10/17 19:38	31.33	16
V2.0b	500	1/27/17 15:39	2/2/17 1:23	129.72	3.9
V1.0a	400	9/6/16 15:26	9/7/16 0:36	9.17	44

10/9/17 and 10/13/17 can be accounted for by a bug discovered and then fixed in the tag code. While conducting these battery tests, both V2.2b and V2.2f were showing lower than expected battery life. The main change between V2.2b and V2.1/0 was the addition of an interrupt line from the FXAS21002 gyroscope. This line was not used in the code, but was added to the hardware to allow for future development. During the investigation we discovered that the default behavior of the gyroscope was to expect the interrupt pin to be connected to a pulled down line [9], however it was connected to a pulled up line. This created a virtual short that significantly impacted the average constant current draw of the tag. Upon changing the code for the gyroscope to expect to be connected to a pulled up line the constant current draw of both the V2.2b and V2.2f decreased significantly (Table 7.1). This brought the V2.2b into expected levels, while V2.2f was improved but still operating with a higher than expected constant current draw.

Finally the change in average constant current draw between the tests on tag V2.0b conducted between 1/27/17 and 10/8/17 can also be attributed to changes in code. At the time of the 1/27/17 test the data provided by the FXAS21002 gyroscope

was incorrect due to a bug in the implemented library. This bug was repaired before the test conducted on 10/8/17. The reason that this bug had such an impact on constant current draw is still unknown. We are working with the software team to compare the code base between these tests and identify the cause of this change.

Chapter 8

Conclusion

8.1 Further Work

There are still many possible ways to improve the system. Some smaller scale improvements have already been discussed, but large changes to the system have also been considered.

A possible improvement to the system would be the inclusion of a magnetometer. This sensor detects magnetic fields, and would allow us to know the device's bearing relative to the earth's magnetic field (i.e. North, South, East, or West). This, in conjunction with the accelerometer and gyroscope, would allow us to get precise orientation information for the animal and allow us to understand its movements with greater precision [14]. Additionally we would gain insight into the migratory patterns of the animal. The addition of a magnetometer does however come with some disadvantages. It consumes power, reducing the battery life of the tag. It also adds complexity to both the hardware design and code. This increases the probability of errors or malfunctions on the tag. Finally it would slightly increase the price of the tag. At this time we do believe that the benefits of the magnetometer outweigh the costs. We are currently working on selecting a magnetometer and creating initial designs for it. At the time of this writing the most likely magnetometer for us to use

is the MAG3110 [15].

We can also improve the system without changing the sensor package. The board can be made smaller and more streamlined. There is also a possibility to make the boards wireless by using Bluetooth Low Energy (BLE) for data communication, and inductive charging for power. Making the board fully wireless would greatly reduce the cost and complexity of potting the system. It would also be a novel feature that does not exist commercially.

Another aspect of the tag that needs to be developed is the recovery system. The tag must be able to detach itself from an animal, surface, and emit a signal so that it can be located and retrieved. [16], [17]. Commercial options for this do exist, but research needs to be done into which one is the best for our sensor. Additionally research is needed into whether it would be more cost effective or more desirable to design our own recovery mechanism. Since our deployments focus on the Chesapeake Bay many communication systems, such as cellular, are available to us. Because a large amount of marince science research occurs in similar costal zones, this would be useful not just for our tag, but for all the researchers working in areas where cellular and other communication systems are available.

Finally research can be done into other non-marine science areas in which this sort of small, cheap sensor system could be useful. An example of this is in shipping packages: a Sharkduino like system could easily be included in a delicate package to allow the recipient or shipper to tell if the package has been exposed to extreme movement. Another example is in aviation. The system we have made is similar to the inertial measurement units (IMUs) used in drones. It however is very low power usage so it could be used on smaller drones that are tightly power constrained. We believe that this sort of cheap and durable sensor system can be used for a wide variety of tasks in many fields.

Bibliography

[1] *Xtrinsic mma8452q 3-axis, 12-bit/8-bit digital accelerometer*, 8.1, Freescale Semiconductor (Oct. 2013).

[2] *L3gd20h: mems motion sensor: three-axis digital output gyroscope*, 2nd ed., STMicroelectronics (Mar. 2013).

[3] *Ds3234: extremely accurate spi bus rtc with integrated crystal and sram*, 3rd ed., Maxim Integrated (July 2010).

[4] *Mcp73831/2: miniature single-cell, fully integrated li-ion, li-polymer charge management controllers*, Microchip Technology Inc. (Mar. 2014).

[5] *Ms5803-14ba miniature 14 bar module*, 5th ed., TE Connectivity Corporation (Sept. 2012).

[6] *Tmp102: low power digital temperature sensor with smbus^{TM}/two-wire serial interface in sot563*, Texas Instruments (Oct. 2008).

[7] M. J. Ross and J. H. Mccormick, "Effects of external radio transmitters on fish", The Progressive Fish-Culturist **43**, 67–72 (1981).

[8] I. A. Bouyoucos, C. D. Suski, J. W. Mandelman, and E. J. Brooks, "Effect of weight and frontal area of external telemetry packages on the kinematics, activity levels and swimming performance of small-bodied sharks", Journal of Fish Biology **90**, 2097–2110 (2017).

[9] *Fxas21002c: 3-axis digital angular rate gyroscope*, 2.1, Freescale Semiconductor, Inc. (May 2015).

[10] *Ds1339b: low-current, i2c, serial real-time clock for high-esr crystals*, 1st ed., Maxim Integrated (Apr. 2015).

[11] *Xtrinsic mma8451q 3-axis, 14-bit/8-bit digital accelerometer*, 8.1, Freescale Semiconductor (Oct. 2013).

[12] C. Mauney, *Power-path li-ion charger selection*, Application Report SLUU332 (Texas Instruments, Aug. 2008).

[13] X. Gong and C. C. Mi, "Temperature-dependent performance of lithium ion batteries in electric vehicles", in 2015 ieee applied power electronics conference and exposition (apec) (Mar. 2015), pp. 1065–1072.

[14]H. J. Williams, M. D. Holton, E. L. C. Shepard, N. Largey, B. Norman, P. G. Ryan, O. Duriez, M. Scantlebury, F. Quintana, E. A. Magowan, N. J. Marks, A. N. Alagaili, N. C. Bennett, and R. P. Wilson, "Identification of animal movement patterns using tri-axial magnetometry", Movement Ecology **5**, 6 (2017).

[15]*Xtrinsic mag3110 three-axis, digital magnetometer*, 9.2, Freescale Semiconductor (Feb. 2013).

[16]K. O. Lear and N. M. Whitney, "Bringing data to the surface: recovering data loggers for large sample sizes from marine vertebrates", Animal Biotelemetry **4** (2016).

[17]B. M. Whitmore, C. F. White, A. C. Gleiss, and M. M. Whitney, "A float-release package for recovering data-loggers from wild sharks", Journal of Experimental Marine Biology and Ecology **475**, 49–53 (2016).