UNIVERSITY OF CALIFORNIA SAN DIEGO

Livewire Feedback Control and Data Acquisition

A thesis submitted in partial satisfaction of the
requirements for the degree Master of Science

in

Engineering Sciences – Applied Ocean Science

by

Ewan Yulun Bohannon

Committee in Charge:

Professor Andrew Lucas, Chair

Professor Matthew Alford

Professor Renkun Chen

2023

The thesis of Ewan Yulun Bohannon is approved, and it is acceptable in quality and form for publication on microfilm and electronically

University of California San Diego

2023

TABLE OF CONTENTS

LIST OF FIGURES & TABLES

ACKNOWLEDGEMENTS

Firstly, thank you to Dr. Drew Lucas for giving me the opportunity to work directly on the exact type of project I was looking for in my graduate school experience, and thank you also for the opportunity to take on the software and electronics aspect of this project with more enthusiasm than background knowledge. Thank you Mai Bui, San Nguyen, and Arnaud Le Boyer for teaching me software fundamentals and for helping me debug my code. Thank you Riley Baird and Sean Lastuka for teaching me the electrical side and showing me how to not blow up components. Thank you Mike Goldin for guiding me and teaching me in both depth and breadth on engineering- mechanical, electrical, and software. Thanks again, MODLAB, I am definitely a better engineer and researcher now than I was when I got here.

Most importantly, I thank my parents for supporting my aspirations by giving me the freedom to come to UCSD and hone my vision.

ABSTRACT OF THE THESIS

Livewire Feedback Control and Data Acquisition

by

Ewan Yulun Bohannon

Master of Science in Engineering Sciences – Applied Ocean Science

University of California San Diego, 2023

Professor Andrew Lucas, Chair

The Wirewalker is a vertical profiling system that is driven by ocean waves along a cable using its mechanical cam. All onboard power is dedicated to instrumentation, but battery capacity remains one of the main limitations of Wirewalker deployments. A custom electric motor-generated called the Livewire was built by the same lab that created Wirewalkers. Its purpose is to replace the Wirewalker rectifying cam. Using ocean wave regenerative braking, the Livewire will provide both motion and charging. The mechanism for doing so is like the cam- the Livewire either brakes or allows the cable to freely rotate its idler wheel. This paper discusses my successful development of a foundational data acquisition and control system for the Livewire that uses depth and direction data from a CTD and Hall

effect sensors to regulate motion. Next steps are implementing a charge circuit and four quadrant motor control.

CHAPTER 1: INTRODUCTION

Wave energy is one of the major sources of renewable energy alongside wind and solar, yet its development hasn't become mainstream. Yet, surface waves, mostly generated by wind, are ever-present. Because waves are relatively consistent, wave energy conversion has been a long-sought after addition to wind and solar for offshore energy harvesting.

There are limitations to scaling up wave energy conversion to power an electrical grid, including transportation of power from offshore to the grid and producing sufficient power to make infrastructure investment feasible. However, at smaller scales, the challenges associated with wave energy are more tractable. Current ocean instrument research is producing smaller, smarter, and more efficient devices. Some even have solar arrays that enable year-long deployments (Corredor, 2018). The sun is a great resource if the instrument can charge at the surface, but the ocean waves can be used to continuously charge instruments at any depth by coupling between wave motion and a floating surface buoy. Ocean research instruments are prime candidates for wave energy integration because their power consumption is relatively low. Overcoming existing battery capacity limitations would make longer instrument deployments feasible. This paper features a wave energy conversion device designed to power an instrument-bearing platform by generating about 10W. The focus of this paper is my work on the development of a data acquisition and control system for that application.

1.1 The Wirewalker

The Wirewalker is a wave-powered, instrument-bearing platform designed to provide two-dimensional time-depth data series by taking vertical profiles along a surface buoy-suspended, tensioned cable. Its current design is driven by a mechanical cam in the center of its rectangular frame that allows wave motion to propel it downwards, releasing the cable when the wave motion is upwards (Smith et. al, 2012). A physical bottom stop triggers the free ascent of

the Wirewalker to the surface, the speed of which is determined by its buoyancy. This allows users to ballast according to their specific plans and payloads to achieve a desired rise rate. A physical top stop sets the mechanism to cam downwards again. The purely mechanical design accomplishes automation of rapid, consistent vertical profiling using relatively inexpensive materials. The battery it does carry is wholly dedicated to powering onboard sensors. Without onboard charging, however, the Wirewalker has a limited operational time on the scale of weeks because it needs to be brought to the surface to recharge or swap out the batteries. Useful wave energy is lost to friction within the camming mechanism and wears out its parts. Therein lies an opportunity to develop wave energy conversion technology and improve the Wirewalker's capabilities.

1.2 The Livewire

The Livewire electric motor-generator was designed and built by the Multiscale Ocean Dynamics Lab at SIO to replace the mechanical cam and address the battery capacity issue by using wave power as both a motive force along the cable and as a power source to recharge the battery during operation. It is a waterproof, three-phase motor that uses wave-induced cable motion as a mechanical input to provide three-phase electrical output when an electrical load is switched on. There are two main Livewire motor-generator components: the rotor and the stator. The rotor is a rotating cylindrical shell lined with magnets parallel to its central axis. The stator is a fixed cylinder of rectangular, copper coils arranged parallel to its central axis. Brushless DC motors, or BLDCs, have the stator placed concentrically inside the rotor with an air gap between the two. Rotation is controlled by the magnitude and direction of current supplied to the stator. This current induces a magnetic field in the stator by Faraday's Law. The attraction or repelling of the two fields causes the rotor to move. This is also the source of the motor's torque. Torque in an electric motor is maximized when the two magnetic fields are 90^{o} out of phase because the

force of attraction between two poles then is entirely tangential to the motor instead of having both tangential and radial components.

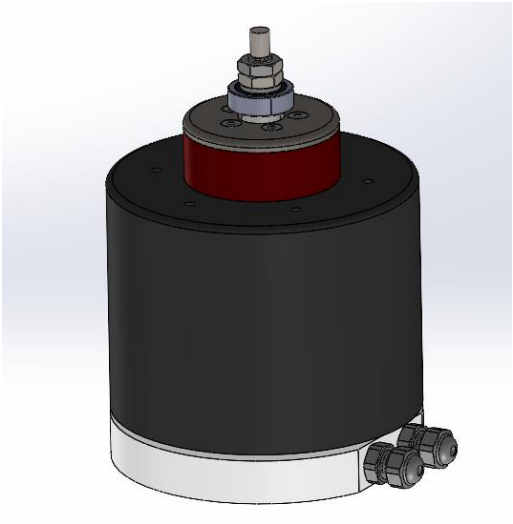

Figure 1. CAD of the Livewire motor-generator showing the red polyurethane idler wheel for gripping the cable. Two connectors at the bottom output the motor phases and Hall effects. CAD courtesy of Jonathan Ladner.

1.3 Regenerative Braking

One way to maintain rotor motion is to power the stator with a 3-phase AC current with each phase shifted 120^{o} from each other. This induces a rotating magnetic field, or RMF, that keeps the rotor in motion because the stator coils do not share the same polarity at a given time. Reversing this process turns a BLDC from a motor into a generator and describes the basics of regenerative braking. When a car brakes, kinetic energy is typically converted to heat in the brake pads to slow down the car. Regenerative braking uses the kinetic energy of the car to rotate its electric motor against an electrical load that includes a battery. When the motor is turning, the moving magnetic fields generate a backwards electromotive force, or back EMF, that opposes the input current to the coil. In this scenario the car's forward momentum keeps the motor spinning with mechanical energy. The spinning rotor magnets generate an EMF that induces current inversely proportional to the resistance of the load plus the motor's inherent resistance. This follows from Ohm's Law: the current through a closed electrical loop is equal to the

electromotive force, or voltage, divided by the total resistance of that loop. The induced current produces a torquing magnetic field by Lenz's Law. Shorting out the motor gives the maximum current and braking because the resistance is nearly zero. Conversely, a highly resistive load or open loop circuit results in no current generated, so no torque opposes rotation besides torque from other sources like friction. The difference between high and low resistance loads relative to the motor's impedance is described by the maximum power transfer theorem, or Jacobi's law, as a tradeoff between efficiency and amplitude. To transfer maximum power from an external source with internal resistance, the load resistance should equal the source resistance (Shaalan, 2005). A higher load is more efficient because it dissipates a greater percentage of the power than the source, but it reduces the induced current and thus overall power. A lower load leads to a larger current and more overall power, but more of the power is dissipated in the source. Thomas Kelly, a Multiscale Ocean Dynamics lab alum, found that the ideal load is 4.3 Ohms by the maximum power transfer theorem after performing an experiment to find the Livewire's impedance.

Generating power from a three-phase motor depends on the configuration of the AC current conductors, which must come in multiples of three to accommodate the three motor phases. The Livewire features six terminal leads, one positive and one negative for each motor phase. These leads can be connected in several ways, but the two most common configurations for three-phase motors are the wye and delta configurations in Figure 2. In wye one end of each coil lead is connected to a common point, and in delta the phases are connected in series. The other ends are available as leads in both cases. Phase current and phase voltage come from the stator windings. The line voltage and line current are carried out from those windings. Wye-configured motors have a line current equal to the phase current and higher line voltage while delta-configured motors have equal line and phase voltages but greater line current. Though they deliver the same power, the wye configuration has less transmission losses due to the lower line

current (Goodstall, 2012). Because the phases are spaced 120 degrees, the increase to voltage or current depending on the configuration is by a factor of $\sqrt{3}$. The delta configuration line current is that much greater than the wye line current. This equates to triple the power loss by transmission:

$$P_{loss} = {I_{line}}^2 R \qquad (1)$$

While the lengths of wire connecting the phases to the control circuitry are relatively short compared to typical industrial applications like transformers, the scale of power generation for the Livewire makes this power savings valuable, so the wye configuration was chosen.

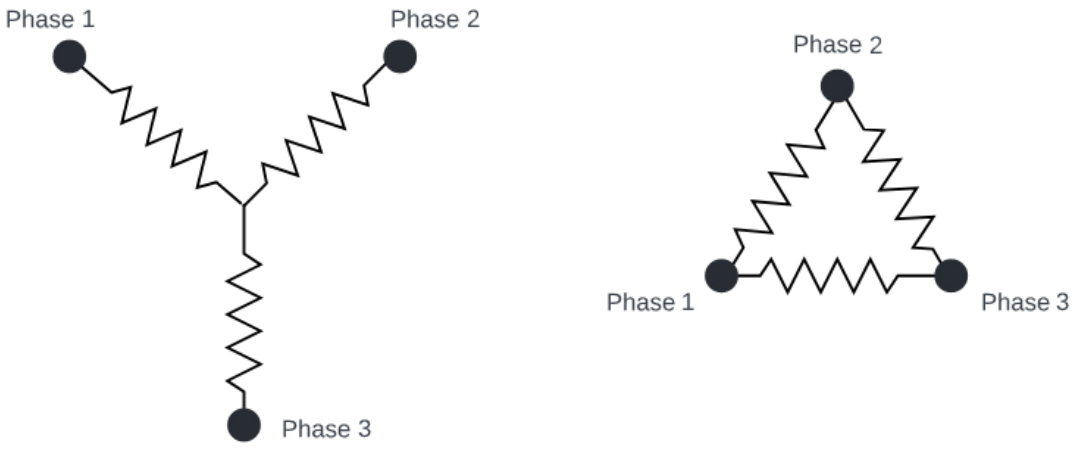

Figure 2. Wye and delta configurations. The common wye point often provides a neutral lead or ground.

One of the Livewire's unique features is its Halbach array. By clocking the 29 magnets in the rotor at specific angles relative to each other, the magnetic field inside the rotor is strengthened while the field outside the rotor is reduced to near zero. This increases the torque density of the motor, which means more current is generated per rotation. The magnetic field within the motor is represented by five pole-pairs. These are visible in Figure 3 as concentrations of magnetic field strength.

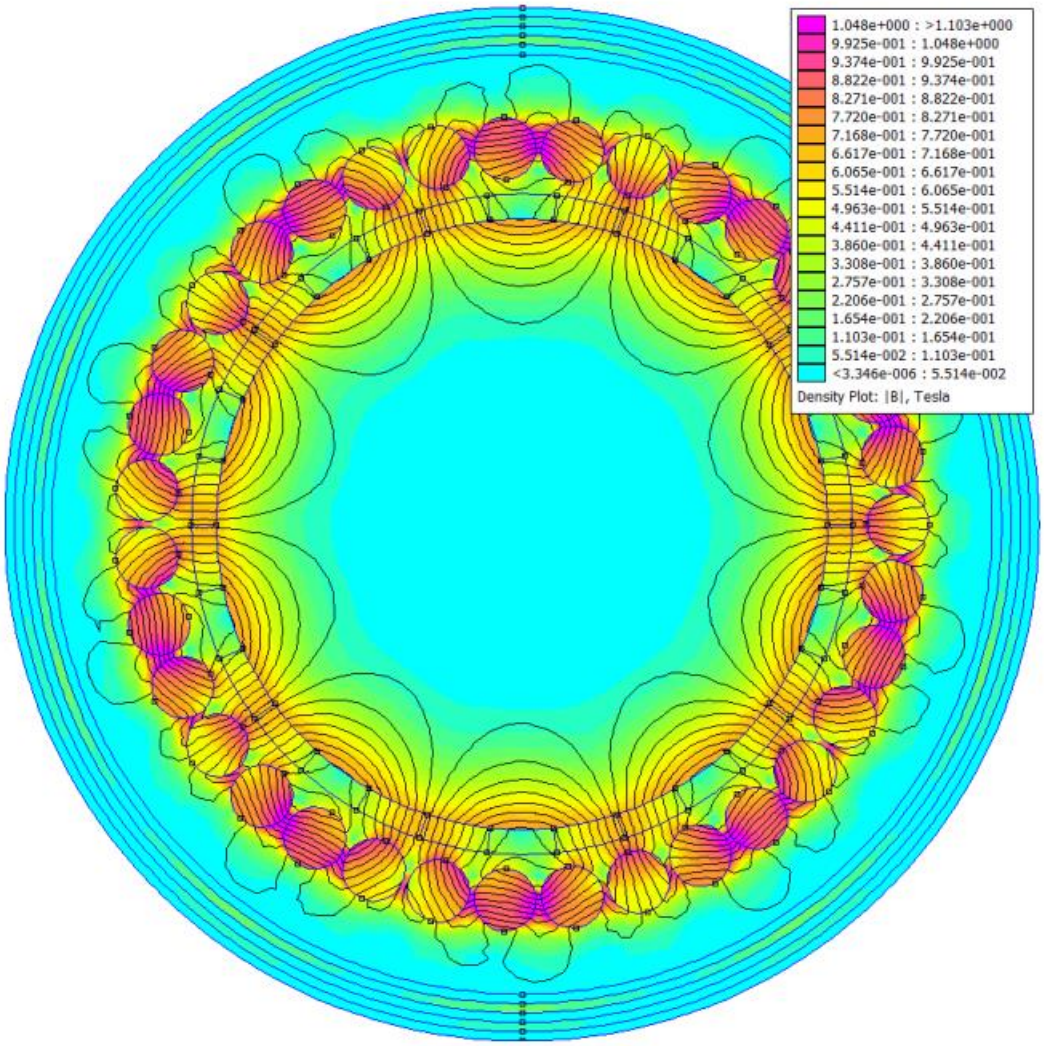

Figure 3. Halbach array for the Livewire showing five pole-pairs. Image courtesy of Aaron Goldin.

1.4 Livewire-Wirewalker Integration

Wave energy conversion with a three-phase motor is possible through a process similar to regenerative braking because wave motion supplies a reliable kinetic input to the system. Consider the Livewire motor-generator, an integrated, switchable electrical load, and the Wirewalker's surface-mounted buoy. If the Livewire is constrained to rotate along the length of the cable, then the waves oscillating the surface buoy will rotate the rotor. The permanent magnets in the rotor then induce a current in the stator coils. Because the Livewire is a three-phase motor, the output current will be 3-phase AC current. If the load includes a battery, then

converting the output AC current to DC current via a rectifier concludes the wave energy conversion from kinetic to potential energy.

The Livewire is defined such that switching on the charging circuit closes its loop and is otherwise connected in an open loop to ground. By default, the load is switched off, so the Livewire rotates freely. Regenerative braking makes the rotor harder to turn and forms the basic method for traversing the cable. By braking on the upward wave motion and switching off the electrical load during downward wave motion, the Livewire can emulate the Wirewalker's mechanical cam. Unlike the mechanical cam, the Livewire is not limited to travel in just one direction so long as an appropriate cable gripping mechanism is installed that constrains it to the cable without slippage. The Livewire should grip the cable when braking and freely rotate along the cable otherwise. The Livewire rotor has a polyurethane wheel mounted to it, and a prototype cable gripper with two wheels designed by Thomas Kelly currently grabs the cable in an omega formation. This prototype was able to generate 0.1 Watts in the pool, but this measurement, which is far below the test production on the lab bench, was probably confounded by friction in the power take-off system fabricated by Kelly.

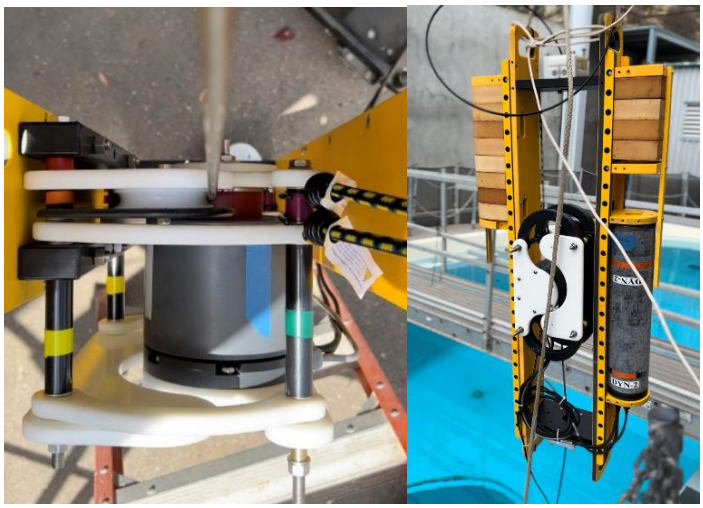

Figure 4. Left: Top-down view of the cable being gripped. The red is the wheel attached to the Livewire's rotor. Right: Full view of a Livewire-Wirewalker. The Wirewalker frame was widened to accommodate the Livewire's bulk. The electronics for the pool test are in the pressure case mounted on the lower right.

Besides a cable-gripping mechanism, the other required element to integrate the Livewire into the Wirewalker is a system for toggling the charging circuit because this determines how it moves and recharges the battery. This requires the development of a control system to adhere the system to a mission plan since the combined system will no longer travel solely based on physical stops and cable lengths. The benefit of an onboard control system is that it can use data from the onboard sensors to make smart decisions about charging and profiling with minimal delay; therefore, an ideal Livewire Wirewalker system could be deployed for longer and would operate more efficiently. A typical Wirewalker payload includes a CTD while the Livewire has Hall effect sensors built into it. The CTD provides pressure data, and the Hall effect sensors provide a means to calculate the direction and rpm of the Livewire. The bare minimum Livewire control system should decide when to use the Livewire's braking mode to ascend or descend the cable based on these data. My thesis project was to develop such a system. In this paper I discuss the system elements and how I developed a complete control system for the Livewire that can accomplish profiling and hovering based on default states or user input using CTD and Hall effect data.

CHAPTER 2: SYSTEM LAYOUT

2.1 Control Environment

Controlling the Livewire requires the selected control device to interface with multiple sensors and travel with the Livewire-Wirewalker system along the cable. Iridium satellite communication capability has been developed for the Wirewalker but is outside the scope of this project. Therefore, the deployed Livewire should function first on its own. The Multiscale Ocean Dynamics Lab builds and deploys instruments regularly with electronics safely packaged in a cylindrical pressure case. Making use of an existing pressure case was an easy choice because its volume should easily fit an electronic control board and attached circuitry, including batteries. The end caps of the pressure case feature subsea connectors that allow electronics within the case to communicate with external sensors.

2.2 The EFM32GG11 STK Microcontroller

Like the pressure case, the Silicon Labs' EFM32GG11 STK microcontroller, or STK, is already used in the Multiscale Ocean Dynamics Lab and has the necessary pins and peripherals to develop the Livewire controls. It has a 32-bit ARM Cortex-M4 processor that can handle more software operations than would be developed for the controls. The STK features more than 100 general-purpose I/O, or GPIO, pins, but only seven are necessary to get sensor inputs/outputs for the Hall effects, Seabird49 CTD, and simple control of the motor. To gather data rather than high/low signals, we used the alternate functions of some of those pins to enable USART, which is a transmitter-receiver hardware for serial communication. This simply allows the board to read in actual data rather than high/low voltage states on the GPIO pins. The STK features multiple USARTs, UARTs, and other communication modules. Two USARTs are used in this project- one for reading Seabird49 data and one for streaming out a compound data string for debugging and user input. The STK's complementary software IDE is Simplicity Studio. It provides a working

environment in C with libraries for STK board functions. It also features a SEGGER J-Link debugger that allows users to set breakpoints in the code to stop and run a program on the STK line-by-line to help find issues. The 4.1.4 version of the gecko SDK is currently used.

2.3 STK Sensor Inputs

The Seabird49 CTD, or SBE49, provides a 24-character, hex-encoded data string describing the local conductivity, pressure, and temperature. It also includes a pressure temperature correction and two delimiter characters. The SBE49 is typically mounted to the Wirewalker frame in deployment and is an external input to the STK's pressure case. Due to its bulk and the need for a bench-testing method for the Livewire controls, a simulated SBE49 was used to provide varied pressure inputs. Arnaud Le Boyer and San Nguyen, both engineers of the Multiscale Ocean Dynamics lab, developed SBE49 simulator software. For this project, Arnaud's Python program was modified to output random SBE49 data or a linear depth profile depending on the testing requirements. The simulator provides data through a computer's serial port to the STK via a Pmod connector board at a rate of 16 Hz, which is the SBE49's actual stream rate. While the connection between the actual SBE49 and STK could require a couple additional components, the STK pins selected would work for either input. The SBE49 data is streamed in through an STK USART using the sl_iostream functions developed by Silicon Labs. Then, it is decoded and streamed out using another USART for debugging purposes.

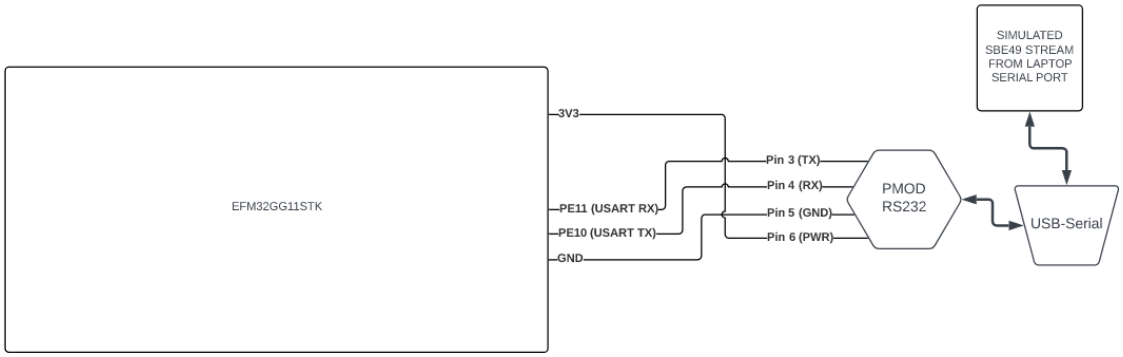

Figure 5. Schematic for the GG11 to receive simulated SBE49 data. Both the actual Seabird49 and the simulated program use the same USART communication, but the actual Seabird49 would need power from the onboard battery rather because the GG11 3V3 or 5V bus would be insufficient.

Hall effect sensors change states on the rising or falling edge of a magnet passing it. In magnetism, the south pole has a positive magnetic field strength, and the north pole has a negative magnetic field strength. A strong south pole turns on the sensor, which toggles the GPIO pin HIGH. A strong north pole turns off the sensor, toggling the GPIO pin LOW. Hall effect data is not gathered at a specific rate; rather, it is collected using a software interrupt method. When any of the three GPIO pins assigned to the three Hall effect sensors changes state, the software pauses its current task to address the interrupt flag that the change raises. After completing the interrupt routine, the software resumes its tasks. It is during the interrupt routine that the Hall effect data is used to calculate the direction and RPM of the Livewire. The RPM can be found by finding the difference in time between the current Hall interrupt and the previous one and dividing it by the defined pulses per revolution for the Livewire. Because the Halbach array arranges the magnetic field into five pole-pairs that will trigger each of the Hall effect sensors twice, there are 30 pulses per mechanical revolution. The Hall effect sensors are laid out such that they can never all be the same state at the same time. Overlaying the stator from the CAD in Figure 6 with the five pole-pair Halbach array in Figure 3 helps visualize this in Figure 7. A specific pulse pattern is created for rotation in each direction. The direction can be found by checking the state of either of the other Hall effect GPIO pins.

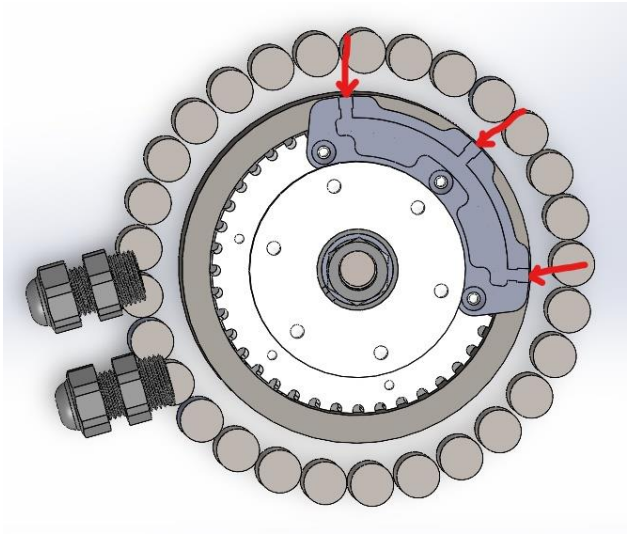

Figure 6. Cutout of the Livewire motor-generator CAD showing the Hall effect sensor positions within the stator and the magnets arranged around the outside in the rotor. CAD courtesy of Jonathan Ladner.

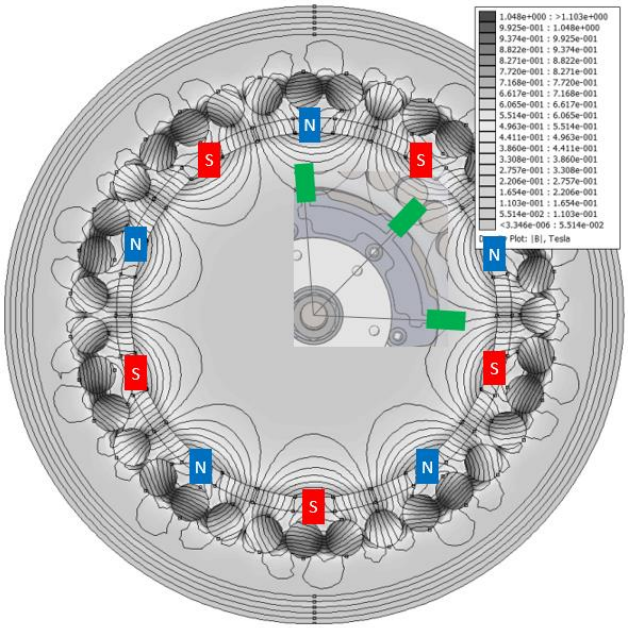

Figure 7. Visualization of magnetic poles showing the offset positioning of the Hall effect sensors, in green, that prevents them all from reaching the same state at the same time. The state displayed is 0,1,1, or 011. CAD courtesy of Jonathan Ladner, Halbach array image courtesy of Aaron Goldin.

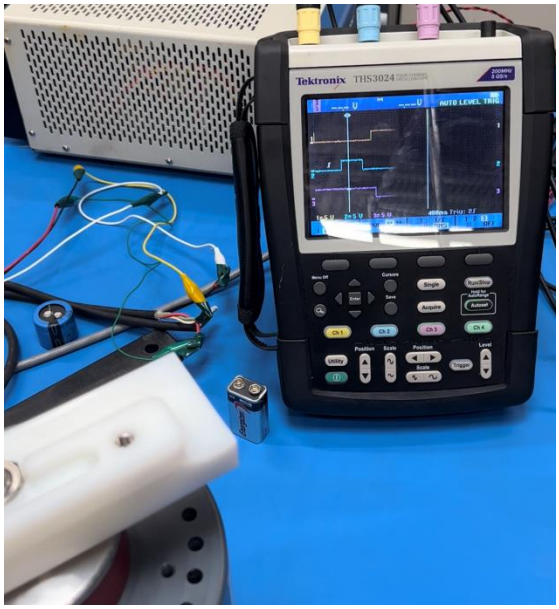

Figure 8. Livewire Hall effect pulses on an oscilloscope.

For example, the oscilloscope reading in Figure 8 shows four Hall effect triggers across the three sensors. The middle signal shows a rising edge and falling edge, so a magnetic pole entered the view of the sensor and then left it. If we assign the signals from top to bottom in the figure to sensors U, V, and W, it is clear that at the instant sensor V went HIGH, sensor W was HIGH, and sensor U was LOW. The sensors are arranged clockwise around the stator in the order U, V, W. The Livewire rotation must be counterclockwise because sensor V and sensor W are now both in the same state. This deduction depends on the Hall effect sensor order and the magnetic poles alternating north-south around the rotor. The magnetic poles are the only forces that change the state of the sensor, so in the absence of a magnetic field they remain in the same state caused by the most recent magnetic pole passage. In this case, sensor V triggering and going HIGH reveals that a south pole must have just switched it and should be physically next to it. Before this, a north pole flipped sensor V. Sensor U is LOW with a north pole while sensor V is HIGH with a south pole. The state of the Hall effects before the switch was 001 and is now 011. Each state can only go to two possible states based on rotation because the patterns in each direction are fixed. Based on the figures and the truth table below, it is only possible to reach the 011 state from the 001 state if the Livewire is rotating counterclockwise.

Table 1. Hall effect truth table values with direction indicated to go from one state to another

^ | | CCW	HALL_U	HALL_V	HALL_W	 CW | | V
	0	1	1	
	0	0	1	
	1	0	1	
	1	0	0	
	1	1	0	
	0	1	0	

Although the Livewire controls will use the SBE49 pressure to reference its position, it is also possible to calculate the Livewire's displacement from its starting position by tracking the net pulses in the clockwise or counterclockwise direction. Given the rotor wheel's outer circumference of 9.029 inches, dividing by the pulses per revolution results in a linear distance traveled per pulse of 0.30 inches. While this method doesn't account for slippage of the cable past the wheel, it provides one more way to debug the software and check the Livewire's position on the bench. In the ocean it could serve as a backup in case the CTD stops sending pressure data.

The Hall effect sensors in the Livewire require power, which is supplied by the one of the STK's 3.3V and ground pins. The STK's GPIO pins can only handle up to 3.3V themselves, so the Hall effect inputs are shifted from a maximum of 26V down to 3.3V using a hex buffer as a level shifter because the value of the sensor doesn't matter for simple control. The HIGH or LOW state of the sensor is what is useful.

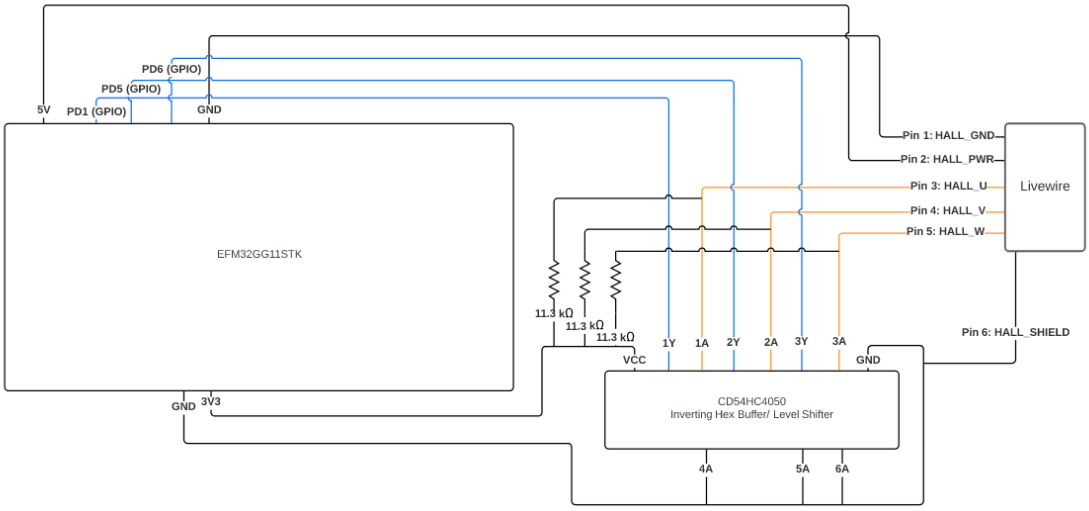

Figure 9. Schematic showing the components for communicating Hall effect pulses from the Livewire to the GG11. Unused inputs to the hex buffer are grounded to prevent floating voltages disturbing the system. Pull-up resistors on the inputs similarly keeps them from self-biasing when not toggled.

The next subsystem of the Livewire concerns the Livewire's motor phases and involves motor control and battery charging. Shorting out the three negative phases gives the wye configuration for the motor. The three positive phases pass into a rectifier to change the input AC current to DC current. Toggling the load on the motor is the main requirement of the control system. This requires a few components to execute. A solid-state relay grabs the rectifier outputs- one pin to a load to the 1st relay pin, the other just to the 2nd relay pin. The relay's 3rd and 4th pins use 5V and a MOSFET respectively to switch the load in and out of the circuit. The relay's first two pins are DC power, and the last two pins are DC control. The MOSFET connected to the relay is controlled by the switching of a GPIO pin. Toggling this single pin based on the input data forms the basis for the current control system.

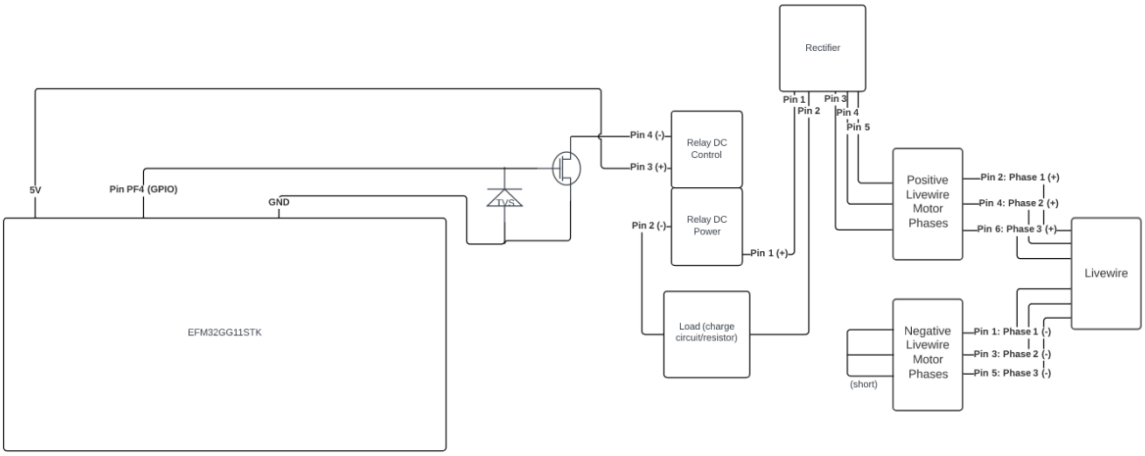

Figure 10. Motor control schematic using a GPIO pin and solid state relay to toggle a load in and out of the rectifier DC power. The negative motor phases are shorted out to create a wye motor configuration.

2.4 Tasks

Even though the STK only has one processor, it can handle multiple continuous software loops using tasks as defined in its micriumOS. The micriumOS kernel schedules tasks and runs them from highest to lowest priority, although it can do round-robin scheduling where each task is also only run for a certain amount of time before switching tasks. A task is an infinite loop of code with a user-defined stack size and priority. The stack is a list of memory registers in the order that the kernel must run functions for the task. Each task's stack also serves as saved context for the kernel that is restored when the task is run again. The priority of the task simply tells the kernel what order each task is run. Even though each task has a priority and stack size, they each require a wait function to inform the kernel when to switch off of the current task. Each task only waits a few ticks, but this is enough to allow other tasks to run. The STK's high frequency clock runs at 50 MHz, which converts to 50 million ticks per second.

Organizing the Livewire's software needs into appropriate tasks was done based on priority and computational load. The main jobs of the Livewire software are to acquire data, log data to an SD card, stream data, accept user input, and toggle the control pin. Grouping everything into one task would require a massive stack size and could place a lot of strain on the

processor that could lead to the system freezing. Partitioning the software into multiple tasks increases performance and makes the software easier to debug.

The highest priority task is the data acquisition task. Its main function is to gather data from every source and combine it into one string. The combined data string whether encoded or decoded has a delimiter, timestamp, CTD data, signed Hall effect rpm, and an ADC voltage. The ADC voltage isn't used yet but is set up to help read voltage later of a battery. Data acquisition is synchronized with the SBE49 because its 16Hz clock is fast and sets the control system to a consistent rate. Using any of the STK timers would have worked too, but that would create additional work for data synchronization and additional load on the system. The data handling and control tasks are also based off of the receipt of a new SBE49 packet. The STK continuously reads the SBE49 serial port and checks one character in its read buffer at a time to find the SBE49 delimiter, "\n". This synchronizes the task with the SBE49 because the next expected character would be the first one of a new data packet. The next 24 bits are read since this is the full SBE49 data string length. After checking for the full delimiter, "\r\n", the SBE49 string is decoded from hex to decimal, and the full data string can then be assembled by grabbing a timestamp from the STK's sleeptimer, the Hall effect rpm, and the ADC pin value. The ADC pin is set up with the intent of measuring analog battery voltage or analog motor commutation- features that are not yet implemented. Because the Hall effect code is interrupt driven, the last calculated rpm is assumed to be the current rpm and included with each new SBE49 packet. Two data strings are created: one encoded for logging and one decoded for streaming out.

Circular buffers are a way to store data that balances memory management with the desire to capture every incoming data packet. The Livewire software features two such buffers: one for encoded data and one for decoded data. Each buffer is a struct in C that can store up to 100 entries. Incoming data fills the buffer, then the 101st entry overwrites the first entry and so on. The data acquisition task is a producer that fills circular buffers, and intelligent use of these

buffers requires consumers that use data from the buffers at the rate they are added. Timing the Livewire system to the SBE49 16Hz clock is possible and best in the simple control application because the logging, streaming, and control loops are all set up to access the buffers right after a new entry is added. An alternative to this method would be to process data one entry at a time, but that would naturally consume data slower than it's produced and desync from the input stream. Writing data to a queue with no size limit is dangerous because it could consume all the system's memory, and once data is used there wouldn't be a reason to hold onto it.

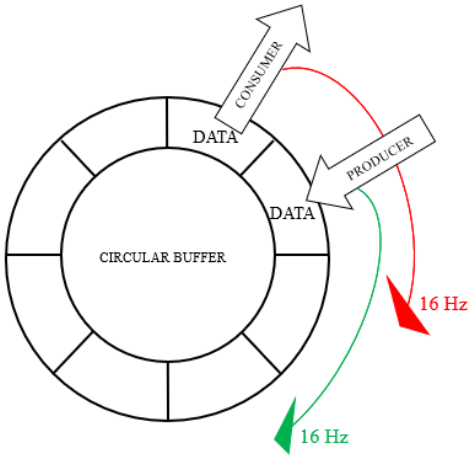

Figure 11. Circular buffer. Note that the consumer doesn't remove data- it represents software making use of data after the producer adds it to the buffer.

Logging and streaming the data is combined into one task. This task is the most intense on the processor but is relatively less important than the data acquisition and control tasks. File management on the SD card was tricky to get right due to an aging driver and processor load. After the data acquisition task creates the decoded and encoded data strings, the logging and streaming task grabs them both from their respective circular buffers. The decoded string is streamed out via the VCOM port USART. Then, the encoded string is written to a text file.

```
LWG data is formatted as follows: timestamp - hex encoded CTD data - hall rpm - adc
  $LWG,0000014375,000005000A110000750002,0000000000,0000000000
  $LWG,0000014454,000005000A110000740002,0000000000,0000003828
  $LWG,0000014533,000005000A110000730002,0000000000,0000003575
  $LWG,0000014612,000005000A110000720002,0000000000,0000003386
  $LWG,0000014690,000005000A110000710002,0000000000,0000003242
  $LWG,0000014769,000005000A110000700002,0000000000,0000003108
```

Figure 12. Livewire data file setup with comma-separated values.

The 78 millisecond gaps between the timestamps in Figure 12 show that the system is not entirely in step with the SBE49 clock, and this is likely due to the runtime of each task including its required delays. The timestamp is grabbed in the data acquisition task. The tasks that check for a new data entry like the logging and streaming task currently check only while they are running. It is possible that another method, perhaps using interrupts or round-robin task scheduling, but 12.8 Hz based on the 78 millisecond gaps is sufficient for Livewire controls because it just needs to make decisions in the context of ocean waves. The frequency of surface ocean waves depends on wind speed, and we expect surfaces waves of interest to have a period of at least one second. The sampling rate of the Livewire system is an order of magnitude greater than the waves it will feel, so missing a value due to buffer overflow or task delays is acceptable so long as the producers and consumers otherwise work at a consistent rate.

Communication between a laptop and the STK is simple on a test bench and doable for shallow water tests. The USB power cable between the STK and laptop also serves as a USART, so no additional pins are needed. For submerged tests with limited depth, a 30-foot tether cable links the pressure case to a laptop on land. Mike Goldin and Mai Bui of the Multiscale Ocean Dynamics Lab laid the groundwork for command-line interface, or CLI, communications and SD card file system management for the lab's work with the STK. The CLI allows a user to communicate to the STK by typing on a keyboard within a terminal connected to one of the STK's serial ports. The base level of the CLI allows a user to view and create files and directories, and mount and unmount the SD card. It has since been modified for this project to allow a user to tell the Livewire system to profile or hover at user-specified depths.

2.5 Control System

A finite state machine is a system that can be in only one state at a time. The Livewire's states are defined as: initialization, idle, upward, downward, and hover. Both state machines use the Livewire's position and user input to determine whether or not they should switch between these states. The only difference between the two state machines is that one is based on the SBE49 pressure data while the other uses the Hall effect-based displacement as its inputs. The pressure-based machine is a closed loop controller because the SBE49 pressure is calibrated and provides accurate positional feedback. While the Hall effect calculations are a form of positional feedback, the machine is more akin to an open loop controller because it has no way of correcting itself if the Livewire slips or otherwise changes position without triggering the Hall effects. The only reference point for the Hall effect displacement currently is that it is zeroed when the GG11 is first powered on. The Livewire controller is defined around profiling, so when the receipt of a new SBE49 packet alerts the system to check its state, it checks based on an upper depth limit and a lower depth limit. While in the upward state, the Livewire only allows upward motion until it reaches the upper limit, and the opposite happens in the downward state. The hover state merely sets the upper and lower limits to equal the same value. It sets the Livewire to profile a length of zero meters at a target depth. The Livewire only enters the initialization state on startup. Once a new SBE49 packet is received, it moves to the idle state. The Livewire never returns to the initialization or idle state. This method stems from best practices. While the system is warming up in these two states, sensor noise from starting up is kept from causing unwanted behavior. For example, the Hall effect sensors on another motor showed a huge, negative rpm on startup several times even though the motor wasn't moving.

To achieve positioning based on a target depth or depth boundaries, the Livewire simply allows the waves to propel it only in the direction that brings it closer to its target. The Livewire is either fully braking or freewheeling. This simple form of control is enacted by a lag controller,

which is analogous to a thermostat. A thermostat regulates temperature by letting hot or cold air flow to reach a target temperature. If the temperature deviates from the target, the thermostat just tries to reverse the deviation. A user can change the target temperature at will. These three thermostat traits are represented in the Livewire controls. In this case, the waves are to position what air is to temperature for the thermostat. It regulates position by allowing the Livewire motor to rotate either clockwise or counterclockwise to ascend or descend the cable based on depth and direction data. If the Livewire is pushed beyond a boundary, it uses the waves to return to it. A user can change profiling or hover depths through the CLI at any time. The Livewire controls are isolated to their own task that checks the state of the Livewire every time it receives a new SBE49 packet. It is a higher priority than logging and streaming and self-contained to keep the controller simple, independent, and as fast as possible.

2.6 Controller Overview

The Livewire's thermostatic controller established a launching pad for additional Livewire development. It is a data acquisition and storage system that provides positional control of the Livewire based on depth and direction data. While the pressure case package will evolve to include more sophisticated software, motor control hardware, and battery management, the current system featuring the STK persists as a customizable, working system that communicates with the Livewire and onboard sensors.

2.7 Software Overview

Upon startup, the software first runs through core initializations and then application initializations, which set up tasks, a data file, and allocates memory for the Livewire circular buffers. If the STK receives SBE49 data, it will gather all data in its data acquisition task. Then, a global write counter ticks up. The logging and streaming task and control task continuously check this value to determine if they should run their loops or sleep. The control task takes precedence

because it has a higher task priority. Hall effect data is gathered via an interrupt routine that calculates rpm and direction. The software was coded in C through the Simplicity Studio IDE and runs on the micriumOS kernel on the STK board.

CHAPTER 3: CONTROLLER DEVELOPMENT PROCESS

3.1 Data Acquisition Software Development

I got started on the Livewire software and electronics development with simple example projects on the STK. The first milestone was to demonstrate effective capture of Hall effect pulses. This meant writing software to monitor the status of GPIO pins and writing an algorithm to calculate rpm and direction during the GPIO interrupt routine. Simplicity Studio provides a pushbutton example that uses GPIO interrupts raised by the buttons to toggle LEDs. This example was used to build out the Hall effect code- the current software still blinks the STK's LEDs when the Hall effects trigger. The two pushbuttons mounted on the STK were used to simulate Hall effect inputs and demonstrate the Hall effect code. It worked, but the pushbuttons couldn't provide an exact, known rpm to reference.

Because the pushbuttons are technically GPIO pins, the button-based Hall code was retained but with the pins remapped using the Simplicity Studio slcp setup file. The slcp provides an interface for users to set up or enable functions like serial ports and GPIO pins, and often they can be changed using a simple drop-down menu of available options. Simplicity Studio uses the slcp to autogenerate handles specific to the selected options that are used to fill out generic templates of functions. When a GPIO pin was always reading as HIGH despite a lack of inputs, I used the slcp to quickly change pins. It also lists every software component available and included on the STK along with a button to view the source code. This was used to debug and patch some of the sl_iostream code used to read in the SBE49 data and stream out to the terminal.

Pushing the buttons quickly was one way to test the Hall effect code, but the next step was to provide more realistic inputs. Simulated Hall effect signals were provided courtesy of a Waveforms Analog Discovery 2, or AD2, which is an oscilloscope and logic analyzer. Three GPIO pins were connected to three of the AD2 pins, and the Waveforms software was used to

generate square waves offset by 60 degrees each. The typical spacing between Hall effect sensors is either 60 or 120 degrees, and it's 60 degrees for the Livewire. Testing only required a laptop, the STK, and the AD2 sending signals via Waveform as in Figure 13.

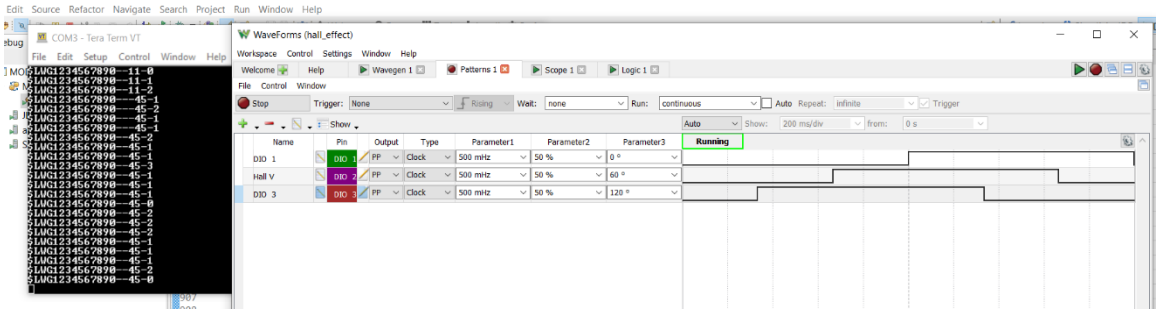

Figure 13. Waveforms interface showing three signals set up to simulate Hall effect pulses. Changing the frequency of the trio simulates different rpms. Flipping the phase difference changed direction. The terminal on the left shows the STK streaming data- this setup simulates 45 RPM counterclockwise.

It was at this point in development that other features were narrowed down like the usage of the SBE49 and its clock, logging on the SD card, and streaming out to a terminal. The task structure was also implemented. Before the control code was written, the following tests on the motors were primarily focused on getting accurate Hall effect rpm and direction. The Hall effect portion of testing was relatively straightforward because issues that showed up in the logic analyzer were due to simple errors and quickly fixed. Most of the issues encountered during testing were during development of the logging and streaming functionalities. The aforementioned patch to the sl_iostream was written by Mike Goldin, who fixed the issue where a read function read only a few bits at a time instead of reading in the requested number. Before that, I'd written a workaround that achieved the same result but with more computations. Mike also resolved an issue in a read function where the developers wrote in a condition for the software to hang forever. Issues with writing to the SD card weren't explicitly resolved, but I was able to successfully build and validate the logging task after the usual isolating and debugging. Creating the text file to store data was moved into the initialization stage of the code because it would cause the system to hang if inside a task loop, function, or initialization. In one version of

the Simplicity Studio software, a large increase in stack size enabled the logging, but in the most recent version, that stack size increase caused the whole system to freeze. Reducing the stack size down to its initial size resolved logging in the most recent version even though the SD card driver didn't change. Regardless, logging and streaming were developed and achieved in parallel with the following tests.

3.2 Software Validation on the Kinco Servo Motor

Before testing on the Livewire itself, I tested the software on a smaller, cheaper, off-the-shelf Kinco servo motor was a good intermediary step for educational purposes and to make sure the nascent control system was safe to install. My goal was to stress test the code with real hall effect inputs. Changing the pulses per revolution parameter in the code was the only software adaptation needed to make it work. The Kinco test was necessary to introduce the electronics that would be required to connect Hall inputs from a real motor to the STK pins. The circuit was relatively simple and included a connector cable, a jumper board, and the level shifter. The jumper board was used throughout testing to view the Hall effect pulses using an AD2. This demonstrated that there are spots where the Hall effects are not triggered when the motor is oscillating such that the sensors remain viewing the same poles. During this test, the Hall effect sensors of the Kinco motor spiked on startup and gave an rpm of about two million. On the next Hall effect trigger, the rpm was correct, but this startup behavior was described in section 2.6 as a reason to let the system run a few idle cycles before making any control decisions.

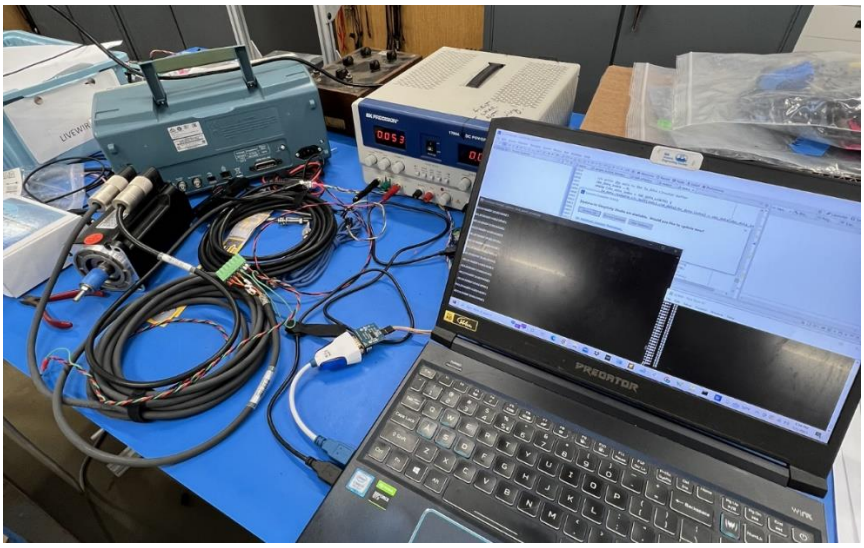

Figure 14. Bench test setup for the Kinco servo motor. The power supply powers the motor's Hall effect sensors. Of note here is that the Kinco motor's cogging made it easier to validate Hall pulses.

After verifying the hall effect code on the Kinco motor, I tested the software on the Livewire. The goal here was to make sure that data acquisition, streaming, and logging tasks were working correctly by looking at the hall effect and SBE49 data in the stream and SD card. The SBE49 simulator was useful because it could be programmed to provide various inputs and could be paused in the Windows Command Prompt if anything unusual appeared on the STK output stream. The data stream and user input both use the same terminal to display on the computer, so pausing the simulator made it easier to debug by cross-referencing simulator inputs to data stored in the SD card or streamed on the terminal. Plugging into the Livewire required additional circuitry compared to the Kinco motor. Thomas Kelly built a prototype circuit to switch on and off the Livewire braking by hand. It also provided leads for viewing the voltage across the 4.3 Ohm load using a multimeter. I added the level shifter to that circuit to make the Hall effects work and swapped the manual switch for a MOSFET to give the STK digital control over the relay.

3.3 Controlling the Livewire

Development of the control loop began after these tests validated the data acquisition, streaming, and logging processes. The first goal was to demonstrate any response to an external

input, so the Hall effect inputs were used to toggle a GPIO pin based on the Livewire's rotation direction. If it was rotating clockwise, the STK would set the GPIO pin LOW. If it was rotating counterclockwise, the STK would set the GPIO pin HIGH. This would switch in and out the load. The physical output of this test was to make the Livewire easy and hard to rotate. The STK was used to stream out the state of the pin while the AD2 showed the Hall effect pulses on its Waveforms program to validate the control.

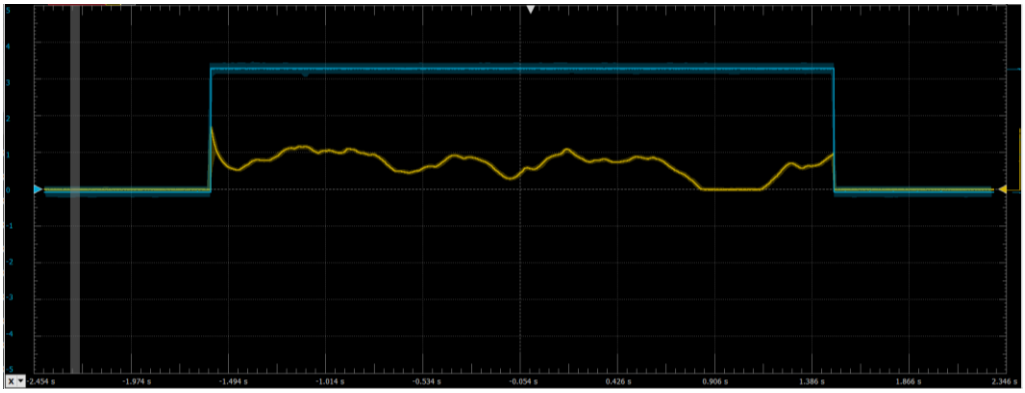

Figure 15. Blue shows the voltage of the control pin. Yellow shows the voltage drop across the load resistor. This oscilloscope picture shows zero energy harvesting on either end of the control toggle and rpm-dependent generation when enabled.

Toggling a GPIO pin occurs quickly, but switching the brakes on the Livewire did not physically feel as fast because there is a delay between physically changing direction and detection of that change. There was a consistent ramping-up braking and ramping-down release feeling to the Livewire when the control pin was toggled that lasted a fraction of a second. This delay between freewheeling and full braking modes helps characterize the motor but doesn't cause any significant issues in the context of the lag controller. There are two causes for this phenomenon. One is the spacing of the Hall effect sensors because their resolution for determining rotor position is coarse compared to a motor encoder that explicitly tracks motor shaft position. For a given state of the Hall effect sensors, it is possible for rotation of the magnetic poles such that the sensors remain in the same state. This wiggle room is slight but can delay the brakes toggling because that depends on the direction sense updating. The other cause for the ramping is that the GPIO pin doesn't instantaneously switch between 0 V and 3.3 V.

Based on the oscilloscope readings in Figure 16, the GPIO pin takes between 4.8 and 5.0 milliseconds to switch either HIGH or LOW. The voltage drop across the resistor rises and falls in time with the GPIO pin regardless of the amplitude of motor rotation.

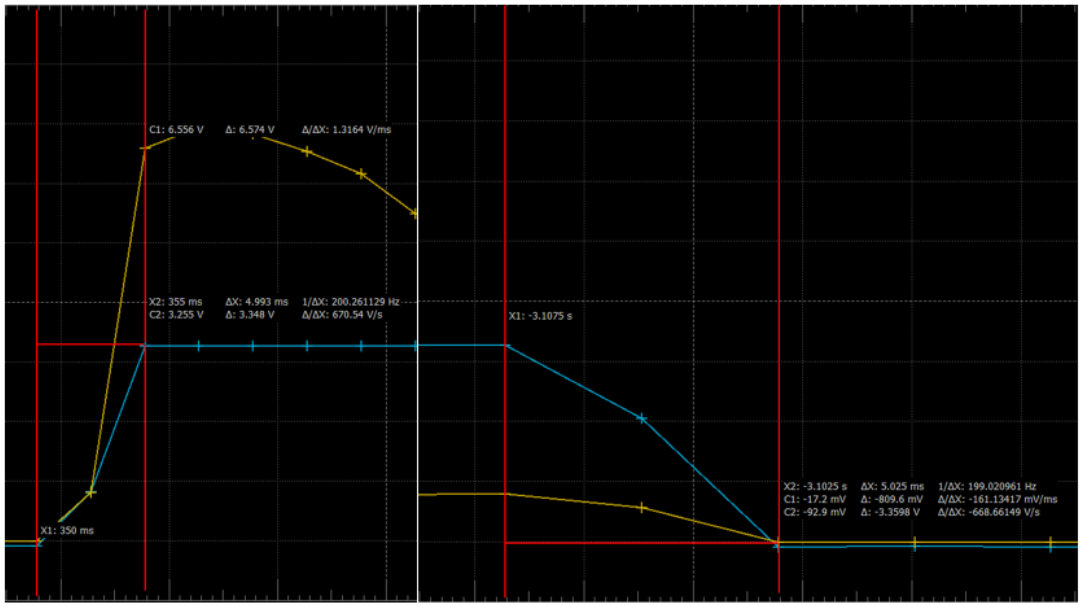

Figure 16. Oscilloscope showing the load voltage rising and falling with the GPIO pin toggle. The yellow line is the load voltage. The blue line is the control pin voltage. The Livewire was rotated at various speeds.

Switching the Livewire rotation direction can be broken down into three steps. First, the rotation direction switches but is not yet sensed by the Hall effects. The load remains either fully on or fully off. Then, the Hall effects detect the change in direction, and when the next SBE49 packet is received, the control loop toggles the GPIO pin. The load varies during this step. The third step is when the load is fully on or off in the correct direction. The combination of the first two steps and the expectation of an instant reaction from the system creates the ramping feeling of the Livewire.

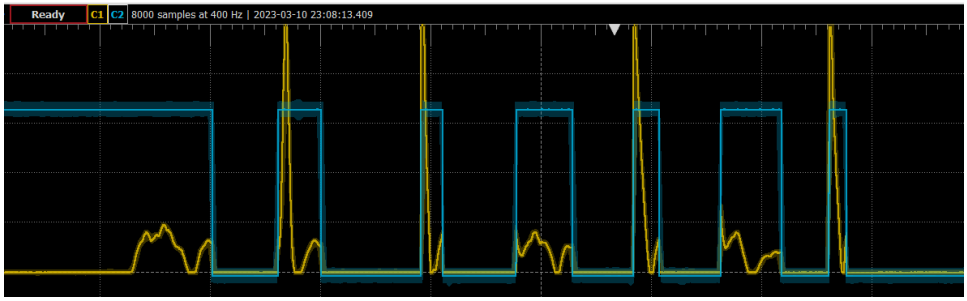

Figure 17. Oscillating the Livewire. The load resistor voltage is directly related to the Livewire motion.

Demonstration of the direction-based control spurred quick development of the Livewire thermostatic controller. It was first written with hard-coded depth values, proven successful on the bench, and then successfully tested with the user inputs for profiling and hover added. A sample profiling plan was issued to the Livewire to keep between 1- and 3-meters depth using the Hall effect displacement state machine. The Hall effect displacement was used because neither the actual SBE49 nor the SBE49 simulator could be quickly or easily configured for a bench test where the Livewire wasn't moving. Coupling the SBE49 pressure, real or simulated, to the Livewire's rotation will occur when the Livewire-Wirewalker system is prepared for a pool test though. While the cable-gripping mechanism remains installed on the Livewire on the bench, the test was done by rotating the Livewire rotor by hand since it would be easier to validate the software by avoiding the cable slip issue. Although it would have been interesting to test with SBE49 data, the success of the displacement-based control is a proxy for the success of the pressure-based control because both machines are functionally the same except for the sources of their depth data. The SBE49 simulator was used for its clock rate, which means the control loop was able to access the global data struct where the SBE49 pressure is stored. It did work when providing a fixed depth, but further testing with the actual SBE49 would confirm its validation.

Packaging the electronics for a pressure case was the next step in Livewire development. The circuitry external to the motor leads was disassembled for redesign to accommodate the STK and Hall effects. Most elements on the circuit were connectorized on one protoboard with the

STK for ease of development. The components that are mounted in sockets in Figure 18 are easily replaceable, and the sockets have long pins suitable for wire wrapping in Figure 19, which made connecting pins quicker. Wire wrapping makes connections between pins by physically wrapping stripped wire tightly around the pin. A wire wrap gun was used for this that automatically stripped wire while making the connections.

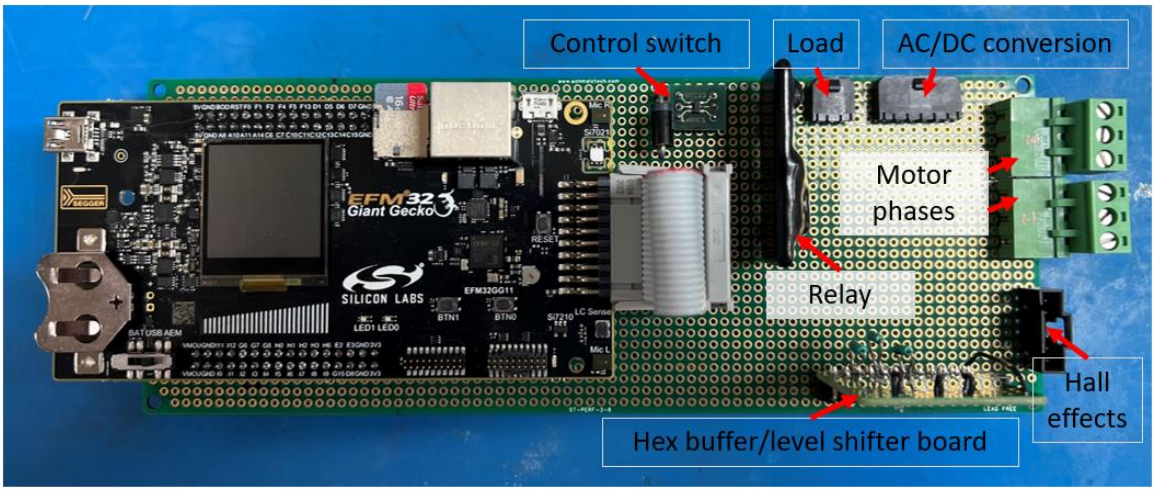

Figure 18. Connectorized board showing connector functions. Figure 19 below shows the wiring between them. There is space for more components, and the existing connectors can be used to swap peripherals.

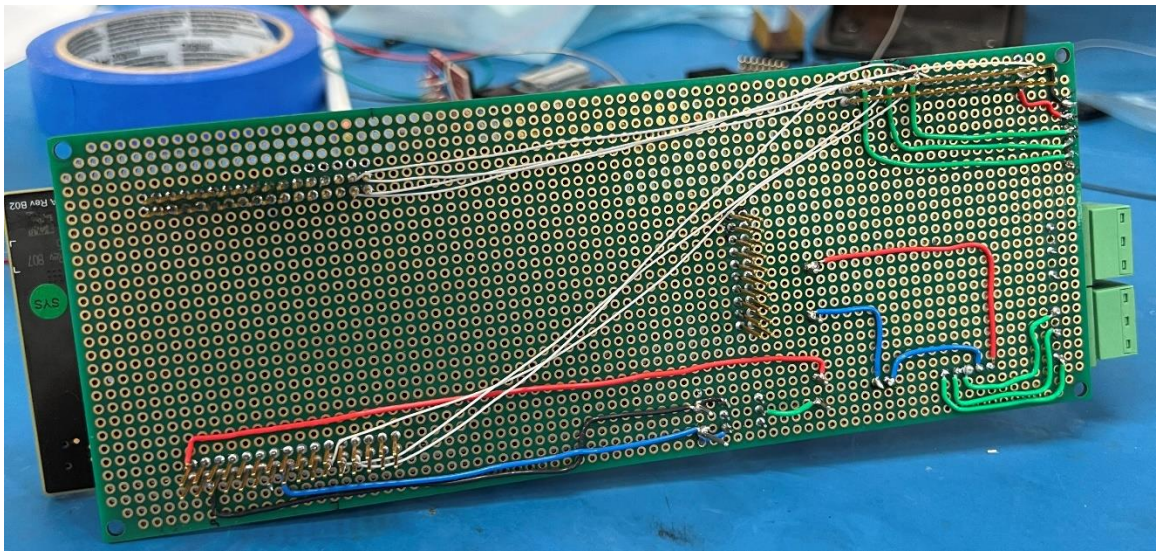

Figure 19. Underside of the Livewire control board showing wire wrap connections in white between the STK and peripherals. These connections are made mechanically and have a special tool for removal.

Demonstrating the Livewire's regenerative braking was done by hooking up an ideal diode bridge, capacitor, and resistor together. The three positive motor phases were fed into the three diode bridge inputs, and the three negative phases were shorted out for wye configuration of the motor. This test was also used to determine the suitability of the ideal diode bridge in replacing or complementing the rectifier. Their circuitry is similar because they are both three-phase bridge devices, but the ideal diode uses MOSFETs instead of Schottky diodes. The MOSFETs dissipate less power and consume less voltage than the Schottky diodes, so less heat is generated, and more DC power is available to the charging circuit.

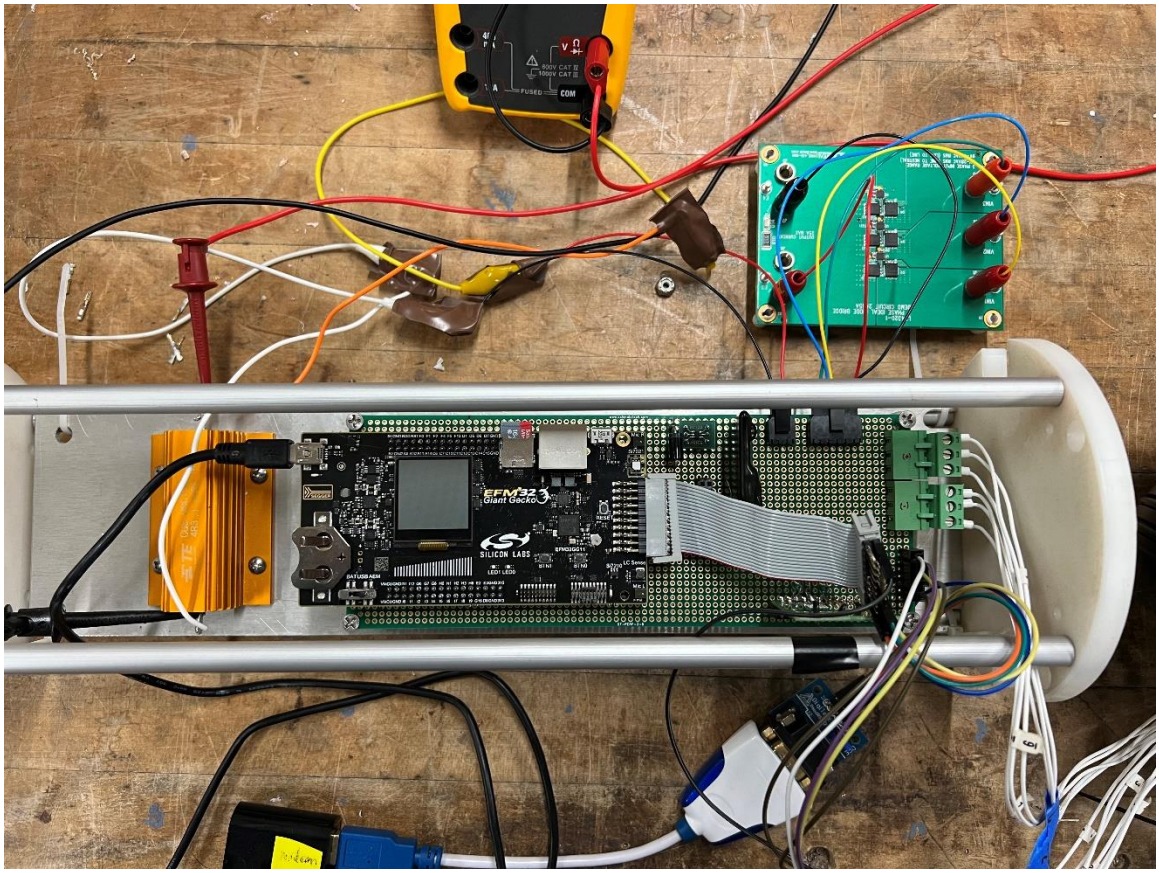

Figure 20. Test setup with ideal diode bridge on the top right.

3.4 Next Steps

There are two subsystems left to pack in the pressure case. The first is the communications. Communication with the Livewire in shallow water or in the pool will be done

through a tether cable and will allow the same functionality from the bench tests: CLI input and data streaming. Integrating an actual SBE49 CTD would only require a few additional components, one of which would be another subsea cable. The second subsystem to flesh out would be the charging system and use it to power the Livewire electronics. This charging system would need a battery, a charge controller, and a way to redirect excess power when it is fully charged.

The lag controller will be succeeded by a full four quadrant motor controller. Four quadrant motor control divides motor operation into four stages: forward motoring, forward braking, reverse motoring, and reverse braking (Kim 2017). This requires control over motor torque and speed in both directions. While the lag controller can technically put the Livewire into those four stages, it can only control by reacting to wave motion and does not have active control over the motor. Implementing PWM control over the Livewire by pulsing the control pin is a next step and would make use of the STK's two ADC channels to try to better follow the wave motion. Pairing the STK with an off-the-shelf motor controller, such as the O-Drive Pro, would give the Livewire system the capability to precisely control its position and charging. It could charge its battery with the waves and then use that power to motor along the cable to maintain a specified profiling rate.

One more element that would complete the system would be an auxiliary power supply to kickstart the control board and onboard sensors if the main battery pack was completely expended or otherwise becomes unavailable. This could happen on a day where the winds and waves completely die down, or it could be dead on deployment. The auxiliary power system would run only sensors required to detect if there is sufficient wave motion to recharge. Because the Livewire could extend the operating time of Wirewalker profiling, it is important to design a backup system so that the Livewire-Wirewalker could operate autonomously and indefinitely.

CONCLUSION

The Livewire electric motor-generator harvests energy from ocean waves to power itself and the onboard sensors on a Wirewalker vertical ocean profiler. It is intended to replace the Wirewalker's mechanical camming system as its method of movement. By engaging and disengaging from the cable, the Livewire can act like a cam but allow motion freely in either direction along the cable. When the Livewire clamps onto the cable, it switches on an electrical load that turns generated AC current from the Livewire's rotation into DC power to recharge an onboard power supply. The Wirewalker's autonomy came from a ratcheting system, but the Livewire requires a more sophisticated control system. The development of the thermostatic lag controller led to successful demonstration of Livewire motion control by toggling a load fully on or fully off to allow the cable to move in a desired direction. The control is driven by the 16 Hz clock of a Seabird49 CTD. Pressure data is collected along with Hall effect rpm and direction using the new data acquisition system. User input through the CLI gives the Livewire's state machines parameters for profiling or hovering in place. Simple charge and discharge of a capacitor was completed, so the system can now progress to developing onboard power and four quadrant motor control.

REFERENCES

Corredor, J. E. *Coastal Ocean Observing: Platforms, Sensors and Systems*. Germany, Springer International Publishing, 2018.

Goodstal, G. *Electrical Theory for Renewable Energy*. United States, Cengage Learning, 2012.

Kim, S-H. *Electric Motor Control: DC, AC, and BLDC Motors*. Netherlands, Elsevier Science, 2017.

Shaalan, H. E., and J. H. Bentley. *Electrical Engineering: A Referenced Review*. United States, Kaplan, 2005.

Smith, J. A., R. Pinkel, M. Goldin, O. Sun, S. Nguyen, T. Hughen, M. Bui, and A. Aja. 2012. "Wirewalker dynamics". Journal of Atmospheric and Oceanic Technology 29(1):103-115, http://dx.doi.org/10.1175/ JTECH-D-11-00049.1.