ULTRASONIC COMMUNICATIONS SYSTEM

FOR HEALTH MONITORING OF

HYDROKINETIC TURBINE BLADES

by

ANDREW JAMES HECKMAN

A THESIS

Presented to the Faculty of the Graduate School of the

MISSOURI UNIVERSITY OF SCIENCE AND TECHNOLOGY

In Partial Fulfillment of the Requirements for the Degree

MASTER OF SCIENCE IN MECHANICAL ENGINEERING

2012

Approved by:

Joshua L. Rovey, Advisor
K. Chandrashekhara
Daniel S. Stutts

ABSTRACT

A health monitoring approach is investigated for hydrokinetic turbine blade applications. In-service monitoring is critical due to the difficult environment for blade inspection and the cost of inspection downtime. Composite blade designs provide a medium for embedding sensors into the blades for in-situ health monitoring. The major challenge with in-situ health monitoring is transmission of sensor signals from the remote rotating reference frame of the blade to the system monitoring station. In the presented work, a novel system for relaying in-situ blade health measurements in hydrokinetic systems is described and demonstrated. An ultrasonic communication system is used to transmit health data underwater from the rotating frame of the blade to a fixed relay station. Data are then broadcast via radio waves to a remote monitoring station. Results indicate that the assembled system can transmit simulated sensor data with an accuracy of ±5% at a maximum sampling rate of 500 samples/sec. A power investigation of the transmitter within the blade shows that continuous max-sampling operation is only possible for short durations (~days), and is limited due to the capacity of the battery power source. However, intermittent sampling, with long periods between samples, allows for the system to last for very long durations (~years). Finally, because the data transmission can operate at a high sampling rate for short durations or at a lower sampling rate/higher duty cycle for long durations, it is well-suited for short-term prototype and environmental testing, as well as long-term commercially-deployed hydrokinetic machines.

ACKNOWLEDGMENTS

In having completed my thesis paper, I would like to say thank you to the many people that helped me get this far. So thank you to my advisors, Dr Rovey, Dr Stutts, and Dr. Chandrashekhara. Thanks also to all the folks in the lab who've helped me out and answered questions when I had them. Finally, thanks to my family for the help they have given in getting me this far.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

1. INTRODUCTION

1.1. BACKGROUND

The energy in flowing river streams, tidal currents, and waves is being investigated as a potential source of renewable energy[1-2]. Because hydrokinetic systems must be located underwater where access is difficult, monitoring technologies are important for timely notification of operation and maintenance (O&M) issues. In the following sections we describe a data transmission system that can transmit structural health information about the turbine blades of a hydrokinetic energy system.

Hydrokinetic energy systems are designed to convert kinetic energy of a fluid (e.g., river or tidal current, waves) into electricity. They are generally classified in two broad categories: (a) wave energy conversion devices and (b) rotating devices. Wave energy conversion devices create a system of reacting forces in which two or more bodies move relative to each other while at least one body interacts with the waves. Rotating devices are deployed within a stream or current capturing kinetic energy from flowing water via a turbine that powers a generator. In a typical rotating device, the river or tidal current passes through a protection screen and into the turbine channel. Kinetic energy of the fluid causes the turbine to rotate and this rotational energy is extracted by the generator attached at the top of the turbine. Power from the hydrokinetic turbine is then coupled to the electrical grid through a power converter. A number of resource quantization and demonstrations have been conducted throughout the world and it is believed that both in-land water resources and the offshore ocean energy sector will benefit from this technology[3-4]. A 2009 market analysis by Pike Research predicts the

marine and hydrokinetic energy market value to grow considerably over the next 5 years. Specifically, at the end of 2008, the total installed capacity of marine and hydrokinetic systems was only 10 MW. By 2015, market predictions based on planned projects and installations suggest that between 2.7 to 10 GW of installed power will be available, representing a market value between $6 to $20 billion [5]. With this much potential gain, there is much interest in the continued development of hydrokinetic energy systems.

Hydrokinetic energy systems must be remotely monitored. By their very nature hydrokinetic systems are remotely located underwater in areas that are generally difficult to view, inspect, and analyze. In addition, many hydrokinetic energy systems plan to be located off-shore or in remote areas far from dense population. For example, a study of potential Alaska river in-stream power plants selected three river locations where the nearest villages have populations of 50-100 people and the nearest cities are 100's of miles away [6]. Simply getting to these sites represents a significant challenge. Accessing the components to perform any kind of analysis would be a difficult, expensive, and nearly impossible task.

The main benefit of remote monitoring is lower O&M costs and a higher return on investment. The Pew Center on Global Climate Change recently found that "O&M costs could remain high due to difficult access and working conditions unless machines are developed that can be unattended for long periods of time" [7]. In other words, leaving hydrokinetic systems unattended and remotely monitoring them is imperative for reducing O&M costs and making it a viable and promising technology. It is difficult to estimate the future O&M costs of hydrokinetic systems because the technology is still being developed. However, wind energy systems are plagued by similar issues and, over

the lifetime of the machine, have typical O&M costs of 70-95% of the total investment cost [8]. These estimates were for a project that considered six hundred 750 kW wind turbines. O&M costs are typically higher for smaller power machines and near-term hydrokinetic systems are likely to be a few 100 kW, in which case O&M costs may be higher. In other words, reducing O&M costs with remote monitoring is imperative for making these systems viable on a large scale with widespread use.

The structural health of hydrokinetic energy systems needs to be monitored. While methods for remotely monitoring hydrokinetic system parameters such as voltage and power output have been developed, the ability to monitor the structure, specifically the turbine blades, has yet to be developed. Monitoring the turbine blade is essential as it is the critical component for energy extraction and as such it also bears the most dynamic load. Fatigue causes degradation of structures and eventually leads to structure failure. Over time, small cracks develop and propagate, reducing the structure integrity, and eventually cause it to fail. In addition to the natural slow fatigue of the blade structure, there is also significant concern about transient impacts and loads due to environmental factors. Oceans and rivers are heterogeneous bodies of water that contain aquatic and marine organisms, vegetation due to run-off, and pollution that pose a significant hazard to hydrokinetic energy systems. Multiple reports have been published evaluating the effect of hydrokinetic energy systems on marine life (e.g., fish and other aquatic organisms, diving birds, and mammals) [9-12]. While a single impact may not be immediately catastrophic, it can create a crack at the point of impact that eventually propagates, compromising the structural integrity of the blade and leading to failure. In addition, other transient environmental effects may cause increased loads on the blade,

degrading structural integrity. For example, earthquakes, tsunamis, and flooding are all natural phenomena that will occur and may cause damage.

There are multiple scenarios where remote monitoring of turbine blade structural health is beneficial. For example, it can lead to timely replacement of a blade by notifying service personnel when the blade structural integrity is likely to fail; in the event of damage due to transient environmental effects, the remote monitor would immediately notify service personnel so the system could be repaired (patched) quickly to minimize down-time; knowledge of the turbine blade structural health could lead to enhanced operational lifetime. For example, as the blades age, the system may be operated at reduced capacity to reduce structural load, thereby maintaining power generation (albeit at a reduced level) for a longer period of time.

Application of remote monitoring technology to hydrokinetic energy systems is still in its infancy. There are many techniques for monitoring the health of any structural component, but most of these techniques are off-line monitoring technologies, which mean they cannot be used on a system while it is operating. For example, acoustic emissions, thermal imaging, ultrasonic detection, fiber optics, laser Doppler vibrometry, electrical resistance-based damage detection, strain memory alloy methods, x-radioscopy methods, and eddy current methods have all been used to assess the structural integrity of a wind turbine blade [13]. However, of these methods, only acoustic monitoring has been used while the blade was in service [13-15].

The challenge with in-service monitoring the structural health of any turbine blade is that it is a rotating component that is always moving when the system is operating. So connecting a wire to a fiber optic, acoustic sensor, electrical strain gage,

etc. is not practical. In other words, structural health data need to be acquired and then wirelessly transmitted from the rotating frame of the blade. This was the approach that successfully showed radio transmission acoustic monitoring of a wind turbine blade [13-15]. Simply adapting this wind turbine system to a hydrokinetic turbine blade is not possible because in an aquatic environment radio communication is only possible for short distances (< 1 m), at very low frequencies (30-300 Hz), with large antennae, and high transmission power [16].

Structural health monitoring of hydrokinetic turbine blades has never been investigated and is the focus of the work presented here. In the following sections we describe proof-of-principle laboratory experiments that demonstrate transmission of a simulated structural health monitoring signal through the aquatic environment and then wirelessly through air to a remote monitoring station. A schematic of the general concept is shown in Figure 1.1. Composite turbine blades, embedded with a fiber optic strain gage and acoustic transducer, are attached to the turbine and used to generate electricity. The fiber optic strain gage senses the degradation of the blade structure over time due to cyclic loading (fatigue) and transient environmental factors. A power and electronics module inside the blade conditions the fiber optic strain gage signal into an acoustic signal that is transmitted by the acoustic transducer. The acoustic waves propagate through the water to a receiver that is located near the shore or on the system foundation. The received acoustic signal is then broadcast above water long distances by radio waves to the monitoring station. The broadcast signal is then interpreted at the monitoring station, yielding strain data from the blade, which can be used to notify service and maintenance personnel.

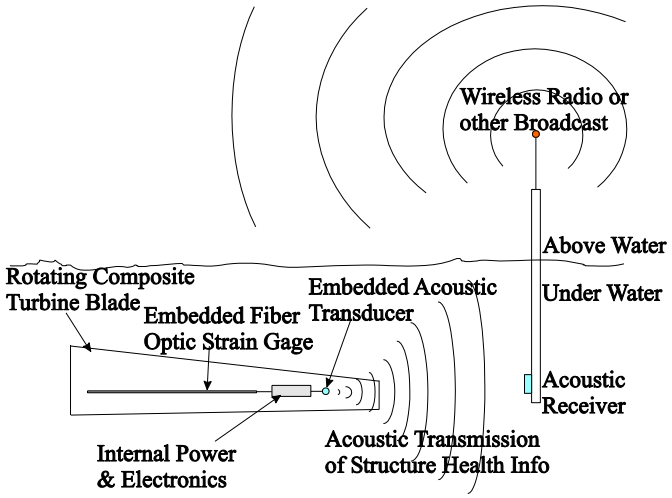

Figure 1.1. Schematic of the hydrokinetic turbine blade structural health monitoring concept

In the following sections we describe our proof-of-concept demonstration of the data transmission part of the blade structural health monitoring concept. We have already demonstrated our ability to manufacture a composite turbine blade with embedded fiber optic strain gage in Ref. 17. In the following study, a simulated strain gage output was used as the input to the data transmission system. The system was demonstrated in both a bench top environment and on a prototype hydrokinetic turbine in a laboratory environment (water tunnel). In addition to demonstrating the feasibility of the technique, we also present results from a power analysis that is used to assess the viability of the approach and its suitability for both commercial and prototype hydrokinetic energy systems. We envision this novel concept and the resulting capability to enable blade structural health monitoring in both long-term commercial hydrokinetic system deployment, as well as prototype environmental impact studies.

1.2. MOTIVATION

Due to the complications just described that arise in health monitoring and maintenance for an underwater turbine system, new monitoring methods must be used in order to allow for cost effective maintenance of the system.

1.3. APPROACH

In order to properly design this acoustic health monitoring communications system, some basic desired capabilities for the system as a whole were first plotted out. Figure 1.2, shown below, displays the desired stages of operation of the device.

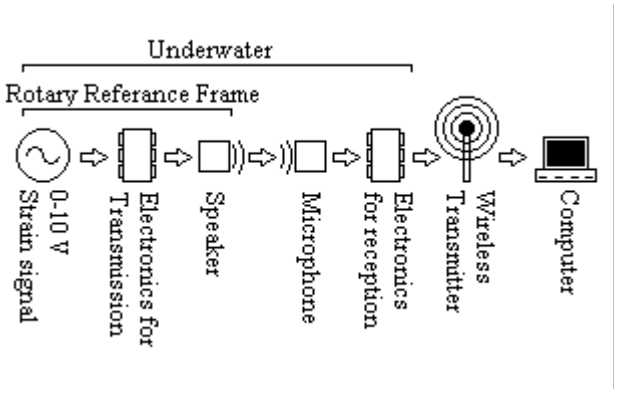

Figure 1.2. Diagram of System Operation

Ideally, the system will eventually generalize to include more types of inputs than just the signal from the fiber optic strain sensor. But for now, the system will be designed with just a strain sensor signal in mind. In designing the finished communications system, two design stages will be completed. The first design stage will replicate the desired capabilities that were outline by any means possible, as quickly as possible, as a proof of

concept setup. The second design stage will be the prototyping stage, taking the first system and refining it into a more practical design that could be used in the real world for its intended purpose. The completion of these design stages will result in a rough, but functional product that can fulfill its desired design goals.

2. DESIGN PROCEDURE

2.1. PROOF OF CONCEPT SYSTEM

This first build was the simple proof of concept design. For this particular case, a normal desktop PC was the vital central component. This computer did all the necessary processing, and a series of peripherals attached to it performed all the rest of the needed tasks. In this case, since no actual strain gauge data was available, a simple 9 V battery was used as a proxy for the 0-10 V signal that would be sent in the real system. A National Instruments DAQ, shown in Figure 2.1, was used to read this signal into the computer where LabView was used to take in the data and send it to a text file.

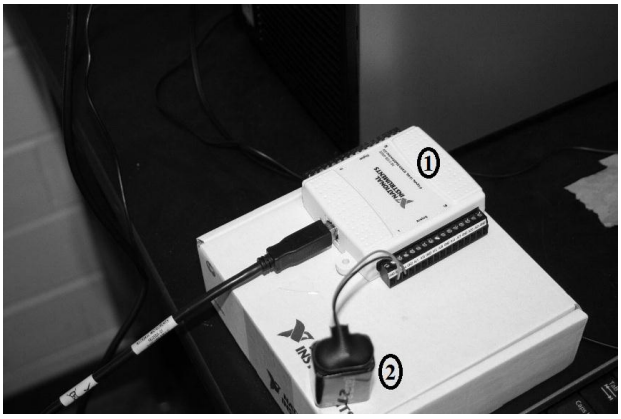

Figure 2.1. DAQ(1) with battery(2) as proxy strain sensor signal

Matlab was then used for the next portion of the process. Upon the text file being edited, a program running in Matlab acquired the new signal data. This data was then turned into a signal comprising of a summed set of sine waves with differing frequencies. This signal corresponded to a ‘symbol’ that was to be then transmitted. For this simple

implementation, an arbitrary set of symbols was set up, eleven of them, corresponding to the numbers 0 through 9 and a period. The signal representing a given symbol was then played from the computer via a simple headphone. This headphone, suitably enclosed as illustrated in Figure 2.2, was placed underwater and the sound it emitted was picked up by a microphone, which relayed that data back to the computer.

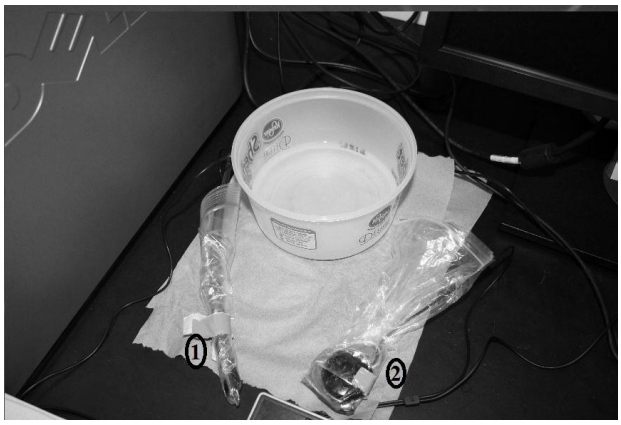

Figure 2.2. Headphone(1) and microphone(2) with waterproof 'enclosures'

There, a Fast Fourier Transform (FFT) was used to determine the frequency components of the received sound. Using this frequency component data, the 0 to 10 V data reading was reconstructed. From here, the data was written once more to a text file, which was then read into LabView. That data was then transmitted using a USB Bluetooth attachment to a similar USB attachment on another computer using LabView's built in Bluetooth functions. At the second computer, the data was received, and could then be used. This total process accomplishes the desired goal as was set out in the approach above. Of course, the system as presented (shown in Figure 2.3) was completely impractical, as an entire desktop is not what one would call 'embeddable.'

Nor was the swapping of data via text files between two operating environments terribly efficient. The next step was the implementation of this system in a more usable format.

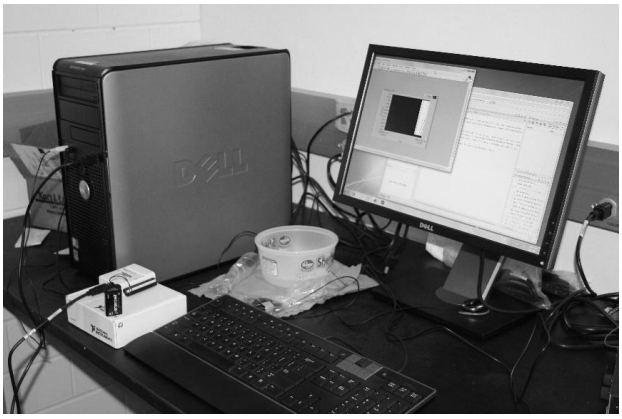

Figure 2.3. Total proof of concept system

2.2. PROTOTYPE SYSTEM

For this phase of the design, the proof of concept system was replaced with a more practical prototype system, in order to insure that the system was feasible from a commercial standpoint. Amongst the changes made was the replacement of the computer with two microcontrollers (MCUs), along with both the speaker and microphone being replaced by a pair of ultrasonic transducers. Furthermore, the encoding of the signal was changed to a much simpler series of 40 kHz pulse so that the ultrasonic transducers could optimally receive and transmit data. Two electrical circuits were used to facilitate the transmission and reception of the signal, whilst the microcontrollers provided the processing necessary to encode and decode the signals. The two circuits, the choice of a

40 kHz transmission frequency, and the MCUs software are explained in more detail below.

2.2.1. Acoustic Transmitter and Receiver Circuitry. The acoustic transmitter and receiver circuits consist of two main components: a microcontroller and an ultrasonic transducer. The foundation of the transmission and reception circuitry is two PIC18F46K20 Microchip microcontrollers (MCUs), one for each circuit. These MCUs provide the necessary processing to encode and decode the data. They were selected because they have a variety of useful built-in features, such as several analog to digital conversion (ADC) ports, the capability of implementing EUSART serial communication, and an extremely low power consumption when 'sleeping.' A pair of MA40MF14-5B Murata ultrasonic transducers serve as the acoustic transmitter and receiver. These transducers are optimized for 40 kHz frequency, and this data transmission frequency was selected based on the rationale described further below. Details regarding both the transmission and reception circuits (shown in Figures 2.4 and 2.5, respectively) are described next.

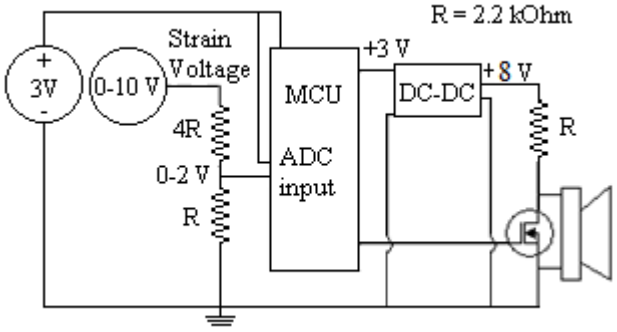

Figure 2.4. Transmitter Circuit

The main tasks of the transmitter circuit are to measure the strain sensor voltage, measure the battery voltage, enter and awake from “sleep” mode, encode the strain data into an acoustic signal, and transmit the data with the ultrasonic transducer. Measuring the sensor voltage and battery voltage is done using the MCUs built-in ADC feature. Measuring the battery voltage allows us to also track the life of the power supply. The voltage divider at the strain sensor is used to lower the 0 to 10 V simulated strain gage signal to 0 to 2 V, since the MCU input pins can only take 0 to 5 V safely and the pins will saturate at voltages over the supplied power voltage. The effect of duty cycle on the potential lifetime of the monitoring system is studied in subsequent sections by using the “sleep” feature of the MCU, which is controlled using the MCU software. MCU software is also used to encode the measured strain gage voltage into the acoustic transmission signal. Finally, transmission of the acoustic signal uses the 40 kHz ultrasonic transducer, a DC to DC converter, and a logic level mosfet. While initial tests showed that the MCU alone can drive the transducer, the resulting ultrasonic signal was too weak to ensure accurate reception. Instead a simple amplification circuit is used. The MCU output operates the mosfet switch, which in turn modulates the transducer driving signal from the DC to DC converter. Finally, a 3 V power supply is used to power the circuit.

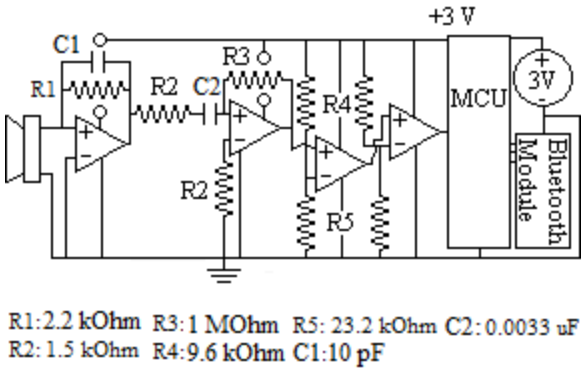

Figure 2.5. Receiver Circuit

The major tasks for the receiver circuit, shown in Figure 2.5, are to receive the acoustic signal, process it, and re-transmit wirelessly through air (Bluetooth in this case). An ultrasonic transducer receives the transmitted acoustic signal. This signal is then conditioned to make it more easily 'readable' for the MCU. First, the signal passes through a charge amplifier which converts the charge developed across the piezoelectric transducer into a voltage proportional to the gain of the amplifier. The signal is then passed through an active high pass filter with a cutoff frequency of approximately 32 kHz and a gain value of approximately 667, thus filtering and further amplifying the received signal. The filtering process allows the required 40 kHz signal through while removing anticipated background noise that would be found in a river, such as rain, wind, and fish acoustics. After being filtered and amplified, the signal is then modified by two subsequent comparators. The comparators convert the roughly sinusoidal signal into a clean square wave pulse train, which is much easier for the MCU to process.

2.2.2. Acoustic Transmission Frequency. A 40 kHz acoustic transmission frequency was selected based on three main considerations. First, ambient noise sources encountered in the hydrokinetic energy system environment, both natural and artificial,

were considered. Second, we considered the audibility of the signal to humans. Finally, the relationship between attenuation rate and data transmission rate at acoustic frequencies was considered.

Two natural sources and two artificial sources of ambient noise are likely to be encountered underwater in a river. The two natural sources are wind and rain noise, while the artificial noise sources originate from boat motors and ultrasonic range/fish finders. Both wind and rain have distinct acoustic power spectrums [18]. Wind noise can be approximated using Knudsen curves, which show that at about 500 to 1000 Hz, the magnitude of wind noise decreases with increasing frequency. While the wind noise does not disappear entirely in the ultrasonic region, it is severely reduced in strength and is not considered a significant noise source. Rain has a distinct peak in strength at around 14 to 16 kHz [18], and then also decreases in strength with increasing frequency. However, rain has been shown to suppress wind noise, and its peak is closer to the ultrasonic range. As such, of the two natural sound sources rain is the more concerning. For artificial noise sources, motor boat noise does not exceed 1 kHz [19], and therefore is not a concern for the ultrasonic frequencies used here. However, the commercial sonar products that many boats are equipped with are a concern, as they rely on producing ultrasonic pings to visualize the underwater environment. The frequency for commercial sonar applications ranges from 20 kHz to 400 kHz generally, and the most common frequencies used are 50 kHz and 200 kHz. Therefore selecting an ultrasonic communication frequency away from 50 or 200 kHz is best for mitigating interference from such ultrasonic devices.

The second criterion we evaluated was the audibility of the communication signal to humans. Specifically, we require the operating frequency to be inaudible to the human ear. This criterion is selected to reduce noise pollution that would potentially be generated by the system operation, and as such it is highly desirable that the signal produced be inaudible to both humans and many aquatic river organisms. The selection of an ultrasonic frequency is an easy way to alleviate this concern, as humans cannot hear in the ultrasonic range and many fish species can only hear in a range of 20 to 300 Hz [19].

The third criterion considered was signal attenuation and transmission rate in an underwater environment. In acoustic communication in water, the rate of attenuation and maximum data transmission rate are directly linked. More specifically, as frequency increases both the attenuation rate and the data transmission rate increase. Thus, a very low frequency signal can travel very large distances in water, but has a very slow rate of data transmission, whereas a high frequency signal travels shorter distances, but has higher data transmission rate. The attenuation of acoustic waves in seawater is shown in Figure 2.6 and is given by Ainslie and McColm in Eqn. 1, [20] where f is the frequency in kHz, T is temperature in degrees Celsius, and D is depth underwater in kilometers. Figure 2.6 assumes a temperature of 22 ºC and depth of 1 m.

$$\alpha = 0.00049 f^2 \exp(-(T/27 + D/17)) \quad (1)$$

The transmission rate is simply the frequency of the waveform divided by the number of periods that are used to represent one bit of data. The system described here uses 5

periods of the 40 kHz waveform to represent one bit, as described in the next section, and the resulting transmission rate is also shown in Figure 2.6. Since having both low attenuation rate and a high data transmission rate are mutually exclusive, the operating frequency for the system must be selected for the systems' requirements.

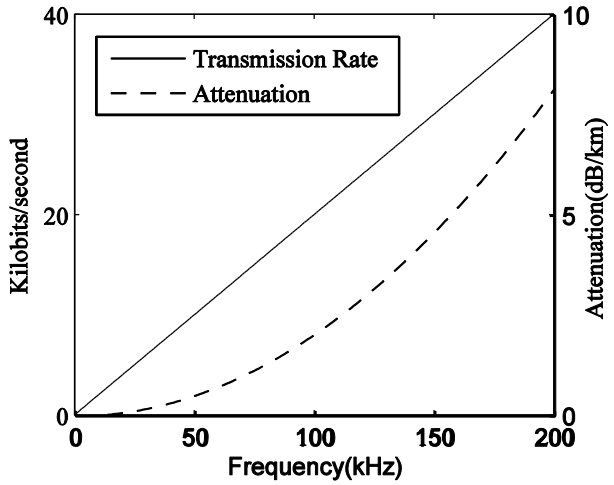

Figure 2.6. Attenuation and transmission rate of transmitter circuit vs. frequency in sea-water (22ºC) at depths of 1m

Based on these factors, we selected a 40 kHz signal. This ultrasonic frequency is not near either 50 or 200 kHz, and therefore alleviates both natural and artificial noise concerns. Because 40 kHz is in the ultrasonic frequency range it is also inaudible to humans and most aquatic animals. Finally, 40 kHz has low attenuation rate on the order of 0.5 dB/km and relatively high 8 kbits/sec rate of data transfer. While higher frequency and the resulting higher data transfer rate would increase the "real-time" nature of the transmission, for this proof-of-concept system we place more emphasis on low

attenuation so as to maintain a high signal-to-noise ratio and maintain reliable communication.

2.2.3. Data Encoding. The software running on the two MCUs encodes the structural health data for underwater transmission from the rotating blade and then decodes the data once received at the static relay station. In order to transmit the acquired data (simulated here as a voltage signal between 0-10 V), the transmitter MCU encodes the data before it is sent. This encoding for transmission is a modified on-off keying (OOK) encoding scheme, where a '1' data bit is indicated with 5 periods of the 40 kHz sine wave and a '0' data bit is indicated with a period of silence that lasts the equivalent time of 5 periods of the 40 kHz sine wave (i.e., 125 μs). Figure 2.7 illustrates the encoding scheme. We have modified the standard OOK encoding by including four clock train '1' bits at the beginning of the data transmission sequence and also padding the space between each clock-train bit in the transmission with a period of silence. This modification reduces the effect of echo encountered in the underwater environment. An example of a transmitted '1' data bit and the subsequent echo is shown in Figure 2.8. The four clock bits allow the receiving circuit to synchronize with the transmitting circuit, thus preventing the receiving circuit from triggering on an erroneous echo signal. In effect the initial clock train of data provides an initial condition from which the receiving circuit can synchronize the remainder of the transmitted data bits. In this way the transmission and reception circuits are synchronized and the 10-bit structural health data are transmitted from the rotating blade to the static relay station.

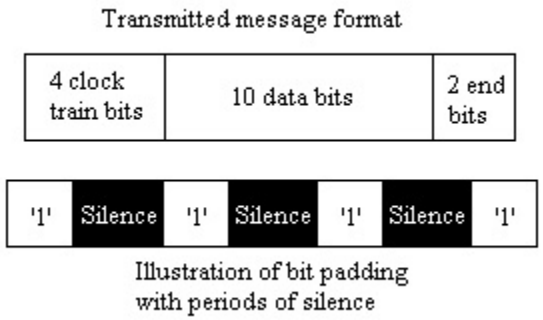

Figure 2.7. Signal encoding scheme

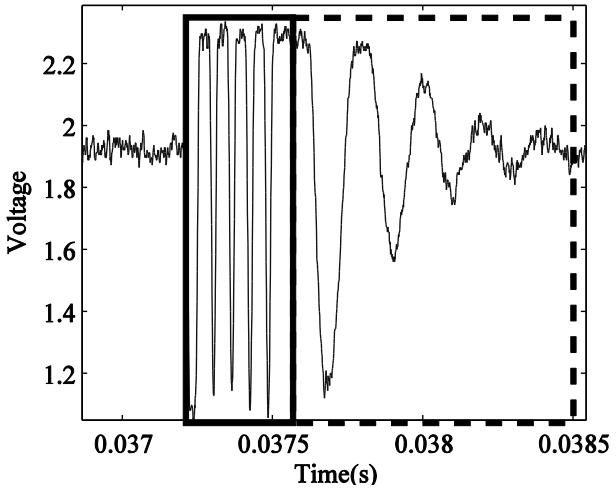

Figure 2.8. Transmitted '1' bit (solid box) and the following echo (dashed box)

The MCU of the receiver circuit listens for and receives transmissions, determines if the transmission is valid, decodes the transmission, and then rebroadcasts the data via Bluetooth. Since the receiving circuit is not in the rotary frame of reference, it can be powered directly from the turbine (or wall during lab testing) and therefore does not spend time sleeping like the transmitter. Instead it spends most of its time listening for an incoming transmission. Upon initial detection of a 40 kHz signal, the MCU determines if

the signal is the 4-bit clock-train indicating initiation of data transmission by listening for the four ‘1’ bits with silence in between (as illustrated in Figure 2.7). The MCU then times the duration between each of the clock bits, and takes the average time of the periods of silence between them. In this way the MCU synchronizes with the transmitter. The next 10 data bits are then received as well as the two stop bits, which serve to indicate that the message has been successfully transmitted. If both stop bits are not received, the MCU rejects the data as not being a valid communication. In the case of correct transmission, the 10 data bits are converted back to a positive integer value and retransmitted via the Bluetooth. Following Bluetooth transmission, the receiver MCU returns to listening for the next communication.

2.2.4. Static Bench Top Testing. The electronics and software were tested to validate correct operation and to characterize the power consumption of the system. For these experiments a static bench top setup was used. For this setup only the acoustic transducers were placed in water, the other electronics are dry, and all components are statically mounted, i.e., there is no rotating turbine blade reference frame, only the lab frame. A photograph of the setup is shown in Figure 2.9. In the figure, a NI USB-6008 DAQ (1) provides a constant 3 V power supply to the circuits. A Sparkfun BlueSMiRF Silver chip (2) is the Bluetooth transmitter connected to the receiver circuit (3). Both the receiver and transmitter circuits (4) are connected to an ultrasonic transducer in the aquatic environment (5). Finally, a Tektronix TDS-2014B oscilloscope (6) was used to measure voltage input and output from the various electric circuit components. To characterize the power of the transmitter circuit, the required current and time duration of the different modes of the MCU were measured. The current was measured with a Fluke

multi-meter in series with the battery and circuit, while the time duration for each mode was measured with the oscilloscope. Finally, isolated operation of the transmitter circuit, by replacing the DAQ power supply with a 3 V coin cell battery, showed identical results for all tests.

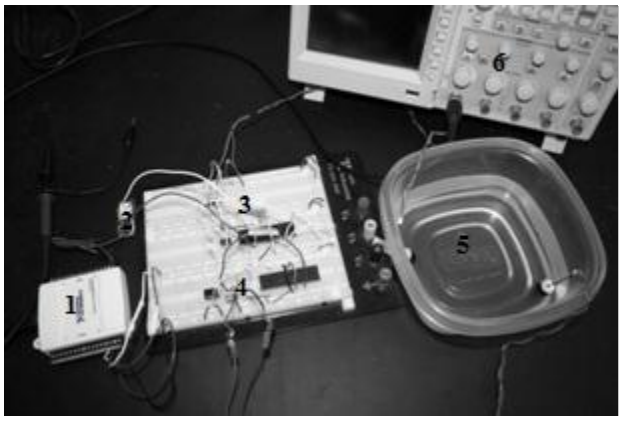

Figure 2.9. Bench top setup for communication validation and power consumption investigation

2.2.5. Dynamic Hydrokinetic Testing Platform. The data transmission system was also tested on a hydrokinetic turbine in a water tunnel in order to better replicate the real world operating environment. A numerical investigation of this machine is presented in Ref. 21. The water tunnel is capable of laminar flow up to 1 m/s (1.9 knots) and has a test section cross-section of 38.1x50.8 cm and length of 152.4 cm. For tests presented here, the water flow speed was 0.25 m/s and the hydrokinetic turbine had a rotational speed of approximately 42 rpm. The hydrokinetic turbine has a shaft diameter of 0.94 cm and is made of aluminum. The turbine has three constant chord blades that are made of aluminum and with a blade width of 1.27 cm and a length of 12.7 cm.

The data transmission system was mounted to the hydrokinetic machine and water tunnel as shown in Figure 2.10. The receiver circuit was mounted on top of the tunnel (dry), with the receiving transducer located in the water and approximately 7.5 cm directly upstream of the transmitter along the centerline axis of the turbine shaft. The receiving transducer was oriented to collect acoustic signals travelling in the upstream direction from the hub. The transmitting circuitry and transducer were attached to the rotating frame of the hydrokinetic machine underwater. The transducer was mounted to the nose of the turbine hub on the centerline axis of the shaft and oriented to transmit data in the upstream direction. The transmitter circuitry was encased in a waterproof container attached to the turbine shaft.

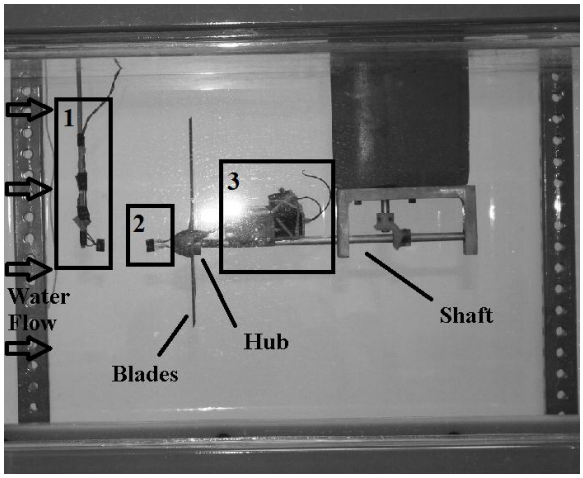

Figure 2.10. Water Tunnel Test Section with receiver transducer (1), transmitting transducer (2), and power and electronics module (3)

3. RESULTS & CONCLUSIONS

The structural health data transmission process described above was tested in a static environment. First we describe the initial validation process that demonstrates accurate transmission of simulated structural health data from the blade to the remote monitoring station. This process is demonstrated for multiple simulated health data signals. Then we analyze the power requirements of the prototype system and investigate the effect of operation mode on transmitter power supply lifetime. Finally, the prototype systems ability to work in a 'real life' environment is validated by mounting the system to a rotating hydrokinetic turbine and then having it transmit data.

3.1. DATA TRANSMISSION VALIDATION

To demonstrate and validate the data communication process, several different health data waveforms were communicated. These different waveforms simulate possible outputs from the embedded fiber optic strain gauge. Initially the error associated with the ADC of the MCU is quantified, along with the precision of the measurement electronics. Then data from the transmitted waveforms is presented, which include a constant value, a square wave, and one half-cycle of a sine wave. For the results presented below, the communication system is tested in a static bench top environment.

The ADC of the MCU has a resolution of 10 bits. This corresponds to 1023 separate voltage values that can be measured in the range from 0 V to the saturation voltage of the ADC, which is either 5 V or the voltage supplied to the MCU. In this case, the MCU supply voltage is 3 V, meaning that the theoretical precision of the ADC is 2.9 mV. To experimentally determine the accuracy of the ADC and associated transmission

circuitry, we applied known voltages to the voltage divider and measured the resulting received signal from the monitoring station. Specifically, voltages from 3.0 to 1.2 V were applied with a resulting experimentally determined error of ±5%. So the 10-bit resolution provides a precision of 2.9 mV with an error in the reported data of ±5%.

Received data for the constant value and square wave waveforms are shown in Figure 3.1. For the constant value data transmission, a voltage of 0.499±0.005 V was input into the transmitter circuit. As Figure 3.1 shows, the remote monitoring station received 50 data points over a time period of 6.2 seconds, all of which have a value of 0.499±0.005 V. For the square wave, the period of the wave was chosen as 10 data samples with a min and max voltage of 0.367±0.005 V and 0.733±0.005 V, respectively. The waveform was connected to the transmitter circuit and the remote monitoring station received the signal shown in Figure 3.1, with all output voltages within ±5% of the input.

Results from the half-cycle of a sine wave data transmission are shown in Figure 3.2, along with the input signal. A slowly varying 0.08 Hz sine wave with an amplitude of 4.70±0.01 V was the input. Over the 6.2 second long half-cycle, the communication system transmitted 50 data points corresponding to discretized points of the sine wave. The received data are within the previously measured ±5% of the measured input voltage.

Finally, we assessed the fraction of transmitted data points that result in an error output from the receiver circuit, and we call this the data error fraction. The system was setup to act as it would in actual operation, transmitting a set of data(simulated in this case), sleeping, then transmitting more data. Over the course of four iterations of this procedure, consisting of 4400 data points, we found that 0.1% of the transmitted data points were in error.

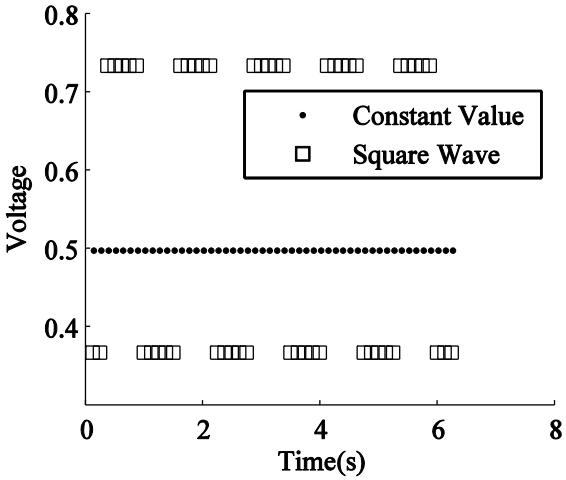

Figure 3.1. Constant and square wave simulated structural health data transmitted to the remote monitoring station

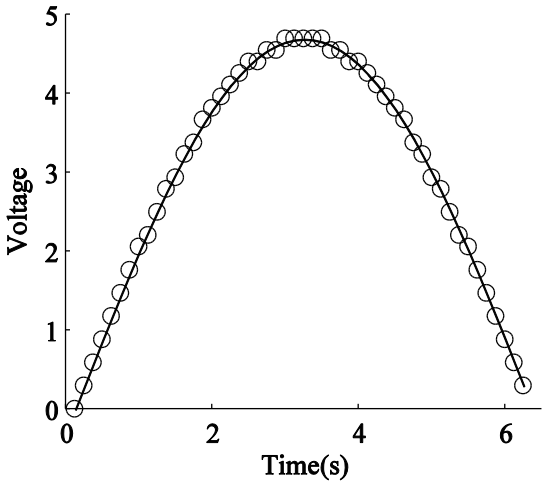

Figure 3.2. Half-cycle sine wave simulated structural health data transmitted to the remote monitoring station

3.2. POWER ANALYSIS

The power requirements of the transmitter circuit are investigated in order to assess and predict the operational lifetime of the communication system. We envision

the transmitter circuit embedded into the hub of the turbine shaft and battery powered within the rotating reference frame of the turbine. While other power options are available, including energy harvesting and a shaft-mounted power take-off, the following analysis assumes the transmitter is operated with a battery of known capacity. The receiver circuit is statically located and can be fed power directly from the generator, so its power consumption is not limited and not investigated here.

Power consumption is calculated using the measured voltage and current output by the power supply of the transmitter circuit. Isolated operation of the transmitter circuit, by replacing the DAQ power supply with a 3 V coin cell battery, showed identical results. Current drawn from the supply was measured for the three possible modes of transmitter circuit operation: sleep, sampling, and transmission. The time the transmitter spends in each of these modes is also measured. These results are given in Table 3.1. The 'X' used in Table 3.1 is the duty cycle, the ratio of time spent "asleep" to time "awake". A plot of the current consumption of the transmitter as a function of duty cycle is shown in Figure 3.3.

Table 3.1. Transmitter current usage by mode (3 V battery supply)

Mode	Time (ms)	Current (mA)	Current*Time(mA*ms)
Sleep	160.1X	0.03	4.80X
ADC sampling	0.1	2.31	0.23
Transmission	160	10.01	1601.6

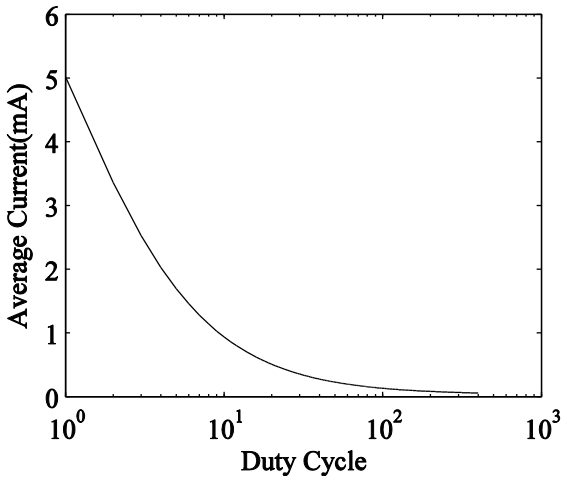

Figure 3.3. Time averaged current draw of the transmitter circuit vs. duty cycle

The maximum operational duration of the transmitter can be calculated using the average current data and known battery capacity. Battery capacity (measured in mA-hr) is divided by the average current, yielding the maximum time in hours that the battery can source the required current. Using the average current results from Figure 3.3, the maximum operational lifetime of the transmitter circuit is plotted in Figure 3.4 for batteries with different capacity. The battery capacities shown in Figure 3.4 are typical of commercial off-the-shelf coin cell batteries. Table 3.2 highlights the duty cycle required to achieve a desired duration of operation, along with the time the transmitter spends sleeping for that duty cycle. For example, to power the transmitter for 1 year using a 560 mA-hr coin cell battery requires a duty cycle of 294, which means data will be acquired and transmitted about every 47 seconds for the entire year.

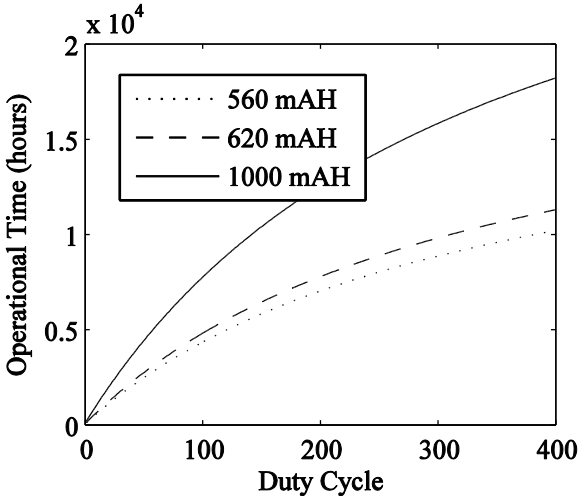

Figure 3.4. Predicted transmitter circuit operating time vs. duty cycle for different battery capacities

Table 3.2. Duty cycle and sleep duration for a desired transmitter circuit operating time for various battery capacities

Operating Time	**Duty Cycle ('Sleep' Time (sec))**		
	560 mA-hr	**620 mA-hr**	**1000 mA-hr**
1 week	3 (0.48)	2 (0.32)	2 (0.32)
2 weeks	6 (0.96)	5 (0.80)	3 (0.48)
1 month	12 (1.92)	11 (1.76)	6 (0.96)
½ year	91 (14.57)	80 (12.81)	45 (7.20)
1 year	294 (47.07)	244 (39.06)	118 (18.89)
2 years	5080 (813)	1851 (296)	368 (58.92)

The ability to remotely monitor the structural health of hydrokinetic turbine blades is of interest for two main applications. First, structural health data can provide service and maintenance personnel with much needed knowledge of the state of the blades, allowing timely repair and servicing before failure. Second, health data can aid environmental assessment studies of prototype machines, providing information on both the impact of the machine on the environment and the environment on the machine. In

both cases, the investigated system will ideally enable faster future commercial deployment of hydrokinetic energy systems.

Results indicate that the data transmission system should be operated in different modes depending on the desired application. Specifically, the transmitter circuit should be operated with a duty cycle suited for the duration of the application. To achieve longer operational duration of the transmitter, a higher duty cycle is required. We anticipate that commercially deployed hydrokinetic machines will require long operational duration and thus high duty cycle, while shorter-term environment assessment and prototype studies will benefit from lower duty cycle operation. For example, environmental assessment testing of a prototype machine is a relatively short timeline on the order of days or weeks compared with a commercially deployed device that will be in water for years. Fast detection of changes in structural health are typically required for environment assessment (i.e., immediate detection of an impact), while more intermittent assessment of structural health can be tolerated with long-term deployed machines. Therefore, for short duration environmental assessment continuous max sampling should be used, providing a sample rate of 500 samples/sec. A 1000 mA-hr battery would last at most around 4 days at this rate. For a two-year deployment of a commercial system with a 1000 mA-hr battery, the transmission system should be operated with at least a duty cycle of 368 to provide a data sample about every 59 seconds.

3.3. TESTING ON A DYNAMIC HYRDROKINETIC ENERGY PLATFORM

The data transmission system was setup and operated on a dynamic hydrokinetic

energy platform as described above. Transmitter circuitry and transducer were attached to the rotating reference frame of the blade, while the receiver was positioned directly upstream in the water. Received data were transmitted wirelessly via Bluetooth to the remote monitoring station. The transmitter circuit was pre-programmed to output the constant, square wave, and sinusoid data, exactly the same as shown in Figure 3.1 and Figure 3.2.

Data received by the remote monitoring station during dynamic testing on the hydrokinetic machine appeared similar to that shown in Figure 3.1 and Figure 3.2. This result indicates that the data transmission system can be operated in a relevant environment and within the rotating reference frame of the hydrokinetic blades. However, initially we find that more of the transmitted data results in an error from the receiver circuitry. While the benchtop setup had a data error fraction of 0.1%, we find that when testing on the hydrokinetic machine about 2% of the data points result in an error output from the receiver.

The increase in data error fraction during the initial test is due to the larger aquatic environment and the resulting increase in expansion and attenuation of the acoustic signal. We modified the setup shown in Figure 2.10 by placing a 7.62-cm-diameter by 11.43-cm-long cylinder between the transmitting and receiving transducers. The cylinder was attached to the receiver and was therefore statically mounted (not rotating). The cylinder reduces the expansion and attenuation of the acoustic signal by guiding the acoustic waves from the transmitter to the receiver. With this modified setup, we find that only 0.9% of the data result in an error output.

4. CONCLUSIONS

The data transmission system can successfully communicate representative structural health data in a benchtop setting and also from the rotating reference frame of a prototype hydrokinetic energy system. Data can be transmitted acoustically underwater from an isolated source (transmitter, blade reference frame) to a static relay station (receiver, fixed reference frame), and then broadcast wirelessly to a remote monitoring station (via Bluetooth). Error in the electronics measuring the structural health signal limit the accuracy of the measurements to ±5%. Results from testing on a prototype hydrokinetic machine indicate that the ultrasonic and Bluetooth communication can transmit 10 bit data with an error fraction of only 0.9% when the acoustic signal is properly guided from the transmitter to receiver. If the acoustic signal is allowed to expand using the current acoustic transducers and signal power level, the data error fraction increases to 2%.

The maximum rate at which the structural health data can be transmitted is 500 samples/sec. Each sample of structural health data is captured by the measurement electronics with 10 bit resolution, and the entire 10 bit data packet is transmitted. With our modified on-off keying encoding scheme, the underwater acoustic communication system must transmit a total of 16 bits: 10 data bits, 4 clock train bits, and 2 termination bits. With the 40 kHz acoustic frequency, each bit requires 0.125 ms, resulting in a total transmission time for 1 data sample of 2 ms. Thus, assuming continuous data transmission, a maximum rate of 500 samples per second can be achieved by the data transmission system.

The optimum mode of operation of the data transmission system is dependent on the hydrokinetic machine application. We examined application of the data transmission system to short term environmental impact testing of prototype hydrokinetic machines and long term commercially-deployed machines. For prototype testing, the desired goal of the data transmission system is to precisely record in real time the structural health and blade loading experienced under various conditions. This could include detailed strain response of the system over time, the response to impact or other transient events, and monitoring the evolution of the degradation and damage the blades experience. For prototype testing the data transmission system should operate at maximum sampling rate. Based on our analysis, max sampling rate could be achieved for 50+ hours of testing with any of the battery power sources described.

A commercially-deployed hydrokinetic system requires a longer lifetime than a prototype system. In this case, the data transmission system should make periodic structural health measurements, transmit the data, and then enter sleep mode to conserve power. While this mode of operation does not allow for "real-time" transient events to be monitored, it does extend the life of the monitoring system while still detecting deterioration and damage to the hydrokinetic blade. Based on our results, a data transmission system lifetime of two years can be achieved by increasing the duty cycle (sleep time) of the system. Using a relatively low capacity 560 mA-hr battery, a duty cycle of 5080 would provide operation for two years, and would measure the structural health data every 13 minutes 33 seconds throughout the 2 year period. If higher sampling rate is desired, a 1000 mA-hr battery could be used at a duty cycle of 368, which would acquire and transmit data every 59 seconds. Based on the capacity of the battery and

duty cycle (sleep time) of the system, intermittent monitoring can be achieved for relatively long periods of time.

5. FUTURE WORK

Further improvements can of course be made to the prototype system as has been outlined above. The main points where improvements would be desired are the reliability of the communication and the duration that the system can continue to operate. The reliability could be improved by either using a more reliable data-encoding scheme or by increasing the signal to noise ratio (SNR) of the transmitted data. Increasing the SNR of the transmitted signal could be accomplished with better signal conditioning and noise filtering or by increasing the signal amplitude, but increasing the signal amplitude would in turn require more power for transmission. Increasing the duration of the systems operation would require either more power to be available, or a reduction in the amount of power drawn by the transmitter circuit. Reducing the power used by the transmitter circuit would entail choosing electrical parts for the circuit that would consume the least amount of power possible, as opposed to parts that allowed for ease of prototyping as well as low power consumption. Some additional software optimizations could also potentially be made to decrease the amount of power consumed by the transmitter. Increasing the available amount of power could be accomplished in a number of different ways. The simplest of them would be to either use heavier duty coin cell batteries, or to simply hook in additional batteries to the system. This would increase the systems operating lifetime, but would still lead to the batteries needing to be replaced eventually. To make it so that the system would not need replacement batteries, a generator of some sort would need to be included. This could entail various energy scavenging methodologies or the design and inclusion of a secondary mini turbine to power the

transmitter circuit. Either option could allow for the systems operational life to be extended indefinitely, but further work would need to be done to determine which would be the more practical option. In short, future work on the system will involve increasing the reliability of communications, increasing the duration of the systems operation, and the reduction of both the system's size and power consumption.

APPENDIX A.

SYSTEM SOFTWARE

This appendix will document the software used to program the MCU's for both the receiving and transmitting circuits. This will include the programs used to do the bench top and dynamic testing, as well as laying out and explaining the code necessary for individual portions of the systems operation. It should also be noted that more detail on the how's and whys of programming the MCU can be found by looking at its datasheet, which can be found by searching the Microchip website for pic18f46k20. Four files in total encompass the software used for the system, an executable file and a header file for each MCU.

A.1. ADC

Two important tasks for the transmitter circuit are the measurement of the voltage from the strain sensor and the monitoring of the battery supplies voltage level. In order to do this, the ADC feature must be setup and executed. Setting up the ADC requires configuring one or more pins of the MCU to act as an ADC. For the MCU being used for this system, configuring pins to act as an ADC requires correctly setting the value of the three control registers ADCON0, ADCON1, and ADCON2 as well as setting the given pin as an input pin. The control registers here are used to control which pin is selected as an ADC, when to start the ADC conversion, whether or not the ADC functionality is turned on, and the details of the conversion process, which is dependent to an extent on the clock speed currently selected for the MCU. Shown below is a snippet of code that configures pin RA0 as an ADC and the function used in the program to select and read an ADC pin.

```
TRISAbits.RA0 = 1;
ADCON0 = 0b00000000; //Channel 0 selected, ADON and GO off
ADCON1 = 0b00000000; //Sets Vrefs to Vdd and Vss
ADCON2 = 0b00101001; //12 TAD aqct, FOSC/8 for TAD


//ADC
//Input: unsigned char x (for ADC channel selection)
//Output: The value read from channel x
//Reads an ADC setup pin selected with variable x. Reads the value
// on that pin, and returns it.
int ADC(unsigned short x)
{
        int y=0;

        if(x == 0)//RA0 being used
                ADCON0 = 0b00000000;
        if(x == 1)//RA1 being used
                ADCON0 = 0b00000100;
        if(x == 2)//FVR being measured
                ADCON0 = 0b00111100;

//ADC on, starting conversion
        ADCON0bits.ADON = 1;
        ADCON0bits.GO = 1;

//Waiting till conversion completed
        while(ADCON0bits.GO);

//Conversion complete, reading value
        y = 0;
        y = ADRESH;
        y <<= 2;
        y += (ADRESL>>6);

//Turning off ADC,returning read value
        ADCON0bits.ADON = 0;
        return y;
}
```

It can be noted that the function shown returns an integer value. This value will be a value of 0 to 1023 and must be converted to get the actual voltage received. For the transmitter circuit, there are two separate voltages to be read, the battery voltage and the strain sensor voltage. The strain sensor voltage is 0 to 10 V, but is reduced to 0 to 2 V by

a voltage divider, a factor of 5 reduction. As such, the equation to convert the ADC reported value to voltage is given by equation A.1, as shown below.

$$V_{in} = 5 * (V_{sup} * ADC / 1024) \quad (A.1)$$

As can be seen by equation A.1, the supplied voltage level affects the ADC measurements, so that the supplied voltage must be monitored to take accurate strain measurements and in order to determine when a new battery is needed. Monitoring of the batteries voltage level is accomplished by using the Fixed Voltage Reference (FVR) feature of the ADC, which reports a constant 1.2 V when monitored. As the supplied voltage slowly drops as the battery is drained, the FVRs reported value will increase. For example, when a fresh battery is in place, supplying 3 V, the FVR will have a reported value of 410. Later on though, as the supplied voltage drops to 2.8 V, the FVR will have a reported value 439. This is due to the fact that the maximum ADC reading of 1023 corresponds to the saturation voltage of the pins, which in turn is the supplied voltage level. Thus, as the supplied voltage level drops, the 1.2 V signal from the FVR gets closer to the saturation voltage and therefore reads as being higher. Therefore, getting the current battery voltage level is accomplished by the equation A.2.

$$V_{sup} = 1.2 / (ADC /1024) \quad (A.2)$$

A.2. ACOUSTIC TRANSMISSION

Having measured some voltage data, the data must then be transmitted acoustically to the receiving circuit. The transmitting methodology having been explained previously shall not be explained again here. Shown below is the code for the formatting the data into the appropriate messaging scheme, as well as the code used to actually broadcast the individual '0' and '1' bits of the transmission.

```
//Transmit
//Input: 10 bit integer of data
//Returns: Nothing
//Takes ADC data, converts it to 16 bit binary string, and then goes ahead
//and outputs each bit for 5 wavelengths of 40 kHz.
void Transmit(int data)
{
        int c = 0,d = 0,e = 0;
        int delay = 100;
        int message[16];

//Making message packet for transmission
//16 bit transmission.

//First 4 bits are headers, value 1
        message[0]=message[1]=message[2]=message[3] = 1;

//Next 10 are data,
        for(c = 0;c < 10;c++)
        {
          if ((data & 0b1000000000))//Added 2 0s for 10 bit. hope it works
                message[c+4] =  1;
    else
           message[c+4] = 0;

         data <<= 1;
        }

//Last two bits are stoppers
        message[14] = message[15] = 1;
```

```
// Transmitting 5 periods of 40kHz signal per bit
//Straight 50% duty square wave seems to do the best
//Delay between bits to eliminate echoes
        for(c = 0;c < 16;c++)
        {
                if(message[c] == 1)
                        transOne();
                else
                        transZero();

                Delay100TCYx(delay);
        }
}
```

Shown below is only the function used to transmit a '1' bit, since the function used to transmit a '0' bit, the *transZero* function referenced in the *Transmit* function above, is essentially exactly the same, with the exception that both the pin assignments are set to 0, as opposed to the first being set to 1 and the second being set to 0 as shown below.

```
void transOne()
{
        int d= 4;
        PORTCbits.RC0 = 1;
        Delay10TCYx(d);
        PORTCbits.RC0 = 0;
        Delay10TCYx(d);

        PORTCbits.RC0 = 1;
        Delay10TCYx(d);
        PORTCbits.RC0 = 0;
        Delay10TCYx(d);

        PORTCbits.RC0 = 1;
        Delay10TCYx(d);
        PORTCbits.RC0 = 0;
        Delay10TCYx(d);

        PORTCbits.RC0 = 1;
        Delay10TCYx(d);
        PORTCbits.RC0 = 0;
```

```
        Delay10TCYx(d);

        PORTCbits.RC0 = 1;
        Delay10TCYx(d);
        PORTCbits.RC0 = 0;
        Delay10TCYx(d);
}
```

A.3. SLEEP MODE

Having measured and transmitted the relevant data, the transmitter then enters a low power sleep mode till data needs to be sampled and transmitted again in order to conserve power. The use of this function involves the use of the MCU's Watchdog timer (WDT) functionality. For the communications system, most of the default values for the bits needed to configure the WDT were already at their desired value, and so no changes were made. The WDT's job is to reset the MCU if a certain period of time has passed and nothing has happened to turn off or reset the WDT's counter. This feature is intended to allow for the program to interrupt itself should it get stuck in a loop, or as we use it, to wake the MCU from a low power mode. The period of time the WDT waits before interrupting the program is 4 ms times a 16 bit postscaler that can be set by the programmer. The default value for the post-scaler is 32768, which when multiplied by 4 ms gives an approximately 2 minute and 11 seconds waiting time. The only configuration bit that needed to be adjusted in order to use the WDT as desired was the OSCCON IDLEN bit, which controls which low power mode the MCU enters on the SLEEP command being entered. Setting this bit to 1 set the WDT to the appropriate mode, and entering the SLEEP command automatically activates and clears the WDT's counter. The code below was used to send the MCU to sleep for arbitrarily long periods of time.

```
/Sleepytime
//Inputs: int doDo
//Returns: A 0 upon successful completion
// Recursively loops, sleeps for about 2 minutes, substracts 1 from
// toDo, and if toDo == 0, exits. If toDo not 0, function calls itself.
// In short, each cyle sends PIC to sleep for 2 min, and toDo is the
// number of cycles to sleep.
int sleepyTime(unsigned int toDo)
{
	_asm
	SLEEP
	_endasm

	ticks++;
	toDo--;
	if(toDo == 0)
		{
			//Setting WDT flag to prevent triggering
			RCONbits.TO = 1;
			//Turning off watchdog timer
			WDTCONbits.SWDTEN = 0;
			return 0;
		}
	else
		sleepyTime(toDo);
}
```

It should be noted in the function above that the WDT is cleared and turned off after the specified sleep time has been completed in order to ensure that the WDT does not interrupt the operation of the MCU when not desired. A final note on the above function is that whilst the code thus far has been written in a variation of C written for PIC microchips (C18 to be specific), the SLEEP command is an assembly command, and the *_asm* and *_endasm* tags are notes to the compiler that the code within the tags is written in assembly and not C.

A.4. ACOUSTIC RECEPTION

Having transmitted the measured data and then gone to sleep, the receiving circuit must pick up and decode the transmitted message. As mentioned previously, a total of 16 bits are in each transmitted message with the first four bits being the clock train bits. The method used to synchronize the data transmission with the data reception is accomplished by picking up each of these initial 4 '1' bits, and timing how long it is before the next clock train bit is received. Using the average of these times, the receiver can then confidently time how long it needs to wait before looking for each subsequent data bit. This procedure was decided upon to allow for adjustment of the delay period between the bits without having to manually reconfigure the code each time that the delay was changed. It also allows for unexpected changes in the acoustic environment that could alter the transit times to be accounted for. The reception code is shown below.

```
/Recieve
//Inputs: None
//Returns: Data upon successful completion, -1 upon an error, and negative data if the
 // transmission might be in error
// Waits for the first bit of a transmission to be picked up. Upon reception, then times how
//long between each of the next three bits in the clock train. This total time is averaged
//and used to determine when to check for the next 10 data bits and the two end bits.
int Recieve()
{
        //Transmitter delay 100 at 8 mHz, currently reciever delay is 150 at 16 mHz
        //So, should start listening roughly 90 cycles in to the tranmitter delay
        int c = 0,data = 0,y = 0,bad = 0,del = 180,total = 0;
        int message[16];

        unsigned int upper = 0, lower = 0,under = 0;

        while(c < 4)
        {
                if(c == 0)
                {
```

```
                if(PORTCbits.RC0)
                {
                        message[c] = getBit();
                        if(message[c] == 1)
                        {
                                c++;
                                Delay100TCYx(del);
                        }
                }
        }
        else
        {
                while(!PORTCbits.RC0 && y >= 0)
                        {
                                y++;
                        }

                if(y >= 0)
                {
                        message[c] = getBit();
                        c++;
                        Delay100TCYx(del);
                }
                else
                {
                        y = 0;
                        c = 0;
                }
        }
}
y = (y-1)/3;

while(c < 16)
{
        data = 0;
        if(c == 12)
        {
                y = y+1;
        }
        while(data != y)
        {
                Nop();
                data++;
        }
        message[c] = getBit();
```

```
            c++;
            Delay100TCYx(del);
        }

        c = 0;
        data = 0;

        //Checking that headers and stoppers were recieved, message
        // probably okay then.
        if(message[14] + message[15] >= 1)
        {
            if(message[4])
                data += 512;
            if(message[5])
                data += 256;
            if(message[6])
                data += 128;
            if(message[7])
                data += 64;
            if(message[8])
                data += 32;
            if(message[9])
                data += 16;
            if(message[10])
                data += 8;
            if(message[11])
                data += 4;
            if(message[12])
                data += 2;
            if(message[13])
                data += 1;

            if(message[14] + message[15] == 1)
                return -data;

            return data;
        }

        return -1;
}
```

It should be noted that some small amount of manual adjustment may still be necessary to get the timing spot on. In the function above, this adjustment is in the form

of the line *y = (y-1)/3*, where y is the total time counted for the clock bits and 1/3 is essentially subtracted off the average to allow for better timing. Another point to note is the *getBit* function, which checks to see if a '1' bit has been received at a given time. This code is shown below.

```
//getBit
//Inputs: None
//Returns: A 1 if a bit was picked up, and a 0 if not
// Checks 5 times to determine if there is a digital high voltage value on the pin being
// polled. Waits a bit between each check so that the 5 checks are spread over the time
// that five 40 kHz periods takes up. If more than three of the checks had high voltage on
// them, then it is presumed that a '1' bit was received and 1 is returned. If not, 0
// returned.
int getBit()
{
	int i = 0,x = 0;
	for(i = 0;i < 5;i++)
		{
			PORTBbits.RB1 = 1;
			if(PORTCbits.RC0 == 1) //Check
			{
				x++;
			}
			TMR0L = 0b11100110;//40 kHz for 16 mHz

			T0CONbits.TMR0ON = 1;
			while(!INTCONbits.TMR0IF);//Delay till next check
			INTCONbits.TMR0IF = 0;	//Reset flag
			T0CONbits.TMR0ON = 0;

			PORTBbits.RB1 = 0;
		}
	PORTBbits.RB1 = 1;
	if(PORTCbits.RC0 == 1) //Check
		x++;
	PORTBbits.RB1 = 0;

	if(x >= 3)
		return 1;
	return 0;
}
```

A.5. BLUETOOTH COMMUNICATIONS

Having received data, the receiver circuit then needs to rebroadcast that data on. For this prototype system, that data is rebroadcast using Bluetooth, which the MCU interfaces with using its inbuilt EUSART functionality, which is a serial I/0 device. In order for the MCU to properly communicate with the Bluetooth chip, or any other serial device, the EUSART module must be turned on, and be properly configured to communicate with the other device. This configuration involves both the MCU and the device connected to it to have the same baud rate, and for the messages sent to have the same number of bits. For the Bluetooth connection, the baud rate was set to 115200 kbs and there were 8 bits in each message packet sent. In order to set this up, first both pins that corresponded to the transmission and reception pins of the EUSART line were set to being inputs. Then the baud rate, transmission, and reception settings were entered. This is shown below.

```
/Setting up uart for Bluetooth
//Setting TRIS bits of Tx and Rx pins
TRISCbits.RC6 = 1;
TRISCbits.RC7 = 1;

//Setting up baud rate 115200
BAUDCONbits.BRG16 = 1;
TXSTAbits.BRGH = 1;
//For 16 MHz running
SPBRGH:SPBRG = 34;

//Transmission Settings
TXSTAbits.TX9 = 0; //8 bit mode activated
TXSTAbits.SYNC = 0;//Asynchronous mode active

//Reciever Settings
RCSTAbits.SPEN = 1;//Serial mode enabled
RCSTAbits.RX9 = 0;//8 bit mode enabled
```

```
RCSTAbits.FERR = 0;//Clear this error bit for safekeeping
```

As can be seen, most complicated portion of the setup is in the setting the baud rate. The baud rate is determined by the clock speed of the MCU and the value assigned to the configuration register pair SPBRGH:SPBRG. The method of determining the value needed to assign to the control register pair is clearly outlined in the PIC datasheet and shall not be repeated here, as it is somewhat involved. Having setup the serial connection with the Bluetooth device, data must now be written to it. The function used to do this is shown below.

```
//writeBluetooth
//Inputs: A character
//Returns: Nothing
// Takes in a character to be transmitted and enables transmission over the serial line.
//Makes sure that nothing is still being transmitted, then once the transmission buffer is
//clear adds the received character to the buffer for transmission. Having now gotten the
//transmission process underway, further transmission is then disabled.
void writeBluetooth(char data)
{
	TXSTAbits.TXEN = 1;//Enable transmission

//wait till register clear for transmission, TXIF = 0 when still full

	//while(!PIR1bits.TXIF)
	while(TXSTAbits.TRMT == 0);

//Putting data into TXREG for transmission
	TXREG = data;

	//while(TXSTAbits.TRMT == 0);
	TXSTAbits.TXEN = 0;//No longer transmitting
}
```

A.6. CODE FOR BENCH TOP AND DYNAMIC TESTING

The functions needed for the transmission, reception, and wireless transmission of data have been shown. To that end, shown below are the main files for both the transmitter and receiver MCU's to illustrate how these functions were used in communicating the simulated waveforms used in the transmission testing.

A.6.1. Receiver Code

```
#include <p18f46k20.h>
#include <commFuncs.h>
#include <stdio.h>

//#pragma config FOSC = INTIO67
#pragma code IntVectHigh = 0x08
#pragma interrupt IntHigh

void main()
{
	int x = 0,k = 0,c = 0,d = 0,e = 0,data = 0;
	int done = 0;
	char test='a',A = 'a';
	char array[10];		char message[50];
	float t1 = 0;

	TRISB = 0;
	TRISC = 0;
	PORTC = 0;
	PORTB = 0;

	OSCCONbits.IRCF = 0b111;//Clock set for 16 MHz

	setup();

	TRISCbits.RC0 = 1;
	TRISBbits.RB0 = 1;

	TRISBbits.RB3 = 0;
	PORTBbits.RB3 = 1;

	for(;;)
	{
```

```
                if(d == 0)
                {
                x = Recieve();

                c=sprintf(array,"%d",x);
                for(k = 0;k < c;k++)
                        message[e+k] = array[k];
                e = e+c;
                message[e] = ' ';
                e++;

                if(e+5 >= 50) //10 >= 50)
                {
                        for(k = 0;k < e;k++)
                        {
                                writeBluetooth(message[k]);
                        }
                        e = 0;
                }

                }
        }
}
```

A.6.2. Transmitter Code

```
#include <p18f46k20.h>
#include <commFuncsR2.h>
#include <stdio.h>

int sinArray[50] =
{0,20,40,60,80,100,120,140,150,170,190,200,220,230,250,260,270,280,290,300,
300,310,310,320,320,320,320,320,310,310,300,300,290,280,270,260,250,230,220,
                                        200,190,170,150,140,120,100,80,60,40,20};

//#pragma config FOSC = INTIO67

void main()
{
        int x = 0,k = 0,c = 0,d = 0,e = 0;
        int done = 0;

        TRISB = 0;
        TRISC = 0;
        PORTC = 0;
        PORTB = 0;
```

```
OSCCONbits.IDLEN = 0;
OSCCONbits.IRCF = 0b110;//Clock set for 8 MHz

setup();

TRISCbits.RC0 = 0;
TRISAbits.RA4 = 1;

PORTBbits.RB3 = 1;
PORTCbits.RC3 = 1;

sleepyTime(1);
done = 1;

for(;;)
        {
                RCONbits.TO = 1;//Setting WDT flag to prevent triggering
                WDTCONbits.SWDTEN = 0;//Turning off watchdog timer

        if(done == 1)
        {

                Transmit(34);
                d++;
                Delay100TCYx(200);
                if(d == 100)
                {
                        done = 2;
                        d = 0;
                        for(c = 0;c < 50;c++)
                        {
                                Delay100TCYx(500);
                                Delay100TCYx(500);
                                Delay100TCYx(500);
                                Delay100TCYx(500);
                        }
                }
        }

        if(done == 2)
        {
                if(e < 5)
                        Transmit(25);
                else
                        Transmit(50);
                e++;
```

```
                    if(e >= 10)
                        e = 0;
                    d++;
                    Delay100TCYx(200);
                    if(d == 100)
                    {
                        done = 3;
                        d = 0;
                        for(c = 0;c < 50;c++)
                        {
                            Delay100TCYx(500);
                            Delay100TCYx(500);
                            Delay100TCYx(500);
                            Delay100TCYx(500);
                        }
                    }
                }

                if(done == 3)
                {

                    Transmit(sinArray[x]);
                    x++;
                    if(x >= 50)
                        x = 0;
                    d++;
                    Delay100TCYx(200);
                    if(d == 100)
                    {
                        done = 1;
                        d = 0;
                        for(c = 0;c < 50;c++)
                        {
                            Delay100TCYx(500);
                            Delay100TCYx(500);
                            Delay100TCYx(500);
                            Delay100TCYx(500);
                        }
                    }
                }
            }
        }
```

A.7. MISCELLANEOUS PROGRAMMING ISSUES

A few random bits of code should also be noted. Amongst these are the inclusion of the proper header files for the MCU, the setting of configuration bits, and the altering of the MCU clock speed.

In order to ensure that the compiler knows how to properly interpret an MCU's various features, pins, and configuration bit names, the header file for that MCU must be included. The programming environment used for writing the MCU code comes with a number of default PIC MCU header files. If however, a chip is being used who's header file is not included, an internet search will need to be conducted to find and then download that header file. The inclusion of the header file is as expected of a C based program, and the header code used for the systems MCUs are shown below.

```
#include <p18f46k20.h>
```

In order to ensure the certain features of the PIC are either on or off, sometimes certain parameters, referred to as configuration bits, must be set as the PIC starts up. After this, these parameters stay the same until changed upon another startup procedure, they cannot be changed during runtime. For this system, the only configuration bit we were concerned with was the one dealing with the selection of the clock source. The MCUs used could either generate a clock source internally, or receive one externally. To ensure the MCU works properly, the clock source needed to be set to being internally generated, which was accomplished with the following command.

#pragma config FOSC = INTIO67

Finally, the clock speed may be desired to be altered for various reasons. The configuration register that controls this clock speed, from the internally generated clock source, is the OSCCON register. This register controls the selection of the clock speed, which type of low powered mode is entered upon the SLEEP command being entered, and configures what type of clock source is being used. To change the clock speed, the IRCF bits must be adjusted. Shown below is an example that shows the clock speed being adjusted to 8 MHz.

OSCCONbits.IRCF = 0b110; //Clock set for 8 MHz

APPENDIX B.

PROGRAMMING MICROCONTROLLERS

In general, programming MCUs requires special hardware and sometimes special software. This is due to the need to hook a MCU up to a computer to program it, and the need for very specific conditions to be applied to the MCU to accomplish this programming. These conditions, and the hardware necessary to produce them, tend to be specific to the company that manufactures the MCU. The MCUs used were manufactured by Microchip, and as such a programmer from Microchip was used to facilitate the programming of the MCUs. These programmers, for various MCUs from various companies, can be bought whole or in kits, or can be put together from various programmer circuit designs that can be found on the internet. In this particular case, the programmer used was the PICDEM 2 Demonstration board. This board is shown below in Figure B.1.

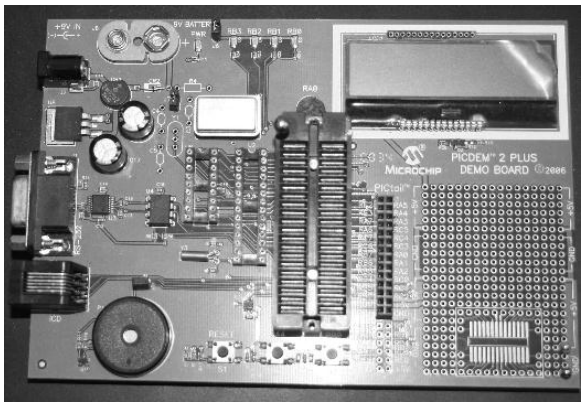

Figure B.1. PICDEM 2 Demonstration Board

This board provides the necessary connections to hold, program, and to a certain extent, test the programming of an MCU with various built in features such as buttons, LEDs, and a buzzer. To hook this board up to the computer, an In Circuit Debugger

(ICD) unit was used to interface between the board and the computer, as shown in Figure B.2.

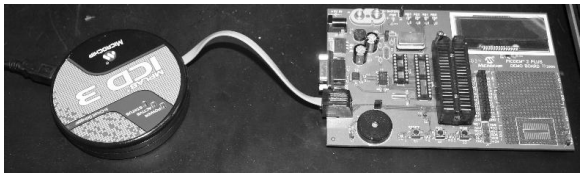

Figure B.2. ICD hookup

This particular ICD was made to interface with a software environment called MPLAB that is used to write, debug, compile, and finally physically program the code to be used on a MCU. This software environment can be downloaded for free from the Microchip website. The MPLAB software does not include the capability to program in the C variant C18 by default, this needs to be downloaded and installed. This too can be found for free on the Microchip website, and there are clear directions on how to go about installing the C18 compiler. Once the actual software and hardware has been setup and a program has been written, the actual programming of the MCU is quite simple. With the ICD connecting the computer and the demo board, one merely needs to hit the program button in the MPLAB toolbar. Upon completion, the demo board can be unplugged, the chip removed, and then used as is desired.

BIBLIOGRAPHY

[1] Khan, M. J., Bhuyan, G., Iqbal, M. T., Quaicoe, J. E., "Hydrokinetic energy conversion systems and assessment of horizontal and vertical axis turbines for river and tidal applications: A technology status review," *Applied Energy*, Vol. 86, pp. 1823-1835, 2009.

[2] Khan, M. J., Iqbal, M. T., Quaicoe, J. E., "River current energy conversion systems: Progress, prospects and challenges," *Renewable and Sustainable Energy Reviews*, Vol. 12, pp. 2177-2193, 2008.

[3] Haas, K. A., Fritz, H. M., French, S. P., Smith, B. T., Neary, V., "Assessment of Energy Production Potential from Tidal Streams in the United States," Final Project Report, DE-FG36-08GO18174, Georgia Tech Research Corporation, June 29, 2011.

[4] Scott, G., "Mapping and Assessment of the United States Ocean Wave Energy Resource," Final Report, 1024637, Electric Power Research Institute, Dec. 2011.

[5] Asmus, P., Wheelock, C., "Hydrokinetic and Ocean Energy: Renewable Power Generation from Ocean Wave, Tidal Stream, River Hydrokinetic, Ocean Current, and Ocean Thermal Technologies," Market Research Report, Pike Research, 2009.

[6] Previsic, M., "System Level Design, Performance, Cost and Economic Assessment - Alaska River In-Stream Power Plants," EPRI RP 006 Alaska Electric Power Research Institute, Oct. 31, 2008.

[7] "Hydrokinetic Electric Power Generation," Climate TechBook, Pew Center on Global Climate Change, Dec. 2009.

[8] Walford, C. A., "Wind Turbine Reliability: Understanding and Minimizing Wind Turbine Operation and Maintenance Costs," Sandia Report SAND2006-1100, Prepared by Global Energy Concepts, LLC, Mar. 2006.

[9] "Report to Congress on the Potential Environmental Effects of Marine and Hydrokinetic Energy Technologies," U.S. Dept. of Energy, Dec. 2009.

[10] Sale, M. J., et al., "DOE Hydropower Program Biennial Report for FY 2005-2006," ORNL/TM-2006/97, U.S. Dept. of Energy, July 2006.

[11] Cada, G., et al., "Potential Impacts of Hydrokinetic and Wave energy Conversion Technologies on Aquatic environments," *Fisheries*, Vol. 32, No. 4, pp. 174-181, April 2007.

[12] Coutant, C. C., Cada., G. F., "What's the future of instream hydro?," *Hydro Review XXIV*, Vol. 6, pp. 42-49, 2005.

[13] Ciang, C. C., Lee, J.-R., Bang, H.-J., "Structural health monitoring for a wind turbine system: a review of damage detection methods," *Measurement Science and Technology*, Vol. 19, pp. 122001, 2008.

[14] Dutton, A. G., et al., "Acoustic Emission Condition Monitoring of Wind Turbine Rotor Blades: Laboratory Certification Testing to Large Scale In-Service Deployment," *European Wind Energy Conference - EWEC*, Madrid, Spain, 2003.

[15] M.J.Blanch, A.G.Dutton, "Acoustic Emission Monitoring of Field Tests of an operating Wind Turbine," *Key Engineering Materials*, Vol. 245-246, pp. 475-482, July 2003.

[16] Akyildiz, I. F., Pompili, D., Melodia, T., "Underwater acoustic sensor networks: research challenges," *Ad Hoc Networks*, Vol. 3, pp. 257-279, 2005.

[17] Robison, K., Watkins, S. E., Nicholas, J., Chandrashekhara, K., Rovey, J. L., "Instrumented Composite Turbine Blade for Health Monitoring," *SPIE Smart Structures and Materials, Non-destructive Evaluation, and Health Monitoring Conference*, San Diego, CA, Mar. 11-15, 2012.

[18] H. C. Pumphrey, L. A. Crum and L. Bjorno, "Underwater sound produced by individual drop impacts and rainfall," *JASA*, vol 4, no. 4, April 1989.

[19] R. B. Mitsen, "Underwater Noise of Research Vessels: Review and Recommendations," International Council for the Exploration of the Sea, Copenhagen, Denmark, May 1995.

[20] M. A. Ainslie, J. G. McColm, "A simplified formula for viscous and chemical absorption in seawater," *J. Acoust. Soc. Am.*,vol 103, no. 3, March 1998.

[21] S. Mukherji, N. Kolekar, A. Banerjee, R. Mishra, "Numerical investigation and evaluation of optimum hydrodynamic performance of a horizontal axis hydrokinetic turbine," *J. Renewable Sustainable Energy* 3, 063105 (2011).

[22] Chuan Li, D. A. Hutchins, R. J. Green, "Short-Range Ultrasonic Digital Communications in Air," \ *IEEE Trans. Ultrason., Ferroelect., Freq. Contr.*, vol 55, no. 4, April 2008.

VITA

Andrew Heckman was born to Joseph and Elizabeth Heckman in 1987. Graduated from Hazelwood West High School in 2006 and was accepted to the then University of Missouri Rolla. He then graduated with a Bachelor's degree in Mechanical Engineering and a minor in mathematics 2010, whereupon he then continued on as a master's level graduate student.