Design of a Radio Transceiver for use in Underwater Autonomous Robotic Fish

Patrick K. Bettale (Hall)
### TABLE OF CONTENTS

Abstract ........................................................................................................... 03  
Introduction .................................................................................................. 03  
Project Discussion ......................................................................................... 04  
  - Requirements and Specifications ................................................................. 04  
    - Printed Circuit Board Specifications ......................................................... 05  
    - Hardware Specifications ........................................................................... 05  
    - Software Specifications ........................................................................... 06  
    - Range and Reliability Specifications ......................................................... 06  
Design Procedure ............................................................................................ 06  
  - Component Selection .................................................................................... 07  
  - Board Layout ................................................................................................ 08  
  - Antenna Design ............................................................................................ 10  
  - RF Modulation .............................................................................................. 11  
  - Software Design ........................................................................................... 12  
System Description ........................................................................................... 15  
Software Implementation .................................................................................... 16  
  - Main Module ................................................................................................ 16  
  - Initialization Module ..................................................................................... 16  
  - Timer Module ................................................................................................ 16  
  - Transmitter Module ....................................................................................... 17  
  - Receiver Module ............................................................................................ 17  
  - Serial Port Module ......................................................................................... 18  
Hardware Implementation ................................................................................... 18  
Test Plan ............................................................................................................. 18  
  - Hardware Testing .......................................................................................... 18  
  - Software Testing ........................................................................................... 19  
  - Performance Testing ..................................................................................... 19  
Discussion and Analysis of Results ................................................................. 20  
Problems Encountered ..................................................................................... 21  
Summary and Conclusion .................................................................................. 22  
References ......................................................................................................... 22  
Appendix A – Schematics ............................................................................... 23  
Appendix B – Pseudocode Listing ................................................................. 23  
  - Main Module ................................................................................................ 23  
  - Initialization Module ..................................................................................... 23  
  - Serial Module ................................................................................................. 23  
  - Timer Module ................................................................................................. 23  
  - Transmitter Module ....................................................................................... 24  
  - Receiver Module ............................................................................................ 24
ABSTRACT

The work discussed in this report addresses the task of enabling underwater communication in a multiple vehicle system for purposes of studying linked control. The system involves three fin-actuated robotic fish operating in a tank of dimensions 20 feet long by 8 feet wide by 8 feet deep. Each fish in the system requires one transceiver and an external fourth transceiver is used to synchronize and monitor the activity in the system. Because coordinated control of this multivehicle system is impossible without communication, a reliable communication system must be developed using low cost, off-the-shelf components. The robotic platform was not designed with communication electronics in mind, so custom hardware must be developed that will fit inside the constrained hull of the fish. Once the board hardware has been selected and the layout complete, software must be written that provides an interface to the main fish processor, encodes data before transmitting and decodes data after receiving. Testing of the system will show that a wireless transceiver can be made to work very reliably in air and water.

INTRODUCTION

The Nonlinear Dynamics and Control Lab at the University of Washington has designed and built three fin-actuated robotic fish for use in their research. The specific research focuses primarily on control methods for nonlinear and coordinated systems. To support these efforts, a communication system was developed that allowed the fish to transmit and receive telemetry data such as depth, heading, and time.

Several methods of communication were considered including optical, acoustic and radio frequency. The highly direction optical method makes the transmission and reception of data difficult because numerous external receivers would need to cover the outer surface of the fish so that it can communicate at any heading. The acoustic method designed for open water, fails consistently because of reflections on the walls of the tank. Radio frequency is the only method that allows for omnidirectional communication. However, the conductivity of water complicates the design of a radio system because radio waves do not propagate well in a conductive medium.
Given these limitations, it is still possible to design and build a radio-based communication system that can transmit underwater.

Linx Technologies is a company that makes low cost radio frequency (RF) modules for use in wireless applications. The key advantage to using these modules, besides cost, is the simplicity they offer. They require no external passive components and are very reliable in modulating and demodulating RF signals. A microcontroller from the Atmel AVR family was selected to handle the interface between the transceiver’s RF components and the outside, which in this case is the fish’s processor. The AVR design tools are inexpensive, easy to use and very stable. These tools include the JTAG ICE (in-circuit emulator) for on-chip debug as well as flash programming, and the AVR Studio IDE. Additionally, the CodeVision AVR C compiler was used because of its feature set which includes high optimization, bit level register access and efficient RAM usage. Finally, board layout was done with Altium Designer and the gerber files were sent to PCBFabExpress, a company in Sunnvale, CA for fabrication.

PROJECT DISCUSSION

The robotic platform was designed to support an MPC555 control board, a depth sensor and a series of servo motors for fin actuation. Due to the size of the test bed, a water tank measuring 20 feet by 8 feet by 8 feet, the fish were designed to be as small as possible while still accommodating their electronic payload. Communications hardware was not considered during the robot design process and so an added degree of difficulty exists in designing the transceiver board and antenna to fit inside the hull of the fish. The metal construction of the fish also affects the propagation of RF and so antenna placement becomes an important issue as well.

The goal of this project is to provide the robotic fish with a method of reliable underwater communication. The complete project includes at least four functional transceivers with a digital interface to the fish’s onboard processor. Additionally, an effective antenna must be purchased or designed and integrated into the fish in such a way as to maximize transmission and reception of data.

The timely completion of this project will support the research efforts of Dr. Kristi Morgansen and the Nonlinear Dynamics and Control lab. A tentative schedule was defined at the beginning of the quarter with specific milestones to be met along the way.

<table>
<thead>
<tr>
<th>Week</th>
<th>Project Goal by the end of the week</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Finish schematic capture and layout</td>
</tr>
<tr>
<td></td>
<td>Send Gerber files off for fabrication</td>
</tr>
<tr>
<td>2</td>
<td>Receive new boards</td>
</tr>
<tr>
<td></td>
<td>Verify fabricated boards are correct according to the layout</td>
</tr>
<tr>
<td></td>
<td>Stuff boards</td>
</tr>
<tr>
<td>3</td>
<td>Modify boards as necessary to accommodate a baud rate crystal</td>
</tr>
<tr>
<td></td>
<td>Complete a skeleton software module to perform basic I/O and timing</td>
</tr>
<tr>
<td>4</td>
<td>Begin software UART module</td>
</tr>
</tbody>
</table>
Enable basic communication between the dev PC and AVR microcontroller

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>Begin writing transmitter and receiver code as separate modules  &lt;br&gt; Verify software can successfully enable the transmitter and receiver modules independently  &lt;br&gt; Transmit a very basic signal</td>
</tr>
<tr>
<td>6</td>
<td>Begin coding the RF protocol use to transmit data  &lt;br&gt; Continue working on the receiver code as necessary</td>
</tr>
<tr>
<td>7</td>
<td>Finalize the RF protocol  &lt;br&gt; Finish the receiver code and receive a basic message  &lt;br&gt; Finish the transmitter and transmit a basic message</td>
</tr>
<tr>
<td>8</td>
<td>Combine the transmitter and receiver code into a single compiled hex file</td>
</tr>
<tr>
<td>9</td>
<td>Finish testing the transceivers in air  &lt;br&gt; Verify transmitters can transmit at least 300 feet in air with obstructions</td>
</tr>
<tr>
<td>10</td>
<td>Finish testing the transceivers in water  &lt;br&gt; Verify that the transceivers reliability is within spec</td>
</tr>
</tbody>
</table>

**Table 1: Project Milestones**

### Requirements and Specifications

Printed Circuit Board Specifications:

The first specification is the size of the board. Due to a limited research budget, a two-layer board is the most cost effective means to produce custom hardware. Two-layer boards lack the internal signal layers that multilayer boards have and so they are often larger than their multilayer counterparts. In a two-layer board, the top layer is where the components and traces are routed, and the bottom layer holds the ground plane and a small number of signal traces. Because of the nature of RF boards, a well designed ground plane is highly desirable, and this requires the majority of the signal traces to occupy the top layer. The size of the ground plane also affects the RF signals and must be as large as possible. This creates a design paradox in that a large ground plane is desired for efficient RF propagation yet a small board is required to fit inside the constrained compartment of the fish. Based on measurements of the fish electronics compartment and ground plane calculations the best board area is: 2000 mil by 3000 mil.

Hardware Specifications:

Hardware specifications include electronics that run at 3V. The fish has both a 5V and 3V rail so if components require 5Vs that supply will be present. Additionally, the fish has a single universal asynchronous receiver transmitter (UART) available for communication with external devices. Therefore the host processor interface must be serial port that supports TTL for current communication and RS232 for communication with a PC-104 based host processor when the fish hardware is upgraded in 2008. The UART will need to communicate at 9600 baud and use no parity or flow control as these are not implemented in the fish software.
The board should be designed to minimize power consumption. However, this is not a power critical application. The majority of power used by the fish is from the servo motors that actuate the fins and tail. These are standard model airplane servos which have enough torque to push the control surfaces of an airplane against the resistance of irregular wind. However, when these servos are used to actuate the fin control surfaces, they are not designed the push against the constant resistance of water and so they sink a great deal of current while driving the fins at maximum torque. Furthermore, the more power output by the transmitter, the further the radio signal will propagate. So the transceiver can be expected to use enough power to reliably send a radio signal. Initial estimates indicate that the transceiver will use around 200 to 500 mA during transmission.

Software Specifications:

Software should be highly optimized to use the minimum amount of random access memory (SRAM) and flash memory used by the code and variables in storage. A smaller AVR will be used for this project and small microcontrollers typically have limited SRAM and flash memory available.

The software will be responsible for initializing the microcontroller peripherals, communicating serially with the host processor on the fish, encoding and sending data, and finally receiving and decoding data. Interrupt service routines must be efficient because reception (decoding), transmission (encoding) and serial communication all rely on uninterrupted timing. Interrupt service routines will need to handle the input capture input from the receiver, the serial transmit and receive interrupts and the timer interrupts.

Range and Reliability Specifications:

RF does not propagate well in a conductive medium so the signal will need to be as powerful as possible. Higher frequencies of RF are attenuated more than lower frequencies so the carrier frequency of the RF signal should be around 50 to 350 MHz. It is expected that some signals will be dropped and the control algorithm for the multi-vehicle system will be designed to be robust to signal loss. Given this limitation, the transceiver should reliably send and receive data underwater to a depth of up to 5 feet, over a distance of 8 feet. This represents the fact that the fish will be swimming relatively close together in a coordinated formation and this shorter distance will increase the chances of a successful communication.

Design Procedure

The first step in the transceiver design is to select components and complete a printed circuit board (PCB). Many circuits can be successfully prototyped by placing the unsoldered components into a breadboard or protoboard. The problem with using a breadboard with high frequency signals is that the capacitive loading inherent in all breadboards will directly affect the signal quality and render the design unstable. Fortunately, some limited pre-PCB prototyping can be done with a wire-wrap tool and a perf board. This is still not an ideal situation but it can be used for proof-of-concept testing before a PCB design is started.
Component Selection:

Due to the limited board area, component size and cost are important in considering which components to use. Small Outline Integrated Circuit (SOIC) and Thin Quad Flat Package (TQFP) integrated circuit packages were selected because of their size advantage over the larger Dual Inline Package (DIP) components. All passives, including resistors and capacitors, were selected to be either 0805 or 0603. These are standard sizes for surface mount components. For example, an 0805 resistor is 80 mils long and 50 mils wide. Smaller components such as 0402 would be too difficult to solder by hand.

The Linx modules were selected during the early design process and the LR series transmitter and receiver pair at 315 MHz were purchased because these modules represent the longest range and lowest frequency of the Linx product line. The LR series receiver is ultra sensitive and will pick up very faint signals, making it ideal for use in the water. To boost the transmit stage, a Linx broadband amplifier was also purchased and integrated into the design. This module is a Darlington current amplifier designed to greatly boost the power of the RF signal at the antenna. Unfortunately, the amplifier requires a 5V supply. This is a minor disadvantage since higher power signals travel further, therefore an amplifier is essential for success. Linx Technologies also offers a transceiver module, but this module was very new and only available in limited quantities at the time the design was started. Furthermore, designing a transceiver using an independent transmitter and receiver module allows for only the transmitter to be amplified. It is unnecessary to amplify the input to the receiver since it is already highly sensitive.

The design will feature a single antenna shared between the receive and transmit stages of the transceiver. This requires an RF switch whose function serves two primary purposes. First, it maintains a 50-ohm match between the antenna and the modules, preventing both modules from seeing a 25-ohm impedance. This mismatch would result in losses that would degrade the range and performance of the system. Second, it prevents the full output power of the transmitter from being placed directly on to the sensitive front end of the receiver. An ideal RF switch will feature a low insertion loss of less than 0.5 dB and a high isolation of greater than 30dB. These requirements suggest that an RF signal injected into one input of the switch will not be attenuated by any noticeable factor and it will not couple onto the other input pin.

An ATTiny2313 microcontroller was selected because of its size, cost and features. This microcontroller has an input capture pin which is necessary to measure the pulses and spaces provided by the receiver’s output pin. The ATTiny2313 also features a debugWire interface for on chip debug, simplifying the software debugging process. It includes 1k of flash memory which was enough for basic functionality. This part was later replaced with an ATMega168 which had eight times the flash memory, features an analog to digital input and is offered in a 32-pin TQFP package which is actually smaller than the 20-pin SOIC package of the ATTiny2313. The Linx receiver is able to output an analog signal representing the received signal strength. The analog to digital converter of the ATMega168 will be able to make use of this feature.

The processor on the fish requires 3V logic levels and an additional 5V rail is available for the broadband amplifier. The current fish processor, an embedded PowerPC 555 module, will
eventually be replaced by an AMD Geode in a PC-104 form factor in early 2008. The PowerPC electronics can communicate using direct 3V TTL serial, but the future electronics will only be able to communicate using RS-232. This makes it necessary to design both RS-232 and TTL serial on the transceiver board. Since TTL and RS232 can’t be active simultaneously, some board real estate must be dedicated to either a multiplexer or a set of jumper pins.

Connectors are also important and it is beneficial to consider how the embedded system under design will connect to the larger system that it will enhance. All connections to the transceiver board were designed to use polarity-enhanced Molex connectors which prevent a cable from being connected in reverse. This removes the requirement to design reverse-polarity protection into the board’s power inputs. Polarity enhanced Molex connectors will provide inputs for the +3V input, +5V input, and UART transmit and receive lines. All RF connections will use reversed-polarity SubMiniature version A (RP-SMA) connectors, a type of 50-ohm coaxial cable and connector designed for excellent performance from DC through 18GHz and specifically designed for small RF applications.

Board Layout:

When dealing with high frequency RF signals, board layout becomes critical. The first and most important feature of the board is the ground plane which will occupy the bottom layer. The ground plane should be as large as possible because it provides a low impedance path for the RF signal back to ground. If the ground plane is small or cleaved by signal traces, then the performance of the antenna will be greatly affected. Therefore, it is necessary to minimize the number of signals on the bottom layer. If a signal needs to be routed there, it should be routed away from the antenna in order to minimize the trace’s profile.

A transmission line is a medium where RF energy is transferred from one place to another with minimal loss. Therefore, any trace that carries an RF signal becomes a critical element, especially because the trace leading to the antenna can effectively contribute to the antenna’s length and change its resonant bandwidth [2]. Note that antenna length is designed to be either a half or a quarter of the wavelength of the carrier signal. Furthermore, some antennas are designed in a helical style to reduce height, and these antennas are more susceptible to detuning when they are not fed with a perfectly matched trace. In order to minimize detuning and power attenuation from reflections, an impedance-matched transmission line must be used. In the case of RP-SMA, all traces must be matched to 50-ohms. The variables that come into play when designing a matched 50-ohm transmission line (also known as a microstrip trace) are the width of the trace, the thickness of the PCB and the dielectric constant of the PCB material. The transceiver PCB is 62 mils thick and made of fire resistant 4 (FR4) material, which has a dielectric constant of 3.
The Linx modules also require layout considerations. The data sheet suggests that each module must not have signal traces routed under the module to reduce the chance of signal coupling. Conductive elements must be kept at least 150 mils from the edges of the module. Each ground pin must have its own via to the ground plane, rather than routing it through a top-layer ground trace. Each module should be isolated from other high frequency components on the board, such as the crystal oscillator that feeds the microcontroller. This also eliminates the use of a switching voltage regulator in close proximity to the modules.

The layout for the final transceiver board is still under development. The layout for v0.4 of the board can be seen in Figure 3.
Antenna Design:

The antenna is arguably the most important element of an RF project. Whip style antennas are typically inexpensive or easily fabricated and provide outstanding overall performance and stability [2]. Whip antennas are typically designed to be either a half or quarter of the wavelength of the carrier frequency. In the case of a quarter-wave whip antenna, also called a quarter-wave monopole, only half of the antenna is present in the physical antenna device. The other half of the antenna is the ground plane which is ideally designed to be infinitely large. Unfortunately, the amount of board area which approximates an infinite ground plane is far too large for most embedded boards and so these antennas are typically less efficient at radiating and receiving signals. They do have a size advantage over the half-wave dipole antenna which is twice as long but does not require a large ground plane. This is because the whole antenna is contained within the physical antenna device. Helical antennas use an internal winding of the conductive element to reduce the height below the quarter wave standard size. This also reduces the antenna’s bandwidth making it more susceptible to detuning, but the reduction in size is invaluable for compact applications such as the robotic fish.

![Whip Style Antennas](image)

Figure 4: Whip Style Antennas [2]

In addition to the design of the antenna, the radiation pattern of the antenna will also affect performance. The radiation pattern refers to the way in which RF energy is directed into the surrounding medium, which in the case of the robotic fish is a 1280 cubic foot tank of water. Antennas radiate poorly in some directions and efficiently in others. The quarter-wave monopole antenna, for example, has a half donut shaped radiation pattern with no energy dispersed below the ground plane. It is this directionality that makes the mounting and orientation of the antenna a critical factor to consider in the design.
Ultimately, it will come down to testing both styles of whip antenna and possibly testing a fabricated antenna made from flexible wire. There is truth to the saying that antenna design is black magic. The style of antenna chosen for the final design will be that which has the best test performance, not the best theoretical performance.

**RF Modulation:**

The type of modulation chosen for the RF application will affect its performance in several ways. While OOK modules were chosen for this project due to their cost and simplicity, it is worth mentioning some key drawbacks of OOK. As explained previously, OOK modules convey digital data by representing a logic one with the presence of carrier and a logic zero with the absence of carrier. This is very simple to implement in silicon and the ease of implementation is reflected in the component’s cost. However, a significant drawback is that signals modulated using OOK require a much wider band than non-OOK signals. This increases the possibility of interference from near-band signals. Another drawback to OOK is that the RF signal is not continuous. The receiver has no way of knowing the difference between signal dropout and a logic zero.

Another modulation type, frequency shift keying (FSK), uses a pair of frequencies to convey digital data. Because the signal is continuous, it is easy for the receiver to detect a loss of signal. A more important advantage is that FSK requires a narrower band, so the front end of an FSK receiver may be designed with a very narrow band pass filter, rejecting all but the carrier and increasing noise immunity over its OOK counterpart.
Software Design:

The software design involves writing an AVR firmware that will successfully interface the fish’s processor with the onboard RF electronics. In addition to the interface, the software is responsible for the encoding of data to be transmitted and the decoding of received data. This requires an efficient protocol to sustain the wireless network.

The Linx modules are enabled or disabled by grounding an active-high power down pin. The power down input is important since the transmitter and receiver must not be active at the same time to ensure the sensitivity of the receiver. The same lines that control the power down function of the modules may also be electrically tied to the control line of the antenna switch. This allows the microcontroller to control the switching of the antenna and powering down of the modules with a single output rather than two or more.

The software also needs to implement a protocol responsible for encapsulating important data such as heading, time and depth. The goals of the protocol are to synchronize communications between the transmitting and receiving ends, identify valid data packets, verify that the data packets are correct, and possibly even correct bad data in a packet [5]. A protocol provides a degree of noise immunity and reliability, and is almost always implemented in software. Considering the generalized communications channel of Fig. 6, the protocol and software will be responsible for all but the propagation path.
The data source includes an identification number used to identify the transmitting fish, the heading of the fish, its depth and a time stamp to mark the time of the transmission. The software then encodes the data to provide a degree of structure and security. Once encoded, the data is transmitted and nearby fish will perform the process in reverse from reception through interpretation of the received data.

The implementation of a protocol is a major design issue and should involve the majority of the software design process. The important features of a protocol include packetization, overhead, reliability, robustness, uniqueness and the encoding method used to represent data bits. Each of these features must be thoroughly examined before the first line of code is written.

The packetization of a protocol represents how the important data is divided up and included in the data stream. The type of encoding will also affect this, as some encoding methods require additional bits. The order of presentation also affects the packetization of data. What order should the data be sent in? Should the data be sent as a single long transmission or as multiple short transmissions?

The protocol overhead refers to the amount of additional bits per bit or per unit of data in the protocol. Stop bits, start bits, sync bits and error correction bits all contribute to overhead. The amount of information added should be the minimum amount required to achieve all the goals of the wireless data transfer.

Reliability refers to a protocol’s ability to detect errors in the transmission. Without some method of error checking, the receiver would accept any data sent to it as valid. Reliability is most often enabled in a protocol by using a parity bit or a series of checksum bits. Because processing power is at a minimum in a low-memory 8-bit microcontroller, more advanced methods of error checking, such as the cyclic redundancy check (CRC), are discouraged.

Robustness refers to a protocol’s ability to recover from errors. Forward error correction (FEC), which embeds codes into the data stream, will allow the receiver to correct and recover from single bit errors, but increases protocol overhead considerably. The testing phase will show if many single bit errors are present in which case a FEC enhanced protocol would be very useful.
Three fish with transceivers operate in what is known as a broadcast network. That is, each fish that transmits will be received by two others in its vicinity. If the transmitting fish cannot identify itself to the others, the receiving fish will not know which of the three sent the message. Uniqueness refers to the ability of the protocol to include some type of identifier in the message so the receivers know who sent the message.

Finally, the encoding of the data refers to how the bits are represented. The Linx modules allow the designer total control over which encoding method to use. Two prominent methods include pulse space encoding and Manchester (biphasic) encoding. Pulse space encoding conveys digital data by changing the period of the transmission. Manchester encoding conveys data based on the digital transition, either low to high or high to low.

![Figure 8: An example of pulse-space encoding](image1)

![Figure 9: An example of Manchester encoding](image2)

The RF modules have specific turn-on times which affect the quality of the received signal by stretching the received pulse slightly. This is further aggravated by the effects of an RF phenomenon called dispersion where different frequency components arrive at the receiver at different times. Because data conveyed by transition ignore the actual length of the pulse, Manchester encoding would be a better method to implement.

![Figure 10: Scope trace showing the effects of pulse-stretching](image3)
System Description

The block diagram of the system can be seen in Fig 11.

![Figure 11: Transceiver System Block Diagram](image)

The transceiver communicates with the outside using a standard serial port. This interface includes the UART transmit and receive pins. A third ground pin is not necessary since the UART ground, transceiver ground, and host processor ground is the same node. The power input to the system includes a +5V and +3V rail provided by the host processor.

The ATMega168 processor is responsible for encoding and decoding data from the transmitter and receiver modules respectively. Microstrip traces are used to route all RF signals throughout the system. The antenna switch is controlled directly by the ATMega168 and switches between transmit and receive as necessary.

Software flow though the system is detailed in Figure 12. The current software encodes data one character at a time and transmits each character individually. When a serial character is received by the ATMega168, it is placed into the UART receive buffer by the UART receive interrupt service routine (ISR). The getchar function is used to fetch the character from the UART receive buffer and then the transmit routine is used send the character out through the propagation path. When the transceiver receives data from its antenna, the data is processed by a Manchester decoder state machine that verifies the data is valid. If valid data is present, the state machine places it via putchar in the UART transmit buffer. The UART transmit ISR then outputs any characters in its buffer to the UART TX pin.
The current software implementation uses Manchester encoding where the bit period is 1800 us. Eight data bits are sent per transmission and are encapsulated in a frame that contains two sync bits and a header bit. The header bit serves the function of waking up the receiver and increasing the signal-to-noise ratio so the receiver can differentiate between random noise and valid data.

**Software Implementation**

The software is divided into six modules: the main module, the initialization module, the timer module, the transmitter and receiver modules, and the serial port module.

**Main Module:**

The main module is responsible for calling the initialization module and entering an infinite loop that switches between transmit and receive. The software is primarily interrupt driven and exists in a receive state most of the time unless a UART event occurs. When a character appears in the UART receive buffer, the main loop immediately powers down the receiver, powers up the transmitter, pulls the character from the buffer and calls the transmit module to send it. The main loop will remain in transmit mode until there are no characters in the UART receive buffer.

**Initialization Module:**

The initialization module is used to initialize the AVR hardware and peripherals. This includes powering down any unused peripherals. Both the 8 and 16-bit timers are set to their default values here. The digital IO ports are set to their default values and this includes the control lines to the Linx modules. The AVR UART is initialized here as well, set to 9600 baud, no parity, 8 data bits and 1 stop bit.

**Timer Module:**

The timer module holds all the functions necessary to start and stop the timers and includes the interrupt service routines for the timers. It also contains the two initialization routines for the timers which are called by the initialization module. Timer 1 doubles as an input capture, a hardware peripheral that measures and timestamps incoming pulses. The time stamps from this peripheral feed the software state machine responsible for decoding incoming transmissions.
Transmitter Module:

The transmitter software module uses the 8-bit timer1 to generate the Manchester encoded signals and feeds the input to the Linx transmitter. The main function in this module is the send_byte function used by the main loop to send an 8-bit word out through the transmitter. The send_byte function calls both the send_zero and send_one functions to send Manchester-encoded zeros and ones as necessary. The send_byte function also packetizes the 8 data bits with two sync bits and a header bit.

Receiver Module:

The receiver module uses a modified version of the software state machine developed by Guy Carpenter of Clearwater Software. Carpenter’s software was written to decode RC-5 encoded infrared data but it was easily modified for use with a Manchester radio system. (The software is freely available without warranty or license for private and commercial use). The state machine is fed a series of pulse and space widths provided by the input capture module. The initial sync bit provides the state machine with a leading one bit, which it expects. The pulses and spaces advance the state machine through each state until an error occurs or a whole word has been decoded. The states are named for the position in the signal, for example, a mid1 state represents the middle of a one bit. That is, a high to low transition has just occurred.

![Manchester Decoder State Machine](image)

**Figure 13: Manchester Decoder State Machine**

The main function of the receiver module is rc5_notify which returns a 1 when a valid word has been decoded. This function is continually called by the main loop and when a valid word is available; the word is retrieved with rc5_getcode and sent out through the UART.

An added benefit of using a software state machine to decode incoming transmissions is that the state machine will automatically fail when noise is being received rather than data. The Linx receiver’s sensitivity is below the noise floor of the board, so thermal noise will be output as data. This data appears to be randomly switching pulses and spaces and must be handled in software. The software state machine can easily distinguish between this and valid data because
the noise pulses are consistently out of spec and will trigger the state machine to reset via its error state.

![Scope trace showing received noise and valid data.](image)

**Figure 14:** Scope trace showing received noise and valid data. The transmitted signal is probed on channel 1 and the received signal is probed on channel 2. Notice that the received signal follows the transmitted signal but exhibits random switching before the transmitted signal is sent.

Serial Port Module:

The serial module drives the AVR’s serial port hardware. It includes both interrupt service routines to handle a UART receive or transmit event. The receive ISR copies the data byte into the receive buffer while the transmit ISR sends out through the serial port any data that exists in the send buffer. The functions getchar and putchar define the public interface to this module which allow other code to retrieve data from the receive buffer or place data in the transmit buffer to be sent immediately.

**Hardware Implementation**

The block diagram of Figure 11 details the interconnection of hardware in the system. A schematic for v1.0 of the transceiver board is located in the appendix.

**TEST PLAN**

Testing the system involves testing the hardware, software and performance of the complete system.

**Hardware Testing**

Hardware testing is an ongoing process. The first step involves verifying that the PCB returned from the fab house is valid and matches the layout. It is important to check for any shorts or
opens, especially in vias or traces that are routed close together. A quality fab house will produce boards that match the gerber files generated by the board cad software. However, cheaper fabs will produce low quality boards that may exhibit unexpected flaws. After the board is verified and components are stuffed, another electrical verification should be done. Solder joints must be well formed and checked for proper conduction. A multimeter’s continuity tester is a useful tool during this type of testing.

**Software Testing**

Software testing involves finding and removing any bugs present in the code. This is done using the JTAG ICE, a tool developed by Atmel which enables on-chip debug. With this tool, code can be stepped through line by line and the contents of both memory and registers can be examined for proper operation. The debugger, together with a digital oscilloscope, can verify the internal software and the external circuit operate correctly.

**Performance Testing**

Performance testing involves both sending a message or series of messages from one transceiver to another and recording how many of the transmitted messages and characters are received. This allows a failure rate to be determined for the testing conditions. This type of testing must be performed in both air and water with and without obstructions in the propagation path. The fish system is designed to operate successfully if some data are lost so the transceivers do not have to perform perfectly. These tests will allow the hardware to be characterized for a specific environment and will determine if further refinement is needed.

The transceiver testing completed so far includes both in-air and in-water testing. The first test involved leaving the receiver inside the lab and relocating the transmitter at least 250 feet away to the north and, in a second trial, to the east. A 37-character message was transmitted with obstructions present that included the walls of the lab building, trees and metal objects such as parked cars and a trash dumpster. The results of the test are recorded in Table 2.

<table>
<thead>
<tr>
<th>Test 1 Results:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Total Characters Transmitted: 6843</td>
<td></td>
</tr>
<tr>
<td>Total Characters Received: 6712</td>
<td></td>
</tr>
<tr>
<td>Character Reliability: 98.1%</td>
<td></td>
</tr>
<tr>
<td>Total Messages Transmitted: 90</td>
<td></td>
</tr>
<tr>
<td>Total Messages Received: 65 (without error)</td>
<td></td>
</tr>
<tr>
<td>Message Reliability: 72.2%</td>
<td></td>
</tr>
<tr>
<td>Total Messages Received (with 1 missing character): 3</td>
<td></td>
</tr>
<tr>
<td>Total Messages Received (with 2 missing character): 7</td>
<td></td>
</tr>
<tr>
<td>Total Messages Received (with 3 missing character): 3</td>
<td></td>
</tr>
<tr>
<td>Total Messages Received (with 4 missing character): 3</td>
<td></td>
</tr>
<tr>
<td>Total Messages Received (with 5+ missing character): 9</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Test 2 Results:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Total Characters Transmitted: 6080</td>
<td></td>
</tr>
<tr>
<td>Total Characters Received: 5824</td>
<td></td>
</tr>
<tr>
<td>Character Reliability: 95.8%</td>
<td></td>
</tr>
<tr>
<td>Total Messages Transmitted: 80</td>
<td></td>
</tr>
<tr>
<td>Total Messages Received: 56 (without error)</td>
<td></td>
</tr>
<tr>
<td>Message Reliability: 70.0%</td>
<td></td>
</tr>
</tbody>
</table>
Table 2: Results of Transceiver Testing in-Air

The second test involved testing the transmitter and receiver in the actual water tank testbed where the fish system will operate. The transmitter was placed in the hull of a robotic fish and the receiver was placed on the edge of the pool. The test involved sending a 37-character message from the submerged fish to the receiver attached to a laptop computer. The results were as follows:

- Fish submerged to 4 feet: 100% character reception.
- Fish submerged to 8 feet at front of tank: 59% character reception for 27 messages sent.
- Fish submerged to 5 feet at rear (opposite end from receiver) of tank: 64% character reception for 32 messages sent.

Table 3: Results of Transceiver Testing in-Water

**DISCUSSION AND ANALYSIS OF RESULTS**

The testing of the existing system proved to be very promising and, as expected, the system performed better in air. The water test, while helpful, was not a true indicator of water performance. The test suffered from antenna placement because the antenna was not secured to the fish in any way. The polarization pattern of the antenna was constantly changing from optimal to non-optimal as the fish’s hull changed position in the water. It is clear that more testing is needed and a test that uses a separate style of antenna would also be beneficial.

The transceiver software at this point is functional, providing the host processor with the ability to send and receive serial data over a wireless link. However, it acts only as a transparent serial pass-through. The protocol needs to be enhanced with error detection and possibly error correction. Further testing is needed to determine if a series of small transmissions holding a portion of the data are more reliable then a longer transmission containing all the data. This testing will commence once the software has been enhanced with the ability to send a long transmission. The current serial pass-through functionality will not be replaced; rather it will be included as one of two operational modes. The other mode will be a single transmission of many bits. Both modes may be enhanced with error detection and correction.

The software will also be expanded to enable communication with the host processor over the synchronous serial peripheral interface (SPI) and inter integrated circuit (I2C) busses. Adding this functionality increases the flexibility of the transceiver system since a new sensor module may be added to the fish in the future which requires use of the serial port. By switching the hardware to SPI or I2C mode, the serial port may be freed up for the new sensor and the transceiver may still be used.
The hardware has proved to be reliable and there is no reason, at this point in time, to change it. The ATMega168 microcontroller is more than capable of performing the encoding and decoding tasks and has the flash memory and SRAM for future expansion of the transceiver software. The Linx modules may be replaced in the future with a lower frequency custom FSK transceiver, but this will be a project in itself. Testing of the multivehicle system is needed to determine what changes, enhancements and refinements need to be pursued.

PROBLEMS ENCOUNTERED

The processes of developing software and hardware for an embedded system is an endless cycle of finding errors in hardware and software, debugging them and verifying that they are indeed fixed. This process occurred many times during the development of the transceiver.

An early error manifested itself as a constant failure of the Manchester decoder state machine. The failure was traced to an inability of the CodeVision AVR compiler to compile multidimensional arrays of structures. Once this error was identified, the functionality that was coded using a multidimensional array of structs was replaced with a nested case statement. While this control structure was less efficient in flash memory, it did compile correctly, fixing the error.

During later development of the protocol, the receiver state machine was working correctly but the received data bits were not being received correctly. Bit shifting was occurring constantly even though the transmitted bits were correctly received by the receiver. Examining the state machine code revealed that it requires the data to be framed by spaces. Therefore, a pair of sync bits were needed to frame the data so the state machine could operate correctly. Once this bug was corrected, testing could finally begin on the range performance of the system.

Another bug involved the transmitter output operating erratically even though the code was known to be correct. While the behavior appeared to be related to a software bug, in fact it was a hardware issue. The original board had to be modified and these changes were later incorporated into the v1.0 schematic which appears in Appendix A. The modifications changed a few microcontroller pins and required that several traces be cut so the modifications were not electrically tied to their old pin assignments. On the malfunctioning board, the old traces were not cut and the receiver output was feeding the UART receiver on the AVR microcontroller. This caused many receiver ISRs to trigger unexpectedly which, in turn, affected the output of the transmitter. This hardware bug was easily fixed by cutting the traces, but it prompted a serious code review to determine how the ISRs affect the operation of the system and if there are any chances during normal operation where an ISR could invalidate a good data packet or throw the system out of sync. The bug led to better interrupt handling since the current software disables receiver interrupts when transmitting.

An existing bug is present in the code that prevents the transceiver’s serial port receiving characters faster than 25ms. This bug is currently under investigation but it does not affect the ability of the system to operate faster than the data rate of an acoustic modem which it is designed to approximate.
SUMMARY AND CONCLUSION

The goals of the project have been met. A set of specifications were outlined early in the quarter and a schedule of milestones was drafted to aid progress. Components were selected based on cost, size and reliability. These components were then input into a board cad program and a board layout was completed. The resulting PCB was stuffed with components and the design cycle progressed into the software development phase. Several design changes were made during software development, but none of these changes impacted the schedule. A functional transceiver was brought up and testing began to evaluate performance both in and out of the water.

With the initial round of testing complete, it has been determined that the existing design is functional and will be sufficient to complete a series of data gathering trials with the three-fish system for an upcoming research paper due in early 2008. The current board, while functional, does have room for improvement and these design changes have been recorded and the software and hardware will be updated as necessary. The most important software change is the expansion of functionality including an I2C and SPI interface to the transceiver and a new method of encoding data into a longer transmission. The current software that implements a serial pass-through will remain unchanged and will be integrated into the future code as one of two modes of communication, switched by an external hardware switch. These changes, once implemented, will need to be tested for correct operation.

REFERENCES

APPENDIX A - SCHEMATICS

Schematics are available in the hardcopy only.

APPENDIX B – PSEUODOCODE LISTING

Main Module

Initialize AVR hardware
Output SW version number over AVR serial port
Enter main infinite loop
    While (data exists in the UART receive buffer)
        Transmit characters as Manchester encoded words
        If a Manchester encoded character has been decoded
        Output the character to the AVR’s serial port

Initialization Module

Initialize Digital I/O PORT A
Initialize Digital I/O PORT A
Initialize Digital I/O PORT A
Initialize Digital I/O PORT A
Initialize transmitter and receiver default state – transmitter off, receiver on
Initialize external interrupts
Initialize Timer 1
Initialize Timer 2
Initialize Analog Comparator – disable
Enable Sleep Code
Initialize timer interrupts
Enable global interrupts

Serial Port Module

UART Receive Interrupt Service Routine
{
    If UART data has been received without error
        Copy data to the UART receive buffer
}

Getchar
{
    While no data exists in the receive buffer {}
    Create a copy of the data in the UART receive buffer
    If the buffer index has overflowed
        Reset the index
    Decrement the UART counter that keeps track of how many characters are in the buffer
    Return the copy of the data
}

UART Transmit Interrupt Service Routine
{ 
    If data exists in the transmit buffer
    Copy the data to UDR, the serial port transmit register
    If the index has overflowed
        Reset the index
}

Putchar
{ 
    Copy the data to be transmitted into the UART transmit buffer
    If the index has overflowed
        Reset the index
    Increment the counter, that keeps track of how many characters are in the buffer
}

**Timer Module – Key Functions**

Timer 0 Overflow ISR
{ 
    Do nothing
}

Timer 0 Compare A Match ISR
{ 
    Do nothing
}

Timer 0 compare B Match ISR
{ 
    Stop timer 0
    Reset the timer 0 count register
}

**Transmitter Module – Key Functions**

Send_byte
{ 
    Enable transmitter
    Disable receiver
    Send header byte (start byte)
    Send sync bit (one)
    Call send_zero_bit() and send_one_bit() to send the next 8 bits of the data word
    Send sync bit (zero)
    Disable transmitter
    Enable receiver
}

Send_zero_bit
{ 
    Set timer 0 compare A match to 800 us
    Set timer 0 compare B match to 1900 us
}
Bring transmitter line high
Start timer 0
Enter low power mode and wait for timer 0 to interrupt on compare A match
Upon exiting from the timer 0 compare a match ISR, bring TX line low
Enter low power mode and wait for timer 0 to interrupt on compare B match

Send_one_bit
{
  Set timer 0 compare A match to 800 us
  Set timer 0 compare B match to 1900 us
  Bring transmitter line low
  Start timer 0
  Enter low power mode and wait for timer 0 to interrupt on compare A match
  Upon exiting from the timer 0 compare a match ISR, bring TX line high
  Enter low power mode and wait for timer 0 to interrupt on compare B match
}

Receiver Module – Key Functions

Timer 1 input capture ISR
{
  Copy time stamp of current ICP event to current time stamp variable
  rf_width = current time stamp – last time stamp
  if the RX data line is low
    then the port is currently a 0 and a high to low transition just occurred.
    The last state was high, so this is a pulse width
    Rf_type = 1
    Set the ICP module to trigger on a rising edge
    Copy the current time stamp into the last time stamp variable
  Else if the RX data line is high
    Then the port is currently a 1 and a low to high transition just occurred
    The last state was low, so this is a space width
    Rf_type = 0
    Set the ICP module to trigger on a falling edge
    Reset timer 1
    Set the last time stamp to 0
}

Manchester Decoder State machine

The Manchester decoder state machine is available unmodified at this web site: