HEART RATE ALARM SYSTEM FOR SWIMMER IN TRIATHLON

By

Yunye Gong

Zilin Dou

Final Report for ECE 445, Senior Design, Fall 2012

TA: Justine Fortier

12 December 2012

Project No. 26

Abstract

The heart rate alarm system is designed to help triathlon swimmers get noticed when their heart rates behave abnormally or when they feel uncomfortable during competition. The system consists of two units. The swimmer unit is triggered by a microcontroller when either manual switch is pressed on or pulse sensor detects an out-of-range heart rate. Then LEDs and buzzer on swimmer side will alert surrounding swimmers. While in the rescuer unit, LEDs and buzzer are triggered by wireless transmitted alarm signal to notify rescuers. The system can successfully function when the swimmer unit is placed underwater and the rescuer unit is 60 meters away. And there is still room for improvements such that the system can be implemented more professionally with surface mount PCB boards and a suitable waterproof container for the swimmer unit. Additional functions such as monitoring sudden heart rate changes can also be added via programming.

Contents

1. Introduction

1.1 Purpose

In recent years' triathlon races, several sudden deaths of swimmers have occurred because of heart attack or some other acute heart problems. If uncomfortable swimmers can get noticed by other competitors or professional rescuers nearby, their chances of being helped before tragedy happens will greatly increase. However, in real triathlon swimming competitions, it's hard for athletes to notice emergencies happening around and it's hard for rescuers to find people needing help in crowded water. Therefore the motivation for our project is to provide those swimmers a remote alarm system that will alert surrounding swimmers as well as rescuers some distance away when either their heart rate goes abnormal or they feel uncomfortable, such that they can get timely help which may possibly save their lives.

1.2 Project Functions

The system consists of a swimmer unit and a rescuer unit. The pictures of finished units are shown as Figure B.2 and B.3 in Appendix B. The swimmer unit is designed to be portable and waterproof such that it can be carried by swimmers without affecting their performances during competition. Placed in a transparent container specified as shown in Figure B.1, it contains a pulse sensor which can be installed on earlobe and provides consistent real-time pulse signal. When the heart rate calculated based on the pulse signal goes out of a preset range, the alarm will be automatically triggered. The swimmer unit also contains a pushbutton switch such that the swimmer can manually call for help when they feel uncomfortable. When the swimmer unit is triggered, the LED array will light up and the buzzer will alarm to notify surrounding swimmers. At the same time, the Xbee transmitter on the swimmer unit will send out a wireless signal. Once the signal is received by the Xbee receiver on the rescuer unit, it will be extended by a timer circuit and thus keep the rescuer side LED array and buzzer on for 11 seconds, in order to alarm rescuers who are at least 25 meters away from the swimmer.

1.3 Blocks Overview

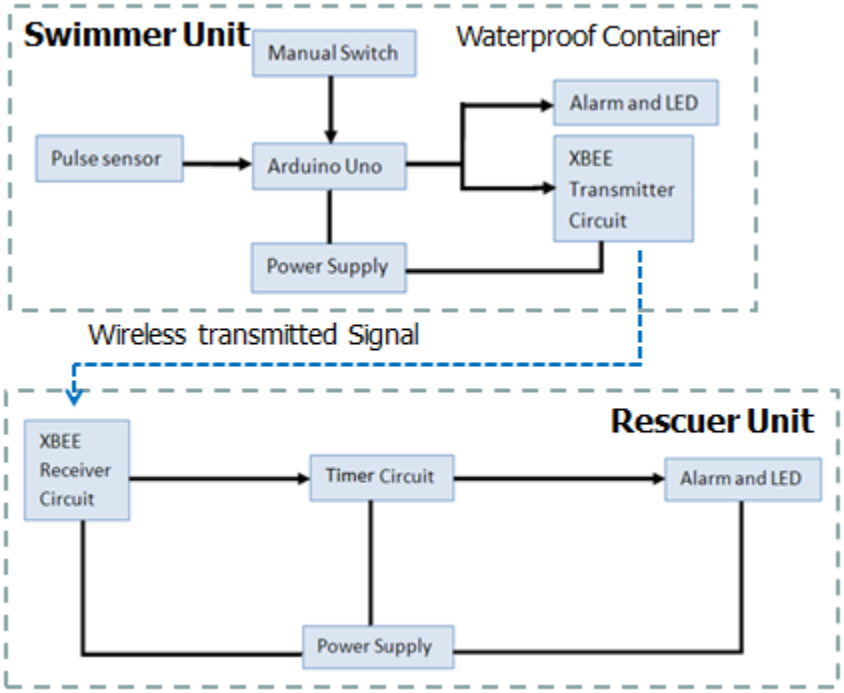

Figure 1. System Block Diagram.

As specified in the block diagram (Figure 1), our system contains two separate units. The swimmer unit is controlled by an Arduino Uno board, which accepts input from the pulse sensor and the manual switch. Arduino feeds outputs to the 2x4 LED array, buzzer and the Xbee transmitter in the swimmer unit, and triggers them on when either pulse sensor detects abnormal heart rate or manual switch is on. Compared with the originally proposed design, a select circuit is removed since the logic to generate high output according to both input signals can be efficiently done by Arduino programming.

Once the swimmer unit is triggered on, the Xbee receiver at rescuer unit will output an alarm signal pulse according to the input of Xbee transmitter. A timer circuit built with LM555 chip is used to extend the alarm pulse to 11 seconds. The output of timer circuit will be used to trigger the 3x3 LED array and the buzzer in the rescuer unit.

2 Design

2.1 Design Procedure

2.1.1 Power Supply

Considering the power requirements and performing stabilities of the containing parts, each of the swimmer unit and the rescuer unit is designed to be powered up by a 9 V battery. Since the Arduino Uno board has a recommended input voltage range 7-12 V and a lowest input limit at 6 V [1], a 9 V battery can support it to work appropriately before battery voltage drops to 6 V. The UA78M33C voltage regulators are used in both units to regulate voltages to 3.3 V since Xbee modules require specific 3.3 V VCC and logic high. Besides that, in rescuer unit, a LM7805 voltage regulator is used to regulate 9 V to 5 V since a 74LS04 inverter with 5 V VCC is used in the unit [2].

2.1.2 Pulse Sensor

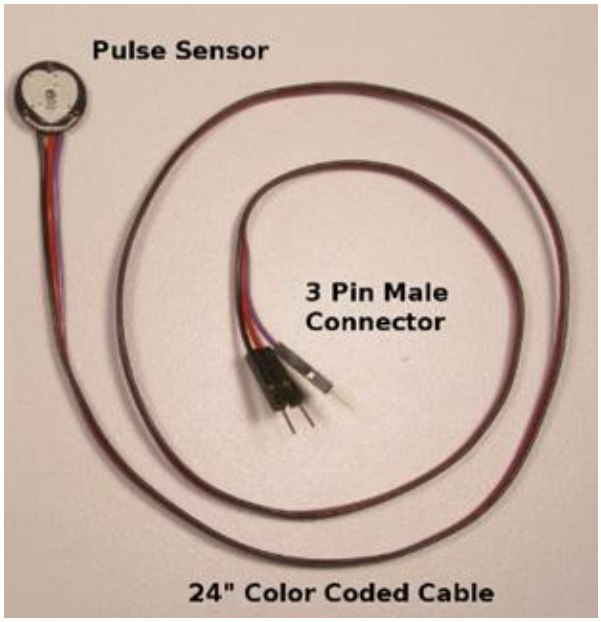

Figure 2. Pulse Sensor [3].

A pulse sensor specified as Figure 2 is utilized in our design to provide real-time heart rate information. Compared with other alternatives, although it is not specifically designed for underwater use, this sensor is chosen due to its simple connections, easy installation and relative low price. Moreover, this sensor is designed to be work together with Arduino board with clear open-source test codes provided.

2.1.3 Manual Switch

A SPST push-on-push-off button switch is used as the manual switch such that assuming heart rate is normal, pushing it once will trigger the system and pushing it again will stop the alarm. Although a momentary switch may be cheaper and easier to get, we need a push-on-push-off switch such that the output could stay stable until the button is pushed again. Compared to another alternative, a 3-pin push button particularly designed for Arduino, the chosen switch has simpler connections and lower price.

2.1.4 Microcontroller

An Arduino Uno board with ATmega328 microcontroller is used to control the functioning of the system. Since the selected sensor is designed to fit Arduino Uno, they can work together smoothly and the collected heart rate information can be used to determine Arduino outputs via programming. The multiple digital pins also enable the Arduino board to control the LED array, buzzer and Xbee transmitter separately at the same time. Compared with the original design, pin 3 instead of pin TX is used to output signal to Xbee transmitter. Pin TX/RX should not be used unless necessary since they will function when the programming is uploaded into the board. In the aspect of programming, the heart rate is calculated based on code provided by pulse sensor producer and original code is used to determine outputs according to input logics. Compared with the original logic flow chart, a specific issue of initial state is stressed on and the first several heart beat data are discarded in heart rate calculation. And the original proposed delay function is removed since the usage of delay in Arduino loop function would cause the unintended halt of the entire Arduino function during the delay time [4]. The intended function of extend alarm pulse is actually achieved by the timer circuit in the rescuer unit.

2.1.5 Xbee Modules

To realize wireless signal transmission from an underwater unit to another unit at least 25 meters away, Xbee Pro modules are used since they are featured with relatively long transmission distance up to 1500 meters outdoor [5]. They also provide multiple baud rates to be chosen such that the two modules could talk to each other without potentially influenced by other working modules in the same area. Compared with the original proposed design, pin D3 instead of pin TX/RX is used as input/output of the transmitter/receiver, since the alarm signal fed from Arduino is either high or low and no serial input/output is required.

2.1.6 LED Arrays

In both the swimmer unit and the rescuer unit, LED arrays implemented with red LED are designed to provide obvious light indicating emergency. A resistor is connected in series with each LED column. The resistance needed is calculated by the Equation (2.1).

$$R = \frac{V_{in} - n \times V_{drop}}{I} \tag{2.1}$$

In Equation (2.1), V_{in}, n, V_{drop} and I refer to the Input Voltage, number of LEDs in each column, voltage drop of each LED and the current in each column respectively. Compared with the original design, in both units, the resistances of the resistors are modified according to test results and calculation. The size of LED array in the swimmer unit is changed from 3x3 to 2x4 since 5 V of voltage high from Arduino cannot support efficient work for 3 LEDs in a column.

2.1.7 Buzzer Alarm

Each of the units utilizes a MCP320B2 buzzer to generate loud alarm when the system is triggered. The input high is 5 V from Arduino in the swimmer unit, and 9 V from timer circuit in the rescuer unit. A resistor is connected in series with each buzzer and the applied resistance is modified according to test results. Particularly for rescuer unit, in the original design, the buzzer is connected to a capacitor used in

the timer circuit before it is grounded. Test results shows that the buzzer could not work appropriately with a capacitor in series. Therefore the final design revises the connection such that the buzzer is connected between timer outputs and ground only in series with a resistor.

2.1.8 Timer Circuit

In the rescuer unit, a timer circuit built with LM555 timer is designed to extend received pulse of alarm signal before it is fed to LED array and buzzer, such that the alarm time can be kept long enough to get rescuers' attention. According to the reference tutorial [6], the delay time generated by the timer circuit is calculated by the Equation (2.2).

$$t = 1.1RC \tag{2.2}$$

Therefore the delay time is determined by the resistor and capacitor used in the circuit. Compared with the original design, the capacitor used is change from 50 µF to 100 µF to double the delay time. Two control switches are added to make sure the 74LS04 inverter is turned on before the LM555 timer to avoid the error high output pulse happens whenever the inverter and timer are turned on at the same time.

2.2 Design Details

2.2.1 Power Supply

Considering safety, efficiency and availability, we use Duracell 9 V batteries as the power supply of the design.

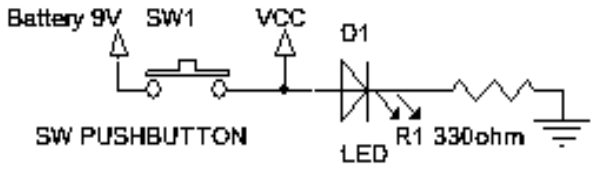

Figure 3. Schematic of Overall Control Switch.

As specified in Figure 3, each unit is powered by 9 V VCC with an overall control switch. When the pushbutton switch is pressed, the circuit will be connected such that the battery input will be used as swimmer unit VCC and a green LED will light up indicating the switch is on. The resistor value needed in series of each column of LEDs is calculated shown in Equation (2.3).

$$R = \frac{V_{in} - n \times V_{drop}}{I} = \frac{9\text{V} - 2 \times 2\text{V}}{0.025\text{A}} = 280\Omega \tag{2.3}$$

According to test results, the voltage drop of each LED and forward current are assumed to be 2 V and 0.025 A respectively. 330 Ω resistors are used due to the availability in lab.

As specified in Figure A.1 and A.2 in Appendix A, in swimmer unit, the 9 V VCC is directly fed to Arduino Vin, while a UA78M33C linear voltage regulator is used to regulate 9 V to 3.3 V which used as VCC for

Xbee transmitter. Another UA78M33C is used to regulate Arduino output at pin3 which can be 5 V for logic high before it is fed to D3 input of Xbee transmitter since it has logic high at 3.3 V. Similarly, in rescuer unit, 9 V is fed directly to timer circuit, and is regulated to 3.3 V by UA78M33C before used as VCC of Xbee receiver. In addition, the 9 V VCC is regulated to 5 V by LM7805 before used as VCC for 74LS04 inverter. The detail connections of UA78M33C and LM7805 voltage regulators are shown in Figure 4 and Figure 5 respectively. The circuit in Figure 4 is designed according to the data sheet of LD1117 voltage regulator, which is used in original design but substituted by UA78M33C due to availability. The connection in Figure 5 is based on data sheet of LM7805. Capacitors specified in the schematics are used to avoid voltage ripples.

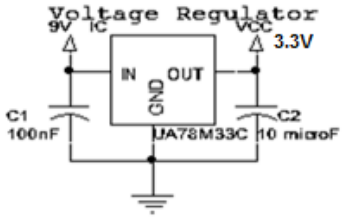

Figure 4. Connection of UA78M33C [7].

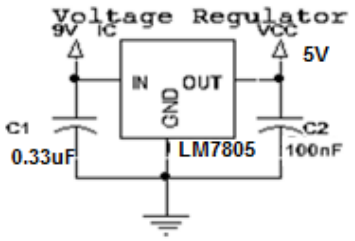

Figure 5. Connection of LM7805 [8].

Based on the detailed schematics, the battery life is estimated according to the power consumption of the system. Specifically, the 9 V battery utilized in our final design has a capacity of 565 mAh [9]. The energy a battery can provide is calculated in Equation (2.4).

$$E_{battery} = Capacity \times V_{out} = 565\text{mAh} \times 9\text{V} = 5.085\text{Wh} \tag{2.4}$$

In swimmer unit, control switches with indicating LEDs, pulse sensor and microcontroller are working under no alarm condition. According to the producer, the pulse sensor has around 3 mA at 5 V [10]. According to reference, the operating current of Arduino is about 25 mA [11]. Therefore the power consumption is calculated in Equation (2.5).

$$\begin{aligned} P_{no\_alarm} &= \sum VI \\ &= 5\text{V} \times 3\text{mA} + 25\text{mA} \times 9\text{V} + (120\Omega + 330\Omega) \times (25\text{mA})^2 + 2 \times 2\text{V} \times 25\text{mA} \\ &\approx 601.25\text{mW} \end{aligned} \tag{2.5}$$

When alarm is on, XBEE transmitter, alarm buzzer and LED array are all on. Based on test measurements, the total power consumption is calculated in Equations (2.6) – (2.9).

$$P_{LED} = 4 \times [(25\text{mA})^2 \times 39\Omega + 2 \times 25\text{mA} \times 2\text{V}] \approx 497.5\text{mW} \tag{2.6}$$

$$P_{Buzzer} = 5\text{V} \times 2\text{mA} + (2\text{mA})^2 \times 120\Omega = 490\text{mW} \tag{2.7}$$

$$P_{Xbee} = 3.3\text{V} \times 215\text{mA} = 709.5\text{mW} \tag{2.8}$$

$$\begin{aligned} P_{full\_function} &= P_{no\_alarm} + P_{LED} + P_{Buzzer} + P_{xbee} \\ &= 601.25\text{mW} + 497.5\text{mW} + 490\text{mW} + 709.5\text{mW} \\ &= 2.298\text{W} \end{aligned} \tag{2.9}$$

Assuming the alarm system is on with 9 V input, the battery life is calculated in Equation (2.10).

$$t = \frac{E_{battery}}{P_{full\_function}} \approx 2.21\text{h} \tag{2.10}$$

Similarly, Equations (2.11) – (2.15) show the battery life calculation for the rescuer unit under same assumptions.

$$P_{LED} = 3 \times [(25\text{mA})^2 \times 120\Omega + 3 \times 25\text{mA} \times 2\text{V}] \approx 675\text{mW} \tag{2.11}$$

$$P_{Buzzer} = 9\text{V} \times 0.65\text{mA} + (0.65\text{mA})^2 \times 10k\Omega = 10.075\text{mW} \tag{2.12}$$

$$\begin{aligned} P_{no\_alarm} &= P_{xbee} + P_{timer} + P_{switchcircuit} \\ &= 709.5\text{mW} + 9\text{V} \times 6\text{mA} + 2 \times 330 \times (25\text{mA})^2 + 2 \times 2\text{V} \times 25\text{mA} \\ &= 1.276\text{W} \end{aligned} \tag{2.13}$$

$$P_{full\_function} = P_{LED} + P_{Buzzer} + P_{no\_alarm} = 675\text{mW} + 10.075\text{mW} + 1.276\text{W} = 1.961\text{W} \tag{2.14}$$

$$t = \frac{E_{battery}}{P_{full-function}} \approx 2.59\text{h} \tag{2.15}$$

In general, the batteries can support the system work with alarm on for more than 2 hours, which is long enough for one triathlon swimming competition, although the power consumption of the finalized design is larger than the estimation of the originally proposed design.

2.2.2 Pulse Sensor

An optical sensor detecting blood pulse is used to collect heart rate information for Arduino. As shown in Figure 2 on page 3, the three pins of the sensor are 5 V Power (Red), Ground (Black) and Signal Output (Purple), and they are connecting to 5 V, GND and A0 (Analog Input 0) on Arduino Uno respectively. Once the sensor is powered on, it will transmit pulse signal to pin A0 when the circle sensor is contacting

people's skin. The small sensor is supposed to be insulated by hot glue for safety. In addition, the sensor can be clipped on ear easily without influencing swimmer's movement.

2.2.3 Manual Switch

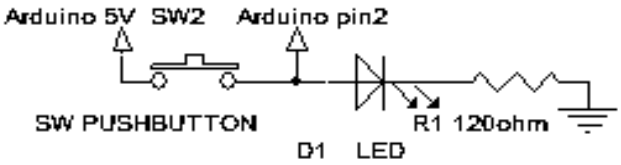

Figure 6. Schematic of Manual Switch.

As specified in Figure 6,a SPST pushbutton switch is connected between Arduino 5 V and Arduino pin 2. When it is pushed, the circuit is connect and pin 2 will received high input. The circuit will be open and pin 2 will get low input if the button is pressed again. Since the pushbutton switch looks the same when it is on and off, we add a yellow LED in series with a resistor between pin 2 and ground to indicate whether the switch is on or off. Equation (2.16) shows the resistance calculation.

$$R = \frac{V_{in} - V_{drop}}{I} = \frac{5\text{V} - 2\text{V}}{0.025\text{A}} = 120\Omega \tag{2.16}$$

The data of 2 V voltage drop and 25 mA forward current are based on the test results such that LEDs will light obviously in such condition.

2.2.4 Microcontroller

An Arduino Uno in swimmer unit is used to control the alarm system. The inner connections and mapping of pins from microcontroller ATmega328 to pins on Arduino board is specified in Figure A.3. The circuit connection of Arduino is specified in the schematic Figure A.1. Specifically, it accepts an analog pulse signal at pinA0 and a digital input from manual switch. These two inputs are analyzed and processed in Arduino to generate an alarm signal output according to the code in Appendix D. In detail, an open source test code provided by pulse sensor producer is revised and utilized to convert pulse signal from sensor to heart rate value [12] . Specifically, the interrupt function is used to update BPM value every 10 pulses. Several boolean data named as "firstbeat" and "secondbeat" are intialized as true. They cause direct return at the first time, then are called in program, thus they create empty loops to avoid inaccurate data of the first several heart beat. In the following programming, the BPM value is compared with preset thresholds and Arduino outputs will go high when either input from mannual switch is high or the BPM value is out of range. The same high or low output will be fed to LED array, buzzer and XBEE transmitter (through a 3.3 V voltage regulator) at pin 8,4,3 respectively.

2.2.5 Xbee Modules

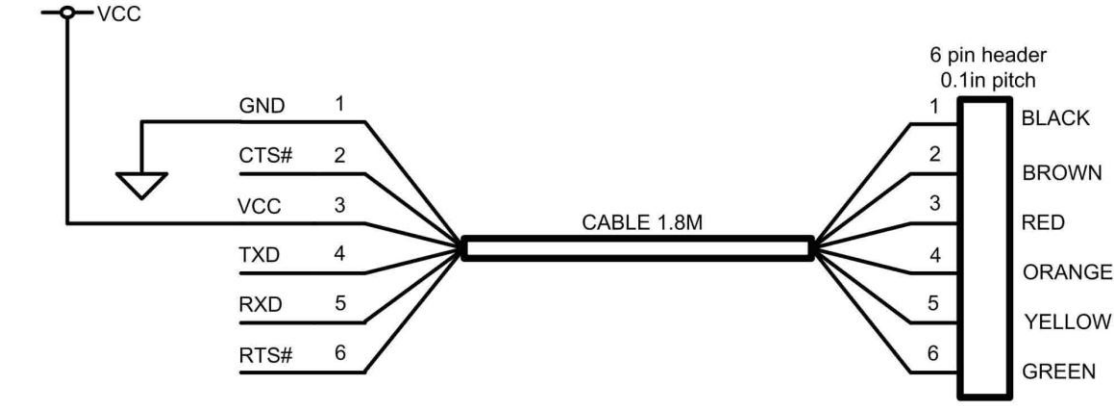

Figure 7. Schematic of FTDI [13].

Two Xbee Pro modules are used to build a wireless transceiver system to enable the rescuer unit to respond the alarm signal from swimmer unit. As specified in Schematics (Figure A.1 and A.2), two 10-pin headers are used for each Xbee module to avoid direct soldering on them. A FTDI connector is used to connect Xbee modules to computer where they are initially configured. The specific connection is showed in Figure 7, the 6 wires on FTDI are connected to pin 10 (GND), 12 (CTS#), 1 (VCC), 3 (DIN), 2 (DOUT) and 16 (RTS#) respectively. Using X-CTU software, two modules are set with same channel, panID and baud rate and opposite address parameters to make sure they are talking to each other. D3 is chosen as the input/output pin such that the transmitter receives Arduino output at its pin D3 and this value will determine the D3 output of receiver module through wireless transmission. Moreover, in order to have stable functions, according to default setting specified in datasheet [5], the pin NOT RESET at each module is connected to VCC with a 10 kΩ pull-up resistor and the VREF pin is connected to VCC.

2.2.6 LED Arrays

A LED array is implemented in each unit to indicate alarm with obvious red light. In swimmer unit, the LED array is controlled by input from Arduino pin8 which will be at 5 V logic high when the system is triggered. In the original design, each column of LED array needs a 30 Ω resistor in series. Equation (2.17) shows the calculation of the resistance 1.2 V voltage drop and 50 mA forward current [14].

$$R = \frac{V_{in} - n \times V_{drop}}{I} = \frac{5\text{V} - 3 \times 1.2\text{V}}{0.005\text{A}} = 28\Omega \approx 30\Omega \tag{2.17}$$

However, according to test result, the voltage drop and forward current of the LED at desired behavior are 2.0 V and 25 mA respectively. Therefore 5 V cannot support 3 LEDs in series to work appropriately at same time. Since amplifying the voltage will increase complexity of the design, we change the implementation of the array to be 2x4, and the resistance calculation is shown in Equation (2.18).

$$R = \frac{V_{in} - n \times V_{drop}}{I} = \frac{5\text{V} - 2 \times 2\text{V}}{0.025\text{A}} = 40\Omega \tag{2.18}$$

In actual implementation, four 39 Ω resistors are used due to the availability.

For rescuer unit, the originally proposed 3x3 LED array receives 9 V high input from timer circuit. Equation (2.19) shows the resistance calculation with same initial assumptions.

$$R = \frac{V_{in} - n \times V_{drop}}{I} = \frac{9\text{V} - 3 \times 1.2\text{V}}{0.005\text{A}} = 108\Omega \approx 100\Omega \tag{2.19}$$

Equation (2.20) shows the resistance calculation in actual implementation with revised connection.

$$R = \frac{V_{in} - n \times V_{drop}}{I} = \frac{9\text{V} - 3 \times 2\text{V}}{0.025\text{A}} = 120\Omega \tag{2.20}$$

The final design is specified in Schematics (Figure A.1 and A.2) in Appendix A.

2.2.7 Buzzer Alarm

Both units contain a MC320B2 buzzer which will alarm loudly when the system is triggered. In swimmer unit, the connection of buzzer is specified in Figure 8.

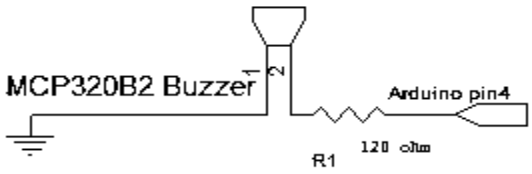

Figure 8. Connection of Buzzer in Swimmer Unit.

When Arduino pin4 outputs high at 5 V (±0.5 V), the buzzer will alarm to notify surrounding swimmers and when pin4 goes low at 0 V (±0.3 V), the buzzer will be off. A 120 Ω resistor is connected in series with the buzzer instead of a proposed 100 Ω due to availability.

In rescuer unit, the connection of buzzer is specified in Figure 9.

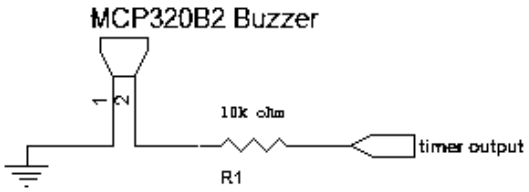

Figure 9. Connection of Buzzer in Rescuer Unit.

This buzzer receives input from timer pin3, and alarms when timer outputs high at 9 V (±0.5 V) and be off when timer outputs low at 0 V (±0.3 V). A 10 KΩ resistor is connected according to test data to support appropriate alarm behavior.

2.2.8 Timer Circuit

A timer circuit, originally called “alarm circuit” in proposed design, is implemented in the rescuer unit to extend the pulse alarm signal received from XBEE receiver, such that the LEDs and buzzer in rescuer unit

can be turned on for time that long enough to get attention from rescuers, even the received signal is a small pulse. The circuit design shown in Figure 10 is based on datasheet of NE555 timer [6], we finalized the design to use LM555 timer due to the availability of parts in the part shop. The test results verify that the revised circuit worked well with the same configuration. The output of XBEE receiver is fed to inverter and the signal at pin3 of timer is fed to LED array and buzzer.

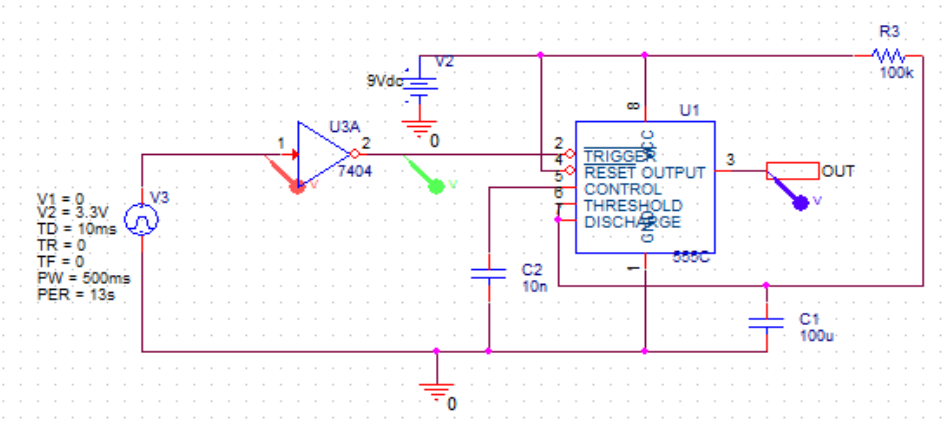

Figure 10. Timer Circuit [6].

The prelimanry simulation is done by PSPICE and the result is showed in Figure 11.

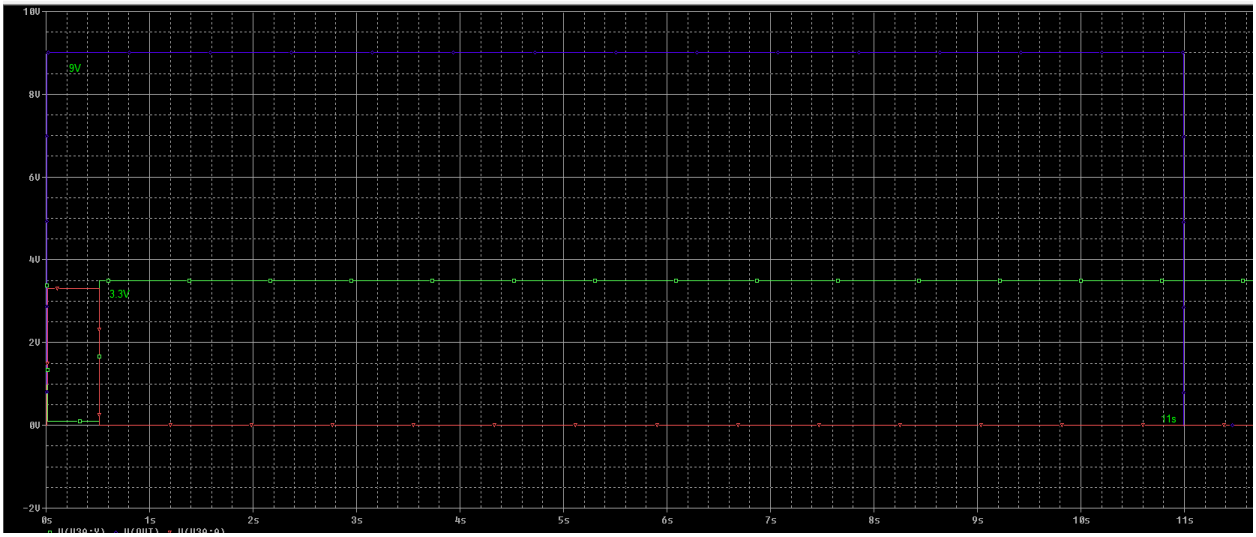

Figure 11. Timer Circuit Simulation.

Red signal is the XBEE output pulse while the green signal is inverted pulse. The resulting purple signal indicates that the pulse is extended to 11 s, corresponding to the result of preliminary calculation based on reference tutorial [15] shown in Equation (2.21).

$$t = 1.1RC = 1.1 \times 100\text{k}\Omega \times 100\mu\text{F} = 11\text{s} \qquad (2.21)$$

Additional revision is done on the timer circuit such that two pushbutton switches are added as specified in rescuer unit schematic shown in Figure A.2. The first switch is connected between 9 V power

input from battery and 9 V input to two voltage regulators. Therefore when it is pressed, it will turn on the Xbee module which gets 3.3 V VCC from UA78M33C and 74LS04 inverter which gets 5 V VCC from 78LM05. The second switch is connected between the first switch and the VCC for LM555 timer such that the timer will be turned on only after this switch is pressed. A yellow and a green LED with 330 Ω resistors in series are added into the circuit to indicate the switch behavior. The calculation of resistance is done in Equation (2.3) on page5. This revision is necessary to avoid the error high output generated at condition that the timer circuit receives initial low input while the inverter and timer are turned on at the same time. Assuming an initial low input is given from Xbee receiver, the timer input will also be low when inverter is off (i.e. no voltage inverting) and the voltage will be flip once it is turned on. If the timer is turned on at the same time of inverter, it will capture this voltage flip and will generate an 11 second high output. Simulation result in Figure 12 corresponds to this error condition such that when input to timer(green line) is low, the timer will generate high output (purple line).

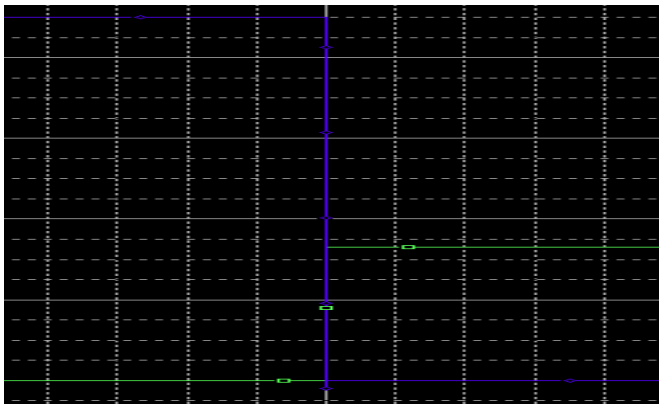

Figure 12. Simulation of Timer Circuit Error.

Therefore we need sperate control switches to make sure the inverter is turned on before the timer to get expected result, which is shown in Figure 13 where red, green and purple lines refer to Xbee output, inverter output and timer output respectively.

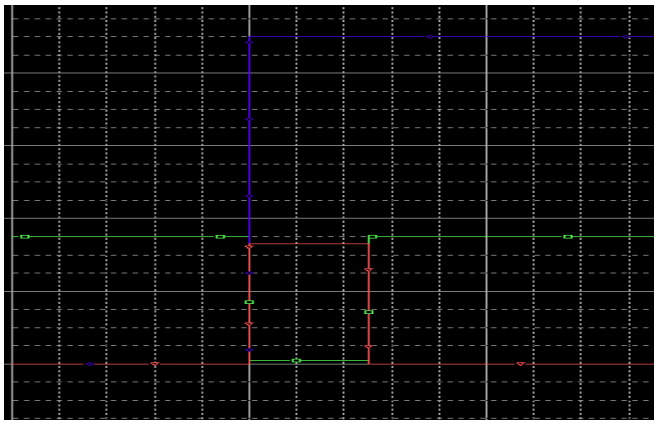

Figure 13. Simulation of Ideal Timer Circuit Performance.

3. Design Verification

The detail requirements and verification procedures are spepcified in Table C.1 in Appendix C.

3.1 Power Supply

To test the appropriate input range of Arduino, we uploaded a simple test code of blinking LED [16] to the board. Power supply was used to simulate battery voltage connected to Arduino pin Vin and voltage output at pin 13 was measured by multimeter. Specifically, when Vin is supplied with 6 V, 7 V and 9 V, the LED on board corresponding to pin 13 blinked correctly. The logic high at pin 13 was measured to be 4.975 V while the logic low is around 0.003 V. Therefore the Arduino board is verified to work appropriately when power input is in the range from 6 V to 9 V, where 9 V is the ideal voltage input supplied by batteries in our design and 6 V is the low limit for input voltage as specified in its datasheet [1]. The high and low outputs are within the required ranges of 5 V (±0.5 V) and 0 V (±0.3 V) respectively.

To verify the function of voltage regulator, we built the circuit as stated in Figure 4 on page 6. Power supply was used to provide voltage input from 0 V to 9 V for UA78M33C and the detail test results are specified in Table 1.

Table 1 UA78M33C Voltage Regulator Test Results

VCC (Power Supply)	0 V	1 V	2 V	3 V	4 V	5 V	9 V
Vout	0.0001 mV	0.01 mV	0.05 mV	2.42 mV	3.20 V	3.29 V	3.29 V

According to the test results, when input voltage is smaller than 3 V, the voltage regulator gives low output. For 5 V and 9 V inputs, which are required in our design, the outputs are all at 3.29 V which fulfill the requirement at 3.3 V (±0.5 V).

3.2 Pulse Sensor

Function of the pulse sensor was tested by the open source test code provided by sensor producer [12]. After loading the code into Arduino, the 3 pins of pulse sensor were plugged into the corresponding pins on Arduino which is powered with 9 V at pin VIN. The other side of the sensor was placed on finger and the LED on Arduino for pin 13 blinked corresponding to the heart beat appropriately when finger pressed onto the sensor with a moderate strength. There were interrupts and abnormal blinks when the finger was initially placed on sensor and removed from it, which indicated that the detected sensor output signal for the first several beats may not be accurate. Therefore in Arduino programming, the first several pulses are discarded and the BPM value will not be used until the pulse signal become stable.

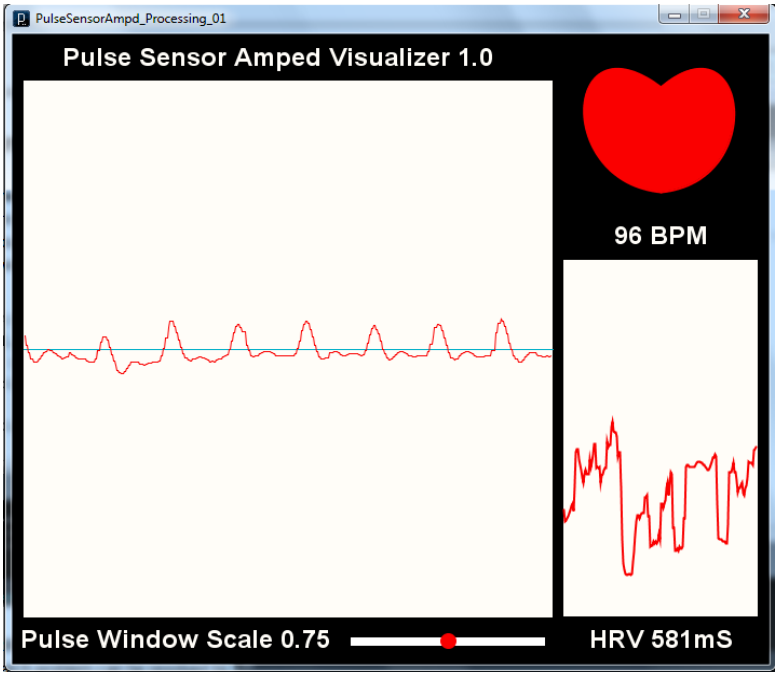

Figure 14. Visualizer Window of Pulse Sensor Signal.

The test result was also visualized via software processing as Figure 14 shows, utilizing the code from sensor producer [17]. The stable wave in the main window verified that the pulse sensor could provide stable heart beat information to Arduino. HRV showed in right lower corner referred to time interval between last two pulses detected and is used in calcualting heart rate. The continuous updated BPM value visualized in the right upper window would be analyzed in Arduino program to determine alarm signal.

3.3 Switch Board

In actual implementation, we integrated the circuits of overall control switch for the swimmer unit and the manual switch onto a single small PCB board. The board accepts a 9 V from battery and a 5 V VCC from Arduino as the inputs of two switches respectively, and their outputs are fed to the swimmer unit board as 9 V VCC and pin 2 input. The detail test result of the switch board is shown in Table 2.

Table 2 Switch Board Test Results

Test Procedure	Output voltages	LED behaviors
1.Provide overall switch 9V input	0.02 mV for both switches	No LED lights up
2.Press overall switch	8.99 V for overall switch	Green LED lights up
3.Provide manual switch 5V input; press manual switch	4.99 V for manual switch	Yellow LED lights up
4. Press manual switch again	0.03 mV for manual switch	Yellow LED off
5. Press overall switch again	0.03 mV for overall switch	Green LED off

The results verified that the SPST switches could appropriately control the circuit. When switches were not pressed, the circuit was open, and the requirement of low outputs at 0 V (±0.3 V) was satisfied. When either the switch was pressed once, the corresponding output turned high satisfying either 9 V (±0.5 V) or 5 V (±0.5 V) requirements. When either the switch was pressed again, the output returned to low. The accompanying LEDs also functioned appropriately to light up only when corresponding switch was on.

3.4 Microcontroller

To test the function of Arduino Uno, we simulated different heart rate behaviors by loading different range thresholds in program. Since it is hard to generate abnormal heart rate in actual test, we set a BPM range from 0 to 200 ensuring that test heart rate would be within the range to simulate normal behavior, and then we change the range to 0-20 such that the test heart rate would fall out of the range to simulate abnormal behavior. Connecting Arduino board as specified in Figure A.1, we test the program function with different heart rate ranges and manual switch status by measuring outputs at pin3, 4 and 8 using multimeter. The results are showed in Table 3.

Table 3 Arduino Test Results

Preset BPM Range	Manual Switch (pin2)	Vout at pin3,4,8
0-200 (Normal)	Off (Low)	3.77 mV
0-200 (Normal)	On (High)	4.97 V
0-20 (Abnormal)	Off (Low)	4.98 V

The test results verified that the Arduino could appropriately trigger the system when either the heart rate was out of range or the manual switch was on. The outputs satisfying the requirement with logic high at 5 V (±0.5 V) and logic low at 0 V (±0.3 V) at pin3, 4 and 8.

3.5 Xbee Modules

To test Xbee transceiver system, the modules were firstly connected to computer via FTDI cable with connection specified in Figure 7. Using X-CTU software, the two modules were configured based on tutorials with parameters specified in Table 4.

Table 4 Xbee Configurations

	Channel	PanID	ATMY	ATDL	ATBD	D3	IU
Transmitter	C	3137	10	11	6	DI	/
Receiver	C	3137	11	10	6	Do Low	Disabled

According to one online tutorial [18], the two modules were set with same channel, panID and baud rate (ATBD). The PanID was changed to a random number from the default value to make sure no other working modules in the lab would affect our system. ATMY and ATDL referring to the source address and destination address were set as showed in Table 4 to make sure two modules talk to each other.

According to another tutorial [19], the D3 pins were set as input/output pin. Do Low for receiver indicated that D3 output at receiver will be default low without a transmitter. The setting of IU parameters ensured that D3 output of receiver would be determined by D3 input of transmitter.

Then we powered up two Xbee modules as specified in Figure A.1 and A.2. The output at receiver D3 was measured using multimeter with different D3 inputs at transmitter. The detail test results are showed in Table 5.

Table 5 Xbee Test Results

D3 Input of Transmitter	No transmitter	GND	0 V - 1.7 V	1.8 V	3.3 V(VCC)
D3 Output of Receiver	Low (Default)	3.72 mV	Low	High	3.28 V

According to Table 5, the function of Xbee modules are verified that when transmitter received input low (GND) at D3, the receiver would output low satisfying 0 V (±0.3 V) requirement; when transmitter received input high (3.3 V) at D3, the receiver would output high satisfying 3.3 V (±0.5 V) requirement. The test results also showed that the low input tolerance could increased to 1.7 V since when transmitter input was between 0 and 1.7 V the receiver would always output low. When the transmitter was not functioning, the D3 output of receiver would be low corresponding to the setting in Table 4.

3.6 LED Arrays

For LED array in swimmer unit, tests were firstly performed based on the original 3x3 design. The initial test results were showed in Table 6.

Table 6 Initial Test Results for Swimmer Unit LED Array

Vin (Power Supply)	**Current in Each Column**	**LEDs Observation**	**Voltage Drop for Single LED**
0 V - 4.6 V	/	Very dim	/
5 V	0.002 A	Visible from above in daylight	1.66 V
5.5 V	0.025 A	Obvious in daylight	1.99 V

According to the results of initial tests, the original design failed since the ideal 5 V input could not support appropriate LED performance. Therefore, the forward current of 0.025 A and voltage drop around 2 V stated in Table 6 which were measured when the LED produced satisfying light intensity were used in all calculations and revisions relevant to LED in our final design. Specifically swimmer unit array was revised into 2x4 layout. Rescuer unit was revised using 120 Ω. The Tests then performed with revised circuit shown in Figure A.1 and A.2. Table 7 gives the results of final tests.

Table 7 Revised LED arrays Test Results

Swimmer Unit (2x4 array)		**Rescuer Unit (3x3 array)**	
Vin	**LEDs Observation**	**Vin**	**LEDs Observation**
0 V – 4.4 V	Off	0 V – 4.6 V	Off

Table 7 (continued)

4.5 V	Start to light up	4.7 V	Dim
5 V	Obvious in daylight	8.5 V	Obvious in daylight
5.5 V	Obvious in daylight	9 V	Obvious in daylight
/	/	9.5 V	Obvious in daylight

According to Table 7, the revised design satisfied the requirements such that when input voltages were low at 0 V (±0.3 V), the LEDs were off; when input votages were high at either 5 V (±0.5 V) or 9 V (±0.5 V) for swimmer unit and rescuer unit respectively, the LEDs could provide obvious light display without being burned out.

3.7 Buzzer Alarm

The buzzers in swimmer unit and rescuer unit were tested with connections specified in Figure 8 and Figure 9 on page 10 respectively. To satisfy the requirements such that the buzzers should be off when input voltages are low at 0 V (± 0.3 V) and alarm loudly when input voltages are high at 5 V (± 0.5 V) for swimmer unit and 9 V (± 0.5 V) for rescuer unit, the resistors in series were determined according to the test results showed in Table 8.

Table 8 Buzzer Circuits Test Results

Unit	Resistor in Series	Vin	Buzzer Performance
Swimmer	120 Ω	0 V – 0.6 V	Off
		0.7 V	Audible
		4.5 V-5.5 V	Noisy
Rescuer	10 kΩ	0 V – 0.9 V	Off
		1V	Audible
		8.5 V – 9.5 V	Noisy
	20 kΩ, 40 kΩ,100 kΩ	9 V	Not loud enough

3.8 Timer Circuit

Initial tests were performed with original design circuit specified in Figure 10. The circuit was required to provide consistent high timer output for 5.5 s once a short high input 3.3 V was provided by Xbee receiver. However, the test failed such that once 9 V VCC of circuit was turned on, without any input at to circuit, there is a small period corresponding to correct delay time that timer gave high output to light LEDs, although the circuit worked appropriately after this initial error.

Circuit is then revised to double the delay time. The initial error problem was solved based on analysis related to simulations in Figure 12 and Figure 13 on page 12. Then the tests were performed based on final design specified in Figure A.2. Specifically, the timer produced output at pin 3 with 4.35 mV for logic low and 8.7 V for logic high. Once provided 3.3 V for 3 seconds as input, the circuit outputs high at

pin3 would show on oscilloscope for about 12 seconds before disappeared, which satisfied the requirement of extend short pulse input to 11 s (±1 s).

3.9 Complete System

The overall requirement for the complete system is that the two units should work efficiently when they are 25 m (± 5m) away from each other, while the swimmer unit is placed underwater. This function is verified by our test result. When swimmer unit was placed in waterproof case and placed underwater, manual switch was turned on to trigger the swimmer unit where LED arrays provided obvious light and buzzer provided audible alarm underwater, although the loudness was decreased compared with that in air. It turned out that the rescuer unit worked efficiently with light LEDs, loud buzzer and correct delay time when it was placed 60 m away from the swimmer unit.

4. Costs

4.1 Parts

Table 9 Cost of Parts

Part	Manufacturer	Quantity	Retail Cost ($)	Bulk Purchase Cost ($)	Actual Cost ($)
Pulse Sensor	Pulsesesnor.com	1	25	25	25
Waterproof case	Snapway	1	3.8	3.8	3.8
SPST On/Off Pushbutton	All Electronics	6	1.35	1.35	8.1
XBEE PRO 802.15.4	Digi International	2	38	38	76
Arduino UNO	Arduino	1	21	21	21
Voltage Regulator UA78M33C	Texas Instruments	3	0.525	0.356	1.575
Voltage Regulator LM7805AC	Fairchild Semiconductor	1	0.568	0.475	0.568
9V Battery	Duracell	2	1.5	1.33	3
LEDs	Kingbright	21	0.65	0.42	8.82
Buzzer MCP320B2	Mallory	2	3.42	3.42	6.84
LM555CN Timer	Fairchild Semiconductor	1	0.375	0.375	3.375
SN74LS04N invertor	Texas Instruments	1	0.602	0.602	0.602
Resistors	Vishay	15	0.05	0.05	0.75
Capacitors	Kemet	11	1	1	11
Total					**170.43**

4.2 Labor

Table 10 Cost of Labor

Name	Rate	Hours	Total = Rate x 2.5 x Hours
Zilin Dou	$40/hr	150	$15000
Yunye Gong	$40/hr	150	$15000
Total			$30000

4.3 Grand Total

Table 11 Cost of Grand Total

Labor	$30000
Parts	$170.43
Grand Total	$30170.43

5. Conclusion

5.1 Accomplishments

The project was able to work as expected with three PCB boards performing all required functions successfully. All three boards were well-designed with tidy layouts and small sizes to achieve portability. The system was able to provide effective alarm in response to real-time heart rate behavior and manual request. In addition, the efficient wireless communication led to our greatest success such that the system was able to function properly when swimmer unit was underwater and rescuer unit was 60 meters away. This distance is more than twice of the proposed value.

5.2 Uncertainties

For the distance test, we started from 25 meters and kept increasing the distance up to 60 meters. Since the two units were able to work efficiently 60 meter apart and no test was performed with larger distance, there is no accurate data of the maximum distance that the project could work with.

5.3 Ethical considerations

During the whole experiment process, we have abided following the IEEE Code of Ethics [20]

3. to be honest and realistic in stating claims or estimates based on available data;

All estimating calculations were conducted based on formulas and values from datasheet with proper citation. Claims about system efficiency and functioning distance were made according to real test data.

7. to seek, accept, and offer honest criticism of technical work, to acknowledge and correct errors, and to credit properly the contributions of others;

During the whole semester, we got advises from Professors, TA and the review teams to improve our project. We made necessary changes to our original design for proper functions. Working as a two people group, we didn't hesitate to point out each other's mistakes to make sure we were in the correct direction.

9. to avoid injuring others, their property, reputation, or employment by false or malicious action;

We use waterproof case to insulate swimmer unit to avoid electric shock since it will work underwater.

5.4 Future work

There is still room for improvements for our project. Firstly the system can be implemented with professional surface mount boards instead of through hole boards for reduce size and weight. A more suitable waterproof case for swimmer unit will also be helpful considering practical use. Arduino programming can also be improved to enrich the system with more functions, such as detecting sudden heart rate changes. Adding positioning device into the project can also be a good choice such that it will help the rescuer to locate the swimmer triggering the alarm in a faster and easier way.

5. References

[1] "Arduino Uno," [Online]. Available: http://arduino.cc/en/Main/ArduinoBoardUno. [Accessed 28 9 2012].

[2] "HEX INVERTERS," Texas Instruments, 1 2005. [Online]. Available: http://www.ti.com/lit/ds/symlink/sn74ls04.pdf. [Accessed 10 12 2012].

[3] "Pulse Sensor Getting Started Guide," [Online]. Available: https://docs.google.com/document/d/1iOZv-ubb-cbfhLEYUawFpGXLxOGqULidrHE5UD5vx9s/edit. [Accessed 28 September 2012].

[4] "How and Why to avoid delay()," Arduino, [Online]. Available: http://playground.arduino.cc/Code/AvoidDelay. [Accessed 10 12 2012].

[5] "XBee™/XBee-PRO™ OEM RF Modules," [Online]. Available: http://www.libelium.com/squidbee/upload/3/31/Data-sheet-max-stream.pdf. [Accessed 28 9 2012].

[6] "NE555 General Purpose Single Bipolar Timers," [Online]. Available: http://www.datasheetcatalog.org/datasheet/SGSThomsonMicroelectronics/mXvzqv.pdf. [Accessed 28 9 2012].

[7] "Low Drop Fixed And Adjustable Positive Voltage Regulators," [Online]. Available: http://www.datasheetcatalog.org/datasheet/SGSThomsonMicroelectronics/mXuqtqv.pdf. [Accessed 28 9 2012].

[8] "LM78XX/LM78XXA 3-Terminal 1A Positive Voltage Regulator," FAIRCHILD SEMICONDUCTOR, 8 2012. [Online]. Available: http://www.fairchildsemi.com/ds/LM/LM7805.pdf. [Accessed 10 12 2012].

[9] "DURACELLl-PC1604BKD-ALKALINE MN02 BATTERY, 9V," [Online]. Available: http://www.newark.com/duracell/pc1604bkd/alkaline-mno2-battery-9v/dp/88M0921. [Accessed 10 12 2012].

[10] "Battery Life on Pulsesensor?," [Online]. Available: http://pulsesensor.proboards.com/index.cgi?board=allaboutatoms&action=display&thread=6. [Accessed 5 10 2012].

[11] "Power Consumption Arduino," [Online]. Available: http://arduino.cc/forum/index.php/topic,5536.0.html. [Accessed 28 9 2012].

[12] "Latest Arduino code for Pulse Sensor Amped," [Online]. Available: http://pulsesensor.myshopify.com/pages/code-and-guide. [Accessed 9 10 2012].

[13] "Setting up an Xbee Network for a Project," [Online]. Available: http://courses.engr.illinois.edu/ece445/wiki/?n=Topics.MaxstreamXbee.. [Accessed 28 9 2012].

[14] "How to make LEDs glow not blow!," [Online]. Available: http://letsmakerobots.com/node/4948. [Accessed 4 10 2012].

[15] "555 Timer Tutorial," [Online]. Available: http://www.electronics-tutorials.ws/waveforms/555_timer.html. [Accessed 4 10 2012].

[16] "Blink," [Online]. Available: http://www.arduino.cc/en/Tutorial/Blink. [Accessed 9 10 2012].

[17] "Lastest Processing code for Pulse Sensor Amped," [Online]. Available: http://pulsesensor.myshopify.com/pages/code-and-guide. [Accessed 9 10 2012].

[18] "XBEE Basics," 21 6 2009. [Online]. Available: http://forums.trossenrobotics.com/tutorials/how-to-diy-128/xbee-basics-3259/. [Accessed 13 11 2012].

[19] "Xbee Radios," 17 10 2012. [Online]. Available: http://www.ladyada.net/make/xbee/arduino.html. [Accessed 13 11 2012].

[20] "IEEE Code of Ethics," [Online]. Available: http://www.ieee.org/about/corporate/governance/p7-8.html. [Accessed 8 12 2012].

[21] "Arduino Uno," [Online]. Available: http://arduino.cc/en/uploads/Main/Arduino_Uno_Rev3-schematic.pdf. [Accessed 28 9 2012].

Appendix A System Schematics

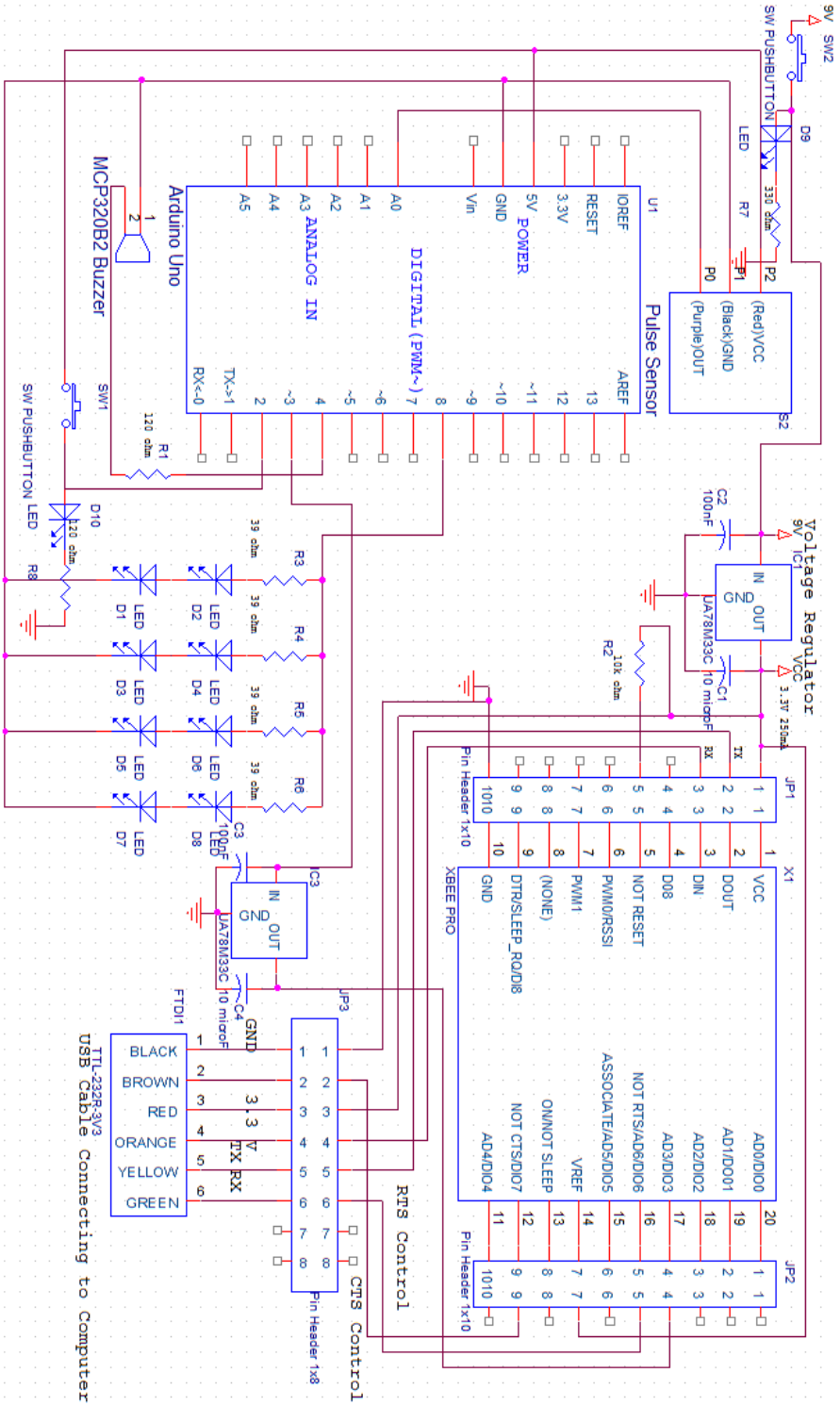

Figure A.1 Swimmer Unit Schematic.

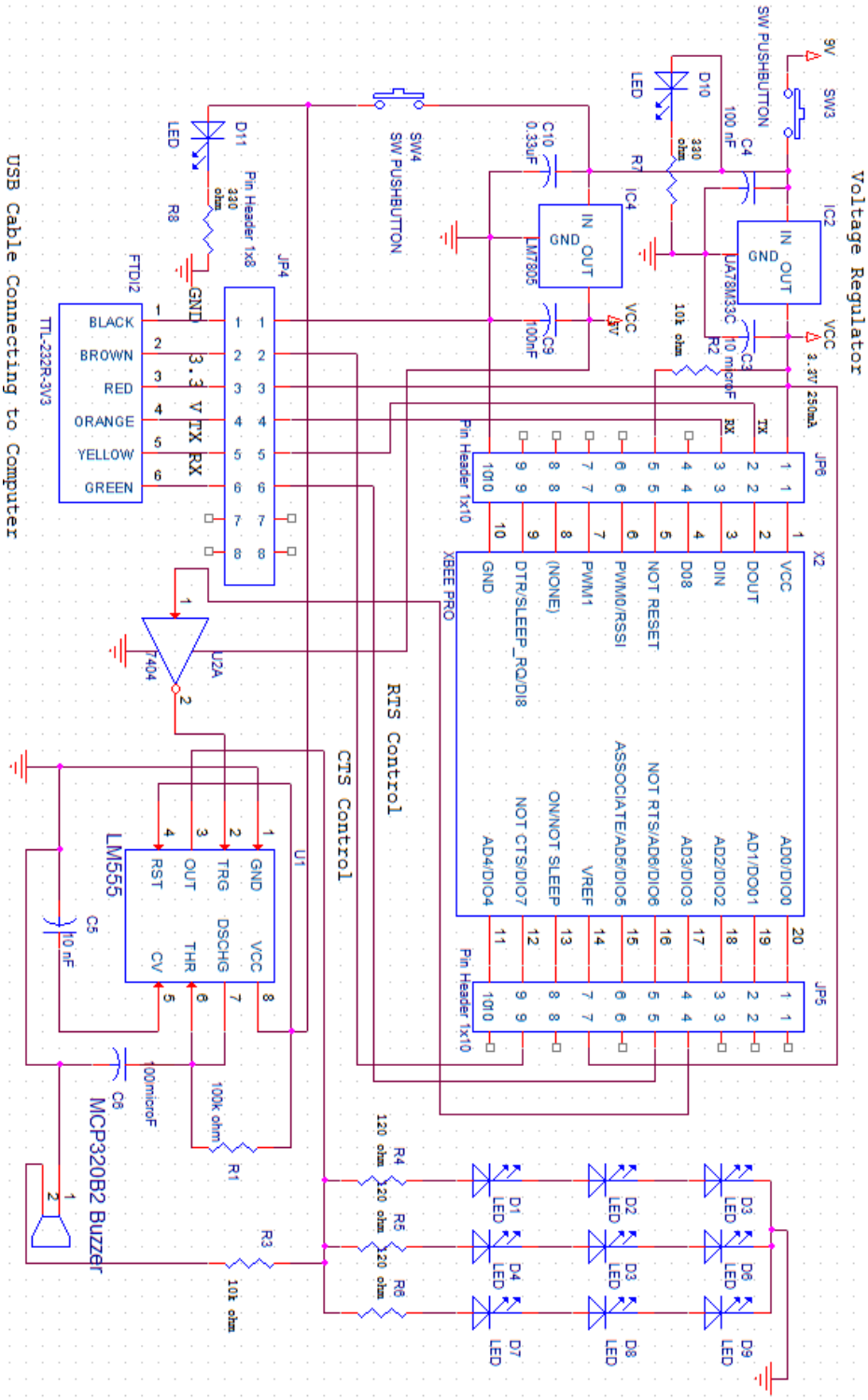

Figure A.2 Rescuer Unit Schematic.

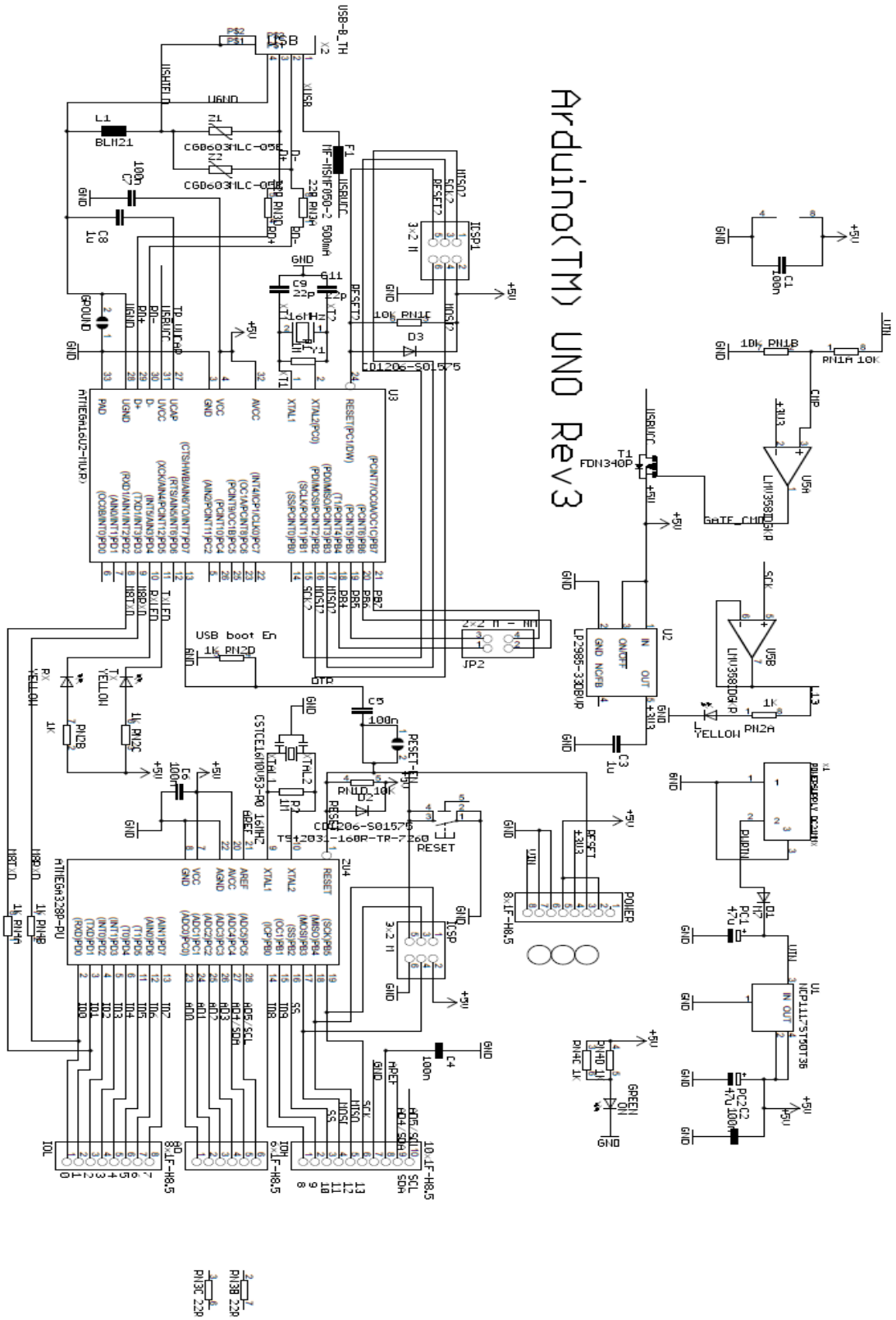

Figure A.3 Arduino Uno Schematic [21].

Appendix B Pictures

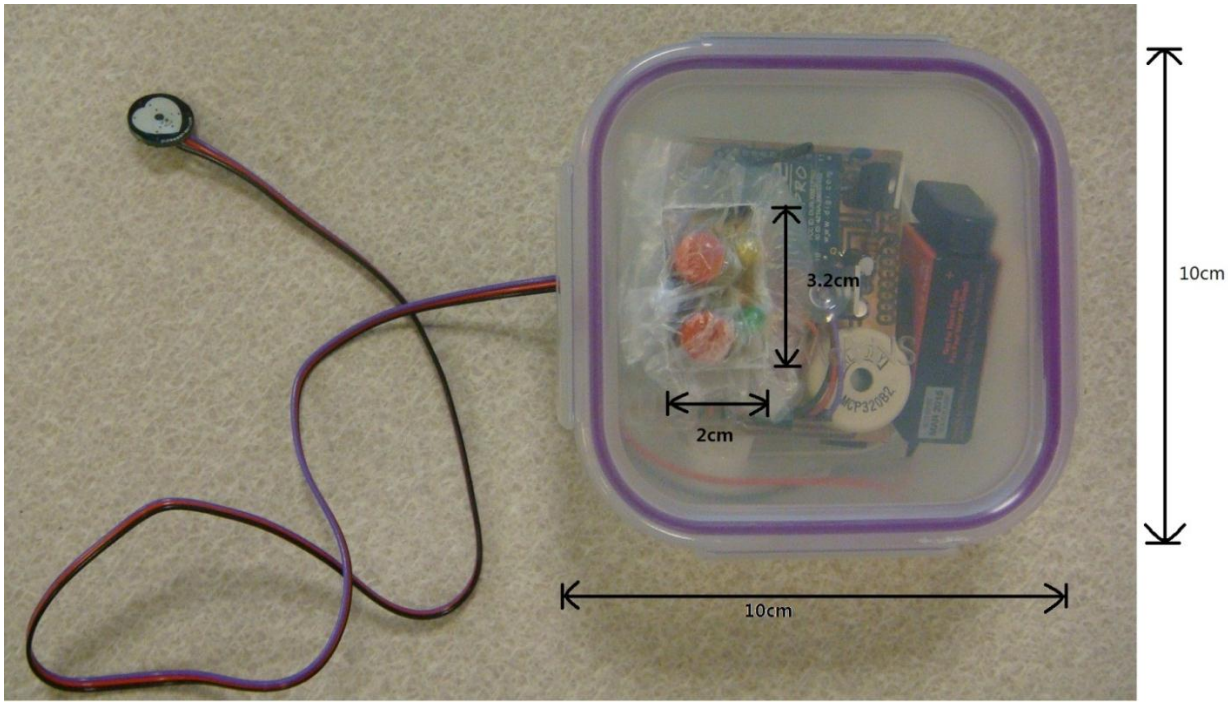

Figure B.1 Picture of Swimmer Unit Container.

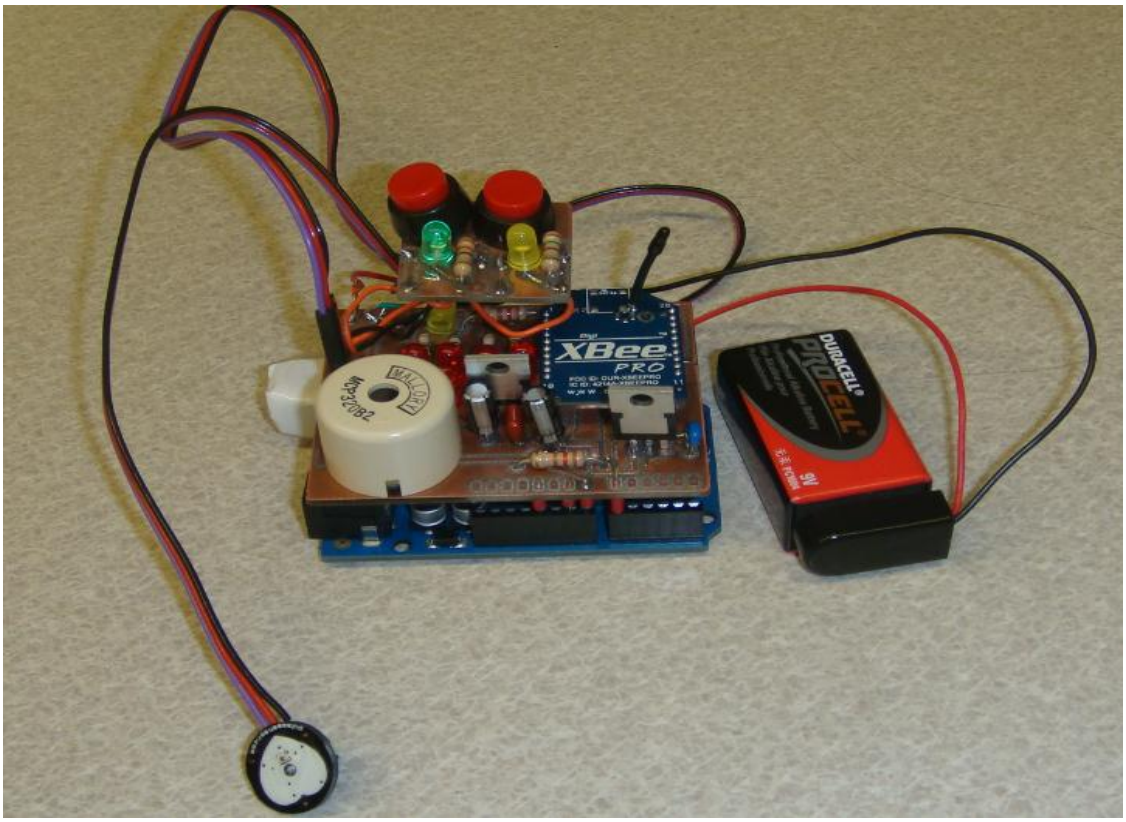

Figure B.2 Picture of Swimmer Unit Without Container.

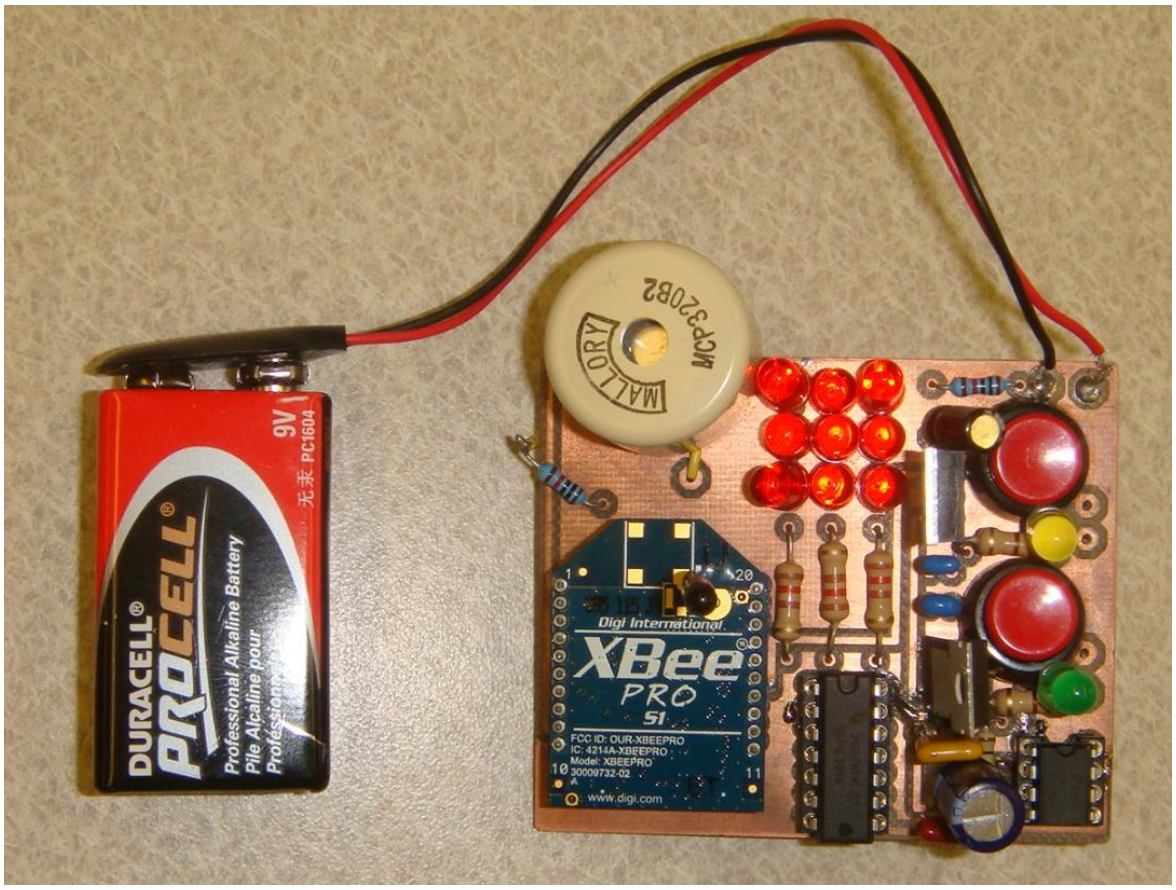

Figure B.3 Picture of Rescuer Unit.

Appendix C Requirement and Verification Table

Table C.1 System Requirements and Verifications

Block	Requirements	Verification Process	Verification status (Y or N)
Power Supply	1. The batteries need to provide a stable voltage at 9 V, with a -3 V tolerance, since 6 V is the low limit of input voltage required by arduino.	1. Measure the voltage across battery using multimeter.	Y
	2. UA78M33C voltage regulator need to convert 9 V and 5 V to 3.3 V (±0.5 V).	2. Build voltage regulator circuit on breadboard with 9 V and 5 V as VCC respectively, measure the output voltage using multimeter.	Y
	3. LM7805AC voltage regulator need to convert 9V to 5 V (±0.5 V).	3. Build voltage regulator circuit on breadboard with 9 V VCC, measure the output voltage using multimeter.	Y
Pulse Sensor	1. Pulse sensor should continuously read heart rate data to Arduino under 5 V VCC, when appropriately installed on people.	1. Connect red, black and purple pins of sensor to 5 V, GND and A0 pins on arduino respectively. Run provided test code, the pulse signal should be stable and continuous showing on the visualize window when sensor is touched by finger.	Y
	2. The provided code should be able to convert analog pulse data into BPM value used in arduino programming.	2. Run test code in Processing software; verify that pulse signals are calculated into BPM value showing in visualizer window when sensor is touched by finger.	Y
Manual Switch	1. The manual switch should output low 0 V (±0.3 V) when it is not pressed.	1. Connect the switch between 5 V VCC and GND following a 120 Ω resistor in series. Measure the voltage across the resistor using multimeter when the switch is not pressed.	Y
	2. The manual switch should output high 5 V (±0.3 V) when it is pressed.	2. Measure the voltage across the resistor using multimeter when the switch is pressed one time.	Y
	3. The manual switch should output low 0 V (±0.3 V) when it is pressed again.	3. Measure the voltage across the resistor using multimeter when the switch is pressed again.	Y

Table C.1 (continued)

Arduino Uno	1. Arduino Uno should function appropriately when voltage supply is in the range between 6 V to 9 V. The logic high output should around 5 V (±0.5 V) and the logic low output should around 0 V (±0.3 V).	1. Connect pin Vin to power supply set of 6 V, 7 V and 9 V. Verify the Arduino with a simple test code. Connect power supply to Vin and GND pins as input; verify output low and high by multimeter.	Y
	2. Arduino should control the alarm system with following functionalities:	2. Set the heart rate range to 0 – 20 BPM for abnormal heart rate and 0 – 200 BPM for normal heart rate.	Y
	2.1. When manual button switch is not pressed and heart rate data from pulse sensor is normal, the Arduino should output low 0 V (±0.3 V) at digital pin 3, 4 and 8.	2.1 Load program to Arduino with normal heart setting. Touch the sensor while remaining the switch button unpressed, verify the output at pin 3, 4 and 8 are low 0 V (±0.3 V) using multimeter.	Y
	2.2. When manual button switch is not pressed but the heart rate goes out of preset range, the Arduino should output high 5 V (±0.5 V) at digital pin 3, 4 and 8 to turn on the alarm system.	2.2 Load program to Arduino with abnormal heart setting. Touch the sensor while remaining the switch button unpressed, verify the output at pin 3, 4 and 8 are high 5 V (±0.5 V) using multimeter.	Y
	2.3 When heart rate is normal but manual button switch is pressed, the Arduino should output high 5 V (±0.5 V) at digital pin 3, 4 and 8 to turn on the alarm system.	2.3 Load program to Arduino with normal heart setting. Touch the sensor and press the switch button, verify the output at pin 3, 4 and 8 are high 5 V (±0.5 V) using multimeter.	Y
	2.4 The outputs should be able to go back low when the inputs indicate there is no need to alarm.	2.4 Following step 2.3, press the switch button again, verify the output at pin 3, 4 and 8 are low 0 V (±0.3 V) using multimeter.	Y

Table C.1 (continued)

XBEE Transmitter and Receiver	1. The XBEE module should function appropriately on an adapter with 3.3 V (±0.5 V) VCC.	1. Plug the XBee module onto the adapter, connect output pin 1-6 on adapter header to FTDI cable, the XBEE should be able to be programmed on computer.	Y
	2. When transmitter D3 pin has low input at 0 V (±0.3 V), the transmitter and receiver should not send out or receive any alarm signal. The D3 pin of receiver should be low at 0 V (±0.3 V).	2. Use power supply to give low input of 0.3 V to D3 pin of transmitter, verify receiver D3 pin outputs low 0 V (±0.3 V) using multimeter.	Y
	3. When transmitter D3 pin has high input at 3.3 V (±0.3 V), the transmitter should sends out alarm signal to the receiver. The D3 pin of receiver should be high at 3.3 V (±0.3 V).	3. Use power supply to give a high input of 3.3 V to D3 pin of transmitter, verify receiver D3 pin outputs high 3.3 V (±0.3 V) using multimeter.	Y
	4. The receiver should be able to receive signal efficiently 25 m (±5 m) away from the transmitter.	4. Put receiver 25 m away from transmitter and repeat the verification step 3.	Y
Timer Circuit	1. The LM555 chip should work appropriately with 9 V VCC.	1. Connect VCC of LM555 to power supply 9 V. Give high input 3.3 V to inverter, feed inverter output to pin2 TRG of timer, verify that the output pulse at pin3 OUT at is high 9 V (±1 V)using multimeter.	Y
	2. The circuit should not trigger the buzzer and LEDs when input from XBEE is low. Therefore output should be low 0 V (±0.3 V) when low input 0 V (±0.3 V) is fed to inverter.	2. Give 0.3 V low input to inverter using power supply, verify that the timer output at pin3 is low 0 V (±0.3 V) using multimeter.	Y
	3. The circuit should extend the high input pulse and turn on the buzzer and LED for extended time (11 s) when XBEE received alarm signal and output high 3.3 V (±0.3 V) to the timer circuit.	3. Use 100uF capacitor and 100k ohm resistor in timer circuit. Simulate pulse signal at 3.3V for 1s as input for timer circuit. Verify that the pin OUT should be a high pulse at 9V (VCC) extended to 11 s (±1 s).	Y

Table C.1 (continued)

Buzzer	1. The buzzer should not sound when input voltage is low 0 V (±0.3 V).	1. Use power supply to generate a 0.3 V input voltage, connect to buzzer in series with 120 ohm resistor, and ground the other pin of buzzer. Verify the buzzer doesn't sound.	Y
	2. The buzzer should alarm when input voltage is high at 5 V (±0.5 V) in swimmer unit.	2. Use power supply to generate input voltage at 4.5 V and 5 V, connect 120 ohm resistor to buzzer in series, and ground the other pin of buzzer. Verify the buzzer sounds loudly.	Y
	3. The buzzer should alarm when input voltage is high at 9 V (±0.5 V) in rescuer unit.	3. Use power supply to generate input voltage at 9 V connecting a 10 k ohm resistor to buzzer in series, and ground the other pin of buzzer. Verify the buzzer sounds loudly.	Y
LED arrays	1. All LEDs in two arrays should be off when input is low at 0 V.	1. Use power supply to generate input voltage at 0.3 V, verify that all LEDs are off.	Y
	2. For swimmer unit, all 8 LEDs should be on but not burned when input is high at 5 V (±0.5 V). A 39 ohm resistor is connected in series with each column.	2. Use power supply to generate input voltage at 5 V (±0.5 V), verify that all LEDs are turned on and can light consistently.	Y
	3. For rescuer unit, all 9 LEDs should be on but not burned when input is high at 9 V (±0.5 V). A 120 ohm resistor is connected in series with each column.	3. Use power supply to generate input voltage at 9 V (±0.5 V), verify that all LEDs are turned on and can light consistently.	Y
Whole System	1. Underwater test. The system should be able to work efficiently underwater.	1. Put the empty waterproof case underwater to verify that no water will leak into it before install circuit into the box. Put the swimmer unit into water, trigger the alarm and check if rescuer unit's alarm is on.	Y
	2. The complete system should be able to work when two units are 25 m (±5 m) away from each other.	2. Put the swimmer unit into water, move the rescuer unit to 25m away from swimmer unit. Trigger the alarm and check if rescuer unit's alarm is on.	Y

Appendix D Arduino Code

```
int pulsePin = 0;        // pulse sensor input; analog pin 0
int upthres = 200;      // uplimIt of heart rate in BPM
int lowthres = 0;       // lowlimit of heart rate in BPM
volatile boolean pulseselect = false;
volatile boolean switchselect = false;
// following are volatile because they are used during the interrupt!
volatile int BPM;        // used to hold the pulse rate
volatile int Signal;       // holds the incoming raw data
volatile int HRV;        // holds the time between beats
volatile boolean Pulse = false; // true for high, false for low
volatile boolean QS = false;     // becomes true when pulse rate is determined

void setup(){
pinMode(13,OUTPUT);    // pin 13 will blink to your heartbeat!
pinMode(2,INPUT);         // pin 2 accepts switch inuput
pinMode(4,OUTPUT);     //  pin4 outputs to buzzer
pinMode(8,OUTPUT);     //  pin8 outputs to leds
pinMode(3,OUTPUT);     //  pin3 outputs to XBEE
Serial.begin(115200);     // we agree to talk fast!
interruptSetup(); }        // sets up to read Pulse Sensor signal every 1mS

void loop(){
select();                      //  OR logic of two inputs
delay(20); }                  //  take a break

void switchcontrol(){
if (digitalRead(2) == HIGH){switchselect = true;} // manual switch is on
else{switchselect = false;}  }                              // manual switch is off

void pulsecontrol(){
if ((BPM > upthres) || (BPM < lowthres)){pulseselect = true;}   // out of range
else{pulseselect = false;}  }                                                  // normal

void select(){
switchcontrol();
pulsecontrol();
if (pulseselect||switchselect){     // trigger alarm
digitalWrite(3,HIGH);
digitalWrite(4,HIGH);
digitalWrite(8,HIGH);}
```

```
else{
digitalWrite(3,LOW);                // output low
digitalWrite(4,LOW);
digitalWrite(8,LOW);} }

volatile int rate[10];                          // get running average of HRV values
volatile unsigned long lastBeatTime = 0;  // find the time between beats
volatile int sampleCounter;                 // determine pulse timing
volatile int runningTotal;                  // keep track of pulses
volatile boolean firstBeat = true;          // seed rate array
volatile boolean secondBeat = true;       // startup with reasonable BPM

void interruptSetup(){ // Initializes Timer1 to throw an interrupt every 1mS.
TCCR1A = 0x00;         // DISABLE OUTPUTS AND PWM ON DIGITAL PINS 9 & 10
TCCR1B = 0x11;         // 'PHASE AND FREQUENCY CORRECT' MODE, NO PRESCALER
TCCR1C = 0x00;         // DON'T FORCE COMPARE
TIMSK1 = 0x01;         // ENABLE OVERFLOW INTERRUPT (TOIE1)
ICR1 = 8000;                  // TRIGGER TIMER INTERRUPT EVERY 1mS
sei();  }                       // MAKE SURE GLOBAL INTERRUPTS ARE ENABLED

// THIS IS THE TIMER 1 INTERRUPT SERVICE ROUTINE.
ISR(TIMER1_OVF_vect){                  // triggered every time Timer 1 overflows
// Timer 1 makes sure that we take a reading every millisecond
Signal = analogRead(pulsePin);          // read the Pulse Sensor
sampleCounter++;// keep track of the time with this variable (ISR triggered every 1mS
//  NOW IT'S TIME TO LOOK FOR THE HEART BEAT
int H = sampleCounter-lastBeatTime;  // monitor the time since the last beat to avoid noise
if ( (Signal > 520) && (Pulse == false) && (H > 500) ){
// signal surges up in value every time there is a pulse
Pulse = true;                                // set the Pulse flag when we think there is a pulse
digitalWrite(13,HIGH);                    // turn on pin 13 LED
HRV = sampleCounter - lastBeatTime; // measure time between beats in mS
lastBeatTime = sampleCounter;          // keep track of time for next pulse
if(firstBeat){                               // if it's the first time we found a beat
firstBeat = false;                           // clear firstBeat flag
return;}                                     // HRV value is unreliable so discard it
if(secondBeat){                            // if this is the second beat
secondBeat = false;                          // clear secondBeat flag
for(int i=0; i<=9; i++){rate[i] = HRV;}}   // seed the running total; get a realisitic BPM at startup
// keep a running total of the last 10 HRV values
for(int i=0; i<=8; i++){rate[i] = rate[i+1];}  // shift data in the rate array; drop the oldest HRV value
rate[9] = HRV;                               // add the latest HRV to the rate array
runningTotal = 0;                            // clear the runningTotal variable
```

```
for(int i=0; i<=9; i++){runningTotal += rate[i];}   // add up the last 10 HRV values
runningTotal /= 10;                                 // average the last 10 HRV values
BPM = 60000/runningTotal;}                          // how many beats can fit into a minute? that's BPM!
QS = true;                                  // set Quantified Self flag when beat is found and BPM gets updated.
// QS FLAG IS NOT CLEARED INSIDE THIS ISR
if (Signal < 500 && Pulse == true){ // when the values are going down, it's the time between beats
digitalWrite(13,LOW);                               // turn off pin 13 LED
Pulse = false; }                                    // reset the Pulse flag so we can do it again!
}                                                    // end isr
```