



pennsylvania

DEPARTMENT OF TRANSPORTATION

Remote Sensing for Bridge Scour

FINAL REPORT

Date: July 31, 2009

By: Peter J. Hawrylak and Marlin H. Mickle

University of Pittsburgh

COMMONWEALTH OF PENNSYLVANIA
DEPARTMENT OF TRANSPORTATION

CONTRACT # 510601
WORK ORDER # PIT 018



Technical Report Documentation Page

1. Report No. FHWA-PA-2009-012-PIT018	2. Government Accession No.	3. Recipient's Catalog No.
4. Title and Subtitle Remote Sensing for Bridge Scour PennDOT Work Order #18 Final Report		5. Report Date July 31, 2009
7. Author(s) Peter J. Hawrylak (PI), Marlin H. Mickle (PI), Kirsten McCane (Graduate Student Researcher)		6. Performing Organization Code
9. Performing Organization Name and Address RFID Center of Excellence University of Pittsburgh Pittsburgh, PA 15261		8. Performing Organization Report No. Bridge Sour Monitoring Report July 2009
12. Sponsoring Agency Name and Address The Pennsylvania Department of Transportation Bureau of Planning and Research Commonwealth Keystone Building 400 North Street, 6 th Floor Harrisburg, PA 17120-0064		10. Work Unit No. (TRAIS)
15. Supplementary Notes		11. Contract or Grant No. 510601
16. Abstract A large percentage of bridges within the state of Pennsylvania are located over waterways. For such bridges, much of the supporting structure is positioned within the river or stream bed of the waterway. As a result, these bridges are susceptible to bridge scour, the washing away of fill around structures, which compromises the safety of the bridge. Bridge inspections have a limited frequency at which they can occur. In between these inspections, unmonitored and difficult to detect events may happen that create an immediate danger to the general public. The purpose of Work Order #18 (this research) was to create a prototype system for bridge scour monitoring that would provide continuous monitoring. The technique utilized for this particular bridge scour monitoring was detection using float out devices. The float out device concept is to bury transmitters at various locations around a bridge structure. These devices would then be released due to the scour's removal of material around the device. A receiver on the bridge would receive the transmission and perform an action. A float out device system would provide an initial indication of scour severity for further investigation. To realize this technique, a prototype remote sensing system was designed to have three main components that work together to indicate bridge scour. The first component is a transmitter coupled with circuitry and encased such that it is watertight and can be buried under the materials surrounding the bridge structure. It is able to be armed such that it will remain dormant while buried and become active upon its release and rise to the surface of the water. Lastly, the communication link and time it takes the unit to rise are such that transmission of a message can consistently occur. This component in its entirety is referred to as the Sensor Unit within this document. The second component is a receiver coupled with circuitry that will be encased and installed on the bridge overpass. The receiver is able to store and interpret the RF messages sent by the Sensor Unit. Given this interpretation it is able to provide a visual indication of scour severity by interfacing with the third component described shortly. The receiver circuitry will have the capability to be powered continually. This component in its entirety is referred to as the Receiver Unit. The third component is a set of lights (LEDs) encased with its supporting circuitry. The purpose of these lights is to correspond to a particular Sensor Unit. This will allow for immediate comprehension by the inspector without having to be on the bridge. This component must be able to interface with the Receiver Unit as the light that will be lit is controlled by the Receiver Unit. This component in its entirety is referred to as the Light Indicator. This document contains documentation of the work done for Work Order #18. This document contains all analysis leading up to the system design as well as the rationale behind design decisions made. An initial assessment of what the Scour Monitoring System would be composed of and its functionality is found in the requirements section. A less technical overview of the system designed to meet those requirements can be found in the specifications section. Following this are design		13. Type of Report and Period Covered Final Report: June 1, 2008 – July 31, 2009
		14. Sponsoring Agency Code

sections explaining the system design in detail. Lastly, the implementation and testing of the system is presented. This document also contains all items related to the system such as *Bill of Parts*, *Schematics*, *Layouts*, and *Embedded Software Code* .)

17. Key Words Bridge Scour		18. Distribution Statement Restricted.	
19. Security Classif. (of this report) Unclassified	20. Security Classif. (of this page) Unclassified	21. No. of Pages 202	22. Price

Form DOT F 1700.7

(8-72)

Reproduction of completed page authorized

Table of Contents

1.0	Executive Summary	14
2.0	Terms, Definitions, Symbols, and Abbreviated Terms.....	16
2.1	Terms and Definitions.....	16
2.2	Abbreviated Terms.....	16
3.0	Requirements	18
3.1.1	List of Requirements.....	19
3.1.2	Wireless Link Communication Frequency	20
3.1.3	Sensor Unit Capsule Dimensions and Tethering Option	20
3.1.4	Data Requirements.....	20
3.1.5	System Interfacing and Programming	21
3.1.6	Power Considerations	22
4.0	Specifications	23
4.1	Sensor Unit Specifications.....	28
4.1.1	Senor Unit Hardware Architecture	28
4.1.2	Sensor Unit Data Block	30
4.1.3	Sensor Unit Software Architecture	32
4.2	Receiver Unit Specifications.....	33
4.2.1	Receiver Unit Hardware Architecture	33
4.2.2	Receiver Unit Software Architecture.....	36
4.3	Light Indicator Specifications.....	38
4.3.1	Light Indicator Hardware Architecture.....	38
5.0	Tether Option Analysis	40
5.1	Transmitter –Receiver Range	40
5.2	Sensor Unit Rise Time Analysis	41
5.2.1	Buoyancy Calculation.....	41
5.2.2	Drag Calculation	42
5.2.3	Simulation Method.....	43
5.3	Rise Time Results	47
5.3.1	Distance Traveled While Transmitting.....	47
5.4	Tethered Option	50
5.4.1	Description of Tethered Option Solution.....	51
5.4.2	Pros and Cons of Tethered Option.....	51
5.5	Un-Tethered Option	52
5.5.1	Description of Un-Tethered Option Solution	52
5.5.2	Pros and Cons of Un-Tethered Option.....	52
6.0	Design	53
6.1	Sensor Unit Description of Operation.....	53
6.2	Sensor Unit Design	55
6.2.1	Microcontroller	56
6.2.2	RF Transmitter	58
6.2.3	Relay Switch	59
6.2.4	Arm Switches.....	60
6.2.5	Tilt Switches	64
6.2.6	Voltage Regulator	64
6.2.7	JTAG/BSL Connector.....	65
6.2.8	Mini USB Port	66

6.2.9	Batteries	67
6.3	Receiver Unit Design	68
6.3.1	Microcontroller	68
6.3.2	RF Receiver	70
6.3.3	JTAG/BSL Connector.....	72
6.3.4	Light Indicator Header	73
6.3.5	Power Switch	74
6.3.6	Power Jack	74
6.3.7	Voltage Regulator	75
6.3.8	SMA Connector	76
6.3.9	Batteries	76
6.4	Light Indicator Design	77
6.4.1	Receiver Headers	77
6.4.2	OR Gates	79
6.4.3	LEDs	80
7.0	Sensor Unit Software Design.....	82
7.1	Initialization	82
7.2	Sensor Unit Data Block Storage	83
7.3	Data Transmission	85
7.3.1	Manchester Encoding.....	85
7.3.2	Synchronization	85
7.3.3	Transmission.....	86
7.3.4	Delay Addition for Collision Avoidance	87
8.0	Receiver Software Design.....	91
8.1	Initialization	91
8.2	Main Program	92
9.0	System Testing.....	94
9.1	Preliminary Component Testing	94
9.1.1	Transmitter Signal Strength	94
9.1.2	Transmitter-Receiver RF Link.....	97
9.2	Sensor Unit Preliminary Component Testing	98
9.2.1	Battery Test.....	100
9.2.2	Tilt Switch.....	100
9.2.3	Arm Switch	102
9.2.4	Relay Switch	102
9.3	Receiver Unit Preliminary Component Testing.....	104
9.3.1	Voltage Regulator (3.0 Volts DC)	104
9.4	Light Indicator Preliminary Prototype Testing	105
9.5	Capsule Testing.....	105
9.5.1	Buoyancy	106
9.5.2	Rise Times	106
9.5.3	Water Tightness	108
9.5.4	Pressure Testing	109
9.6	System Functionality Test.....	110
9.6.1	Overall System Demonstration	111
9.6.2	Multiple Sensor Functionality.....	116
9.7	UART Interface Functionality Tests.....	120
9.7.1	Sensor Unit Serial Interface Protocol and Testing.....	120

9.7.2	Receiver Unit UART Protocol.....	123
10.0	Assembly and Installation.....	126
10.1	System PCB Assemblies.....	126
10.2	Sensor Unit Encapsulation and Installation	126
10.2.1	Step 1 - Sensor Unit PCB Preparation	126
10.2.2	Sensor Unit PCB – Capsule Attachment	127
10.2.3	Sensor Unit Capsule - External Switch.....	127
10.2.4	Sealing the Sensor Unit Capsule.....	128
10.2.5	Sensor Unit Installation/Deployment.....	129
11.0	Conclusions.....	130
12.0	Bill of Materials	131
12.1	Sensor Unit Bill of Materials	131
12.2	Receiver Unit Bill of Materials.....	134
12.3	Light Indicator Bill of Materials	137
12.4	Printed Circuit Boards.....	139
13.0	Hardware Design files.....	140
13.1	Schematics	140
13.1.1	Sensor Unit Schematic	141
13.1.2	Receiver Unit Schematic.....	142
13.1.3	Light Indicator Schematic.....	143
13.2	PCB Layout Designs.....	144
13.2.1	Sensor Unit PCB Design.....	144
13.2.2	Receiver Unit PCB Design	147
13.2.3	Light Indicator PCB Design.....	150
14.0	Source Code	152
14.1	Sensor Unit Source Code	152
14.2	Receiver Unit Source Code.....	165
15.0	Development Software Programs Used	180
16.0	References.....	181
17.0	System Manual.....	182
18.0	Hardware Components.....	183
18.1.1	Sensor Unit Hardware.....	183
18.1.2	Receiver Unit Hardware	186
18.1.3	Light Indicator Hardware.....	188
19.0	Software	190
19.1	Software Utilities	190
19.1.1	Code Composer Essentials v3.1 Core Edition	190
19.1.2	FET-Pro430.....	192
19.1.3	HyperTerminal.....	193
19.2	UART Operation and Protocol	195
19.2.1	Sensor Unit UART Protocol	195
19.2.2	Receiver Unit UART Protocol.....	197
19.3	Microcontroller Program Code.....	199
20.0	System PCB Assemblies.....	200
21.0	Sensor Unit Encapsulation and Installation	201
21.1	Step 1 - Sensor Unit PCB Preparation	201
21.2	Sensor Unit PCB – Capsule Attachment	201
21.3	Sensor Unit PCB Capsule Peripheral Attachment	201

21.4	Sealing the Sensor Unit Capsule.....	202
21.5	Sensor Unit Installation.....	203

List of Tables

Table 1: Breakdown of the Location of Sensor Field	31
Table 2: Interpretation and Action Resulting from the Color Code Sub-field by the Receiver Unit.	31
Table 3: Length of the Data Fields Stored in the Sensor Unit Data Block.	32
Table 4: Sensor Unit Data Block	32
Table 5: Reynolds Number for Diameters of 2 and 4 inches.....	43
Table 6: Message Transmission Time and Distance Traveled	47
Table 7: Delay Percentages for Each Color Code, $T \approx 100$ ms	89
Table 8: Signal Strength Tests in Free Air and Within Capsule (TX Test 1 and TX Test 2)	95
Table 9: Signal Strength Tests For Horizontal Floating Capsule (TX Test 3)	96
Table 10: Signal Strength Tests for Vertically Floating Capsule (TX Test 4)	97
Table 11: Transmitter-Receiver RF Link Tests	97
Table 12: Battery Test.....	100
Table 13: Tilt Switch Test.....	101
Table 14: Arm Switch Test.....	102
Table 15: Relay Switch Test.....	102
Table 16: Voltage Regulator Test.....	104

List of Figures

Figure 1: Scouring Solution.....	18
Figure 2: System Diagram with Tethered Option on the Left and Un-tethered Option on the Right.....	24
Figure 3: Multiple (three) Sensor Units Deployed in the Same Location (hole).....	25
Figure 4: Plan View of a Sample Installation with a Near Side Abutment (NAB1) and One Pier (P001). (Three Sensor Units are placed (at different depths) in each hole.).....	26
Figure 5: High-level System Flow Diagram.....	27
Figure 6: Top-level Block Diagram of the Sensor Unit.....	28
Figure 7: Sensor Unit Software.....	33
Figure 8: Receiver Unit Block Diagram.....	34
Figure 9: Flow Chart of the Receiver Unit Software.....	37
Figure 10: Light Indicator Hardware Diagram.....	38
Figure 11: Travel Distances for Capsule with a Radius of 0.5 inches and a Length of 3.0 inches.....	48
Figure 12: Travel Distances for Capsule with a Radius of 1.0 inches and a Length of 5.0 inches.....	49
Figure 13: Travel Distances for Capsule with a Radius of 1.5 inches and a Length of 7.0 inches.....	50
Figure 14: Tilt Switch Diagram.....	54
Figure 15: Activation Position of First Tilt Switch.....	54
Figure 16: Activation Position of Second Tilt Switch.....	55
Figure 17: Activation Position of Third Tilt Switch (naturally-oriented position).....	55
Figure 18: Schematic of the Microcontroller-MSP430F2132.....	56

Figure 19: Schematic of the TXM-433-LR Transmitter.....	58
Figure 20: Schematic of the DS1E-SL-DC3V Relay Switch	59
Figure 21: (A) On-Board Arm Switch Schematic and (B) On-Board Arm Switch Pin View	60
Figure 22: Schematic Showing the Switch Connections for the On-board Arm Switch When in the Armed State (Left Position).	62
Figure 23: Schematic Showing the Switch Connections for the On-board Arm Switch When in the Reset State (Right Position).	62
Figure 24: (A) External Arm Switch Schematic and (B) External Arm Switch Pin View	63
Figure 25: Schematic of the S1234 Tilt Switch.....	64
Figure 26: Schematic of the LM3940 Voltage Regulator.....	65
Figure 27: Schematic of JTAG/BSL Header	65
Figure 28: Mini USB Port Schematic	67
Figure 29: Schematic of the CR-2/BE Batteries.....	67
Figure 30: Schematic of Microcontroller-MSP430F2132	68
Figure 31: Schematic of RXM-433-LR Receiver.....	71
Figure 32: Schematic of JTAG/BSL Header	72
Figure 33: Schematic of Light Indicator Component	73
Figure 34: Schematic of SPDT Switch.....	74
Figure 35: Schematic of Power Jack.....	75
Figure 36: Voltage Regulator.....	75
Figure 37: Schematic of SMA Connector.....	76
Figure 38: Schematic of the CR-2/BE Batteries	77

Figure 39: Schematic of Receiver Header	78
Figure 40: Schematic of OR Gates	79
Figure 41: Schematic of LED Circuit	80
Figure 42: Flowchart for Data Storage	84
Figure 43: Manchester Encoding	85
Figure 44: Bit Transition Sequences for Consecutive Bits	86
Figure 45: Flowchart for Data Transmission Program	87
Figure 46: Collision Scenarios of Two Transmissions	88
Figure 47: Collision Scenario with Delays Added	89
Figure 48: Oscilloscope shot of Red Priority Delays and Yellow Priority Delays	90
Figure 49: Flowchart for Receiver Program	93
Figure 50: Transmitter Prototype - Top Side	99
Figure 51: Transmitter Prototype - Bottom Side	100
Figure 52: Tilt Switch Test Vertical Orientation	101
Figure 53: Tilt Switch Test Horizontal Orientation	101
Figure 54: Relay Switch Test in Horizontal Orientation	103
Figure 55: Relay Switch Test in Vertical Position	103
Figure 56: Receiver Unit Prototype	104
Figure 57: Light Indicator Unit Prototype and Test Stand	105
Figure 58: Capsule Floating with No Additional Weight Added	106
Figure 59: Rise Time Test Before Capsule Release	107
Figure 60: Rise Time Test After Sensor Release	107
Figure 61: Weight Added to Keep Capsule Submerged for Water Tightness Tests	108

Figure 62: Water Collected from the First Water Tight Test Using Only the PVC Pipe.	108
Figure 63: Capsule with Teflon Tape Added.....	109
Figure 64: One Silica Gel Desiccant Packet.....	109
Figure 65: Capsule with Valve and Hose Attachment for Pressure Testing.....	110
Figure 66: Transmitter Memory View of Dummy Data Block	112
Figure 67: Sensor Unit Ready for the Final Assembly Step.....	113
Figure 68: Container with Aggregate and Pipe.....	113
Figure 69: Container with Aggregate, Pipe, and Water.....	114
Figure 70: Receiver Unit and Light Indicator with No Message Received from a Sensor Unit.	114
Figure 71: Sensor Unit after Release from Aggregate.....	115
Figure 72: Receiver and Light Indicator after Sensor Unit Release	115
Figure 73: Demonstration Setup with No Sensor Unit Active	116
Figure 74: Step 1 - Green Sensor Unit is Activated. Note that the Light Indicator shows the Green Condition.....	117
Figure 75: Step 2 - Yellow Sensor Unit is Activated with Green Sensor Unit Already Active. Note That the Light Indicator Shows the Yellow Condition.	118
Figure 76: Step 3 - Orange Sensor Unit is Activated with the Green and the Yellow Sensor Units Already Active. Note that the Light Indicator Shows the Orange Condition.	119
Figure 77: Step 4 - Red Sensor Unit is Activated with the Orange, Yellow, and Green Sensor Units Already Active. Note that the Light Indicator Shows the Red Condition.	120
Figure 78: Transmitter UART Communication: Transfer of Data Block from txt File	121

Figure 79 : Transmitter UART Communication: File Sent and Data Block Read Out .	122
Figure 80: Sensor Unit Data Block Read Out.....	122
Figure 81: Bridge ID Transfer and Read Out	124
Figure 82: Transferred Data Block Read Out and System Reset.....	125
Figure 83: External Arm Switch Construction Steps.....	128
Figure 84: CCE3 Launch	191
Figure 85: New Project Prompt	191
Figure 86: Project Setup.....	192
Figure 87: FET-Pro430 Interface.....	193
Figure 88: FET - Pro430 Setup.....	193
Figure 89: Hyper Terminal Startup.....	194
Figure 90: Com Settings	195
Figure 91: Sensor Unit Data Block Read Out.....	196
Figure 92: Transmitter UART Communication: Transfer of Data Block from txt File	197
Figure 93: Bridge ID Transfer and Read Out	198
Figure 94: Transferred Data Block Read Out and System Reset.....	199
Figure 95: External Arm Switch Construction Steps.....	202

ACKNOWLEDGEMENTS

The study group at the University of Pittsburgh Swanson School of Engineering would like to acknowledge the following people and groups who have worked on and contributed to this project:

Erik Blank, Tom Knieriem, Paul Majoris, Tom Macioce, Jeff Matko, Becky Lorah (Pennsylvania Department of Transportation); and Derek Constable (Federal Highway Administration)

This work was sponsored by the Pennsylvania Department of Transportation and the U.S. Department of Transportation, Federal Highway Administration. The contents of this report reflect the views of the authors, who are responsible for the facts and the accuracy of the data presented herein. The contents do not necessarily reflect the official views or policies of the Federal Highway Administration, U.S. Department of Transportation, or the Commonwealth of Pennsylvania at the time of publication. This report does not constitute a standard, specification, or regulation.

1.0 Executive Summary

A large percentage of the bridges within the state of Pennsylvania are located over waterways. For those bridges, much of the supporting structure is positioned within the river or stream bed of the waterway. As a result, these bridges are susceptible to bridge scour, the washing away of fill around structures, which compromises the safety of the bridge. Bridge inspections have a limited frequency at which they can occur. In between these inspections, unmonitored and difficult to detect events may happen that create an immediate danger to the general public. The purpose of Work Order #18 (this research) was to create a prototype system for bridge scour monitoring that would provide continuous monitoring.

The technique utilized for this particular bridge scour monitoring was detection using float out devices. The float out device concept is to bury transmitters at various locations around a bridge structure. These devices would then be released due to the scour's removal of material around the device. A receiver on the bridge would receive the transmission and perform an action. A float out device system would provide an initial indication of scour severity for further investigation.

To realize this technique, a prototype remote sensing system was designed to have three main components that work together to indicate bridge scour. The first component is a transmitter coupled with circuitry and encased so that it is watertight and can be buried under the materials surrounding the bridge structure. It is able to be armed so that it will remain dormant while buried and become active upon its release, rising to the surface of the water. Lastly, the communication link and time it takes the unit to rise are such that transmission of a message can consistently occur. This component in its entirety is referred to as the **Sensor Unit** within this document.

The second component is a receiver coupled with circuitry that will be encased and installed on the bridge overpass. The receiver is able to store and interpret the RF messages sent by the Sensor Unit. Given this interpretation it is able to provide a visual indication of scour severity by interfacing with the third component described shortly. The receiver circuitry will have the capability to be powered continually. This component in its entirety is referred to as the **Receiver Unit**.

The third component is a set of lights (LEDs) encased with its supporting circuitry. The purpose of these lights is to correspond to a particular Sensor Unit. This will allow for immediate comprehension by the inspector without having to be on the bridge. This component must be able to interface with the Receiver Unit as the light that will be lit is controlled by the Receiver Unit. This component in its entirety is referred to as the **Light Indicator**.

This report contains documentation of the work done for Work Order #18. This document contains all analysis leading up to the system design as well as the rationale behind design decisions. An initial assessment of what the Scour Monitoring System would be composed of and its functionality is found in the *Requirements* section. A less technical overview of the system designed to meet those requirements can be found in the *Specifications* section. Following these sections are *Design* sections explaining the system design in detail. Lastly, the implementation and testing of the system is presented. This document also contains all items related to the system such as *bill of parts, schematics, layouts, and embedded software code*.

2.0 Terms, Definitions, Symbols, and Abbreviated Terms

This section contains a list of terms with definitions and abbreviations used throughout the rest of the document.

2.1 Terms and Definitions

- **Byte** Eight bits
- **C** C programming language
- **Nibble** Four bits

2.2 Abbreviated Terms

- **AC** Alternating Current
- **BMS** Bridge Management System
- **BSL** Boot Strap Loader
- **COTS** Commercial Off the Shelf
- **CRC** Cyclic Redundancy Check
- **CRC-16** 16-bit CRC
- **DC** Direct Current
- **DPDT** Double Pull Double Throw
- **HEX** Hexadecimal
- **IDE** Integrated Development Environment
- **ISM** International Scientific Medical
- **ISR** Interrupt Service Routines
- **JTAG** Joint Test Action Group
- **LED** Light Emitting Diode
- **OOK** On-Off Keying
- **PennDOT** Pennsylvania Department of Transportation
- **PVC** Poly-Vinyl-Chloride
- **Re** Reynolds number
- **RF** Radio Frequency
- **SMA** Type of connector
- **UART** Universal Asynchronous Receiver/Transmitter
- **USB** Universal Serial Bus

- **VAC** Voltage AC
- **VDC** Voltage DC

3.0 Requirements

Bridges over waterways, such as streams and rivers, have foundation structures that extend down into the stream or river beds. Many of these structures are surrounded by aluvial soil and other materials that are prone to scour.

During flood events the stream flow velocities are such that there is a tendency for the foundation materials to wash away. This process is termed Scouring. PennDOT has been looking for a real time solution to detect this Scouring without success over the last few years. During flooding, PennDOT can send a crew to check the bridges, but it is difficult to see below the surface of the water to determine if Scouring has taken place to a level where the bridge needs to be closed.

The objective is the development of a sensor and detection system that can be used to produce a visual indication to an inspector in the vicinity of the bridge that scouring has taken place and possibly to what level. Through relatively straightforward communications channels, the system could be enhanced so that the reporting mechanism is directed to a central facility from numerous remote locations.

The solution to the Scouring problem can be illustrated using Figure 1, where the focus is on a single substructure member. Other substructure members are a simple replication of Figure 1. The key to the solution is a sensor possibly fabricated in a cylindrical container something like an aluminum drink can although the material forming the container will be Radio Frequency (RF) friendly and watertight.

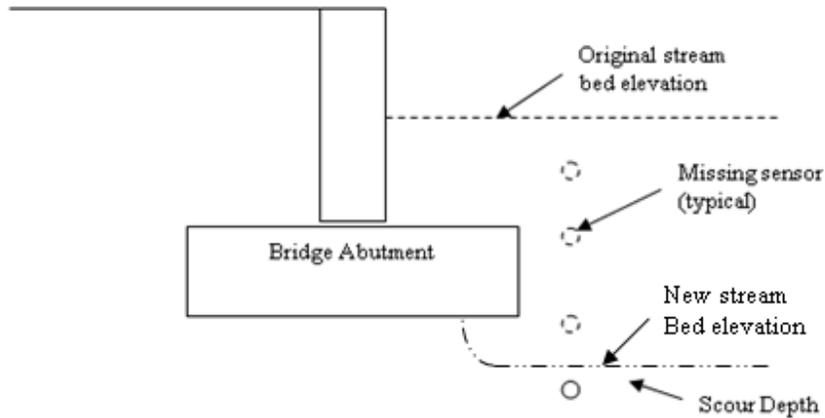


Figure 1: Scouring Solution.

The sensor will contain an RF transmitter, operating in the ISM (Industrial, Medical, Scientific) Band, and a microprocessor. The sensor circuitry will include a position orientation sensitive switch and an external arming switch. Once the sensor is armed, any deviation from the vertical shown in Figure 1, will activate the microprocessor which will continuously transmit an RF beacon identifying the particular sensor signifying a particular substructure member, original planted level, and possibly an x and y location designation.

The sensor is placed in a hole drilled with a standard NX (3-3/16 inch ID) hollow stem auger or steel casing. The sensor is placed and covered with sand or whatever is

appropriate for the application. The possibility of placing an anchor with a tether will also be explored.

As the soil is scoured from the site, once the sensor is uncovered (freed), the relative buoyancy causes the sensor to rise to the surface of the stream or river. The sensor container is given an uneven weight distribution so as to activate the position sensing switch which activates the RF beacon notifying the receiver of the location and the level of the scouring.

This section reviews the basic requirements for this system. These were major design decisions made during the outset of the project. These topics are listed below as well as discussed in a broader scope in this section.

3.1.1 List of Requirements

- Sensor Unit to be fabricated in a cylindrical container
- Sensor Unit must be watertight
- Sensor Unit container must be RF friendly (allow RF [wireless signal] through)
- Sensor Unit must be sufficiently buoyant to rise to the water surface within range of the Receiver Unit
- Sensor Unit must contain an RF (wireless) Transmitter
- Sensor Unit must transmit in an ISM Band
- Sensor Unit must be controlled by a microcontroller
- Sensor Unit must have a positional orientation sensitive switch
- Sensor Unit must have an external arming/disarming switch
- Sensor Unit must activate and transmit upon deviation from vertical orientation
- Once activated the Sensor Unit must transmit continuously
- Sensor Unit must transmit within range under flood conditions
 - Tethering the Sensor Unit to an anchor must be analyzed
- Sensor Unit must transmit information identifying the bridge and structure it is assigned to, a serial number, and the severity of the scour it is monitoring
- Sensor Unit must be installed using a standard NX (3-3/16 inch interior diameter) hollow stem auger
- The Sensor Unit must provide depth accuracy to ± 1 foot.
- The Receiver Unit must be placed at a fixed position on the bridge
- The Receiver Unit must receive and store Sensor Unit messages
- The Receiver Unit must interpret Sensor Unit messages
- The Receiver Unit must control a Light Indicator (visual indication of scour) based on Sensor Unit messages received
 - The Light Indicator must be visible to the inspector without the inspector being on the bridge, i.e., from the bridge approach
- The Sensor Unit information must be programmed via a host PC
- The bridge identifier associated with a particular Receiver Unit must be programmed via a host PC
- The Receiver Unit must provide a communication mechanism to allow bridge inspectors to download stored Sensor Unit messages
- The Receiver Unit must provide a communication mechanism to allow bridge inspectors to reset the Receiver Unit to a no alarm condition

- The Receiver Unit must provide a communication mechanism to allow bridge inspectors to reset the Receiver Unit to erase all stored Sensor Unit messages
- The Light Indicator will indicate four different levels of scour.

3.1.2 Wireless Link Communication Frequency

Several frequency ranges are available for use in this system, including 433 MHz and 915 MHz. Lower frequencies, such as 125 kHz and 13.56 MHz, have a shorter range (a few feet) with smaller antenna dimensions and configurations compared to the 433 MHz and 915 MHz-based systems. The 915 MHz-based systems are more susceptible to interference from debris (soil, branches, etc.) that may be present during a flood. Commercial off the shelf (COTS) devices exist for both the 433 MHz and 915 MHz-based systems.

The 433 MHz-based systems are less susceptible to interference from debris because the lower frequency can be expected to more easily penetrate such debris and is compatible with typical sensor dimensions. For these reasons, the 433 MHz-based solution was chosen.

3.1.3 Sensor Unit Capsule Dimensions and Tethering Option

Given any frequency and transmitter/receiver link (wireless communication link) properties, there is a limit on the range data that can be reliably transmitted. As a result, several options have to be evaluated involving the choices related to the Sensor Unit capsule and whether or not to tether the Sensor Unit. The Sensor Unit capsule size and material were the main factors in how quickly the Sensor Unit rises to the water surface where it can transmit to the Receiver Unit.

The system requires that the Sensor Unit capsule be such that it rises within transmission range. Increasing the capsule's volume is the most direct way to increase the speed at which it rises to the surface. However, the Sensor Unit should not become overly large so that it cannot be installed using a 3-3/16 inch internal diameter hollow stem auger drill used by PennDOT. This system will provide an accuracy of scour depth to ± 1 foot. The depth will be relative to the base of the bridge footing.

A tether option for the Sensor Unit was considered, but was found not to be required because the Sensor Unit would reach the water surface within range of the Receiver Unit. One major drawback of the tether option is that it introduces an additional failure point and complicates installation.

3.1.4 Data Requirements

The Sensor Unit must contain information to identify its location and depth, as well as a bridge identification to link it to the bridge at which it is deployed. Storing the bridge identification enables the Receiver Units to distinguish between Sensor Units belonging to the bridge they are mounted on and the sensors from other bridges that will follow the stream flow. This feature will also provide the capability to link multiple Receiver Units together and transfer information among them, allowing a Receiver Unit on one bridge to potentially pick up a Sensor Unit from another bridge, and report that information to the other bridge or to a centralized location.

The following data items will be stored in non-volatile memory on the sensor unit.

1. Bridge Identification Number (from the BMS system [1])
2. Sensor Serial Number
3. Location of Sensor
4. Color Code

The information in these four fields will be coordinated with the Receiver Unit on the bridge during installation. The Receiver Unit needs to relate each Sensor Unit with a particular action indicating the level of bridge scour. This action is contained in the Color Code field. The Receiver Unit only takes the action required by the Color Code if the Bridge Identification Number of the Sensor Unit and the Receiver Unit are identical. This prevents a Receiver Unit from Bridge A reporting scour from a Sensor Unit assigned to Bridge B.

The four fields mentioned above are stored in non-volatile memory as groups of bytes. Each data field is stored in a specific location in the Sensor Unit's non-volatile memory. These data fields are transmitted to the Receiver Unit once the Sensor Unit is activated by its release from the soil. Because messages may be corrupted during transmission (similar to static on a cell phone), the sensor system incorporates error detection information via a cyclic redundancy check (CRC) [12]. The transmitter (Sensor Unit) will perform a calculation on the data being transmitted to compute the CRC. The data and CRC are transmitted to the receiver (Receiver Unit) and another calculation using the data and CRC is performed at the receiver (Receiver Unit). Based on the result of this calculation the receiver (Receiver Unit) can determine if the message was corrupted. This error detection information enables the Receiver Unit to detect transmission errors in the message. The messages that were determined to contain an error via the CRC will be discarded (ignored).

Adding the error detection information increases the amount of data to transmit, which increases the time required for one transmission. The CRC-16 (16 bit CRC) offers robust error detection capability and is sufficient for this application. Because the data in the Sensor Unit does not change, the CRC-16 can be calculated before deployment of the Sensor Unit. The CRC-16 is computed beforehand and stored in non-volatile memory at the same time the information above is being written to the Sensor Unit. This will save time when the Sensor Unit is transmitting and will reduce the footprint of the Sensor Unit's embedded software.

Receiver Units will be programmed with the Bridge Identification Number, and will only activate the display in the event that they receive a broadcast from a sensor unit that is programmed with a matching Bridge Identification Number. Including the Bridge Identification Number in the message prevents a Sensor Unit assigned to one Receiver Unit from triggering an alert on another Receiver Unit assigned to a different bridge. The Sensor Unit was designed so that it can hold all four values and the CRC-16 (error detection information) in non-volatile memory, and the set of data items to transmit can be easily changed in software. These values are combined to generate the Sensor Unit Data Block.

3.1.5 System Interfacing and Programming

A technician accesses the Sensor Unit and Receiver Unit for two different purposes. First, the units have to be programmed by some means. Second, the data must be downloaded from and to the units easily, as well as and basic settings must be reconfigured. The connection from a host PC used to do this must be as simple as

possible and require a standard connection means i.e., USB connection or parallel port. Here a USB connection was used as it is typically present on all computing machines. Simple interfaces to both download the program code and transfer program data needed to be developed.

3.1.6 Power Considerations

The Sensor Unit and Receiver Unit each have particular power requirements. The Light Indicator will be connected to a Receiver Unit and depends on the Receiver Unit for its power source. The Sensor Unit will require batteries as its power source. The batteries used by the Sensor Unit will not need to be active for an extended amount of time. The batteries need to support the functioning of the Sensor Unit long enough that a message can be transmitted, preferably several times, until the Sensor Unit is out of range. However, the Receiver Unit only needs to have one of these messages sent without error.

Given the speed of current microcontrollers, this transmission will normally require no more than 0.184 seconds but the Sensor Unit should be capable of transmitting for a minimum of xxx minutes to ensure that transmissions are received prior to the sensor being moved out of range due to water flow. The battery should also have the available current to support all Sensor Unit circuitry operating simultaneously. Lastly, the Sensor Unit must be able to remain buried for several years. Therefore, the battery chosen must have as long a shelf-life in the dormant state. The Receiver Unit has several options for its power source because it is installed above the surface of the water. The Receiver Unit is powered from a standard 120 VAC source with a standard AC/DC converter (similar transformer on a laptop power cord) providing the DC voltage required by the Receiver Unit. The Receiver Unit is designed to be able to be easily extended in order to accept power input from other sources such as solar panels or batteries.

4.0 Specifications

The system consists of three types of devices, (1) Sensor Units, (2) Receiver Units, and (3) Light Indicators. The Sensor Units are buried in the riverbed or streambed and remain dormant until they are activated as a result of a scour event. Each installation (bridge site) has at least one, but potentially multiple Sensor Units. The Receiver Unit is mounted on the bridge. The Receiver Unit listens for Sensor Unit messages indicating scour and operates a Light Indicator (green, yellow, orange, and red lights) in response to detected Sensor Units (scour events). Typically, a single Receiver Unit is assigned to a bridge, but multiple Receiver Units per bridge may be required for larger structures. The Light Indicator is designed to handle input from multiple Receiver Units.

Figure 2, shown below, illustrates the basic idea of the system given a Tether Option or Non-tethered Option. On the left-hand side, the Tethered Option is shown with a possible solution for the anchor and tether. One solution for the Tether Option is to coil the tether, in this case a wire, around a rod that is attached to the anchor. The wire will unravel as the Sensor Unit rises and this should reduce the risk of the wire getting hung up during both installation and operation. This causes problems if multiple Sensor Units are buried at the same location, i.e. above other Sensor Units at lower depths. The right-hand side illustrates the Un-tethered Option, which is simpler and is not affected by the problem of the tether wire getting caught in debris or soil.

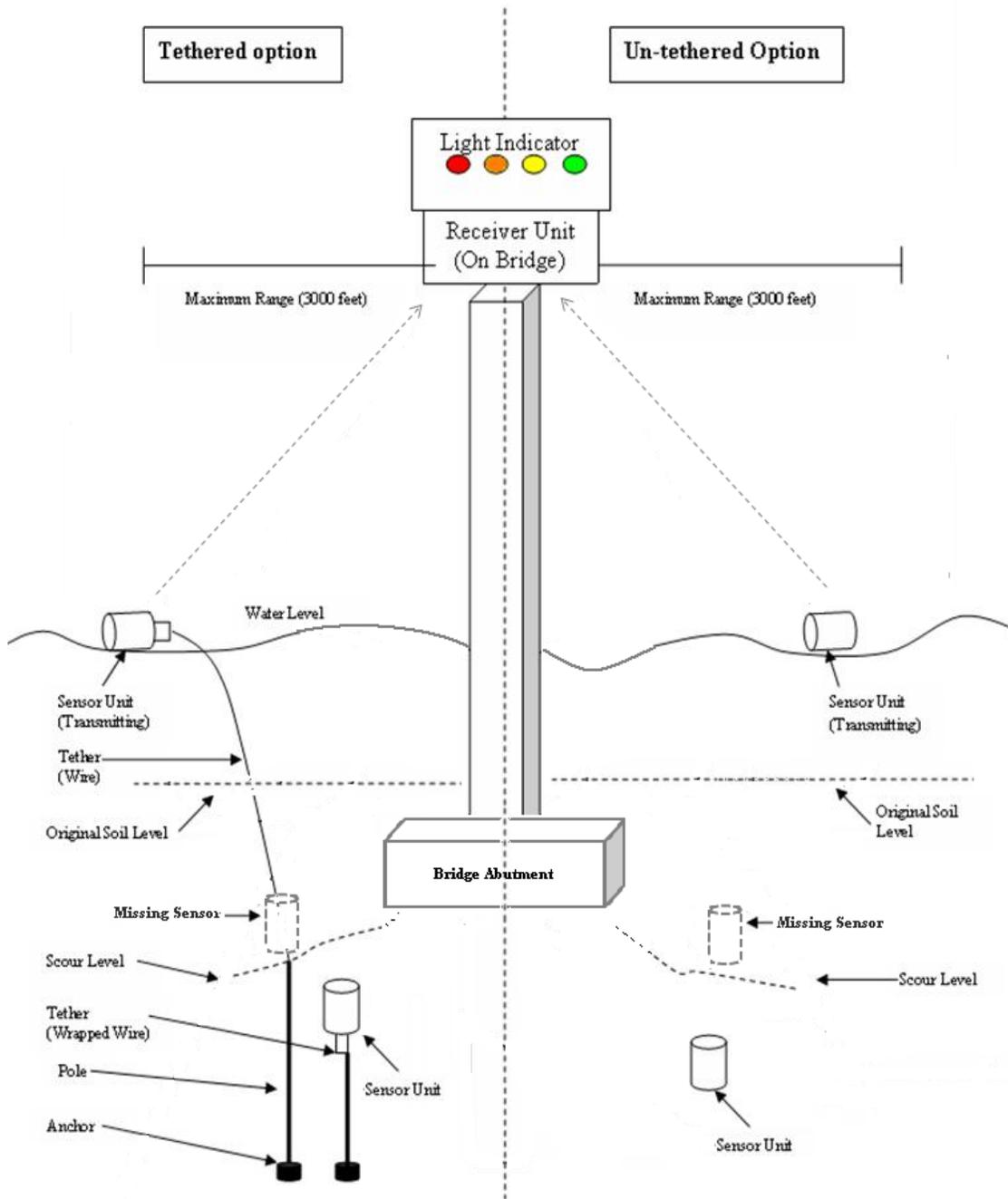


Figure 2: System diagram with Tethered Option on the left and Un-tethered option on the right

The Sensor Units can be buried on top of each other to monitor different levels (severity) of scour as shown in Figure 3.

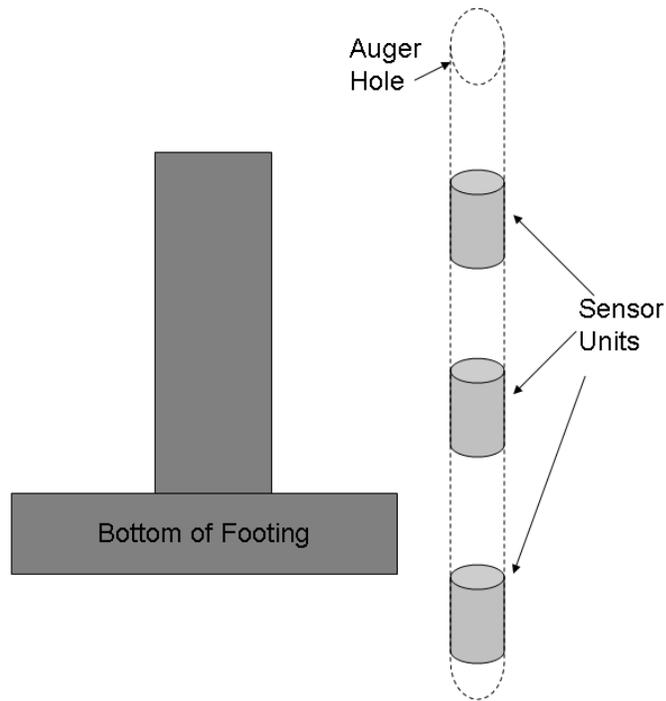


Figure 3: Multiple (three) Sensor Units deployed in the same location (hole).

Each Sensor Unit is linked to a specific bridge and a specific structural member, for instance nearside abutment (NAB) or a pier (P001). A serial number uniquely identifies each Sensor Unit deployed to monitor a particular structural member. An illustration of a plan (top) view of a bridge with a Nearside Abutment (NAB1) and a Single Pier (P001) is provided in Figure 4. Each hole (represented by a filled circle in Figure 4) contains three Sensor Units, each at a different depth as illustrated in Figure 3. A plan view showing this type of information will be used by the team to determine where to deploy the Sensor Units in the field.

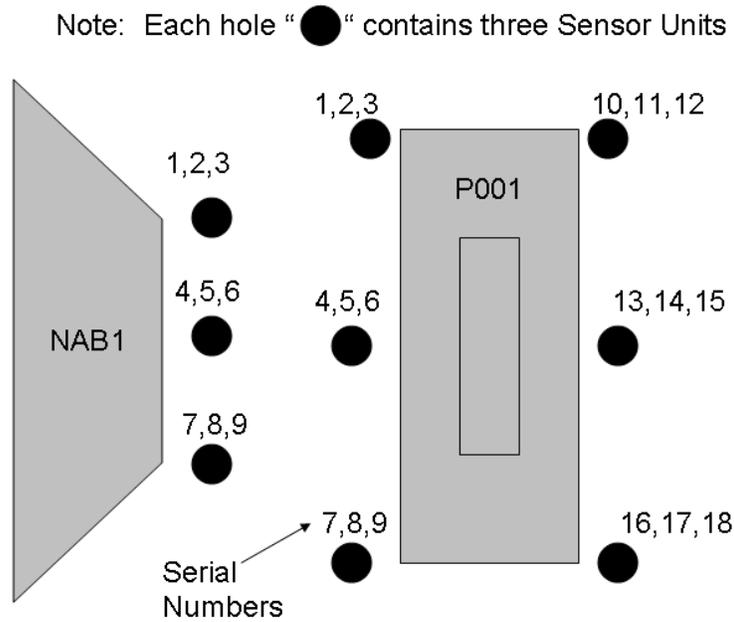


Figure 4: Plan view of a sample installation with a Near Side Abutment (NAB1) and One Pier (P001). Three Sensor Units are placed (at different depths) in each hole.

Figure 5 below shows the high-level operational view of the overall system. The general operation, including installation, of the system is explained in this diagram. The indication lights only transition to a more severe state in response to a message from a Sensor Unit, i.e., no transition from red to green is possible. To clear a light state, a reset on the Receiver Unit must be used. The Receiver Unit incorporates the Manual Reset Switch and provides protection to allow only PennDOT personnel to activate it, i.e., switch is in a locked box or enclosure and PennDOT personnel would have the key. The system can also be reset through digital communication, again protected against non-PennDOT personnel access. The specific processing flow for both the Sensor Unit and Receiver Unit are discussed in detail in the following sections.

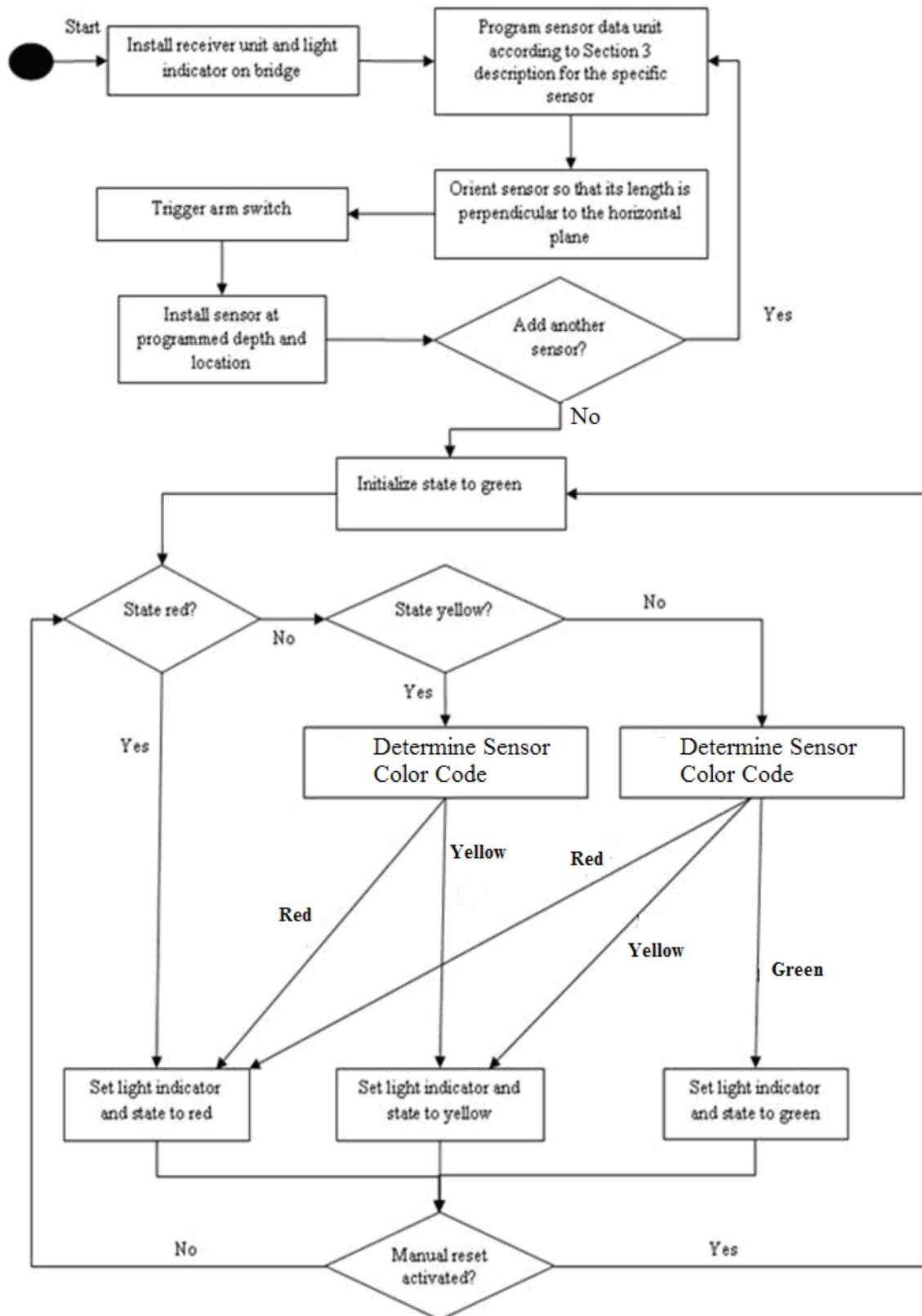


Figure 5: High-level System Flow Diagram

4.1 Sensor Unit Specifications

The specifications for the Sensor Unit device are presented in this section. The Sensor Unit is buried in the stream or riverbed. During a scour event the Sensor Unit floats to the surface of the water once the material covering the Sensor Unit has been removed by the scour. The Sensor Unit repeatedly transmits a message to the Receiver Unit. The Sensor Unit provides information as to the bridge it monitors, the nearest bridge structure ID, the serial number, and the Color Code of the Sensor Unit.

4.1.1 Sensor Unit Hardware Architecture

This sub-section presents the hardware architecture of the Sensor Unit. The figure below provides a top-level block diagram of the architecture of the system. This includes all major components for the Sensor Unit. Specifications for each of the major components in Figure 6 are presented in this section.

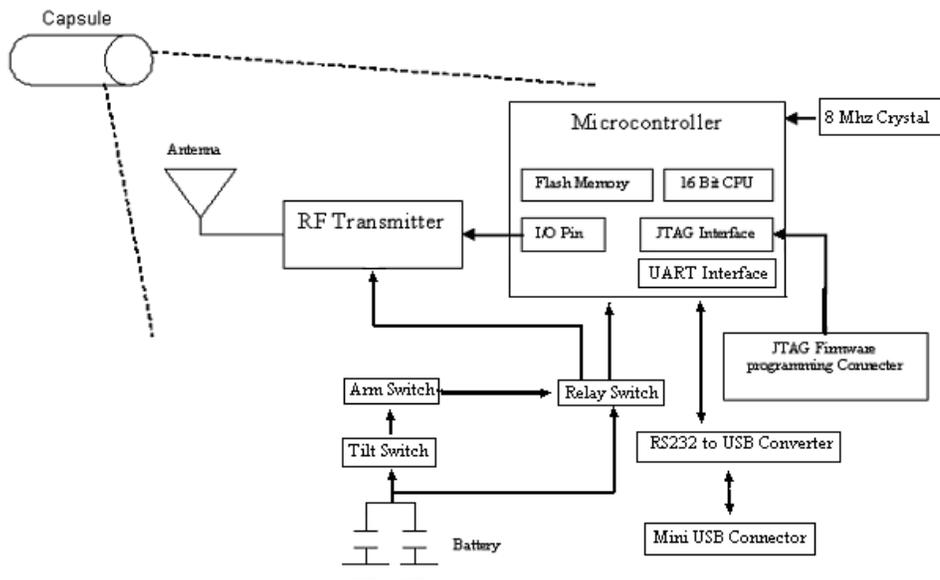


Figure 6: Top-level Block Diagram of the Sensor Unit

4.1.1.1 Microcontroller

The microcontroller controls the operation of the Sensor Unit. The microcontroller must wake-up after being triggered by tilt switches and transmit the Sensor Unit Data Block to the Receiver Unit. Thus, the microcontroller must read the Sensor Unit Data Block from non-volatile memory, setup and initialize the RF transmitter, and interface with the RF transmitter to send the Sensor Unit Data Block to the Receiver Unit. The microcontroller stores the program code (software) and the Sensor Unit Data Block in on-board non-volatile memory. The software is written in C (computer programming language). The microcontroller will be programmed through the Joint Test Action Group (JTAG) interface [13].

4.1.1.2 RF Transmitter

A commercial, off-the-shelf (COTS) RF transmitter transmitting at 433 MHz was used. The microcontroller interfaces with the RF transmitter and controls the operation of the transmitter. The RF transmitter is capable of transmitting at a data-rate of 10 Kbps (kilobits per second) and has a range of 3000 ft. The RF transmitter uses on-off keying (OOK) to modulate the data stream from the microcontroller. The RF transmitter connects to a COTS 50-Ohm antenna.

4.1.1.3 Relay Switch

A relay switch is used to form a permanent connection between the battery supply and the active components of the device. The need for a relay switch is due to the use of a tilt switch to activate the device. This tilt switch, however, cannot be relied upon to continually provide a steady connection as the orientation of the Sensor Unit will most likely be in a state of continual change when floating on the surface of the water. Once the tilt switch forms a closed circuit, the relay switch forms a permanent connection between the active components and the battery. This permanent connection is not broken until the Sensor Unit is *manually* reset using the external reset switch.

4.1.1.4 Arm Switches

The arm switch determines the switching direction of the relay switch. In an Armed or ON position, the arm switch connects the poles of the relay switch so that when voltage is applied, the relay switch forms a connection between the batteries and the rest of the active circuitry. In an Unarmed or OFF position, the arm switch connects the poles of the relay switch so that when voltage is applied, the relay switch forms an open circuit between the batteries and the rest of the active circuitry. To implement this arm switch, two devices were used. The first was a switch located on the PCB that can be used during testing and redeployment. The other was a switch embedded in the capsule and accessible externally. Once the external switch is implemented, the on-board switch can be eliminated from the design.

4.1.1.5 Tilt Switches

The tilt switches provide a connection between the arm switch and the battery supply. Each tilt switch is composed of two leads that can connect to a common electrode based on the orientation of the tilt switch. At 35 degrees from the horizon in either direction, the common electrode connects to one of the leads and the tilt switch acts as a short circuit. When the tilt switch is less than 35 degrees from the horizon in either direction the tilt switch acts as an open circuit. The Sensor Unit will be designed so that when the Sensor Unit reaches the surface, at least one of the multiple tilt switches are oriented to be at least 35 degrees from the horizon and thus act as a short circuit. As a result, the Sensor Unit activates (starts to transmit). Multiple tilt switches may be used to achieve this.

4.1.1.6 JTAG Connection

The JTAG Connection provides the means to transfer (write) the software and data to the microprocessor memory. This connection is also used to debug the system while in the testing phase.

4.1.1.7 RS232 to USB Converter

A COTS chip will be used to convert the serial RS232 communication used by the microcontroller to USB communication.

4.1.1.8 Mini-USB Connection

The Mini-USB Connection will be used to interface the device with a USB Connection of a host PC.

4.1.1.9 Battery

The battery will provide power to the Sensor Unit. The battery is a COTS and provides 3.0 VDC.

4.1.1.10 Antenna

The antenna is connected to the output of the RF transmitter. The antenna is a COTS device that is designed for 50-Ohm micro strip (trace on the printed circuit board (PCB)). The antenna conforms to the form factor and size of the Sensor Unit.

4.1.1.11 Capsule

The capsule is of a cylinder shape and is watertight. The capsule is buoyant and made of sufficiently strong material to prevent damage once activated (floating). The electronics will be contained within the capsule. An arm switch will be accessible from the outside of the capsule.

4.1.2 Sensor Unit Data Block

This section presents a detailed description of how the data items are stored in the Sensor Unit's non-volatile memory. This data is transmitted to the Receiver Unit. These items are in accordance with PennDOT labeling practices. The first three data items are:

1. Bridge Identification Number (BMS)
2. Serial Number
3. Location of Sensor
 - 3.0 Nearest Structure Unit ID – item 5D02 of BMS2 [1]
4. Color Code – this will be used to determine if bridge is in need of inspection or should be closed

The 14-digit Structure ID number that PennDOT uses for the BMS2 System (BMS2 item 5A01) is used as the format of the Bridge Identification Number [1]. The

Bridge Identification Number is designed to hold 14 numerical digits (1 byte per numerical digit) to accommodate the 14-digit Structure ID number.

The Serial Number is a unique identifier for each Sensor Unit stored at a particular bridge location. The Serial Number is stored in a 2-byte field allowing 65536 unique Serial Numbers.

The Location of Sensor Field consists of one data item used by the BMS2 system; the ID of the nearest structural unit (BMS2 item 5D02). The ID of the nearest structural unit is used to provide inspectors with an indication of where scour was detected. The nearest structure unit ID sub-field will be 4-bytes and is stored as four alphanumerical characters. Each alphanumeric character requires one byte of memory.

Table 1: Breakdown of the Location of Sensor Field

Location of Sensor Field				
Byte	1	2	3	4
Offset¹	0	1	2	3
Sub-Field	Nearest Structure Unit ID			

The Color Code is used to indicate the severity of the scour risk the Sensor Unit is monitoring and simplifies the Receiver Unit software. Thus, the Color Code depends on the depth the Sensor Unit is buried at and the location of the Sensor Unit. The depth each color Sensor Unit is buried at is specific to each bridge and is determined by PennDOT staff. The Color Code field is 1-byte representing four different colors. Each color represents a different depth (severity) of scour and typical interpretations are shown in Table 2. The Color Code Field is used by the Receiver Unit to determine what action to take, i.e., the LED indicator setting (green, yellow, orange, or red), when a Sensor Unit is detected (released from stream or riverbed). The Receiver Unit interprets the Color Code as defined in the following table.

Table 2: Interpretation and action resulting from the color code sub-field by the receiver unit.

Color Code	Integer Representation	Typical Interpretation	Receiver Unit Action
Green	0	+ 5 ft	Indicator: Green
Yellow	1	+ 3 ft	Indicator: Yellow
Orange	2	+ 1 ft	Indicator: Orange
Red	3	- 1 ft	Indicator: Red

The entire message has a 16-bit CRC appended to it for error detection. The CRC enables the Receiver Unit to detect and then discard any message containing errors. Adding the CRC adds two bytes to the length of the message (data packet). The CRC is computed as follows:

- CCITT polynomial ($x^{16} + x^{12} + x^5 + 1$) is used²
- The computation is initialized with zeros (0x0000)

The data is stored in the non-volatile memory of the Sensor Unit at specific locations. The data can start at any valid memory location if there are enough contiguous memory bytes to hold the entire data block. Otherwise, the start of data must be pushed

¹ Byte offset from start of the Location of Sensor Field.

² CCITT polynomial reference: ITU-T Recommendation V.41 (Blue Book)

back to a lower memory location. The data block consists of five fields with the Location of Sensor Field containing two sub-fields,

1. Bridge Identification Number (BMS)
2. Serial Number
3. Location of Sensor
 - a. Nearest Structure Unit ID – item 5D02 of BMS2 [1]
4. Color Code – used to determine if bridge is in need of inspection or should be closed
5. CRC-16 (error detection mechanism)

The length, in bytes, of the Sensor Unit data block is 23 bytes. The following table lists the lengths, in bytes, of each field in the data block.

Table 3: Length of the data fields stored in the Sensor Unit data block.

<u>Field Name</u>	<u>Number of Bytes</u>
Bridge Identification Number	14 Bytes
Serial Number	2 Bytes
Location of Sensor	4 Bytes
Color Code	1 Bytes
CRC-16	2 Bytes

The following table shows the layout of the Sensor Unit Data Block in memory.

Table 4: Sensor Unit Data Block

Byte	1	2	3	4	5	6	7	8
Offset	0	1	2	3	4	5	6	7
Field	Bridge Identification Number							
Sub-Field	N/A							
Byte	9	10	11	12	13	14	15	16
Offset	8	9	10	11	12	13	14	15
Field	Bridge Identification Number						Serial Number	
Sub-Field	N/A						N/A	
Byte	17	18	19	20	21	22	23	
Offset	16	17	18	19	20	21	22	
Field	Location of Sensor				Color Code	CRC-16		
Sub-Field	Nearest Structure Unit ID				N/A	N/A		

4.1.3 Sensor Unit Software Architecture

The software program controls the operation of the Sensor Unit. The software performs the following tasks; (1) setup and initialization the RF transmitter; (2) retrieval of the Sensor Unit Data Block from non-volatile memory; and (3) interfacing with the RF transmitter to transmit the Sensor Unit Data Block to the Receiver Unit.

The Sensor Unit-embedded software executes after the Sensor Unit is activated (tilted to a horizontal orientation) as a result of being released due to scour. Figure 7 shows the High-level Flow Chart for the Sensor Unit-embedded software. First, the RF

Transmitter must be initialized. The initialization of the RF transmitter is a simple process of asserting a few signals. Next, the software reads the Sensor Unit Data Block from non-volatile memory and converts that into the appropriate format expected by the RF transmitter. The CRC-16 is part of the Sensor Unit Data Block and is calculated before the Sensor Unit is deployed (buried). The software then outputs a synchronizing message and/or preamble to the RF transmitter, which precedes each message and informs the Receiver Unit that a message is present. The preamble is used to inform the receiver that a message is present and for the receiver to lock onto the message. Next, the software will output the contents of the Sensor Unit Data Block to the RF transmitter. The software will send the preamble and message, repeatedly, until the Sensor Unit is deactivated or loses power (battery dies).

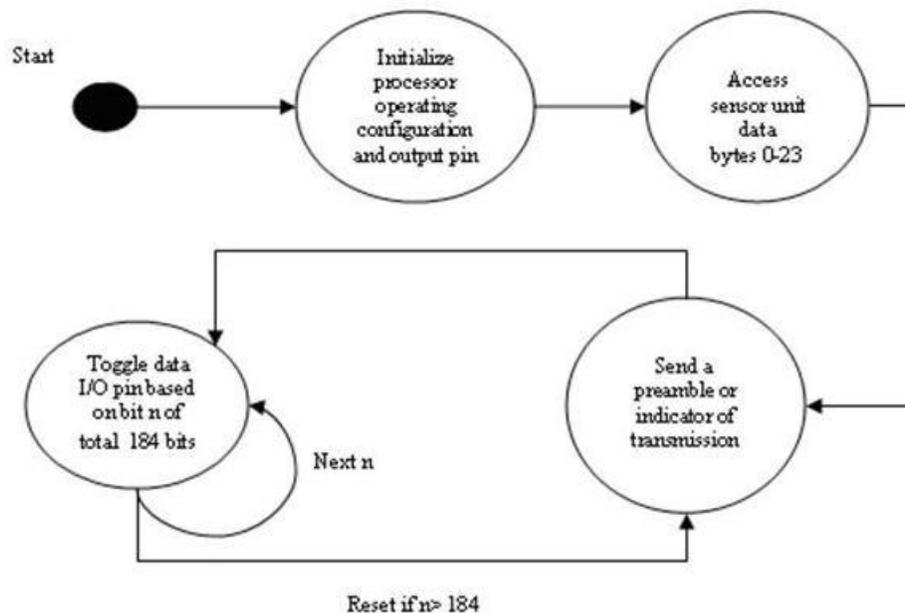


Figure 7: Sensor Unit Software

4.2 Receiver Unit Specifications

The Receiver Unit is mounted on the bridge structure and listens for RF transmissions sent by released Sensor Units. The Receiver Unit interprets and stores these messages. Finally, the Receiver Unit drives the Light Indicator based on the information it has collected.

4.2.1 Receiver Unit Hardware Architecture

The Receiver Unit is mounted on the bridge structure and controls a Light Indicator signifying if there is scour indicated by Sensor Unit release at the bridge. The

Receiver Unit (1) receives messages from the Sensor Units; (2) controls the Light Indicator; and (3) accepts input through Universal Asynchronous Receiver/Transmitter (UART) communication. The UART communication allows PennDOT personnel to program the Receiver Unit's Bridge ID, clear a green, yellow, orange, or red condition on the Light Indicator (scour has been detected), and read out (retrieve) collected Sensor Unit Data Blocks. Figure 8 shows the block diagram of the Receiver Unit, each of the main components is described in this section.

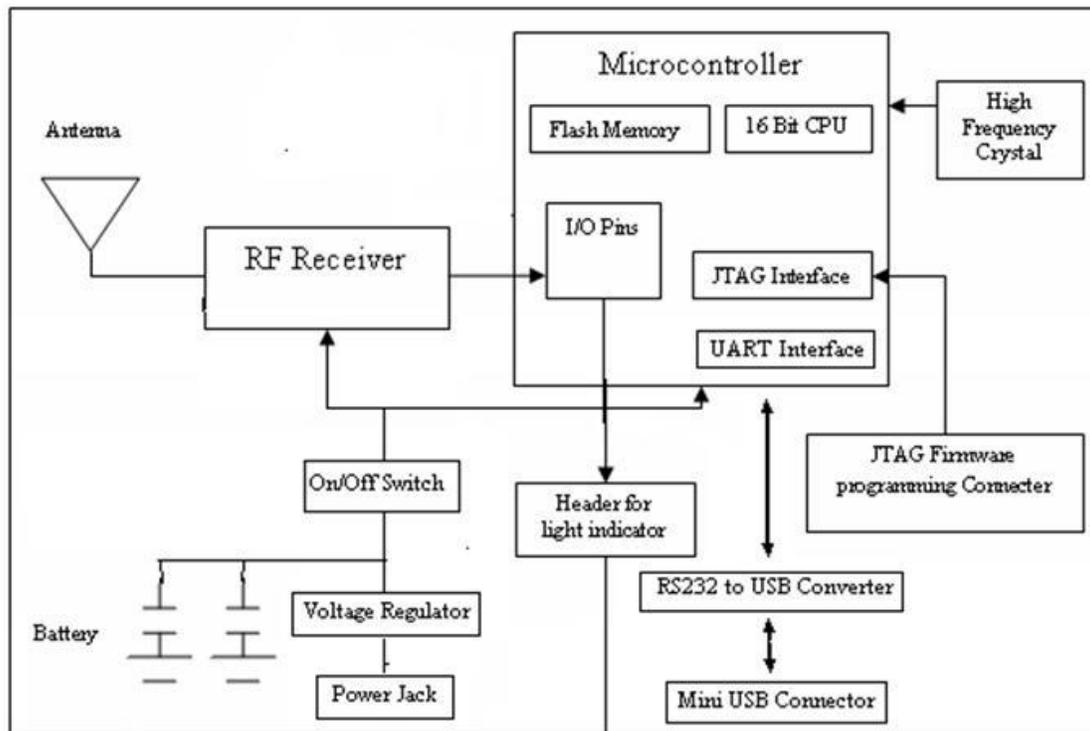


Figure 8: Receiver Unit Block Diagram

4.2.1.1 Microcontroller

The microcontroller contains the software program to interface with and control the other blocks that make up the Receiver Unit. The microcontroller interfaces with and controls the RF Receiver, and monitors the data output line of the RF Receiver for messages from Sensor Units. The data from the RF Receiver is a serial string of bits and is input into a single data I/O pin on the microcontroller. The microcontroller updates the state of the Light Indicator based on the data in the Sensor Unit message. The microcontroller uses four output pins to control the four lights of the Light Indicator. The microcontroller accepts input from the UART communication (inspector's laptop) interface, which is used to clear the Light Indicator condition. The Receiver Unit provides a means to download all received Sensor Unit Data Blocks to an inspector's computer or laptop. The software program is stored in non-volatile memory on-board the microcontroller. The software is written in C or assembly language. The microcontroller is programmed through the JTAG interface. The Receiver Unit microcontroller is programmed at manufacturing time.

4.2.1.2 On/Off Switch

The On/Off switch is a switch or button that connects or disconnects the Receiver Unit to power. The On/Off switch is physically protected, i.e., enclosed in a locked box from unauthorized use. Only PennDOT personnel have access to the On/Off switch. A locked enclosure facilitates this feature.

4.2.1.3 RF Receiver

The RF Receiver receives and decodes data transmitted by the Sensor Unit. The RF Receiver is a COTS device that is compatible with the RF transmitter used in the Sensor Unit. The RF Receiver demodulates an On-Off Keying (OOK) signal. The RF Receiver connects to a 50-Ohm antenna.

4.2.1.4 JTAG Connection

The JTAG connection provides the means to transfer (write) the software and data to the microprocessor's memory.

4.2.1.5 RS232 to USB Converter

A COTS chip is used to convert the serial RS232 communication used by the microcontroller to USB communication.

4.2.1.6 Mini USB Connection

The mini USB connection is used to interface the device with a USB connection of a host PC.

4.2.1.7 Header for Light Indicator

The header for the Light Indicator is a six pin header connecting the four microcontroller I/O pins assigned to the Light Indicator, a power, and a ground connection. Any signal conditioning required in converting between operating levels of the microcontroller and the lights will be completed in this block.

4.2.1.8 Battery

The battery provides power to the Sensor Unit. The battery is a COTS device and provides 3.0 VDC.

4.2.1.9 Antenna

The antenna is connected to the output of the RF transmitter. The antenna will be a COTS device that is designed for 50-Ohms. The antenna will conform to the form factor and size of the Receiver Unit.

4.2.1.10 Light Indicator

The Light Indicator contains four lights of red, orange, yellow, and green. The Light Indicator provides inspectors with a visual indication of the extent of scour events at the bridge. The inspectors can use this information to take further action. One possible use of this information is to use the red light to indicate that the bridge is not safe to cross. The yellow and orange lights to indicate that the bridge may be safe to cross with caution, and a green light or no light to indicate that the bridge is safe to cross. For initial prototyping purposes, a simple circuit of four light emitting diodes (LEDs) with the colors of red, orange, yellow, and green is used.

4.2.2 Receiver Unit Software Architecture

The Receiver Unit Software controls the operation of the Receiver Unit. The software performs the following tasks; (1) setting up and controlling the interface with the RF Receiver; (2) monitoring the data output of the RF Receiver for the Sensor Unit preamble and message; (3) decoding the Sensor Unit messages; (4) parsing the Sensor Unit message and updating the Light Indicator to reflect the scour severity reported in the Sensor Unit message; (5) taking input from the UART and performing the required action such as resetting the system.

The software verifies that the Sensor Unit Data Block was received without error by computing the CRC. The software discards all messages containing errors (invalid CRC). The software uses the color code of the Sensor Unit Data Block to determine what condition to set the system to. The software does not set the Light Indicator to a less severe condition in response to a Sensor Unit message. Thus, the Light Indicator may change from green to yellow, orange, or red, or from yellow to orange or red, etc., but not from yellow to green or orange to yellow or green, etc. Figure 9 shows the high-level flow of the Receiver Unit software.

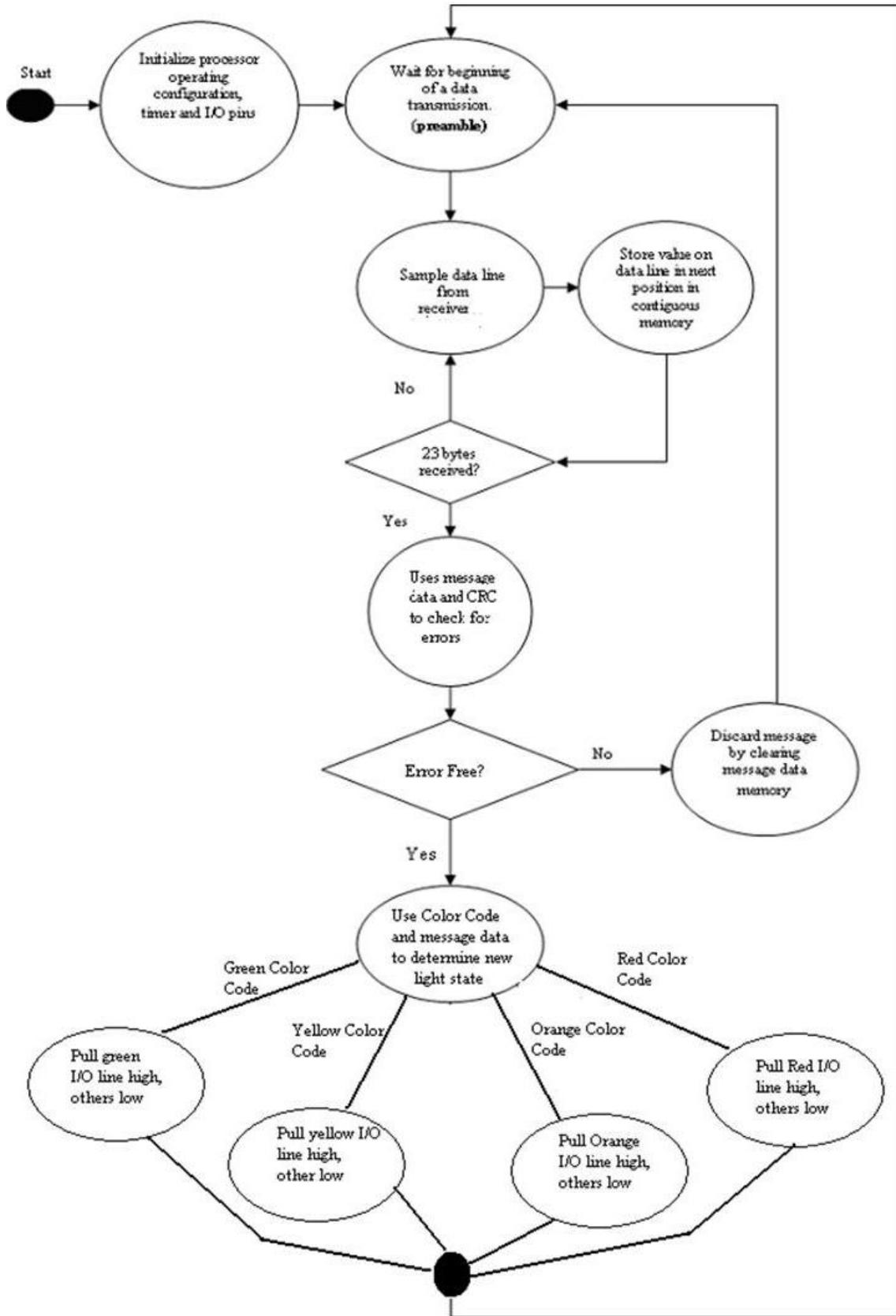


Figure 9: Flow Chart of the Receiver Unit Software.

4.3 Light Indicator Specifications

The Light Indicator provides a visual indication of the scour severity reported by the Receiver Unit. The Light Indicator is mounted on or near the bridge. The Light Indicator is visible to an inspector not physically on the bridge, i.e., from the bank. The Receiver Unit controls the Light Indicator.

4.3.1 Light Indicator Hardware Architecture

This sub-section presents the hardware architecture of the Light Indicator. The figure below provides a top-level block diagram of the architecture. This includes all major components for the Light Indicator. Specifications for each of the major components in Figure 10 are presented in this section.

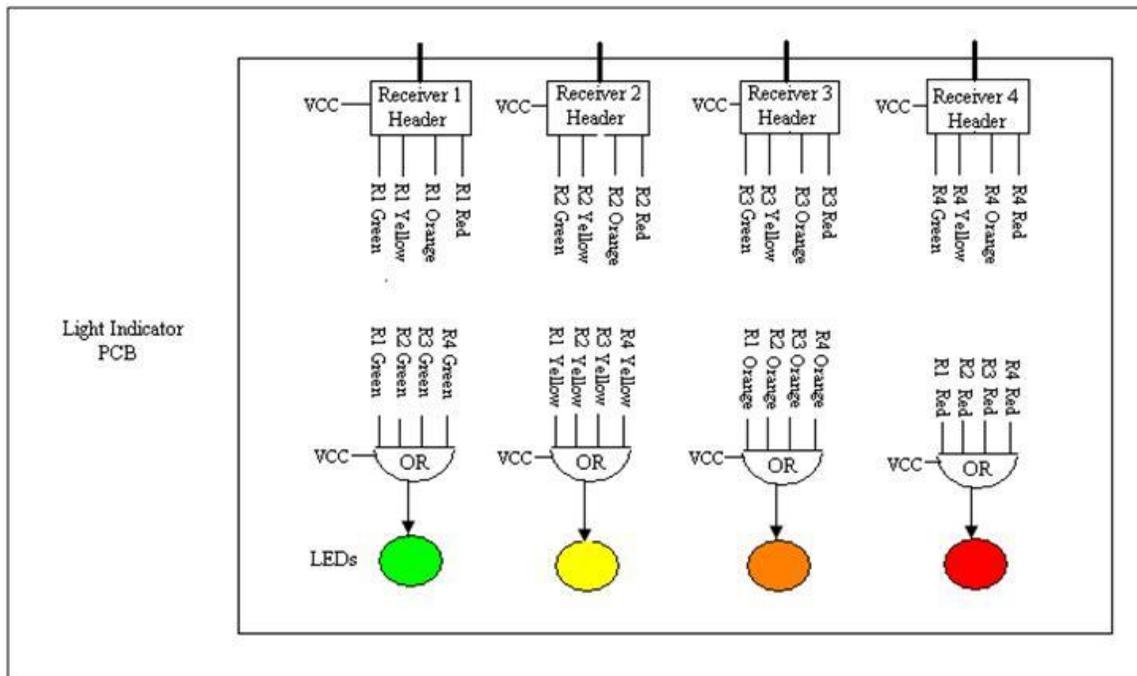


Figure 10: Light Indicator Hardware Diagram

4.3.1.1 6-Pin Header

Four 6-pin headers are located on the Light Indicator. Each takes input from a separate Receiver Unit. The case of multiple Receiver Units connected to a single Light Indicator would only occur if redundancy was required. Of the six pins on each header, four correspond to one of four LEDs, one is to connect to the power of the Receiver Unit, and one is to connect to the ground of the Receiver Unit.

4.3.1.2 OR Gate

Logical OR gates are used to drive the individual LEDs. Each OR gate connects to the inputs on the 6-pin header corresponding to the LED it is driving. Each OR gate performs the logical OR function on the four inputs. The logical OR function defines a single binary output that is a logical '1' (true) when one or more of the inputs is a logical '1' (true) and a logical '0' (false) when all inputs are a logical '0' (false).

4.3.1.3 LEDs

Four LEDs are used on the Light Indicator, one for each color. These LEDs are of the colors green, yellow, orange, and red. The LEDs chosen were three by Lite-On Inc. for the green, yellow, and red LEDs. For the orange LED, part SSL-LX5093SOC by Lumex Opto/Components Inc., was chosen.

5.0 Tether Option Analysis

This section describes the Tethered and Un-tethered Options for the Sensor Unit. The decision to tether or un-tether the Sensor Unit was based primarily on the ability of the Sensor Unit to rise to the surface and transmit the message before going out of range of the Receiver Unit on the bridge. The time required for the Sensor Unit to reach the surface is determined by the buoyancy force of the Sensor Unit, the drag on the Sensor Unit as it rises to the surface, and the distance (depth) of the Sensor Unit to the surface of the water (river/stream).

The following section details how the Sensor Unit rise time was computed. The following sub-sections detail the pros and cons of the Tethered and Un-tethered Options respectively. Based on the results, the Un-tethered option is recommended.

5.1 Transmitter –Receiver Range

The Linx 433 MHz transmitter-receiver pair has a range of 3000 feet [2], [3]. This value is used as an upper limit for the range in which a message can be transmitted and received. However, the conditions in which the system operates may differ greatly from those that produced the 3000 feet limit. These conditions include less than maximum transmitting power, as well as antenna and orientation factors.

The Friis Equation is used to determine the distance between the transmitter and receiver (range),

$$P_r = \frac{G_t G_r P_t \lambda^2}{4\pi d^2} \quad (1)$$

where P_r is the received power, P_t is the transmit power, G_r is the receiver antenna gain, G_t is the transmitter antenna gain, d is the distance between the transmitter and the receiver, and λ is the wavelength of carrier frequency. To simplify, both receiver and transmitter antenna gain are set to one. The transmit and receive power ranges specified by the radio frequency (RF) transmitter and receiver datasheets are used for P_t and P_r [2], [3]. The wavelength for a 433 MHz carrier frequency is 27.56 inches (0.7 meters). Using the parameters equation (1) is solved for the distance, d .

Given the range of output power, and received power, the resulting range of distances was calculated. The worst-case distance calculated was 731 feet.

5.2 Sensor Unit Rise Time Analysis

The calculated rise-time of the Sensor Unit from a given depth is presented in this sub-section. The rise-time calculations and assumptions are presented, followed by a description of the simulation process used to determine the rise-time. A simulation was used to model the drag force on the Sensor Unit. This sub-section concludes with a presentation of the calculated rise-times for a variety of depths, water velocities, and Sensor Unit dimensions (the radius of cylinder and the length of cylinder).

5.2.1 Buoyancy Calculation

The rise-time calculation is based on a combination of buoyancy and drag forces. The buoyant force must be greater than the force of gravity on the object in order for the object to rise. This is expressed as follows where a positive force exerts a downward (in the direction of gravity) force on the object,

$$F_{net-buoyant} = mg - \rho Vg \quad (2)$$

where, m is the mass of the object, g is the acceleration due to gravity, V is the volume of the object and ρ is the density of the fluid the object is in. The first term, mg , in (2) is the force due to gravity and the second term, ρVg , is the buoyant force of the object in a fluid having density ρ .

The capsule will be constructed out of Poly-Vinyl-Chloride (PVC) pipes. In order to compute the mass of the capsule the density of PVC will be multiplied by the volume of PVC pipe used to make the capsule; the mass of the electronics is neglected here. The capsule will have a thickness of 0.00256 m (0.1 inches) which is the minimum thickness of PVC [4]. The density of PVC is 1380 kg/m³ [5]. The surface area, A_S , of the capsule is found using Equation (3),

$$A_S = 2\pi r^2 + 2\pi rh \quad (3)$$

where r is the exterior radius of the cylinder, and h is the height (or length) of the cylinder. The volume of the capsule is computed as follows,

$$V = 2\pi r^2 h \quad (4)$$

where r is the exterior radius of the cylinder, and h is the height (or length) of the cylinder. Lastly, the acceleration due to gravity has a value of $g = 9.78045 \text{ m/s}^2$ and the density of water is 1000 Kg/m³. Together, these values can be used in the buoyancy formula presented above to produce a net force.

5.2.2 Drag Calculation

While buoyancy produces a net force for a given object within a fluid (water in this case), the net force will change depending on the drag force incurred by the object while moving through the water. There are two types of drag forces that can be encountered by the object. At faster velocities, the drag force can be determined using the Quadratic Drag Equation [5], [6]. At slower velocities, viscous forces dominate and drag force takes the form of viscous resistance [5], [6]. To determine which drag force is appropriate for a given set of circumstances, a value known as the Reynolds Number is used.

5.2.2.1 Quadratic Drag

Quadratic Drag is typical for objects moving through a fluid at a high velocity. The Quadratic Drag depends on a number of factors and is calculated using Equation (5),

$$F_d = \frac{AC_d\rho V^2}{2} \quad (5)$$

where F_d is the force due to drag, C_d is the drag coefficient, ρ is the density of the fluid (water in this case), V is the velocity of the object, and A is the cross-sectional area of the surface moving through the fluid. The drag coefficient is a unit-less number found by experimentation and a value of 1.17 is used for this study [5]. The cross-sectional area is the area of the top of the cylinder, or a circle of radius r (distance from center to exterior of the cylinder).

The velocity of the object changes as the object rises to the surface, requiring F_d to be recomputed. To effectively model the velocity of the capsule, a simulation was developed which steps through time in small discrete units (1 milli-second) updating the velocity and computing F_d for each unit of time. Using smaller discrete units of time i.e., 1 micro-second showed changes only in the fifth or more significant digit, so a 1 milli-second time-step was used.

5.2.2.2 Linear Drag

Linear drag occurs when the viscous force of the fluid is the dominant opposing force and is used for slow moving objects. Linear drag is computed using Equation (6),

$$F_d = 6\pi\eta rv \quad (6)$$

where, η is the viscosity of the fluid (water in this case), r is the Stokes radius (radius, center to exterior of capsule, of the capsule in this case), and v is the velocity of the object. The viscosity of water is 1.12 g/(m*s) [5]. The Stokes radius will be set to the radius of the cylinder. Linear drag, just as quadratic drag, depends on the velocity of the object. Again, a simulation model based on (6) was developed.

5.2.2.3 Reynolds Number

The Reynolds Number (Re) is used in the calculation of the drag coefficient and to characterize fluid flow conditions. Generally, large Reynolds numbers indicate that the quadratic drag equation should be used, whereas lower numbers indicate that the linear drag equation should be used. Reynolds Numbers up to 100 have the characteristics appropriate for the linear equation with $Re < 0.1$ being the more common boundary [5], [6]. The Reynolds Number can be computed for a cylindrical object using the following equation,

$$Re = \frac{v_s d}{\nu} \quad (7)$$

where v_s is the velocity of the object, d is the diameter of the cylinder, and ν is the absolute kinetic viscosity of the fluid the object moves through.

The Reynolds number for exterior diameters of 2 inches and 4 inches were computed for cylinders of the sizes and water velocities being investigated. A 4-inch exterior diameter was used to bound the Reynolds Number calculation even though the hollow stem auger interior diameter is only 3-3/16 inches. The results show that the Reynolds Numbers for both exterior diameters are larger than the boundary point defined in [5], hence the quadratic drag equation should be used. Simulations will be run using both the linear and quadratic drag equations to compute the rise time and the larger of the two times will be used, to be conservative. Table 5 shows the Reynolds Numbers for a representative set of velocities and capsule sizes.

Table 5: Reynolds Number for Diameters of 2 and 4 inches

Velocity	Exterior Diameter = 0.05 meter (2 inches)	Exterior Diameter = 0.1 meter (4 inches)
10 m/s (33 ft/s)	4.98E+05	9.96E+05
1 m/s (3.3 ft/s)	4.98E+04	9.96E+04
0.1 m/s (4 inch/s)	4.98E+03	9.96E+03
0.01 m/s (0.4 inch/s)	4.98E+02	996.0159

5.2.3 Simulation Method

A simulation was developed in MATLAB to calculate the velocity and then the drag force of the Sensor Unit as it rises to the surface. As the velocity of an object increases, the drag force will also increase.

The simulation divides time into discrete units (1 milli-second units) and computes the velocity and drag force for that unit of time. Then the simulation moves onto the next unit of time, until the Sensor Unit reaches the surface. The net force is computed as follows,

$$F_{net} = F_{net-buoyant} - F_d \quad (8)$$

where $F_{net-buoyant}$ is the net force due to buoyancy (upward) and F_d is the force due to drag (downward).

The first program, `constants_for_rise_time_calcs.m`, is used to set up the variables needed for the calculations. The inputs to this program are the radius and length of the Sensor Unit capsule. The parameters for density, water, gravity, the volume displaced by the object, the drag coefficient, the cross-sectional area of the object, and a starting depth and initialized or calculated within this program.

The second program, `calc_rise_time.m`, takes these variables as inputs along with a time interval, `delta_t`, for its computations. For this experiment, a `delta_t` of 1 milli-second was used. First, the initial F_{net} is computed with the velocity set to zero. If F_{net} is less than or equal to zero then the Sensor Unit is not buoyant and will not float to the surface, and the program will report an error and exit. However, if F_{net} is greater than zero the Sensor Unit is buoyant and will float to the surface, and the program will begin to loop (step) through time in steps of length `delta_t` (1 milli-second).

Within this loop, acceleration is first calculated from the net force calculated during the last iteration of the loop or the initial net for in the initial case. Given this acceleration, and the velocity of the last iteration, a new velocity is calculated for the next time interval. Using this velocity and the position of the Sensor Unit, the program then calculates the position of the Sensor Unit at the start of the next time interval. Next, the drag force is calculated from the velocity of the object at that time. Either the linear or quadratic drag equation can be used to compute F_d . Two programs; one for the linear drag and the second for quadratic drag were developed and used. The version of the program that performs linear drag is `calc_rise_time_viscous.m`. This simply replaces the quadratic equation with the linear one. The net force on the Sensor Unit, F_{net} , is computed as shown in Equation (8). Finally, the total time, current position, and current velocity variables are updated for the next iteration. Once the current position (vertical distance traveled) is larger than the depth that the Sensor Unit was buried, the total time calculated is considered rise time.

5.2.3.1 Simulation Code

This section contains the MATLAB code for the rise time simulations. Below is the code for the program that initializes the variables used in the calculation.

Calc_for_rise_time_constants.m

```
r = ((radius)*2.56)/100; %radius of sensor unit (single number in (s
is in inches) => m
h = ((height)*2.56)/100; %height of sensor unit (single number in (s
is in inches) => m

density_water = 1000*1e3; % g/(m^3)
water_dynamic_viscosity = 1.12;%(gm/m*sec)
density_polyvinyl_chloride = 1380000; % g/(m^3)
g = 9.78045; % m/(s^2)
volume_displaced = pi*r*r*h; % volume of sensor unit (cylinder) =>
(m^3)
```

```

C_D = 1.17; % drag constant => unitless
area = pi*r*r; % area of top of sensor unit cylinder => (m^2)
start_depth = 5; %distance between sensor unit and top of water => m
surface_area = 2*(pi*r^2) + (2*pi*r)* h; %surface area of a cylinder
wall_thickness = 0.00254;
mass = density_polyvinyl_cholride * surface_area * wall_thickness; %
mass of sensor unit => g

```

Below is the code to compute the rise time of the Sensor Unit using the Quadratic Drag Equation.

Calc_rise_time.m

```

function rise_time = calc_rise_time(density_water, g, volume_displaced,
C_D, area, start_depth, mass, delta_t)

% set initial conditions
rise_time = 0.0;
x_current = 0.0;
F_Drag = 0.0;
v_current = 0.0;
%calculate buoyant force - constant throughout
F_buoyant = (density_water*g*volume_displaced) - mass*g;
%calculate acceleration due to net Force, F_net
F_net = F_buoyant - F_Drag;
if (F_net < 0),
    disp('Under this set of parameter object will sink and loop will not
finish');
end

%LOOP
while( x_current < start_depth )
    % Calculate acceleration for this interval
    a_current = (F_net / mass);
    %%disp(sprintf("acc = %f\n",a_current));
    %Calculate new velocity
    v_new = v_current + (a_current*delta_t);
    % Calculate new position
    %%x_new = x_current + (a_current)*(delta_t^2);
    x_new = (v_new*delta_t) + x_current;
    % Calculate velocity for this interval
    %%v_current = (x_new - x_current) / (delta_t);
    % Calculate drag force
    F_Drag = (C_D*area*density_water*(v_current^2)) / 2;
    F_net = F_buoyant - F_Drag;
    % Update counters
    x_current = x_new;
    rise_time = rise_time + delta_t;
    v_current = v_new;
end % while( x_current > 0 )

end %calc_rise_time

```

Below is the code to compute the rise time of the Sensor Unit using the Linear Drag Equation.

Calc_rise_time_viscous.m

```
function rise_time =
calc_rise_time_viscous(density_water,water_dynamic_viscosity, r, g,
volume_displaced, C_D, area, start_depth, mass, delta_t)

% set initial conditions
rise_time = 0.0;
x_current = 0.0;
F_Drag = 0.0;
v_current = 0.0;
%calculate buoyant force - constant throughout
F_buoyant = (density_water*g*volume_displaced) - mass*g;
%calculate acceleration due to net Force, F_net
F_net = F_buoyant - F_Drag;
if (F_net < 0),
    disp('Under this set of parameter object will sink and loop will not
finish');
end

%LOOP
while( x_current < start_depth )
    % Calculate acceleration for this interval
    a_current = (F_net / mass);
    %%disp(sprintf("acc = %f\n",a_current));
    %Calculate new velocity
    v_new = v_current + (a_current*delta_t);
    % Calculate new position
    %%x_new = x_current + (a_current)*(delta_t^2);
    x_new = (v_new*delta_t) + x_current;
    % Calculate velocity for this interval
    %%v_current = (x_new - x_current) / (delta_t);
    % Calculate drag force
    %F_Drag = (C_D*area*density_water*(v_current^2)) / 2;
    F_Drag = 6 * pi * r *water_dynamic_viscosity *v_current;
    F_net = F_buoyant - F_Drag;
    % Update counters
    x_current = x_new;
    rise_time = rise_time + delta_t;
    v_current = v_new;
end % while( x_current > 0 )

end %calc_rise_time
```

5.3 Rise Time Results

The rise time results show rise times for a large variety of Sensor Unit sizes and depths. The results computed using the Quadratic Drag Equations are consistently larger than those computed using the Linear Drag Equations. Because the critical factor is to verify if the Sensor Unit will surface within range of the Receiver Unit and is based on the computed Reynolds Numbers, the results generated using the Quadratic Drag Equation are used.

5.3.1 Distance Traveled While Transmitting

The distance traveled by the Sensor Unit downstream is the primary concern for the tether/un-tethered decision. Given the time it takes the Sensor Unit to rise and the velocity of the stream, this distance is determined. However, the Sensor Unit also requires some amount of time to transmit the message once it has surfaced.

The river currents used are based on stream flow predictions made in 2002, estimating a maximum water velocity of approximately 23 ft/s [7]. A water velocity of 33 ft/s is used in this evaluation to allow for a conservative safety factor.

The time required for the Sensor Unit to transmit the message depends on the rate at which the data is transmitted, b , and the number of bytes (length) of the message, L ,

$$t_{msg} = 8bL_3 \tag{9}$$

The amount of data in the Sensor Unit data block is 23 bytes. The maximum data transfer rate, b , is 10 Kbps (kilobits per second). With these two values, the time it takes the transmitter to complete the transfer of one message can be determined. Table 6 shows the time it takes for this transfer at the maximum 10 Kbps and a slower 1 Kbps for the slow water velocity of 3 ft/s and the extreme water velocity of 33 ft/s.

Table 6: Message Transmission Time and Distance Traveled

Data Rate, b	10 Kbps	10 Kbps	1 Kbps	1 Kbps
Water Velocity	3 ft/s	33 ft/s	3 ft/s	33 ft/s
Time to Transmit (seconds)	0.0184	0.0184	0.184	0.184
Distance Traveled While Transmitting (feet)	0.0552	0.6072	0.552	6.072

Table 6 shows that the distance traveled while transmitting a message is 0.0184 feet to 6.072 feet. Therefore, the distance traveled by the Sensor Unit will be primarily based on the distance traveled while the sensor rises. The following three figures show the rise times for different capsule (cylinder) sizes (exterior diameters) and water velocities; 3 ft/s and 33 ft/s. The following figures show the distance away from the bridge that the Sensor Unit surfaces under different depths and different water velocities. Three different Sensor Unit capsule sizes are investigated. The minimum range is a

³ 1 byte contains 8 bits

rough approximation of the transmitter-receiver range under the worst-case conditions. The actual range in practice is at least 3000 feet.

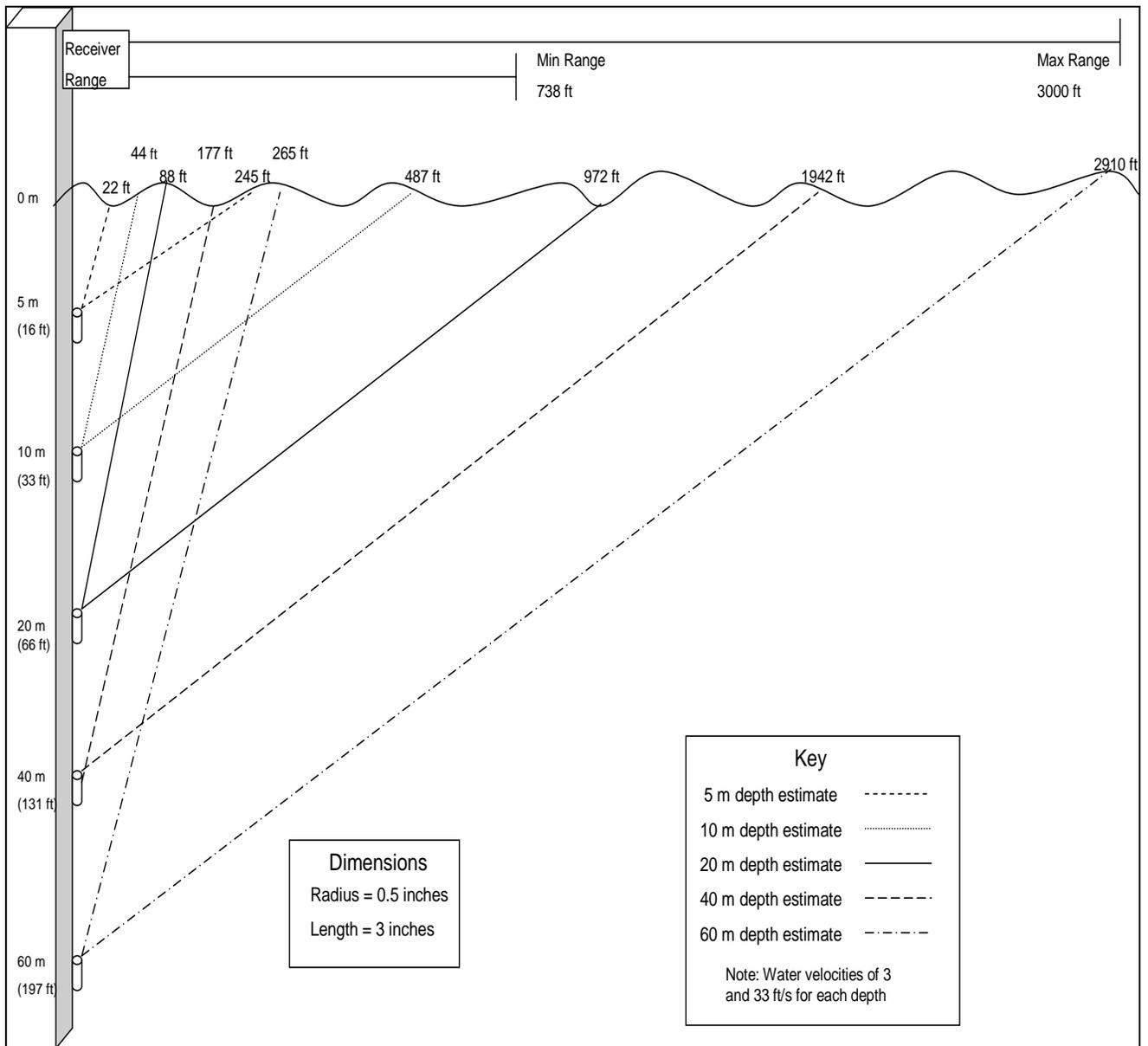


Figure 11: Travel distances for capsule with a radius of 0.5 inches and a length of 3.0 inches.

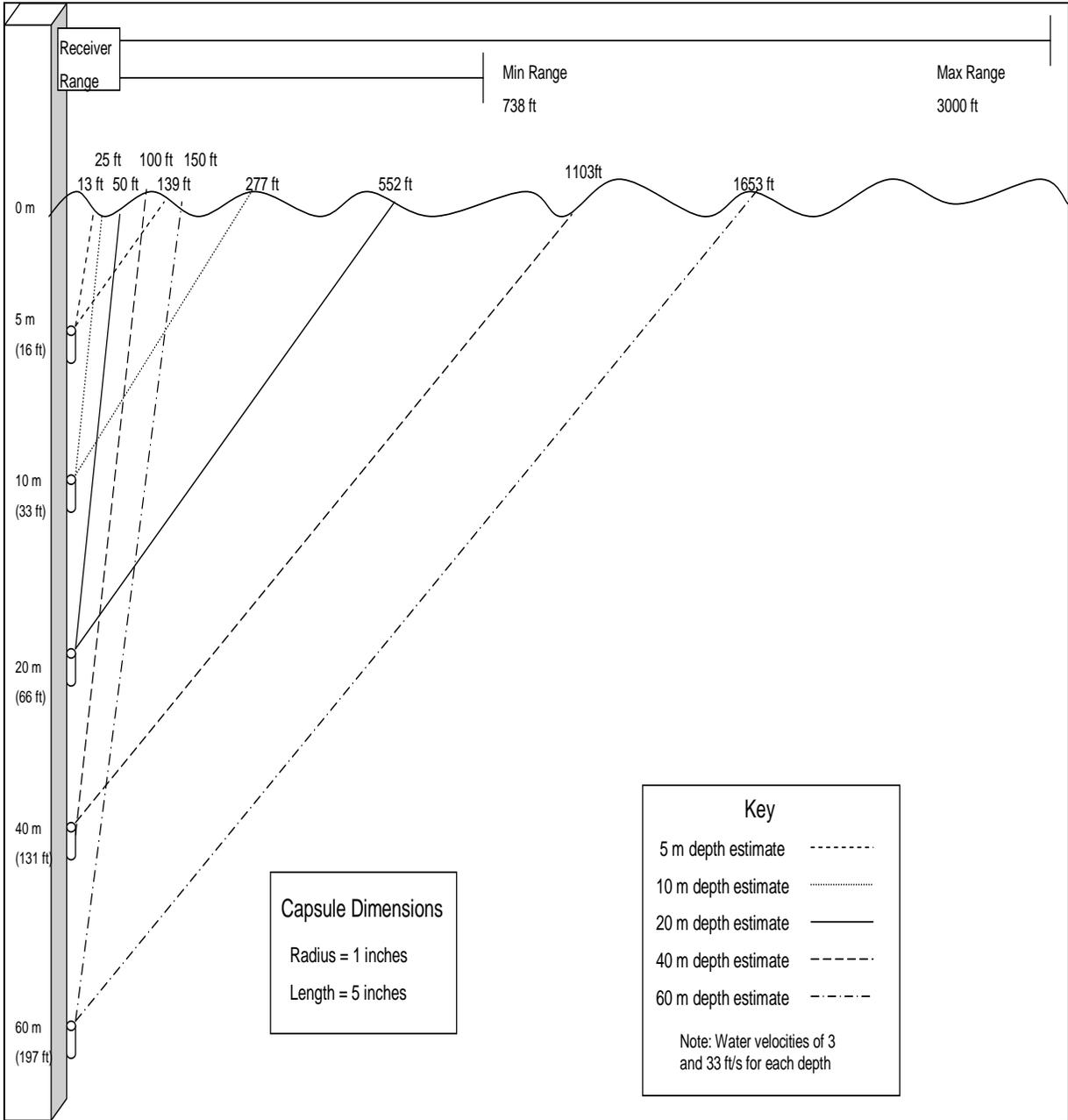


Figure 12: Travel Distances for Capsule with a Radius of 1.0 inch and a Length of 5.0 inches.

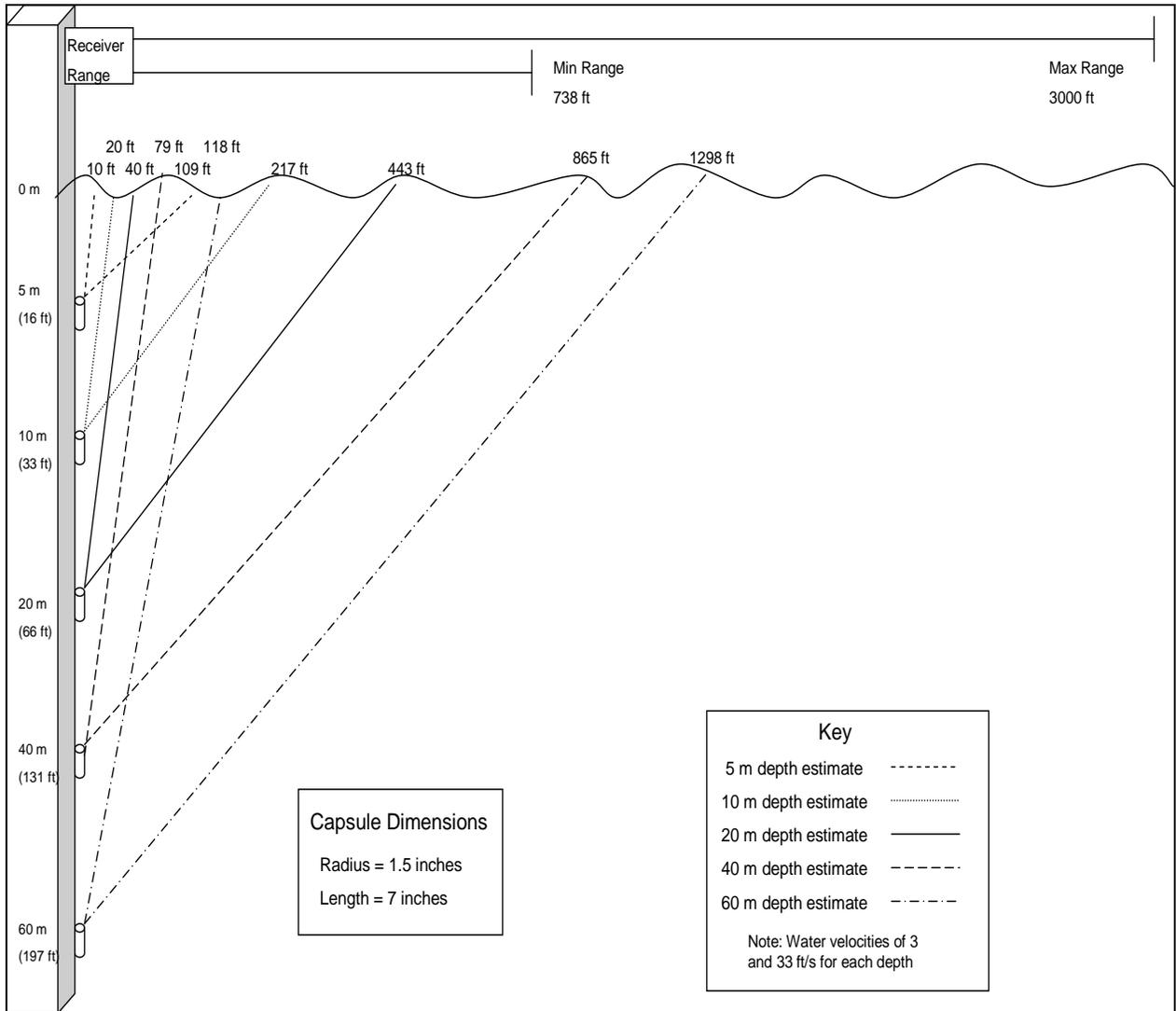


Figure 13: Travel Distances for Capsule with a Radius of 1.5 inches and a Length of 7.0 inches.

5.4 Tethered Option

The Tethered Option for the system calls for all Sensor Units to be anchored to the foundation area in order to limit their distance from the Receiver Unit. The reason for this implementation is to ensure reception by the Receiver Unit of the Sensor Unit Data Block. However, the Tethered Option introduces complexity for the installation and release of the Sensor Unit from the foundation. A possible implementation of this option is discussed in the first subsection. Additionally, the full range of pros and cons is summarized in the second subsection. For this option, the primary issue is substantiating

that the added complexity of system is warranted and necessary for proper system operation.

5.4.1 Description of Tethered Option Solution

Implementation of the Tethered Option requires two additional requirements to be met. First, the anchor must be placed in the foundation of the bridge in such a way and at a depth that it can remain in that position permanently. Second, the anchor must be connected to the Sensor Unit in such a way that the Sensor Unit can rise and activate properly while being restricted to a given area.

For installation, the Sensor Unit must be installed using standard NX (3-3/16 inch ID) hollow stem augers. Due to this size restraint, the anchor must be thin and simple. Therefore, one method to ensure that the anchor cannot not be moved by the flood event is to make it sufficiently heavy and buried deeply. If the anchor is buried below depths susceptible to scour, the only force pulling on the anchor should be the deployed Sensor Unit. Using the 33 ft/s maximum water velocity, the floating deployed Sensor Unit would produce maximum possible accelerations of 33 ft/s^2 . This is only slightly larger than gravity. Therefore, given a buried anchor with weight significantly larger than the Sensor Unit, one can assume that the anchor would be sufficiently unmoved by the Sensor Unit.

The installation and attachment of the tether to the Sensor Unit is another issue. Given the fact that using the steel auger would require placement of the anchor followed by soil and then placement of the Sensor Unit followed by soil, the length of the tether must be available at the depth that the Sensor Unit is buried. In one solution, the tether must be wound around a point on the Sensor Unit. This allows the Sensor Unit to unwind the tether as it rises.

5.4.2 Pros and Cons of Tethered Option

The Tether Option fixes the Sensor Unit in the vicinity of the bridge. Assuming that the tether does not break during a flood event, the Sensor Unit cannot go out of range of the Receiver Unit. Hence, the tether enables operation in events with extremely high water velocity. In addition, the Sensor Units would be easy to recover because they remain within the vicinity of the bridge.

The tether option introduces an additional potential failure point into the system, namely the tether getting hung up during installation or on debris during a flood event. Additionally, the tether could break during the flood, or the tether's anchor could be washed away. Finally, the tether introduces difficulty in the installation process because an anchor point and tether must be installed. This limits the potential locations for the Sensor Units because of the necessity of the anchor point.

5.5 Un-Tethered Option

In the un-tethered mode of operation, the Sensor Unit floats to the surface of the river/stream (water) and transmitting the message to the Receiver Unit. Here the critical factor is that the Sensor Unit is within range of the Receiver Unit long enough to transmit at least one message.

5.5.1 Description of Un-Tethered Option Solution

The Un-tethered Option would simply place the Sensor Unit at a particular depth below the surface of the streambed/riverbed. As the streambed/riverbed is removed during a scouring event the Sensor Unit is released, floats to the surface, and is carried downstream. It is improbable that every Sensor Unit could be economically recovered in the Un-tethered Solution Option because the search area (downstream and the floodplain) is too large to search.

5.5.2 Pros and Cons of Un-Tethered Option

Lack of a tether provides added freedom and flexibility in the deployment of the Sensor Units. No anchor point on an existing structure or the placement of a dedicated anchor point for the Sensor Units is required. Thus, deploying Sensor Units away from existing structures does not have the added cost of placing or connecting the Sensor Units to an anchor point. Furthermore, removing the tether removes a potential failure point in the system. For example, if the tether gets tangled or hung up during installation, or gets hung up on some debris during a flood event the Sensor Unit may never reach the surface to communicate with the Receiver Unit. Without the Tether Option, the Sensor Unit may move out of range of the Receiver Unit (too far downstream) before the complete message is transmitted. Calculations show that this should not be the case. Given stream velocities of 33 ft/s and a depth of approximately 195 feet, the Sensor Unit could travel 2910 ft. However, for normal flood event stream velocities and depths this would not be a problem. Furthermore, the Sensor Units float downstream indefinitely or are deposited on a floodplain after the flood event after they are released by the flood event. Hence, it is improbable that the Sensor Units could be economically recovered once they are released.

6.0 Design

The design of the Sensor Unit, Receiver Unit, and Light Indicator is described in the following sections. Individual schematics are shown for components followed by a detailed listing of their connections.

6.1 Sensor Unit Description of Operation

The first operational requirement of the Sensor Unit is the ability of the Sensor Unit to be *armed* upon installation. When the Sensor Unit is *armed*, the tilt switch controls the connection to the power supply (battery) of the Sensor Unit. Hence, when the Sensor Unit is *armed* and the tilt switch is asserted (activated – tilted to the horizontal orientation), the Sensor Unit transmits. The tilt switch is asserted when the Sensor Unit is horizontal after being released and floating on the surface of the water. Specifically, the tilt switch triggers a relay switch permanently connecting the power supply (battery) to the Sensor Unit electronics. The Sensor Unit then becomes active and transmits the Sensor Unit Data Block. Conversely, when the Sensor Unit is *not armed* (completely off), there is no path from the Sensor Unit electronics to the power supply (battery) of the Sensor Unit. Hence, when it is *not armed* and the tilt switch is asserted, the Sensor Unit remains inactive and does not transmit any data. In this inactive state, the Sensor Unit is not drawing power from the battery. The Sensor Unit is armed immediately before it is placed into the hollow-stemmed auger during installation. This requires the use of the arming switch located externally on the Sensor Unit. Prior to the arming of the Sensor Unit, the circuitry within the Sensor Unit should not be active and the Sensor Unit does not transmit under any circumstances. Upon installation, the arming switch sets the circuit to a state where its activation can be triggered at the correct orientation.

Once armed, the Sensor Unit is inactive while in a vertical orientation, and active while in a horizontal orientation (or *tilted* 35° from the vertical orientation). Vertical and horizontal/tilted orientation in this context means that the Sensor Unit length (end cap to end cap) is perpendicular to ground and water surface (vertical orientation) or parallel to the ground and water surface (horizontal orientation). The actuation of this orientation-sensitive activity is realized with three tilt switches. The switches are unconnected when in the Sensor Unit is in a vertical orientation and buried state as shown in Figure 14. All switches will form a 55-degree angle with the horizon in this state. In the horizontal orientation at least one of the tilt switches becomes active. The three tilt switches are positioned so that when the face of the PCB (PCB where the majority of the components are located) is furthest down or either side of the PCB is furthest down, one of the tilt switches will form a 35-degree angle below the horizon. This will cause one of the tilt switches to become active. The activation points for each tilt switch are shown in Figure 15, Figure 16, and Figure 17. The tilt switch activated is shown in red. All angles in between these three positions will also result in the triggering of a tilt switch. Given the

turbulent environment the Sensor Unit could be introduced to, this increases the certainty of the activation of the Sensor Unit.

The design is also based on the observed orientation the Sensor Unit floats in water due to the weight of the PCB and its components. The Sensor Unit has a tendency toward the orientation illustrated in Figure 17 due to the fact that the center of mass and turbulent waters would only cause a momentary divergence from this orientation. A fourth tilt switch could be added to cover the last axis of orientation. However, it is very unlikely that the sensor would be oriented in this position for any extended period of time. Laboratory tests have not shown this orientation to occur.

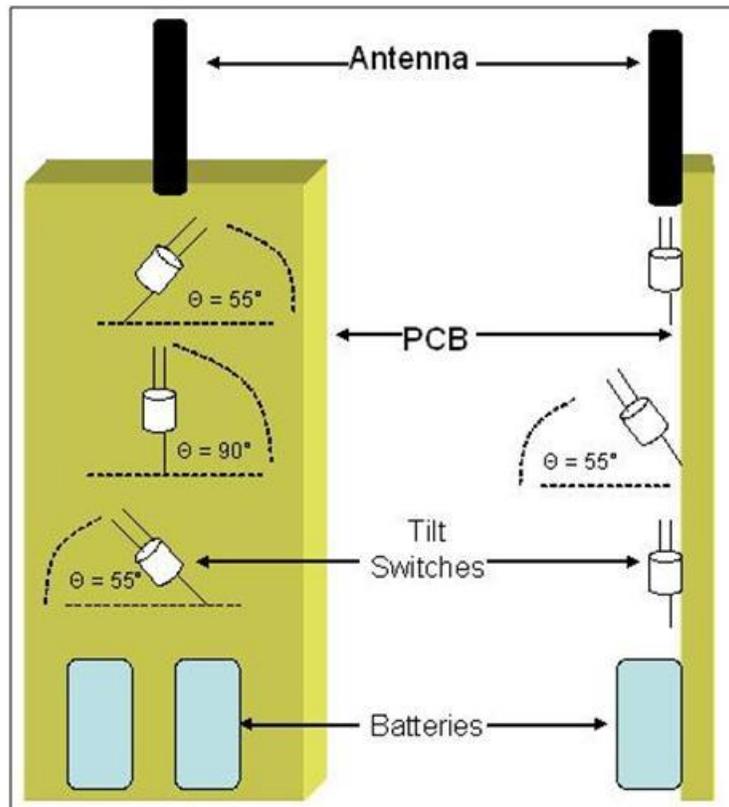


Figure 14: Tilt Switch Diagram

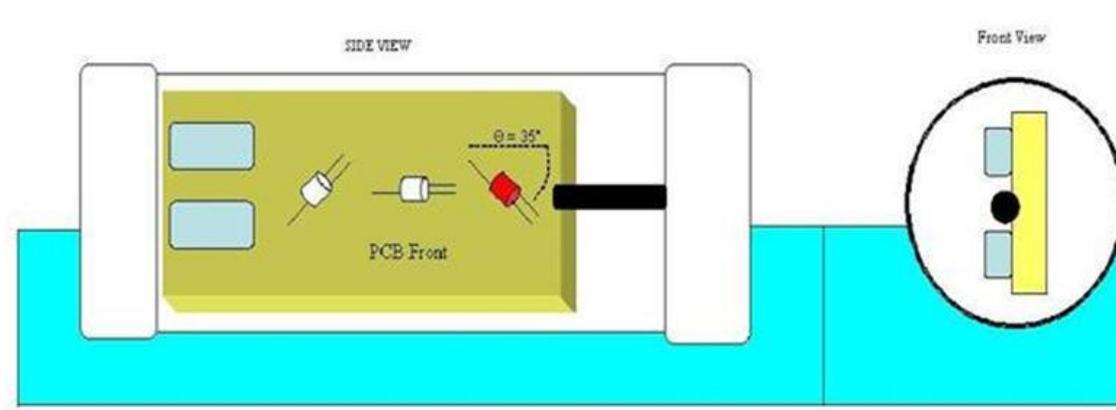


Figure 15: Activation Position of First Tilt Switch

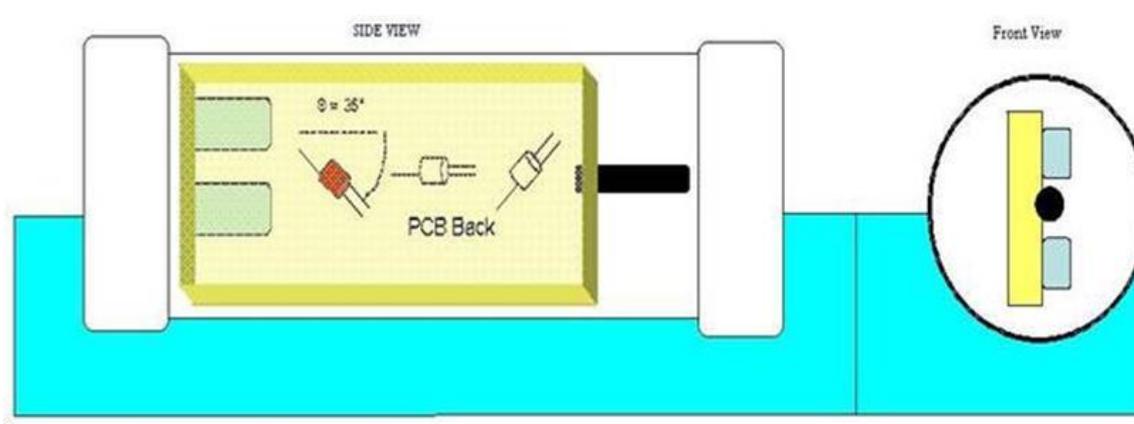


Figure 16: Activation Position of Second Tilt Switch

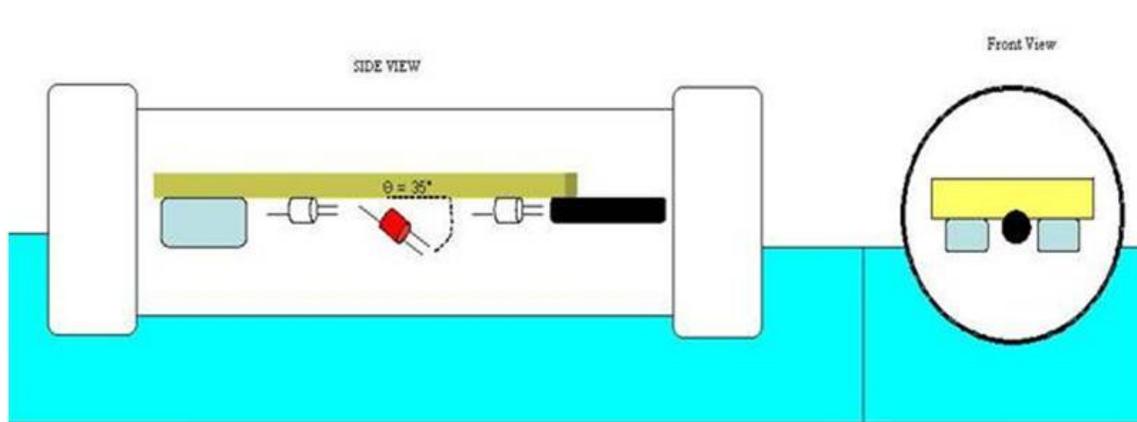


Figure 17: Activation Position of Third Tilt Switch (naturally oriented position)

If any tilt switch becomes active, the relay switch will be polarized (asserted) so that it connects the batteries to the power supply (V_{DD}) permanently. At this point, the microcontroller will start, and the program stored in the on-board flash memory will be executed. The basic operation of the program is to transmit a preamble and the data unit containing all data pieces outlined in the description of the Sensor Unit Data Block repeatedly. The Sensor Unit will transmit this repeated sequence until the batteries are drained or until the unit is retrieved. Calculations indicate that in most cases, the Sensor Unit will be well out of range when the batteries die.

6.2 Sensor Unit Design

This section describes the design of each of the functional blocks identified in the Sensor Unit specifications. The design of each functional block shows the components needed for that functional block and how that functional block is interfaced with other functional blocks.

6.2.1 Microcontroller

The microcontroller that is used for the Sensor Unit is the MSP430F2132. Figure 18 shows the schematic of the microcontroller used in the Sensor Unit design. An explanation of the pin connections and processor configuration follow the schematic.

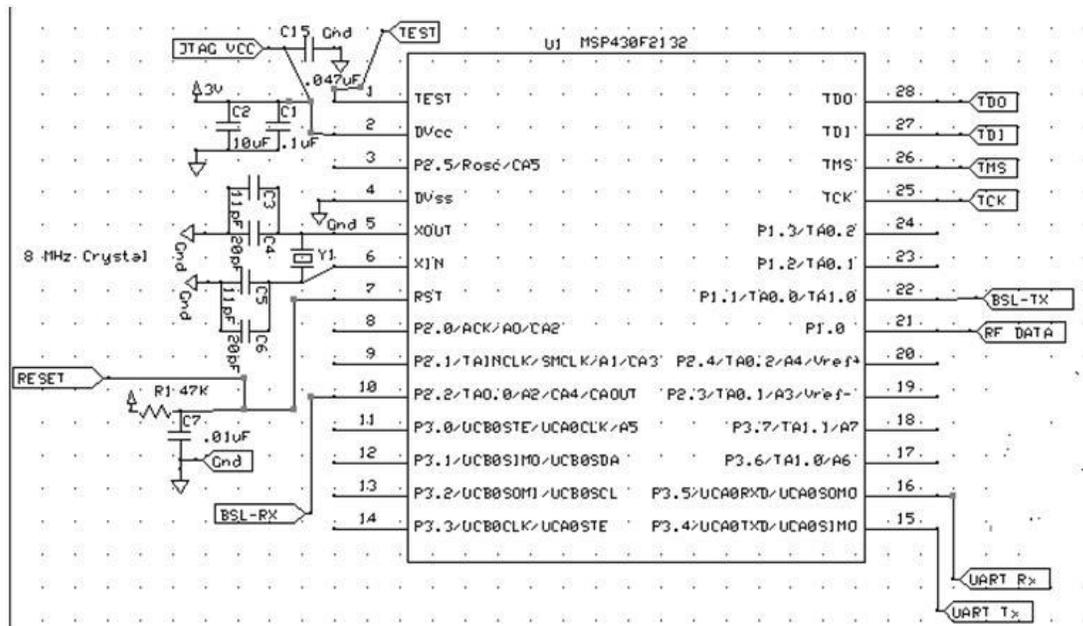


Figure 18: Schematic of the Microcontroller-MSP430F2132

The pins that are used and their descriptions are as follows:

Pin 1 – TEST: This pin is used to enable the standard 4-wire JTAG function. On the MSP430F2132 microcontroller, Port 1 is shared between an I/O function and a JTAG function. To enable the standard 4-wire JTAG function, this pin must be given a logic level of 1. This pin can also be given a high-low sequence to initialize the BSL operation. This pin is connected to the 14-pin header.

Pin 2 - DV_{CC}: This is the digital voltage supply pin. This pin connects to the power rail supplying 3 volts. Capacitors are used to eliminate noise. The capacitors C1 and C2 have values of 10 μF (C2) and 0.1 μF (C1) respectively, based on the hardware setup of MSP430 microcontrollers presented in [8]. This pin also connects to the 14-pin JTAG/BSL header for the purposes of powering the microcontroller directly from the header when trying to program. This also has a 0.047 μF capacitor (C15) to eliminate noise.

Pin 4 – DV_{SS}: This is the digital voltage supply pin with a negative reference. This pin is connected to ground.

Pin 5 – XOUT and Pin 6 – XIN: These two pins connect to the ends of the 8 MHz crystal (Y1). This oscillator is used to drive the clock of the microcontroller at its highest frequency. Capacitors (C3, C4, C5, and C6) are placed between the ends of the oscillator and ground given the load capacitance described in the crystal datasheet [9]. The values

of the capacitors are determined using a common equation for determining the value of these capacitors. The equation is:

$$C_{crystal} = 2 * C_{load} - C_{stray} \quad (10)$$

where, $C_{crystal}$ is the value of each of the two capacitors that are used with the crystal; C_{load} is load capacitance of the crystal specified in its datasheet as 18 pF; and C_{stray} is the capacitance present from the traces and input capacitance of the microcontroller. The typical value of C_{stray} is 5 pF. Given these values, $C_{crystal}$ is 31 pF ($C_{crystal} = 2 * 18 \text{ pF} - 5 \text{ pF} = 31 \text{ pF}$). Because 31 pF is not a standard value for COTS capacitors the 31 pF capacitor must be made using two capacitors in parallel to get a 31 pF capacitor. To obtain the 31 pF capacitor an 11 pF capacitor is put in parallel with a 20 pF capacitor to equal the required 31 pF capacitor. This structure is used for both terminals of the crystal (C3 and C4 make one 31 pF capacitor, while C5 and C6 make the second 31 pF capacitor).

Pin 7 - $\overline{\text{RST}}$: This pin is used as a reset pin. It is controlled by the JTAG/BSL header used for programming purposes. If the BSL is not necessary it can be connected to power by adding the resistor and capacitor. It is active low, therefore a pull-up 47 k Ω resistor (R1) circuit with a 0.01 μF capacitor (C7) will be used to keep this high. These values are based on a model hardware implementation provided from a MSP430 user guide [10].

Pin 10 – P2.2 RX: This pin is used as the data receive pin for the bootstrap loader within the microcontroller. This pin connects to Pin 12 of the JTAG/BSL connection.

Pin 15 – UCA0RXD: This pin is used as the receiver pin for the serial UART of the microcontroller. This pin connects to Pin 1 of the FT232RL chip.

Pin 16 – UCATXD: This pin is used as the transmit pin for the serial UART of the microcontroller. This pin connects to Pin 5 of the FT232RL chip.

Pin 21 – P1.0: This is an I/O pin on port P1 of the chip. This pin is configured to be an output pin. The output of this pin is used as an interface to the RF Transmitter chip.

Pin 22 – P1.1 TX: This pin is used as the data transmit pin for the bootstrap loader within the microcontroller. This pin connects to Pin 12 of the JTAG/BSL connection.

Pin 25 – TCK: This pin is the Test Clock of the JTAG connection. It synchronizes internal state machine operations. This pin connects to Pin 7 of the JTAG/BSL connection.

Pin 26 – TMS: This pin is the Test Mode State of the JTAG connection. It is sampled at the rising edge of TCK to determine the next state. This pin connects to Pin 5 of the JTAG/BSL connection.

Pin 27 – TDI: This pin is the Test Data In of the JTAG connection. It represents the data shifted into the device's test or programming logic. It is sampled at the rising edge of TCK when the internal state machine is in the correct state. This pin connects to Pin 3 of the JTAG/BSL connection.

Pin 28 – TDO: This pin is the Test Data Out of the JTAG connection. It represents the data shifted out of the device's test or programming logic and is valid on the falling edge of TCK when the internal state machine is in the correct state. This pin connects to Pin 1 of the JTAG/BSL connection.

6.2.2 RF Transmitter

The RF transmitter used in this design is the Linx Technologies TXM-433-LR Transmitter. Figure 19 shows a schematic of the design for the transmitter used in the Sensor Unit.

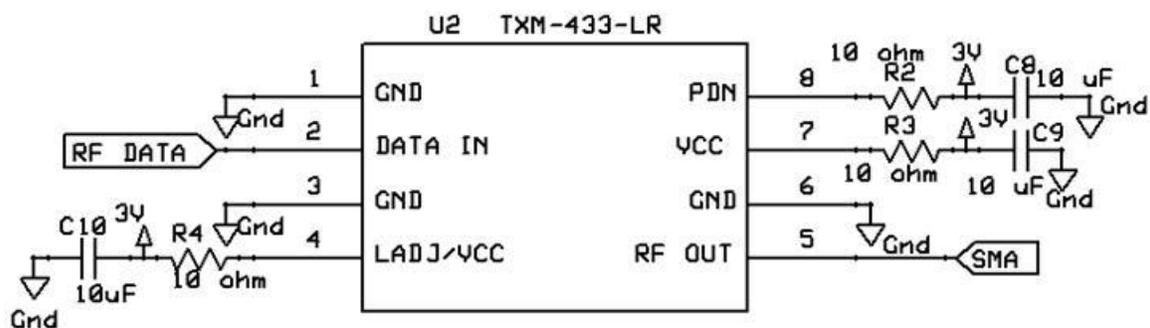


Figure 19: Schematic of the TXM-433-LR Transmitter

Pin 1, Pin 3, Pin 6 – GND: These pins are used to ground the chip. They are connected to the ground plane of the printed circuit board.

Pin 2 – DATA IN: This pin is used for the input of digital data to the chip. The MSP430 microcontroller drives this pin. The output of the transmitter is modulated based on the input of this pin.

Pin 4 – LADJ/VCC: This pin is the Level Adjust line of the chip. It allows the output power of the transmitter to be adjusted based on the voltage applied to this line. For our purposes, maximum range is desired. Therefore, to achieve maximum range, this pin should be connected to V_{DD} . As with the other lines connected to the power supply, a noise eliminating circuit is placed between the power and the pin. This circuit is composed of a 10 Ω resistor (R4) placed between the power and the pin as well as a 10 μ F capacitor (C10) placed between the power and the ground at that connection.

Pin 5 – RF OUT: This pin produces the modulated RF output of the chip. It is modulated between the carrier signal (on) and the absence of the carrier signal (off). This is connected to the Antenna through a 50-Ohm micro-strip that connects to a SMA connector.

Pin 7 – VCC: This pin is the Supply Voltage of the chip. It is connected to 3 VDC of power from the Batteries. This pin also has the noise eliminating circuitry as described for Pin 4.

Pin 8 - PDN: This pin is the Power Down pin of the chip. It is used to disable the chip if desired. In the case of our application, the chip does not need to be powered down. This pin powers down the chip if the line is pulled low. Therefore, this pin is kept high and connected to the 3 VDC power supply through a noise eliminating circuit.

6.2.3 Relay Switch

The relay switch used in this design is the Panasonic Electric Works' DS1E-SL-DC3V. Figure 20 shows a schematic of the design for the relay switch used in the Sensor Unit. An explanation of the pin connections and configurations follow this figure.

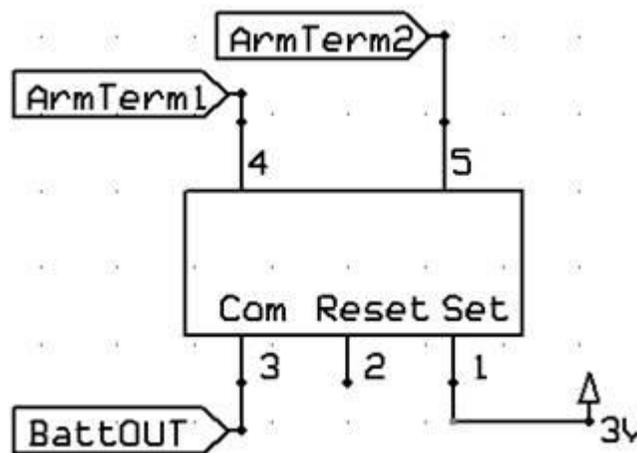


Figure 20: Schematic of the DS1E-SL-DC3V Relay Switch

Pin 1 – Set: This pin is used for a contact position that can be connected to common contact (Pin 3). The relay switch connects these two pins once a current with a voltage potential of 3.0 VDC is applied in the direction from Pin 5 to Pin 4. Once the on-board or external arm switch has changed so that current is applied in this direction, it will remain in this position until current is applied in the direction from Pin 4 to Pin 5. This pin will serve as the power rail (V_{DD}) for the circuitry following the switches.

Pin 2 – Reset: This pin is used for a contact position that can be connected to common contact (Pin 3). The relay switch connects these two pins once a current with a voltage potential of 3.0 VDC is applied in the direction from Pin 4 to Pin 5. Once the on board or external arm switch has changed so that current is applied in this direction, it remains in this position until current is applied in the direction from Pin 4 to Pin 5. This pin is not connected.

Pin 3 – Com: This is the common contact pin of the relay switch. It can either be connected to Pin 1 or Pin 2. The method for which contact this pin is connected to was detailed in the Pin 1 and Pin 2 descriptions. This pin is connected to the 3.0 VDC potential of the batteries.

Pin 4: This pin is the left terminal of the coil within the relay. It is connected to Pin 5 of the on-board and External Arm Switch. Depending on the output of these connections, this is either the negative or the positive terminal for the coil.

Pin 5: This pin is the right terminal of the coil within the relay. It is connected to Pin 2 of the on-board arm switch and external switch. Depending on the output of the arm switches, this is either the negative or the positive terminal for the coil.

6.2.4 Arm Switches

The arm switches are used to place the Sensor Unit in a state where it can either (1) be activated for transmission given the output of the tilt switches or (2) be reset such that the Sensor Unit is deactivated and does not transmit. Two options are available for this function. First, an on-board arm switch can be used for the purposes of arming and reset. The on-board arm switch is meant to be used during the testing phase of the PCB construction. After the board has been tested and is ready to be deployed, the external switch must be connected. When the external switch is connected, the on-board arm switch will no longer work properly and should be placed in the neutral position or removed.

6.2.4.1 On-Board Arm Switch

The on-board arm switch that will be used in this design is the NKK Switches of America Inc's AS23AP. Figure 21 shows a schematic of the design for the arm switch that will be used in the Sensor Unit. An explanation of the pin connections and configurations follow this figure.

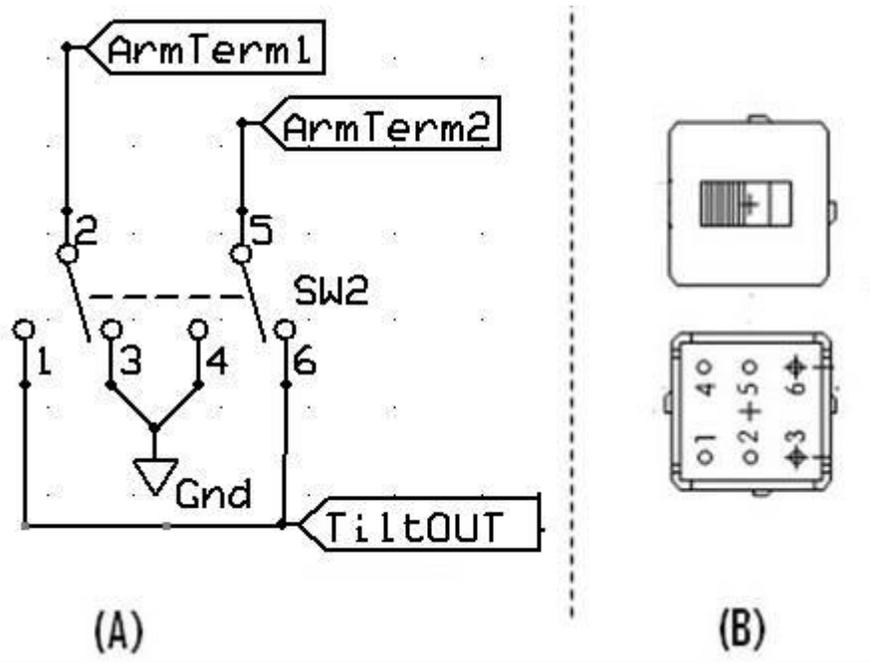


Figure 21: (A) On-Board Arm Switch Schematic and (B) On-Board Arm Switch Pin View

Pin 1: This is one of the two contacts that Pin 2 can connect to. It is connected when the actuator (mechanical slider part of the switch) of the switch is in the left position. The left position is when the actuator is over Pins 1 and 4 of the component as shown in Figure 21. When connected, this will connect Pin 4 of the relay switch to 3.0 VDC output from the tilt switches. If not connected, it serves as an open circuit.

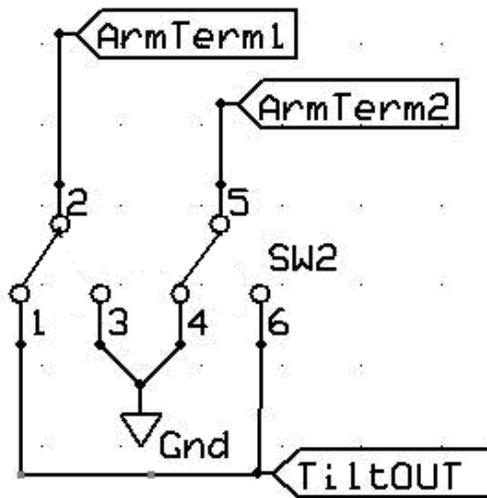
Pin 2: This is the common contact for Pins 1 and 3. In the left position (actuator over Pins 1 and 4), this pin connects to Pin 1 and will pass 3.0 VDC from the tilt switches to Pin 4 of the relay switch. In the middle position (actuator over Pins 2 and 5), this pin does not connect to any pin and is open. In the right position (actuator over Pins 3 and 6), this pin connects to Pin 3 and will connect Pin 4 of the relay switch to ground; this position resets the relay switch, resetting the Sensor Unit.

Pin 3: This is one of the two contacts that Pin 2 can connect to. It is connected when the actuator of the switch is in the right position. The right position is when the actuator is over Pins 3 and 6 of the component as shown in Figure 21. When connected, this will connect Pin 4 of the relay switch to ground; this causes the relay switch to reset, resetting the Sensor Unit. If not connected, it serves as an open circuit.

Pin 4: This is one of the two contacts that Pin 5 can connect to. It is connected when the actuator of the switch is in the left position. The left position is when the actuator is over Pins 1 and 4 of the component as shown in Figure 21. When connected, this connects Pin 5 of the relay switch to ground; this causes the relay switch to reset, resetting the Sensor Unit. If not connected, it serves as an open circuit.

Pin 5: This is common contact for Pins 4 and 6. In the left position (actuator over Pins 1 and 4), this pin connects to Pin 4 and connects Pin 5 of the relay switch to ground. In the middle position (actuator over Pins 2 and 5), this pin does not connect to any pin and is open. In the right position (actuator over Pins 3 and 6), this pin connects to Pin 6 and will pass 3.0 VDC from the tilt switches to Pin 5 of the relay switch; this causes the relay switch to reset, resetting the Sensor Unit.

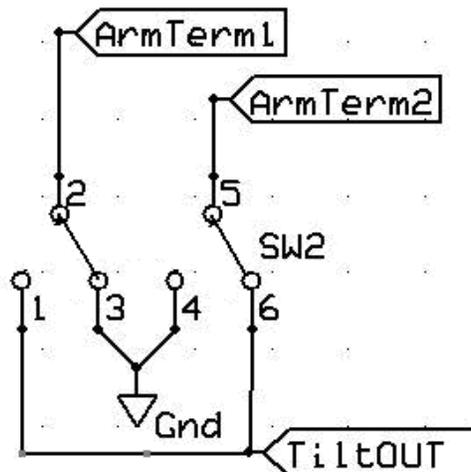
The on-board Arm Switch can set the Sensor Unit to either the Armed State or the Reset State. The Sensor Unit must be in the Armed State to respond to a tilt event (vertical orientation to horizontal orientation). The schematic of the on-board Arm Switch when in the Armed State is illustrated in Figure 22.



Armed State

Figure 22: Schematic Showing the Switch Connections for the On-board Arm Switch When in the Armed State (left position).

The Reset State is used before the Sensor Unit is deployed or to reset a Sensor Unit that has been triggered (tilted to a horizontal orientation). The schematic of the on-board Arm Switch when in the Reset State is illustrated in Figure 23.



Reset State

Figure 23: Schematic Showing the Switch Connections for the On-board Arm Switch When in the Reset State (right position).

6.2.4.2 External Arm Switch

The external arm switch used is part 16311-SW by the MPJA, Inc. This part is DPDT toggle switch. It should be emphasized here again that once the external switch is connected, the on-board arm switch must either be removed or placed in its middle position. The on-board arm switch is for PCB level testing only and does not provide a

watertight seal, while the external arm switch is for production and field use. **Figure 24** shows a schematic of the design for the external arm switch. This is nearly identical to the on-board switch other than the toggle mechanism. An explanation of the pin connections and configurations follow this figure. When the external switch is in the reset state, a current will be continually applied to the relay switch. The reset switch is only used to reset the Sensor Unit before setting the Sensor Unit to the armed state (arming the Sensor Unit). Once armed, the Sensor Unit is inactive (drawing no current) while vertical, but activates once the Sensor Unit is tilted horizontally (active and transmitting).

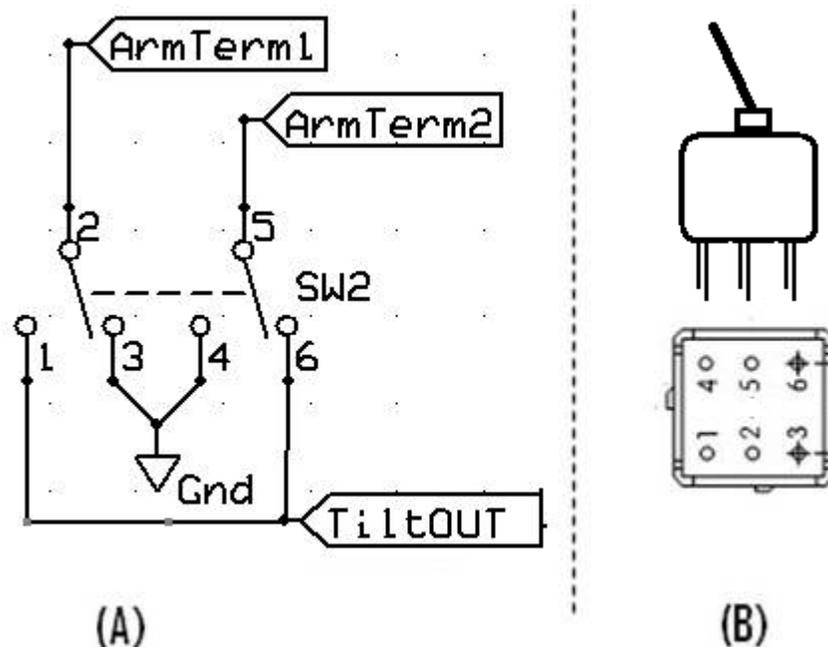


Figure 24: (A) External Arm Switch Schematic and (B) External Arm Switch Pin View

Pin 1: This is one of the two contacts that Pin 2 can connect to. It is connected when the actuator of the switch is in the left position. The left position is when the actuator is over Pins 1 and 4 of the component as shown in Figure 24. When connected, this connects Pin 5 of the relay switch to 3.0 VDC output from the tilt switches. If not connected, it serves as an open circuit.

Pin 2: This is the common contact for Pins 1 and 3. In the left position (actuator over Pins 1 and 4), this pin connects to Pin 1 and will pass 3.0 VDC from the tilt switches to Pin 5 of the relay switch. In the middle position (actuator over Pins 2 and 5), this pin does not connect to any pin and is open. In the right position (actuator over Pins 3 and 6), this pin connects to Pin 3 and will connect Pin 5 of the relay switch to ground.

Pin 3: This is one of the two contacts that Pin 2 can connect to. It is connected when the actuator of the switch is in the right position. The right position is when the actuator is over Pins 3 and 6 of the component as shown in Figure 24. When connected, Pin 5 of the relay switch is connected to ground. If not connected, it serves as an open circuit.

Pin 4: This is one of the two contacts that Pin 5 can connect to. It is connected when the actuator of the switch is in the left position. The left position is when the actuator is over Pins 1 and 4 of the component as shown in Figure 24. When connected, Pin 4 of the relay switch is connected to ground. If not connected, it serves as an open circuit.

Pin 5: This is common contact for Pins 4 and 6. In the left position (actuator over Pins 1 and 4), this pin connects to Pin 4 and connects Pin 4 of the relay switch to ground. In the middle position (actuator over Pins 2 and 5), this pin does not connect to any pin and is open. In the right position (actuator over Pins 3 and 6), this pin connects to Pin 6 passing 3.0 VDC from the tilt switches to Pin 4 of the relay switch.

6.2.5 Tilt Switches

The tilt switch used in this design is the Comus Group of Companies S1234. Figure 25 shows a schematic of the design for the tilt switch used in the Sensor Unit. An explanation of the pin connections and configurations follow this figure.

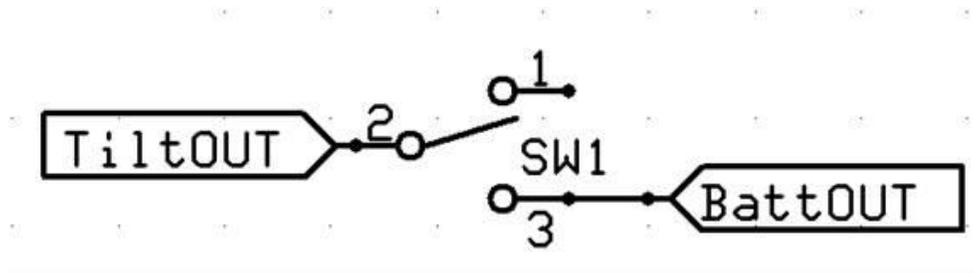


Figure 25: Schematic of the S1234 Tilt Switch

The pins of the tilt switch and their descriptions are provided below:

Pin 1: This pin will connect to Pin 2 when the Sensor Unit is oriented in a vertical position. In this case, a ground connection is provided by the tilt switch.

Pin 2: This is the common contact of the tilt switch. When the Sensor Unit is in a vertical position it connects to Pin 1 and when the Sensor Unit is in a horizontal position it will connect to Pin 3. If connected to Pin 1, it provides a ground connection. If connected to Pin 3, it will provide the 3.0 VDC potential of the batteries.

Pin 3: This pin connects to Pin 2 when the Sensor Unit is oriented in a horizontal position. In this case, the positive terminal of the batteries supply 3.0 VDC to the output of the tilt switch.

6.2.6 Voltage Regulator

The voltage regulator used in this design is the LM3940. Figure 26 shows a schematic of the design for the voltage regulator used in the Sensor Unit. An explanation of the pin connections and configurations follow this figure.

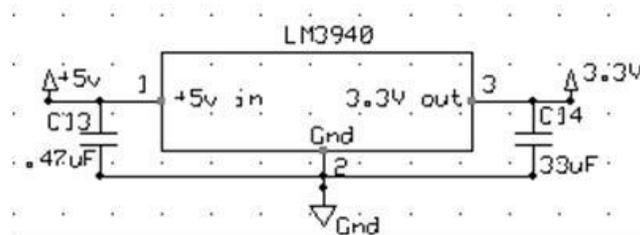


Figure 26: Schematic of the LM3940 Voltage Regulator

Pin 1 – +5Vin: This pin is a 5.0 VDC source input, however, inputs as low as 4.5 VDC can be used. A 0.47 μ F capacitor is connected between this pin and ground as per the datasheet for the voltage regulator [11].

Pin 2 – GND: This pin is connected to a common ground.

Pin 3 – +3.3Vout: This pin provides a 3.3 VDC output. It requires a 33 μ F capacitor connected between this pin and ground as per the datasheet for the voltage regulator [11].

6.2.7 JTAG/BSL Connector

The JTAG/BSL connection used is through 0.1-inch pitch header pins provided by Molex/Waldom Electronics Corp. Figure 27 below shows a schematic of the design for the JTAG/BSL header used in the Sensor Unit. An explanation of the pin connections and configurations follow this figure.

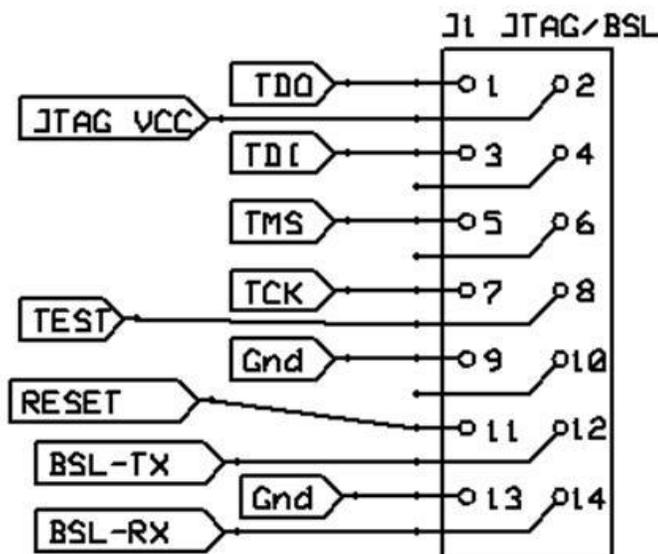


Figure 27: Schematic of JTAG/BSL Header

The pins of the JTAG/BSL Header and their descriptions are provided below:

Pin 1: This pin is the Test Data Out of the JTAG connection. It represents the data shifted out of the device's test or programming logic and is valid on the falling edge of

TCK when the internal state machine is in the correct state. This is connected to Pin 28 of the microcontroller.

Pin 2: This pin is the VCC pin of the JTAG connection. It is used to provide power to the processor. The voltage of 3.0 VDC must be set in the IDE (Integrated Development Environment) for proper operation.

Pin 3: This pin is the Test Data In of the JTAG connection. It represents the data shifted into the device's test or programming logic. It is sampled at the rising edge of TCK when the internal state machine is in the correct state. This is connected to Pin 27 of the Microcontroller.

Pin 5: This pin is the Test Mode State of the JTAG connection. It is sampled at the rising edge of TCK to determine the next state. This is connected to Pin 26 of the microcontroller.

Pin 7: This pin is the Test Clock of the JTAG connection. It synchronizes internal state machine operations. This is connected to Pin 25 of the microcontroller.

Pin 8: This pin is used to control the Test pin of the microcontroller. This connects to Pin 1 of the microcontroller.

Pin 9, 13: This pin is available to provide a common ground between the host computer/laptop and the board.

Pin 11: This pin is used to control the Reset pin of the microcontroller. This is Pin 7 of the microcontroller.

Pin 12: This is the transmit data output pin from the bootstrap loader of the microcontroller. It is connected to Pin 22 of the microcontroller.

Pin 14: This is the receive data input pin to the bootstrap loader of the microcontroller. It is connected to Pin 10 of the microcontroller.

All other pins are unconnected and unused.

6.2.8 Mini USB Port

The Mini-USB port used on the Sensor Unit is part H2961CT-ND by Hirose Electric Co. Ltd. Figure 28 below shows a schematic of the Mini-USB. An explanation of the pin connections follows this figure.

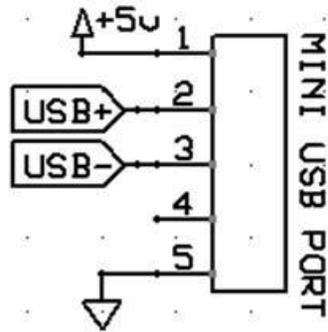


Figure 28: Mini USB Port Schematic

The pins of the mini USB port and their descriptions as follows:

Pin 1: This pin supplies 5.0 VDC from the host device (external PC or laptop). The USB interface provides a +5 VDC line as part of the interface. It connects to Pin 1 of the voltage regulator.

Pin 2: This pin is connected to the USB positive terminal of the host machine. It also connects to Pin 15 of the serial to USB chip (FT232RL).

Pin 3: This pin is connected to the USB negative terminal of the host machine. It also connects to Pin 16 of the serial to USB chip (FT232RL).

Pin 4: This pin is not used.

Pin 5: This pin connects the host machine to the ground of the PCB.

6.2.9 Batteries

The batteries used in this design are Panasonic –BSG’s CR-2/BE batteries. Figure 29 shows a schematic of the design for the batteries used in the Sensor Unit. An explanation of the pin connections and configurations follow this figure.

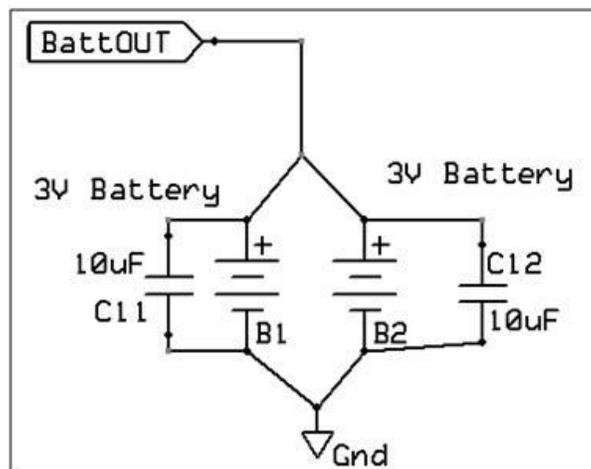


Figure 29: Schematic of the CR-2/BE Batteries

Displayed in Figure 29 are two instances of the CR-2 batteries. They are placed in parallel alignment in order to increase the capacity of the batteries. Each battery has a positive and negative terminal. The positive terminals have 3.0 VDC potential and are connected to a line that connects to Pin 2 of the tilt switches. The negative terminals serve as the negative reference and are connected to the ground. Capacitors are also placed in parallel with the batteries to increase the stability of the batteries' power output.

6.3 Receiver Unit Design

This section takes the functional blocks described previously and displays the design of each. The design of each functional block shows the components needed for that functional block and the how that functional block is interfaced.

6.3.1 Microcontroller

The microcontroller used for the Receiver Unit will be the MSP430F2132. Figure 30 shows a schematic of the microcontroller design used in the Receiver Unit design. An explanation of the pin connections and configurations follows the schematic. Notice that the pins in use have one label while the used pins have multiple descriptions. This is because each pin may have more than one optional function so only the used pins have a specified function.

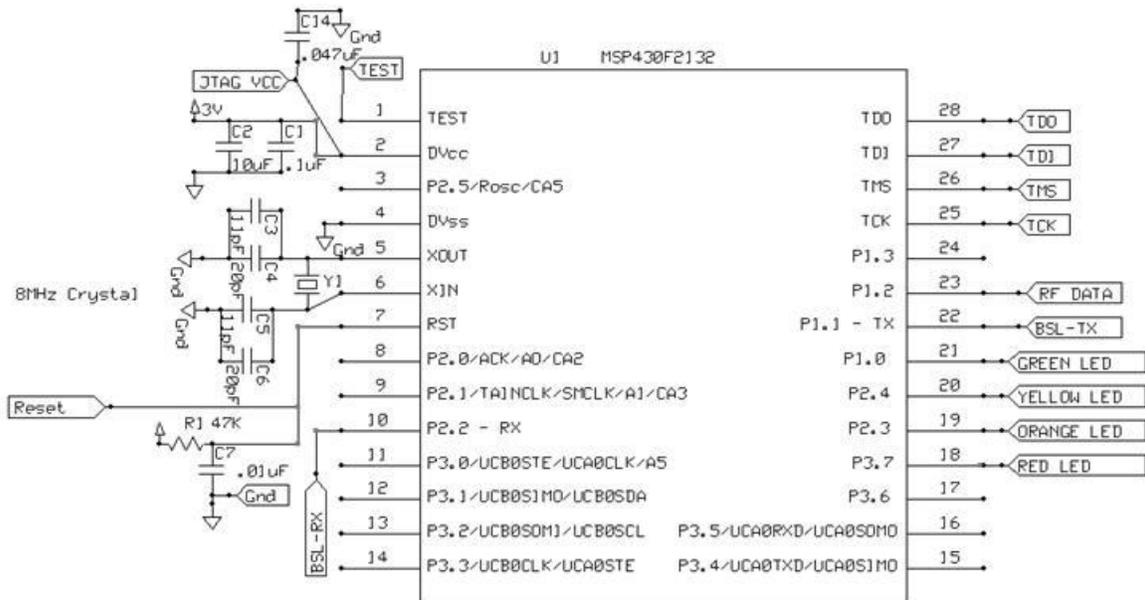


Figure 30: Schematic of Microcontroller-MSP430F2132

The pins that are used and their descriptions are described below:

Pin 1 – TEST: This pin is used to enable the standard 4-wire JTAG function. On the MSP430F2132 microcontroller, Port 1 is shared between an I/O function and a JTAG function. To enable the standard 4-wire JTAG function, this pin must be given a logic level of 1. This pin must also be given a high-low sequence to initialize the BSL operation. This pin is connected to the 14-pin header.

Pin 2 - DV_{CC}: This is the digital voltage supply pin. This pin connects to the power rail supplying 3 volts. Capacitors are used to eliminate noise. The capacitors C1 and C2 have values of 0.1 μ F and 10 μ F respectively, based on the recommended hardware setup for MSP430 microcontrollers presented in Appendix A of [10]. This pin also connects to the 14-pin JTAG/BSL header for the purposes of powering the microcontroller directly from the header when trying to program. This also has a capacitor (C15) to eliminate noise.

Pin 4 – DV_{SS}: This is the digital voltage supply pin with a negative reference. This pin is connected to the ground.

Pin 5 – XOUT and Pin 6 – XIN: These two pins connect to the ends of the 8 MHz crystal (Y1). This oscillator is used to drive the clock of the microcontroller at its highest frequency. Capacitors (C3, C4, C5, and C6) are placed between the ends of the oscillator and ground given the load capacitance described in the crystal's datasheet. The values of the capacitors are determined using a common equation for determining the value of these capacitors. The equation is:

$$C_{crystal} = 2 * C_{load} - C_{stray} \quad (11)$$

where $C_{crystal}$ is the value of each of the two capacitors that are used with the crystal; C_{load} is the load capacitance of the crystal specified in its datasheet as 18 pF; and C_{stray} is the capacitance present from the traces and input capacitance of the microcontroller. This is usually accepted to be around 5 pF. Given these values, $C_{crystal}$ is 31 pF ($C_{crystal} = 2 * 18 \text{ pF} - 5 \text{ pF} = 31 \text{ pF}$). Because a 31 pF capacitor is not a standard value for COTS capacitors the 31 pF capacitor must be made using two capacitors in parallel to get a 31 pF capacitor. To obtain the 31 pF capacitor an 11 pF capacitor is placed parallel with a 20 pF capacitor to equal the required 31 pF capacitor. This structure is used for both terminals of the crystal (C3 and C4 make one 31 pF capacitor, while C5 and C6 make the second 31 pF capacitor).

Pin 7 - $\overline{\text{RST}}$: This pin is used as a reset pin. It is controlled by the JTAG/BSL header for programming purposes. If the BSL is not necessary, it can be connected to power by adding the resistor and capacitor. It is active low and therefore a pull-up 47 k Ω resistor (R1) circuit with a 0.01 micro Fared capacitor (C7) will be used to keep this high. These values were based on a model hardware implementation provided from a MSP430 user guide [10].

Pin 10 – P2.2 RX: This pin is used as the receive data pin for the bootstrap loader within the microcontroller. This pin connects to Pin 12 of the JTAG/BSL connection.

Pin 18 – P3.7: This is an I/O pin on port P3 of the chip. This pin is configured to be an output pin. The output of this pin is used to drive the state of the red LED on the Light Indicator Component.

Pin 19 – P2.3: This is an I/O pin on port P2 of the chip. This pin is configured to be an output pin. The output of this pin is used to drive the state of the orange LED on the Light Indicator Component.

Pin 20 – P2.4: This is an I/O pin on port P2 of the chip. This pin is configured to be an output pin. The output of this pin is used to drive the state of the yellow LED on the Light Indicator Component.

Pin 21 – P1.0: This is an I/O pin on port P1 of the chip. This pin is configured to be an output pin. The output of this pin is used to drive the state of the green LED on the Light Indicator Component.

Pin 22 – P1.1 TX: This pin is used as the data transmit pin for the bootstrap loader within the microcontroller. This pin connects to Pin 12 of the JTAG/BSL connection.

Pin 23 – P1.2: This is an I/O pin on port P1 of the chip. This pin is configured to be an input pin. The input on this pin comes from the RF Receiver chip. The data is processed to find Sensor Unit Data Blocks when they are transmitted.

Pin 25 –TCK: This pin is the Test Clock of the JTAG connection. It synchronizes internal state machine operations. This pin connects to Pin 7 of the JTAG/BSL connection.

Pin 26 – TMS: This pin is the Test Mode State of the JTAG connection. It is sampled at the rising edge of TCK to determine the next state. This pin connects to Pin 5 of the JTAG/BSL connection.

Pin 27 – TDI: This pin is the Test Data In of the JTAG connection. It represents the data shifted into the device's test or programming logic. It is sampled at the rising edge of TCK when the internal state machine is in the correct state. This pin connects to Pin 3 of the JTAG/BSL connection.

Pin 28 – TDO: This pin is the Test Data Out of the JTAG connection. It represents the data shifted out of the device's test or programming logic and is valid on the falling edge of TCK when the internal state machine is in the correct state. This pin connects to Pin 1 of the JTAG/BSL connection.

All other pins are unconnected and unused.

6.3.2 RF Receiver

The RF Receiver used in this design is the Linx Technologies RXM-433-LR Receiver. Figure 31 below shows a schematic of the design for the RF Receiver used in

the Receiver Unit. An explanation of the pin connections and configurations follow this figure.

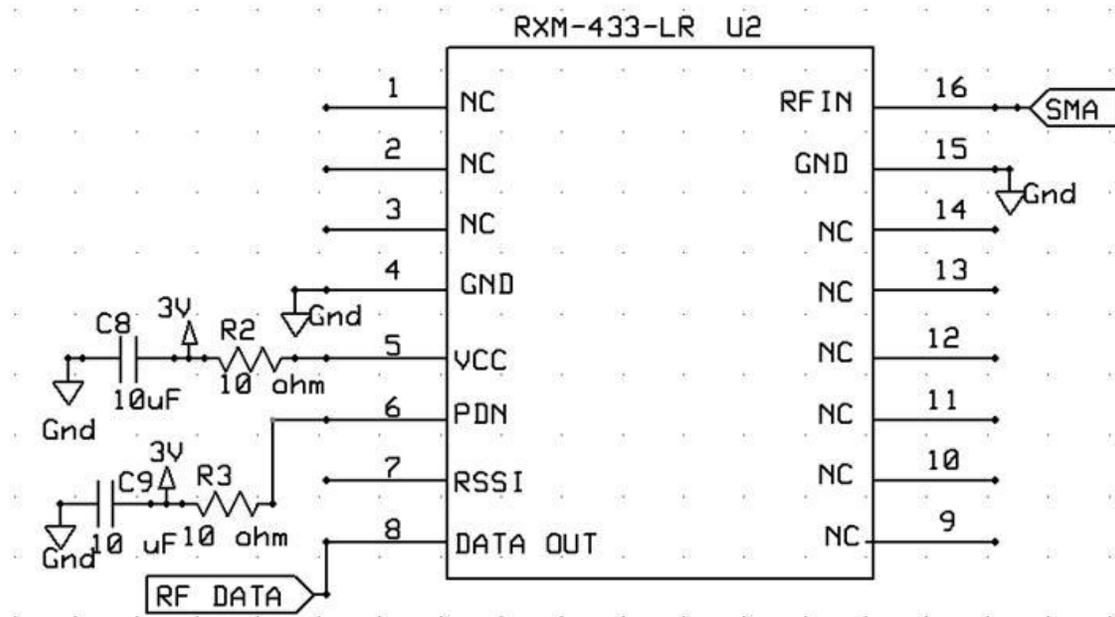


Figure 31: Schematic of RXM-433-LR Receiver

The pins of the receiver chip and their descriptions are provided below:

Pin 1, 2, 3, 9, 10, 11, 12, 13, 14 – NC: These pins have no connection and no function specified.

Pin 4, 15 – GND: These pins are used to ground the chip. They are connected to the ground of the PCB.

Pin 5 –VCC: This pin is the supply voltage of the chip. It is connected to 3 VDC of power from the power jack or batteries. As with the other lines connected to the power supply, a noise-eliminating circuit will be placed between the power and the pin. This circuit is composed of a 10 Ω resistor (R8) placed between the power and the pin as well as a 10 μ F capacitor (C8) placed between the power and the ground at that connection.

Pin 6 – PDN: This pin is the Power Down pin of the chip. It is used to disable the chip if desired. In the case of our application, the chip does not need to be powered down. This pin causes the device to power down if the line is pulled low. Therefore, this pin is kept high and connected to the 3.0 VDC power supply through a noise-eliminating circuit.

Pin 7 – RSSI: This is the Received Signal Strength Indicator pin. It provides an analog voltage representing the strength of the received signal. It will not be used in this design.

Pin 8 – DATA OUT: This pin provides the demodulated digital output of the received signal. This is connected to the microcontroller.

Pin 16 – RFIN: This pin takes the RF signal input. The input should be in the form of On-Off Keying (OOK). This pin will be connected to the SMA connector of the receiver antenna through a 50-Ohm micro-strip.

6.3.3 JTAG/BSL Connector

The JTAG/BSL connection used is through 0.1-inch pitch header pins provided by Molex/Waldom Electronics Corp. Figure 32 below shows a schematic of the design for the JTAG/BSL Header used in the Receiver Unit. An explanation of the pin connections and configurations follow this figure.

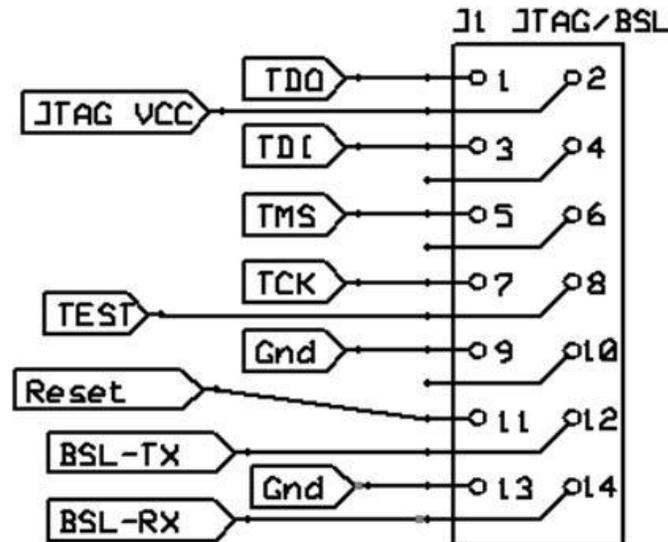


Figure 32: Schematic of JTAG/BSL Header

The pins of the JTAG/BSL Header and their descriptions are provided below:

Pin 1: This pin is the Test Data Out of the JTAG connection. It represents the data shifted out of the device’s test or programming logic and is valid on the falling edge of TCK when the internal state machine is in the correct state. This is connected to Pin 28 of the microcontroller.

Pin 2: This pin is the VCC pin of the JTAG connection. It is used to provide power to the processor. The voltage of 3.0 VDC must be set in the IDE for proper operation.

Pin 3: This pin is the Test Data In of the JTAG connection. It represents the data shifted into the device’s test or programming logic. It is sampled at the rising edge of TCK when the internal state machine is in the correct state. This is connected to Pin 27 of the microcontroller.

Pin 5: This pin is the Test Mode State of the JTAG connection. It is sampled at the rising edge of TCK to determine the next state. This is connected to Pin 26 of the Microcontroller.

Pin 7: This pin is the Test Clock of the JTAG connection. It synchronizes internal state machine operations. This is connected to Pin 25 of the Microcontroller.

Pin 9, 13: This pin is available to provide a common ground between the host computer/laptop and the board.

Pin 11: This pin is used to control the reset (Pin 7) of the microcontroller.

Pin 12: This is the transmit data output pin from the bootstrap loader of the microcontroller. It is connected to Pin 22 of the microcontroller.

Pin 14: This is the receive data input pin to the bootstrap loader of the microcontroller. It is connected to Pin 10 of the microcontroller.

All other pins are unconnected and unused.

6.3.4 Light Indicator Header

The Light Indicator Header uses a 0.1-inch pitch 6-pin header by Molex. The header is connected to the Light Indicator component by a ribbon cable. Figure 33 below shows a schematic of the design for the Light Indicator header that is used in the Receiver Unit. An explanation of the pin connections and configurations follow this figure.

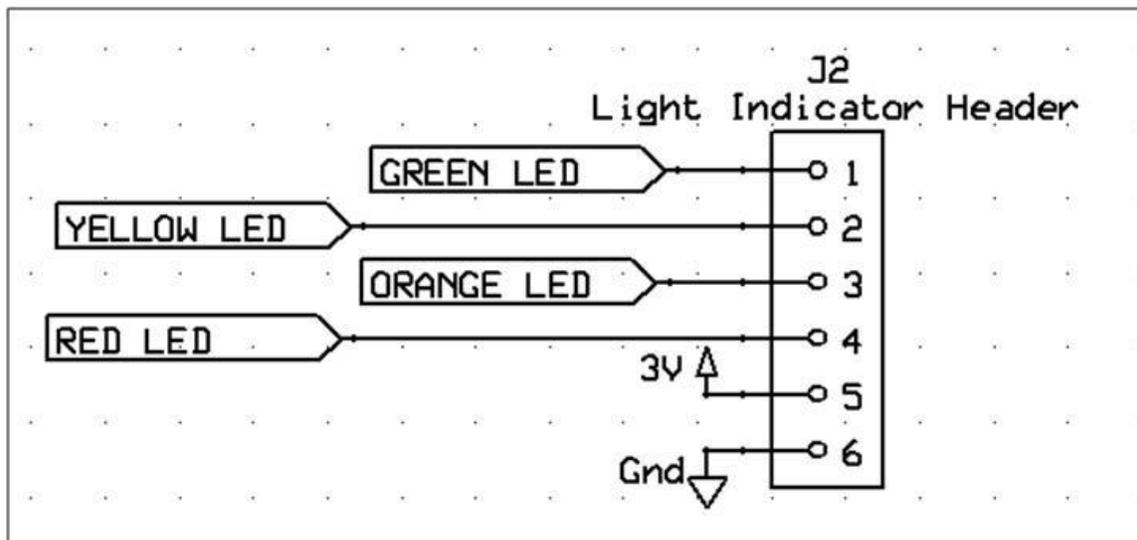


Figure 33: Schematic of Light Indicator Component

The pins of the Light Indicator header and their descriptions are provided below:

Pin 1: This pin connects to Pin 21 of the microcontroller. The line driven by this pin is used to light the green LED of the Light Indicator Component.

Pin 2: This pin connects to Pin 20 of the microcontroller. The line driven by this pin is used to light the yellow LED of the Light Indicator Component.

Pin 3: This pin connects to Pin 19 of the microcontroller. The line driven by this pin is used to light the orange LED of the Light Indicator Component.

Pin 4: This pin connects to Pin 18 of the microcontroller. The line driven by this pin is used to light the red LED of the Light Indicator Component.

Pin 5: This pin connects to the 3 VDC power rail of the Receiver Unit PCB. It provides power to the Light Indicator Component.

Pin 6: This pin connects to the ground plane of the Receiver Unit PCB. It provides a common ground to the Light Indicator Component.

6.3.5 Power Switch

The Power Switch used in this design is the Tyco Electronics Alco switch MHS12304. Figure 34 shows a schematic of the design for the Power Switch used in the Receiver Unit. An explanation of the pin connections and configurations follows this figure.

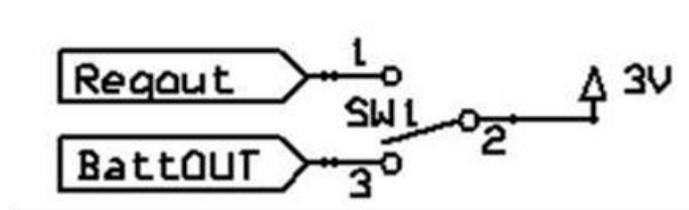


Figure 34: Schematic of SPDT Switch

The pins of the power switch and their descriptions are provided below:

Pin 1: This pin connects to Pin 2 when the SPDT power switch is in its left position. In this case, the voltage potential pin of the power jack is connected to the 3.0 VDC power rail of the PCB.

Pin 2: This is the common contact of the power switch. When the power switch is in its left position, it connects to Pin 1. When the power switch is in its right position, it connects to Pin 3. If connected to Pin 1, the voltage potential pin of the power jack is connected to the 3.0 VDC power rail. If connected to Pin 3, it connects the batteries to the 3.0 VDC power rail.

Pin 3: This pin connects to Pin 3 when the SPDT power switch is its right position. In this case, the voltage pin of the batteries is connected to the 3.0 VDC power rail of the PCB.

6.3.6 Power Jack

The power jack used in the Receiver Unit is a barrel type male pin power jack by Tamura. Figure 35 shows a schematic of the design for the power jack. An explanation of the pin connections and configurations follow this figure.

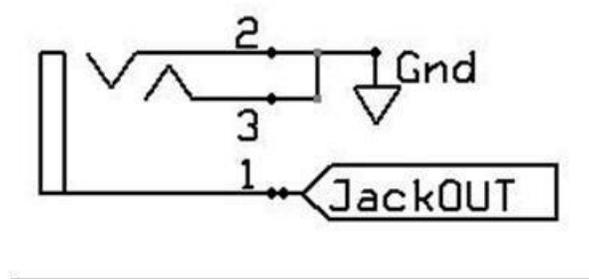


Figure 35: Schematic of Power Jack

The pins of the power jack and their descriptions are provided below:

Pin 1: This pin is the positive terminal of the power jack. This pin has a 3.0 VDC potential. It is connected to Pin 1 of the power switch.

Pin 2, 3: These pins are the negative terminals of the power jack. Pin 3 is only connected when the adapter is plugged in and Pin 2 represents the outer shield of the power jack. Both pins are connected to ground for correct operation.

6.3.7 Voltage Regulator

The voltage regulator used is from MicroChip Technology. It regulates voltages up to 16.0 VDC down to 3.0 VDC. Figure 36 below shows a schematic of the voltage regulator. An explanation of pin connections and configurations follow this figure.

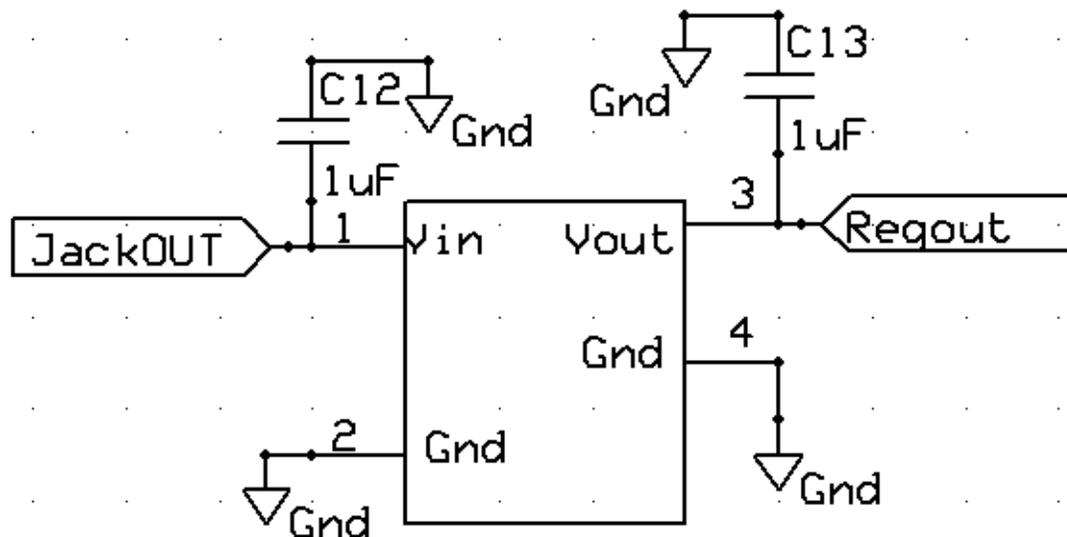


Figure 36: Voltage Regulator

The pins of the voltage regulator and their descriptions are provided below:

Pin 1 –Vin: This is the voltage input pin of the chip. The voltage range here is anywhere between 0 and 16.0 VDC. For this design, it should be between 3.0 VDC and

16.0 VDC. It has a noise-reducing capacitor connected between the voltage input and the ground. This input will come from the power jack.

Pin 2, 4 – Gnd: This pin is the ground reference of the chip. This is connected to the ground plane of the board.

Pin 3 – Vout: This is the output pin of the regulator. It produces 3.0 VDC output. It has a capacitor connected between the output and ground to reduce noise. This output is used by the circuit and is directly connected to the power switch.

6.3.8 SMA Connector

The SMA connector used is of the reverse polarity type to connect to the reverse polarity antenna chosen. The connector is provided by Linx Technologies Inc. Figure 37 shows a schematic of the connector. An explanation of the pin connections and configurations follows this figure.

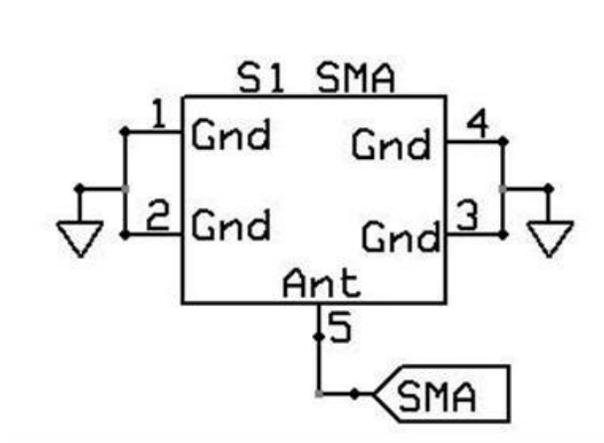


Figure 37: Schematic of SMA Connector

The pins of the SMA Connector and their descriptions are provided below:

Pin 1, 2, 3, 4 – GND: These pins are used to ground the RP-SMA Connector. They are all connected to the ground plane of the PCB.

Pin 5 – ANT: This pin connects to the antenna on the Receiver Unit.

6.3.9 Batteries

The batteries used in this design are Panasonic –BSG’s CR-2 batteries. Figure 38 shows a schematic of the design for the batteries used in the Receiver Unit. An explanation of the pin connections and configurations follow this figure.

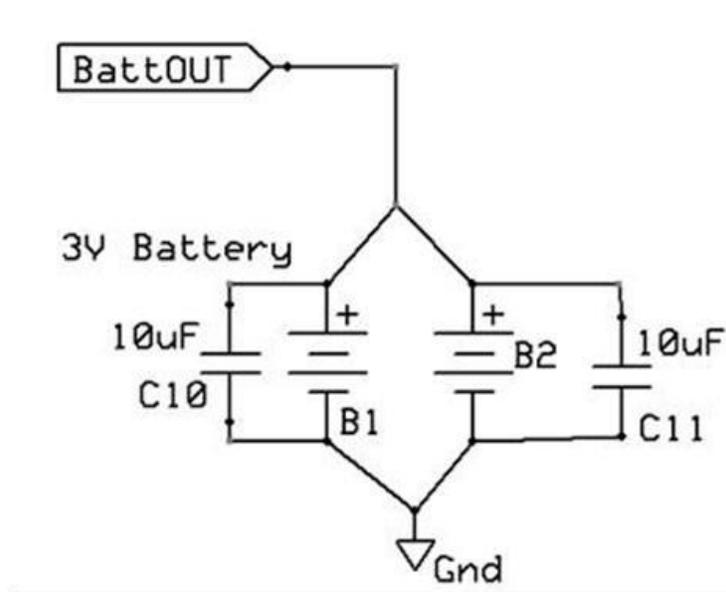


Figure 38: Schematic of the CR-2/BE Batteries

Displayed in Figure 38 there are two instances of the CR-2 batteries. They are placed in parallel structures in order to increase the capacity of the batteries. Each battery has a positive and negative terminal. The positive terminals have a 3.0 VDC potential and are connected to a line that connects to Pin 2 of the tilt switches. The negative terminals serve as the negative reference and are connected to the ground. Capacitors are also placed parallel to the batteries in order to increase the stability of the batteries' power output.

6.4 Light Indicator Design

This section describes the design of the Light Indicator Unit. It takes the functional blocks described previously and displays the design of each. The design of each functional block shows the components needed for that specific functional block and how that functional block should be interfaced.

6.4.1 Receiver Headers

The receiver headers use a 0.1-inch pitch 6-pin header by Molex. The headers are connected to the Receiver Unit by a cable. Figure 39, below, shows a schematic of the design for one of the receiver headers that is used in the Light Indicator Component. There are four identical instances of this header on the component. An explanation of the pin connections and configurations follows this figure.

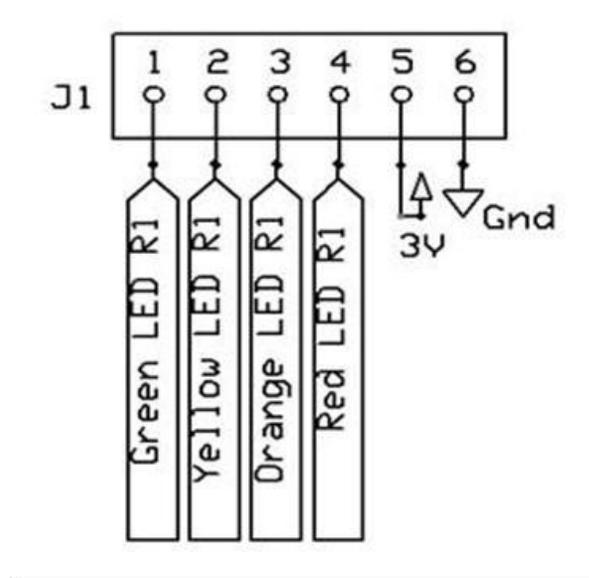


Figure 39: Schematic of Receiver Header

The pins of the receiver headers and their descriptions are provided below:

Pin 1: This pin receives input from the Receiver Unit for the operation of the green LED. This pin is connected to Pin (9, 10, 11, 12)⁴ of OR gate instance L1. If this pin is high, the green LED connected to OR gate (L1) should be lit.

Pin 2: This pin takes input from the Receiver Unit for the operation of the yellow LED. This pin is connected to Pin (5, 4, 3, 2)⁴ of OR gate instance L1. If this pin is high, the yellow LED connected to OR gate (L1) should be lit.

Pin 3: This pin takes input from the Receiver Unit for the operation of the orange LED. This pin is connected to Pin (9, 10, 11, 12)⁴ of OR gate instance L2. If this pin is high, the orange LED connected to OR gate (L2) should be lit.

Pin 4: This pin takes input from the Receiver Unit for the operation of the red LED. This pin is connected to Pin (5, 4, 3, 2)⁴ of OR gate instance L2. If this pin is high, the red LED connected to OR gate (L2) should be lit.

Pin 5: This pin connects to the 3.0 VDC potential of the Receiver Unit. It is used to supply the OR gates.

Pin 6: This pin provides a common ground between the Receiver Unit and the Light Indicator component.

⁴ Corresponds to the pin connection for the different Receiver Header instances. First number corresponds to instance J1. Second number corresponds to instance J2. Third number corresponds to instance J3. Fourth number corresponds to instance J4.

6.4.2 OR Gates

The 4-input OR gates (Figure 40) are provided by Texas Instruments. Part CD4072BNSR contains two, 4 input OR gates on the chip. Two instances of this chip will be used in the Light Indicator Component. An explanation of the pin connections and configurations follow this figure.

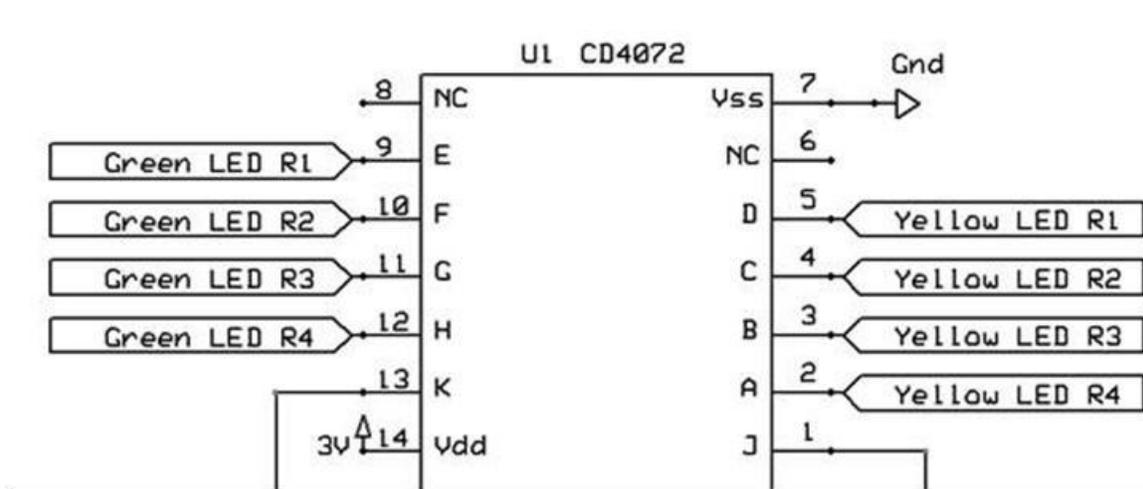


Figure 40: Schematic of OR Gates

The pins of the OR gates and their descriptions are provided below:

Note: * Corresponds to the two instances of the OR gate chips. The order being instance U1/ instance U2.

Pin 1 – J: This is the output for the logical OR of Pins 2, 3, 4, and 5 ($J = A + B + C + D$, where '+' denotes logical OR). This pin connects to the yellow/red* LED through a current limiting resistor. If the result is true, this pin drives the (yellow/red)* LED high causing the LED to light. If the result is false, this pin drives the (yellow/red)* LED low causing the LED not to light.

Pin 2 – A: This is one of 4-input pins that is logically ORed together to produce an output on Pin 1. The input to this pin comes from Pin (2/4)* of Receiver Header 4 (R4).

Pin 3 – B: This is one of 4-input pins that is logically ORed together to produce an output on Pin 1. The input to this pin comes from Pin (2/4)* of Receiver Header 3 (R3).

Pin 4 – C: This is one of 4-input pins that is logically ORed together to produce an output on Pin 1. The input to this pin comes from Pin (2/4)* of Receiver Header 2 (R2).

Pin 5 – D: This is one of 4-input pins that is logically ORed together to produce an output on Pin 1. The input to this pin comes from Pin (2/4)* of Receiver Header 1 (R1).

Pin 6 – NC: This pin has no connection and is not used.

Pin 7 – V_{ss}: This is the negative voltage reference for the chip. This pin is connected to the ground.

Pin 8 – NC: This pin has no connection and is not used.

Pin 9 – E: This is one of 4-input pins that is logically ORed together to produce an output on Pin 13. The input to this pin comes from Pin (1/3)* of Receiver Header 1 (R1).

Pin 10 – F: This is one of 4-input pins that is logically ORed together to produce an output on Pin 13. The input to this pin comes from Pin (1/3)* of Receiver Header 2 (R2).

Pin 11 – G: This is one of 4-input pins that is logically ORed together to produce an output on Pin 13. The input to this pin comes from Pin (1/3)* of Receiver Header 3 (R3).

Pin 12 – H: This is one of 4-input pins that is logically ORed together to produce an output on Pin 13. The input to this pin comes from Pin (1/3)* of Receiver Header 4 (R4).

Pin 13 – K: This is the output for the logical OR of Pins 9, 10, 11, and 12 (K = E + F + G + H, where '+' denotes logical OR). This pin connects to the (green/orange)* LED through a current limiting resistor. If the result is true, this pin drives the (green/orange)* LED high. If the result is false, this pin drives the (green/orange)* LED low.

Pin 14 – V_{dd}: This is the positive reference for the voltage of the chip. This pin is connected to the 3 VDC supplied by the receiver headers.

6.4.3 LEDs

Four LEDs are used on the Light Indicator Component. These LEDs are driven based on the output of the OR gates. The three of the four LEDs are made by Lite-On-Inc. The orange LED is made by Lumex Opto/Component Inc. Figure 41 shows the setup of one of the LEDs. This setup is identical for each LED.

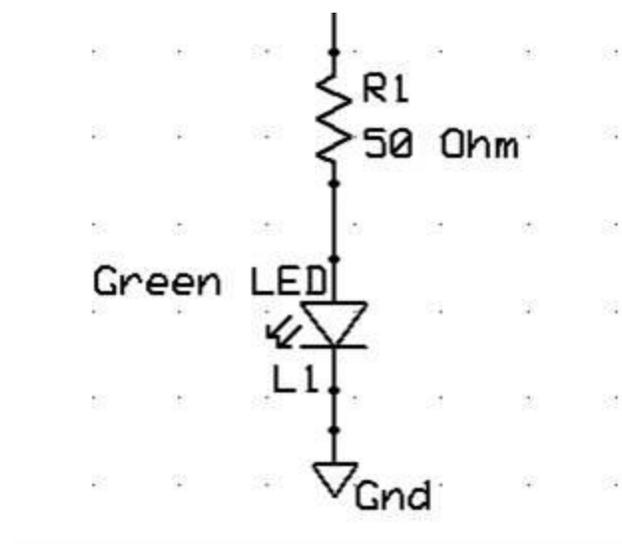


Figure 41: Schematic of LED Circuit

LED setup: The LED Circuit is simple. A current limiting resistor is placed between the LED and the voltage source. The value of this resistor is determined through Ohm's Law. In this case, it would be in the following form:

$$R = \frac{V}{I} \tag{12}$$

The voltage supply comes from the output of the 4-input OR Gate. The LED voltage drop (forward voltage) and LED current rating are specified by each LED's datasheet. For these LEDs, the forward voltage of 2.0 VDC and a current of 20 mA will be used. Subtracting the forward voltage from the supply voltage will give the value for V , while the 20 mA will be used for I . Plugging these values in and solving gives us the minimum value of the resistor as 50 Ohms. The resistor is connected to the anode of the LED. The cathode of the LED is connected to the ground.

7.0 Sensor Unit Software Design

The Sensor Unit software has two main tasks. It must receive input that comprises the Sensor Unit Data Block and store this in the information flash memory. Then the software must take this data and send it to the RF Transmitter through Output Pin P1.0. The next two subsections explain each of these tasks as well as any other steps in more detail with flow charts to illustrate the concept.

7.1 Initialization

To begin, the microcontroller must be setup and initialized. This includes initializing variables, setting the necessary microcontroller registers, and disabling the watchdog timer. Initializing the variables within this program consists of setting all counters to zero, setting pointers to the proper address, setting up a dummy Sensor Unit Data Block, and setting the priority of the Sensor Unit. Next, the program disables the Watchdog Timer of the MSP430. The purpose of the Watchdog Timer is to perform a controlled system restart after a software problem has occurred. This is based on a specified time interval set in the Watchdog Timer registers. On initial power up, this interval is 32768 cycles [1]. If the system is restarted during flash operations, unexpected results could occur. For this reason, the Watchdog Time is disabled.

The first register setup is the register for Timer A. To begin, MSP430 devices come with calibrated Basic Clock Module settings for specific frequencies stored in information memory segment A [1]. The next part of the program will check to ensure this memory is intact within information memory. Unless the processor has been corrupted or the information memory has been purposely erased, this should not be the case. These stored settings are then used in the next step to setup the registers of the Basic Clock Module. The other register that is set up is the Flash Timing Generator. The Flash Timing Generator controls erase and write operations. It has a particular operating frequency that must be used. For our processor this is 333 kHz. To achieve this operating frequency, clock dividers' settings in a flash register must be specifically set.

Another timer must also be setup to control the timing of the output to the RF transmitter chip. Given the 10 kbps data rate constraint given by the RF transmitter chip, bits can only be reliably changed every 0.0001 seconds (1/10 kbps). In this system the processor is operating at 8 MHz. This means a clock cycle occurs every 0.000000125 seconds. Thus if an action is to occur every 0.0001 seconds, 800 clocks' cycles should elapse.

Next, the configuration of registers that set up the pins of the microcontroller needed by the system is performed. First, the I/O line of the microcontroller is setup. For the Sensor Unit this is Pin 21 or P1.0 (software designation of pin 21). This pin is set to the I/O function as an output pin. Next, the UART pins need to be setup. On this microcontroller software, Pins P3.4 and P3.5 need to be set to a UART function. A baud

rate also needs to be set for the UART. In this system, 19200 kbps has been selected. Recommended settings for each baud rate given a crystal frequency can be found in [9]. These settings are then applied.

7.2 Sensor Unit Data Block Storage

The data to be stored in the flash information memory of the MSP430F2132 is based on the Sensor Unit Data Block presented in section 4.1.2. Although a dummy set of data is initialized within the program, the Sensor Unit Data Block specific to the site must be input through use of the UART communication.

An interrupt is set up for the UART communication. Each time a character is sent from the host PC, it is placed in a buffer and the interrupt is entered. Here the input can be processed. For the Sensor Unit, this interrupt is only responsible for either storing to the Sensor Unit Data Block to a temporary array or sending the stored block back to a host pc to be displayed.

To display the Sensor Unit Data Block stored in memory, the UART will look for a “#” character. When this character is received by the embedded software, it enters a function that prints (outputs) the Sensor Unit Data Block. This print function sets a pointer to the flash memory location where the block is located. This location is the beginning of information memory segment B (memory address 0x1080). The function will take each of the 23 bytes one at a time and parse them into nibbles (4 bits). These nibbles are converted to their ASCII equivalent for the hexadecimal (HEX) representation (base-16) of each nibble. Printed to the host machine on their Hyper Terminal display will be a sequence of 46 characters (0-9,A-F) representing the 23 bytes stored in flash memory.

Otherwise, the interrupt has two modes dictated by the state of a variable titled (UART_mode). This variable is initialized to 0. In this state, the interrupt is only looking for two characters. These characters are the aforementioned “#” character or a “*” character. Once the interrupt has seen two consecutive “*” characters in a row, the embedded software enters the second mode where UART_mode is equal to 1. Once in this mode, the next 42 characters are read and stored as the Sensor Unit Data Block. Every two characters received, which must be 0-9 or A-F (digits in the base-16 or HEX number system), will be concatenated to form a byte. After the 21st byte is formed, a function takes the bytes and forms the last two bytes of the Sensor Unit Data Block.

The last part of the program places the Sensor Unit Data Block in information memory. To do this, a pointer is first set to the address of information memory segment B. As mentioned earlier, segment A is used for calibration information. Next, the flash segment is unlocked and set to be erased. This is done using a flash settings register. Following this is a write, termed a *dummy write* that erases the segment [9]. Then the segment can be written with Sensor Unit Data Block. The flash register is first set to write. Next, a loop is entered that writes one byte of the 23 total data bytes. Lastly, the flash write bit is cleared and lock bit is reset within the flash register. Once the data have been placed in memory, a “data_txed” string is displayed on the host PC’s Hyper Terminal window.

Figure 42 shows the flowchart for the process of storing the Sensor Unit Data Block to information memory.

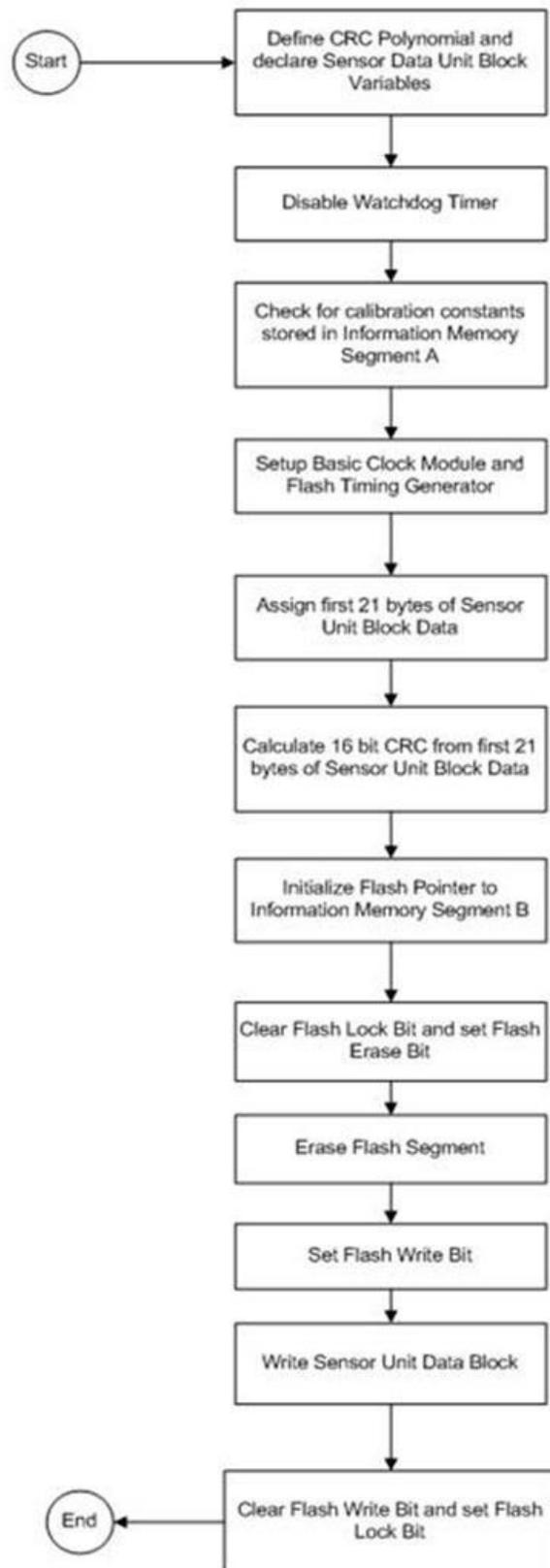


Figure 42: Flowchart for Data Storage

7.3 Data Transmission

This section contains details on how the data is digitally encoded for transmission over the RF link. The software provides a digital representation of the data to the RF transmitter. The RF transmitter converts the digital data to an RF signal for transmission to the Receiver Unit.

7.3.1 Manchester Encoding

The Sensor Unit Data Block is transmitted using Manchester Encoding. Although there are various interpretations on the encoding, the principle is based on the transitions made mid-bit period. Manchester encoding provides a communication mechanism that is more resistant to transmission errors. The bit period is 0.4 milliseconds. This system defines a logical '1' being transmitted as a mid-bit high to low transition as illustrated in the left half of Figure 43. This system defines a logical '0' being transmitted as a mid-bit low to high transition as illustrated in the right half of Figure 43.

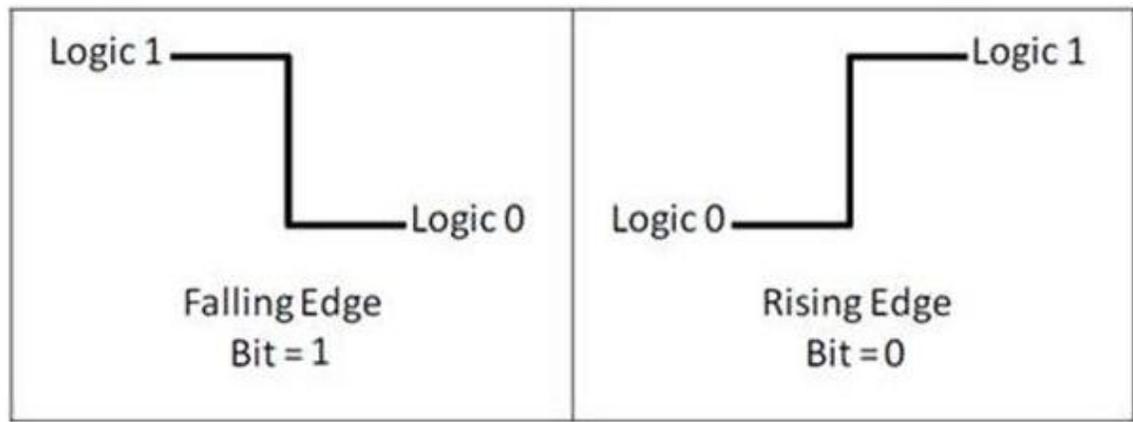


Figure 43: Manchester Encoding

7.3.2 Synchronization

A reference point must be set by the transmitter in order for the Receiver Unit to properly decode the data. A preamble serves this purpose. Normally, in a synchronized system, just a preamble can be sent to signify the beginning of the data packet. However, there are a few constraints, which make this insufficient for this system. First, the systems are not synchronized. This means a set sampling of the transmitted bit stream cannot be used to determine the mid bit transition. Instead, the capability of the Receiver Unit to detect edges and the time between these edges will be used. Secondly, using edge detection brings forth the need to establish whether each edge is rising or falling. If the direction of the first transition is known, the direction of each succeeding edge can be tracked.

The solution to these problems is a synchronization period. Immediately before data is sent, the line is pulled high for a known period. This period must be larger than the largest time difference in edges of the communication protocol. For this system, a

period of 18000 clocks or 2.25 milliseconds is used. This also establishes the direction of edges. Following the synchronization period, the Receiver Unit knows the last edge was a falling edge. Using this reference, the following data can be decoded by the Receiver Unit.

7.3.3 Transmission

The Sensor Unit Data Block is stored at the beginning of information memory segment B (address 0x1080). When data is to be transmitted a pointer is set to this address and each byte is retrieved one at a time for a total of 23 bytes. For each byte, the bits are determined one at a time. This is done using a mask byte containing seven 0's and one 1 digit. The position the 1 digit is within the byte is the bit being tested in the data byte. For example, a byte with the value of 128 or (10000000 binary) tests the most significant digit, a byte with the value of 64 (01000000) tests the second most significant bit, etc. Performing a logical AND operation between the mask and the data byte yields the value of the tested digit. The logical AND operation is a binary function whose output is logical '1' (true) if and only if all inputs are logical '1' (true); otherwise the output is logical '0' (false). If the result of the operation is 1 (true), then the value of the digit is a 1. If the result of the operation is 0 (false), then the value of the digit is a 0.

Given the value of a current bit and the last bit, a certain sequence of transitions will be output. This sequence is based on Manchester Encoding as mentioned above. Figure 44 illustrates what each of these sequences would look like. All four possible combinations for two consecutive bits are shown.

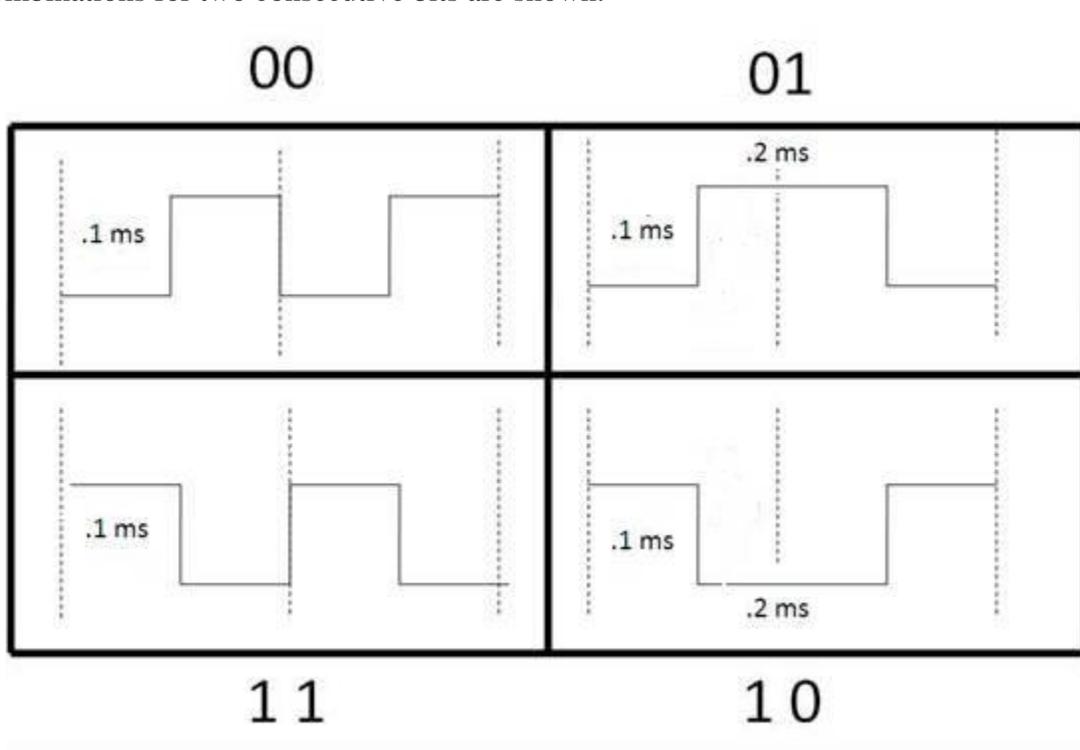


Figure 44: Bit Transition Sequences for Consecutive Bits

To illustrate the transmission process, a flowchart is shown below in Figure 45. This flowchart shows what must be done in relation to the transmission process. It first shows the setup of the microcontroller. Next is shows the double-nested main loop for

transmitting the Sensor Unit Data Block. The inner loop corresponding to a single byte of the block while the outer loop corresponds to the entire block.

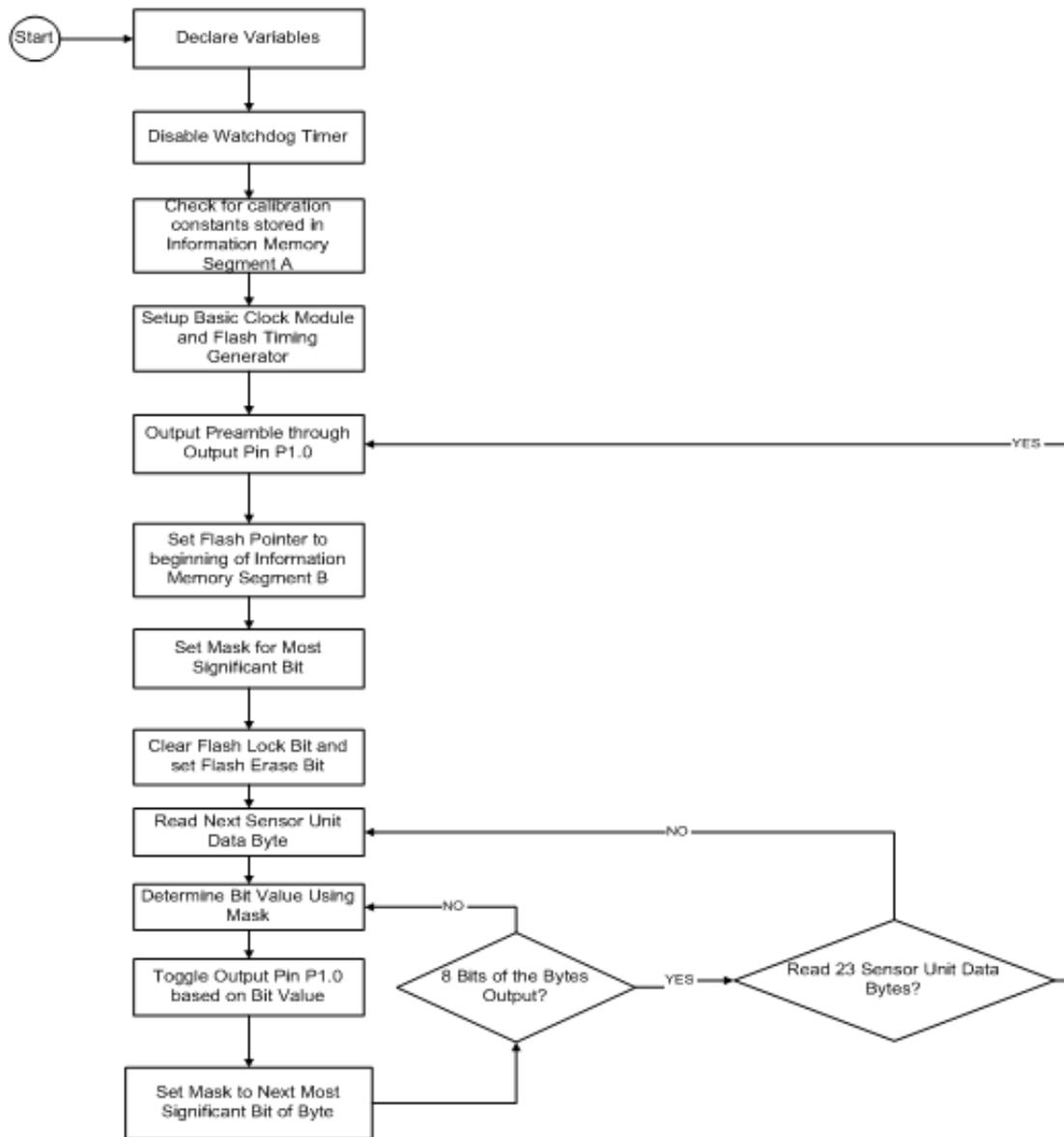


Figure 45: Flowchart for Data Transmission Program

7.3.4 Delay Addition for Collision Avoidance

The use of multiple, independent Sensor Units within the system introduces a chance of collision. A collision is when two packets are sent from separate transmitters on the same channel to the same receiver causing the packets to overlap or *collide*. If this occurs, the overlapping data will be unintelligible and neither packet reaches the receiver

without error. In this system, a collision occurs when two or more of the Sensor Units are transmitting to the same Receiver Unit simultaneously. This would mean two or more Sensor Units were released and rose to the surface while both were within range of the Receiver Unit at the same time. Given any significant difference in depth of the different Sensor Units, this should not be a likely situation. However, a solution was developed to account for this situation.

To give each Sensor Unit a strong probability of having its message accepted at the Receiver Unit (the Receiver Unit will ignore a message containing an error) the transmission algorithm was altered. For a given transmission to succeed, a time slot of at least twice the message length is needed. This message length is denoted by T as shown in Figure 46 below. If Sensor Unit 2's transmission starts T seconds before Sensor Unit 1's transmission, the end of Sensor Unit 2's transmission collides with the start of Sensor Unit 1's transmission. In addition, if Sensor Unit 2's transmission starts anytime during the T seconds that Sensor 1 is transmitting, the two transmissions collide.

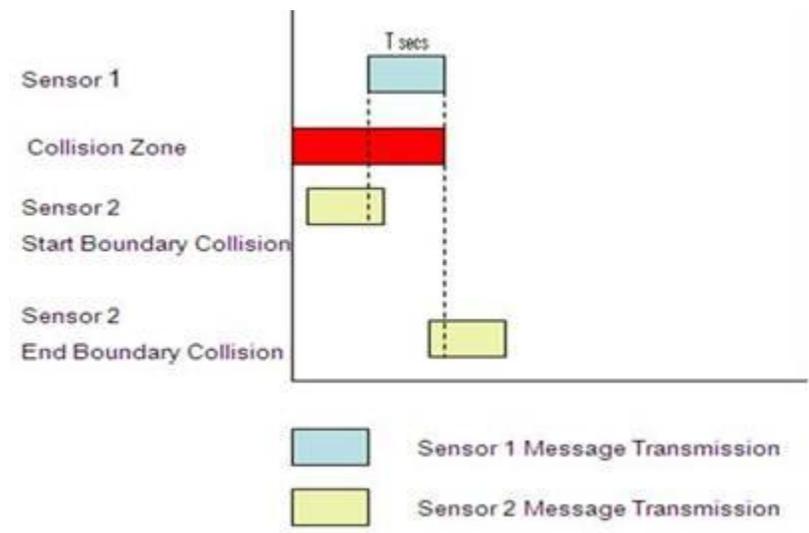


Figure 46: Collision Scenarios of Two Transmissions

If the Sensor Units could be scheduled to transmit at certain *known* times, a delay of T could be added in between each Sensor Unit's transmissions and collisions could be avoided. A centralized synchronizing agent is required to schedule the transmissions. However, for this system, the time at which a Sensor Unit will begin transmitting cannot be scheduled because the unit will be released by natural events. Thus, the centralized synchronizing agent is not an option for this application. Even with a larger delay added between each Sensor Unit's transmissions, the collisions would reoccur if both Sensor Units have the same delay. Therefore, several delay lengths are added to the transmission algorithm. Each delay is a multiple of the $2T$ transmission length of the message ($2T$, $4T$, $8T$, $16T$). The delay length chosen is based on a random number generated by the Sensor Unit. Figure 47 shows an example of a possible scenario using this algorithm. The figure shows an initial collision and then the two Sensor Units select one of the four delay options based on the random number they have generated. If the random numbers are different, then each Sensor Unit selects a different delay and both transmissions get through to the Receiver Unit. In the example below the two Sensor Units would have a 25% chance of a collision, (selecting the same slot) occurring on their next transmission.

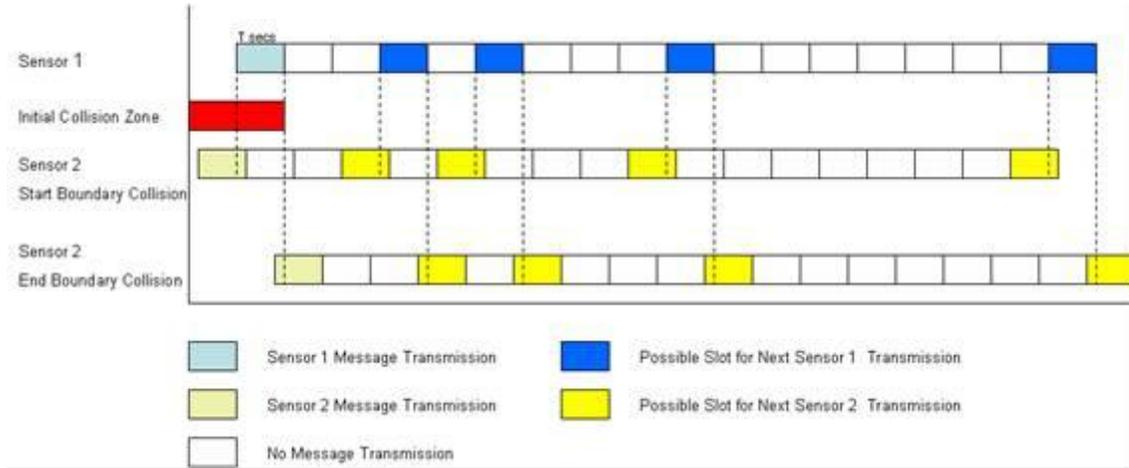


Figure 47: Collision Scenario with Delays Added

Additionally, the algorithm was altered so that the Sensor Unit programmed with a higher priority Color Code transmits more frequently. For example, a Sensor Unit with a red Color Code transmits the most frequently. The following table lists percentages for the frequency each color would have of selecting a certain delay time.

Table 7: Delay Percentages for Each Color Code, $T \approx 100$ ms

	Red Sensor Unit	Orange Sensor Unit	Yellow Sensor Unit	Green Sensor Unit
2T Delay	50.00%	37.50%	33.33%	25.00%
4T Delay	25.00%	37.50%	25.00%	25.00%
8T Delay	25.00%	12.50%	25.00%	25.00%
16T Delay	0.00%	12.50%	16.66%	25.00%

Figure 48, below, shows a Sensor Unit programmed with the red Color Code, and a Sensor Unit programmed with a yellow Color Code. This oscilloscope screen shot shows how the red Color Code's random delays are on average much shorter than that of the yellow Color Code's random delays.

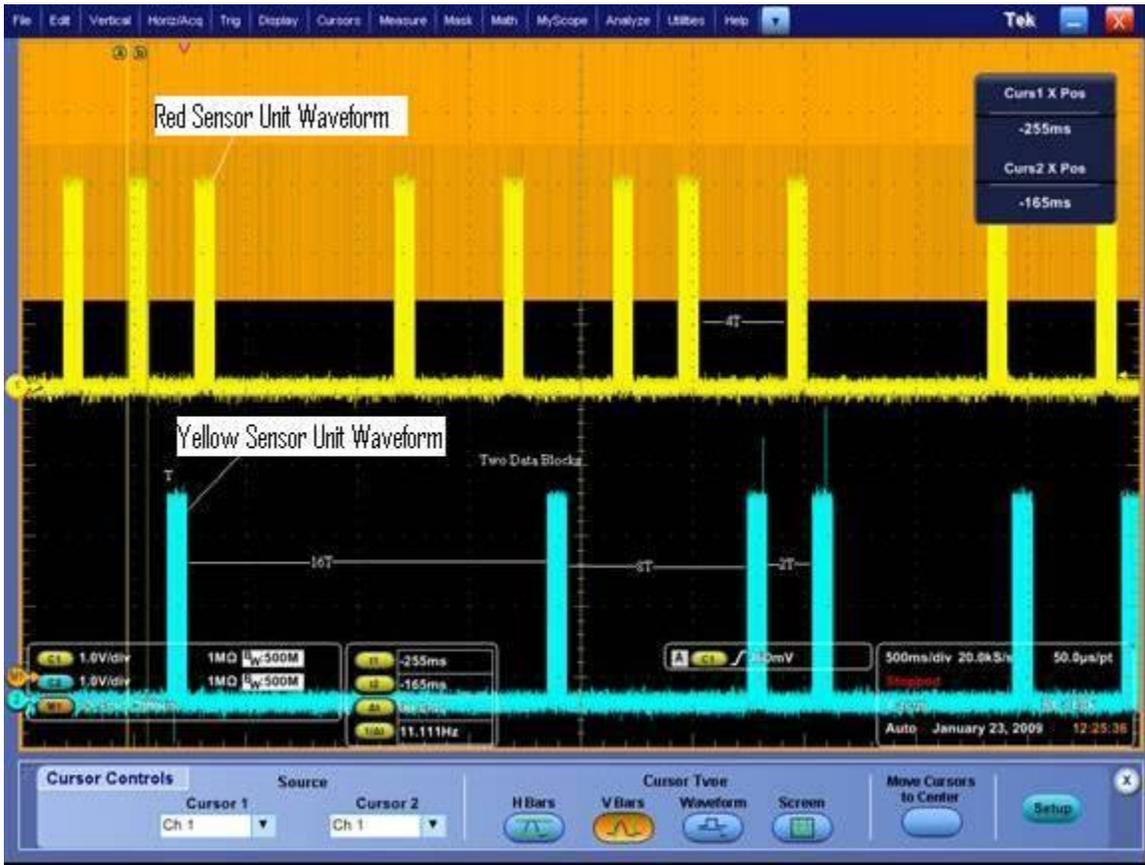


Figure 48: Oscilloscope Shot of Red Priority Delays and Yellow Priority Delays

8.0 Receiver Software Design

This section presents the design of the software that will be run on the Receiver Unit's MSP430F2132 processor. This software has the task of receiving the data from the Sensor Units and driving the LEDs.

8.1 Initialization

Figure 49 shows the flowchart for the program that receives the Sensor Unit transmission as well as controls the LED I/O lines. The program begins by defining the CRC polynomial used to check if the received Sensor Unit Data Block is correct. It also defines the preamble, which is 0xAA (HEX characters for bit pattern: 10101010). The preamble precedes the Sensor Unit Data Block. The program then defines several global variables that will be shared by the function and the interrupt service routines (ISRs).

Next, the program disables the Watchdog Timer of the MSP430. The purpose of the Watchdog Timer is to perform a controlled system restart after a software problem has occurred. This is based on a specified time interval set in the Watchdog Timer registers. On initial power up, this interval is 32768 cycles [1]. If the system is restarted during flash operations, unexpected results could occur. For this reason, the Watchdog Timer is disabled.

MSP430 devices come with calibrated Basic Clock Module settings for specific frequencies stored in information memory segment A [1]. The next part of the program will check to ensure that this memory is intact within the information memory. Unless the processor has been corrupted or the information memory has been purposely erased, this should not be the case. These stored settings are then used in the next step to setup the registers of the Basic Clock Module. The other register that is set up is the Flash Timing Generator. The Flash Timing Generator controls erase and write operations. It has a particular operating frequency that must be used. For this processor, the frequency is 333 kHz. To achieve this operating frequency, clock dividers' settings in a Flash register must be specifically set using the FCTL2 register.

At this stage, I/O Port P1, P2, and P3 are set up for correct operation. Specifically, P1.2 must be an input and P2.3, P2.4, and P3.7 must be outputs. Timer A is then set up to run off the sub-main clock in "up" mode. This means that the timer produces an interrupt every specified number of clock cycles. The number of clock cycles is stored in register TA0CCR0. The incoming data arrives at 10000 Hz. Given the 8 MHz clock, this means an interrupt should be created every 800 clock cycles. The flash pointer is then initialized to the starting address for the storage of the data. The other variables are initialized to 0 except the mask, which is initialized to 4 in order to operate with the third least significant bit (lsb) of P1IN.

8.2 Main Program

The program now enters a continuous loop. The loop does nothing until Mode 2 is reached. The loop is entered in Mode 0 and Mode 2 is only achieved through the interrupt service routine (ISR). The ISR is entered every 800 cycles. Within the ISR, the logic level of the signal on Pin P1.2 is determined using the mask. The value is then shifted to the variable holding the last 8 data bits received. Following this, there is a check to see if the program is in Mode 1. Mode 1 means the program has received a correct preamble and will store the next 23 bytes of data to an array. After the 23 bytes of data have been received, the program will be placed in Mode 2. If the preamble has not been received and the program is not in Mode 1, then it checks the last 8 bits received to see if they match the preamble. If so, the program is placed in Mode 1 and on the next ISR call will begin storing the next 23 bytes of data in an array.

Once in Mode 2, a conditional is entered within the main function's while loop. Within this conditional *CRC_check()* is called first. *CRC_check()* performs the same methodology as the function *CRC_calculation()* from the *Sensor_Data_Write.c*. The only difference between this and the version used for the Sensor Unit is that instead of using only the first 21 bytes of the Sensor Unit Data Block to calculate a remainder, the entire 23 bytes of the Sensor Unit Data Block will be used. Using the same polynomial, if the message is error-free, the remainder produced by dividing the Sensor Unit Data Block by the polynomial is zero. If the function determines that the remainder is zero, no error has occurred and the *store_and_process()* function is called within the main function. If the remainder is not zero, an error has occurred during transmission and the message is discarded. In either case, the mode is restored to Mode 0 and the conditional is then left.

The function *store_and_process()* has the task of storing the 23 byte Sensor Unit Data Block to flash memory and driving the LED lines based off the value of byte 20 of the Sensor Unit Data Block. This function takes the array of the current 23 bytes of data and begins storing it at the next available flash location. When the 20th byte is encountered, it will check its value. If the value is 0, 1, 2, or 3 it drives the corresponding green, yellow, orange, or red I/O line high respectively. The other three lines that are not driven high will be driven low. Additionally, once a state of higher priority has been reached, a lower state cannot be activated.

Once the conditional is left, the program repeats this process infinitely.

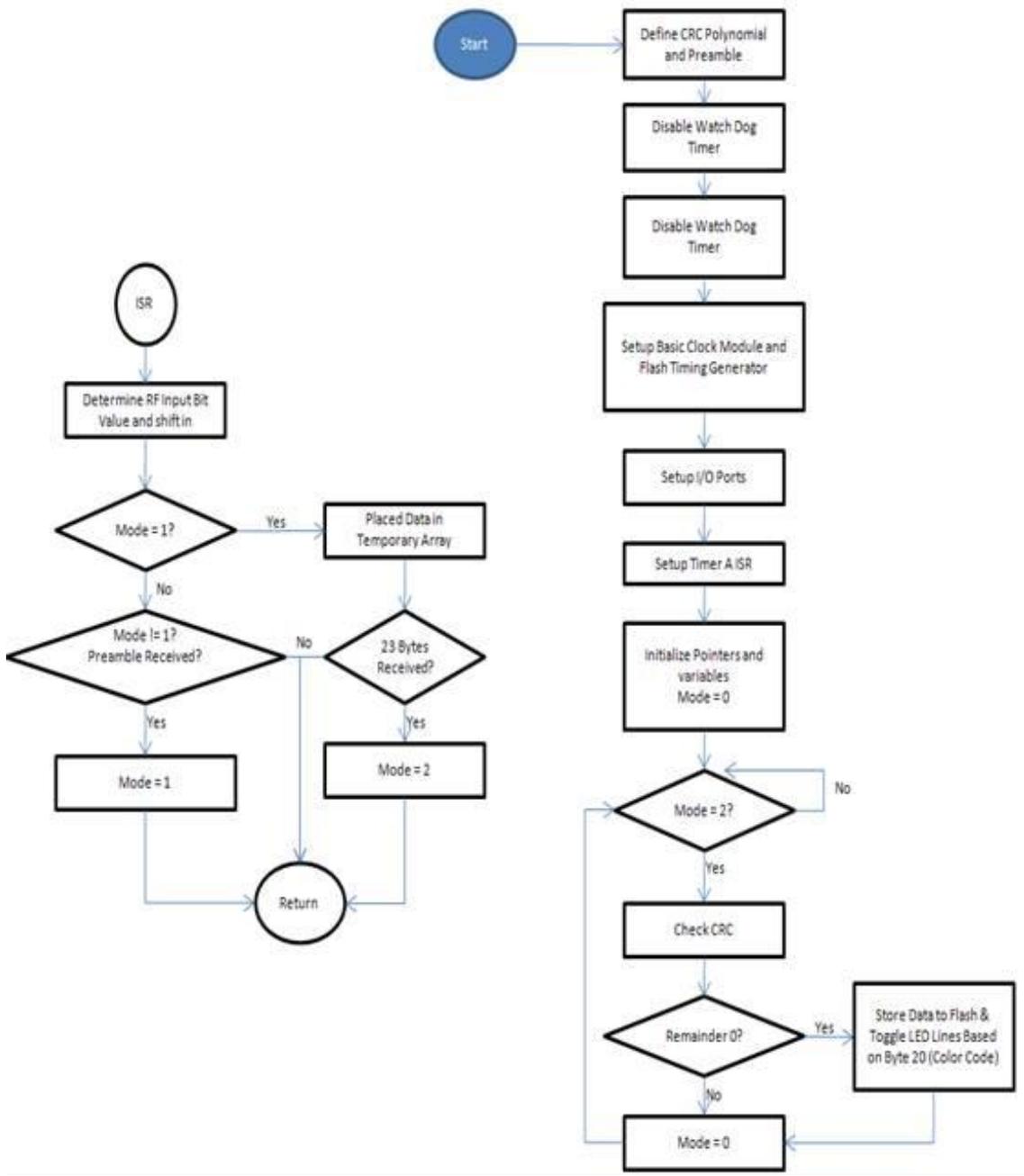


Figure 49: Flowchart for Receiver Program

9.0 System Testing

This section includes a summary of the testing done on the system.

9.1 Preliminary Component Testing

This section contains a review of preliminary tests done on the early RF Receiver, Transmitter, and Light Indicator. This work had the goal of ensuring that these components are suitable to complete the RF tasks necessary for this project. First, tests were performed with the transmitter with respect to its signal strength under various conditions. Second, the ability of the RF Transmitter and Receiver to send and receive data respectively was analyzed. Lastly, the Light Indicator was tested to ensure the proper input would trigger the correct LED.

9.1.1 Transmitter Signal Strength

A Real Time Spectrum Analyzer (RTSA) was used to measure the signal strength of the transmitter under a variety of different conditions. The conditions emulate the normal operating conditions of the transmitter. Overall, the signal strength shown on the instrument due to the transmitter ranged from -23 dBm to -55 dBm. Here the larger the number shown equates to a stronger signal (-23 dBm is stronger than -55 dBm). The Receiver RF chip has sensitivity up to -112 dBm. This means it can recognize the carrier-present signal of the Transmitter if the signal has a strength of at least -112 dBm.

This experiment evaluated four conditions for signal strength. First, the Transmitter was tested in free-air outside of the capsule, denoted *TX Test 1*. Second, the transmitter was tested inside of the capsule, denoted *TX Test 2*. Third, the Transmitter was tested while inside the capsule while floating in water in a horizontal orientation, denoted *TX Test 3*. Finally, the floating test (third test) was repeated with the capsule floating in a vertical orientation, denoted *TX Test 4*.

Table 8 contains the results of the transmitting power in free-air and from within the Capsule, TX Test 1 and TX Test 2. These tests were done at several orientations for both the RTSA probe and the transmitter. The number highlighted in red is the stronger signal. The results show that the transmitter signal is well above the required -112 dBm for the receiver in all orientations in both cases. The results also show that there is no substantial degradation of the signal when the transmitter is placed within the capsule. In fact, in several instances the signal was stronger from within the capsule. This is counter-intuitive from what was expected but is an added benefit.

Table 8: Signal Strength Tests in Free Air and within Capsule (TX Test 1 and TX Test 2)

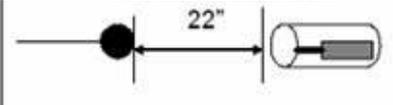
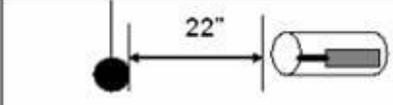
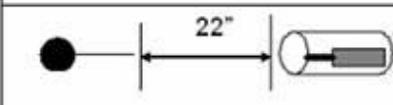
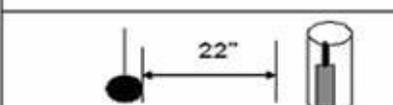
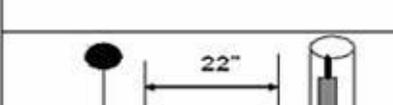
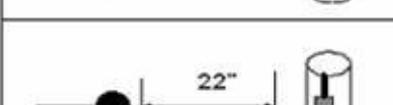
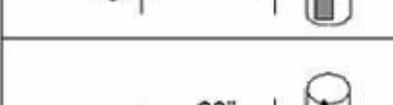
Probe-Capsule Orientation (Overhead View)	Power Seen on RTSA (Free Air)	Power Seen on RTSA (Within Capsule)
	-34 dBm	-31 dBm
	-40 dBm	-55 dBm
	-32 dBm	-29 dBm
	-25 dBm	-30 dBm
	-25 dBm	-23 dBm
	-28 dBm	-43 dBm
	-33 dBm	-41 dBm

Table 9 contains the results of the transmitting power while the capsule is floating in a horizontal orientation, TX Test 3. The RTSA probe was placed in an orientation parallel to the capsule and at an orientation perpendicular to the capsule. For each of these probe orientations, readings were taken with the capsule antenna pointing in four different directions. The resulting signals seen in this test were also well above the necessary strength of -112 dBm. The signal was slightly weaker than the signal tests done out of the water for most comparisons, but still well within the requirement of -112 dBm.

Table 9: Signal Strength Tests For Horizontal Floating Capsule (TX Test 3)

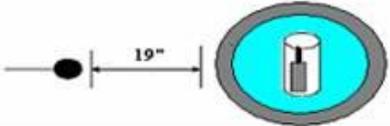
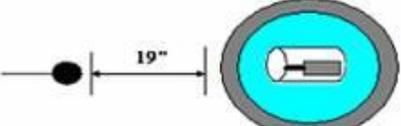
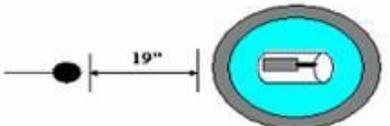
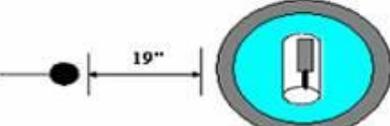
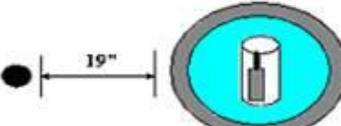
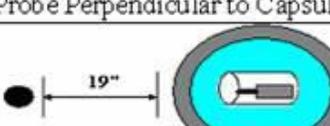
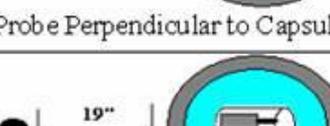
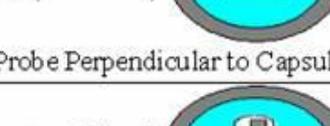
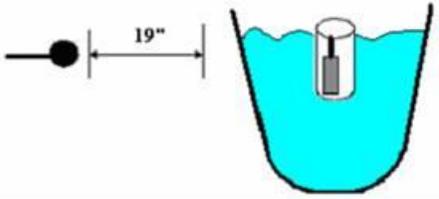
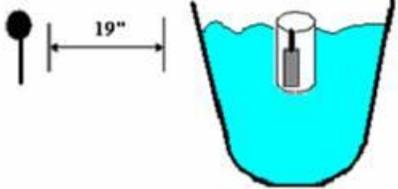
Probe-Capsule Orientation (Overhead View)	Power Seen On RTSA
	-50 dBm
	-51 dBm
	-50 dBm
	-51 dBm
 (Probe Perpendicular to Capsule)	-47 dBm
 (Probe Perpendicular to Capsule)	-42 dBm
 (Probe Perpendicular to Capsule)	-43 dBm
 (Probe Perpendicular to Capsule)	-50 dBm

Table 10 contains the results of the transmitting power while the capsule is floating in a vertical orientation, TX Test 4. Weight was added to the capsule in order to accomplish this orientation. The probe was placed in an orientation both parallel and perpendicular to the capsule. The readings for these tests were also sufficiently strong for the receiver. In comparison to the out-of-water-tests, they were weaker than most readings in Table 8. In comparison to the horizontal floating test, the results were similar with the readings in Table 9 and there is no clear benefit to the vertical orientation over

the horizontal orientation. The horizontal orientation is recommended because of its simpler design and construction requirements.

Table 10: Signal Strength Tests for Vertically Floating Capsule (TX Test 4)

Probe-Capsule Orientation (Side View)	Power Seen On RTSA
	-44 dBm
	-47 dBm

9.1.2 Transmitter-Receiver RF Link

A Transmitter-Receiver RF link was evaluated in order to finalize the choice of the receiver. To verify the Transmitter-Receiver RF link, a known data pattern was provided to the transmitter. Simultaneously, the output of the receiver was monitored for this pattern. The following simple tests shown in Table 11 were completed to achieve this. In each case, the software controlled the logic level on Pin 21 of the microcontroller. This pin serves as the input to the RF Transmitter Chip. An oscilloscope was then used to view the logic level on the output line of the RF Receiver Chip. Results were as expected. In the case where a logical low (0 VDC) was placed on the input of the RF Transmitter Chip some noise was present on the output of the RF Receiver Chip. This noise is noted within the data guide for the RF Receiver Chip and will be handled by the Receiver Unit in software and hardware.

Table 11: Transmitter-Receiver RF Link Tests

	Condition of Input to RF Transmitter Chip	Expected Logic Level seen on output of RF Receiver Chip	Logic Level seen on output of RF Receiver Chip	Test Result
Test 1	Logical High	Logical High	Logical High	Success
Test 2	Logical Low	Logical Low	Low, Some Noise	Success
Test 3	Oscillating High and Low Levels	Oscillating High and Low Levels	Oscillating High and Low Levels	Success

**High Level here means ~3.0 VDC and Low Level means ~0.0 VDC

9.2 Sensor Unit Preliminary Component Testing

The top side of the constructed Sensor Unit prototype is shown in Figure 50 and the bottom side is shown in Figure 51. There are two extra LEDs on the transmitter. These LEDs were simply added for testing the tilt switch, which will be explained shortly and will not be part of the final Sensor Unit. The Sensor Unit used in this testing also does not show the additional tilt switches and UART circuitry added later in the design process. Below is a list of the basic hardware tests done and their outcomes.

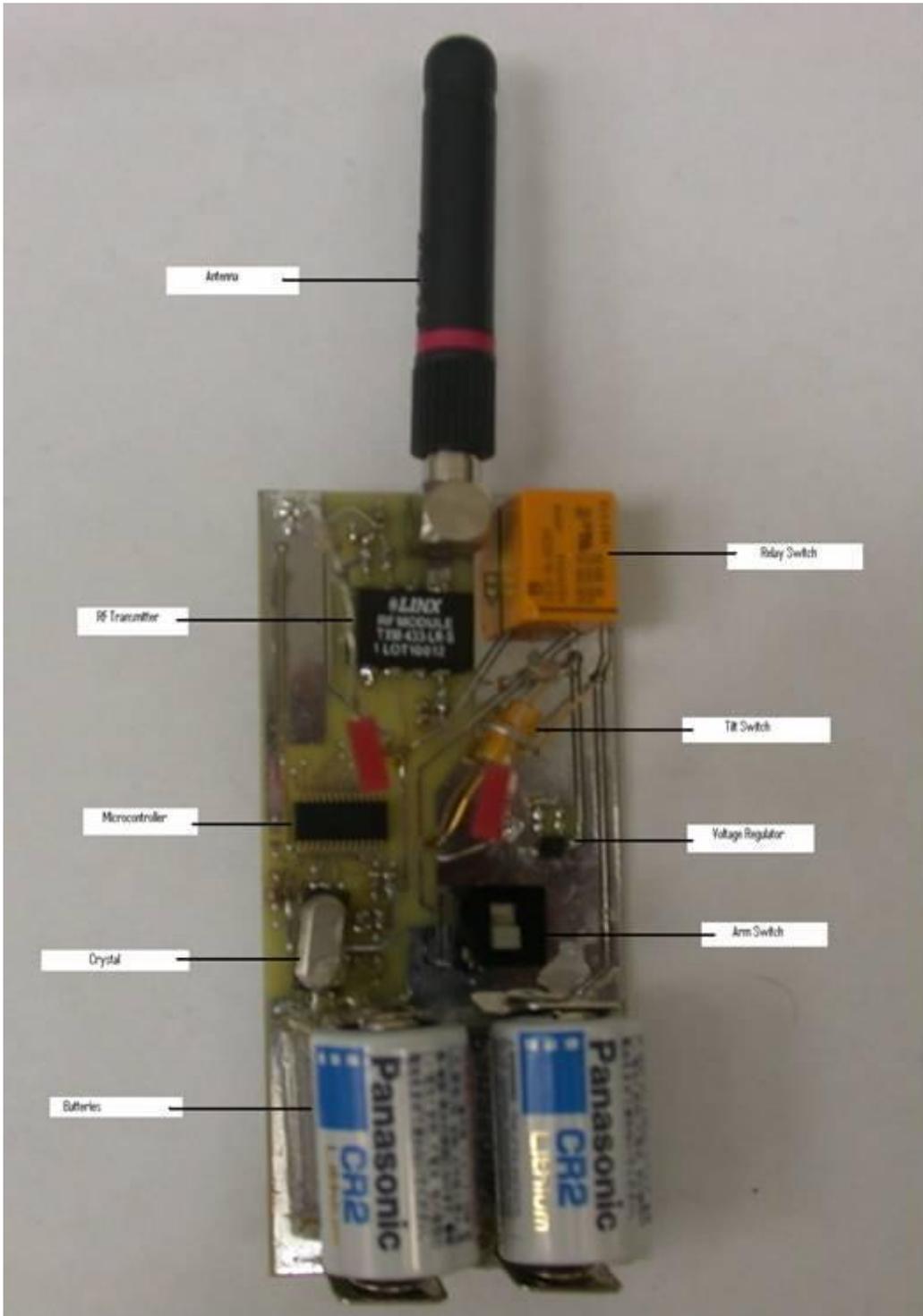


Figure 50: Transmitter Prototype - top side



Figure 51: Transmitter Prototype - bottom side

9.2.1 Battery Test

The batteries are used to power the Sensor Unit. The trace connected to the positive potential of the batteries was probed using a multimeter to determine the voltage output.

Table 12: Battery Test

	Condition	Expected Outcome	Observed Outcome	Test Results
Test 1	Batteries Added	-3 VDC on output trace	-3 VDC on output trace	Success

9.2.2 Tilt Switch

The tilt switch is used to connect the batteries to the arm switch when in the correct orientation (after release from soil). The common electrode is connected to the arm switch and serves as the output of the tilt switch. The common electrode of the tilt switch was tested using a multimeter and an LED. The Sensor Unit is in a vertical position in Figure 52 and a horizontal position in Figure 53. The LED correctly only lights in the horizontal position.

Table 13: Tilt Switch Test

	Tilt Switch Condition	Expected Tilt Switch Outcome	Observed Tilt Switch Outcome	Test Results
Test 1	Vertical Orientation (See Figure 52)	-0 VDC on common electrode -LED not lit	-0 VDC on common electrode -LED not lit	Success
Test 2	Horizontal Orientation (See Figure 53)	-3 VDC on common electrode -LED lit	-3 VDC on common electrode -LED lit	Success



Figure 52: Tilt Switch Test Vertical Orientation

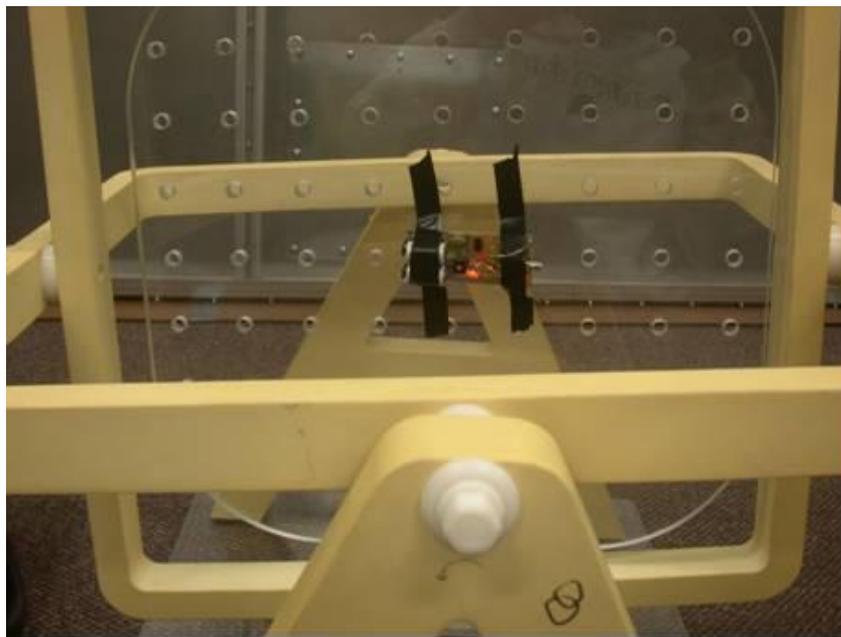


Figure 53: Tilt Switch Test Horizontal Orientation

9.2.3 Arm Switch

The output of the arm switch is used to operate the relay switch. The state of Pins 2 and 5 of the arm switch are used to control the relay switch. These two pins were tested for their voltage output in each of the arms switch's three positions using a multimeter. Table 14 displays the results.

Table 14: Arm Switch Test

	Arm Switch Condition	Expected Arm Switch Outcome	Observed Arm Switch Outcome	Test Results
Test 1	“Armed State” Connects pin 2 to 3 Connects pin 5 to 6	3 VDC on pin 5 0 VDC on pin 2	3 VDC on pin 5 0 VDC on pin 2	Success
Test 2	“Reset State” Connects pin 2 to 1 Connects pin 5 to 4	0 VDC on pin 5 3 VDC on pin 2	0 VDC on pin 5 3 VDC on pin	Success
Test 3	“Off State” No pins interconnected	0 VDC on pin 5 0 VDC on pin 2	0 VDC on pin 5 0 VDC on pin 2	Success

9.2.4 Relay Switch

The relay switch has two states. In one state, the common pin (Pin 2) is connected to the batteries (Pin 1). In the second state, the common pin is connected to nothing. The output of the common pin was tested using a multimeter and an LED. Figure 54 shows the relay switch in a horizontal orientation and Figure 55 shows the relay switch in a vertical orientation and Table 15 displays the results of these tests.

Table 15: Relay Switch Test

	Relay Switch Condition	Expected Relay Switch Outcome	Observed Relay Switch Outcome	Test Results
Test 1	3 VDC on pin 1 3 VDC on pin 4 0 VDC on pin 5 (See Figure 54)	3 VDC on pin 3 LED Lit	3 VDC on pin 3 LED Lit	Success
Test 2	Follows Test 1 3 volts DC on pin 1 0 volts DC on pin 4 0 volts DC on pin 5 (See Figure 55)	3 VDC on pin 3 LED Lit	3 VDC on pin 3 LED Lit	Success
Test 3	3 volts DC on pin 1 0 volts DC on pin 4 3 volts DC on pin 5	0 VDC on pin 3 LED not Lit	0 VDC on pin 3 LED not Lit	Success
Test 4	Follows Test 3 3 volts DC on pin 1 0 volts DC on pin 4 0 volts DC on pin 5	0 VDC on pin 3 LED not Lit	0 VDC on pin 3 LED not Lit	Success

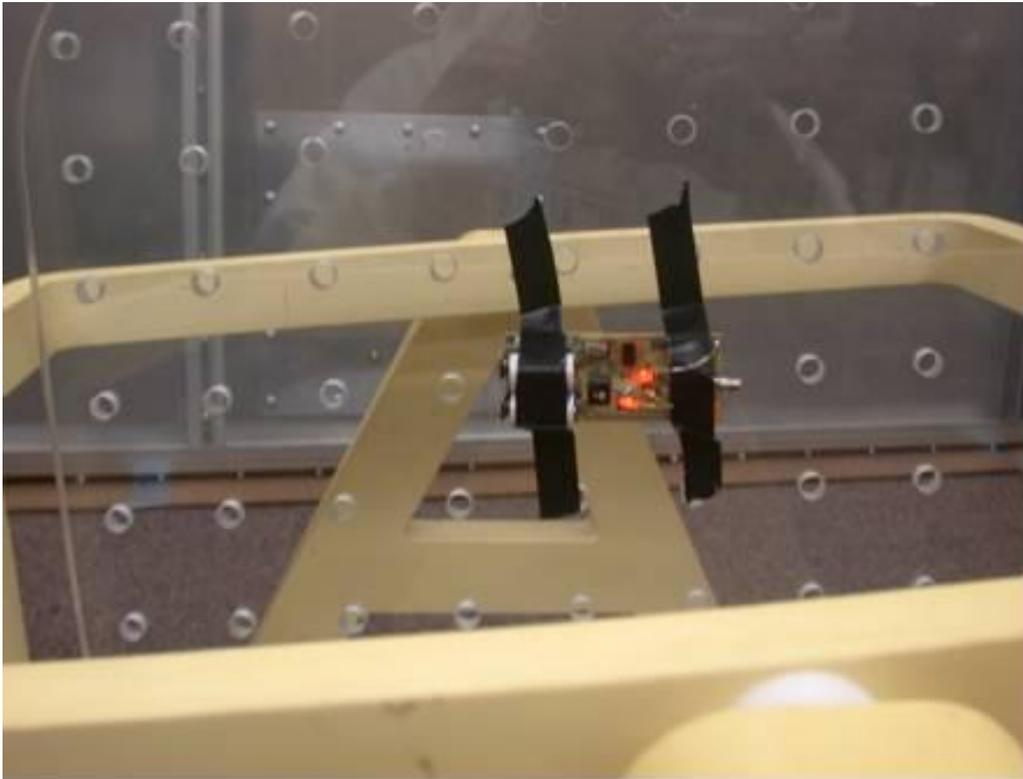


Figure 54: Relay Switch Test in Horizontal Orientation

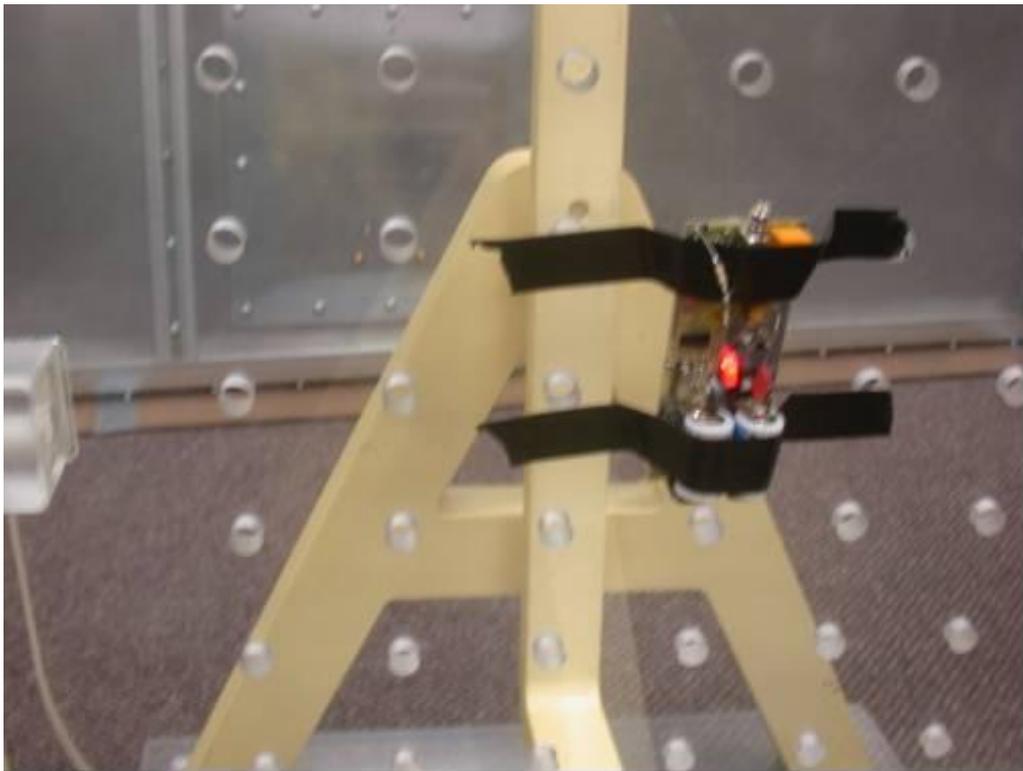


Figure 55: Relay Switch Test in Vertical Position

9.3 Receiver Unit Preliminary Component Testing

The preliminary prototype of the Receiver Unit is shown below in Figure 56. The Receiver Unit was able to be programmed and to receive RF data. The voltage regulator test is the only component test unique to the Receiver Unit and is explained below.

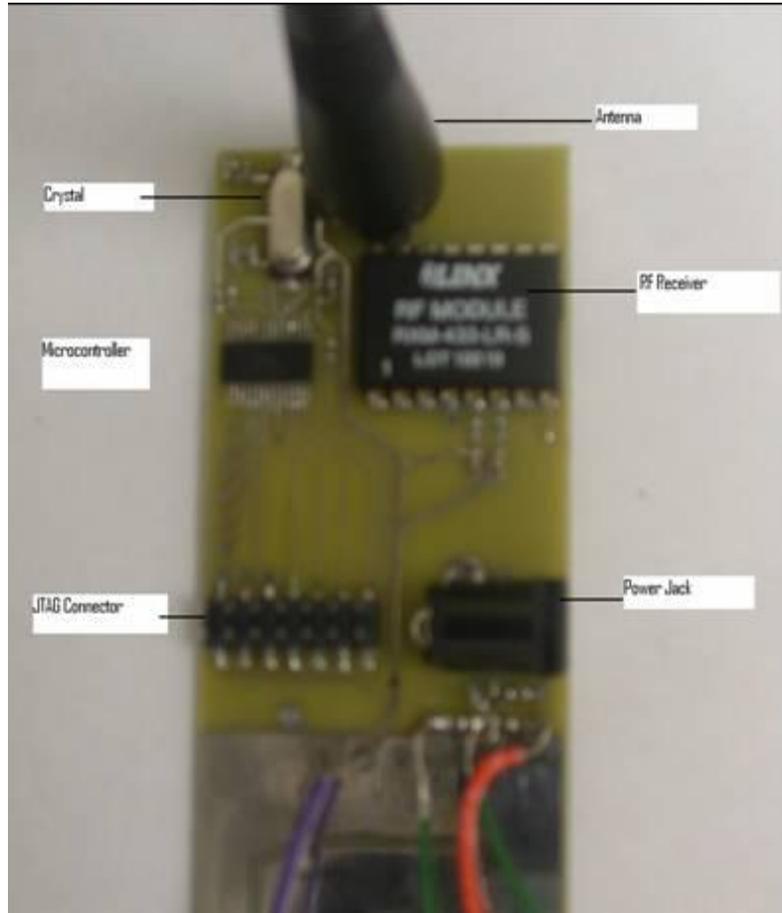


Figure 56: Receiver Unit Prototype

9.3.1 Voltage Regulator (3.0 Volts DC)

The voltage regulator connected to the power jack must be able to take 3.0 VDC to 16.0 VDC input range and output 3.0 VDC output.

Table 16: Voltage Regulator Test

	Voltage Regulator Input Condition	Expected Output	Observed Result	Test Outcome
Test 1	3.0 VDC input	3.0 VDC output	3.0 VDC output	Success
Test 2	16.0 VDC input	3.0 VDC output	3.0 VDC output	Success

9.4 Light Indicator Preliminary Prototype Testing

The prototype of the Light Indicator is shown in Figure 57. A simple test jig to emulate the Receiver Unit was developed to test the Light Indicator. The Light Indicator was connected to four DIP-switch modules. Each DIP-switch module contains eight switches, of which four are used to represent a single Receiver Unit. The four DIP-switch modules emulate four independent Receiver Units. Each switch represents one of the I/O pins of a single Receiver Unit that would control a corresponding LED on the Light Indicator. One of the switch inputs was connected to a 3.0 VDC power source and the output pin was connected to the Light Indicator. Each switch was toggled and the corresponding LED was observed. All inputs lit the correct LED. Figure 57 shows the test setup for the Light Indicator Unit.

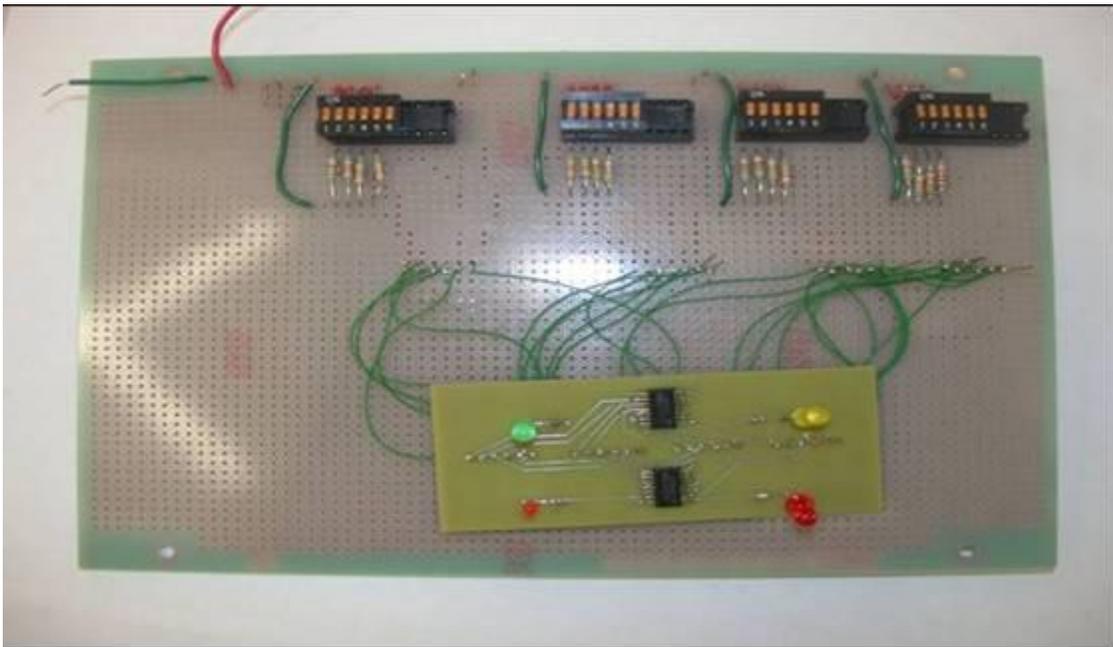


Figure 57: Light Indicator Unit Prototype and Test Stand

9.5 Capsule Testing

The Sensor Unit capsule has three functions it must perform. First, it must be watertight in order to prevent water from damaging the electronics of the Sensor Unit. Second, it must be able to rise to the surface quickly. Lastly, it must be able to float on the surface of the water. The sections below describe tests done to confirm these properties.

9.5.1 Buoyancy

The capsule naturally floats as shown in Figure 58. The capsule continues to have some portion of its shell un-submerged (above the water surface) until over 6 oz. of weight has been added. This confirms its ability to float.



Figure 58: Capsule Floating with No Additional Weight Added

9.5.2 Rise Times

Rise time tests were done to simulate the release of the capsule and measure how quickly the capsule will rise to the surface. A large bucket was used to do this. A housing was also built to place the capsule in while under water. This housing was composed of 3-inch diameter PVC pipe attached to a wooden base. The 3-inch interior diameter PVC pipe is similar to the 3-3/16 inch diameter hollow stem auger that will eventually be used. To perform the tests, the bucket was filled with water up to a height of 29 inches. Then, the capsule was pushed down this pipe and held down using a pole, as shown in Figure 59. The time it took the capsule to rise once the pole was lifted was then recorded. It took the capsule 1.33 seconds with nothing inside. With the transmitter board added, it took 1.635 seconds. These times are similar to rise times using simulation. Figure 59 and Figure 60 illustrate this test.



Figure 59: Rise Time Test Before Capsule Release



Figure 60: Rise Time Test After Sensor Release

9.5.3 Water Tightness

Water tightness tests were performed on the capsule. For each of these tests, weights were placed within the capsule as shown in Figure 61 that would sink the capsule. Once the capsule was sealed, it was left underwater for three days. Following the three days, the capsule would be opened and inspected for water.



Figure 61: Weight Added to Keep Capsule Submerged for Water Tightness tests.

The first test performed only used the threaded caps to seal the capsule. The result of this test was unsuccessful. A substantial amount of water collected within the capsule for this test. The water collected is shown in Figure 62.



Figure 62: Water Collected from the First Water Tight Test Using Only the PVC Pipe.

The second test was done with Teflon tape placed on the threads of the capsule as shown in Figure 63. The result of the test was greatly improved in comparison with the unthreaded test. However, a small amount of water could be seen on the wall of the capsule.



Figure 63: Capsule with Teflon Tape Added.

It was estimated at this point that the moisture was due to condensation. Therefore, Silica Gel Desiccants were used in conjunction with the Teflon tape. A packet of this is shown in Figure 64. The Silica Gel Desiccants absorb small amounts of moisture that may be present within the air. They are commonly used in several storage capacities. Two of the packets were used in the test. The result of this test was a success because no moisture was observed after three days underwater. Additionally, the Silica Gel Desiccants change color as they near the limit of moisture they can absorb. The Silica Gel Packets did not change color and therefore are capable of absorbing more moisture.



Figure 64: One Silica Gel Desiccant Packet

9.5.4 Pressure Testing

The unit will be buried at a certain depth below water. This causes a pressure to be exerted on the capsule depending on that depth. Therefore, a test was performed to verify that the capsule would remain watertight under the pressure caused by water depth. To begin, the amount of pressure the capsule would experience at a depth was calculated. For this test, a depth of 100 feet was used as a limit. Using the equation below, the pressure can be calculated.

$$P_2 = P_1 + \rho gh \quad (13)$$

Here P_2 is the pressure on the capsule at the depth below water the capsule is buried. P_1 is the pressure at sea level, which is 101000 Pa. ρ is the density of water, here 1000 Kg/m³. g represents gravity, which is 9.8 m/s². Lastly, h is the depth below water that the capsule is buried. In this case, h is 100 feet or 30.48 meters. The result after converting to imperial units is around 58.17 lb/in².

To test if the capsule remained watertight at this pressure, air was added to the inside of a capsule until the internal air pressure was 60 lb/in². The capsule was submerged in water. If the capsule is watertight, no bubbles would be produced by the capsule. To pressurize the capsule interior, a valve was added to the side of the capsule using silicon sealant. Next, a hose connecting the valve to an air pump was connected. The hose-valve connection was sealed using a clamp. This is shown in Figure 65. The capsule was then placed under water. The pump was turned on until the pressure gauge on the pump reached roughly 60 lb/in² at which point the capsule valve was closed and the pump was turned off. Then the test jig was monitored for bubbles coming out of the capsule. No bubbles were observed. This confirms the ability of the capsule unit to withstand water pressure.



Figure 65: Capsule with Valve and Hose Attachment for Pressure Testing

9.6 System Functionality Test

The section below shows tests done to verify the functionality of the system.

9.6.1 Overall System Demonstration

This section describes the demonstration that was run to verify system functionality. The goals realized from the demonstration were as follows:

1. Program Sensor Unit with a Sensor Unit Data Block
2. Arm and Encapsulate Sensor Unit
3. Bury Sensor Unit under soil/aggregate using hollow 3 inch diameter pipe in a water environment
4. Uncover Sensor Unit resulting in its rise to the surface and activation
5. Receive and Decode the Sensor Unit Data Block on the Receiver Unit
6. View visual confirmation of Transmission and Decode on the Light Indicator

The demonstration took place within a lab environment within Benedum Hall. The Sensor Unit Data Block used for the demonstration was a dummy block randomly created. The only piece of the Sensor Unit Data Block critical to the demonstration was that of the Color Code. For this demonstration, the Color Code was given a value of 0x01. This corresponds to the activation of the yellow LED on the Light Indicator. A picture of what the test Sensor Unit Data Block looks like in memory is shown in Figure 66. The Sensor Unit Data Block start address is 0x1080 and the end address is 0x1097.

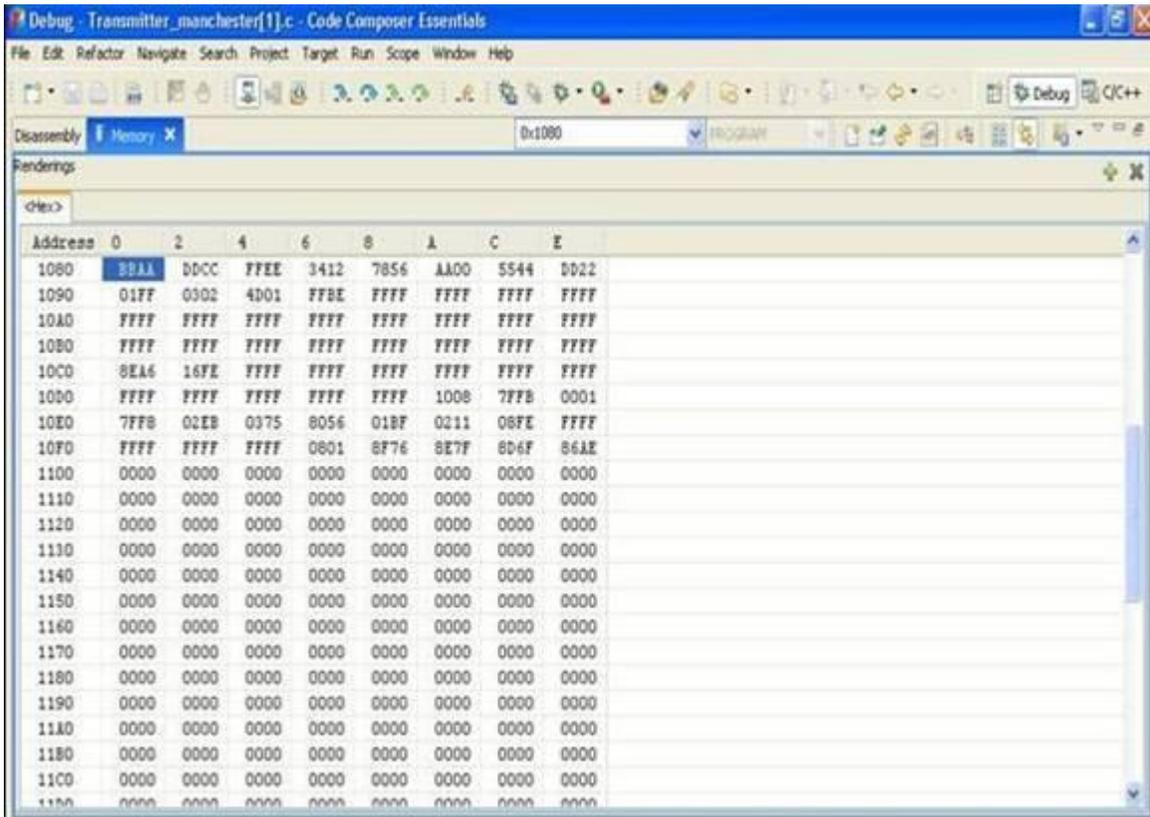


Figure 66: Transmitter Memory View of Dummy Data Block

The Sensor Unit is shown in Figure 67. This Sensor Unit has three tilt switches located on it. Two of the tilt switches can be seen on the front of the PCB while the third is located on the back. Once the Sensor Unit is armed using the arm switch and the Teflon tape is applied to the threads, the caps can be screwed on. The Sensor is now encapsulated and ready to be deployed. Black tape was also referenced on the caps for indication of the orientation of the PCB within the capsule while floating.

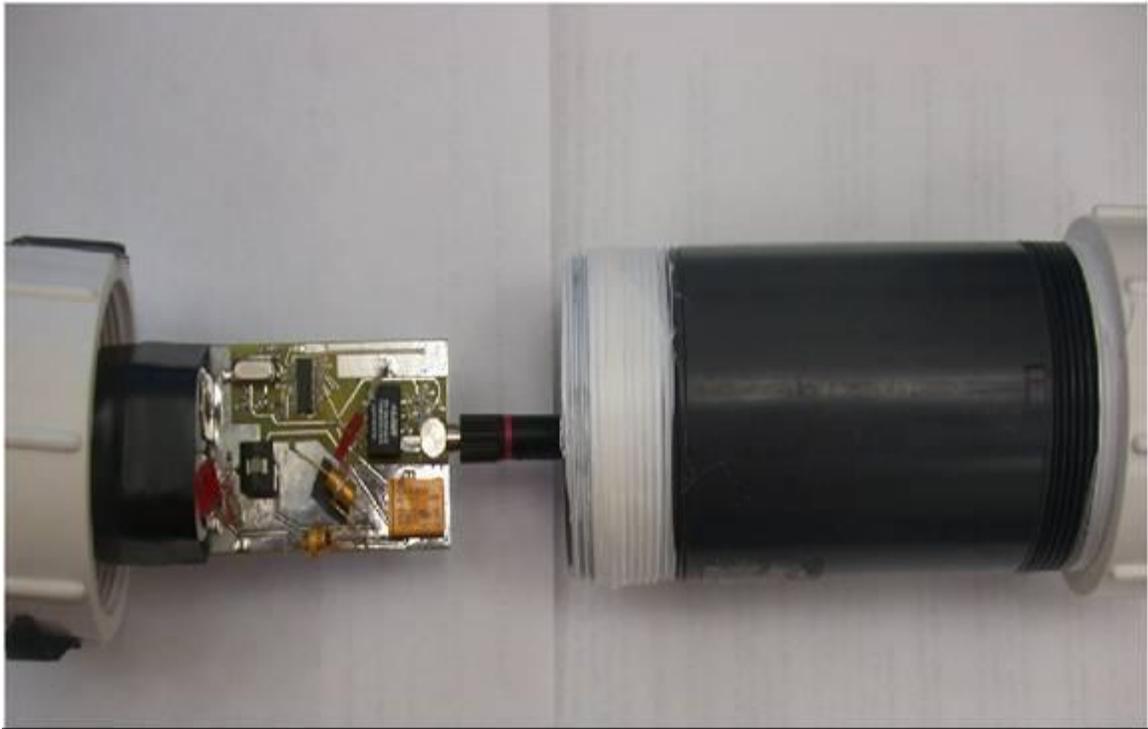


Figure 67: Sensor Unit Ready for the Final Assembly Step

The physical setup that the Sensor Unit was placed in was a large yellow container, rock aggregate, and a 3-inch interior diameter by 5-foot length PVC pipe. The pipe was used to simulate the hollow stem auger that will be used in practice. This pipe was extended to the bottom of the container. It was surrounded by the rock aggregate as shown in Figure 68. The container was then filled with water as shown in Figure 69.



Figure 68: Container with Aggregate and Pipe



Figure 69: Container with Aggregate, Pipe, and Water

The Receiver Unit and Light Indicator were also placed in the same room as shown in Figure 70. This version of the Receiver Unit can be connected through a power jack to a wall outlet. When initially plugged in it initiates the Light Indicator to light the green LED. In practice, the green LED may not be lit to begin given that four Sensor Units are assigned to the Receiver Unit. In this test, the green LED was lit to give a visual indication that the Receiver Unit successfully powered up and began operation.

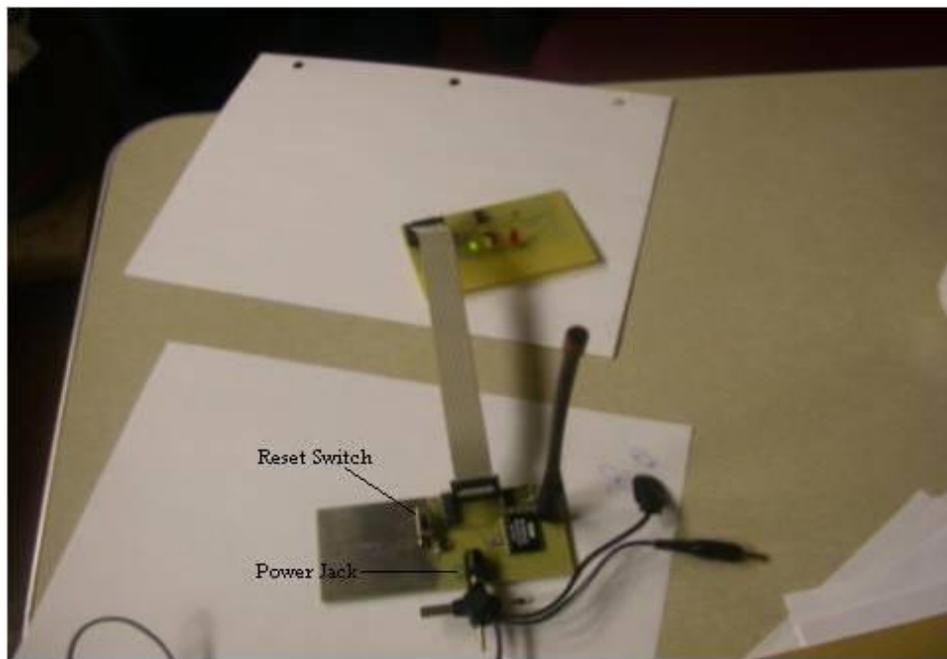


Figure 70: Receiver Unit and Light Indicator with No Message Received from a Sensor Unit.

The armed Sensor Unit was then lowered down the pipe and pushed down to the bottom of the container using a long pole. Aggregate was then dropped down the pipe on top of the Sensor Unit to bury it. The pipe was then pulled up out of the aggregate leaving the Sensor Unit buried. The aggregate was then slowly brushed away until the Sensor Unit released itself and rose to the surface of the water as shown in Figure 71. This resulted in the Light Indicator quickly lighting the yellow LED as shown in Figure 72. This proves the success of system functionality.



Figure 71: Sensor Unit After Release from Aggregate

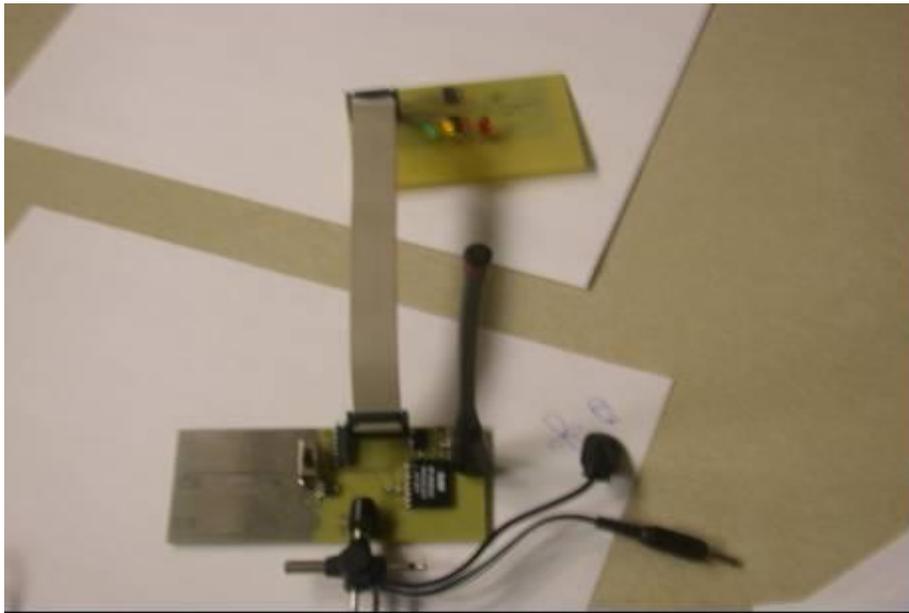


Figure 72: Receiver and Light Indicator After Sensor Unit Release

9.6.2 Multiple Sensor Functionality

This section shows images of the current system operating using four different Sensor Units. Each of the Color Codes (green, yellow, orange, red) will be represented by one of the four Sensor Units. This demonstration does not take place in a water environment because the Sensor Unit has already been shown to have proper functionality within a water environment. For this demonstration the main concerns deal with ensuring that the Receiver Unit triggers the correct output given multiple Sensor Units as well as the system's ability to function properly with multiple Sensor Units transmitting at the same time. It should be noted that Sensor Units are placed very close to the Receiver Unit so that all components can easily be illustrated and photographed. This is not due to the system's inability to operate at greater distances.

Demonstration Setup: Figure 73 shows the setup of the demonstration. At the bottom of the figure are the four Sensor Units used. Written above each Sensor Unit is the Color Code that was programmed to it. At the top of the figure is the Receiver Unit connected to the current Light Indicator. When any of the Sensor Units are active, the red LED on the Sensor Unit will be lit.

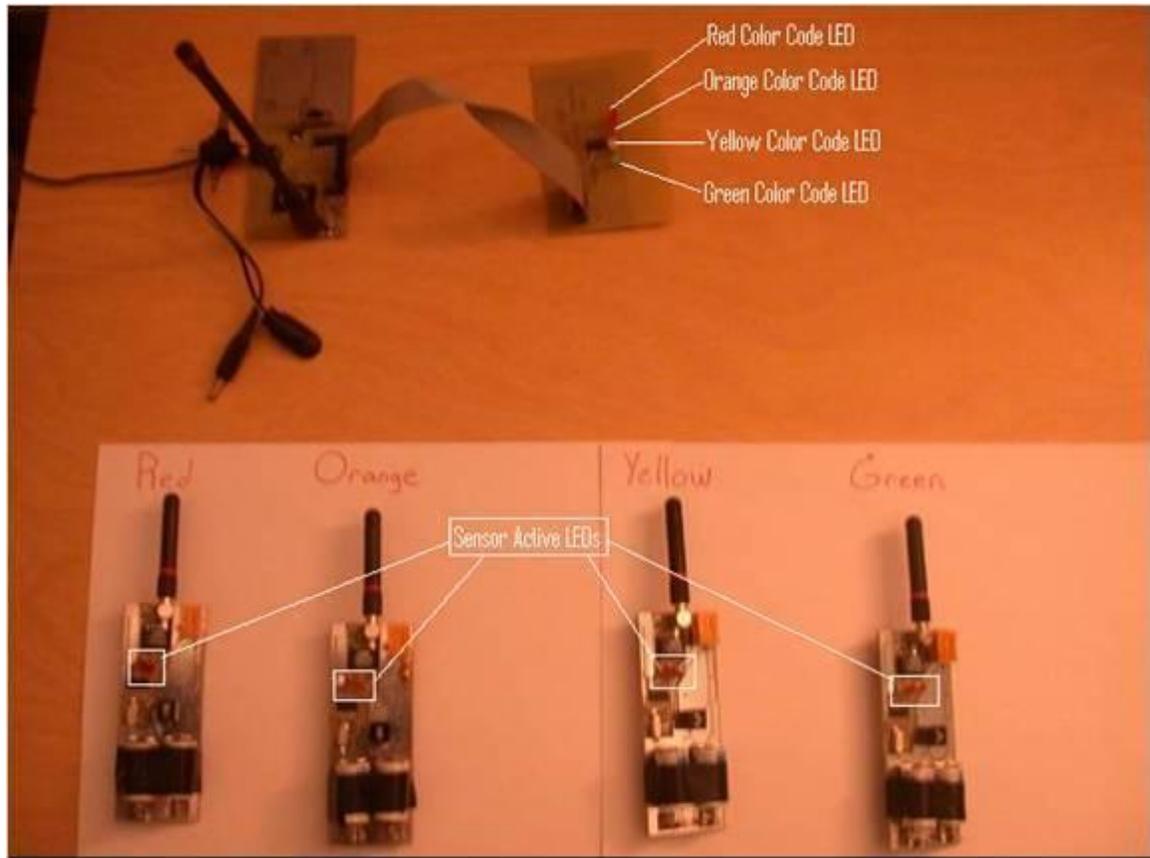


Figure 73: Demonstration Setup with No Sensor Unit Active

Step 1: Green Sensor Unit Activation: Figure 74, below, shows the first action taken in the system demonstration. This action was activating the Sensor Unit programmed with the green Color Code. In the figure, the green Sensor Unit's activation is shown by its

red LED being lit. The proper response by the Receiver Unit is also indicated by the green LED being lit on the Light Indicator. This step was successful.

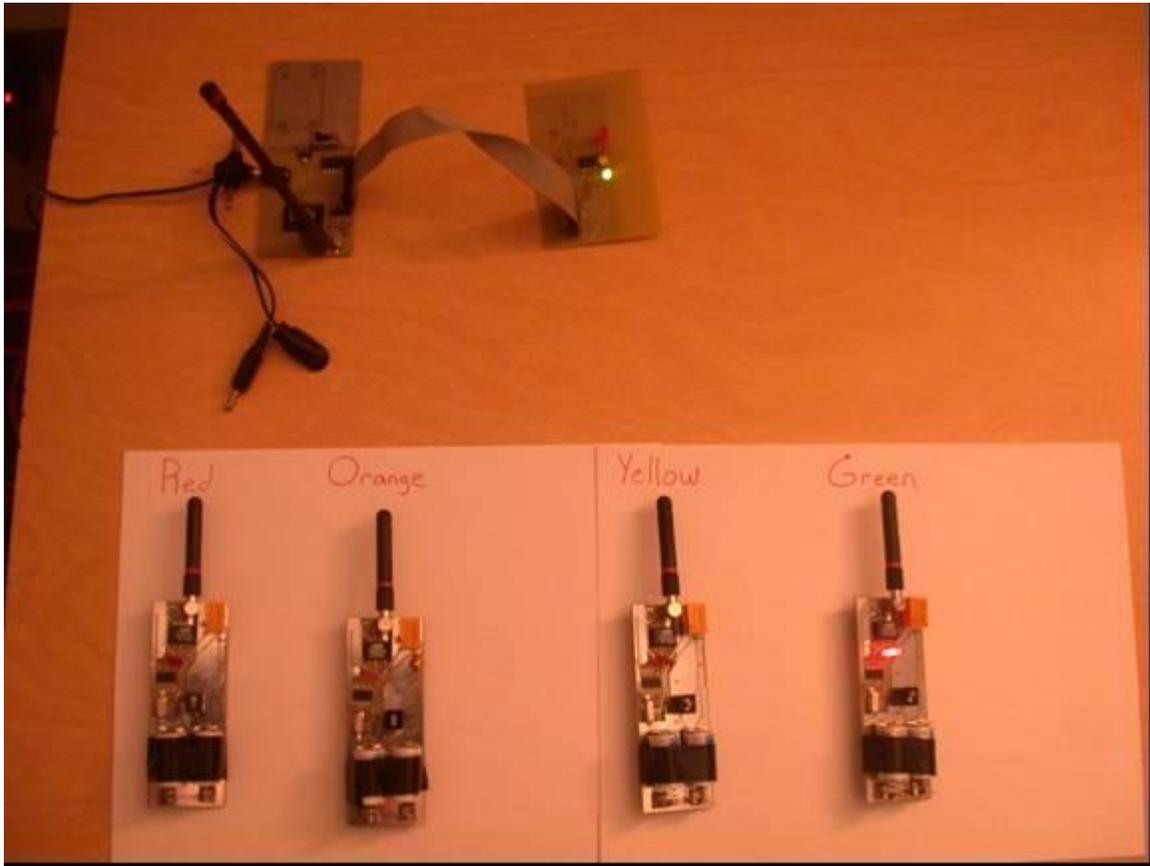


Figure 74: Step 1 - Green Sensor Unit is Activated. Note that the Light Indicator Shows the Green Condition.

Step 2: Yellow Sensor Unit Activation: Figure 75, below, shows the second action taken in the system demonstration. This action was activating the Sensor Unit programmed with the yellow Color Code. In the figure, the yellow Sensor Unit's activation is shown by its red LED being lit. Notice also that the green Sensor Unit is lit and active which would expose the system to collisions. The proper response by the Receiver Unit is also indicated by the yellow LED being lit on the Light Indicator. This proves that the anti-collision protocol was successful and a proper response by the Receiver Unit state machine. This step was successful.

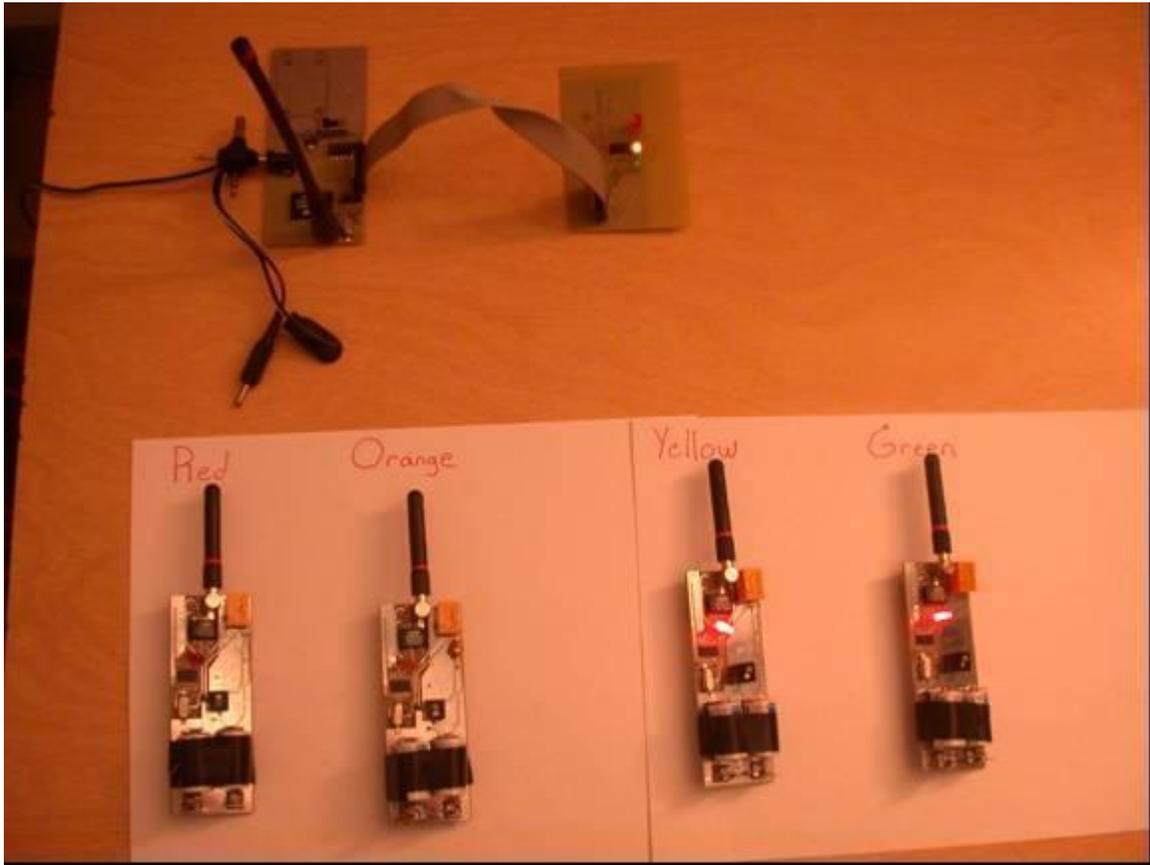


Figure 75: Step 2 - Yellow Sensor Unit is Activated with Green Sensor Unit Already Active. Note that the Light Indicator Shows the Yellow Condition.

Step 3: Orange Sensor Unit Activation: Figure 76 below shows the third action taken in the system demonstration. This action was activating the Sensor Unit programmed with the orange Color Code. In the figure, the orange Sensor Unit's activation is shown by its red LED being lit. Notice also that green and yellow Sensor Units are lit and active, which would expose the system to collisions. The proper response by the Receiver Unit is indicated by the orange LED being lit on the Light Indicator. This proves that the anti-collision protocol can handle three Sensor Units transmitting at the same time and a proper response by the Receiver Unit state machine. This step was successful.

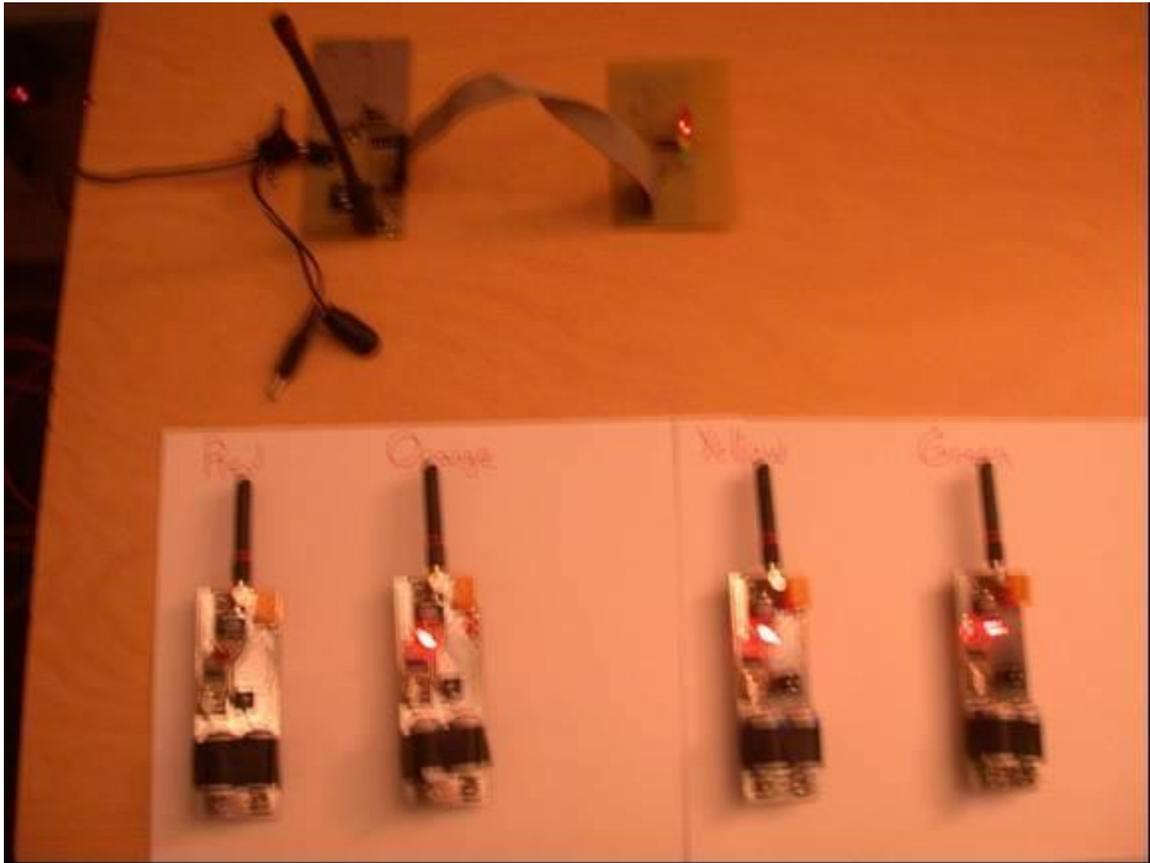


Figure 76: Step 3 - Orange Sensor Unit is Activated with the Green and the Yellow Sensor Units Already Active. Note that the Light Indicator Shows the Orange Condition.

Step 4: Orange Sensor Unit Activation: Figure 77 below shows the fourth action taken in the system demonstration. This action was activating the Sensor Unit programmed with the red Color Code. In the figure, the red Sensor Unit's activation is shown by its red LED being lit. Notice also that green, yellow, and orange Sensor Units are lit and active exposing the system to collisions. The proper response by the Receiver Unit is also indicated by the red LED being lit on the Light Indicator. This proves that the anti-collision protocol allows four Sensor Units to be active at the same time while still allowing the messages to reach the Receiver Unit. The Receiver Unit state machine functioned according to specifications by lighting and maintaining the red LED when lower Color Code messages (orange, yellow, and green) were received. This step was successful.

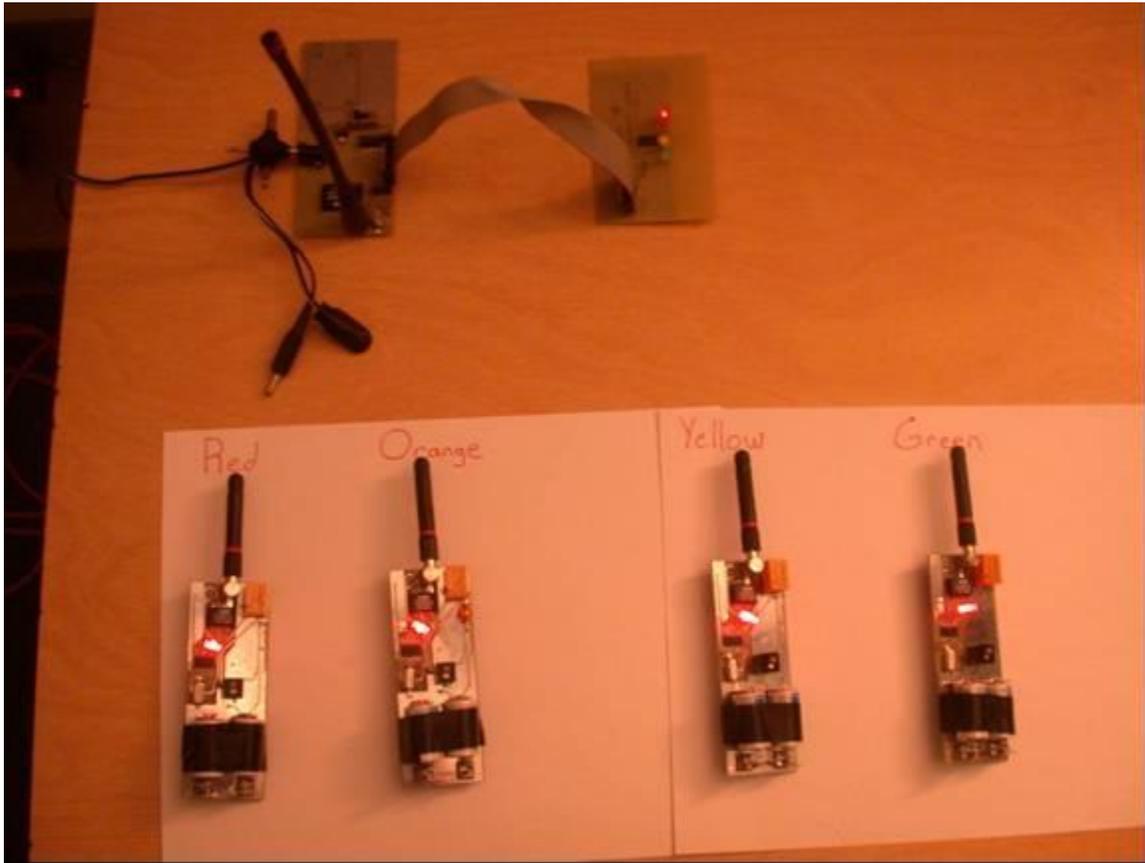


Figure 77: Step 4 - Red Sensor Unit is Activated with the Orange, Yellow, and Green Sensor Units Already Active. Note that the Light Indicator Shows the Red Condition.

9.7 UART Interface Functionality Tests

This section details the testing of the UART functionality of both the Sensor Unit and Receiver Unit. The UART of both units is accessed through a USB connection. A USB to serial chip converts the signals between the UART and USB. The UART allows for communication with the units while they are active. This gives a technician the ability to input and extract data from the units. The Sensor Unit and Receiver Unit each have a protocol for communication with the microcontroller. The following subsections will details this protocol and its testing.

9.7.1 Sensor Unit Serial Interface Protocol and Testing

This section details the protocol used for interface with the Sensor Unit microcontroller and its testing. Below, the available functions are listed along with how to invoke them.

9.7.1.1 Send Sensor Unit Data Block Entry

The Sensor Unit Data Block is composed of 21 bytes of data (2 byte CRC is calculated). This is 168 bits of data. To transfer this data, the data string is represented as 42 hexadecimal (HEX) characters (base 16, 0-9 A-F). Each HEX character represents four bits of information. The first two HEX characters represent the first byte (8 bits make up one byte) of the Sensor Unit Data Block, the next two HEX characters represent the second byte, and so forth. Once the 42 HEX characters have been formed they can be transferred to the Sensor Unit by two means:

1. Text file Transfer
 - a. Place the 42 HEX characters preceded by two asterisks (**) at the beginning of a file.
 - b. Save the file as a .txt file type (text file). A simple text editor such as Notepad or WordPad can be used.
 - c. Within HyperTerminal™ click Transfer→Send Text File.
 - d. Choose the .txt file containing the Sensor Unit Data Block, this is illustrated in Figure 78.
 - e. The file will be transferred to the Sensor Unit and “data_txed” will be displayed in the HyperTerminal™ window. This is shown in Figure 79.
2. Character by Character Entry
 - a. Enter the asterisk character (“*”) two times in a row within the HyperTerminal™ interface. This notifies the microcontroller that the next 42 hex characters are the Sensor Unit Data Block and should be stored as such.
 - b. Enter the Sensor Unit Data Block one hex character at a time.

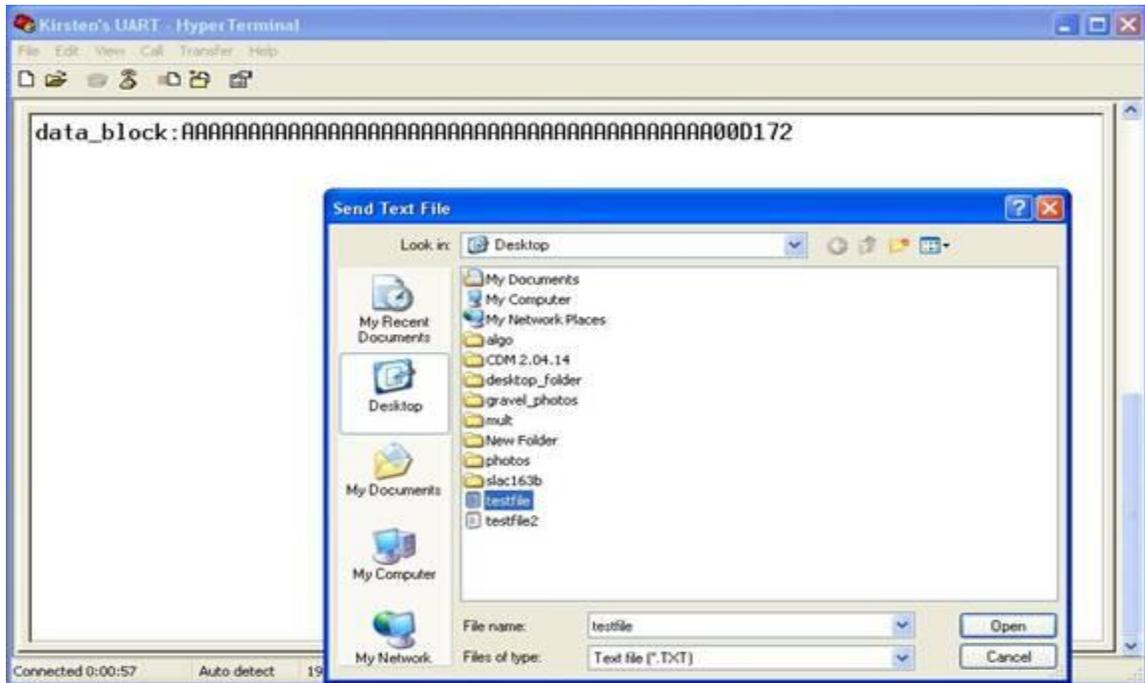


Figure 78: Transmitter UART Communication: Transfer of Data Block from txt File

9.7.2 Receiver Unit UART Protocol

This section describes testing of the protocol used for interface with the Receiver Unit microcontroller and its testing. Below the available functions are listed along with how to invoke them.

9.7.2.1 Bridge ID Entry

The Receiver Unit has a stored Bridge ID to compare with the Bridge ID contained within the Sensor Unit Data Block transferred. The Bridge ID portion of the Sensor Unit Data Block is composed of 14 bytes of data. This is 112 bits of data. To transfer this Bridge ID to the microcontroller, the data string is represented as 28 HEX characters (base 16, 0-9 A-F). The first two HEX characters represent the first byte of the Bridge ID; the next two HEX characters represent the second byte, and so forth. Once the 28 HEX characters have been formed they can be transferred by two means.

1. Text File Transfer
 - a. Place the 28 HEX characters preceded by two asterisks (**) at the beginning of the file.
 - b. Save the file as a .txt file type (text file). A simple text editor such as Notepad or WordPad can be used.
 - c. Within HyperTerminal™ click Transfer→Send Text File. This is the same procedure as Step 1.c for the Sensor Unit serial interface.
 - d. Choose the .txt file (text file) containing the Bridge ID. This is the same procedure as Step 1.d for the Sensor Unit serial interface.
 - e. The file will be transferred to the Receiver Unit and “data_txed” will be displayed in the HyperTerminal™ window. This is the same procedure as Step 1.e for the Sensor Unit serial interface.
2. Character by Character Entry
 - a. Enter the asterisk character (“*”) two times in a row within the HyperTerminal™ interface. This notifies the microcontroller that the next 28 HEX characters are the Bridge ID and should be stored as such.
 - b. Enter the Bridge ID one HEX character at a time.

9.7.2.2 Test Bridge ID Entry

A Bridge ID can be stored in the Receiver Unit for testing purposes. If a Sensor Unit transmits this test Bridge ID, the Receiver Unit will drive all LED I/O lines to verify functionality of the unit. Using a test Sensor Unit programmed with this Bridge ID accomplishes the task of verifying that the Receiver Unit can both accept messages and drive the LED lines properly. Once this test is complete, the Receiver Unit should be reset.

The method for entering the test Bridge ID that the Receiver Unit will recognize is similar to a normal Bridge ID entry. However, two “\$” characters will be used instead of the “*” characters. The two means of entry using the “\$” character are described below.

1. Text File Transfer

- a. Place the 28 HEX characters preceded by two dollar signs (\$\$) at the beginning of file.
 - b. Save the file as a .txt file type (text file). A simple text editor such as Notepad or WordPad can be used.
 - c. Within HyperTerminal™ click Transfer→Send Text File. This is the same procedure as Step 1.c for the Sensor Unit serial interface.
 - d. Choose the .txt file (text file) containing the Bridge ID. This is the same procedure as Step 1.d for the Sensor Unit serial interface.
 - e. The file will be transferred to the Receiver Unit and “data_txed” will be displayed in the HyperTerminal™ window. This is the same procedure as Step 1.e for the Sensor Unit serial interface.
2. Character by Character Entry
- a. Enter the dollar sign character ('\$') two times in a row within the HyperTerminal™ interface. This notifies the microcontroller that the next 28 HEX characters are the Bridge ID and should be stored as such.
 - b. Enter the Bridge ID one HEX character at a time.

9.7.2.3 Bridge ID Read Out

Verification of the Bridge ID entered can be done simply by entering “#” followed by a 1 at any time during the interface. The Bridge ID is displayed as a sequence of HEX characters. The Bridge ID is initialized to all HEX A characters. The read out of this is shown in Figure 81.

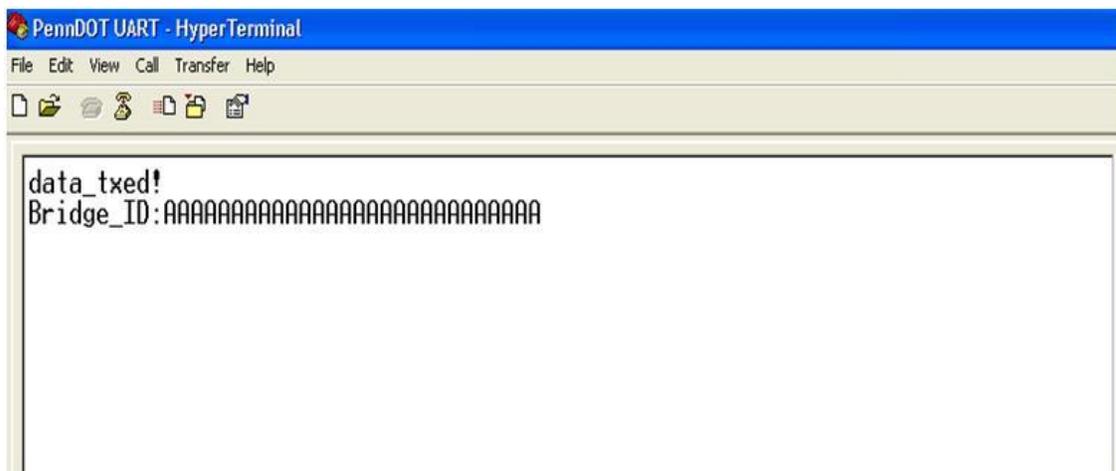


Figure 81: Bridge ID Transfer and Read Out

9.7.2.4 Sensor Unit Data Blocks Received Read Out

When a Sensor Unit Data Block is accepted by the Receiver, it is stored to the flash memory of the Receiver Unit microcontroller. Each block received is stored consecutively after the last. Entering “#” followed by a “2” at any time during the interface will result in a print out of all Sensor Unit Data Blocks received. Each Sensor Unit Data Block is placed on a new line with the first block read out corresponding to the first block received by the Receiver Unit. This is shown in Figure 82.

10.0 Assembly and Installation

This section contains instructions for the assembly of the three main units of the system and installation of the Sensor Unit.

10.1 System PCB Assemblies

There are three main components that must be assembled for the system. These components are the Sensor, Receiver, and Light Indicator Units. Each of these components requires that a printed circuit board or PCB have components added to its surface. Components are added to the printed circuit board through soldering. For the surface mount chips, surface mount machines can be used. However, all chips are sufficiently large in size so hand soldering is also reliable. The .pcb file or printed circuit board file representing the printed circuit board of each unit provides a means to correspond the correct orientation and placement of each component. Within ExpressPCB, each component and its pads can be clicked on so that information regarding the name of the component and the pin clicked will come up. The .pcb file directly corresponds to the .sch or schematic file such that the schematic file can also be used for help. Failure to follow the placement specified by these files will result in malfunction of the unit.

10.2 Sensor Unit Encapsulation and Installation

The encapsulation and installation of the Sensor Unit has several particular steps that must be followed. This section will go through each step in detail.

10.2.1 Step 1 - Sensor Unit PCB Preparation

The Sensor Unit PCB must be assembled according to the schematic and layout presented in Section 13. Once all components have been added to the circuit board, the board should be programmed. The most up-to-date Sensor Unit source code output file should be downloaded to the microcontroller using the MSP-FET430UIF connection and the FET –Pro430 utility.

10.2.2 Sensor Unit PCB – Capsule Attachment

The printed circuit board will be attached to the inner face of the PVC cap. It is attached such that the length of the board will be perpendicular to the inner face of the PVC cap. To do this a slot 0.0634 inches wide by 1.5 inches long by .1 inches deep must be etched. This slot should be centered. Since the PVC inner cap face is slightly concave, the depth should be considered starting at the surface point furthest from the center of the cap. Once this slot has been machined, the bottom 0.1 inch of the PCB should be placed in the slot and junction adhesive should be applied.

10.2.3 Sensor Unit Capsule - External Switch

The external reset switch enables the user to reset the Sensor Unit without opening the capsule. This switch allows the Sensor Unit to be tested immediately before it is installed and then reset. In addition, the reset switch enables the Sensor Unit to be reset in the advent it is mistakenly triggered during installation. The external switch must be durable and watertight.

Figure 83 illustrates the process of installing the reset switch in the Sensor Unit capsule.

- First, a hole must be machined in the PVC cap.
- The rubber boot is then placed on the hole and sealed on the outside of the PVC cap using the silicon sealant.
- The switch is then placed in the machined hole and rubber boot. The switch and hole are completely covered with silicon sealant. This includes the leads of the switch, which should be wired prior to this step.
- Lastly, a 1 inch diameter PVC nipple is placed around the switch and sealed using the silicon sealant. The threads and cap are sealed in the same manner as the main capsule. This described in detail in the next section.

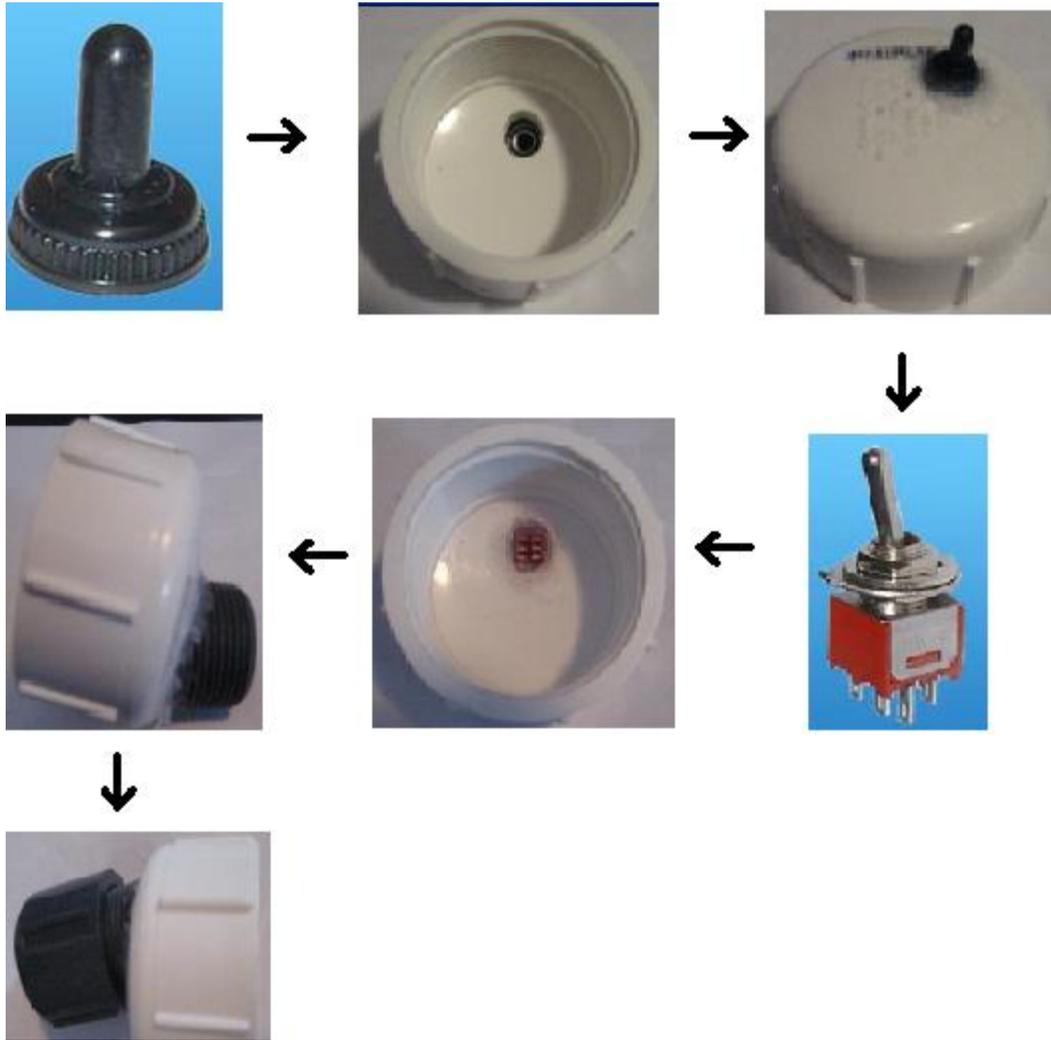


Figure 83: External Arm Switch Construction Steps

10.2.4 Sealing the Sensor Unit Capsule

The Sensor Unit capsule must be watertight. Both the PVC capsule pipe and caps are threaded so that they can be screwed together. However, this alone is not sufficient to keep the inside of the capsule completely dry. To further seal the capsule, Teflon tape must be wrapped around the threads on both ends of the PVC pipe prior to the caps being screwed on. Additionally, silicon sealant must be applied around the junction of the cap and the pipe once they have been screwed together. These steps, along with the Silica Gel Desiccants Packets being placed inside the capsule, will keep the inside of the capsule completely dry. Remember to maintain the verticality of the PCB while sealing the capsule. The cap connected to the PCB should remain the lowest point of the Sensor Unit with the length of the PCB, antenna, and PVC pipe all such that they form a 90-degree angle with level ground. The Sensor Unit should then be placed in a container of water large enough to submerge the Sensor Unit completely. Signs of any leakage are observed as bubbles being released from the Sensor Unit. If this is the case, the Sensor

Unit should be immediately removed from the water. The Sensor Unit should then be unsealed and inspected. If undamaged, the unit should be re-tested and re-sealed.

10.2.5 Sensor Unit Installation/Deployment

The steps to properly deploy the properly assembled and programmed Sensor Unit are described here.

1. Drill hole in the riverbed using hollow stem auger. Drill about 6 inches deeper than depth at which Sensor Unit should be buried.
2. Slowly add material to the hollow stem while slowly retracting auger about 6 inches. This material should fill in the bottom of the hole and provide a base for the Sensor Unit. Compacting this base might help to fill the hole.
3. Ensure the Sensor Unit is armed. The external switch push button should be locked down. It is important to keep the Sensor Unit in a vertical position to prevent it from being triggered.
4. Sensor Unit can be tested (by tilting it) to verify functionality and can be reset using the external arm switch. A test Receiver Unit is required to determine if the Sensor Unit is functioning (transmitting). Any Receiver Unit on the bridge must be reset because it will detect the Sensor Unit being tested.
5. Place Sensor Unit in hollow stem auger and push the Sensor Unit down to the bottom using a push rod. The push rod may need to be attached to the Sensor Unit to keep it in a vertical orientation during the filling stage.
6. Slowly fill the auger with material while slowly retracting the auger until the entire hole is filled.

11.0 Conclusions

The development of a remote sensing system to monitor Bridge Scour has been completed. Given the initial concept received for system operation, all other aspects of the system were worked out. Support of the design decisions was provided with calculation and simulation where necessary. COTS parts were integrated to realize the system. These COTS parts were chosen with cost as a factor, but future steps can be taken to reduce further the cost of the system. Additionally, if improved technology becomes available, such as batteries with a longer shelf life, this technology can be easily incorporated into the design. Keeping these improvements in mind, the system has been fully designed to perform the necessary operations.

A proof of concept for the remote sensing system was assembled and tested. The system is able to transmit, wirelessly, a message from a Sensor Unit to a Receiver Unit with the correct response displayed through a LED on the Light Indicator. The Sensor Unit can be buried under material within a water environment where it can remain dormant for several years. Upon release from this material, the Sensor Unit has shown the ability to float to the water surface and activate its internal circuitry using orientation sensitive switches. Operation with the use of multiple Sensor Units has also proven successful. Collision avoidance for simultaneous transmission by multiple Sensor Units is working correctly. The system displays the correct LED being lit on the Light Indicator given the highest priority Sensor Unit being active in this case. This encompasses the functionality required of the system.

The functionality demonstrated by this system should allow for the monitoring of Bridge Scour for PennDOT in the future. However, testing at this point has been done in a lab environment. Future tests should be done in a real world environment for further verification of the system. Pending the completion of such testing, the system can be ready for production and use by PennDOT.

12.0 Bill of Materials

This appendix contains the Bill of Materials for each of three main units of the system. All information regarding each component used for a given unit is displayed in a labeled table. The references made correspond to the schematics given in Section 13.0.

12.1 Sensor Unit Bill of Materials

This section contains the Bill of Materials for the Sensor Unit.

Item	Qty	Reference	Part	PCB Footprint	Manufacturer(Supplier Used)	Mfg Part Number	Unit Price
1	1	A1	Antenna 433 ¼ Whip	RP-SMA	Linx Technologies Inc (Digi-Key)	ANT-433-CW-RH	\$6.32
2	2	B1, B2	3 Volt Batteries	Button Top	Panasonic – BSG (Digi-Key)	CR-2	\$3.13
3	2	BCP1, BCP2	Negative Batteries Connect	SMT	Keystone (Digi-Key)	5201	\$0.228
4	2	BCP1, BCP2	Positive Batteries Connect	SMT	Keystone (Digi-Key)	5223	\$.137
5	1	C1	Cap 0.1 µF	0603	Panasonic – ECG (Digi-Key)	ECJ-1VB1C104K	\$.045
6	6	C2, C8, C9, C10, C11, C12	Cap 10 µF	0603	Panasonic – ECG (Digi-Key)	ECJ-1VB0J106M	\$.045
7	2	C3, C5	Cap 11 pF	0603	Murata Electronics North America (Digi-Key)	GRM1885C1H110JA01D	\$0.072
8	2	C4, C6	Cap 20 pF	0603	Murata Electronics North America (Digi-Key)	GRM1885C1H200JA01D	\$0.048
9	1	C7	Cap 0.01 µF	0603	Panasonic – ECG (Digi-Key)	ECJ-1VB1C103K	\$0.17
10	2	C13	Cap 0.47 µF	3528-21 (EIA)	Panasonic - ECG (Digi-Key)	ECS-H1VX474R	\$0.31
11	1	C14	Cap 33 µF	7343-31 (EIA)	Nichicon (Digi-Key)	F971C336MNC	\$0.81
12	1	C15	Cap 0.047 µF	0603	Panasonic – ECG (Digi-Key)	ECJ-1VB1E473K	\$0.18
13	2	C16, C17	Cap 1.0 µF	0603	Panasonic – ECG (Digi-Key)	ECJ-1VF1A105Z	\$0.10
14	1	J1	Header 14 P	.1” pitch	Waldom Electronics Corp (Digi-Key)	15-91-2140	\$1.42
15	1	J2	Header 6 P	0.1” pitch	Molex (Mouser)	70553-0005	\$1.24
15	1	J3	Conn Recept USB 5POS	USB mini B	Hirose Electric Co Ltd	UX60A-MB-5ST	\$1.18
15	1	R1	Res 47K Ohm	0603	Panasonic – ECG (Digi-Key)	ERJ-3GEYJ473V	\$.071
16	3	R2, R3, R4	Res 10 Ohm	0603	Panasonic – ECG (Digi-Key)	ERJ-3GEYJ100V	\$.071
17	1	RE1	Relay	through hole	Panasonic Electric Works (Digi-Key)	DS1E-SL-DC3V	\$7.84
18	1	SMA1	Connector SMA	Through hole	Linx Technologies Inc (Digi-Key)	CONREVSMA002	\$3.31

Item	Qty	Reference	Part	PCB Footprint	Manufacturer(Supplier Used)	Mfg Part Number	Unit Price
19	3	SW1, SW3, SW4	Tilt SW	3 position	Comus Group of Companies (Comus Group of Companies)	S1234	\$5.00
20	1	SW2	DPDT Slide SW	6 position	NKK Switches of America Inc (Digi-Key)	AS23AP	\$5.29
21	1	SW5	DPDT Toggle SW	3 position	MPJA Inc (MPJA Online)	12215-SW	\$0.72
22	1	U1	IC Microcontroller	28-TSSOP	Texas Instruments (Digi-Key)	MSP430F2132TPWR	\$4.05
23	1	U2	IC Transmitter	8-SMD	Linx Technologies Inc (Digi-Key)	TXM-433-LR	\$7.46
24	1	U3	IC USB to Serial UART	28-SSOP	FTDI, Future Technology Devices International Ltd	FT232RL	\$4.50
25	1	VR1	IC REG LDO	6-LLP	National Semiconductor (Digi-Key)	LP5900SD-2.2/NOPB	\$1.62
26	1	Y1	8 MHz Crystal	HC49/US	Abrakon Corporation (Digi-Key)	ABLS-8.000MHZ-B2-T	\$0.56
27	2	N/A	PVC Cap Threaded	2" diameter	A to Z supply (A to Z supply)	2PTCA	\$2.49
28	1	N/A	PVC Pipe Threaded	2"diameter X 6"	A to Z supply (A to Z supply)	26PN	\$2.69
29	1	N/A	PVC Cap	3/4" diameter	Mcmaster-Carr (Mcmaster-Carr)	4596K43	\$1.68
30	1	N/A	PVC Pipe Threaded	3/4" diameter	Mcmaster-Carr (Mcmaster-Carr)	9173K34	\$2.76
31	1	N/A	Rubber boot		MPJA Inc (MPJA Online)	16311-SW	\$1.95
						Total Price:	\$85.569

12.2 Receiver Unit Bill of Materials

This section contains the Bill of Materials for the Receiver Unit.

Item	Qty	Reference	Part	PCB Footprint	Manufacturer(Supplier Used)	Mfg Part Number	Unit Price
1	1	A1	Antenna 433 ¼ Whip	RP-SMA	Linx Technologies Inc (Digi-Key)	ANT-433-CW-QW	\$7.70
2	2	B1, B2	3 Volt Batteries	Button Top	Panasonic – BSG (Digi-Key)	CR-2	\$3.13
3	2	BCP1, BCP2	Negative Batteries Connect	SMT	Keystone (Digi-Key)	5201	\$0.228
4	2	BCP1, BCP2	Positive Batteries Connect	SMT	Keystone (Digi-Key)	5223	\$.137
5	2	C1, C14	Cap 0.1 µF	0603	Panasonic – ECG (Digi-Key)	ECJ-1VB1C104K	\$.045
6	5	C2, C8, C9, C10, C11	Cap 10µF	0603	Panasonic – ECG (Digi-Key)	ECJ-1VB0J106M	\$0.45
7	2	C3, C5	Cap 11 pF	0603	Murata Electronics North America (Digi-Key)	GRM1885C1H110JA01D	\$0.072
8	2	C4, C6	Cap 20 pF	0603	Murata Electronics North America (Digi-Key)	GRM1885C1H200JA01D	\$0.048
9	1	C7	Cap 0.01 µF	0603	Panasonic – ECG (Digi-Key)	ECJ-1VB1C103K	\$0.17
10	4	C12, C13, C16, C17	Cap 1 µF	0603	Panasonic – ECG (Digi-Key)	ECJ-1VF1C105Z	\$0.25
11	1	C18	Cap .47 µF	3528-21 (EIA)	Panasonic - ECG (Digi-Key)	ECS-H1VX474R	\$0.31
12	1	C14	Cap 33 µF	7343-31 (EIA)	Nichicon (Digi-Key)	F971C336MNC	\$0.81
13	1	J1	Header 14 P	0.1” pitch	Waldom Electronics Corp (Digi-Key)	15-91-2140	\$1.42
14	1	J2	Header 6 P	0.1” pitch	Molex (Mouser)	70553-0005	\$1.24
15	1	J3	Conn Recept USB 5POS	USB mini B	Hirose Electric Co Ltd (Digi-Key)	UX60A-MB-5ST	\$1.18
16	1	PJ1	Power Jack	Through hole	CUI Inc	PJ-202A	\$ 0.38
17	1	R1	Res 47K Ohm	0603	Panasonic – ECG (Digi-Key)	ERJ-3GEYJ473V	\$.071
18	1	SMA1	Connector SMA	Through hole	Linx Technologies Inc (Digi-Key)	CONREVSMA001	\$2.92
19	1	SW1	SPDT	3 position	Tyco Electronics Alcoswitch (Digi-Key)	MHS12304	\$1.77
21	1	U1	IC Microcontroller	28-TSSOP	Texas Instruments (Digi-Key)	MSP430F2132TPWR	\$4.05
22	1	U2	IC Receiver	16-SMD	Linx Technologies Inc (Digi-Key)	RXM-433-LR	\$13.56

Item	Qty	Reference	Part	PCB Footprint	Manufacturer(Supplier Used)	Mfg Part Number	Unit Price
23	1	U3	IC USB to Serial UART	28-SSOP	FTDI, Future Technology Devices International Ltd (Digi-Key)	FT232RL	\$4.50
24	1	VR1	IC REG LDO	SOT23-5	Texas Instruments (Digi-Key)	TPS78930DBVR	\$0.90
25	1	Y1	8 MHz Crystal	2 Pad	ABRACON CORPORATION (Digi-Key)	ABLS-8.000MHZ-B2-T	\$0.56
26	1	NA	Case	Wall Mount	Pactec (Patec)	OD36-2.0	\$11.75
27	1	NA	Power Adapter	NA	Tamura (Digi-Key)	212AS09012	\$5.13
28	1	NA	Header Connector	pitch	Molex (Mouser)	50-57-9406	\$0.63
						Total Price:	\$72.09

12.3 Light Indicator Bill of Materials

This section contains the Bill of Materials for the Light Indicator.

Item	Qty	Reference	Part	PCB Footprint	Manufacturer (Supplier)	Mfg Part Number	Unit Price
1	4	J1, J2, J3, J4	Header 6 P	0.1" pitch	Molex (Mouser)	70553-0005	\$1.24
2	2	U1, U2	OR-4	14-SOP	Texas Instruments (Digi-Key)	CD4072BNSR	\$0.50
3	1	L1	Green LED	Through Hole	Lite-On-Inc (Digi-Key)	LTL2P3KGKNN	\$0.15
4	1	L2	Yellow LED	Through Hole	Lite-On-Inc (Digi-Key)	LTL-2H3VSKNT	\$0.32
5	1	L3	Orange LED	Through Hole	LUMEX OPTO/COMPONENTS INC (Digi-Key)	SSL-LX5093SOC	\$0.99
6	1	L4	Red LED	Through Hole	Lite-On-Inc (Digi-Key)	LTL2P3SEK	\$0.33
7	4	R1, R2, R3, R4	Res – 51 ohm	0603	Panasonic - ECG	ERJ-3GEYJ510V	\$0.07
8	4	NA	Header Connector	0.1 pitch	Molex (Mouser)	50-57-9406	\$0.63
						Total Price:	\$10.55

12.4 Printed Circuit Boards

A different printed circuit board (PCB) was designed for the Sensor Unit, Receiver Unit, and Light Indicator. Each of these PCBs was designed using ExpressPCB software. The boards could be ordered through ExpressPCB using the design files created with their software. Below are the properties of each of the PCBs and the ordering used.

Sensor Unit PCB:

- Dimensions of 1.5” width by 4.0” length
- Two layer board
- Standard order service

Receiver Unit PCB:

- Dimensions of 2.2” width by 4.5” length
- Two layer board
- Standard order service

Light Indicator PCB:

- Dimensions of 2.5” width by 3.8” length
- Two layer board
- Mini board service

13.0 Hardware Design files

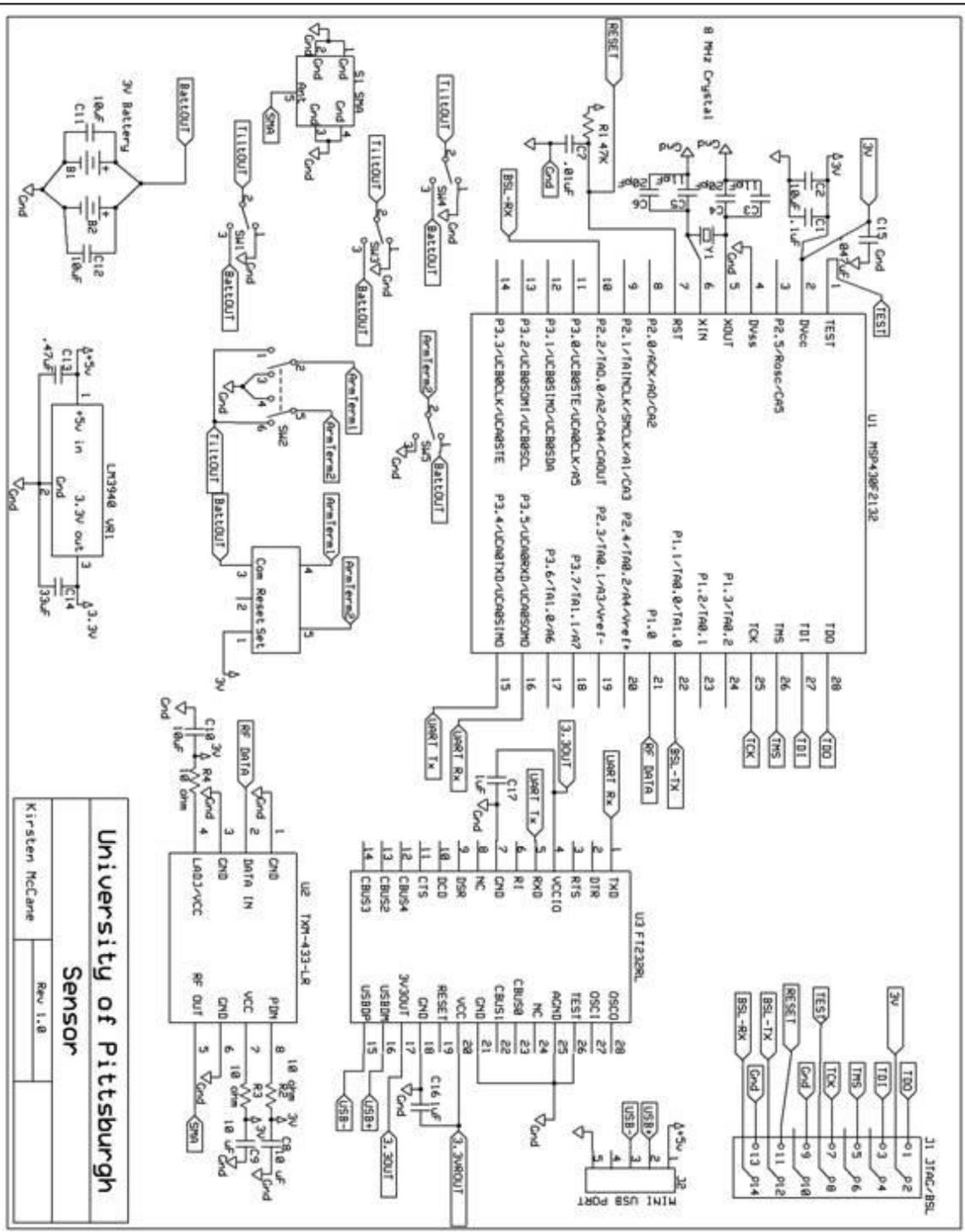
This appendix contains the schematics created for the Sensor Unit, Receiver Unit, and Light Indicator within ExpressSCH software. It also contains the PCB design files created within ExpressPCB software for the three components.

13.1 Schematics

This section contains images of the schematics for the Sensor Unit, Receiver Unit, and Light Indicator. The schematics can be viewed using the ExpressSCH software and the schematic files accompanying this document.

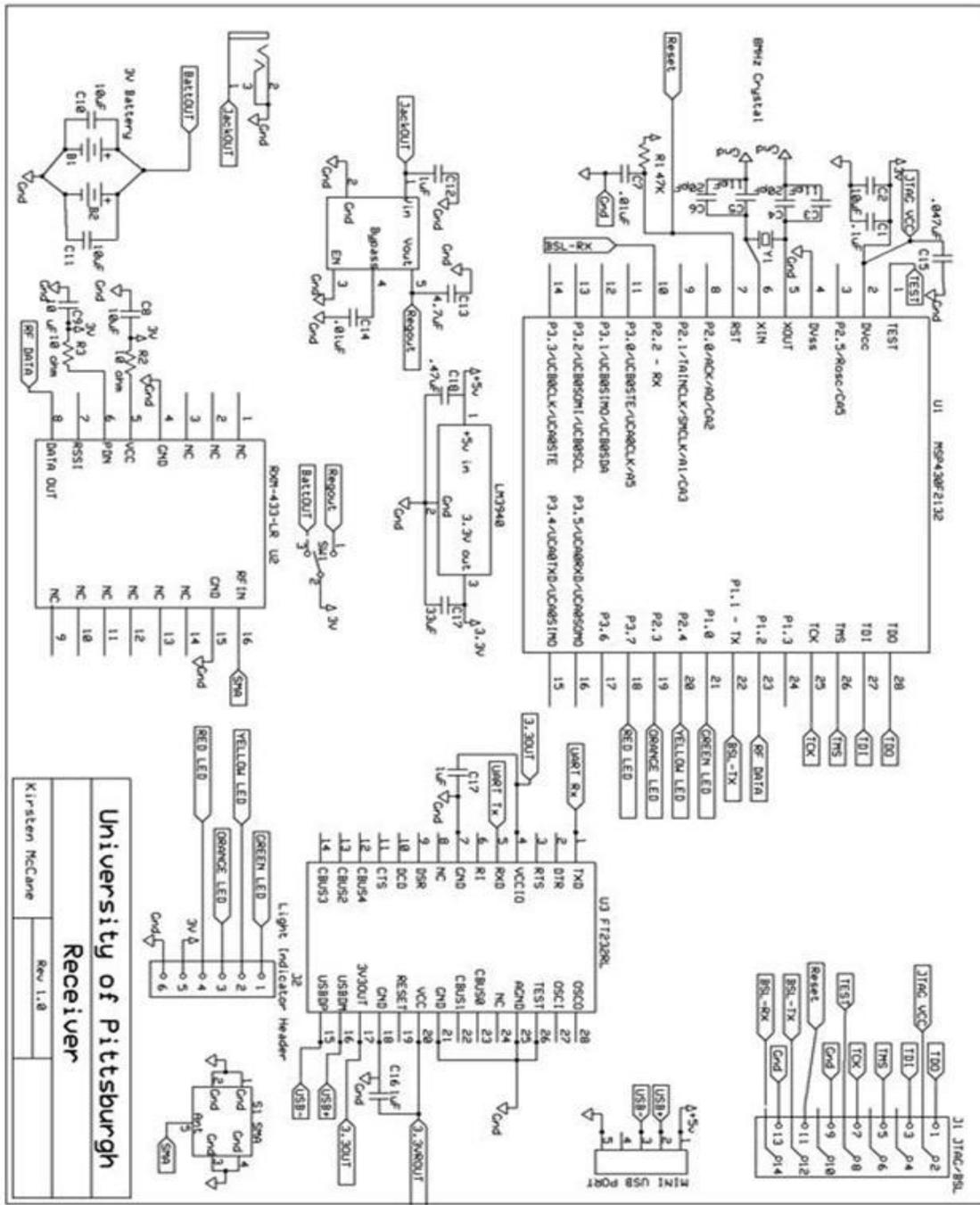
13.1.1 Sensor Unit Schematic

The schematic for the Sensor Unit is shown below.



13.1.2 Receiver Unit Schematic

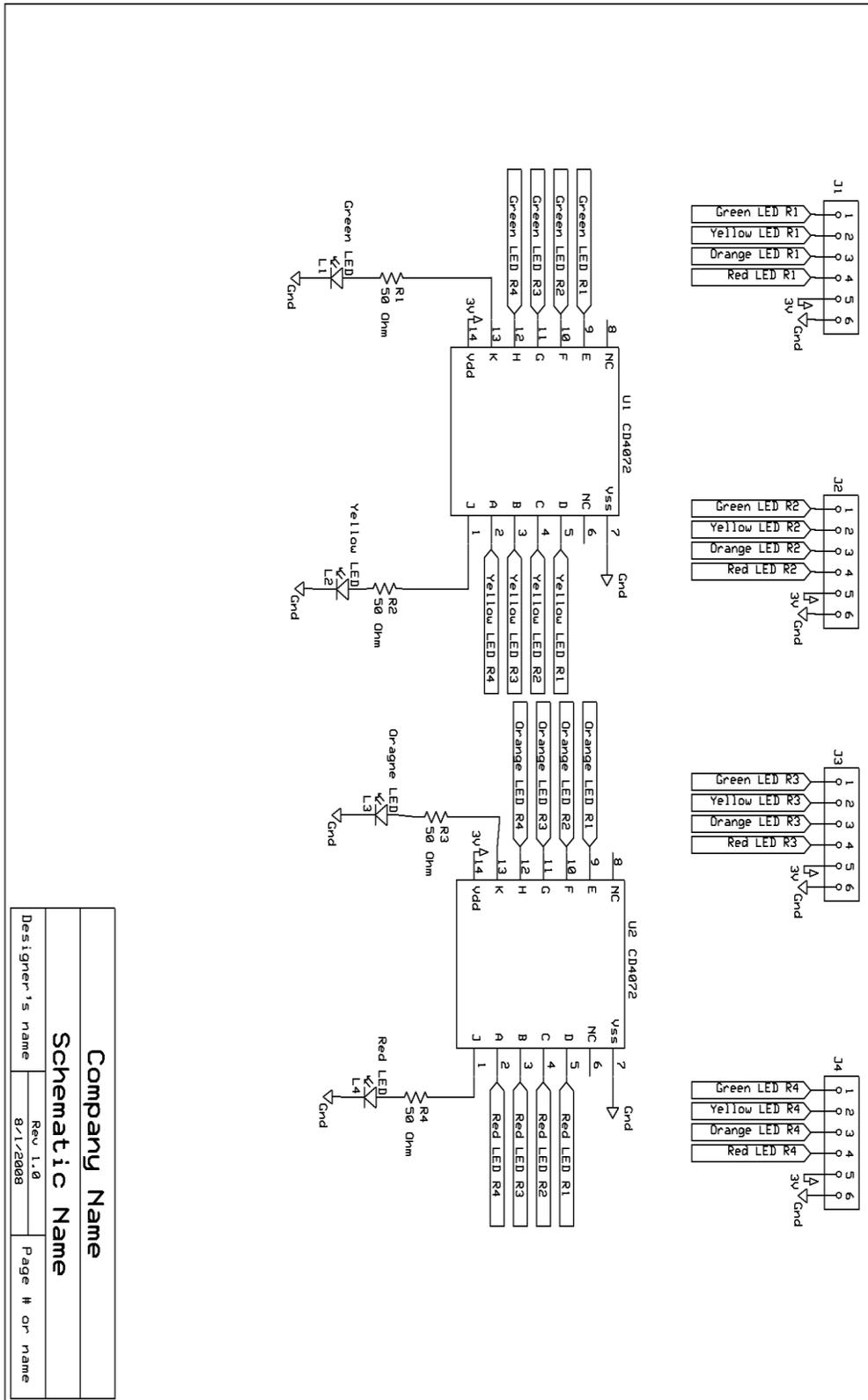
The schematic for the Receiver Unit is shown below.



University of Pittsburgh
Receiver
Kiristeen McNamee
Rev 1.0

13.1.3 Light Indicator Schematic

The schematic for the Light Indicator is shown below.

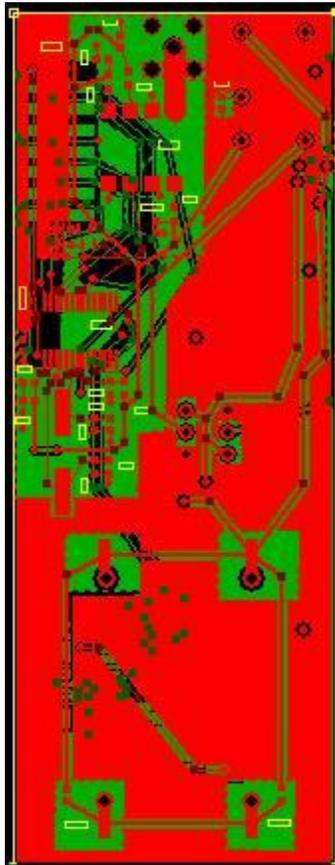


13.2 PCB Layout Designs

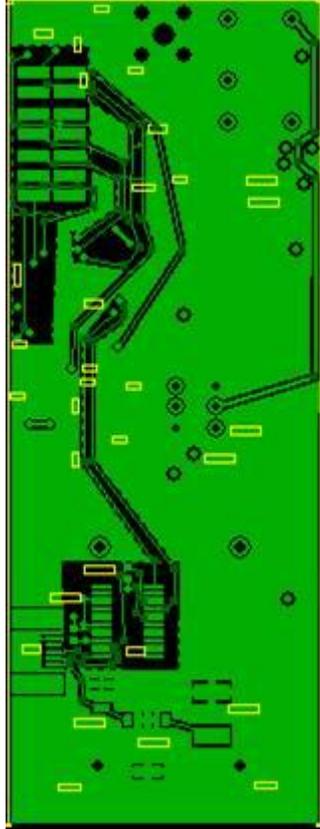
This section contains images of the PCB layout files for the Sensor Unit, Receiver Unit, and Light Indicator. For each component three images are shown; (1) layout showing both top and bottom layers; (2) layout showing only bottom layer; and (3) layout showing only top layer. The layout files can be opened using the ExpressPCB software and the associated layout file included with this package.

13.2.1 Sensor Unit PCB Design

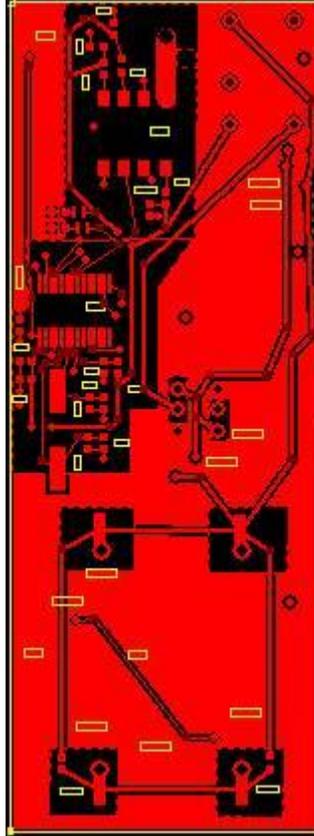
The following figure shows the layout of the Sensor Unit PCB with the top and bottom layers shown.



The following figure shows the layout of the Sensor Unit PCB with only the bottom layer shown.

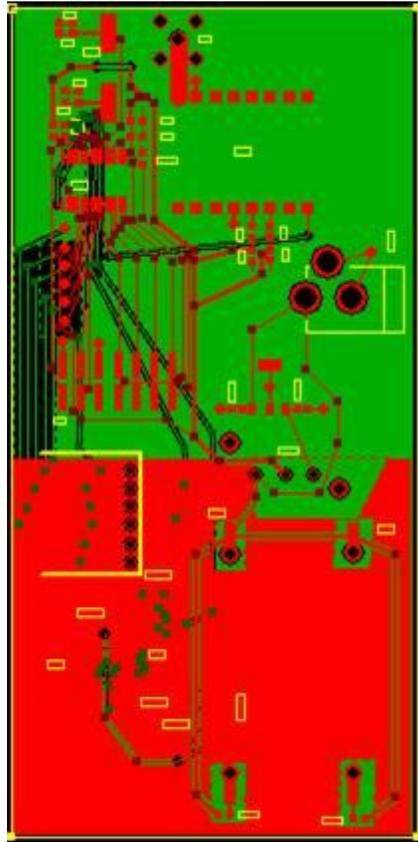


The following figure shows the layout of the Sensor Unit PCB with only the top layer shown.

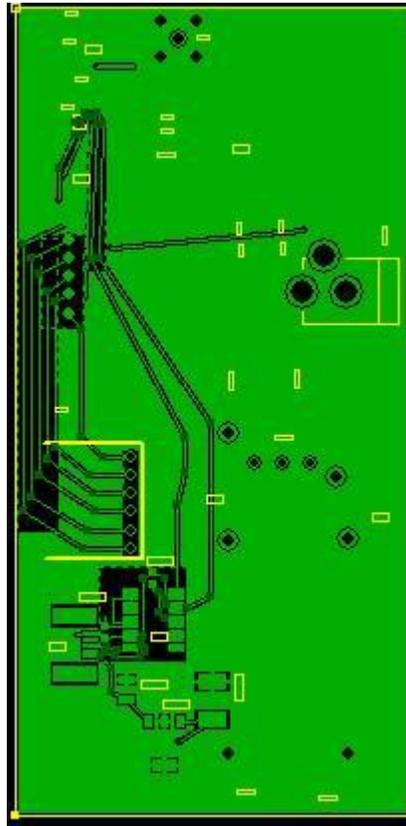


13.2.2 Receiver Unit PCB Design

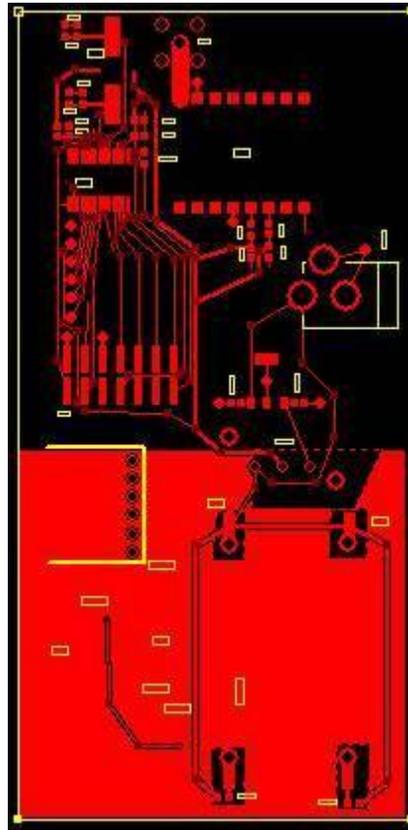
The following figure shows the layout of the Receiver Unit PCB with the top and bottom layers shown.



The following figure shows the layout of the Receiver Unit PCB with only the bottom layer shown.

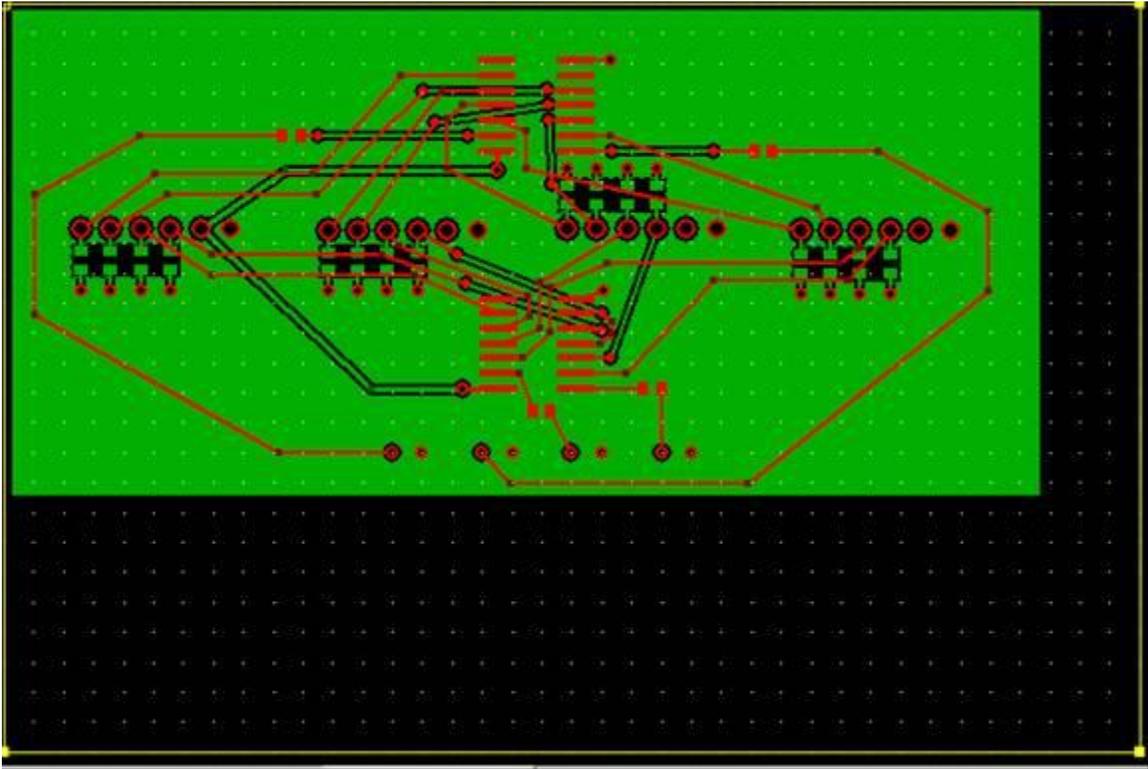


The following figure shows the layout of the Receiver Unit PCB with only the top layer shown.

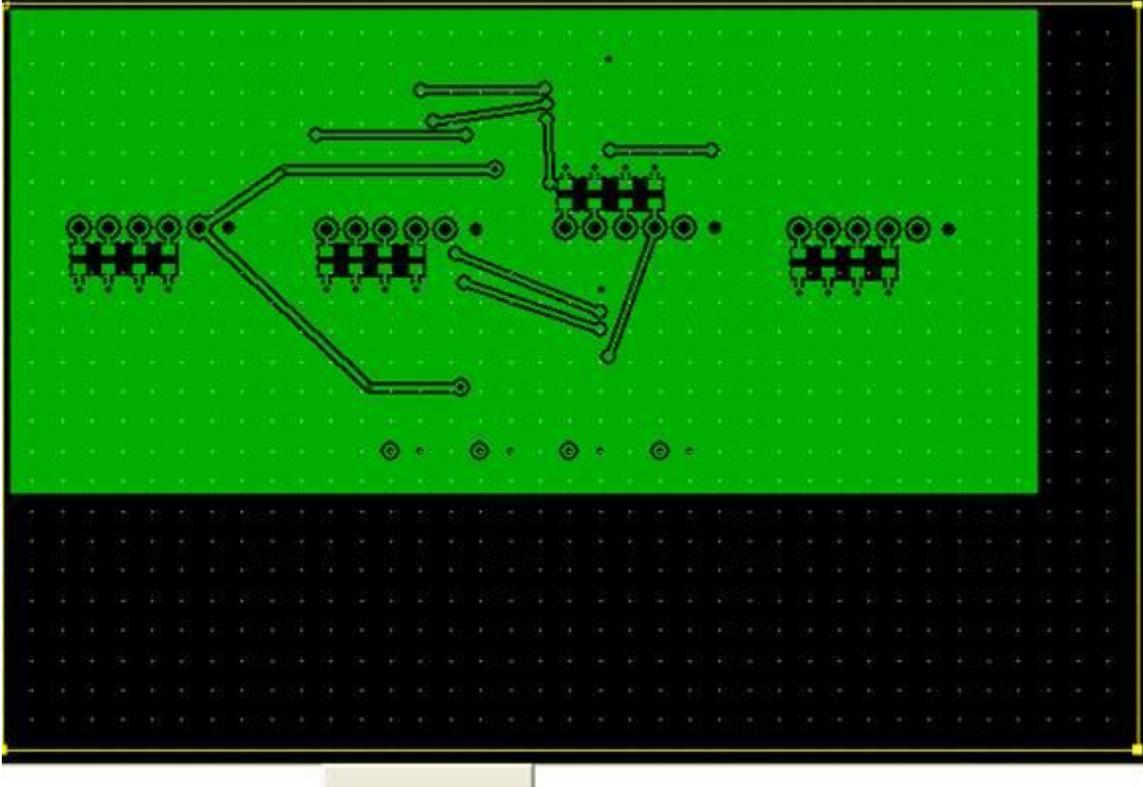


13.2.3 Light Indicator PCB Design

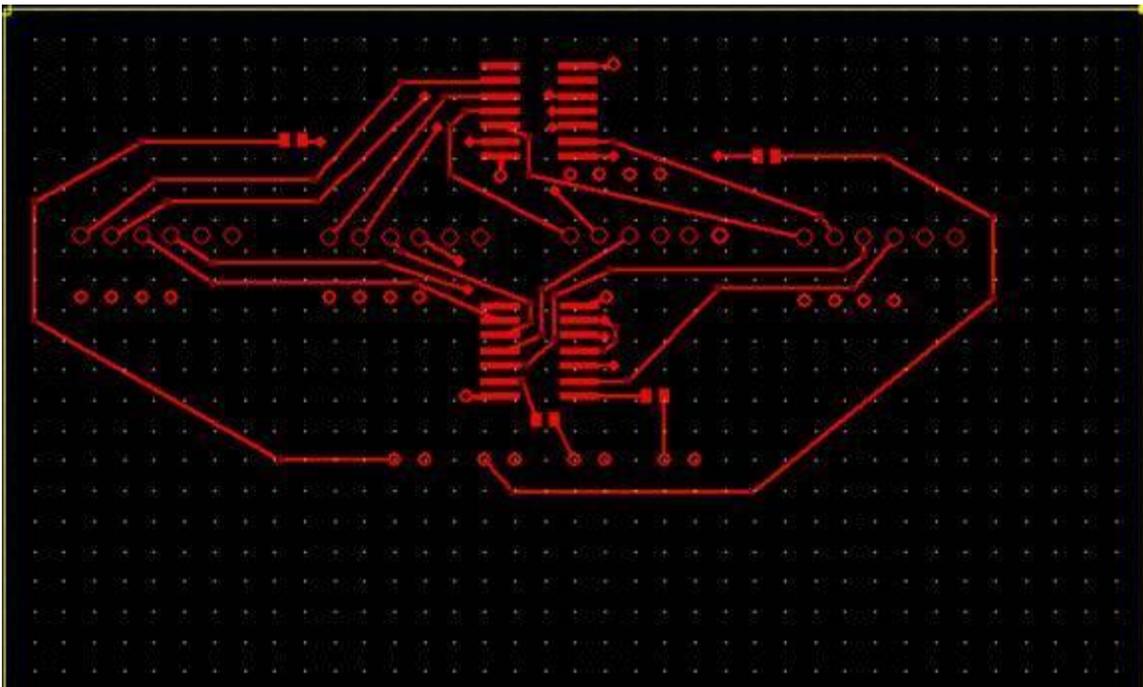
The following figure shows the layout of the Light Indicator PCB with the top and bottom layers shown.



The following figure shows the layout of the Light Indicator PCB with only the bottom layer shown.



The following figure shows the layout of the Light Indicator PCB with only the top layer shown.



14.0 Source Code

This section contains the source code for the embedded software for the Sensor Unit and Receiver Unit.

14.1 Sensor Unit Source Code

```
#include "msp430x21x2.h"
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>

#define POLYNOMIAL 0x88108000 /* CCITT polynomial (x^16 + x^12 + x^5 + 1) followed
by 0's */
#define POLY 0x8408
#define Red_Priority 4
#define Orange_Priority 8
#define Yellow_Priority 12
#define Green_Priority 16

//GLOBAL VARIABLES
unsigned long dataToOutput_bit[17];
unsigned long bitIndex = 0x00;
int arrayIndex = 0;
int UART_rx_index = 0;
int ch_index = 0;
int not_serial = 1;
int UART_mode = 0;
int priority;
int mask;
int i;
int j;
int mainflag;
int ISR_flag;
int bit_next;
int bit_beginning;
int bit_mid;
int bit_end_level;
int delay;
uint16_t crc;
char *ptr;
char data;
char current_data_char = 0x00;
char last_rx_char = 'a';
char *Flash_ptr = (char *)0x1080; // Initialize Flash pointer
char my_array[21];
char ch_buffer[21];
char my_byte[2];

//FUNCTION PROTOTYPES
void configTxTimer();
void configDataOutputLine();
void config_Clocks();
void config_UART();
void print_SegBtoUART();
void print_byte_to_UART(char x);
void write_SegB(char *data_p, int _crc,int length);
```

```

        void read_SegB_data_byte();
        uint16_t crc16(char *data_p,unsigned short length);
//END (FUNCTION PROTOTYPES)

//////////////////////////////////////////////////////////////////START MAIN//////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////

void main(void)
{
    uint32_t k,l, loop_limit;
    int delay_loop_choice, transmit_loop;

//DUMMY DATABLOCK
char bridge_ID_Byte_1 = 0xAA;
char bridge_ID_Byte_2 = 0xAA;
char bridge_ID_Byte_3 = 0xAA;
char bridge_ID_Byte_4 = 0xAA;
char bridge_ID_Byte_5 = 0xAA;
char bridge_ID_Byte_6 = 0xAA;
char bridge_ID_Byte_7 = 0xAA;
char bridge_ID_Byte_8 = 0xAA;
char bridge_ID_Byte_9 = 0xAA;
char bridge_ID_Byte_10 = 0xAA;
char bridge_ID_Byte_11 = 0xAA;
char bridge_ID_Byte_12 = 0xAA;
char bridge_ID_Byte_13 = 0xAA;
char bridge_ID_Byte_14 = 0xAA;
char serial_Num1 = 0xAA;
char serial_Num2 = 0xAA;
char location_1 = 0xAA;
char location_2 = 0xAA;
char location_3 = 0xAA;
char location_4 = 0xAA;
char location_5 = 0xAA;

my_array[0] = bridge_ID_Byte_1;
my_array[1] = bridge_ID_Byte_2;
my_array[2] = bridge_ID_Byte_3 ;
my_array[3] = bridge_ID_Byte_4;
my_array[4] = bridge_ID_Byte_5 ;
my_array[5] = bridge_ID_Byte_6 ;
my_array[6] = bridge_ID_Byte_7;
my_array[7] = bridge_ID_Byte_8 ;
my_array[8] = bridge_ID_Byte_9 ;
my_array[9] = bridge_ID_Byte_10;
my_array[10] = bridge_ID_Byte_11 ;
my_array[11] = bridge_ID_Byte_12;
my_array[12] = bridge_ID_Byte_13;
my_array[13] = bridge_ID_Byte_14;
my_array[14] = serial_Num1 ;
my_array[15] = serial_Num2;
my_array[16] = location_1 ;
my_array[17] = location_2 ;
my_array[18] = location_3 ;
my_array[19] = location_4 ;
my_array[20] = location_5;
ptr = &my_array[0];

    mask = 128; //used to determine bit within byte
    bit_beginning = 1; //signifies beginning of new bit period
    bit_mid = 0; //signifies middle of bit period
    bit_end_level = 1; //level of bit at the end of last bit period
    delay = 0; //control signal for synchronizing bit
    transmit_loop= 0; //ensures two message transmissions take place before delays
are applied

    WDCTL = WDTWP + WDT HOLD; //disables watchdog timer

```

```

//Configure uP registers
config_Clocks();
configTxTimer();
configDataOutputLine();

config_UART();

_BIS_SR(GIE);

//calculates crc from data
crc = crc16(ptr, 21);

//write dummy block to flash memory location
write_SegB(ptr,crc, 23);

//read first byte to transmit
read_SegB_data_byte();

//intialize variables

//initialize priority based on color code
switch(location_5){
case 0:
priority = Green_Priority;
break;
case 1:
priority = Yellow_Priority;
break;
case 2:
priority = Orange_Priority;
break;
case 3:
priority = Red_Priority;
break;
default:
priority = Green_Priority;
break;
}

//generate random seed based on serial number
srand((unsigned int) (serial_Num1 + serial_Num2));

//run continuously if
while(not_serial)
{

//only enter if ISR has been entered
if(ISR_flag == 1)
{

ISR_flag = 0;

//transition to beginning or mid bit processig setting
if(bit_beginning == 1)
{
bit_beginning = 0;
bit_mid = 1;
mask = mask / 2 ; //change masking bit
j++;
}
else
{
bit_beginning = 1;
bit_mid = 0;
}

//if byte has been processed
if(j == 8)
{
j=0;
}
}
}

```

```

mask = 128;

//if entire message has been sent - reset
if (Flash_ptr > (char *)0x1097)
{
    Flash_ptr = (char *)0x107E;
    bit_beginning = 0;
    bit_mid = 0;
    TAOCCCTL0 &= !CCIE;

    //add in delay after message has been sent twice
    if(transmit_loop %2 == 1)
    {

delay_loop_choice = rand() % priority;

if(delay_loop_choice == 0 || delay_loop_choice == 1 || delay_loop_choice == 4 ||
delay_loop_choice == 8)
{
    loop_limit = 50000;
    for(k =
0; k <loop_limit; k++)
        P1OUT
&= 0x00;    // set P1.0 to 0
    }
else
if(delay_loop_choice == 2 || delay_loop_choice == 5 || delay_loop_choice == 6 ||
delay_loop_choice == 12 )
{
    loop_limit = 117000;
    for(k =
0; k <loop_limit; k++)
        P1OUT
&= 0x00;    // set P1.0 to 0
    }
else
if(delay_loop_choice == 3 || delay_loop_choice == 13 || delay_loop_choice == 9 ||
delay_loop_choice == 10 )
{
    loop_limit = 275000;
    for(k =
0; k <loop_limit; k++)
        P1OUT
&= 0x00;    // set P1.0 to 0
    }
else
if(delay_loop_choice == 14 || delay_loop_choice == 15 || delay_loop_choice == 11 ||
delay_loop_choice == 7)
{
    loop_limit = 550000;
    for(k =
0; k <loop_limit; k++)
        P1OUT
&= 0x00;    // set P1.0 to 0
    }
//for(k
= 0; k <1000; k++)
//
P1OUT |= 0x01;    // set P1.0 to 0
    }
    //send sync pulse
    TAOCCR0 = 18000;
    TAOCCCTL0 = CCIE;
    P1OUT |= 0x01;    // set P1.0 to 1
    delay = 1;
    transmit_loop++;
}
}

```

```

        read_SegB_data_byte();
    }
    mainflag = 1;
}

}

}

////////////////////////////////////////////////////////////////START TIMER 0_A0////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

void config_Clocks()
{
    unsigned long i;
    if (CALBC1_8MHZ ==0xFF || CALDCO_8MHZ == 0xFF)
    {
        while(1); // If calibration constants erased
                // do not load, trap CPU!!
    }
    BCSCCTL1 = CALBC1_8MHZ; // Set DCO to 1\16MHz
    DCOCTL = CALDCO_8MHZ;
    FCTL2 = FWKEY + FSSEL0 + FN5 + FN3 + FN2 + FN1; // MCLK/24 for Flash Timing

    BCSCCTL1 |= XTS; // ACLK = LFXT1 = HF XTAL
    BCSCCTL2 |= SELM_3 + SELS; // MCLK = LFXT1 (safe)
    BCSCCTL3 |= LFXT1S1 + XT2S1; // 3 - 16MHz crystal or
resonator
do
{
    IFG1 &= ~OFIFG; // Clear OSCFault flag
    for (i = 0xFF; i > 0; i--); // Time for flag to set
} while (IFG1 & OFIFG); // OSCFault flag still set?
}

void configTxTimer()
{
    TAOCTL = TACLK;
    TAOCERO = 800;
    TAOCTL = TASSEL_1 + MC_1;
    TAOCCTL0 = CCIE;
}

void configDataOutputLine()
{
    P1SEL &= !BIT0;
    P1DIR |= 0x01;
    P1OUT &= !0x01;
}

void config_UART()
{
    P3SEL = 0x30; // P3.4,5 = USCI_A0 TXD/RXD
    UCA0CTL1 |= UCSSEL_3; // SMCLK
    UCA0BR0 = 0xA0; // 8MHz 19200
    UCA0BR1 = 0x01; // 8MHz 19200
    UCA0MCTL = UCBSR0; // Modulation UCBSRx = 1
    UCA0CTL1 &= ~UCSWRST; // **Initialize USCI state machine**
    IE2 |= UCA0RXIE; // Enable USCI_A0 RX interrupt
}

//Sends out data block based on manchester encoding shown in figure 39 of final report
//done so if analyzed in time a high to low transition would be a 1
//while a low to high mid bit transition is a 0
//with bit period being 1600 clock cycles
#pragma vector=TIMER0_A0_VECTOR
__interrupt void Timer0_A0(void)
{
    ISR_flag = 1;
}

```

```

    if ( bit_beginning == 1)
    {
        if (mask & data)          //determine bit value
            bit_next = 1;
        else
            bit_next = 0;
        if (bit_next == 1)
        {
            if(bit_end_level == 0)
                P1OUT |= 0x01;      // set P1.0 to 1
        }
        else
        {
            if(bit_end_level == 1)
                P1OUT &= ~0x01;     // set P1.0 to 0
        }
    }
    else if( bit_mid == 1)
    {
        if (bit_next == 1)
        {
            P1OUT &= ~0x01;        // set P1.0 to 0
            bit_end_level = 0;
        }
        else if ( bit_next == 0)
        {
            P1OUT |= 0x01;         // set P1.0 to 1
            bit_end_level = 1;
        }
    }
}
else if(delay == 1)
{
    TAOCCR0 = 800;
    delay = 0;
    bit_beginning = 1;
}

    _NOP();
mainflag = 0;
} // end timerA0 interrupt

#pragma vector=USCIAB0RX_VECTOR
__interrupt void USCI0RX_ISR(void)
{
    int i;
    char *ptr2;
    char tx_finish[] = {'d','a','t','a','_','t','x','e','d','!'};
    char readout[] = {'d','a','t','a','_','b','l','o','c','k',':'};

    // while (!(IFG2 & UCA0TXIFG));          // USCI_A0 TX buffer ready?
    //UCA0TXBUF = UCA0RXBUF;                // TX -> RXed character

    if(UCA0RXBUF == '#')
    {
        //print byte to UART(0xAB);
        for(i =0;i<11;i++)
        {
            while (!(IFG2 & UCA0TXIFG));          //
            USCI_A0 TX buffer ready?
            while ( (UCA0STAT & UCBUSY ));
            UCA0TXBUF = readout[i];
        }
        print_SegBtoUART();
        while (!(IFG2 & UCA0TXIFG));          //
        USCI_A0 TX buffer ready?
        while ( (UCA0STAT & UCBUSY ));
        UCA0TXBUF = 10;
    }
}

```

```

USCI_A0 TX buffer ready?
}
if(UART_mode == 1)
{
    while (!(IFG2 & UCA0TXIFG)); //
while ( (UCA0STAT & UCBUSY ));
UCA0TXBUF = 13;

if(ch_index % 2 == 0)
{
    switch(UCA0RXBUF) {
        case '0':
            current_data_char = 0x00;
            break;
        case '1':
            current_data_char = 0x10;
            break;
        case '2':
            current_data_char = 0x20;
            break;
        case '3':
            current_data_char = 0x30;
            break;
        case '4':
            current_data_char = 0x40;
            break;
        case '5':
            current_data_char = 0x50;
            break;
        case '6':
            current_data_char = 0x60;
            break;
        case '7':
            current_data_char = 0x70;
            break;
        case '8':
            current_data_char = 0x80;
            break;
        case '9':
            current_data_char = 0x90;
            break;

        case 'A':
            current_data_char = 0xA0;
            break;
        case 'B':
            current_data_char = 0xB0;
            break;
        case 'C':
            current_data_char = 0xC0;
            break;
        case 'D':
            current_data_char = 0xD0;
            break;
        case 'E':
            current_data_char = 0xE0;
            break;
        case 'F':
            current_data_char = 0xF0;
            break;
        default:
            current_data_char = 0x00;
            break;
    }

}

if(ch_index % 2 == 1)
{
    switch(UCA0RXBUF) {

```

```

        case '0':
            current_data_char += 0x00;
            break;
        case '1':
            current_data_char += 0x01;
            break;
        case '2':
            current_data_char += 0x02;
            break;
        case '3':
            current_data_char += 0x03;
            break;
        case '4':
            current_data_char += 0x04;
            break;
        case '5':
            current_data_char += 0x05;
            break;
        case '6':
            current_data_char += 0x06;
            break;
        case '7':
            current_data_char += 0x07;
            break;
        case '8':
            current_data_char += 0x08;
            break;
        case '9':
            current_data_char += 0x09;
            break;

        case 'A':
            current_data_char += 0x0A;
            break;
        case 'B':
            current_data_char += 0x0B;
            break;
        case 'C':
            current_data_char += 0x0C;
            break;
        case 'D':
            current_data_char += 0x0D;
            break;
        case 'E':
            current_data_char += 0x0E;
            break;
        case 'F':
            current_data_char += 0x0F;
            break;
        default:
            current_data_char += 0x00;
            break;
    }

    ch_buffer[UART_rx_index] =

current_data_char;

    UART_rx_index++;
}

    ch_index++;
    if(UART_rx_index ==21)
    {
        //ptr2 = &ch_buffer[0];
        //calculates crc from data

        //write_SegB(ptr2,crc, 23);
        ptr2 = (char *) 0x1080;
        crc = crc16((char *) 0x1080, 21);
        FCTL3 = FWKEY; // Clear Lock bit
    }

```

```

        FCTL1 = FWKEY + ERASE;           // Set Erase bit
        *ptr2 = 0;

        FCTL1 = FWKEY + WRT;
        for(i = 0; i<21;i++)
            *ptr2++ = ch_buffer[i];

        *ptr2++ = (char) (crc >> 8);
        *ptr2++ = (char) crc;

        FCTL1 = FWKEY;                   // Clear WRT bit
        FCTL3 = FWKEY + LOCK;            // Set LOCK bit
        UART_mode = 0;
        UART_rx_index = 0;
        ch_index = 0;
        for( i=0; i<10;i++)
        {
            while (!(IFG2 & UCA0TXIFG)); //
USCI_A0 TX buffer ready?
            while ( (UCA0STAT & UCIBUSY ));
            UCA0TXBUF = tx_finish[i];
// TX character
        }
            while (!(IFG2 & UCA0TXIFG)); //
USCI_A0 TX buffer ready?
            while ( (UCA0STAT & UCIBUSY ));
            UCA0TXBUF = 10;
            while (!(IFG2 & UCA0TXIFG)); //
USCI_A0 TX buffer ready?
            while ( (UCA0STAT & UCIBUSY ));
            UCA0TXBUF = 13;

        TA0CCTL0 = CCIE;
        not_serial = 1;
    }
}

    if (UCA0RXBUF == '*' && last_rx_char == '*')
    {
        UART_mode = 1;
        TA0CCTL0 &= !CCIE;
        not_serial = 0;
    }
    last_rx_char = UCA0RXBUF;
}

void write_SegB(char *data_p, int _crc,int length)
{
    char *This_Flash_ptr; // Flash pointer
    uint8_t temp;
    unsigned int i;
    /*
    *place bytes in array
    */
    char data_byte[23];
    data_byte[0] = data_p[0];
    data_byte[1] = data_p[1];
    data_byte[2] = data_p[2];
    data_byte[3] = data_p[3];
    data_byte[4] = data_p[4];
    data_byte[5] = data_p[5];
    data_byte[6] = data_p[6];
    data_byte[7] = data_p[7];
    data_byte[8] = data_p[8];
    data_byte[9] = data_p[9];
    data_byte[10] = data_p[10];
    data_byte[11] = data_p[11];
    data_byte[12] = data_p[12];

```

```

    data_byte[13] = data_p[13];
    data_byte[14] = data_p[14];
    data_byte[15] = data_p[15];
    data_byte[16] = data_p[16];
    data_byte[17] = data_p[17];
    data_byte[18] = data_p[18];
    data_byte[19] = data_p[19];
    data_byte[20] = data_p[20];
    data_byte[21] = (char) (_crc >> 8);
    data_byte[22] = (char) _crc;

This_Flash_ptr = (char *)0x107E;           // Initialize Flash pointer
FCTL3 = FWKEY;                          // Clear Lock bit
FCTL1 = FWKEY + ERASE;                   // Set Erase bit
*This_Flash_ptr = 0;                      // Dummy write to erase Flash seg

FCTL1 = FWKEY + WRT;                     // Set WRT bit for write operation
    *This_Flash_ptr++ = 0xAA;
    *This_Flash_ptr++ = 0x55;
for (i = 0; i < 23; i++)
{
    *This_Flash_ptr++ = data_byte[i];     // Write value to flash
}

FCTL1 = FWKEY;                          // Clear WRT bit
FCTL3 = FWKEY + LOCK;                    // Set LOCK bit
_NOP();
}

void read_SegB_data_byte()
{
    data = *Flash_ptr;
    *Flash_ptr++;
}

void print_SegBtoUART()
{
    char *print_ptr;
    int i;

    print_ptr = (char *) 0x1080;

    for(i=0;i<23;i++)
    {
        print_byte_to_UART(*print_ptr++);
    }
}

void print_byte_to_UART(char x)
{
    int current_data_char;
    char second_nibble_left_4 = (x << 4);
    char second_nibble_iso = second_nibble_left_4 >> 4;
    char first_nibble_iso = x >> 4;
    switch( first_nibble_iso){
        case 0:
            current_data_char = 48;
            break;
        case 1:
            current_data_char = 49;
            break;
        case 2:
            current_data_char = 50;
            break;
        case 3:

```

```

        current_data_char = 51;
        break;
        case 4:
        current_data_char = 52;
        break;
        case 5:
        current_data_char = 53;
        break;
        case 6:
        current_data_char = 54;
        break;
        case 7:
        current_data_char = 55;
        break;
        case 8:
        current_data_char = 56;
        break;
        case 9:
        current_data_char = 57;
        break;

        case 10:
        current_data_char = 65;
        break;
        case 11:
        current_data_char = 66;
        break;
        case 12:
        current_data_char = 67;
        break;
        case 13:
        current_data_char = 68;
        break;
        case 14:
        current_data_char = 69;
        break;
        case 15:
        current_data_char = 70;
        break;
        default:
        current_data_char = 0x00;
        break;
    }

    while (!(IFG2 & UCA0TXIFG)); //

USCI_A0 TX buffer ready?
while ( (UCA0STAT & UCBUSY ) );
UCA0TXBUF = current_data_char;

switch(second_nibble_iso){
    case 0:
    current_data_char = 48;
    break;
    case 1:
    current_data_char = 49;
    break;
    case 2:
    current_data_char = 50;
    break;
    case 3:
    current_data_char = 51;
    break;
    case 4:
    current_data_char = 52;
    break;
    case 5:
    current_data_char = 53;
    break;
    case 6:
    current_data_char = 54;

```

```

        break;
        case 7:
            current_data_char = 55;
            break;
        case 8:
            current_data_char = 56;
            break;
        case 9:
            current_data_char = 57;
            break;

        case 10:
            current_data_char = 65;
            break;
        case 11:
            current_data_char = 66;
            break;
        case 12:
            current_data_char = 67;
            break;
        case 13:
            current_data_char = 68;
            break;
        case 14:
            current_data_char = 69;
            break;
        case 15:
            current_data_char = 70;
            break;
        default:
            current_data_char = 0x00;
            break;
    }

    while (!(IFG2 & UCA0TXIFG)); //

USCI_A0 TX buffer ready?

    while ( (UCA0STAT & UCBUSY ) );
    UCA0TXBUF = current_data_char;
}

// The CCITT CRC 16 polynomial is X + X + X + 1.
// In binary, this is the bit pattern 1 0001 0000 0010 0001, and in hex it
// is 0x11021.
// A 17 bit register is simulated by testing the MSB before shifting
// the data, which affords us the luxury of specifying the polynomial as a
// 16 bit value, 0x1021.
// Due to the way in which we process the CRC, the bits of the polynomial
// are stored in reverse order. This makes the polynomial 0x8408.

/*
// note: when the crc is included in the message, the valid crc is:
//      0xF0B8, before the compliment and byte swap,
//      0x0F47, after compliment, before the byte swap,
//      0x470F, after the compliment and the byte swap.
*/

/*****
//
// crcl16() - generate a 16 bit crc
//
//
// PURPOSE
//      This routine generates the 16 bit remainder of a block of
//      data using the ccitt polynomial generator.
//
//

```

```

// CALLING SEQUENCE
//     crc = crc16(data, len);
//
// PARAMETERS
//     data    <-- address of start of data block
//     len     <-- length of data block
//
// RETURNED VALUE
//     crc16 value. data is calculated using the 16 bit ccitt polynomial.
//
// NOTES
//     The CRC is preset to all 1's to detect errors involving a loss
//     of leading zero's.
//     The CRC (a 16 bit value) is generated in LSB MSB order.
//     Two ways to verify the integrity of a received message
//     or block of data:
//     1) Calculate the crc on the data, and compare it to the crc
//     calculated previously. The location of the saved crc must be
//     known.
//     2) Append the calculated crc to the end of the data. Now calculate
//     the crc of the data and its crc. If the new crc equals the
//     value in "crc_ok", the data is valid.
//
// PSEUDO CODE:
//     initialize crc (-1)
//     DO WHILE count NE zero
//     DO FOR each bit in the data byte, from LSB to MSB
//         IF (LSB of crc) EOR (LSB of data)
//             crc := (crc / 2) EOR polynomial
//         ELSE
//             crc := (crc / 2)
//         FI
//     OD
//     OD
//     1's compliment and swap bytes in crc
//     RETURN crc
//
*****/

```

```

uint16_t crc16(char *data_p,unsigned short length)
{
    unsigned char i;
    uint32_t data;
    uint32_t crc;

    crc = 0xFFFF;

    if (length == 0)
        return (~crc);

    do {
        for (i = 0, data = (unsigned int)0xff & *data_p++;
            i < 8;
            i++, data >>= 1) {
            if ((crc & 0x0001) ^ (data & 0x0001))
                crc = (crc >> 1) ^ POLY;
            else
                crc >>= 1;
        }
    } while (--length);

    crc = ~crc;

    data = crc;
    crc = (crc << 8) | (data >> 8 & 0xFF);

    return (crc);
}

```

14.2 Receiver Unit Source Code

```
//INCLUDES
#include "msp430x21x2.h"
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>

//DEFINES
#define POLYNOMIAL 0x88108000 /* CCITT polynomial (x^16 + x^12 + x^5 + 1) followed
by 0's */
#define POLY 0x8408

//GLOBALS
unsigned int new_cap=0;
unsigned int old_cap=0;
unsigned int cap_diff=0;
unsigned int store,index=0;
unsigned int j=0;
unsigned int k=0;
unsigned int mode =0;
unsigned int half_count = 0;
unsigned int bit_count = 0;
unsigned int byte_count = 0;
unsigned int mask = 128;
unsigned int bit_prev = 0;
unsigned int block_finished;
unsigned int current_bit = 0;
unsigned int skip_trans = 0;
unsigned int skip_set =0;
//unsigned int diff_array[150]; // RAM array for differences
int hit_red = 0;
int hit_orange = 0;
int hit_yellow = 0;
int hit_green = 0;
int stored_byte_count = 0;
uint32_t i;
int UART_mode = 0;
int UART_rx_index =0;
int ch_index = 0;
int not_serial = 1;
char data_byte = 0x00;
char data;
char current_data_char = 0x00;
char last_rx_char = 'a';
char data_block[25];
char ch_buffer[14];
char ID_array[14];
char universal_ID[14];
char *Flash_ptr = (char *)0x1000; // Initialize Flash pointer

//FUNCTION PROTOTYPES

void config_Clocks();
void config_UART();
void config_RxTimer();
void print_BridgeID();
void print_BlockstoUART();
void print_byte_to_UART(char x);
void store_and_process();
int BridgeID_Repeat();
uint16_t crc16(char *data_p,unsigned short length);
//END (FUNCTION PROTOTYPES)

//////////////////////////////////START MAIN//////////////////////////////////
//////////////////////////////////
```

```

void main(void)
{
    char *ptr;
    uint16_t crc;

    WDTCTL = WDTPW + WDTHOLD; //watchdog timer

    config_Clocks(); //Setup basic clock
    config_UART(); //Setup UART
    config_RxTimer(); //Setup Timer

//Assign Dummy universal ID, can be changed through UART input
universal_ID[0] = 0xBB;
universal_ID[1] = 0xFF;
universal_ID[2] = 0xFF;
universal_ID[3] = 0xFF;
universal_ID[4] = 0xFF;
universal_ID[5] = 0xFF;
universal_ID[6] = 0xFF;
universal_ID[7] = 0xFF;
universal_ID[8] = 0xFF;
universal_ID[9] = 0xFF;
universal_ID[10] = 0xFF;
universal_ID[11] = 0xFF;
universal_ID[12] = 0xFF;
universal_ID[13] = 0xBB;

    for(i = 0; i < 25;i++)
        data_block[i] = 0x00;
    for(i = 0; i < 150; i++)
    {
        // diff_array[i] = 0x0000;
        //capture_array[i] = 0x0000;
    }
    //Outputs SMCLK through 2.1 (pin 9) and ACLK through 2.0 (pin 8)
    //P1DIR = P1_RFSLE + P1_RFSDATA + P1_RFSClk + P1_RFCE; //Sets
Port1 GIO

    //Setup I/O pins for output, initialize to low
    P1SEL = 0x02;
    P1DIR |= 0x01;
    P2DIR |= 0x18;
    P3DIR |= 0x80;
    P1OUT &= !(0x01);

    P2OUT &= !(0x18);
    P3OUT &= !(0x80);
    // P2OUT |= 0x10;
    //P3OUT |= 0x80;
    _NOP();
    //Enable Interrupt for RFDATACLK
    //P1IE |= P1_RFDATACLK;
    //P1IFG &= P1_RFDATACLK;

//Green LED will Blink upon startup then go low
for(i = 0; i < 100000; i++)
    P1OUT |= 0x01;

    P1OUT &= !(0x01);

//Main receiver loop running continuously
while(1)
{
    //Once message has been received process

```

```

        if(mode == 2)
        {
                ptr = &data_block[2]; //point to message starting past
preamble
                crc = crc16(ptr, 23); //get crc of message
                if(crc == 0x470F) //message ok if 0x470F returned
                {
                        //stores data and toggles LED lines
messages
                if(BridgeID_Repeat() == 1) //checks for bridge ID and repeated
                        store_and_process();
                }
                mode = 0; //reset mode
                CCTL0 = CM_3 + CCIS_0 + CAP + CCIE; //Rising Edge, Ssync
Capture Mode,
                //
Capture on Input pin P1.1 Capture Mode,
                //Intterrupts Enabled
                TACTL = MC_2 + TASSEL_1 + TACLK;
                //SMCLK, Continuous Modess
        }
}

//sets up basic clock
void config_Clocks()
{
        unsigned long i;
        if (CALBC1_8MHZ ==0xFF || CALDCO_8MHZ == 0xFF)
        {
                while(1); // If calibration constants erased
                        // do not load, trap CPU!!
        }
        BCSCCTL1 = CALBC1_8MHZ; // Set DCO to 1\16MHz
        DCOCTL = CALDCO_8MHZ;
        FCTL2 = FWKEY + FSSEL0 + FN5 + FN3 + FN2 + FN1; // MCLK/24 for Flash Timing

        BCSCCTL1 |= XTS; // ACLK = LFXT1 = HF XTAL
        BCSCCTL2 |= SELM_3 + SELS; // MCLK = LFXT1 (safe)
        BCSCCTL3 |= LFXT1S1 + XT2S1; // 3 - 16MHz crystal or
resonator
do
{
        IFG1 &= ~OFIFG; // Clear OSCFault flag
                for (i = 0xFF; i > 0; i--); // Time for flag to set
        } while (IFG1 & OFIFG); // OSCFault flag still set?
}

//sets up UART
void config_UART()
{
        P3SEL = 0x30; // P3.4,5 = USCI_A0 TXD/RXD
        UCA0CTL1 |= UCSSEL_3; // SMCLK
        UCA0BR0 = 0xA0; // 8MHz 19200
        UCA0BR1 = 0x01; // 8MHz 19200
        UCA0MCTL = UCBR0; // Modulation UCBR0x = 1
        UCA0CTL1 &= ~UCSWRST; // **Initialize USCI state machine**
        IE2 |= UCA0RXIE; // Enable USCI_A0 RX interrupt
}

//Sets up timer
void config_RxTimer()
{
        CCTL0 = CM_3 + CCIS_0 + CAP + CCIE; //Rising Edge, Ssync Capture Mode,
                //
Capture on Input pin P1.1 Capture Mode,
                //Intterrupts Enabled

```

```

        TACTL = MC_2 + TASSEL_1 + TACLK;           //SMCLK, Continuous
Modess    BIS_SR(GIE);                           //enable global
interrupt;
}

// Interrupt for UART
#pragma vector=USCIAB0RX_VECTOR
__interrupt void USCI0RX_ISR(void)
{
    int i, k = 1;
    char *ptr2;
    char tx_finish[] = {'d','a','t','a','_','t','x','e','d','!'};
    char reset[] = {'R','e','c','e','i','v','e','r','_','R','e','s','e','t'};

//If system reset entered, reset system
if(UCA0RXBUF == '&' && last_rx_char == '&')
{
    hit_green = 0;
    hit_yellow = 0;
    hit_orange = 0;
    hit_red = 0;

    P1OUT &= !(0x01);
    P2OUT &= !(0x18);
    P3OUT &= !(0x80);

    for(i = 0;i<14;i++)
    {
        while (!(IFG2 & UCA0TXIFG));
// USCI_A0 TX buffer ready?
        while ( (UCA0STAT & UCBSY ));
        UCA0TXBUF = reset[i];
    }
}

// while (!(IFG2 & UCA0TXIFG));           // USCI_A0 TX buffer ready?
//UCA0TXBUF = UCA0RXBUF;                 // TX -> RXed character

//Option to print Bridge ID
if(UCA0RXBUF == '1'&& last_rx_char == '#')
{
    print_BridgeID(); //prints bridge ID to UART
    while (!(IFG2 & UCA0TXIFG));           //
USCI_A0 TX buffer ready?
    while ( (UCA0STAT & UCBSY ));
    UCA0TXBUF = 10;
    while (!(IFG2 & UCA0TXIFG));           //
USCI_A0 TX buffer ready?
    while ( (UCA0STAT & UCBSY ));
    UCA0TXBUF = 13;
}

//Prints the blocks received to UART
if(UCA0RXBUF == '2' && last_rx_char == '#')
{
    print_BlockstoUART();
    while (!(IFG2 & UCA0TXIFG));           //
USCI_A0 TX buffer ready?
    while ( (UCA0STAT & UCBSY ));
    UCA0TXBUF = 10;
    while (!(IFG2 & UCA0TXIFG));           //
USCI_A0 TX buffer ready?
    while ( (UCA0STAT & UCBSY ));
    UCA0TXBUF = 13;
}

```

```

}

//Entered if Bridge ID is to be entered (mode 1) or Test Bridge ID (mode 2)
if(UART_mode == 1 || UART_mode == 2)
{
    //Characters from hyperterminal must be translated to
    be stored

    if(ch_index % 2 == 0)
    {
        switch(UCA0RXBUF) {
            case '0':
                current_data_char = 0x00;
                break;
            case '1':
                current_data_char = 0x10;
                break;
            case '2':
                current_data_char = 0x20;
                break;
            case '3':
                current_data_char = 0x30;
                break;
            case '4':
                current_data_char = 0x40;
                break;
            case '5':
                current_data_char = 0x50;
                break;
            case '6':
                current_data_char = 0x60;
                break;
            case '7':
                current_data_char = 0x70;
                break;
            case '8':
                current_data_char = 0x80;
                break;
            case '9':
                current_data_char = 0x90;
                break;

            case 'A':
                current_data_char = 0xA0;
                break;
            case 'B':
                current_data_char = 0xB0;
                break;
            case 'C':
                current_data_char = 0xC0;
                break;
            case 'D':
                current_data_char = 0xD0;
                break;
            case 'E':
                current_data_char = 0xE0;
                break;
            case 'F':
                current_data_char = 0xF0;
                break;
            default:
                current_data_char = 0x00;
                break;
        }
    }

    if(ch_index % 2 == 1)
    {
        switch(UCA0RXBUF) {

```

```

        case '0':
            current_data_char += 0x00;
            break;
        case '1':
            current_data_char += 0x01;
            break;
        case '2':
            current_data_char += 0x02;
            break;
        case '3':
            current_data_char += 0x03;
            break;
        case '4':
            current_data_char += 0x04;
            break;
        case '5':
            current_data_char += 0x05;
            break;
        case '6':
            current_data_char += 0x06;
            break;
        case '7':
            current_data_char += 0x07;
            break;
        case '8':
            current_data_char += 0x08;
            break;
        case '9':
            current_data_char += 0x09;
            break;

        case 'A':
            current_data_char += 0x0A;
            break;
        case 'B':
            current_data_char += 0x0B;
            break;
        case 'C':
            current_data_char += 0x0C;
            break;
        case 'D':
            current_data_char += 0x0D;
            break;
        case 'E':
            current_data_char += 0x0E;
            break;
        case 'F':
            current_data_char += 0x0F;
            break;
        default:
            current_data_char += 0x00;
            break;
    }

    ch_buffer[UART_rx_index] =
current_data_char;

    UART_rx_index++;
}

    ch_index++;

//last character stored, place in flash memory
if(UART_rx_index ==14)
{
    //ptr2 = &ch_buffer[0];
    //calculates crc from data

    //write_SegB(ptr2,crc, 23);

```

```

ptr2 = (char *) 0x10B1;

FCTL3 = FWKEY; // Clear Lock bit
FCTL1 = FWKEY + ERASE; // Set Erase bit
*ptr2 = 0;

FCTL1 = FWKEY + WRT;

for(i = 0; i<14;i++)
{
    if (UART_mode == 1)
    {
        *ptr2++ = ch_buffer[i];
        ID_array[i] = ch_buffer[i];
    }
    else
        universal_ID[i] = ch_buffer[i];
}

FCTL1 = FWKEY; // Clear WRT bit
FCTL3 = FWKEY + LOCK; // Set LOCK bit

UART_mode = 0;
UART_rx_index = 0;
ch_index = 0;
for( i=0; i<10;i++)
{
    while (!(IFG2 & UCA0TXIFG)); //
USCI_A0 TX buffer ready?
    while ( (UCA0STAT & UCBSY ));
    UCA0TXBUF = tx_finish[i];
// TX character
}
    TA0CCTL0 |= CCIE;
    not_serial = 1;
    while (!(IFG2 & UCA0TXIFG)); //
USCI_A0 TX buffer ready?
    while ( (UCA0STAT & UCBSY ));
    UCA0TXBUF = 10;
    while (!(IFG2 & UCA0TXIFG)); //
USCI_A0 TX buffer ready?
    while ( (UCA0STAT & UCBSY ));
    UCA0TXBUF = 13;
    UART_mode = 0;
}

}

//Will enter Bridge ID
if (UCA0RXBUF == '*' && last_rx_char == '*')
{
    UART_mode = 1;
}
//Will enter Test Bridge ID
else if (UCA0RXBUF == '$' && last_rx_char == '$')
{
    UART_mode = 2;
}
last_rx_char = UCA0RXBUF;
}

```

```

//interrupt for RF data line input
#pragma vector=TIMER0_A0_VECTOR
__interrupt void Timer0_A0(void)
{
    new_cap = (unsigned int) TACCR0;
    //capture time on edge
    cap_diff = new_cap - old_cap;
    // if (new_cap != old_cap)
    //{
    //capture_array[index] = new_cap;
    old_cap = new_cap;

    // diff_array[index] = cap_diff;          // record difference to RAM array
    // diff_array[0] = 4;
    index++;
    //}

    //Receiving message, manchester encoded, refer to documentation for method
    (figure 39 of final report)
    //Basic logic:
    //---The sync pulse is high for 18000 clock cycles
    //---Following the sync pulse the first bit will be a 1 (preamble AA55)
    //---a 1 means a high to low mid bit transition
    //---This will add 800 clock cycles to sync pulse
    //---When reciever sees a pulse of around 19000 we assume this is the case
    //---This stores a 1 as the first bit
    //---From that point on the logic is simple:
    //      ---if 1600 clock cycles between edges (in this case midbit to
midbit) is seen the bit
    //      ---will be opposite the previous bit
    //
    //      --if 800 clock cycles between edges is seen, then it will be
followed by
    //      --another 800 clock cycle period between edges
    //      --this represents the new bit being the same as the previous bit
    if(mode == 1 )
    {
        if( (cap_diff > 650) && (cap_diff < 950) && (bit_prev == 0))
        {
            if(half_count % 2 == 1)
            {
                mask = mask / 2;
                bit_count++;
            }
            bit_prev = 0;
            half_count++;
        }
        else if( (cap_diff > 1450) && (cap_diff < 1750) && (bit_prev == 0))
        {
            data_byte = data_byte | mask;
            mask = mask / 2;
            bit_count++;
            bit_prev = 1;
            skip_set = 0;
            half_count = 0;
        }
        else if( (cap_diff > 650) && (cap_diff < 950) && (bit_prev == 1))
        {
            data_byte = data_byte | mask;
            if(half_count % 2 == 1)
            {
                mask = mask / 2;
                bit_count++;
            }
            skip_trans = 1;
            bit_prev = 1;
            half_count++;
            skip_set = 1;
        }
        else if( (cap_diff > 1450) && (cap_diff < 1750) && (bit_prev == 1))
        {
            mask = mask / 2;

```

```

        bit_count++;
        bit_prev = 0;
        skip_set = 0;
        half_count = 0;
    }

// if(skip_trans ==1)
// skip_trans = 0;
//byte done
if(bit_count == 8)
    {
        data_block[byte_count] = data_byte;
        data_byte = 0x00;
        bit_count = 0;
        byte_count++;
        mask = 128;

    }

//if (data_block[0] == 0xAA)
// _NOP();

//data block done
if(byte_count == 25)
{
    mode = 2;
    bit_count = 0;
    byte_count = 0;
    skip_trans = 0;
    mask = 128;

    CCTLO = 0;
    TACTL = 0;

}

//sync pulse plus first bit of preamble
if(cap_diff > 17500&& cap_diff < 30000)
{
    mode = 1;
    bit_prev = 1;
    data_byte = data_byte | mask;
    mask = mask / 2;
    bit_count++;
    skip_trans = 0;

}

//abberant pulse, throw out
if(cap_diff > 30000)
{
    mode = 0;
    bit_count = 0;
    byte_count = 0;
    skip_trans = 0;
    mask = 128;

}

    if(index == 150)        //Is memory full?
    {
        index = 0;
    }
} // end ISR

void store_and_process()
{

    FCTL3 = FWKEY;                // Clear Lock bit

```

```

FCTL1 = FWKEY + WRT;                // Set WRT bit for write operation

for (i = 0; i < 25; i++)
{
    *Flash_ptr++ = data_block[i];    // Write value to flash
    stored_byte_count++;
    if (data_block[22] == 0 && hit_red == 0 && hit_orange == 0 &&
hit_yellow == 0)
    {
        P1OUT |= 0x01;                // Set Green LED high,
others low
        P2OUT &= !(0x18);
        P3OUT &= !(0x80);
        hit_green = 1;
    }
    else if (data_block[22] == 1 && hit_red == 0 && hit_orange == 0)
    {
        P1OUT &= !(0x01);            // Set Orange LED
high, others low
        P2OUT &= !(0x08);
        P2OUT |= 0x10;
        P3OUT &= !(0x80);
        hit_yellow = 1;
    }
    else if (data_block[22] == 2 && hit_red == 0)
    {
        P1OUT &= !(0x01);            // Set Yellow LED
high, others low
        P2OUT &= !(0x10);
        P2OUT |= 0x08;
        P3OUT &= !(0x80);
        hit_orange = 1;
    }
    else if (data_block[22] == 3)
    {
        P1OUT &= !(0x01);            // Set Red LED high,
others low
        P2OUT &= !(0x18);
        P3OUT |= 0x80;
        hit_red = 1;
    }
}

FCTL1 = FWKEY;                      // Clear WRT bit
FCTL3 = FWKEY + LOCK;                // Set LOCK bit
}

//prints bridge ID to UART
void print_BridgeID()
{
    char *print_ptr;
    int i,j,k;
    char readout[] = {'B','r','i','d','g','e','_','I','D',' ':'};

    print_ptr = (char *) 0x10B1;
    k=0;

    for(j =0;j<10;j++)
    {
        while (!(IFG2 & UCA0TXIFG)); //
USCI_A0 TX buffer ready?
        while ( (UCA0STAT & UCBSY ));
        UCA0TXBUF = readout[j];
    }
}

for(i=0;i<14;i++)
{

```

```

print_byte_to_UART(*print_ptr++);
}

//prints data blocks to UART
void print_BlockstoUART()
{
char *print_ptr;
int i,j,k;
char readout[] = {'d','a','t','a',' ','b','l','o','c','k'};

print_ptr = (char *) 0x1000;
k=48;
for(i=0;i<stored_byte_count;i++)
{
    if(i %25 == 0)
    {
        k++;
        for(j =0;j<10;j++)
        {
            while (!(IFG2 & UCA0TXIFG)); //
USCI_A0 TX buffer ready?
            while ( (UCA0STAT & UCBUSY ));
UCA0TXBUF = readout[j];
        }
            while (!(IFG2 & UCA0TXIFG)); //
USCI_A0 TX buffer ready?
            while ( (UCA0STAT & UCBUSY ));
UCA0TXBUF = k;
        }
            while (!(IFG2 & UCA0TXIFG)); //
USCI_A0 TX buffer ready?
            while ( (UCA0STAT & UCBUSY ));
UCA0TXBUF = ':';
        }
    }
print_byte_to_UART(*print_ptr++);
}

//prints individual bytes to UART
void print_byte_to_UART(char x)
{
int current_data_char;
char second_nibble_left_4 = (x << 4);
char second_nibble_iso = second_nibble_left_4 >> 4;
char first_nibble_iso = x >> 4;
    switch( first_nibble_iso){
        case 0:
            current_data_char = 48;
            break;
        case 1:
            current_data_char = 49;
            break;
        case 2:
            current_data_char = 50;
            break;
        case 3:
            current_data_char = 51;
            break;
        case 4:
            current_data_char = 52;
            break;
        case 5:
            current_data_char = 53;
            break;
        case 6:
            current_data_char = 54;
            break;
        case 7:
            current_data_char = 55;
    }
}

```

```

        break;
        case 8:
            current_data_char = 56;
            break;
        case 9:
            current_data_char = 57;
            break;

        case 10:
            current_data_char = 65;
            break;
        case 11:
            current_data_char = 66;
            break;
        case 12:
            current_data_char = 67;
            break;
        case 13:
            current_data_char = 68;
            break;
        case 14:
            current_data_char = 69;
            break;
        case 15:
            current_data_char = 70;
            break;
        default:
            current_data_char = 0x00;
            break;
    }

    while (!(IFG2 & UCA0TXIFG)); //

USCI_A0 TX buffer ready?

while ( (UCA0STAT & UCBSY ) );
UCA0TXBUF = current_data_char;

switch(second_nibble_iso){
    case 0:
        current_data_char = 48;
        break;
    case 1:
        current_data_char = 49;
        break;
    case 2:
        current_data_char = 50;
        break;
    case 3:
        current_data_char = 51;
        break;
    case 4:
        current_data_char = 52;
        break;
    case 5:
        current_data_char = 53;
        break;
    case 6:
        current_data_char = 54;
        break;
    case 7:
        current_data_char = 55;
        break;
    case 8:
        current_data_char = 56;
        break;
    case 9:
        current_data_char = 57;
        break;
}

```

```

        case 10:
            current_data_char = 65;
            break;
        case 11:
            current_data_char = 66;
            break;
        case 12:
            current_data_char = 67;
            break;
        case 13:
            current_data_char = 68;
            break;
        case 14:
            current_data_char = 69;
            break;
        case 15:
            current_data_char = 70;
            break;
        default:
            current_data_char = 0x00;
            break;
    }

    while (!(IFG2 & UCA0TXIFG)); //

USCI_A0 TX buffer ready?
    while ( (UCA0STAT & UCIBUSY ) );
    UCA0TXBUF = current_data_char;
}

//checks for correct Bridge ID, the test Bridge ID, and that the data block isn't a
repeat
int BridgeID_Repeat()
{
    int i, IDok = 1, repeat = 0, test_ID = 1;

    //char *check_ptr;
    //check_ptr = (char *) 0x10B1;
    for (i =2; i <16; i++)
    {
        if (universal_ID[i-2] != data_block[i])
            test_ID = 0;
    }

    if (test_ID == 1)
    {
        P1OUT |= (0x01);
        P2OUT |= (0x18);
        P3OUT |= (0x80);
    }

    for (i =2; i <16; i++)
    {
        if (ID_array[i-2] != data_block[i])
            IDok =0;
        /*check_ptr++;
    }

    if(hit_green == 1 && data_block[22] == 00)
        repeat =1;
    else if(hit_yellow == 1 && data_block[22] == 01)
        repeat = 1;
    else if(hit_orange == 1 && data_block[22] == 02)
        repeat = 1;
    else if(hit_red == 1 && data_block[22] == 03)
        repeat = 1;

    if (IDok == 1 && repeat == 0)
        return 1;
    else
        return 0;
}

```

```

    }

    // The CCITT CRC 16 polynomial is X + X + X + 1.
// In binary, this is the bit pattern 1 0001 0000 0010 0001, and in hex it
// is 0x11021.
// A 17 bit register is simulated by testing the MSB before shifting
// the data, which affords us the luxury of specifying the polynomial as a
// 16 bit value, 0x1021.
// Due to the way in which we process the CRC, the bits of the polynomial
// are stored in reverse order. This makes the polynomial 0x8408.

/*
// note: when the crc is included in the message, the valid crc is:
//      0xF0B8, before the compliment and byte swap,
//      0x0F47, after compliment, before the byte swap,
//      0x470F, after the compliment and the byte swap.
*/

/*****
//
// crc16() - generate a 16 bit crc
//
// PURPOSE
//      This routine generates the 16 bit remainder of a block of
//      data using the ccitt polynomial generator.
//
// CALLING SEQUENCE
//      crc = crc16(data, len);
//
// PARAMETERS
//      data    <-- address of start of data block
//      len     <-- length of data block
//
// RETURNED VALUE
//      crc16 value. data is calculated using the 16 bit ccitt polynomial.
//
// NOTES
//      The CRC is preset to all 1's to detect errors involving a loss
//      of leading zero's.
//      The CRC (a 16 bit value) is generated in LSB MSB order.
//      Two ways to verify the integrity of a received message
//      or block of data:
//      1) Calculate the crc on the data, and compare it to the crc
//         calculated previously. The location of the saved crc must be
//         known.
//      2) Append the calculated crc to the end of the data. Now calculate
//         the crc of the data and its crc. If the new crc equals the
//         value in "crc_ok", the data is valid.
//
// PSEUDO CODE:
//      initialize crc (-1)
//      DO WHILE count NE zero
//          DO FOR each bit in the data byte, from LSB to MSB
//              IF (LSB of crc) EOR (LSB of data)
//                  crc := (crc / 2) EOR polynomial
//              ELSE
//                  crc := (crc / 2)
//          FI
//      OD
//      OD
//      1's compliment and swap bytes in crc
//      RETURN crc
*****/

```

```

uint16_t crc16(char *data_p, unsigned short length)
{
    unsigned char i;
    uint32_t data;
    uint32_t crc;

    crc = 0xFFFF;

    if (length == 0)
        return (~crc);

    do {
        for (i = 0, data = (unsigned int)0xff & *data_p++;
            i < 8;
            i++, data >>= 1) {
            if ((crc & 0x0001) ^ (data & 0x0001))
                crc = (crc >> 1) ^ POLY;
            else
                crc >>= 1;
        }
    } while (--length);

    crc = ~crc;

    data = crc;
    crc = (crc << 8) | (data >> 8 & 0xFF);

    return (crc);
}

```

15.0 Development Software Programs Used

The following software tools are required for this system.

1. Code Composer Essentials v3 Core Edition - This program is used to develop code to be run on MSP430 devices. The code can be developed and downloaded to the microcontroller using this software through a JTAG connection. This program can be freely obtained at the following website:

<http://focus.ti.com/docs/toolsw/folders/print/msp-cce430.html>

2. FET-Pro430 - This is a flash programmer made for the MSP430 devices. Takes an output file created from code for direct download to the microcontroller. The programmer is made by Elprotronic Inc. and can be freely downloaded at:

<http://www.elprotronic.com/download.html>

3. ExpressPCB - This program provides a develop environment for creating printed circuit board layouts. The layout file created by this program can be used to have a printed circuit board produced by ExpressPCB. This software can be freely obtained at the following website:

http://www.expresspcb.com/ExpressPCBHtml/Free_cad_software.htm

4. ExpressSCH - This program provides a develop environment for creating schematics for the printed circuit board. These schematics can be imported into ExpressPCB and used to double check connections. This software can be freely obtained at the following website:

http://www.expresspcb.com/ExpressPCBHtml/Free_cad_software.htm

16.0 References

- [1] "BUREAU OF DESIGN BRIDGE MANAGEMENT SYSTEM 2 (BMS2) CODING MANUAL OFFICE VERSION," Commonwealth Of Pennsylvania Department Of Transportation, Pub. 100A, July 2007.
- [2] "LR SERIES TRANSMITTER MODULE DATA GUIDE". Linx Technologies, Inc. January 2008.
- [3] "LR SERIES RECEIVER MODULE DATA GUIDE". Linx Technologies, Inc. January 2008.
- [4] American Society for Testing and Materials. <http://www.astm.org/Standards/F794.htm>
- [5] Nicholas Cheremisinoff and Ramesh Gupta. "HANDBOOK OF FLUIDS IN MOTION". Butterworth Publishers. Stoneham, MA. 1983.
- [6] Olson, Reuben. "ESSENTIALS OF ENGINEERING FLUID MECHANICS". International Textbook Company. Scranton, Pennsylvania. 1961.
- [7] Lloyd A. Reed and Marla H. Stuckey. "PREDICTION OF VELOCITIES FOR A RANGE OF STREAM CONDITIONS IN PENNSYLVANIA". U.S. Geological
- [8] Markus Koesler, Wolfgang Lutsch. "PROGRAMMING A FLASH-BASED MSP430 USING THE JTAG INTERFACE". Texas Instruments. February 2008.
- [9] "ABLS SERIES". Abacron Corporation. November 2008.
- [10] MSP430x2xx FAMILY USER'S GUIDE". Texas Instruments. 2008.
- [11] "LM3940". National Semiconductor. July 2007
- [12] William A. Shay, Understanding Data Communications & Networks, Pacific Grove, CA, Brooks/Cole Publishing Company, 1999.
- [13] IEEE 1149.1-2001 IEEE standard test access port and boundary-scan architecture, IEEE, 2001, ISBN: 0-7381-2944-5.

17.0 System Manual

The following sections contain the system manual describing how to construct and deploy the Remote Sensing for Bridge Scour System. Details about each section are contained in the preceding sections of this document.

18.0 Hardware Components

This section contains a listing of all hardware used in the system. Following this listing is a brief description of the components. The section concludes with a Bill of Materials that will detail purchasing and cost information of the components.

18.1.1 Sensor Unit Hardware

The following is a list of the hardware used for the Sensor Unit and a short description:

Two-Sided Printed Circuit Board (PCB) – A two-sided PCB was used to house and connect the Sensor Unit circuitry. The board has dimensions of 1.5-inch length by 4.0-inch width. It is composed of FR-4 epoxy glass and copper on the surface. The PCB design file was created using ExpressPCB software. The board was manufactured from ExpressPCB and can be purchased through the ExpressPCB website.

Microcontroller – The microcontroller used is the MSP430F2132, manufactured by Texas Instruments. This microcontroller services all computational, timing, and I/O control required by the Sensor Unit. The microcontroller is a 16-pin SSOP chip (an integrated chip packaging type). This microcontroller corresponds to the label U1 on both the Sensor Unit schematic and PCB design file (layout file). The chip is readily available through Digi-Key.

Radio Frequency (RF) Transmitter – The RF Transmitter used is the TXM-433-LR chip, manufactured by Linx Technology. The transmitter uses On-Off keying (OOK) modulation to transmit, wirelessly, the Sensor Unit Data to the Receiver Unit. The transmitter is an 8-pin chip. The transmitter corresponds to the label U2 on both the Sensor Unit schematic and PCB design file. The chip is readily available through Digi-Key.

Serial (RS-232) to USB Converter – The serial (RS-232) to USB converter used is the FT232RL chip, manufactured by Future Technology Devices International, Ltd. The serial to USB converter acts as an intermediary between the USB front end (connected via the USB Mini-Port Connector to a host PC or laptop) and the UART (Universal Asynchronous Receiver/Transmitter) of the microcontroller. This is a 28-pin surface mount chip. The converter corresponds to the label U3 on both the Sensor Unit schematic and PCB design file. This component is readily available through Digi-Key.

Crystal Oscillator – The crystal used with the microcontroller on the Sensor Unit was an 8 MHz crystal. This crystal is part number MA-505 8.0000M-C0, produced by Epson Toyocom Corporation. The crystal has two surface mount pads. The crystal corresponds to the label Y1 on both the Sensor Unit schematic and PCB design file. This component is readily available through Digi-Key.

Relay Switch – The relay switch used is the DS1E-SL-DC3V relay produced by Panasonic Electric Works. This relay is responsible for connecting the voltage potential of the batteries to the other active components of the circuit, given the state of both the arm switch and tilt switches. The relay has five pins. The relay corresponds to RE1 on both the Sensor Unit schematic and PCB design file. This component is readily available through Digi-Key.

Arm Switches – Two switches are used to arm and reset the Sensor Unit. One is an on-board switch (used for testing and debugging purposes) the other is an external switch (used for field deployment). Both are manufactured by NKK Switches of America Inc. The on-board switch is part number AS23AP. The external switch is part number HB16CKW01.

Tilt Switches – Three tilt switches are used on the Sensor Unit. These tilt switches are made by Comus Group of Companies with the part number of S1234. These switches are used to trigger the relay switch in the case that the Sensor Unit is released and floats (tilts) to its natural horizontal position. The switch is composed of three contacts. The tilt switch corresponds to SW1 on both the Sensor Unit schematic and PCB design file. This component is readily available through Digi-Key.

Voltage Regulator (3.3 Volt) – The voltage regulator used on the Sensor Unit is the LM3940IMP-3.3/NOPB voltage regulator produced by National Semiconductor. This regulator is used to reduce the five volts provided by the USB connection (provided by the host PC's or laptop's USB interface) to provide the required 3.3 volt power input to the serial to USB converter chip. The voltage regulator is composed of four pins. The regulator corresponds to the label VR1 on both the Sensor Unit schematic and PCB design file. This component is readily available through Digi-Key.

JTAG Connector – The JTAG connection on the Sensor Unit is realized using a 14 pin header. This header is of 0.1-inch pitch. The header used is part number 15-91-2140 by Molex/Waldom Electronics Corporation. The JTAG connection is used to download the program to run on the microcontroller. The JTAG corresponds to the label J1 on both the Sensor Unit schematic and PCB design file. This component is readily available through Digi-Key.

SMA Connector – The SMA type of connector used on the Sensor Unit is the CONREVSMA002 by Linx Technologies Inc. This is a reverse polarity SMA connector that forms a right angle with the face (top) of the PCB. It connects the output of the RF Transmitter chip to the antenna. The SMA connector corresponds to the label S1 on both the Sensor Unit schematic and PCB design file. This component is readily available through Digi-Key.

Antenna – The antenna used on the Sensor Unit is part ANT-433-CW-RH by Linx Technologies Inc. This is a quarter-wave, whip-type antenna with a length of two inches. The antenna is connected to the PCB by the SMA connector. The antenna is only shown as an output of the SMA connector (S1) on the Sensor Unit schematic and is not present on the PCB design file because it connects to the SMA connector. This component is readily available through Digi-Key.

USB Mini-Port Connector - The USB port used is part H2961CT-ND by Hirose Electric Co. Ltd. The port is used to connect the serial to UART chip to a host computer's USB port. The port is a standard mini-USB 2.0 connector with five pins. The USB connector corresponds to the label J2 on both the Sensor Unit schematic and PCB design file. This component is readily available through Digi-Key.

Batteries – The batteries used are part number CR-2 by Panasonic BSG. These batteries provide 3.0 volts for 850 milli-Amp hours. These batteries are the voltage source for the Sensor Unit. The batteries correspond to labels B1 and B2 on both the Sensor Unit schematic and PCB design file. This component is readily available through Digi-Key.

Positive Battery Connector – The positive battery connectors used are part number 5223 from Keystone. These connectors connect to the positive terminal of the batteries and help to hold the batteries to the PCB. The clips correspond to labels PBC1 and PBC2 on both the Sensor Unit schematic and PCB design file. This component is readily available through Digi-Key.

Negative Battery Connector – The negative battery connectors used are part number 5201 by Keystone. These connectors connect to the negative terminal of the batteries and help to hold the batteries to the PCB. The clips correspond to the label NBC1 and NBC2 on both the Sensor Unit schematic and PCB design file. This component is readily available through Digi-Key.

Capacitors – Several capacitors are used on the Sensor Unit. Each of their values and manufacturers are not listed here but can be found in the *Bill of Materials*, the *Sensor Unit Schematic*, or *Final Report*. There are 15 capacitors used in total on the Sensor Unit. Each of these capacitors is a 0603 package type. The capacitors correspond to C1 through C15 on both the Sensor Unit schematic and PCB design file.

Resistors – Four resistors were used on the Sensor Unit. Please see the *Bill of Materials*, the *Sensor Unit Schematic*, or *Final Report* for further details. Three of these resistors are part number ERJ-3GEYJ100V. These resistors are 10 Ohms. The other resistor is part number ERJ-3GEYJ473V. This resistor has a value of 47 kilo-Ohms. Both resistors are a 0603 package type and are made by Panasonic ECG. The 47 kilo-Ohm resistor corresponds to labels R1 while the 10-Ohm resistors correspond to labels R2 through R4 on both the Sensor Unit schematic and PCB design file. These resistors are readily available through Digi-Key.

Threaded PVC Pipe – The container (capsule) of the Sensor Unit is made of threaded PVC (Poly-Vinyl-Chloride) pipe also known PVC Pipe Nipple. The pipe used is part number 26PN by A to Z supply. The pipe is of 6-inch length and 2-inch inside diameter. In addition, circuitry besides an external switch must be placed within this pipe. This pipe is available through A to Z supply.

Threaded PVC Cap – Two caps are used to seal the 6-inch threaded PVC pipe. The caps obtained are part number 2PTCA by A to Z supply. The caps are 2-inch inside

diameter caps. They must be screwed onto the ends of the threaded PVC Pipe to seal it. These caps were obtained through A to Z supply.

18.1.2 Receiver Unit Hardware

The following is list of the hardware used on the Receiver Unit and short descriptions:

Two-Sided Printed Circuit Board (PCB) - A two-sided PCB was used to house and connect the Receiver Unit circuitry. The board has dimensions of 2.2-inch length by 4.5-inch width. It is composed of FR-4 epoxy glass and copper on the surface. The PCB design file was created using ExpressPCB software. The board was manufactured from ExpressPCB and can be purchased through the Express PCB website.

Microcontroller – The microcontroller used is the MSP430F2132, manufactured by Texas Instruments. This microcontroller services all computational, timing, and I/O control required by the Receiver Unit. The microcontroller is a 16-pin SSOP chip. This microcontroller corresponds to the label U1 on both the Receiver Unit schematic and PCB design file. The chip is readily available through Digi-Key.

RF Receiver - The RF Receiver used is the RXM-433-LR chip by Linx Technology. The receiver chip takes the On-Off keying modulation transmitted by the Sensor Unit and converts it to a digital output, which is then input into the microcontroller. The receiver is an 8-pin chip. The receiver corresponds to the label U2 on both the Receiver Unit schematic and PCB design file. The chip is readily available through Digi-Key.

Serial (RS-232) to USB Converter – The converter used is the FT232RL chip by Future Technology Devices International, Ltd. The Serial to USB converter acts as an intermediary between the USB front end (connected via the USB Mini-Port Connector to a host PC or laptop) and the UART of the microcontroller. This is a 28-pin surface mount chip. The converter corresponds to the label U3 on both the Receiver Unit schematic and PCB design file. This chip is readily available through Digi-Key.

Crystal Oscillator – The crystal used with the microcontroller on the Sensor Unit was an 8 MHz frequency crystal. This crystal is part MA-505 8.0000M-C0 by Epson Toyocom Corporation. The crystal has two surface mount pads. The crystal corresponds to the label Y1 on both the Sensor Unit schematic and PCB design file. This component is readily available through Digi-Key.

Voltage Regulator (3.0 Volts) – A voltage regulator used on the Receiver Unit is TPS78930DBVR component by Texas Instruments. This voltage regulator is used to reduce the voltage input from the power jack to 3.3 volts for use by the Receiver Unit circuitry. This regulator can handle any input voltage under 10 volts DC (direct current). The regulator is a 5-pin chip. The regulator corresponds to the label VR1 on both the Receiver Unit schematic and PCB design file. This component is readily available through Digi-Key.

Voltage Regulator (3.3 Volts) – Another voltage regulator used on the Receiver Unit is the LM3940IMP-3.3/NOPB voltage regulator by National Semiconductor. This regulator is used to reduce the five volts provided by the USB connection (the USB interface on the host PC or laptop provides these five volts) to provide the required 3.3 volt input to the Serial to USB chip. The voltage regulator is composed of four pins. The regulator corresponds to the label VR2 on both the Receiver Unit schematic and PCB design file. This component is readily available through Digi-Key.

JTAG Header - The JTAG connection on the Receiver Unit is realized using a 14-pin header. This header was of 0.1-inch pitch. The header used is part 15-91-2140 by Molex/Waldom Electronics Corporation. The JTAG connection is used to download the embedded software program that is run on the microcontroller. The JTAG corresponds to the label J1 on both the Receiver Unit schematic and PCB design file. This component is readily available through Digi-Key.

6-Pin Header – A 6-pin header is used on the Receiver Unit. This header was of 0.1-inch pitch. The header used is part TSW-106-07-G-S by Samtec Inc. This header is used to connect the 4 I/O (input/output) pins controlling the Light Indicator LEDs as well as the power and ground connections. The 6-pin header corresponds to the label J3 on both the Receiver Unit schematic and PCB design file. This component is readily available through Digi-Key.

SMA Connector – The SMA Connector used on the Receiver Unit is CONREVSMA001 by Linx Technologies Inc. This a reverse polarity SMA Connector that forms a right angle with the face (top) of the PCB. It is used to connect the RF Receiver chip to the antenna. The SMA Connector corresponds to the label SMA1 on both the Receiver Unit schematic and PCB design file. This component is readily available through Digi-Key.

Antenna – The antenna used on the Receiver Unit is part ANT-433-CW-QW by Linx Technologies Inc. This is a quarter-wave, whip-type antenna with a length of 6.81-inches. The antenna is connected to the PCB by the SMA Connector. The antenna is only shown as an output of the SMA Connector (SMA1) on the Receiver Unit schematic and is not shown on the PCB design file because the antenna connects to the SMA Connector. This component is readily available through Digi-Key.

USB Mini-Port Connector - The USB port used is part H2961CT-ND by Hirose Electric Co. Ltd. The port is used to connect the Serial to UART chip to a host computer's USB port. The port is a USB 2.0 connector with five pins. The USB Connector corresponds to the label J2 on both the Receiver Unit schematic and PCB design file. This component is readily available through Digi-Key.

Power Jack – A barrel-type power jack is used on the Receiver Unit. It is part PJ-202A by CUI Inc. The barrel has a 2.1 mm inside diameter and it has three through-hole connectors. The power jack corresponds to the label J4 on both the Receiver Unit schematic and PCB design file. This component is readily available through Digi-Key.

Batteries – The batteries used are part number CR-2 by Panasonic BSG. These batteries provide 3.0 volts for 850 milli-Amp hours. These batteries provide power for the Sensor Unit. The batteries correspond to the labels B1 and B2 on the Receiver Unit Schematic. This component is readily available through Digi-Key.

Positive Battery Connector – The positive battery connectors used are part number 5223 by Keystone. These connectors connect to the positive terminals of the batteries and help to hold the batteries to the PCB. The clips correspond to the labels PBC1 and PBC2 on both the Receiver Unit schematic and PCB design file. This component is readily available through Digi-Key.

Negative Battery Connector – The negative battery connectors used are part number 5201 by Keystone. These connectors connect to the negative terminal of the batteries and help to hold the batteries to the PCB. The clips correspond to the labels NBC1 and NBC2 on both the Receiver Unit schematic and PCB design file. This component is readily available through Digi-Key.

Capacitors – Several capacitors are used on the Receiver Unit. Each of their values and manufacturers are not listed here, but can be found in the *Bill of Materials*, the *Sensor Unit Schematic*, or *Final Report*. There are 15 capacitors used in total on the Sensor Unit. Each of these capacitors is a 0603 package type. The capacitors correspond to C1 through C15 on both the Sensor Unit Schematic and PCB Design File.

Resistor – One resistor is used on the Receiver Unit. The resistor used is part ERJ-3GEYJ473V. This resistor has a value of 47 kilo-Ohm. It is a 0603 package type and is made by Panasonic ECG. The 47 kilo-Ohm resistor corresponds to R1 on the Receiver Unit schematic and PCB design file. This resistor is readily available through Digi-Key.

18.1.3 Light Indicator Hardware

The following is a list of the hardware components used on the Light Indicator and short descriptions:

Dual 4 Input OR Gate – Two instances of the dual 4-input OR (logical OR) gate are used on the Light Indicator. The part number for this component is CD4072BNSR by Texas Instruments. These chips are used to drive the LEDs high, based on the I/O from the Receiver Unit. These chips are 14-pin surface mount packages. The OR gates correspond to labels U1 and U2 on the Light Indicator schematic and PCB design file. This component is readily available through Digi-Key.

6-Pin Header – Four instances of a 6-pin header are used on the Light Indicator. The header is part number TSW-106-07-G-S by Samtec. The headers receive the input from the Receiver Unit as well as a ground and power connections. The headers are labeled J1 through J4 on the Light Indicator schematic and PCB design file. This component is readily available through Digi-Key.

LEDs – Four LEDs are used on the Light Indicator. The four colors used are red, orange, yellow, and green. The red LED is part LTL-307EE; the orange LED is part LO 3360-

JM-24-0-10-BULK; the yellow LED is part LTL-10253W; and the green is part LTL-10233W. The red, yellow, and green LEDs are by Lite-On-Inc. while the orange LED is by Osram Opto Semiconductors Inc. The red, yellow, orange, and green LEDs are labeled L1, L2, L3, and L4 respectively. All four LEDs are available through Digi-Key.

Resistors – There are several resistors used on the Light Indicator. Four resistors are used on the Receiver Unit in total. Please see the *Bill of Materials*, the *Sensor Unit Schematic*, or *Final Report* for further details. These resistors are labeled R1 through R4 and their values are listed in the *Bill of Parts* for the Light Indicator Unit shown later.

19.0 Software

There are both software utilities and codes required to program and communicate with the Sensor and Receiver Units. The software utilities shown in this manual and used by the University of Pittsburgh during the design and testing are the recommended tools, but not the sole means by which the MSP430F2312 can be programmed or communicated with. The utilities used were chosen based on their availability, ease of use, and effectiveness. First, the software tools used for the embedded software development, to download the compiled source code to the microcontroller, and to interface with the devices is described. Next, the serial interface (UART Operation) and the protocol to store the Sensor Unit Data Block for the Sensor Unit and to access the Receiver Unit to reset, download received Sensor Unit Data Blocks, or to store the Bridge ID number is described. This section concludes with a brief description of the embedded software code.

19.1 Software Utilities

Three software utilities are used with the system. The first is Code Composer Essentials v3 Core Edition. This is used for the development and compilation of program code as well as for the debugging of the system. The second tool used is a flash programmer called FET-Pro430. This tool simply takes the final output file created from the program code and downloads it to the microcontroller. The other is HyperTerminal™. This is used for the UART communication and is installed with a standard Windows™ installation. These three utilities are discussed in the following subsections

19.1.1 Code Composer Essentials v3.1 Core Edition

The Code Composer Essentials v3.1 Core Edition (CCE3) utility is produced by Texas Instruments for the programming and debugging of the MSP430 family of microcontrollers. It utilizes a user-friendly Eclipse-style interface. The use of this tool within this Remote Sensing for Bridge Scour System is to modify program code and debug the microcontrollers on the Sensor and Receiver Units. This utility uses the USB/JTAG Connector MSP-FET430UIF to interface with the 14-pin JTAG Header on the Sensor Unit and Receiver Unit PCBs.

The program can be obtained at no cost directly from the Texas Instruments website at the following link: <http://focus.ti.com/docs/toolsw/folders/print/msp-cce430.html>. The program runs on either the Windows XP or Windows Vista Operating System. The tool has up to 16 KB of code space available for a given download. The Remote Sensing for Bridge Scour System does not approach this limit. Once

downloaded, follow the setup steps presented by the installation program to install and setup the program on the host PC or laptop.

When you start up the Code Composer Essentials, a directory must be chosen for a workspace as shown in Figure 84 below. This is the directory where the source code files are saved (stored). Code Composer Essentials suggests a default directory but any appropriate directory will also work. The source code to be used must be placed (stored) within this directory.

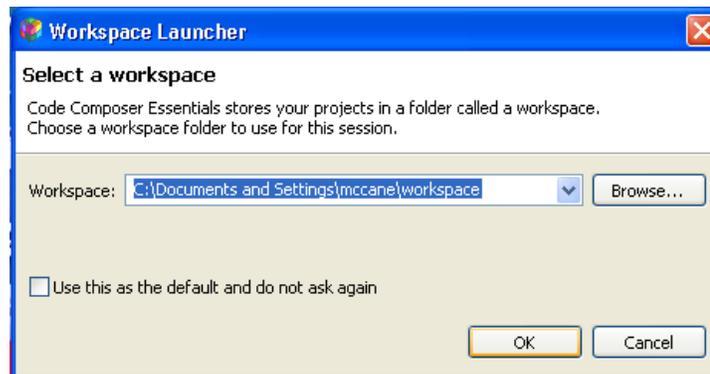


Figure 84: CCE3 Launch

Upon entering, the Integrated Development Environment (IDE) should bring up a welcome page with options that will help a user become accustomed to the environment. At this point, a project should be set up. To do this, click on the menu option: **File → New → Managed Make C/ASM Project**. This should bring up a screen prompting the user for a title to the project as shown in Figure 85 below. The project can be named as a user sees fit. Click “Next” after a name has been designated.

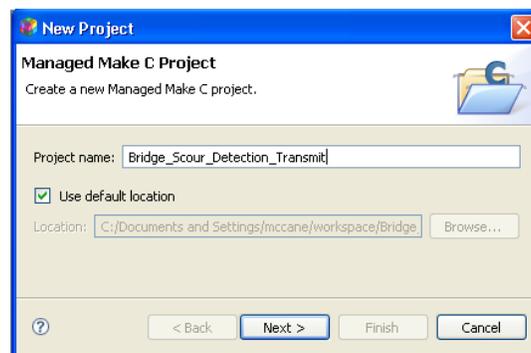


Figure 85: New Project Prompt

The two following screens require no action, and “Next” should be clicked on both screens. The next screen that comes up requires the user to select the microcontroller that is being used. For this system, the MSP430F2132 is being used. Select the **MSP430F2132** option from list “Device Variant” and the prompt should look as shown in Figure 86. All other options should be left alone and “Finish” should be clicked.

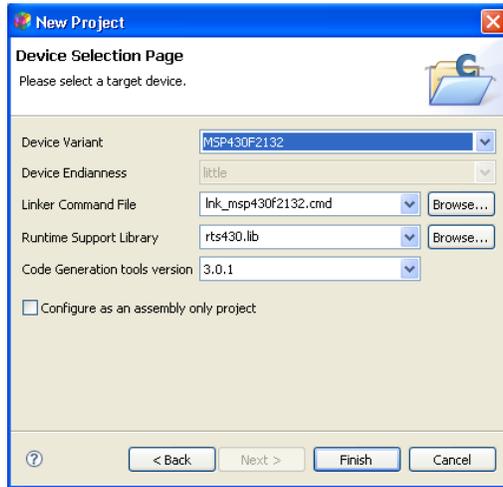


Figure 86: Project Setup

Upon clicking “Finish”, a new project is created and is shown C/C++ Projects Window. The source code must now be added to the project. In the case of the Sensor Unit, the source has the name of **Bridge_Scour_Transmitter_Source.c** while the Receiver Unit source has the name **Bridge_Scour_Receiver_Source.c**. To add the source code click the menu item **Project → Add Files to Active Project.**, then select the source code file to be used.

19.1.2 FET-Pro430

The software tool FET-Pro430 is a flash programmer made for the MSP430 devices. The programmer is made by Elprotronic Inc. and can be freely downloaded at <http://www.elprotronic.com/download.html>. Although Code Composer Essentials is also able to download program code to the microcontroller, this application simplifies the process once the system is in a production phase. This program has a simple GUI interface and can take a .hex output file (file created by compiling the source code in the Code Composer Essentials IDE) of the source code and download it to the flash memory of the microcontroller without any of the debugging features of an IDE like Code Composer Essentials.

Once the FET-Pro430 is installed on the machine, the executable can be run. The interface looks as shown in Figure 87 below. A couple settings must be established the first time the tool is run. First, the “Group” pull-down in the Microcontroller Type section of the GUI must be changed to **MSP430F2xx**. Under this, the **MSP430F-** pull-down list must be changed to “2132”. Then, under the “Power Device from Adapter” section, the pull-down must be changed to “3.0 V”. Lastly, a setting must be changed in the “Connection / Device Reset” menu. This menu can be reached by clicking on “Settings” at the top of the GUI and clicking on “Connection / Device Reset”. A window will appear as shown in Figure 88. Make sure the “JTAG” is selected in the “Communication with Target Device” section. Also, ensure “USB” is selected under the “COM” port section and click “OK”. Once these settings are established, the file to be downloaded must be selected. This can be done by clicking the “Open Source File” button near the top left of the interface. Once a “.hex” file has been selected, the

microcontroller can be programmed by clicking the “AUTOPROGRAM” button. The result should look as shown in Figure 87.

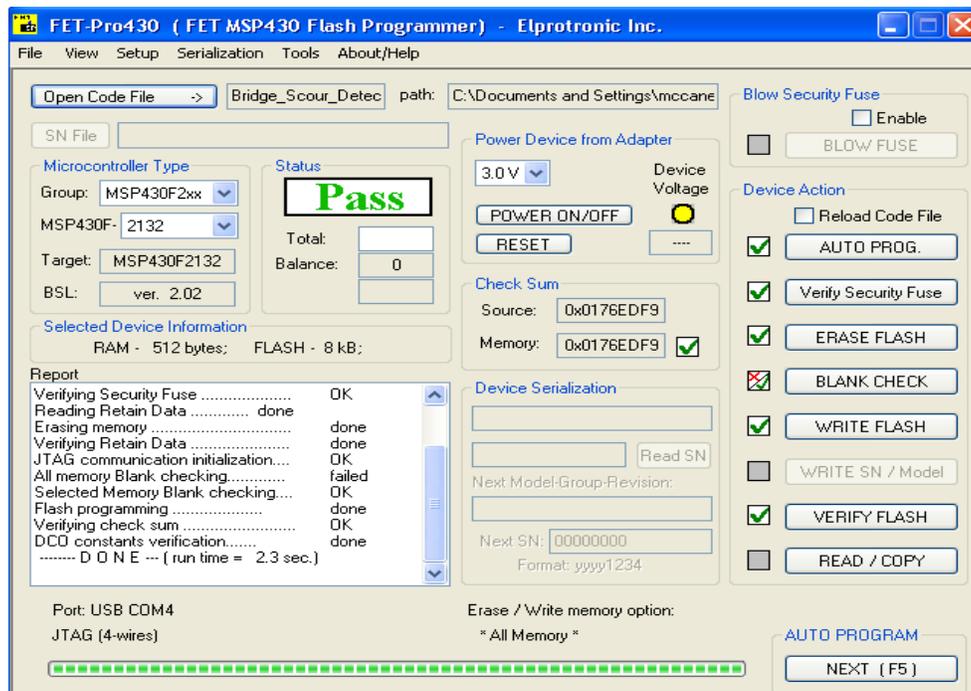


Figure 87: FET-Pro430 Interface

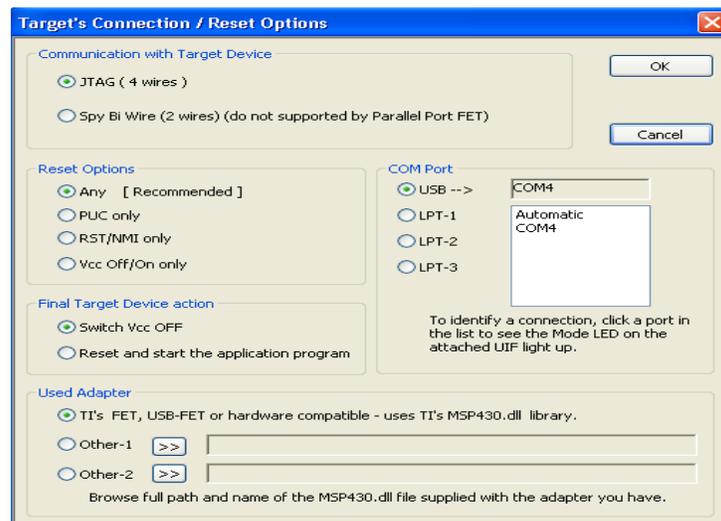


Figure 88: FET - Pro430 Setup

19.1.3 HyperTerminal

HyperTerminal™ is a utility that comes pre-installed on all Windows Operating Systems. It is a communications application. This system utilizes it for communication with the UART of the microcontroller. For convenience, a USB connection is used as an intermediary between the host computer and the UART of the microcontroller. A Serial to USB converter chip converts between the USB connection and the UART connection. Using the USB within the HyperTerminal™ requires installation of a Virtual Com Port

(VCP) made to interact with the Serial to USB chip. Using a VCP, the USB will show up within HyperTerminal™ as regular COM port. The VCP drivers can be found freely available at: <http://www.ftdichip.com/Drivers/VCP.htm>. Download the latest version of the VCP driver and install.

Once the VCP drivers are installed the HyperTerminal™ must be setup. The HyperTerminal™ is run by going to the “Start” button within the Windows Operating System, clicking on “Run”, and typing “hypertm”. This will bring up a prompt for the user to name the connection and to choose an icon to represent the connection as shown in Figure 89. Any name and icon will suffice, but “Bridge_Scour_Detection_UART” is recommended.



Figure 89: Hyper Terminal Startup

On the next prompt the COM port must be chosen. The COM port used must correspond to the VCP installed. To find this, the device manager within Windows must be utilized. Choose this COM port next to the “Connect Using” line while leaving all other settings as is. Lastly, a window with settings for that port will appear. Change the “Bits per second” pull-down menu to “19200” and the “Flow control” pull-down menu to “None” as shown in Figure 90. HyperTerminal™ is now ready to use.

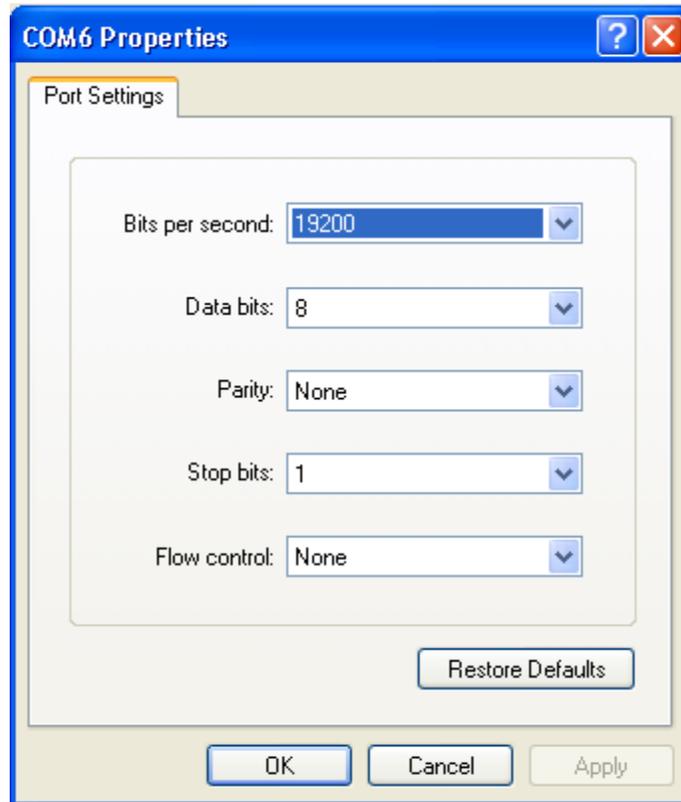


Figure 90: Com Settings

19.2 UART Operation and Protocol

Once HyperTerminal™ has been set up and hardware has been assembled and connected, and the microprocessors programmed, data can be transferred serially between the host PC and the microcontroller. However, a protocol is necessary to invoke the specific functionality desired by the technician. The two following sub-sections go over the protocol setup for interface with the Sensor Unit and Receiver Unit.

19.2.1 Sensor Unit UART Protocol

This section details the protocol used to interface with the Sensor Unit microcontroller. Below the available functions are listed along with how to invoke them.

19.2.1.1 Send Sensor Unit Data Block

The Sensor Unit Data Block is composed of 21 bytes of data (and two bytes for error detection). This is 168 bits of data. To transfer this data the data string will be

represented by 42 hexadecimal (HEX) characters (digits 0-9 and letters A-F). Two HEX characters are required to represent one byte of data. The first two HEX characters represent the first byte of the Sensor Unit Data Block. The next two HEX characters represent the second byte, and so forth. Once the 42 HEX characters have been formed, they can be transferred by two means.

3. Text (Txt) File Transfer
 - a. Place the 42 HEX characters preceded by two asterisks (**).
 - b. Save the file as a .txt file type
 - c. Within HyperTerminal™ click Transfer→Send Text File.
 - d. Choose the text (.txt) file containing the Sensor Unit Data Block
 - e. The file will be transferred and should result in a “data_txd” statement showing up in HyperTerminal™.
2. Character by Character Entry
 - a. Enter the asterisk character (“*”) two times in a row within the HyperTerminal™ interface. This notifies the microcontroller that the next 42 HEX characters are the Sensor Unit Data Block and should be stored as such.
 - b. Enter the Sensor Unit Data Block one HEX character at a time.

19.2.1.2 Sensor Unit Data Block Read Out

Verification of the Sensor Unit Data Block entered can be done simply by pressing entering “#” at any time during the interface. The data entered concatenated with the two bytes of CRC will be displayed. The data block will be displayed as a sequence of HEX characters. This is shown in Figure 80.

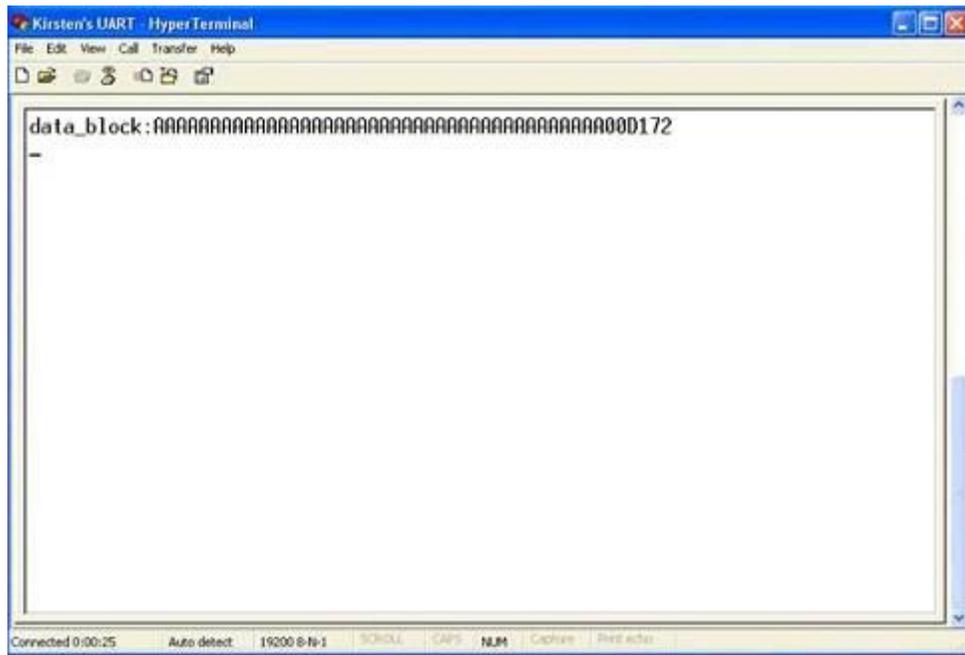


Figure 91: Sensor Unit Data Block Read Out

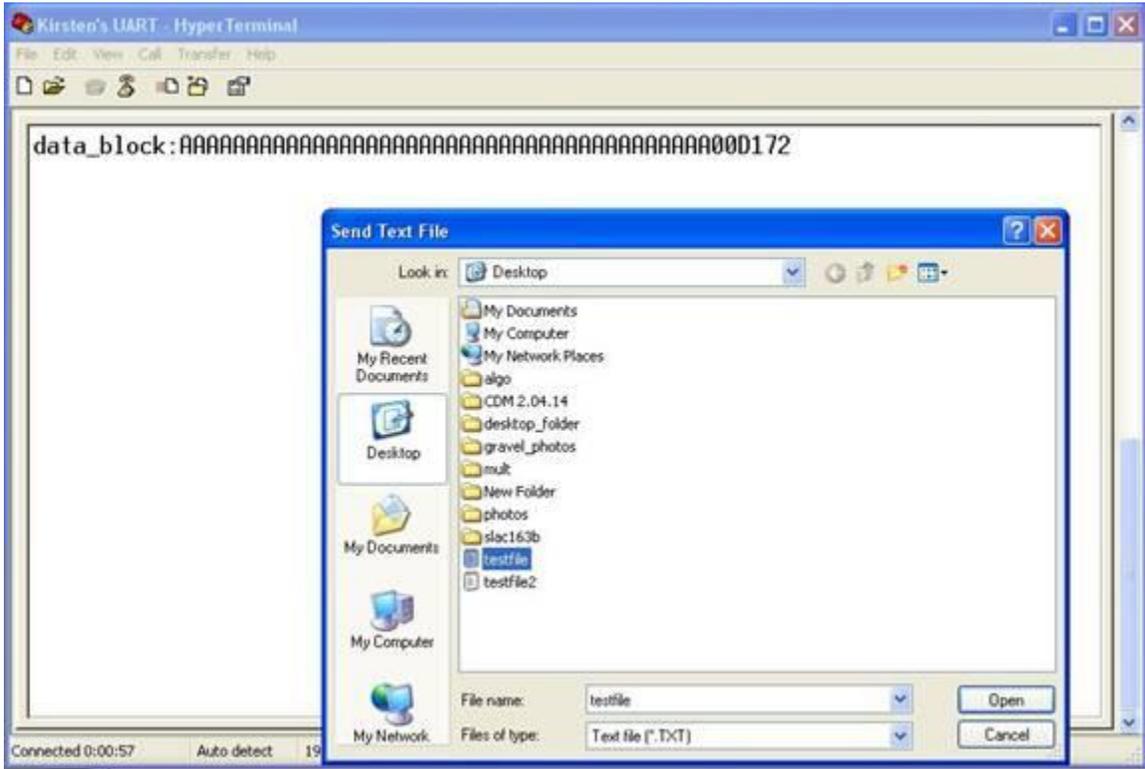


Figure 92: Transmitter UART Communication: Transfer of Data Block from txt file

19.2.2 Receiver Unit UART Protocol

This section describes the protocol used to interface with the Receiver Unit microcontroller. Below the available functions are listed along with how to invoke them.

19.2.2.1 Bridge ID Entry

The Receiver Unit has a stored Bridge ID to compare with the Bridge ID contained within a received Sensor Unit Data Block. The Bridge ID portion of the Sensor Unit Data Block is composed of 14 bytes of data. This is 112 bits of data. To transfer this Bridge ID to the microcontroller, the data string will be represented by 28 HEX characters (base 16, 0-9 A-F). The first two HEX characters represent the first byte of the Bridge ID; the next two HEX characters represent the second byte, and so forth. Once the 28 HEX characters have been formed, they can be transferred by two means.

3. Text File Transfer

- a. Place the 28 HEX characters preceded by two asterisks (**) at the beginning of file.
- b. Save the file as a text (.txt) file type (text file). A simple text editor such as Notepad or WordPad can be used (both are part of the Windows Operating System).
- c. Within HyperTerminal™ click Transfer→Send Text File. This is the same procedure as Step 1.c for the Sensor Unit serial interface.
- d. Choose the text (.txt) file containing the Bridge ID. This is the same procedure as Step 1.d for the Sensor Unit serial interface.

- e. The file will be transferred to the Receiver Unit and “data_txed” will be displayed in the HyperTerminal™ window. This is the same procedure as Step 1.e for the Sensor Unit serial interface.
4. Character by Character Entry
- a. Enter the asterisk character (*) two times in a row within the HyperTerminal™ interface. This notifies the microcontroller that the next 28 HEX characters are the Bridge ID and should be stored as such.
 - b. Enter the Bridge ID one HEX character at a time.

19.2.2.2 Bridge ID Read Out

Verification of the Bridge ID entered can be done simply by entering “#” followed by a “1” at any time during the interface. The Bridge ID will be displayed as a sequence of HEX characters.

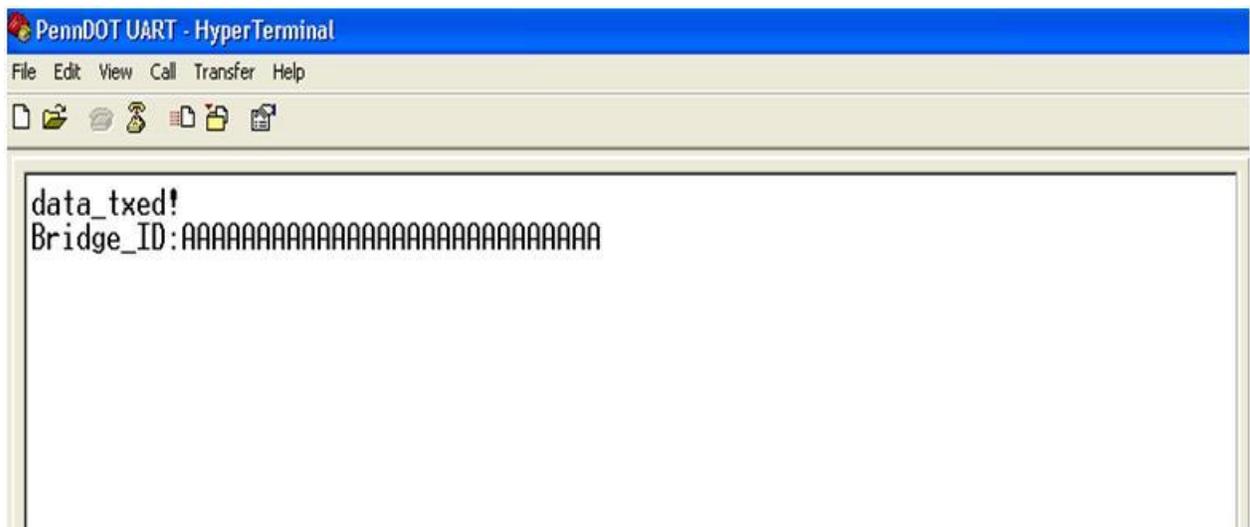


Figure 93: Bridge ID Transfer and Read Out

19.2.2.3 Received Sensor Unit Data Blocks Read Out

When a Sensor Unit Data Block is accepted by the Receiver Unit it is stored into the flash memory of the Receiver Unit microcontroller. Each block received is stored consecutively after the last. Entering “#” followed by a “2” at any time during the interface will result in a read out of all Sensor Unit data blocks received. Each block is placed on a new line with the first block printed out corresponding to the first block received by the Receiver Unit.

19.2.2.4 Receiver Unit Reset

The state of the Receiver Unit is based on the color code of the Sensor Unit Data Blocks received. To reset this state to the default starting state (no lights lit) two ampersand characters (&&) should be entered consecutively. A “Receiver Reset” statement will be displayed in the HyperTerminal™ window upon doing this.

20.0 System PCB Assemblies

There are three main components that must be assembled for the system. These components are the Sensor, Receiver, and Light Indicator Units. Each of these components requires that a printed circuit board or PCB have components added to their surface. Components are added to the printed circuit board through soldering. For the surface mount chips, surface mount machines can be used. However, all chips are sufficiently large in size that hand-soldering is also reliable, but not recommended. The .pcb file or printed circuit board file representing the printed circuit board of each unit provides a means to identify the correct orientation and placement of each component. Within ExpressPCB, each component and its pads can be clicked on so that information regarding the name of the component and the pin clicked will come up. The .pcb file directly corresponds to the .sch or schematic file such that the schematic file can also be used for help. Failure to follow the placement specified by these files will result in malfunction of the unit. The schematic and design (layout) files must be viewed using the ExpressPCB software which is freely available from ExpressPCB.

21.0 Sensor Unit Encapsulation and Installation

The encapsulation and installation of the Sensor Unit has several steps that must be followed. This section presents each step in detail.

21.1 Step 1 - Sensor Unit PCB Preparation

The Sensor Unit PCB must be assembled as described in Section 3.0. Once all components have been added to the circuit board, the board must be programmed. The most up-to-date Sensor Unit source code output file should be downloaded to the microcontroller using the MSP-FET430UIF connection and the FET –Pro430 utility described in section 2.2.1.2.

21.2 Sensor Unit PCB – Capsule Attachment

The printed circuit board must be attached to the inner face of the PVC cap. It is attached such that the length of the board is perpendicular to the inner face of the PVC cap. A slot 0.0634-inches wide by 1.5-inches long by 0.1-inches deep must be etched. This slot should be centered. Since the PVC inner cap face is slightly concave, the depth must be considered starting at the surface point furthest from the center of the cap. Once this slot has been machined, the bottom 0.1-inch of the PCB should be placed in the slot and junction adhesive should be applied.

21.3 Sensor Unit PCB Capsule Peripheral Attachment

The external reset switch enables the user to reset the Sensor Unit without opening the capsule. This switch allows the Sensor Unit to be tested immediately before it is installed and then reset. In addition, the reset switch enables the Sensor Unit to be reset in the event that it is mistakenly triggered during installation. The external switch must be durable and watertight.

Figure 95 illustrates the process of installing the reset switch in the Sensor Unit capsule.

- First, a hole and a recess must be machined in the PVC cap.

- Then a rubber gasket must be added with silicon sealant.
- On top of the gasket, a metal ring must be added using silicon sealant.
- The switch must then be placed in the hole and sealed on the inside and outside of the PVC cap using the silicon sealant. The wiring to the three leads must be attached prior to this step. Ensure that the switch part exposed on the inside of the cap is completely covered by sealant.
- A rubber boot is placed on top of the metal ring and sealed using silicon.
- Lastly, a 1-inch diameter PVC nipple is placed around the switch and sealed using silicon sealant. The threads and cap are sealed in the same manner as the main capsule. This is described in detail in the next section.

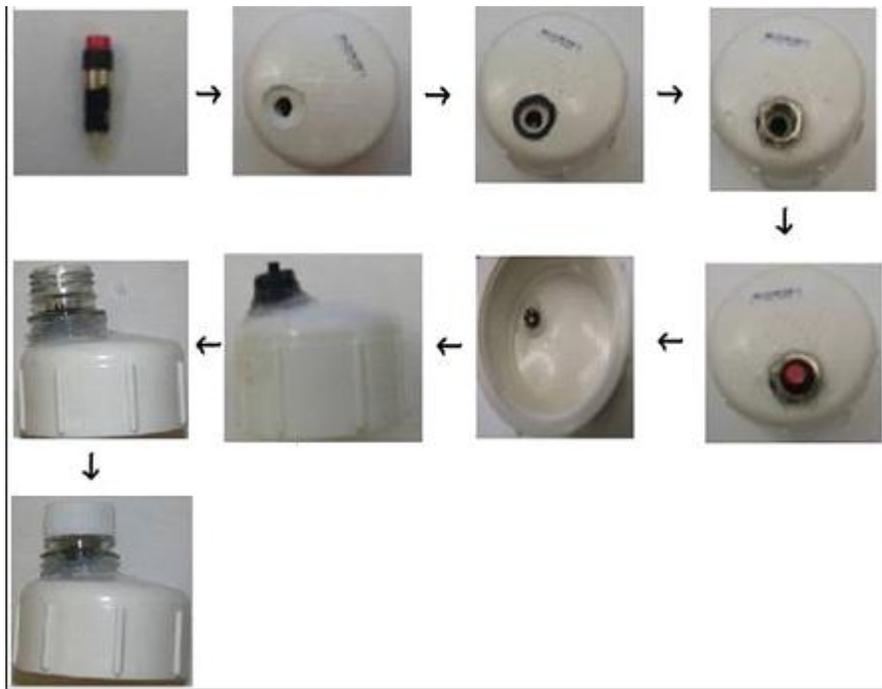


Figure 95: External Arm Switch Construction Steps

21.4 Sealing the Sensor Unit Capsule

The Sensor Unit Capsule must be watertight. Both the PVC Pipe and caps that comprise the capsule are threaded such that they can be screwed together. However, this alone is not sufficient to keep the inside of the capsule completely dry. To further seal the capsule Teflon tape should be wrapped around the threads on both ends of the PVC pipe prior to the caps being screwed on. Additionally, silicon sealant must be applied around of the junction of the cap and the pipe once they have been screwed together.

These steps, along with the placement of Silica Gel Desiccants packets inside the capsule, keep the inside of the capsule completely dry. Remember to maintain the verticality of the PCB while sealing the capsule. The cap connected to the PCB should remain the lowest point of the Sensor Unit with the length of the PCB, antenna, and PVC Pipe all such that they form a 90-degree angle with level ground.

The Sensor Unit should then be placed in a container of water large enough so that the unit is completely submerged. Signs of any leakage are observed as bubbles are being released from the Sensor Unit. If this is the case, immediately remove the Sensor Unit from the water. The Sensor Unit must then be unsealed and inspected. If undamaged, the Sensor Unit can be re-tested and re-sealed.

21.5 Sensor Unit Installation

The steps to properly deploy a properly assembled and programmed Sensor Unit are described here.

1. Drill a hole in the riverbed using a hollow stem auger. Drill about 6-inches deeper than the depth Sensor Unit is to be buried.
2. Slowly add the material to the hollow stem while slowly retracting the auger about 6-inches. This material should fill in the bottom of the hole and provide a base for the Sensor Unit. Compacting this base might help to fill the hole.
3. Ensure that the Sensor Unit is armed. The external switch push button should be locked down. It is important to keep the Sensor Unit in a vertical position to prevent it from being triggered.
4. Sensor Unit can be tested (by tilting it) to verify functionality and can be reset using the external arm switch. A test Receiver Unit is required to determine if the Sensor Unit is functioning (transmitting). Any Receiver Unit on the bridge requires inspection to determine if it was tripped by the test of the Sensor Unit. If so, the Receiver Unit must be reset. The USB interface described previously is used to reset the Receiver Unit.
5. Place the Sensor Unit in the hollow stem auger and push it down to the bottom using a push rod. The push rod may need to be attached to the Sensor Unit to keep it in a vertical orientation during the filling stage.
6. Slowly fill the auger with material while slowly retracting the auger until the entire hole is filled.
7. The Sensor Unit is now deployed.