Department of Computer Science and Engineering  
The University of Texas at Arlington

Team: Survivors

Project: Sight By Touch

Team Members:   
*Gerardo Guevara*

*Kevin Tran*

*Victor Rodriguez*

*Margaret Floeter*

*Henry Loh*

Last Updated: 2/28/2014

# Table of Contents

[Table of Contents 2](#_Toc386372165)

[Document Revision History 11](#_Toc386372166)

[List of Figures 12](#_Toc386372167)

[List of Tables 14](#_Toc386372168)

[1. Introduction 16](#_Toc386372169)

[1.2 Purpose and Use 16](#_Toc386372170)

[1.3 Product Concept 17](#_Toc386372171)

[1.4 Product Scope 20](#_Toc386372172)

[2. Architecture Overview 21](#_Toc386372173)

[2.1 Hardware Interface Layer 22](#_Toc386372174)

[2.1.1 System Control Module subsystem 22](#_Toc386372175)

[2.1.2 Sensor Module(s) subsystem 22](#_Toc386372176)

[2.2 Warning Layer 22](#_Toc386372177)

[2.2.1 Battery Level Monitor subsystem 22](#_Toc386372178)

[2.2.2 Detection Filter subsystem 22](#_Toc386372179)

[2.3 Notification Layer 23](#_Toc386372180)

[2.3.1 On Notification subsystem 23](#_Toc386372181)

[2.3.2 Object Notification subsystem 23](#_Toc386372182)

[2.3.3 Battery Notification subsystem 23](#_Toc386372183)

[2.4 Module Decomposition 23](#_Toc386372184)

[2.4.1 Power 26](#_Toc386372185)

[2.4.2 Range Setting 26](#_Toc386372186)

[2.4.3 Detection 26](#_Toc386372187)

[2.4.4 Routing 26](#_Toc386372188)

[2.4.5 Range Adjuster 26](#_Toc386372189)

[2.4.6 Distance Analyzer 26](#_Toc386372190)

[2.4.7 Voltage Reader 26](#_Toc386372191)

[2.4.8 Battery Analyzer 26](#_Toc386372192)

[2.4.9 On Pattern 27](#_Toc386372193)

[2.4.10 Object Pattern 27](#_Toc386372194)

[2.4.11 Low Battery Pattern 27](#_Toc386372195)

[2.4.12 Critical Battery Pattern 27](#_Toc386372196)

[3. System Hardware Description 28](#_Toc386372197)

[3.1 Arduino Mega 2560 28](#_Toc386372198)

[3.1.1 Quantity 28](#_Toc386372199)

[3.1.2 Purpose 28](#_Toc386372200)

[3.1.3 Specifications 29](#_Toc386372201)

[3.1.4 Interfaces 29](#_Toc386372202)

[3.2 HC-SR04 Ultrasonic Sensor 30](#_Toc386372203)

[3.2.1 Quantity 30](#_Toc386372204)

[3.2.2 Purpose 30](#_Toc386372205)

[3.2.3 Specifications 31](#_Toc386372206)

[3.2.4 Interfaces 31](#_Toc386372207)

[3.3 SPST High-Current Mini Toggle Switch 32](#_Toc386372208)

[3.3.1 Quantity 32](#_Toc386372209)

[3.3.2 Purpose 32](#_Toc386372210)

[3.3.3 Specifications 32](#_Toc386372211)

[3.3.4 Interfaces 32](#_Toc386372212)

[3.4 XBOX 360 Vibration Motor 33](#_Toc386372213)

[3.4.1 Quantity 33](#_Toc386372214)

[3.4.2 Purpose 33](#_Toc386372215)

[3.4.3 Specifications 34](#_Toc386372216)

[3.4.4 Interfaces 34](#_Toc386372217)

[3.5 Flat Ribbon Cable Roll 34](#_Toc386372218)

[3.5.1 Quantity 34](#_Toc386372219)

[3.5.2 Purpose 35](#_Toc386372220)

[3.5.3 Specifications 35](#_Toc386372221)

[3.5.4 Interfaces 35](#_Toc386372222)

[3.6 Ribbon Crimp Connector 35](#_Toc386372223)

[3.6.1 Quantity 36](#_Toc386372224)

[3.6.2 Purpose 36](#_Toc386372225)

[3.6.3 Specifications 36](#_Toc386372226)

[3.6.4 Interfaces 37](#_Toc386372227)

[3.7 Jumper Wires (Male to Male and Male to Female) 37](#_Toc386372228)

[3.7.1 Quantity 38](#_Toc386372229)

[3.7.2 Purpose 38](#_Toc386372230)

[3.7.3 Specifications 38](#_Toc386372231)

[3.7.4 Interfaces 38](#_Toc386372232)

[3.8 Push Switch 38](#_Toc386372233)

[3.8.1 Quantity 39](#_Toc386372234)

[3.8.2 Purpose 39](#_Toc386372235)

[3.8.3 Specifications 39](#_Toc386372236)

[3.8.4 Interfaces 39](#_Toc386372237)

[3.9 Resistor 39](#_Toc386372238)

[3.9.1 Quantity 40](#_Toc386372239)

[3.9.2 Purpose 40](#_Toc386372240)

[3.9.3 Specifications 40](#_Toc386372241)

[3.9.4 Interfaces 41](#_Toc386372242)

[3.10 Solderless Plug-in BreadBoard 41](#_Toc386372243)

[3.10.1 Quantity 41](#_Toc386372244)

[3.10.2 Purpose 41](#_Toc386372245)

[3.10.3 Specifications 41](#_Toc386372246)

[3.10.4 Interfaces 42](#_Toc386372247)

[3.11 Rechargeable Battery Pack w/ Charger 42](#_Toc386372248)

[3.11.1 Quantity 42](#_Toc386372249)

[3.11.2 Purpose 42](#_Toc386372250)

[3.11.3 Specifications 42](#_Toc386372251)

[3.11.4 Interfaces 43](#_Toc386372252)

[3.12 Other 43](#_Toc386372253)

[3.12.1 Computer 43](#_Toc386372254)

[3.12.2 Arduino USB Cable 43](#_Toc386372255)

[3.12.3 Custom Encasing 43](#_Toc386372256)

[3.12.4 Body Suit 43](#_Toc386372257)

[3.13 Conceptual Hardware Diagrams 43](#_Toc386372258)

[3.13.1 Voltage Divider Circuit 43](#_Toc386372259)

[3.13.2 Power Supply Connection 44](#_Toc386372260)

[3.12.3 Sensor Module Internal Connection 46](#_Toc386372261)

[4. System Software Description 47](#_Toc386372262)

[4.1 Data Description 47](#_Toc386372263)

[4.1.1 Global Variables 47](#_Toc386372264)

[4.2 System Notification Priorities 47](#_Toc386372265)

[5. Hardware Interface Layer: System Control Module 48](#_Toc386372266)

[5.1 Power 48](#_Toc386372267)

[5.1.1 Prologue 48](#_Toc386372268)

[5.1.2 Interfaces 49](#_Toc386372269)

[5.1.3 External Data Dependencies 49](#_Toc386372270)

[5.1.4 Internal Data Descriptors 49](#_Toc386372271)

[5.1.5 Service Dependencies 49](#_Toc386372272)

[5.1.6 Module Processing (Pseudo-code Algorithm) 50](#_Toc386372273)

[5.2 Range Setting 50](#_Toc386372274)

[5.2.1 Prologue 50](#_Toc386372275)

[5.2.2 Interfaces 51](#_Toc386372276)

[5.2.3 External Data Dependencies 51](#_Toc386372277)

[5.2.4 Internal Data Descriptors 52](#_Toc386372278)

[5.2.5 Service Dependencies 52](#_Toc386372279)

[5.2.6 Module Processing (Pseudo-code Algorithm) 52](#_Toc386372280)

[6. Hardware Interface Layer: Sensor Module(s) 53](#_Toc386372281)

[6.1 Detection 53](#_Toc386372282)

[6.1.1 Prologue 53](#_Toc386372283)

[6.1.2 Interfaces 54](#_Toc386372284)

[6.1.3 External Data Dependencies 54](#_Toc386372285)

[6.1.4 Internal Data Descriptors 55](#_Toc386372286)

[6.1.5 Service Dependencies 55](#_Toc386372287)

[6.1.6 Module Processing (Pseudo-code Algorithm) 56](#_Toc386372288)

[6.2 Routing 57](#_Toc386372289)

[6.2.1 Prologue 57](#_Toc386372290)

[6.2.2 Interfaces 57](#_Toc386372291)

[6.2.3 External Data Dependencies 58](#_Toc386372292)

[6.2.4 Internal Data Descriptors 58](#_Toc386372293)

[6.2.5 Service Dependencies 58](#_Toc386372294)

[6.2.6 Module Processing (Pseudo-code Algorithm) 58](#_Toc386372295)

[7. Warning Layer: Detection Filter 59](#_Toc386372296)

[7.1 Range Adjuster 59](#_Toc386372297)

[7.1.1 Prologue 59](#_Toc386372298)

[7.1.2 Interfaces 60](#_Toc386372299)

[7.1.3 External Data Dependencies 60](#_Toc386372300)

[7.1.4 Internal Data Descriptors 60](#_Toc386372301)

[7.1.5 Service Dependencies 60](#_Toc386372302)

[7.1.6 Module Processing (Pseudo-code Algorithm) 60](#_Toc386372303)

[7.2 Distance Analyzer 61](#_Toc386372304)

[7.2.1 Prologue 61](#_Toc386372305)

[7.2.2 Interfaces 61](#_Toc386372306)

[7.2.3 External Data Dependencies 61](#_Toc386372307)

[7.2.4 Internal Data Descriptors 62](#_Toc386372308)

[7.2.5 Service Dependencies 62](#_Toc386372309)

[7.2.6 Module Processing (Pseudo-code Algorithm) 62](#_Toc386372310)

[8. Warning Layer: Battery Level Monitor 63](#_Toc386372311)

[8.1 Voltage Reader 63](#_Toc386372312)

[8.1.1 Prologue 63](#_Toc386372313)

[8.1.2 Interfaces 64](#_Toc386372314)

[8.1.3 External Data Dependencies 64](#_Toc386372315)

[8.1.4 Internal Data Descriptors 64](#_Toc386372316)

[8.1.5 Service Dependencies 64](#_Toc386372317)

[8.1.6 Module Processing (Pseudo-code Algorithm) 65](#_Toc386372318)

[8.2 Battery Analyzer 65](#_Toc386372319)

[8.3.1 Prologue 65](#_Toc386372320)

[8.3.2 Interfaces 66](#_Toc386372321)

[8.3.3 External Data Dependencies 66](#_Toc386372322)

[8.3.4 Internal Data Descriptors 66](#_Toc386372323)

[8.3.5 Service Dependencies 66](#_Toc386372324)

[8.3.6 Module Processing (Pseudo-code Algorithm) 67](#_Toc386372325)

[9. Notification Layer: On Notification 68](#_Toc386372326)

[9.1 On Pattern 68](#_Toc386372327)

[9.1.1 Prologue 68](#_Toc386372328)

[9.1.2 Interfaces 69](#_Toc386372329)

[9.1.3 External Data Dependencies 69](#_Toc386372330)

[9.1.4 Internal Data Descriptors 69](#_Toc386372331)

[9.1.6 System Dependencies 69](#_Toc386372332)

[9.1.7 Module Processing (Pseudo-code Algorithm) 69](#_Toc386372333)

[10. Notification Layer: Object Notification 70](#_Toc386372334)

[10.1 Object Pattern 70](#_Toc386372335)

[10.1.1 Prologue 70](#_Toc386372336)

[10.1.2 Interfaces 70](#_Toc386372337)

[10.1.3 External Data Dependencies 71](#_Toc386372338)

[10.1.4 Internal Data Descriptors 71](#_Toc386372339)

[10.1.5 Service Dependencies 71](#_Toc386372340)

[10.1.6 Module Processing (Pseudo-code Algorithm) 71](#_Toc386372341)

[11. Notification Layer: Battery Notification 73](#_Toc386372342)

[11.1 Low Battery Pattern 73](#_Toc386372343)

[11.1.1 Prologue 73](#_Toc386372344)

[11.1.2 Interfaces 73](#_Toc386372345)

[11.1.3 External Data Dependencies 74](#_Toc386372346)

[11.1.4 Internal Data Descriptors 74](#_Toc386372347)

[11.1.5 Service Dependencies 74](#_Toc386372348)

[11.1.6 Module Processing (Pseudo-code Algorithm) 74](#_Toc386372349)

[11.2 Critical Battery Pattern 74](#_Toc386372350)

[11.2.1 Prologue 75](#_Toc386372351)

[11.2.2 Interfaces 75](#_Toc386372352)

[11.1.3 External Data Dependencies 75](#_Toc386372353)

[11.1.4 Internal Data Descriptors 75](#_Toc386372354)

[11.1.5 Service Dependencies 75](#_Toc386372355)

[11.1.6 Module Processing (Pseudo-code Algorithm) 76](#_Toc386372356)

[12. Quality Assurance 77](#_Toc386372357)

[12.1 Test Plan and Procedures 77](#_Toc386372358)

[12.2 Module/Unit Testing 77](#_Toc386372359)

[12.2.1 Hardware Interface Layer 77](#_Toc386372360)

[12.2.2 Warning Layer 78](#_Toc386372361)

[12.2.3 Notification Layer 78](#_Toc386372362)

[12.3 Component Testing 79](#_Toc386372363)

[12.3.1 Distance Sensor(s) 79](#_Toc386372364)

[12.3.2 Vibration Motor(s) 79](#_Toc386372365)

[12.3.3 System Control Module 79](#_Toc386372366)

[12.3.4 Arduino Mega 2560 79](#_Toc386372367)

[12.4 Integration Testing 79](#_Toc386372368)

[12.4.1 Hardware Interface Layer 79](#_Toc386372369)

[12.4.2 Warning Layer 80](#_Toc386372370)

[12.4.3 Notification Layer 80](#_Toc386372371)

[12.5 System Verification Testing 80](#_Toc386372372)

[12.6 Test Cases 80](#_Toc386372373)

[13. Requirements Traceability 81](#_Toc386372374)

[13.1 Purpose 81](#_Toc386372375)

[13.2 Requirements Traceability Matrix By Module 82](#_Toc386372376)

[13.3 Hardware Interface Layer Module Mapping 82](#_Toc386372377)

[13.4 Warning Layer Module Mapping 83](#_Toc386372378)

[13.5 Notification Layer Module Mapping 84](#_Toc386372379)

[13.6 Analysis of Requirements Mapping 85](#_Toc386372380)

[14.1 Packaging and Installation 86](#_Toc386372381)

[14.2 Acceptance Testing 87](#_Toc386372382)

[14.3 Acceptance Criteria 87](#_Toc386372383)

[14.3.1 The System Shall Be Intuitive and Accessible. 87](#_Toc386372384)

[14.3.2 The System Shall Help Visually Impaired Individuals Traverse in both Indoor and Outdoor Environments 87](#_Toc386372385)

[14.3.3 The System Shall Notify The User That The Battery Is Low 88](#_Toc386372386)

[14.3.4 The System Shall Be Safe To Wear 88](#_Toc386372387)

[Appendix A: Arduino Libraries 89](#_Toc386372388)

[Standard Libraries 89](#_Toc386372389)

[Contributed Libraries 90](#_Toc386372390)

[**Sensing:** 90](#_Toc386372391)

[**Timing:** 90](#_Toc386372392)

[Formulas 90](#_Toc386372393)

[Converting microseconds from HC-SR04 Ultrasonic Sensor to inches 90](#_Toc386372394)

# Document Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Revision Number | Revision Date | Description | Rationale |
| 1.0 | 2/11/2014 | Initial DDS Draft ready for review | Initial and Review Draft |
| 2.0 | 2/28/2014 | Baseline DDS | Design changed based on feedback |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# List of Figures

Figure # Title Page #

1-1 Sensors on the Sight By Touch System shown around the body 18

1-2 Closer look of a Sensor Module 19

1-3 Sketch of the System Control Module 19

2-1 Architecture Design Diagram 21

2-2 Module Decomposition Chart 24

3-1 Arduino Mega 2560 28

3-2 HC-SR04 Ultrasonic Sensor 30

3-3 SPST High-Current Mini Toggle Switch 32

3-4 DC Vibration Motor 33

3-5 Flat Ribbon Cable Roll 34

3-6 Ribbon Crimp Connector 36

3-7 Push Switch 37

3-8 Resistor 38

3-9 Solderless Plug-in BreadBoard 39

3-10 Rechargeable Battery Pack w/ Charger Cable 40

3-11 Simple Voltage Divider Circuit 42

3-12 Conceptual Power Supply Diagram 43

3-13 Conceptual Sensor Module Diagram 44

5-1 Power Module Diagram 46

5-2 Range Setting Module Diagram 48

6-1 Detection Module Diagram 51

6-2 Routing Module Diagram 55

7-1 Range Adjuster Module Diagram 57

7-2 Distance Analyzer Module Diagram 59

8-1 Voltage Reader Module Diagram 61

8-2 Conversion Module Diagram 63

8-3 Battery Analyzer Module Diagram 64

9-1 On Pattern Module Diagram 67

10-1 Intensity Level Module Diagram 69

10-2 Object Pattern Module Diagram 70

11-1 Low Battery Patten Module Diagram 73

11-2 Critical Battery Pattern Module Diagram 74

# List of Tables

Table # Title Page #

2-1 Module Decompositions Data Flows Table 25

3-1 Arduino Mega 2560 Specifications 29

3-2 HC-SR04 Ultrasonic Sensor Specifications 31

3-3 SPST High-Current Mini Toggle Switch Specifications 32

3-4 DC Vibration Motor Specifications 34

3-5 Flat Ribbon Cable Roll Specifications 35

3-6 Ribbon Crimp Connector Specifications 37

3-7 Push Switch Specifications 38

3-8 Resistor Specifications 39

3-9 Solderless Plug-in BreadBoard Specifications 40

3-10 Rechargeable Battery Pack Specifications 41

5-1 Power Module Interfaces 47

5-2 Power Module External Data Dependencies 47

5-3 Range Setting Module Interfaces 49

5-4 Range Setting Module External Data Dependencies 49

5-5 Range Setting Module Internal Data Descriptors 50

6-1 Detection Module Interfaces 52

6-2 Detection Module External Data Dependencies 52

6-3 Detection Module Internal Data Descriptors 53

6-4 Routing Module Interfaces 55

6-5 Routing Module Internal Data Descriptors 56

7-1 Range Adjuster Module Interfaces 58

7-2 Range Adjuster Module Internal Data Descriptors 58

7-3 Distance Analyzer Module Interfaces 59

7-4 Distance Analyzer Module Internal Data Descriptors 60

8-1 Voltage Reader Module Interfaces 61

8-2 Voltage Reader Module External Data Dependencies 62

8-3 Voltage Reader Module Internal Data Descriptors 62

8-4 Conversion Module Interfaces 63

8-5 Conversion Module Internal Data Descriptors 64

8-6 Battery Analyzer Module Interfaces 65

8-7 Battery Analyzer Module Internal Data Descriptors 65

9-1 On Patten Module Interfaces 68

9-2 On Pattern Module Internal Data Descriptors 68

10-1 Intensity Level Module Interfaces 69

10-2 Intensity Level Module Internal Data Descriptors 70

10-3 Object Pattern Module Interfaces 71

10-4 Object Pattern Module Internal Data Descriptors 71

11-1 Low Battery Pattern Module Interfaces 73

11-2 Low Battery Pattern Module Internal Data Descriptors 74

11-3 Critical Battery Pattern Module Interfaces 75

11-4 Critical Battery Pattern Module Internal Data Descriptors 75

12-1 Test Cases 81

13-1 Requirements Traceability Matrix by Module 82

13-2 Hardware Interface Layer Module Mapping 83

13-3 Warning Layer Module Mapping 84

13-4 Notification Layer Module Mapping 85

# 1. Introduction

* 1. **Document Overview**

The Detailed Design Specification document is intended to break down in detail each module from each subsystem that was discussed in the Architecture Design Specification document. Included in this document are the architecture overview, component and module designs, quality assurance, requirements traceability matrix, acceptance plan, and additional definitions and formulas in the appendices.

## Purpose and Use

The purpose of the Sight By Touch System is to aid visually impaired individuals since they are unable to rely on their sight. More often than not, visually impaired individuals require some form of aid to help guide them, usually through the use of a cane or a service animal. Our team wishes to remove, or decrease, this dependency by allowing the user to be guided by vibrations from the system. In this way, the user’s hands will be free from having to hold a cane or a leash. With this system, a user shall be able to avoid collision with obstacles that are found in their environment. Our team has decided to make the system an indoor and outdoor system. The system shall lead the user in a safe direction by warning them when an object is within the detection range of the sensors through the use of vibrations from its vibration motors. When the sensors sense an object, the vibration motor closest to that object will vibrate. This system is used solely for the purpose of helping visually impaired users navigate in unknown environments. This product is not intended to be used in a crowded area because there would be too much interference from the surrounding objects.

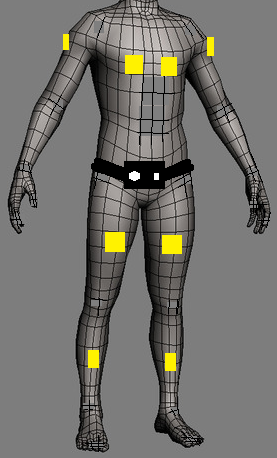
## Product Concept

The Sight By Touch System shall consist of an external belt that holds the main battery, which connects to the sensors and vibration motors. When the user is not using the product and wishes to charge the battery, it will be removed from the system so it can be charged.

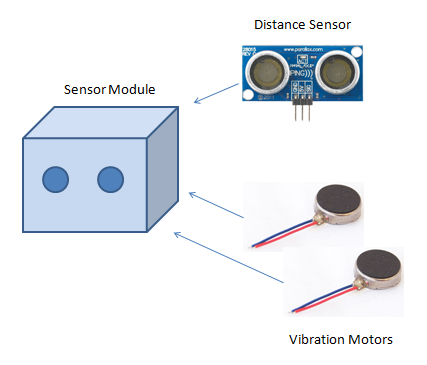
The system will be a full-blown suit with the sensors underneath the clothes and with their wires connected to the microcontroller. The sensors will be detachable to allow for flexibility and cleanliness. The system will focus on the front of the user (180 degrees, at least a 4 foot radius horizontally, and from the neck to the ankles vertically).

There will be multiple vibration motors distributed along the system, embedded in the suit. When the sensor connected to those vibration motors senses an object within the range, the sensor will send a signal to the microcontroller, which will then forward the message to the vibration motor. In order for the user to have a better idea where a detected object is, multiple sensors will detect it and multiply vibration motors will vibrate at different intensities depending on how far the object is. The closer the object is the more intense the vibration will be. In addition, depending on the location of the object being detected, the vibration motors closest to that object will vibrate the most.

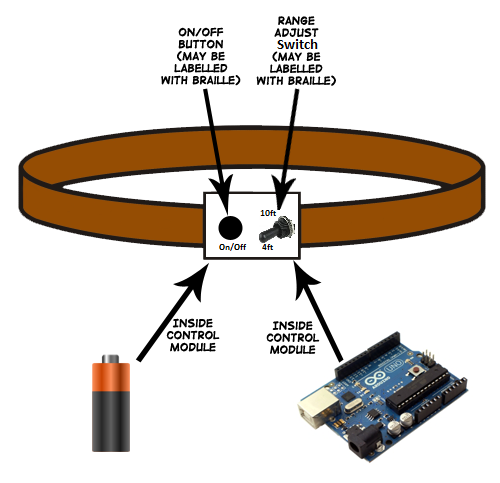
The belt will allow the user to interface with the device. The interface will have an on/off button to turn the sensors on/off. In addition, there will be a switch that allows the user to adjust the range that the sensors will detect. There will be no external elements for all the functions and data manipulation will be done internally.



**Figure 1-1: Sensors on the Sight By Touch System shown around the body**



**Figure 1-2: Closer look of a Sensor Module**



**Figure 1-3: Sketch of the System Control Module**

## Product Scope

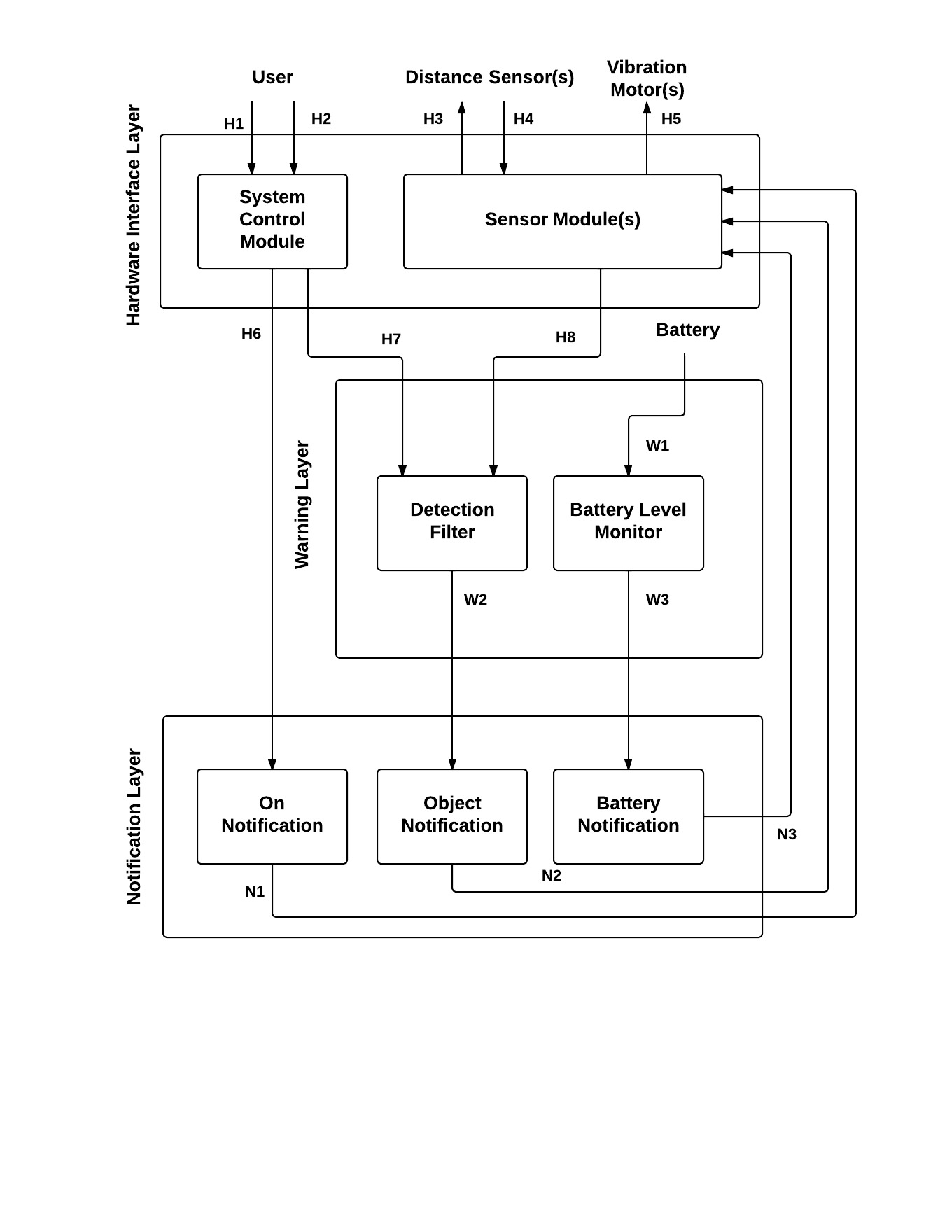
The Sight By Touch System is made to aid visually impaired users through the use of vibrations. Users of the Sight By Touch System will be able to move around and gauge where they are. When a user walks near an object, the system will vibrate in the direction where the object is closest. This warns the user that there is an object nearby and thus prevents the user from colliding with the object. With the use of these vibrations created by the system, a visually impaired user will be able to avoid obstacles. The system will not be able to detect the absence of flooring such as a hole or a staircase.

The intended audiences for the Sight By Touch System are individuals who are visually impaired. We consider the visually impaired to be based on the following metrics provided by the American Optometric Association:

* 20/70 to 20/160 is considered moderate visual impairment, or **moderate low vision**
* 20/200 to 20/400 is considered severe visual impairment, or **severe low vision**
* 20/500 to 20/1,000 is considered profound visual impairment, or **profound low vision**
* less than 20/1,000 is considered near-total visual impairment, or **near total blindness**
* no light perception is considered total visual impairment, or **total blindness**

# ****2. Architecture Overview****

This section reviews the Sight By Touch System architecture, which is the overall structure of how the system will be built, and the decomposition of the system into modules. The Sight By Touch System architecture consists of three main layers. Each layer contains subsystems that will carry out the functions their corresponding layer is responsible for. Figure 2-1 is a visual representation of the architecture including the layers, subsystems and data flows.



**Figure 2-1: Architecture Design Diagram**

## 2.1 Hardware Interface Layer

This layer is responsible for handling input to the system and output from the system by providing an interface between the system and any external inputs and outputs. This layer will consist of the system control module and the sensing modules. This layer will be standardizing the input from the user and distance sensors as well as the output to the vibration motors. The following are the subsystems of the Hardware Interface Layer.

### 2.1.1 System Control Module subsystem

The System Control Module will be responsible for handling the input received from the user and initializing the system. This subsystem will allow the user to turn the system on/off as well as adjust the maximum detection range of the distance sensors.

### 2.1.2 Sensor Module(s) subsystem

The Sensor Module(s) will be triggering the distance sensor(s) and handling the input received from them. This subsystem will also relay messages from the system to the vibration motor(s).

## 2.2 Warning Layer

This layer is responsible for processing all the input and output signals. It will be responsible for making decisions based on the input received from the Hardware Interface Layer and trigger notification messages to the Notification Layer. The following are the subsystems of the Warning Layer.

### 2.2.1 Battery Level Monitor subsystem

TheBattery Level Monitor will be responsible for reading and determining when the battery level is low or about to run out of power. This will help in notifying the user when the battery has reached an insufficient charge level.

### 2.2.2 Detection Filter subsystem

The Detection Filter will determine if the information received from the Sensor Module(s) is within the current maximum detection range.

## 2.3 Notification Layer

This layer is responsible for generating notification messages based on data sent from the Warning Layer and sending them to the Sensor Module(s) subsystem. The types of notifications are the On Notification, the Objected Detected Notification, the Low Battery Notification, and the Critical Battery Notification. The following are the subsystems of the Notification Layer.

### 2.3.1 On Notification subsystem

The On Notification subsystem will receive messages from the System Control Module subsystem and interpret them into appropriate instructions to relay to the Sensor Module(s) for an On Notification.

### 2.3.2 Object Notification subsystem

The Object Notification subsystem will receive messages from the Detection Filter subsystem whenever the system has detected a significant object and interpret them into appropriate instructions to relay to the Sensor Module(s) for an Object Detect Notification.

### 2.3.3 Battery Notification subsystem

The Battery Notification subsystem will receive a message from the Battery Level Monitor subsystem whenever the system’s battery is running low or is about to run out of power and interpret them into appropriate instructions to relay to the Sensor Module(s) for a Low Battery Notification or Critical Battery Notification.

## 2.4 Module Decomposition

This subsection decomposes the subsystems described earlier in this section into modules. These modules will be used to describe the detailed design of the Sight By Touch System. Figure 2-2 is a visual representation of the module decomposition of the system. Each module will be described briefly here. More detailed descriptions will be given in the following sections of this detailed document.

Module Decomposition Chart revised.emf

**Figure 2-2: Module Decomposition Chart**

NOTE: The data flow numbering in Figure 2-2 is different compared to that of Figure 2-1 due to the further decomposition into modules. This resulted in more data flows needed to fully describe the detailed design. A table on the next page will briefly describe these data flows. A more detailed description of data flowing between modules will be discussed in later sections.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Data Element*** |  | ***Descriptions*** | ***Source*** | ***Sink*** |
| ***H1.*** | **User physically presses the button of the system control module to power the system on or off** | | **User** | **Power** |
| ***H2.*** | **User physically turns the knob of the system control module to adjust the sensing range** | | **User** | **Range Setting** |
| ***H3.*** | **A voltage signal to the ultrasonic sensor to operate** | | **Detection** | **Distance Sensor(s)** |
| ***H4.*** | **A voltage signal from the Distance Sensors(s) signifying an object has been detected** | | **Distance Sensor(s)** | **Detection** |
| ***H5.*** | **A voltage signal to the vibration motors to operate** | | **Routing** | **Vibration Motor(s)** |
| ***H6.*** | **A call to the Range Setting module to update the maximum detection range** | | **Power** | **Range Setting** |
| ***H7.*** | **A call to the On Notification module to send an On Notification to the Sensor Modules** | | **Power** | **On Pattern** |
| ***H8.*** | **The position of the toggle switch as a value** | | **Range Setting** | **Range Adjuster** |
| ***H9.*** | **The ID of the sensor that operated and the distance to an object from the sensor** | | **Detection** | **Distance Analyzer** |
| ***W1.*** | **The ID of the sensor that detected an object and the distance to that object from the sensor** | | **Distance Analyzer** | **Object Pattern** |
| ***W2.*** | **The voltage from the battery powering the system** | | **Battery** | **Voltage Reader** |
| ***W3.*** | **The current voltage provided by the battery as a percentage of its full voltage capacity** | | **Voltage Reader** | **Battery Analyzer** |
| ***W4.*** | **A call to the Low Battery Pattern module to send a Low Battery Notification to the Sensor Modules** | | **Battery Analyzer** | **Low Battery Pattern** |
| ***W5.*** | **A call to the Critical Battery Pattern module to send a Critical Battery Notification to the Sensor Modules** | | **Battery Analyzer** | **Critical Battery Pattern** |
| ***N1.*** | **A unique vibration pattern, a unique delay pattern and the vibration motor id representing all vibration motors for an On Notification** | | **On Pattern** | **Routing** |
| ***N2.*** | **A unique vibration pattern, a unique delay pattern and the vibration motor id associated with the sensor that detected an object for an Object Detected Notification** | | **Object Pattern** | **Routing** |
| ***N3.*** | **A unique vibration pattern, a unique delay pattern and the vibration motor id representing all vibration motors for a Low Battery Notification** | | **Low Battery Pattern** | **Routing** |
| ***N4.*** | **A unique vibration pattern, a unique delay pattern and the vibration motor id representing all vibration motors for a Critical Battery Notification** | | **Critical Battery Pattern** | **Routing** |

**Table 2-1: Module Decomposition Data Flows Table**

### 2.4.1 Power

Power is responsible for handling the use case in which the user is turning the system on or off and initializing the system.

### 2.4.2 Range Setting

Range Setting is responsible for handling the use case in which the toggle switch’s position changes, resulting in a new maximum detection range for the system to use.

### 2.4.3 Detection

Detection focuses on detecting objects, determining distance to them from the system and passing distance and the id of the sensor that detected an object to Distance Analyzer for analysis.

### 2.4.4 Routing

Routing focuses on interpreting the notification data (vibration motor id, vibration pattern, and delay pattern) sent from the Notification Layer and operating the correct vibration motor(s) using the vibration motor id with a specific intensity using the vibration pattern and a specific duration using the delay pattern to provide haptic feedback to the user.

### 2.4.5 Range Adjuster

Range Adjuster focuses on updating the maximum detection range used by Distance Analyzer to make filter decisions on using the new switch position sent from Range Setting.

### 2.4.6 Distance Analyzer

Distance Analyzer focuses on making decisions on whether an object’s detected distance is within the current detection range of the system.

### 2.4.7 Voltage Reader

Voltage Reader focuses on acquiring a voltage reading from the battery powering the system and converting it into a percentage of the full voltage capacity that can be analyzed.

### 2.4.8 Battery Analyzer

Battery Analyzer focuses on making decisions on whether the voltage percentage created from Voltage Reader represents a low battery level indicating the system will run out of power soon or a critical battery level indicating the system is about to run out of power.

### 2.4.9 On Pattern

On Pattern is responsible for sending the vibration pattern and the delay pattern that will be used to notify the user that the system is on along with an id representing all vibration motors to Routing to give actual feedback.

### 2.4.10 Object Pattern

Object Pattern focuses on the actual creation of the Object Detected Notification based on the distance of the detected object. It then sends the resulting vibration pattern and delay pattern along with the id of the vibration motor corresponding to the sensor that detected the object to the Routing to give actual feedback.

### 2.4.11 Low Battery Pattern

Low Battery Pattern will be responsible for sending the vibration pattern and the delay pattern that will represent low power left in the system along with the id representing all vibration motors to Routing to give actual feedback.

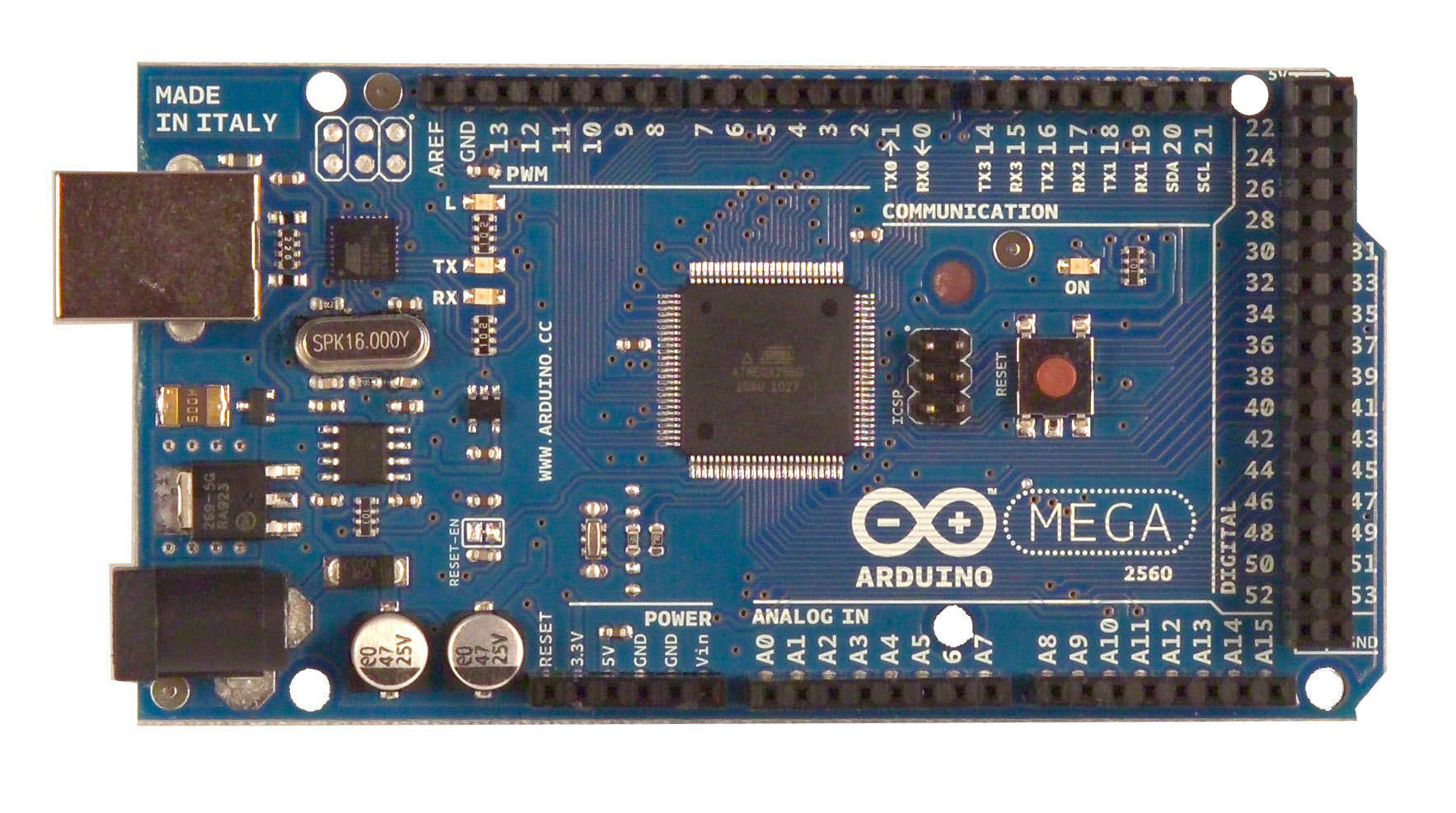
### 2.4.12 Critical Battery Pattern

Critical Battery Pattern will be responsible for sending the vibration pattern and the delay pattern that will represent the system turning off from insufficient battery charge along with an id representing all vibration motors to Routing to give actual feedback.

# ****3. System Hardware Description****

This section will cover the hardware components the Sight By Touch System is composed of. Each component’s quantity, purpose, specifications, and interfaces will be defined below in their own subsections.

## 3.1 Arduino Mega 2560



**Figure 3-1: Arduino Mega 2560**

### 3.1.1 Quantity

The Sight By Touch System will require only 1 Arduino Mega 2560.

### 3.1.2 Purpose

The purpose of the Arduino Mega 2560 is to act as the CPU of the Sight By Touch System. It is responsible for carrying out all tasks internal to the system such as turning the system on or off, updating the maximum detection range, controlling the distance sensors and vibration motors, analyzing distance sensor response data, monitoring the level of the battery powering the system, and creating and managing the different types of notifications.

### 3.1.3 Specifications

The Arduino Mega 2560 contains the following specifications:

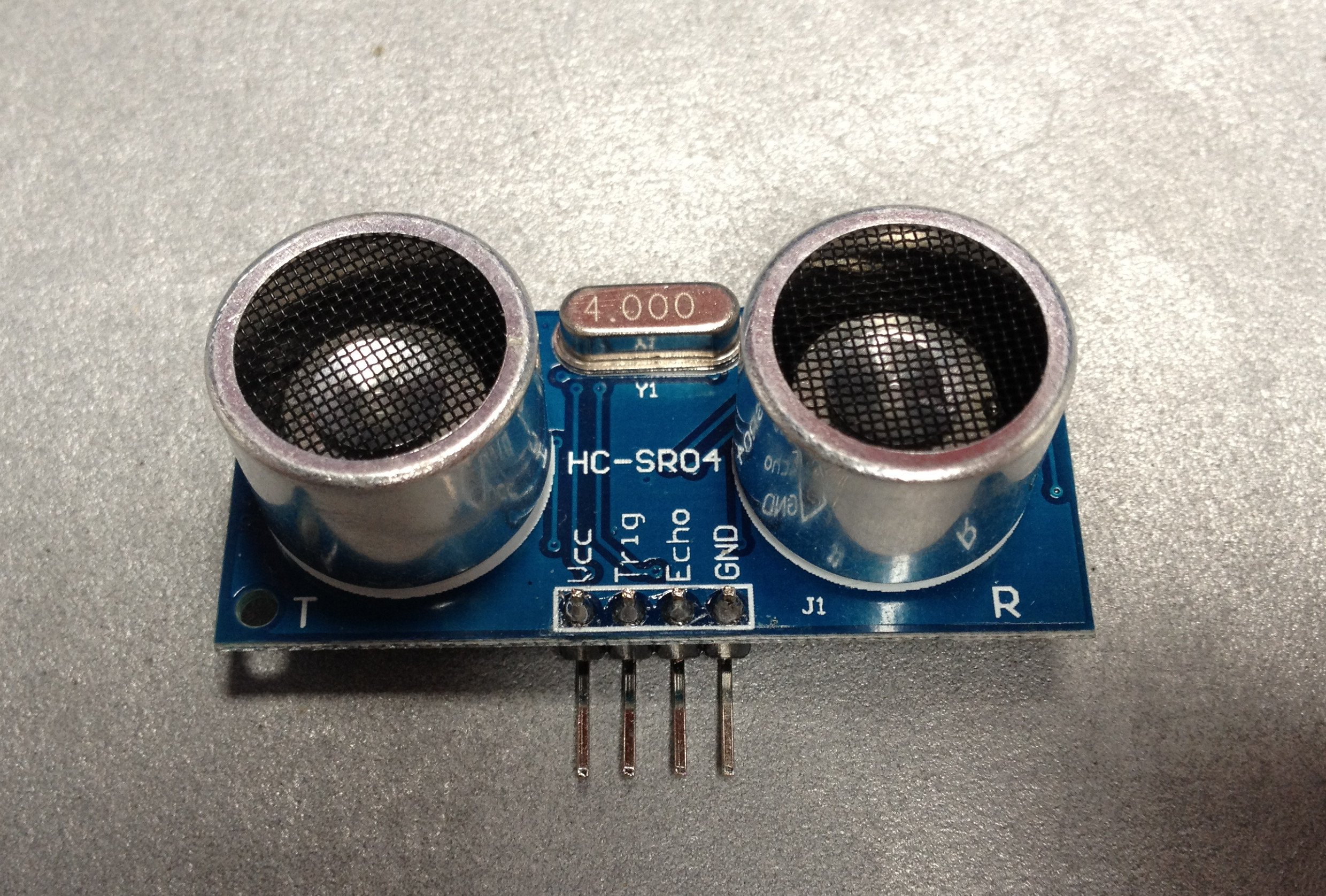
|  |  |
| --- | --- |
| Microcontroller | ATmega2560 |
| Operating Voltage | 5V |
| Input Voltage (recommended) | 7-12V |
| Input Voltage (limits) | 6-20V |
| Digital I/O Pins | 54 (of which 15 provide PWM output) |
| Analog Input Pins | 16 |
| DC Current per I/O Pin | 40 mA |
| DC Current for 3.3V Pin | 50 mA |
| Flash Memory | 256 KB of which 8 KB used by bootloader |
| SRAM | 8 KB |
| EEPROM | 4 KB |
| Clock Speed | 16 MHz |

**Table 3-1: Arduino Mega 2560 Specifications**

### 3.1.4 Interfaces

The Arduino Mega 2560 will interface with all other hardware components. It will interface with the “power button”, the “range adjuster switch”, all distance sensors, all vibration motors, and the external battery. The distance sensors and “range adjuster switch” will communicate with the Arduino Mega 2560 through its digital pins. Any digital pin (22 – 53 on the Arduino Mega 2560) may be used as long as it’s calibrated correctly in the software. The vibration motors will be connected to the digital PWM pins. Any digital PWM pin (2-13 on the Arduino Mega 2560) may be used as long as it’s calibrated correctly in the software and the software doesn’t prevent its use. The external battery will be connected to the Arduino Mega 2560 through its analog pin. As with digital pins, any analog pin (A0 – A15 on the Arduino Mega 2560) may be used as long as it’s calibrated correctly in the software. The external battery will also be connected to the Arduino Mega 2560’s Vin pin to provide power. The “power button” will be connected between the Vin pin and external battery to allow or stop voltage flowing from the external battery to the Arduino Mega 2560.

## 3.2 HC-SR04 Ultrasonic Sensor



**Figure 3-2: HC-SR04 Ultrasonic Sensor**

### 3.2.1 Quantity

The Sight By Touch System will detect objects from 8 different locations on the user’s front and sides. So, 8 HC-SR04 Ultrasonic Sensors are required.

### 3.2.2 Purpose

The purpose of the HC-SR04 Ultrasonic Sensors is to fire an ultrasonic wave to detect objects when a voltage signal is sent by the Arduino Mega 2560. Objects detected will give an echo wave. The sensors will read this echo and send a voltage to the Arduino Mega 2560 to signal that an object has been detected. 1 HC-SR04 Ultrasonic Sensor is part of a Sensor Module on the Sight By Touch System.

### 3.2.3 Specifications

The HC-SR04 Ultrasonic Sensor contains the following specifications:

|  |  |
| --- | --- |
| Working Voltage | DC 5 V |
| Working Current | 15mA |
| Working Frequency | 40Hz |
| Max Range | 5m |
| Min Range | 2cm |
| Measuring Angle | 15 degree |
| Trigger Input Signal | 10uS TTL pulse |
| Echo Output Signal | Input TTL lever signal and the range in proportion |
| Dimension | 45\*20\*15mm |

**Table 3-2: HC-SR04 Ultrasonic Sensor Specifications**

### 3.2.4 Interfaces

The HC-SR04 Ultrasonic Sensors will interface with theArduino Mega 2560. It will communicate with the Arduino Mega 2560 through its digital pins. Any digital pin (22 – 55 on the Arduino Mega 2560) may be used as long as the corresponding pin is used in the software.

## 3.3 SPST High-Current Mini Toggle Switch



**Figure 3-3: SPST High-Current Mini Toggle Switch**

### 3.3.1 Quantity

The Sight By Touch System will have 1 SPST High-Current Mini Toggle Switch.

### 3.3.2 Purpose

The purpose of the SPST High-Current Mini Toggle Switch is to act as the “range adjuster switch” of the system. It will be used to adjust the maximum detection range of the Sight By Touch System from indoors to outdoors or vice versa through a switch toggle.

### 3.3.3 Specifications

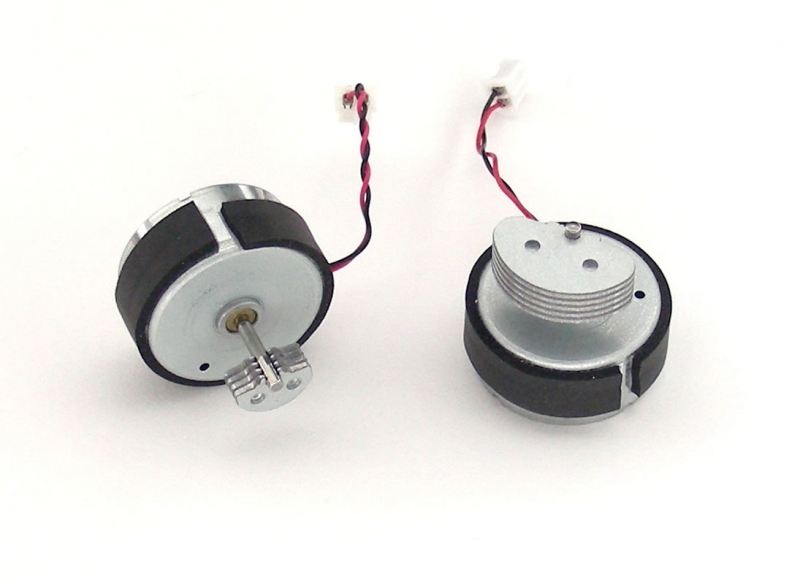
|  |  |
| --- | --- |
| Current Rating | 10 A |
| Voltage Rating | 125 V |
| Mounting Hole Size | ¼ in |

**Table 3-3: SPST High-Current Mini Toggle Switch Specifications**

### 3.3.4 Interfaces

The SPST High-Current Mini Toggle Switch will interface with theArduino Mega 2560. It will communicate with the Arduino Mega 2560 through its digital pins. Any digital pin may be used as long as the corresponding pin is used in the software.

## 3.4 XBOX 360 Vibration Motor



**Figure 3-4: XBOX 360 Vibration Motor**

### 3.4.1 Quantity

The Sight By Touch System will notify the user from the locations of the HC-SR04 Ultrasonic Sensors on the user’s front and sides. There are 8 of these sensors, so at least 8 vibration motors are required (1 for each sensor). More may be used if stronger vibration intensities are desired.

### 3.4.2 Purpose

The purpose of the XBOX 360 Vibration Motor is to provide haptic feedback to the user as a notification that the system is turned on, that an object was detected by the sensor(s), the battery powering the system is low, or the system is about to run out of power. Once it receives a voltage within its operating voltage range, the eccentric shaft in the motor is spun to generate vibrations. An XBOX 360 vibration motor (or more if desired) is part of a Sensor Module.

### 3.4.3 Specifications

The XBOX 360 Vibration Motor contains the following specifications:

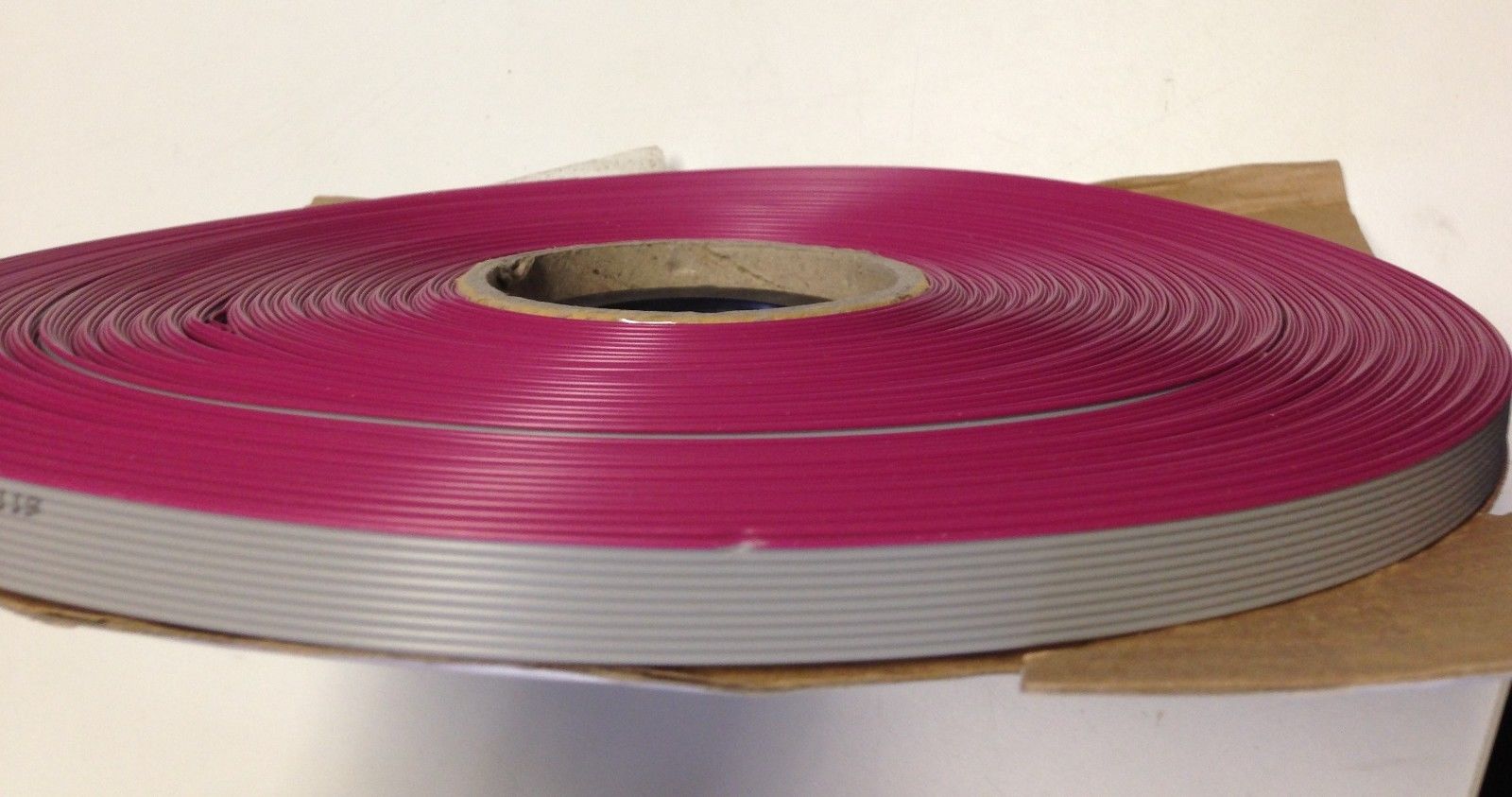
|  |  |
| --- | --- |
| Size | 24 x 12.5 mm |
| Operating Voltage Range | 1 – 5 V |

**Table 3-4: XBOX 360 Vibration Motor Specifications**

### 3.4.4 Interfaces

The DC Vibration Motors will interface with theArduino Mega 2560. It will communicate with the Arduino Mega 2560 through its digital PWM pins. Any digital PWM pin (2 – 13 on the Arduino) may be used as long as the corresponding pin is used in the software and the software doesn’t prevent its use.

## 3.5 Flat Ribbon Cable Roll



**Figure 3-5: Flat Ribbon Cable Roll**

### 3.5.1 Quantity

The Sight By Touch System requires 8 cable connections from the Arduino Mega 2560 to the 8 Sensor Modules. It is estimated that a 12ft. long cuttable Flat Ribbon Cable Roll is an adequate amount to implement all these connections. However, this amount can change depending on the size of the target user, so more or less may be desired.

### 3.5.2 Purpose

The sensors and vibration motors of the Sensor Modules will not be plugged in directly to the Arduino Mega 2560. They must use wires to connect to the Arduino Mega 2560. The purpose of the Flat Ribbon Cable is to provide a method of communication between the Arduino Mega 2560 and the components of the Sensor Modules. The design of this component will allow the multiple pins the Arduino Mega 2560 uses to connect with the components of a Sensor Module in one organized cable. This will minimize any tangling of wires. However, the Flat Ribbon Cables will not directly connect to the hardware components. A Ribbon Crimp Connector (described later) will be attached to each end of the cables to allow them to connect to those hardware components with the help of jumper wires and IDC Box Headers (both described later). We selected a 10 pin ribbon cable because it is an easily accessible component. There may be unused pins.

### 3.5.3 Specifications

The Flat Ribbon Cable Roll contains the following specifications:

|  |  |
| --- | --- |
| Length | 12 ft |
| Pins/Wires in one ribbon | 10 |
| Cuttable | Yes |

**Table 3-5: Flat Ribbon Cable Roll Specifications**

### 3.5.4 Interfaces

The Flat Ribbon Cable pieces will interface directly with the Ribbon Crimp Connectors to be able to connect with the IDC Box Headers and the jumper wires.

## 3.6 Ribbon Crimp Connector

## https://cdn.sparkfun.com/assets/parts/5/4/2/5/10650-02.jpg

**Figure 3-6: Ribbon Crimp Connector**

### 3.6.1 Quantity

Each of the 8 cable connections from the Arduino Mega 2560 to the Sensor Modules will require Ribbon Crimp Connectors (one on each end). So, 16 Ribbon Crimp Connectors will be required.

### 3.6.2 Purpose

The purpose of the Ribbon Crimp Connectors is to allow the Flat Ribbon Cables to connect to the IDC Box Headers and jumper wires (both described later) that are used to connect to the actual hardware components.

### 3.6.3 Specifications

The Ribbon Crimp Connector contains the following specifications:

|  |  |
| --- | --- |
| Pins | 10 |
| Pins by Row X Column | 2 X 5 |
| Pins Head Connection | Female |

**Table 3-6: Ribbon Crimp Connector Specifications**

### 3.6.4 Interfaces

The Ribbon Crimp Connectors will be used to connect the Flat Ribbon Cables with the IDC Box Headers and the smaller jumper wires.

## 3.7 IDC Box Header

## http://www.jayconsystems.com/media/catalog/product/cache/1/image/800x600/040ec09b1e35df139433887a97daa66f/2/5/2593.jpg

**Figure 3-7: IDC Box Header**

### 3.7.1 Quantity

IDC Box Headers will be used on only one end of each cable. With a total of 8 cable connections, only 8 IDC Box Headers are required.

### 3.7.2 Purpose

The purpose of the IDC Box Headers is to allow the cable connections at the System Control Module to be detachable. The Flat Ribbon Cables with the Ribbon Crimp Connectors will simply connect to the IDC Box Headers located on the System Control Module for the connections to the Sensor Modules to be completed.

### 3.7.3 Specifications

The IDC Box Header contains the following specifications:

|  |  |
| --- | --- |
| Pins | 10 |
| Pins by Row X Column | 2 X 5 |
| Pins Head Connection | Male |

**Table 3-7: IDC Box Header Specifications**

### 3.7.4 Interfaces

The IDC Box Headers will act as an interface between the Ribbon Crimp Connectors attached to the Flat Ribbon Cables and the jumper wire connections internal to the System Control Module.

## 3.7 Jumper Wires (Male to Male and Male to Female)

## http://www.elecfreaks.com/store/images/BBC_jumper_01.jpg

**Figure 3-7: Jumper Wires**

### 3.7.1 Quantity

The number of male to male jumper wires may vary depending on the total number of hardware components (especially vibration motors) used, but a pack of 40 is estimated to be sufficient to allow for connections from the vibration motors (assuming one for each Sensor Module) to the Flat Ribbon Cables with the Ribbon Crimp Connectors attached and for other components inside the System Control Module to connect to the Arduino Mega 2560. Similarly, the number of male to female jumper wires may also vary depending on the number of hardware components. But, for the minimum required hardware, a pack of 80 male to female jumper wires is estimated to be sufficient for the Flat Ribbon Cables with Ribbon Crimp Connectors to connect to the 8 HC-SR04 Ultrasonic Sensors and for components internal to the System Control Module to connect to the IDC Box Headers.

### 3.7.2 Purpose

The purpose of the Jumper Wires is to allow for the Flat Ribbon Cables with Ribbon Crimp Connectors to connect to the components of the Sensor Modules and for the components internal to the System Control Module to communicate with each other and connect to the IDC Box Headers to communicate with the Sensor Modules.

### 3.7.3 Specifications

The Jumper Wires contain the following specifications:

|  |  |
| --- | --- |
| Pins Head Connection | Male to Female, Male to Male |

**Table 3-7: Jumper Wire Specifications**

### 3.7.4 Interfaces

The male to male jumper wires will be used to connect the female pin heads (heads with open pin slots) of Ribbon Crimp Connectors to those of the vibration motors. They will also be used to connect the Breadboard (described later), “range adjuster switch”, “power button”, external battery, and Arduino Mega 2560 together through male pins (actual metal pins). The male to female jumper wires will be used to connect the male pins of the sensors to the female pin heads of the Ribbon Crimp Connectors attached to the Flat Ribbon Cables. They will also be used for the internal connections of the System Control Module from the IDC Box Headers attached such as connections to the Breadboard and Arduino Mega 2560.

## 3.8 SPST 3-Amp "Soft-Feel" Push On-Push Off Switch

## SPST 3-Amp

**Figure 3-8: SPST 3-Amp "Soft-Feel" Push On-Push Off Switch**

### 3.8.1 Quantity

The Sight By Touch System will only require 1 SPST 3-Amp "Soft-Feel" Push On-Push Off Switch to act as a “power button”.

### 3.8.2 Purpose

The purpose of the Push Switch is to act as the “power button” for the Sight By Touch System. It has 2 states. When pressed, it will stay pressed and allow the system to turn on. When pressed again, it will return to its original position and cause the system to turn off.

### 3.8.3 Specifications

The Push Switch contains the following specifications:

|  |  |
| --- | --- |
| Mounting Size | Hole of 3/8 in |
| Self-Locking | Yes |
| Electrical Rating | 3 A at 125VAC |

**Table 3-8: SPST 3-Amp "Soft-Feel" Push On-Push Off Switch Specifications**

### 3.8.4 Interfaces

The SPST 3-Amp "Soft-Feel" Push On-Push Off Switch will be connected to the Breadboard (described later) between the external battery and the Vin pin of the Arduino Mega 2560. When the pressed, it will close the circuit and allow power to flow from the battery to the Arduino Mega 2560. When pressed again, it will open the circuit and stop the power flow from the battery into the Arduino Mega 2560.

## 3.9 Resistor



**Figure 3-9: Resistor**

### 3.9.1 Quantity

The Sight By Touch System will use a voltage divider circuit to measure the power remaining in the external battery. This requires 2 resistors. 2 more resistors will be used for the circuitry for the “power button” and the “range adjuster switch”. So, a total of 4 resistors are needed.

### 3.9.2 Purpose

One purpose of the Resistors is be used in a voltage divider circuit to reduce the voltage of the external battery into the 5V operating range of the Arduino Mega 2560 analog pins. This will allow the Arduino Mega 2560 to safely read the voltage of the external battery. Another purpose is to complete the circuitry for the “power button” and the “range adjuster switch” to be functional.

### 3.9.3 Specifications

The Resistor contains the following specifications:

|  |  |
| --- | --- |
| Resistance | 220 k ohms and 120 k ohms (for voltage divider)  10 k ohms (for “power button” and “range adjuster switch”) |
| Composition | Metal Film |
| Power | 0.25 W |
| Tolerance | ± 5% |

**Table 3-9: Resistor Specifications**

### 3.9.4 Interfaces

The 220 k ohms and 120 k ohms resistors will be connected on the Breadboard (described later), between the external battery and an analog pin on the Arduino Mega 2560. Any analog pin (A0 – A15 on the Arduino Mega 2560) can be used as long as the corresponding pin is used in the software. One of the 10 k ohm resistors will be connected from the “power button” to ground on the Breadboard, and the other will be used for a similar connection for the “range adjuster switch”.

## 3.10 Solderless Plug-in Breadboard

## http://ecx.images-amazon.com/images/I/5136ARMCLcL._SX300_.jpg

**Figure 3-10: Solderless Plug-in Breadboard**

### 3.10.1 Quantity

The Arduino Mega 2560 will require 1 Solderless Plug-in Breadboard for connections.

### 3.10.2 Purpose

The purpose of the Solderless Plug-in Breadboard is to be able to hold the required circuitry for the “power button”, “range adjuster switch”, and external battery to connect to the Arduino Mega 2560. It will also be used to distribute power and ground connections from the Arduino Mega 2560 to components such as the “power button”, “range adjuster switch”, sensors, and vibration motors for them to function.

### 3.10.3 Specifications

The Solderless Plug-in Breadboard contains the following specifications:

|  |  |
| --- | --- |
| Size | 2.2 X 3.3 X 0.4 in |
| Weight | 1.1 oz |

**Table 3-10: Solderless Plug-in Breadboard Specifications**

### 3.10.4 Interfaces

The Solderless Plug-In Breadboard will interface with “power button”, “range adjuster switch”, Arduino Mega 2560, external battery, sensors, and vibration motors through jumper wires. The external battery will require an additional DC Barrel Jack Adapter (described later) and resistors, and the “power button” and “range adjuster switch” will require resistors to be connected to the Breadboard for them to function properly.

## 3.11 Portable 12VDC 8-Cell Battery Power Supply

## http://d114hh0cykhyb0.cloudfront.net/images/uploads/BPS-12VDC-AA.jpg

**Figure 3-11: Portable 12VDC 8-Cell Battery Power Supply**

### 3.11.1 Quantity

Only 1 Portable 12VDC 8-Cell Battery Power Supply is needed to power the Sight By Touch System.

### 3.11.2 Purpose

The purpose of the Portable 12VDC 8-Cell Battery Power Supply is to hold the AA batteries, which when assembled together as the external battery, will provide a power supply to the Arduino Mega 2560 to power the Sight By Touch System.

### 3.11.3 Specifications

The Rechargeable Battery Pack contains the following specifications:

|  |  |
| --- | --- |
| Max Voltage Supply Capability | 12 V |
| Battery Holder Type | 8-slot for AA |
| Dimensions | 2.46 in x 2.26 in x 1.16 in |
| Has output connector? | Yes |

**Table 3-11: Portable 12VDC 8-Cell Battery Power Supply Specifications**

### 3.11.4 Interfaces

The Portable 12VDC 8-Cell Battery Power Supply will hold the AA batteries used to provide power. To supply power to the system, an additional DC Barrel Jack Adapter (described later) is needed for the output connector to interface with to allow the output connector to be used to supply power from the batteries.

## 3.11 DC Barrel Jack Adapter

## https://cdn.sparkfun.com/assets/parts/5/8/0/3/10811-01.jpg

**Figure 3-11: DC Barrel Jack Adapter**

### 3.11.1 Quantity

Only 1 DC Barrel Jack Adapter is needed to allow the external battery to supply power to the Sight By Touch System through its output connector.

### 3.11.2 Purpose

The purpose of the DC Barrel Jack Adapter is to allow the external battery to supply power to the entire system through its output connector. It will also allow the external battery to be detached from the system should it need to be replaced or recharged.

### 3.11.3 Specifications

The DC Barrel Jack Adapter contains the following specifications:

|  |  |
| --- | --- |
| Electrical Rating | DC 20 V, 4 A |
| Withstand Voltage | 500 V AC |
| Operating Temperature | -20 deg C to 70 deg C |
| Breadboard friendly? | Yes |

**Table 3-11: DC Barrel Jack Adapter Specifications**

### 3.11.4 Interfaces

The DC Barrel Jack Adapter will interface with the output connector of the external battery and the Breadboard to supply power to the entire system.

## 3.11 Rechargeable Alkaline AA Batteries w/ Charger

## http://ecx.images-amazon.com/images/I/61VwhADfg2L._SL1500_.jpg

**Figure 3-11: Rechargeable Batteries w/ Charger**

### 3.11.1 Quantity

Rechargeable AA batteries often come in a package consisting of 4 AA batteries and a charger. Since 8 AA batteries are needed for the Portable 12VDC 8-Cell Battery Power Supply, 2 of the aforementioned packages are required.

### 3.11.2 Purpose

The purpose of the AA rechargeable batteries are to provide power to the Sight By Touch System through the Portable 12VDC 8-Cell Battery Power Supply. The chargers will be used to recharge the batteries, so they can be reused to power the system. The batteries need to be of the Alkaline type since the voltage they supply is good indicator of the remaining charge level. The Battery Level Monitor subsystem makes use of this characteristic to determine the remaining charge level of the external battery.

Assumption: Since our product will be distributed to the target user through a caretaker, it is assumed that he or she will assist the user in their initial use of the product, including using the chargers to charge the batteries. The chargers packaged with the Sight By Touch System will only work under US standards, so it is assumed that the target user will use the product in the US.

### 3.11.3 Specifications

The Rechargeable Alkaline AA Batteries contain the following specifications:

|  |  |
| --- | --- |
| Capacity of each AA battery | 2000 mAh |
| Voltage | 1.5 V |

**Table 3-11: Rechargeable Alkaline AA Batteries Specifications**

### 3.11.4 Interfaces

The Rechargeable Alkaline AA Batteries will interface with the Portable 12VDC 8-Cell Battery Power Supply to supply power to the Sight By Touch System. To recharge, they will interface with the two chargers, which they will be plugged into a wall outlet to charge the batteries.

## 3.12 Other

This subsection will describe other required hardware components. The Sight By Touch System doesn’t directly interact with these components for its functionalities, but they are still needed for implementation.

### 3.12.1 Computer

A computer with the Arduino IDE software installed is required to create the software that will be utilized by the Sight By Touch System.

### 3.12.2 Arduino USB Cable

An Arduino USB Cable is required to upload the software from the computer to the Arduino Mega 2560 for execution.

### 3.12.3 Custom Encasing

Custom Encasing for the Sensor Module (vibration motors, sensors) and the System Control Module (Arduino Mega 2560, external battery, Breadboard) will be required to protect the user from electricity and heat and to protect the hardware components from physical, dust, heat, water and electrical damage. The encasings will require special holes so the System Control Module and Sensor Modules can communicate through the Flat Ribbon Cables and their attached components.

### 3.12.4 Body Suit

A Body Suit is required to allow the user to comfortably wear the Sight By Touch System and experience safe haptic feedback. It will also hold the Sensor Modules and System Control Module. To hold the Sensor Modules, pockets will need to be sewn on the suit at the locations where the Sensor Modules will be placed. These pockets will also need to have holes for the sensors extrude, so they can accurately detect objects. The suit will require a belt to hold the System Control Module at the waist.

### 3.12.4 Straps

Straps will be needed to secure the Sensor Modules in place when the Sight By Touch System is functioning. This is so the Sensor Modules don’t displace themselves from the vibrations, which can hinder the sensors from detecting objects and the user from experiencing haptic feedback.

### 3.12.4 Screws

Screws will be needed to secure the Arduino Mega 2560 in place, so it isn’t damaged from movement when the user is using the Sight By Touch System.

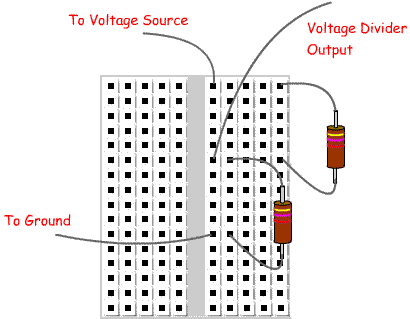
### 3.12.4 Double Edged Tape

Double Edged Tape will be needed to hold the Breadboard in place. This is so the movement of the wiring connected to the Breadboard is minimized when the user is using the Sight By Touch System.

## 3.13 Conceptual Hardware Diagrams

This subsection will describe the concepts behind a few of the complicated hardware connections. As these are conceptual diagrams, these do not completely reflect the actual connections for the implementation of the Sight By Touch System.

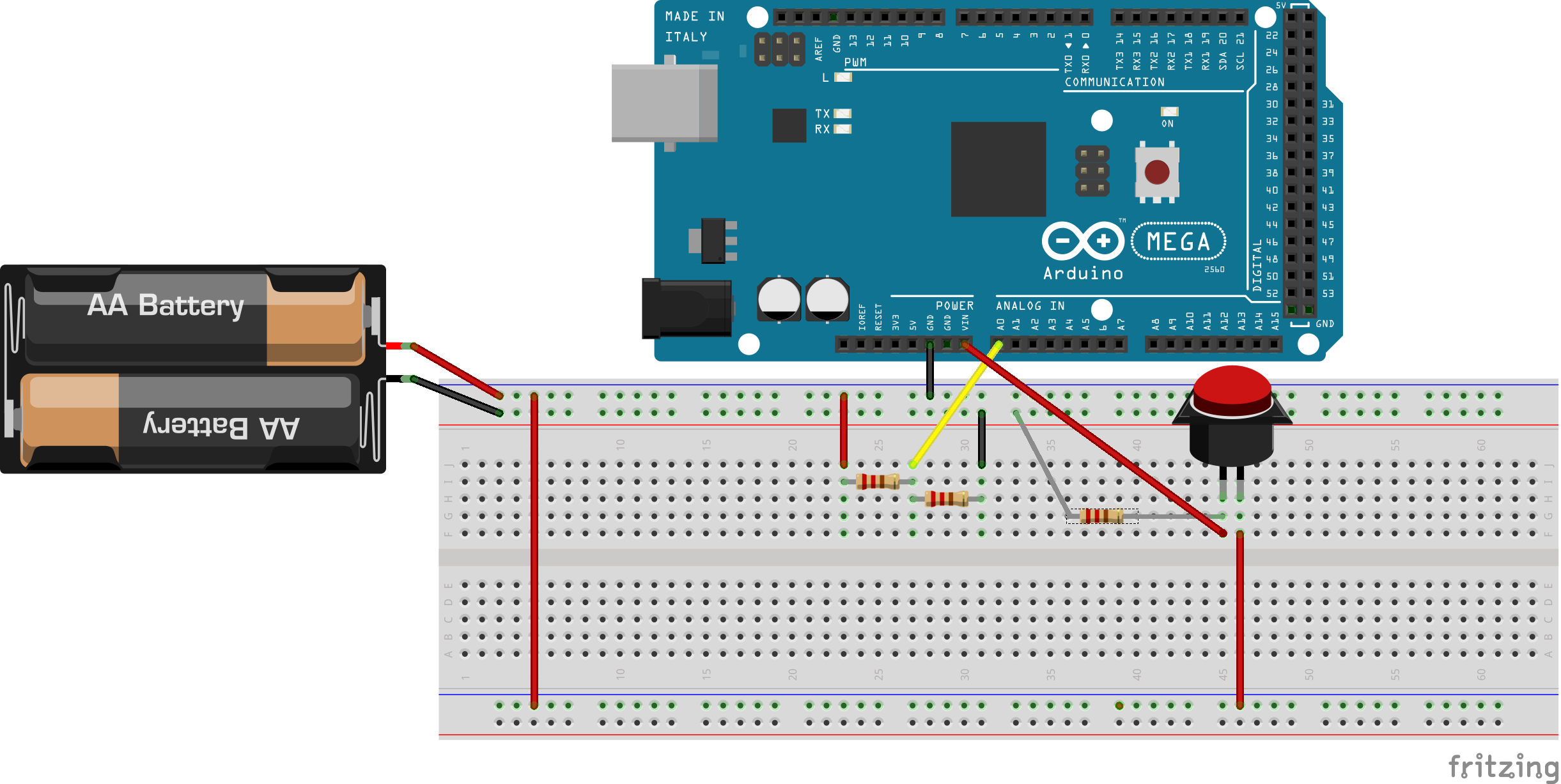
### 3.13.1 Voltage Divider Circuit



**Figure 3-12: Simple Voltage Divider Circuit**

The above diagram is a simple diagram of how a voltage divider is implemented using 2 resistors. The Voltage Divider Output is the reduced voltage. When used in the Sight By Touch System, the reduced voltage will be connect to an analog pin. Note: the resistor connected directly to the voltage source has the greater resistance.

### 3.13.2 Power Supply Connection

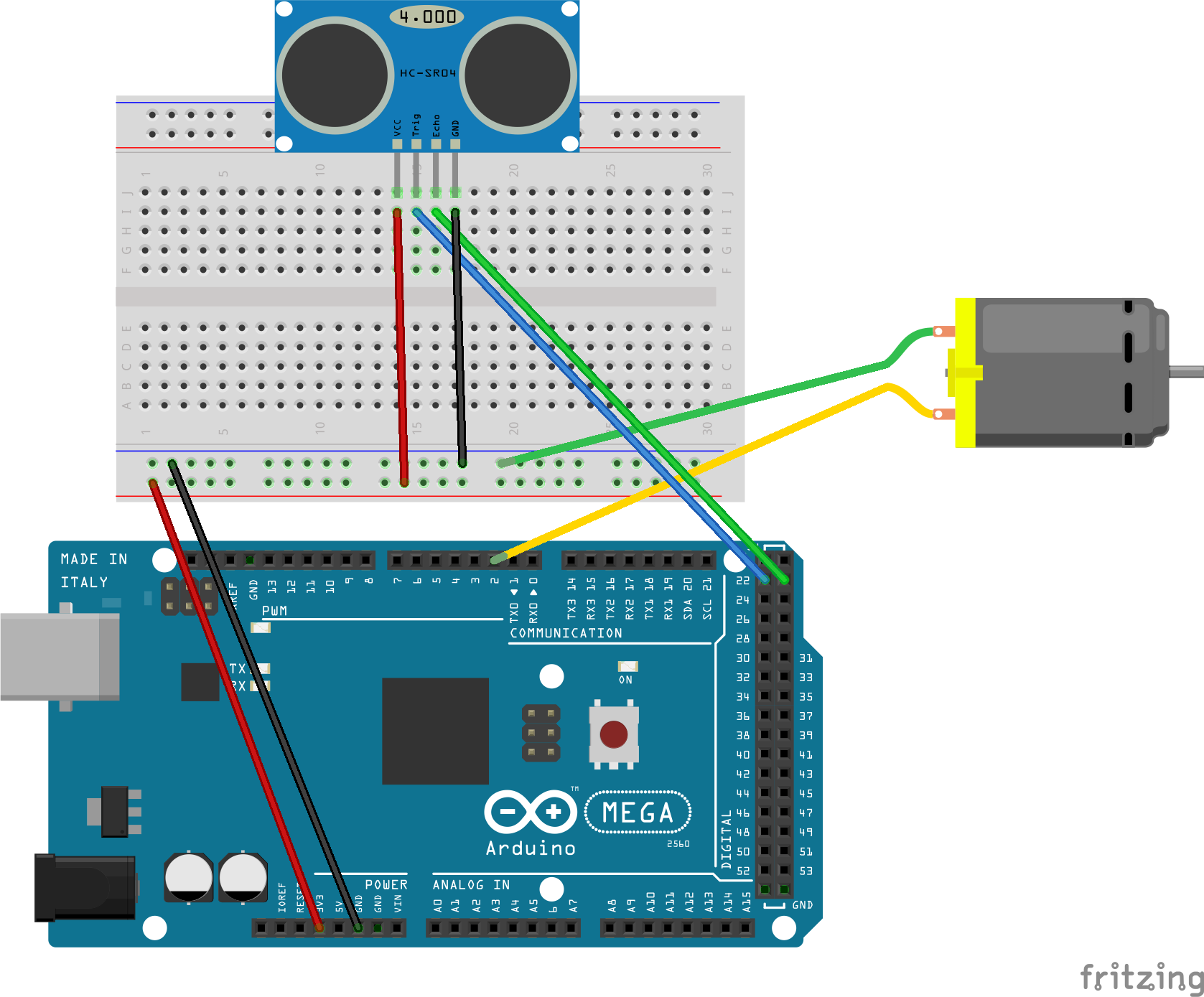


**Figure 3-13: Conceptual Power Supply Diagram**

The above figure is a visual representation of how the external battery will be connected to the Arduino Mega 2560. The external battery connected is an arbitrary voltage source for conceptual purposes and doesn’t completely reflect the actual battery powering the system. The actual battery will have 8 AA batteries and an output connector connected to a DC Barrel Jack. The red and black wires from the battery to the Breadboard will actually come from the jack, but this is skipped in the diagram above for simplicity. 2 of the resistors comprise the voltage divider circuit. The red wires represent the full voltage flow from the battery to the Arduino Mega 2560 and the voltage divider circuit. There is a conceptual push switch connected in the flow from the battery to the Vin pin. This will implement the turning on and off of the system. Note that there is another resistor from the push switch connected to ground to reflect the connection needed for the component to function. When the push switch is pressed when it was in its original position (open circuit), the circuit between the battery and the Vin pin will be closed to allow power to flow into the Arduino Mega 2560, turning it on. When the push switch is pressed again, it will cause the circuit to open and stop the power flow into the Arduino Mega 2560, turning it off. The yellow wire represents the flow from the voltage divider circuit to an Arduino Mega 2560 analog pin. This connection is what allows the Arduino Mega 2560 to safely read the voltage from a battery whose voltage exceeds 5V operating range of the Arduino Mega 2560. Finally, the black wires represent the ground flow to complete the circuit connection.

Note that disregarding the voltage divider circuit and the output to the analog pin, the circuitry for the push switch is similar to that of the “range adjuster switch”, with the exception that the output of the “range adjuster switch” is connected to a digital pin on the Arduino Mega 2560.

### 3.12.3 Sensor Module Internal Connection



**Figure 3-14: Conceptual Sensor Module Diagram**

The above figure is a visual representation of how the internal hardware components of a Sensor Module are connected. The red and black wires represent power and ground respectively. The red and black wires connected from the Arduino Mega 2560 to the Breadboard are to supply power and ground connections to the Breadboard, which can be used by the HC-SR04 Ultrasonic Sensor and conceptual vibration motor pictured. The use of these connections is shown by the red and black wire from the sensor to the other side of the BreadBoard and the green wire from the motor to the Breadboard. The blue and green wires from the Arduino Mega 2560 to the sensors are for the Arduino Mega 2560 to trigger when the sensor fires an ultrasonic wave via trigger pin on the sensor and to receive any response the sensor detects via the echo pin on the sensor. Finally, the yellow wire from the Arduino Mega 2560 to the conceptual vibration motor is for the Arduino Mega 2560 to control the motor itself. Each Sensor Module will follow this conceptual design. For the actual implementation of the Sight By Touch System, the connections to the Arduino in the diagram will be through the Flat Ribbon Cables described earlier, and the vibration motors will be XBOX 360 Vibration Motors connected to the digital PWM pins through the Flat Ribbon Cables.

# ****4. System Software Description****

This section will describe the overall software design of the Sight By Touch System. The system will have 3 different execution paths, but only two of them will be active throughout the operation of the system. The first path will be responsible for initializing the global variables and notifying the user that the system has been turned on, which is why this path will only be executed once for every operation session. The next path will be responsible for the actual detection and notifications to the user with respect to the objects that the system determines to be within the specified range. The final path will be executed regularly based on a timer interrupt and will be responsible for checking the level of the battery and notifying the user if the battery is low or close to being completely discharged.

## 4.1 Data Description

### 4.1.1 Global Variables

* max\_detect\_range – An integer that will hold the current maximum detection range in inches that the system uses to filter out objects detected by the sensor, but not within the range specified by the user. The variable can have either the value of 48 (48in. = 4ft.) for indoors and 120 (120 in. =10 ft.) for outdoors depending on the setting the user has selected.
* low\_flag – An integer used to control the number of times a low battery notification should be sent.
* critical\_low\_flag - An integer used to control the number of times a critical battery notification should be sent.
* vibrationMotors – An integer array holding the pin numbers of the digital PWM pins the vibration motors are connected to on the Arduino Mega 2560. The values in this array will range from 2 – 13.
* trigPins – An integer array holding the pin numbers of the digital pins the trigger pins of the sensors are connected to on the Arduino Mega 2560. The values in this array will range from 22 – 53. These pins will be used to have the distance sensors fire an ultrasonic wave to detect objects.
* echoPins – An integer array holding the pin numbers of the digital pins the echo pins of the sensors are connected to on the Arduino Mega 2560. The values in this array will range from 22 – 53. These pins will be used to read the response sent by the distance sensors when an object has been detected.

## 

## 4.2 System Notification Priorities

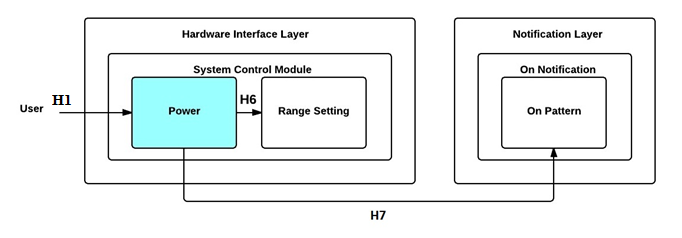
The Sight by Touch System will use interrupts to mimic the importance of each notification.

* The On Notification will can only be called when the system is turning on.
* Battery Level Notifications can only occur when the system generates an interrupt (a timer will be used to control when the interrupt occurs) to read the battery level and determines that the battery is low or close to being completely discharged.
* Object Detected Notifications will be the most common and occur whenever a detected object is within the current maximum detection range. These notifications can be interrupted at any time the system wants to check the battery level.

# ****5. Hardware Interface Layer: System Control Module****

The System Control Module subsystem is responsible for providing an interface between the user and the Sight By Touch System. The user will give commands to the Sight By Touch System through a power button and a knob on a control belt. This subsystem will translate those user commands into system commands the Sight By Touch System can process. This section will cover the System Control Module in detail by describing the detailed design of the modules that constitute the entire subsystem. These modules are the Power and Range Setting.

## 5.1 Power



**Figure 5-1: Power Module Diagram**

### 5.1.1 Prologue

Power is responsible for initializing the global variables, pins on the Arduino Mega 2560, and interrupts used the Sight By Touch System, making a call to update the maximum detection range, and then making a call to generate an On Notification when the user presses the power button on the control belt. This module will have two external inputs which consist of the power button and the toggle switch controlling the detection range. It calls the On Pattern module to signify that the system is to going to power on and calls the Range Setting module to update the maximum detection range of the system upon system startup.

### 5.1.2 Interfaces

|  |  |  |
| --- | --- | --- |
| **Producer** | **Consumer** | **Description** |
| User | Power | The user pressing the power button on the control belt will generate a button pulse that makes the Sight By Touch System call its Power module |
| Power | On Pattern | A call to signal that an On Notification is required. |
| Power | Range Setting | A call to update maximum detection range when the system is turned on. |

**Table 5-1: Power Module Interfaces**

### 5.1.3 External Data Dependencies

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| pulse | voltage | The signal representing that the system is going to turn on or off |

**Table 5-2: Power Module External Data Dependencies**

### 5.1.4 Internal Data Descriptors

This module does not have any internal data descriptors.

### 5.1.5 Service Dependencies

* setup() – an Arduino function that is used to initialize variables, pin modes, start using libraries, etc. The setup function will only run once, after each power up or reset of the Arduino board. Power will be called inside setup().
* attachInterrupt() – an Arduino function that is used to specify a named Interrupt Service Routine (ISR) to call when an interrupt occurs.
* TimerOne.h – an Arduino library that allows the use of a timer. This is can be used to set a timer for when an interrupt can occur. See Appendix for more information.
* Timer1.initialize() – an Timer1 library function that is used to set a period of time in microseconds for when to generate an interrupt. Using this will break the analogWrite function for digital PWM pins 9 and 10 on the Arduino Mega 2560.
* pinMode() – an Arduino function used to set the digital and digital PWM pins on the Arduino to input or output.

### 5.1.6 Module Processing (Pseudo-code Algorithm)

**void** Power**(void)**

**{**

//Initialize the global variables

//Initialize pins used on Arduino Mega 2560

//Initialize interrupts

/\*call Range Setting module so the system can determine the current

maximum detection range of the system\*/

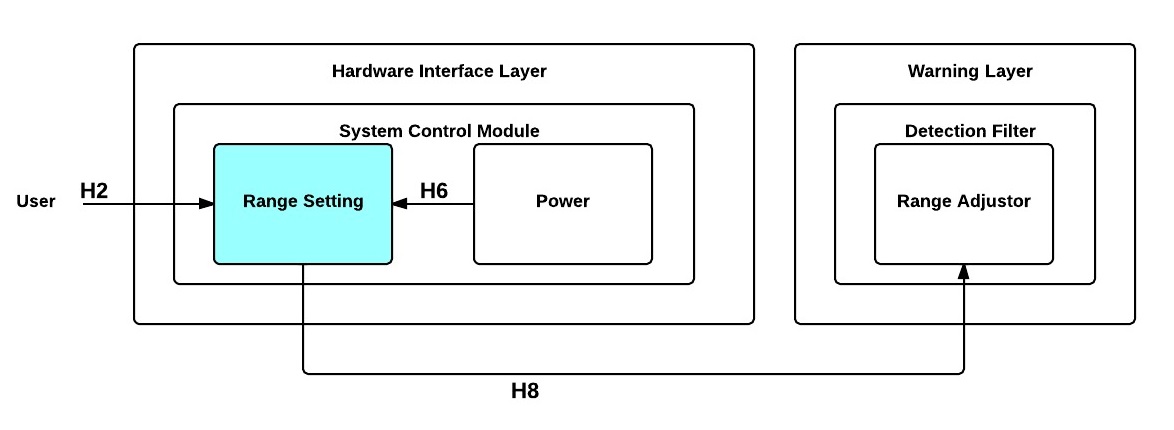
RangeSetting**();**

/\*call On Pattern module to generate On Notification to send to motors\*/

OnPattern**()**;

**}**

## 5.2 Range Setting



**Figure 5-2: Range Setting Module Diagram**

### 5.2.1 Prologue

Range Setting is responsible for accepting the user command representing the maximum detection range changing and translating it into readable format that can be processed by the system. This module will read a signal generated by a switch as input and passes the new maximum detection range to the Range Adjuster module in the Detection Filter subsystem for updating the current maximum detection range.

### 5.2.2 Interfaces

|  |  |  |
| --- | --- | --- |
| **Producer** | **Consumer** | **Description** |
| User | Range Setting | A new signal will be generated when the user flips the switch on the control belt to a certain position |
| Power | Range Setting | A call to update the maximum detection range when the system is turned on. |
| Range Setting | Range Adjuster | A value representing the position of the switch |

**Table 5-3: Range Setting Module Interfaces**

### 5.2.3 External Data Dependencies

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| pulse | voltage | The signal representing the position of the switch on the control belt |

**Table 5-4: Range Setting Module External Data Dependencies**

### 5.2.4 Internal Data Descriptors

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| time | long | An integer value representing the time when the switch was last read |
| debounce | long | An integer value representing how much time should pass to get a stable reading of the switch |
| switchpin | int | The pin number the toggle switch of the control belt will be connected to on the Arduino Mega 2560 |
| state | int | An integer value representing the position of the switch on the control belt translated from the pulse generated by the switch |

**Table 5-5: Range Setting Module Internal Data Descriptors**

### 5.2.5 Service Dependencies

* digitalRead() - an Arduino function that reads the value from the specified digital pin.
* millis() – an Arduino function that gives the time passed since the Arduino started up in milliseconds. This will be used to help determine the time that needs to pass before reading the switch to get a stable reading.

### 5.2.6 Module Processing (Pseudo-code Algorithm)

**void** RangeSetting**(void)**

**{**

/\*compare current time with time switch last read to determine if switch

reading will be stable\*/

**if** **(millis()-**time**>**debounce**)**

**{**

state **=** **digitalRead(**inPin**);**

RangeAdjuster**();**

time **=** **millis();** /\*update time to know how long to wait for a stable

reading\*/

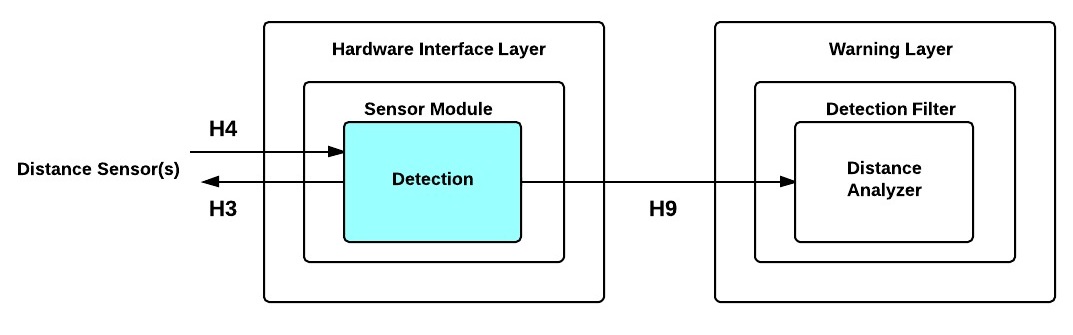
**}**

**}**

# ****6. Hardware Interface Layer: Sensor Module(s)****

The Sensor Module(s) subsystem is responsible for providing an interface between the external hardware (distance sensors and vibration motors) and the Sight By Touch System. The distance sensors will send an ultrasonic sound wave to the environment. This sound wave will get a response from nearby objects, and the distance sensors will read this response. This subsystem will translate those responses into system commands the Sight By Touch System can process. Additionally, this subsystem will also translate any commands to the vibration motors into signals, which the vibration motors can operate upon to provide haptic feedback to the user. This section will cover the Sensor Module(s) in detail by describing the detailed design of the modules that constitute the entire subsystem. These modules are the Detection and Routing.

## 6.1 Detection



**Figure 6-1: Detection Module Diagram**

### 6.1.1 Prologue

Detection is responsible for operating the distance sensors and translating any responses into readable data that can be processed by the system. This module will first output a signal to the distance sensors to send an ultrasonic sound wave to the environment. It will then read any response the distance sensors detect from nearby objects as input and translate them into a distance and sensor id. Finally, it will pass data to the Distance Analyzer module in the Detection Filter subsystem for processing. This process will repeat for all distance sensors in round robin fashion.

### 6.1.2 Interfaces

|  |  |  |
| --- | --- | --- |
| **Producer** | **Consumer** | **Description** |
| Detection | Distance Sensor | A signal to the Distance Sensor to generate ultrasonic sound waves and send them into the environment to detect nearby objects. |
| Distance Sensor | Detection | A signal from the Distance Sensor that represents that an object was detected by that sensor |
| Detection | Distance Analyzer | The distance from the Distance Sensor to inches and the id of the sensor that detected the object. |

**Table 6-1: Detection Module Interfaces**

### 6.1.3 External Data Dependencies

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| pulse | voltage | The signal representing that an object was detected by a distance sensor |

**Table 6-2: Detection Module External Data Dependencies**

### 6.1.4 Internal Data Descriptors

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| sensor\_id | int | The id of the sensor that detected the object |
| duration | long | The time taken for a distance sensor to detect an object. This time is taken from how long it took for a distance sensor to send a response voltage. |
| distance | long | The distance in inches to reach the nearby detected object from the distance sensor that detected it. This distance is derived from the time taken for a distance sensor to detect an object. |
| Size | int | An integer value holding the total number of distance sensors to operate. This is so the module knows how many sensors it must operate in the round robin algorithm. |
| HIGH | int | An Arduino integer constant that represents a voltage of 3 or more volts. In this module, it indicates a signal that tells the distance sensor to fire an ultrasonic wave. |
| LOW | int | An Arduino integer constant that represents a voltage of 2 or less volts. In this module, it indicates a signal that stops the distance sensor from firing an ultrasonic wave. |

**Table 6-3: Detection Module Internal Data Descriptors**

### 6.1.5 Service Dependencies

* delayMicroseconds() - an Arduino function that pauses the program for the amount of time (in microseconds) specified as a parameter. Used to give a distance sensor a little time to properly fire an ultrasonic wave.
* digitalWrite() - an Arduino function that writes a specified value to a specified digital pin. Used to have a distance sensor fire and stop firing ultrasonic waves
* pulseIn() - an Arduino function that reads a pulse (either HIGH or LOW) on a pin. Based on specified value given as a parameter, it waits for the specified pin also given as a parameter to change to the specified value. It then times how long that state last until it changes to the opposite state (HIGH -> LOW or LOW -> HIGH). Returns the length of the pulse in microseconds. Gives up and returns 0 if no pulse starts within a specified time out (default timeout is 1 second). This is used to get the time taken for a distance sensor to respond from when it fired.
* loop() - an Arduino function that will be used to actively control the Arduino board. This module will be called in this function so it can be called repeatedly while the system is on.

### 6.1.6 Module Processing (Pseudo-code Algorithm)

**void** Detection**(void)**

**{**

**for** every sensor\_id in the trigPins

**{**

/\*send signal to fire ultrasonic wave to detect objects using trig\_sensor\_id to specify which sensor\*/

**digitalWrite(**trigPins[sensor\_id]**,** LOW**);** /\*clear pin state\*/

**delayMicroseconds(**2**);** /\*give time to settle\*/

**digitalWrite(**trigPins[sensor\_id]**,** HIGH**);** /\*fire ultrasonic wave\*/

**delayMicroseconds(**10**);** /\*give time to make sure wave is fired properly\*/

**digitalWrite(**trigPins[sensor\_id]**,** LOW**);** /\*clear pin state\*/

/\*read any response from the distance sensor\*/

duration **=** **pulseIn(**echoPins[sensor\_id]**,** HIGH**);**

/\*There are 74 microseconds per inch. Divide by 2 to get the distance to the object. See appendix for more details\*/

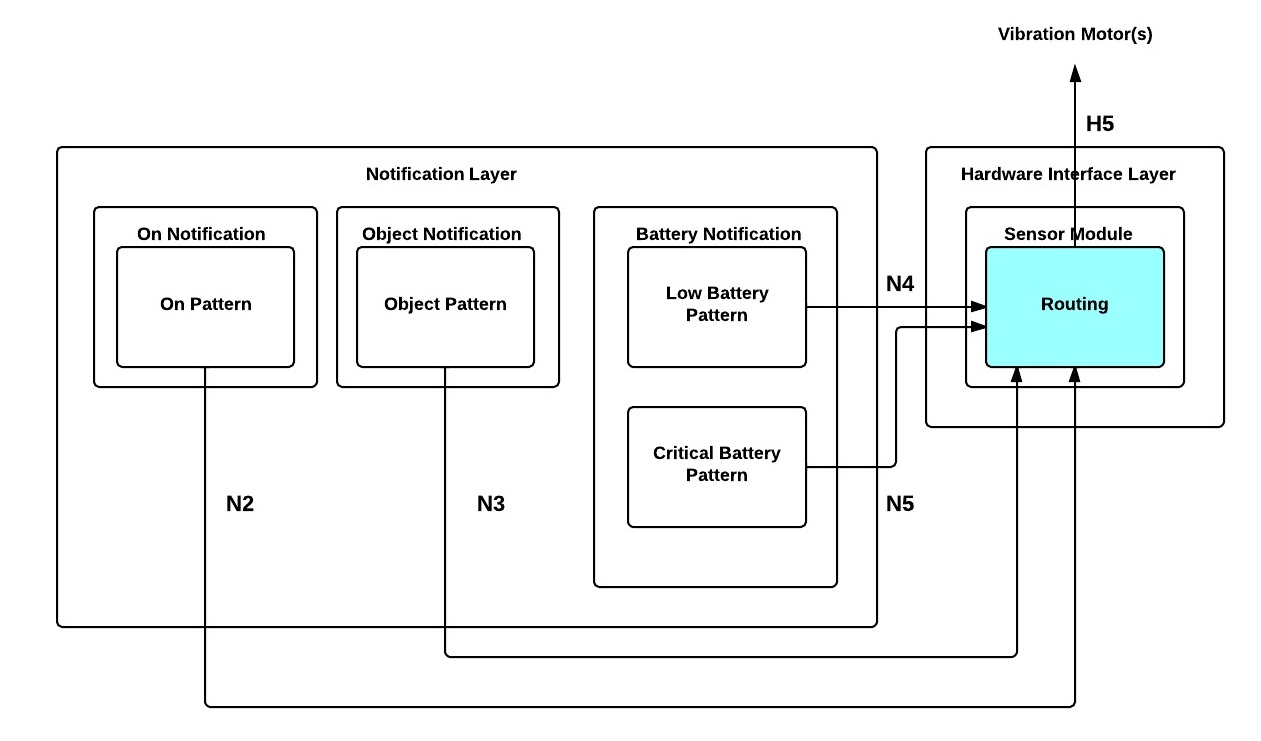
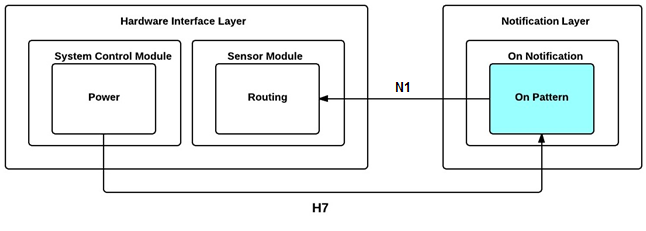
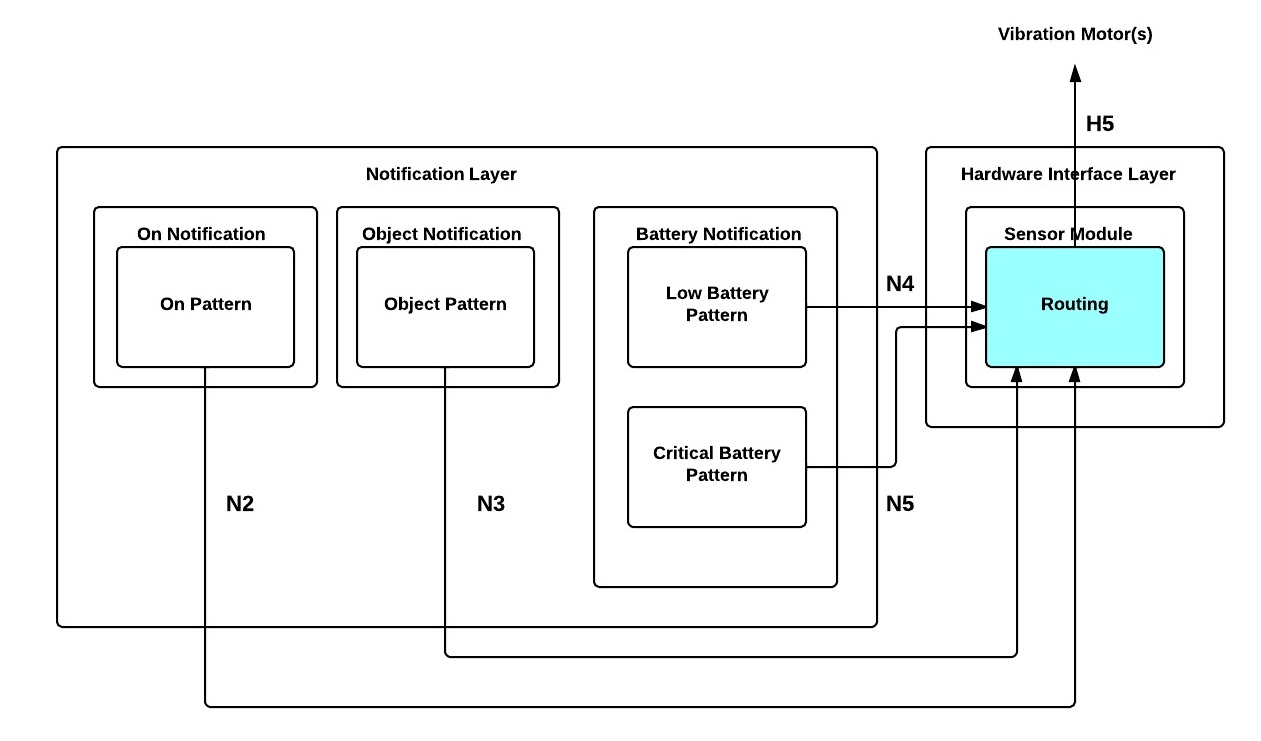
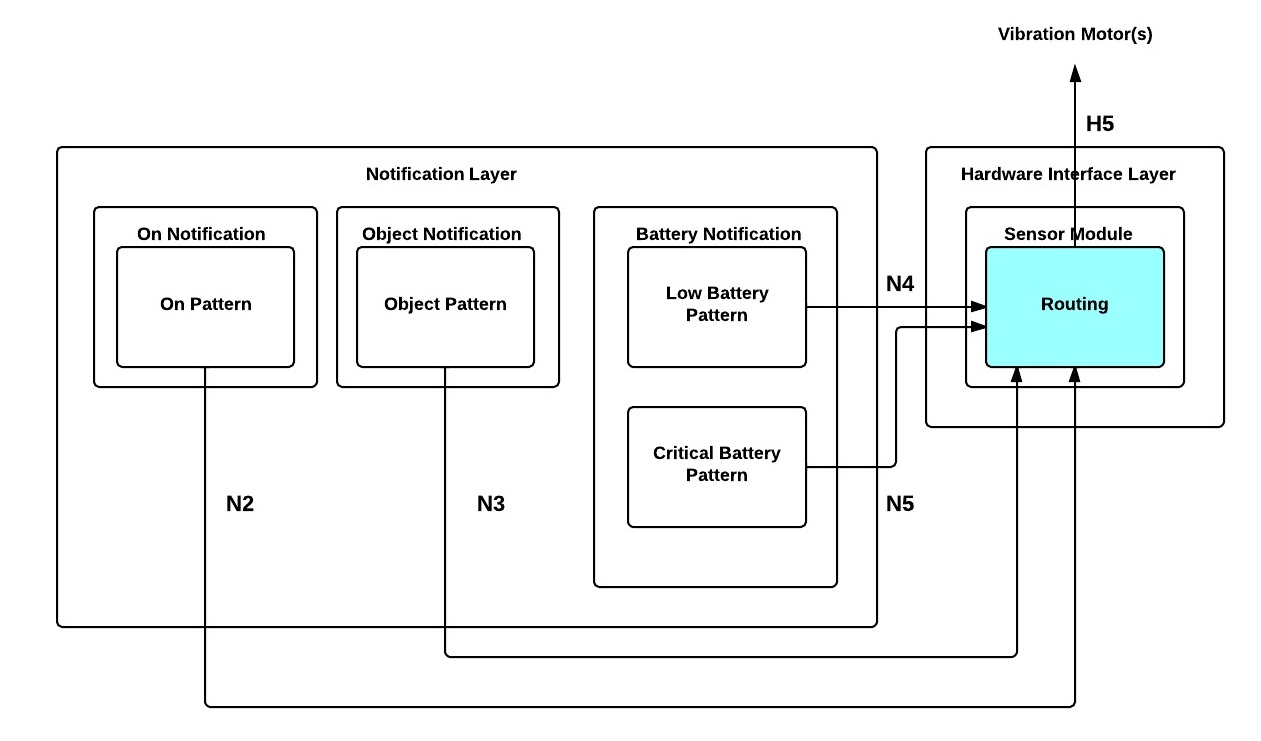
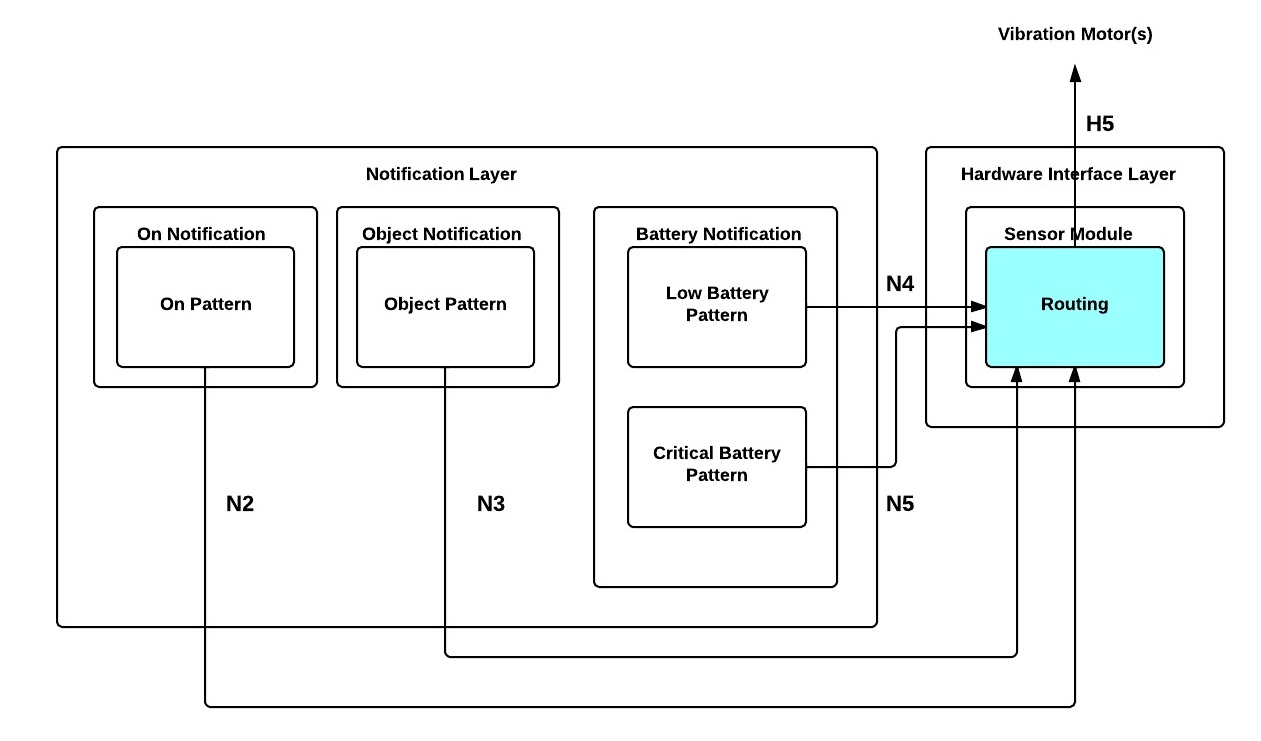
distance **=** duration**/**74**/**2**;**

DistanceAnalyzer**(**distance**,** sensor\_id**);**

**}**

**}**

## 6.2 Routing



**Figure 6-2: Routing Module Diagram**

### 6.2.1 Prologue

Routing is responsible for forwarding the signals sent by the modules in the Notification Layer to specified vibration motor(s) to operate. This module will accept the vibration pattern, id of the vibration motor that needs to be activated, and the delay pattern as input and outputs the signal to the proper vibration motor(s) based on the vibration motor id.

### 6.2.2 Interfaces

|  |  |  |
| --- | --- | --- |
| **Producer** | **Consumer** | **Description** |
| On Pattern | Routing | The vibration pattern, delay pattern, and id of vibration motors |
| Object Pattern | Routing | The vibration pattern, delay pattern, and id of vibration motor |
| Low Battery Pattern | Routing | The vibration pattern, delay pattern, and id of vibration motors |
| Critical Battery Pattern | Routing | The vibration pattern, delay pattern, and id of vibration motors |
| Routing | Vibration Motor(s) | A signal to the specified vibration motor (based on vibration motor id) to work. |

**Table 6-4: Routing Module Interfaces**

### 6.2.3 External Data Dependencies

This module does not have any external data dependencies.

### 6.2.4 Internal Data Descriptors

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| vibration\_id | int | The id of the vibration motor that needs to be activated |
| vibration\_pattern | int | An integer representing the intensity the vibration motor(s) should operate at. |
| delay\_pattern | int | An integer representing how long the vibration motors should operate |
| numOfMotors | int | An integer holding the total number of vibration motors to operate. |
| HIGH | int | An Arduino integer constant that represents a voltage of 3 or more volts. In this module, it indicates a signal that tells the distance sensor to fire an ultrasonic wave. |
| LOW | int | An Arduino integer constant that represents a voltage of 2 or less volts. In this module, it indicates a signal that stops the distance sensor from firing an ultrasonic wave. |

**Table 6-5: Routing Module Internal Data Descriptors**

### 6.2.5 Service Dependencies

* delayMicroseconds() - an Arduino function that pauses the program for the amount of time (in microseconds) specified as the parameter.
* delay() – an Arduino function that pauses the program for the amount of time (in milliseconds) specified as the parameter
* analogWrite() - an Arduino function that writes the value to a specified digital PWM pin. This is used to give motors varying voltage signals to operate at varying intensities

### 6.2.6 Module Processing (Pseudo-code Algorithm)

**void** Routing**(int** vibration\_pattern, **int** vibration\_id,

**int** delay\_pattern**)**

**{**

/\*this id represents operate all vibration motors\*/

**if(**vibration\_id **==** **-**1**)**

**{**

//use analogWrite with vibration\_pattern and vibration\_id to operate

//all motors

//use delayMicroseconds to allow motors to operate for a small time

//use analogWrite with vibration\_pattern and vibration\_id to stop

//all motors

**}**

**else**

**{**

//use analogWrite with vibration\_pattern and vibration\_id to operate

//the specified motor

//use delayMicroseconds to allow the motor to operate for a small

//time

//use analogWrite with vibration\_pattern and vibration\_id to stop

//the specified motor

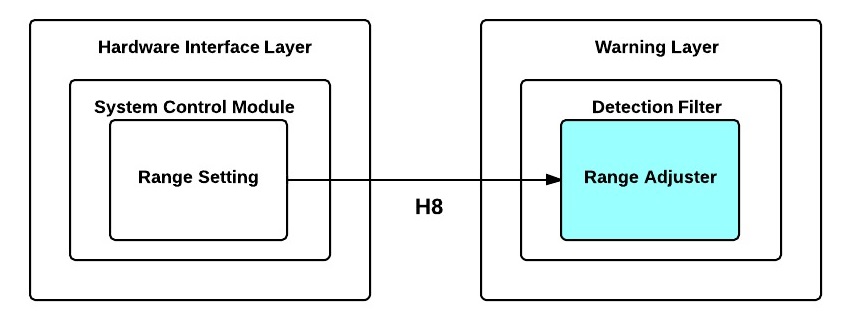
**}**

**}**

# ****7. Warning Layer: Detection Filter****

The Detection Filter subsystem is responsible for deciding whether an object detected by the distance sensor(s) is within the maximum detection range specified by the Sight By Touch System. The detected object’s distance from the Sight By Touch System is compared to the current detection range setting to make decisions. If the distance is within the range, the data will be forwarded to the Object Pattern module for processing. The maximum detection range can be changed between outdoor and indoor mode. When the range is modified, it will be reflected in the decision making process. This section will cover the Detection Filter in detail by describing the detailed design of the modules that constitute the entire subsystem. These modules are the Range Adjuster and Distance Analyzer.

## 7.1 Range Adjuster



**Figure 7-1: Range Adjuster Module Diagram**

### 7.1.1 Prologue

Range Adjuster is responsible for updating the maximum detection range the Distance Analyzer uses to make decisions. This module will receive the new maximum detection range from Range Setting as input.

### 7.1.2 Interfaces

|  |  |  |
| --- | --- | --- |
| **Producer** | **Consumer** | **Description** |
| Range Setting | Range Adjuster | A value representing the position of the switch to Range Adjuster for updating the current maximum detection range. |

**Table 7-1: Range Adjuster Module Interfaces**

### 7.1.3 External Data Dependencies

This module does not have any external data dependencies.

### 7.1.4 Internal Data Descriptors

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| new\_max\_detect\_range | int | An integer value representing the position of the switch |
| outdoor | int | An integer value representing the maximum detection range for outdoors (120 inches for our design) |
| indoor | int | An integer value representing the maximum detection range for indoors (48 inches for our design) |

**Table 7-2: Range Adjuster Module Internal Data Descriptors**

### 7.1.5 Service Dependencies

This module does not have any service dependencies

### 7.1.6 Module Processing (Pseudo-code Algorithm)

**void** RangeAdjuster**(int** new\_max\_detect\_range**)**

**{**

**if(**new\_max\_detect\_range **==** HIGH**)** /\*if switch is flipped up for outdoor

range\*/

**{**

max\_detect\_range **=** outdoor**;** /\*max detect range for outdoor\*/

**}**

**else** /\*must be indoor\*/

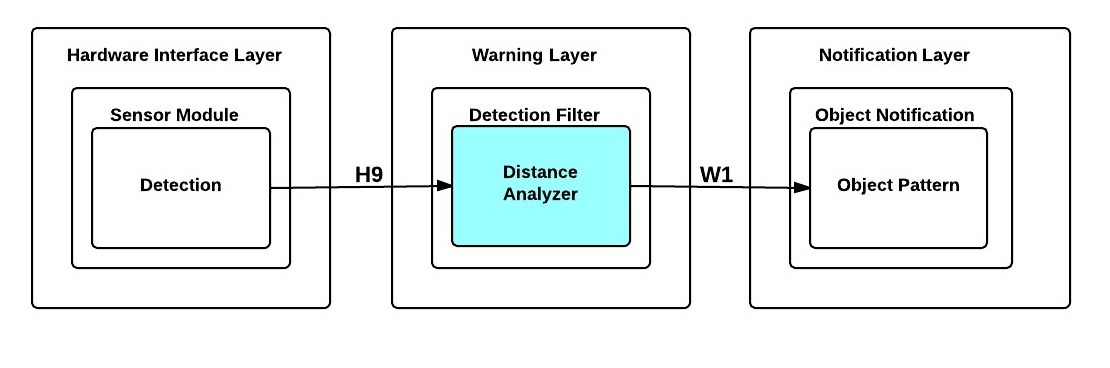
**{**

max\_detect\_range **=** indoor**;** /\*max detect range for indoor\*/

**}**

**}**

## 7.2 Distance Analyzer



**Figure 7-2: Distance Analyzer Module Diagram**

### 7.2.1 Prologue

Distance Analyzer is responsible for checking if the object detected by the distance sensors is within the current detection range of the Sight By Touch System by comparing the distance data sent as input to that current detection range. This module will accept the distance between the object detected and the sensor that detected that object and the id of the sensor that detected the object as inputs and passes the distance and sensor id to the Object Pattern module if the detected object is within the current detection range. Otherwise, it will ignore it.

### 7.2.2 Interfaces

|  |  |  |
| --- | --- | --- |
| **Producer** | **Consumer** | **Description** |
| Detection | Distance Analyzer | The id of the sensor that detected the object and the distance to the object from that sensor. |
| Distance Analyzer | Object Pattern | The id of the sensor that detected the object and the distance to the object from sensor. |

**Table 7-3: Distance Analyzer Module Interfaces**

### 7.2.3 External Data Dependencies

This module does not have any external data dependencies.

### 7.2.4 Internal Data Descriptors

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| sensor\_id | int | The id of the sensor that detected the object |
| distance | int | The distance to the object detected from the sensor that detected it in inches. |
| minimum\_detection\_range | int | An integer representing the minimum distance in inches that an object should be from the user. The value of the variable is 12. |

**Table 7-4: Distance Analyzer Module Internal Data Descriptors**

### 7.2.5 Service Dependencies

This module does not have any service dependencies.

### 7.2.6 Module Processing (Pseudo-code Algorithm)

**void** DistanceAnalyzer**(int** distance**,** **int** sensor\_id**)**

**{**

**if(**MINIMUM\_DETECTION\_RANGE **<=** distance **&&** distance **<=** max\_detect\_range**)**

**{**

ObjectPattern**(**distance**,** sensor\_id**);**

**}**

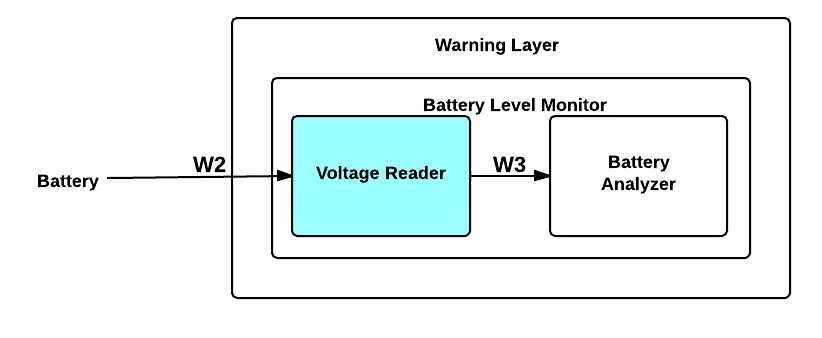
//otherwise ignore detected object

**}**

# ****8. Warning Layer: Battery Level Monitor****

The Battery Level Monitor is responsible for checking the remaining voltage in the battery that powers the Sight By Touch System at regular time intervals. To do so, an interrupt will be generated once a timer goes off, so it can read the voltage level of the battery and converts the readings into a readable data format that can be analyzed. If the remaining voltage reaches specific low levels, a call to the Battery Notification subsystem will be made to generate the appropriate battery notification. This section will cover the Battery Level Monitor in detail by describing the detailed design of the modules that constitute the entire subsystem. These modules are the Voltage Reader and Battery Analyzer.

## 8.1 Voltage Reader



**Figure 8-1: Voltage Reader Module Diagram**

### 8.1.1 Prologue

Voltage Reader is responsible for reading the voltage remaining in the battery powering the Sight By Touch System and converting it into a percentage of the battery’s full voltage capacity at regular time intervals. This module takes the voltage of the battery as input. It passes a voltage reading as a percentage to Battery Analyzer for analysis.

### 8.1.2 Interfaces

|  |  |  |
| --- | --- | --- |
| **Producer** | **Consumer** | **Description** |
| Voltage Reader | Battery Analyzer | A voltage reading as a percentage. |

**Table 8-1: Voltage Reader Module Interfaces**

### 8.1.3 External Data Dependencies

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| battery\_voltage | voltage | The electric potential energy per unit charge of the battery. It will be within 0-5 V, so the analog pin reading the voltage doesn’t burn out. |

**Table 8-2: Voltage Reader Module External Data Dependencies**

### 8.1.4 Internal Data Descriptors

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| DEFAULT | type | the default analog reference of 5 volts (on 5V Arduino boards) or 3.3 volts (on 3.3V Arduino boards) |
| battery\_pin | int | The analog pin number where the battery will send its voltage to. |
| referenceVolts | float | A value representing the analog voltage reference that will be used to convert the 0-5 V sent into the analog pin back to the original voltage level delivered by the battery |
| val | int | A value representing a 0-5 V signal read by the analog pin. |
| volts | int | The integer that represents the original voltage delivered by the battery when powering the system. |
| min\_voltage | float | A value representing the voltage level where the battery is about to run out of power |
| max\_voltage | float | A value representing the voltage level delivered by the battery when it is fully charged |
| volt\_percentage | int | An integer representing the voltage remaining in the battery as a percentage. |

**Table 8-3: Voltage Reader Module Internal Data Descriptors**

### 8.1.5 Service Dependencies

* analogRead() - an Arduino function that reads the value from the specified analog pin.
* analogReference() – an Arduino function that configures the reference voltage used for analog input (i.e. the value used as the top of the input range). Used to help map to correct voltage of battery.
* Timer1 – an Arduino library. This library holds the initialize() function to start a timer and holds an attachInterrupt() function that will make a function an Interrupt Service Routing that will be called when the timer goes off As shown in Power, these will be initialized in that module. See Appendix for more information on this library.

### 8.1.6 Module Processing (Pseudo-code Algorithm)

**void** VoltageReader**(void)**

**{**

**analogReference(**DEFAULT**);**

val**=analogRead(**battery\_pin**);**

//convert the voltage reading back to the original voltage level

//delivered by the battery

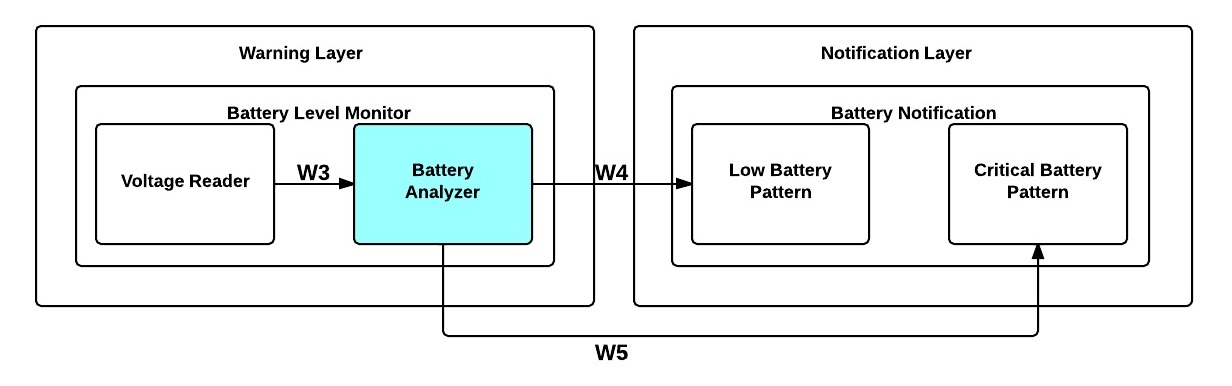
volt\_percentage **=** **((**volts **-** minVoltage**)/(**maxVoltage **-** minVoltage**))**

**\*** 100; /\*formula to convert to a percentage\*/

BatteryAnalyzer(volt\_percentage);

**}**

## 8.2 Battery Analyzer



**Figure 8-2: Battery Analyzer Module Diagram**

### 8.3.1 Prologue

Battery Analyzer is responsible for analyzing the percentage of the battery level remaining to check if the battery level has reached a specific low level of power. This module accepts the percentage from Voltage Reader as input and if necessary will make a call to the Low Battery Pattern module or the Critical Battery Pattern module depending on how low the battery level is.

### 8.3.2 Interfaces

|  |  |  |
| --- | --- | --- |
| **Producer** | **Consumer** | **Description** |
| Voltage Reader | Battery Analyzer | A voltage reading as a percentage. |
| Battery Analyzer | Low Battery Pattern | A call to generate a low battery notification. |
| Battery Analyzer | Critical Battery Pattern | A call to generate a critical battery notification. |

**Table 8-4: Battery Analyzer Module Interfaces**

### 8.3.3 External Data Dependencies

This module does not have any external data dependencies.

### 8.3.4 Internal Data Descriptors

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| percentage | float | A float representing the voltage of the battery as a percentage. |
| battery\_low | int | The battery percentage indicating that a recharge is recommended. |
| battery\_critical | int | The battery percentage indicating the system is about to turn off due to insufficient charge levels. |

**Table 8-5: Battery Analyzer Module Internal Data Descriptors**

### 8.3.5 Service Dependencies

This module does not have any service dependencies.

### 8.3.6 Module Processing (Pseudo-code Algorithm)

**void** BatteryAnalyzer**(int** percentage**)**

**{**

**if((**percentage **<=** battery\_low**)&&(**low\_flag**==**0**))&&**

**(**percentage **>** battery\_critical**))**

**{**

low\_flag **=** 1**;** /\*set flag so this notification only goes off once\*/

LowBatteryPattern**();**

**}**

**else** **if((**percentage **<=** battery\_critical**)&&(**critical\_low\_flag **==** 0**))**

**{**

critical\_low\_flag **=** 1**;** /\*set flag so this notification only goes off

once\*/

CriticalBatteryPattern**();**

**}**

**else** /\*nothing worth alerting yet\*/

**{**

//do nothing

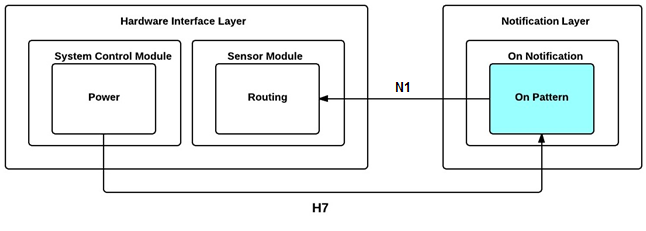
**}**

**}**

# ****9. Notification Layer: On Notification****

The On Notification subsystem is responsible for generating the on vibration pattern for the Sight By Touch System. The subsystem will be alerted by the System Control Module subsystem whenever the system needs to generate an on notification and then will send that pattern to the Sensor Module subsystem. This section will cover the On Notification in detail by describing the detailed design of the module that constitutes the entire subsystem. The module is the On Pattern.

## 9.1 On Pattern



**Figure 9-1: On Pattern Module Diagram**

### 9.1.1 Prologue

On Pattern is responsible for generating the vibration pattern and delay pattern that the system will use to notify the user that the system has been turned on. The vibration pattern and delay pattern are predefined, which each vibration motor will need to follow. This module will be called from the power module and will pass the vibration pattern as a integer, a delay pattern as an integer, and the id representing all vibration motors (-1) to the Routing module to signal that all of the vibration motors need to be activated.

### 9.1.2 Interfaces

|  |  |  |
| --- | --- | --- |
| **Producer** | **Consumer** | **Description** |
| Power | On Pattern | A call to signal that an on notification is required. |
| On Pattern | Routing | The vibration pattern, delay pattern, and id of the vibration motors |

**Table 9-1: On Pattern Module Interfaces**

### 9.1.3 External Data Dependencies

This module does not have any external data dependencies.

### 9.1.4 Internal Data Descriptors

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| vibration\_pattern | int | The vibration pattern that represents the system turning on as an integer. |
| vibration\_id | int | An integer representing the id of the vibration motor that needs to be activated, which in this case will be -1 to denote that all of the vibration motors need to be activated. |
| delay\_pattern | int | An integer representing how long the vibration motors should be operating |

**Table 9-2: On Pattern Module Internal Data Descriptors**

### 9.1.6 System Dependencies

This module does not have any system dependencies.

### 9.1.7 Module Processing (Pseudo-code Algorithm)

**void** OnPattern**()**

**{**

//generate vibration\_pattern, delay\_pattern, and vibration\_id values

/\*multiple calls to Routing may be made to make the on notification more unique\*/

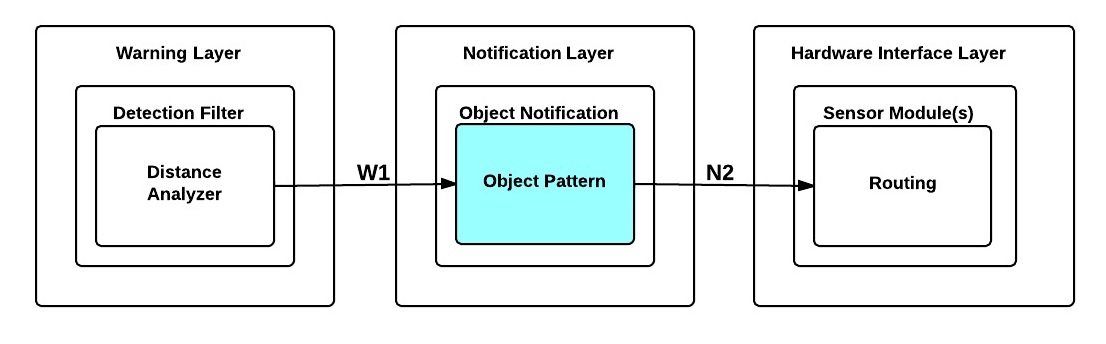
Routing**(**vibration\_pattern, vibration\_id, delay\_pattern**);**

**}**

# ****10. Notification Layer: Object Notification****

The Object Notification subsystem is responsible for generating an object obstruction vibration pattern for the Sight By Touch System. The vibration pattern generated by the subsystem will be proportional to the distance between the user and the object. This section will cover the Object Notification in detail by describing the detailed design of the module that constitutes the entire subsystem. The module is the Object Pattern.

## 10.1 Object Pattern



**Figure 10-1: Object Pattern Module Diagram**

### 10.1.1 Prologue

Object Pattern is responsible for generating the vibration pattern that the system will use to notify the user that an object is close to the user. The module will use the distance an object was detected at to determine the vibration pattern and delay pattern the vibration motor specified by the id should operate. This module accepts the distance of the detected object and id of the sensor from the Distance Analyzer module as input and passes the vibration pattern and delay pattern that need to be executed by a specific vibration motor and the id of that vibration motor ( based on the sensor id) to the Routing module.

### 10.1.2 Interfaces

|  |  |  |
| --- | --- | --- |
| **Producer** | **Consumer** | **Description** |
| Distance Analyzer | Object Pattern | The distance to the detected object from the sensor that detected it and the sensor id to specify which vibration motor to activate. |
| Object Pattern | Routing | The vibration pattern, delay pattern, and id of the vibration motor corresponding to the sensor that detected an object. |

**Table 10-1: Object Pattern Module Interfaces**

### 10.1.3 External Data Dependencies

This module does not have any external data dependencies.

### 10.1.4 Internal Data Descriptors

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| vibration\_id | int | The id of the vibration motor that will need to be activated. |
| sensor\_id | int | The id of the distance sensor that detected the object |
| distance | int | The distance to the object detected in inches. |
| vibration\_pattern | int | The vibration pattern that represents the intensity a vibration motor should operate at based on the distance as an integer. |
| delay\_pattern | int | An integer representing how long the vibration motor should be operating |

**Table 10-2: Object Pattern Module Internal Data Descriptors**

### 10.1.5 Service Dependencies

This module does not have any service dependencies.

### 10.1.6 Module Processing (Pseudo-code Algorithm)

**void** ObjectPattern**(int** distance**,** **int** sensor\_id**)**

**{**

//map distance to a vibration pattern and a delay pattern

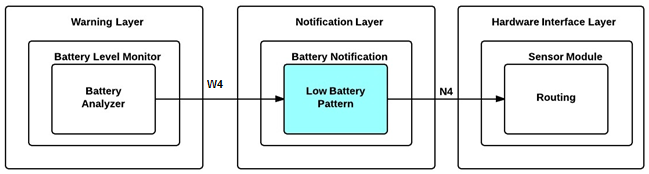
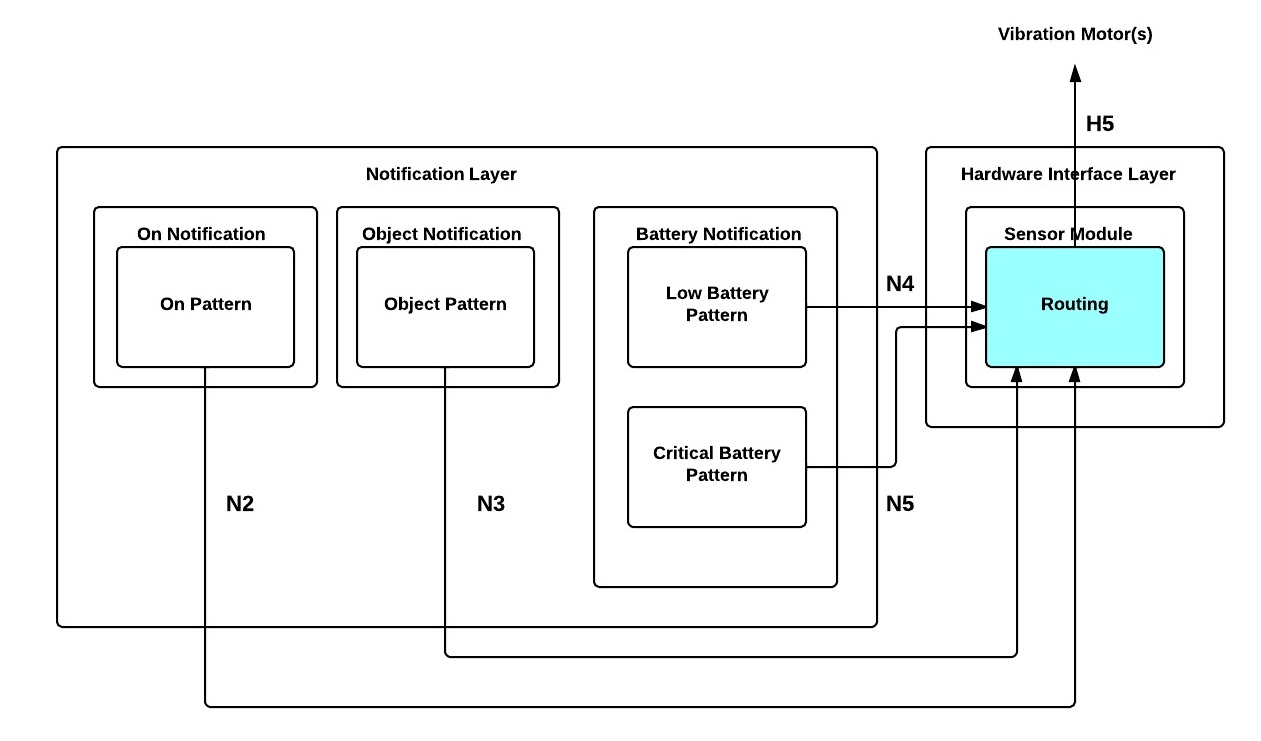
Routing**(**vibration\_pattern**,** sensor\_id**,** delay\_pattern**);**

**}**

# ****11. Notification Layer: Battery Notification****

The Battery Notification subsystem is responsible for generating a low battery vibration pattern and critical battery vibration pattern for the Sight By Touch System. The subsystem will be alerted by the Battery Analyzer module whenever the system needs to generate a low battery vibration pattern or a critical battery vibration pattern and then will send that pattern to the Sensor Module subsystem. This section will cover the Battery Notification in detail by describing the detailed design of the module Low Battery Pattern and Critical Battery Pattern.

## 11.1 Low Battery Pattern



**Figure 11-1: Low Battery Pattern Module Diagram**

### 11.1.1 Prologue

Low Battery Pattern is responsible for generating the vibration pattern and delay pattern that the system will use to notify the user that the battery is low on charge. The vibration pattern and delay pattern are predefined, which each vibration motor will need to follow. This module is called by the Battery Analyzer module and passes the vibration pattern and delay pattern to be executed by all vibration motors and the vibration motor id representing all vibration motors to the Routing module.

### 11.1.2 Interfaces

|  |  |  |
| --- | --- | --- |
| **Producer** | **Consumer** | **Description** |
| Battery Analyzer | Low Battery Pattern | A call to generate a low battery notification. |
| Low Battery Pattern | Routing | The vibration pattern, delay pattern, and id of the vibration motors |

**Table 11-1: Low Battery Pattern Module Interfaces**

### 11.1.3 External Data Dependencies

This module does not have any external data dependencies.

### 11.1.4 Internal Data Descriptors

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| vibration\_pattern | int | The vibration pattern that represents the battery is low as an integer |
| vibration\_id | int | An integer representing the id of the vibration motor that needs to be activated, which in this case will be -1 to denote that all of the vibration motors need to be activated. |
| delay\_pattern | int | An integer representing how long the vibration motors should be operating |

**Table 11-2: Low Battery Pattern Module Internal Data Descriptors**

### 11.1.5 Service Dependencies

This module does not have any service dependencies.

### 11.1.6 Module Processing (Pseudo-code Algorithm)

**void** LowBatteryPattern**()**

**{**

//generate vibration\_pattern, delay\_pattern, and vibration\_id values

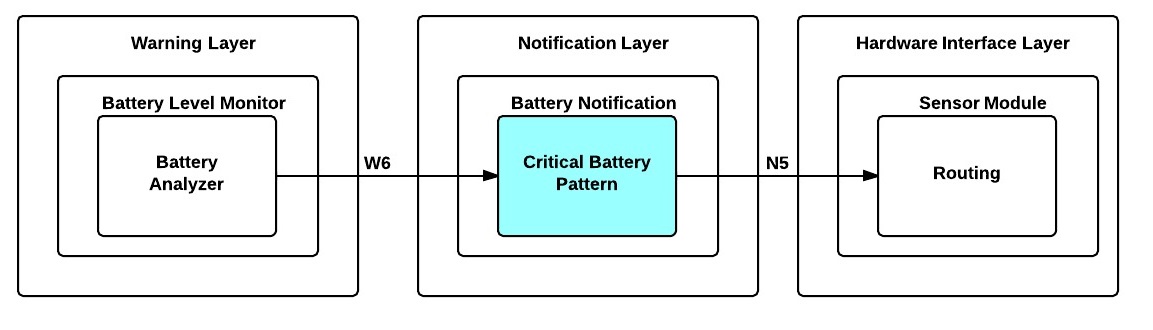
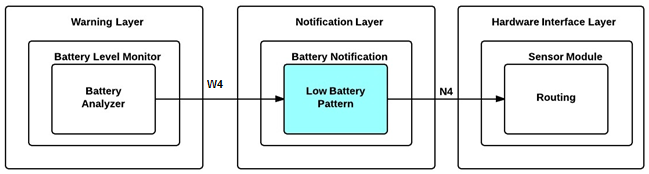
/\*multiple calls to Routing may be made to make the low battery

notification more unique\*/

Routing**(**vibration\_pattern, vibration\_id, delay\_pattern**);**

**}**

## 11.2 Critical Battery Pattern



**Figure 11-2: Critical Battery Pattern Module Diagram**

### 11.2.1 Prologue

Critical Battery Pattern is responsible for generating the vibration pattern and delay pattern that the system will use to notify the user that the battery is too low and the system is about to turn off. The vibration pattern and delay pattern are predefined, which each vibration motor will need to follow. This module is called by the Battery Analyzer module and passes the vibration pattern and delay pattern to be executed by all vibration motors and the vibration motor id representing all vibration motors to the Routing module.

### 11.2.2 Interfaces

|  |  |  |
| --- | --- | --- |
| **Producer** | **Consumer** | **Description** |
| Battery Analyzer | Critical Battery Pattern | A call to generate a critical battery notification. |
| Critical Battery Pattern | Routing | The vibration pattern, delay pattern, and id of vibration motors |

**Table 11-3: Critical Battery Pattern Module Interfaces**

### 11.1.3 External Data Dependencies

This module does not have any external data dependencies.

### 11.1.4 Internal Data Descriptors

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| vibration\_pattern | String | The vibration pattern that represents the system is about to turn off due to insufficient power remaining in the battery as an integer. |
| vibration\_id | int | An integer representing the id of the vibration motor that needs to be activated, which in this case will be -1 to denote that all of the vibration motors need to be activated. |
| delay\_pattern | int | An integer representing how long the vibration motors should be operating |

**Table 11-4: Critical Battery Pattern Module Internal Data Descriptors**

### 11.1.5 Service Dependencies

This module does not have any service dependencies.

### 11.1.6 Module Processing (Pseudo-code Algorithm)

**void** CriticalBatteryPattern**()**

**{**

//generate vibration\_pattern, delay\_pattern, and vibration\_id values

/\*multiple calls to Routing may be made to make the critical battery notification more unique\*/

Routing**(**vibration\_pattern, vibration\_id, delay\_pattern**);**

**}**

# 12. Quality Assurance

## 12.1 Test Plan and Procedures

All aspects of the system architecture shall be tested by Team Survivors to ensure that the Sight By Touch System fulfills the requirements defined in the SRS, ADS and DDS documents. Each component, module, subsystem and layer will be tested individually in order to validate that all specifications are satisfied. The system will also be tested as a whole to validate that each of the components were integrated successfully.

## 12.2 Module/Unit Testing

### 12.2.1 Hardware Interface Layer

**12.2.1.1 System Control Module subsystem**

1. **Power**

The Power module will power on and off the system. Checking with the On Pattern module to ensure that it receives a call and that the system is initialized will verify the Power module.

1. **Range Setting**

The Range Setting module will verify that it reads the new position of the switch correctly by printing out the new position of the toggle switch.

**12.2.1.2 Sensor Module subsystem**

1. **Detection**

The Detection module will be verified by generating a signal to activate the distance sensor, take the response and output the result of the Distance Sensor translated by this module.

1. **Routing**

The Routing module will be sent test parameters containing a vibration pattern, a delay pattern, and vibration motor id within it. It will be verified that this module can translate these parameters and forward a signal to the proper vibration motor(s) based on the vibration motor id.

### 12.2.2 Warning Layer

**12.2.2.1 Detection Filter subsystem**

1. **Range Adjuster**

The Range Adjuster module will be sent an integer representing the switch’s position. We will verify that the system updates the maximum detection range by printing out the result of the new maximum range and making sure it matches what we inputted.

1. **Distance Analyzer**

The Distance Analyzer module will receive a test distance that is within the current detection range and one that is outside to verify that the Distance Analyzer module will be able to filter out the proper distances.

**12.2.2.2 Battery Level Monitor subsystem**

1. **Voltage Reader**

The Voltage Reader module will be called at regular time intervals to check the voltage of the battery. We will verify this module by printing out the results of the readings and compare with a voltage meter reading of the same battery.

1. **Battery Analyzer**

The Battery Analyzer module will receive a test percentage of the battery that is below a particular level of power left as well as one that is above that level to verify that this module is working properly.

### 12.2.3 Notification Layer

**12.2.3.1 On Notification subsystem**

1. **On Pattern**

The On Pattern module will be activated and must send a unique vibration pattern, a unique delay pattern, and the correct vibration motor id. It will be verified by checking the output produced by this module matches the correct pattern.

**12.2.3.2 Object Notification subsystem**

1. **Object Pattern**

The Object Pattern module will be activated and must send a unique vibration pattern, a unique delay pattern, and the correct vibration motor id. It will be verified by checking the output produced by this module matches the correct pattern.

**12.2.3.3 Battery Notification subsystem**

1. **Low Battery Pattern**

The Low Battery Pattern module will be activated and must send a unique vibration pattern, a unique delay pattern, and the correct vibration motor id. It will be verified by checking the output produced by this module matches the correct pattern.

1. **Critical Battery Pattern**

The Critical Battery Pattern module will be called and must send a unique vibration pattern, a unique delay pattern, and the correct vibration motor id. It will be verified by checking the output produced by this module matches the correct pattern.

## 12.3 Component/Hardware Testing

### 12.3.1 Distance Sensor(s)

The Distance Sensor(s) should be able receive a pulse of high voltage, this will initiate the sensor and transmit out a cycle of ultrasonic burst and wait for the reflected ultrasonic burst. When the sensor detects the ultrasonic from the receiver, it will set the Echo pin to high and delay for a period (width) proportional to the distance. It should be able to connect to the Arduino Mega 2560 without issues.

### 12.3.2 Vibration Motor(s)

The Vibration Motor(s) should function properly during communication and respond to provide haptic feedback to the user.

### 12.3.3 System Control Module

The System Control Module should allow the user to be able to communicate with the system directly by turning the system On/Off and adjusting the maximum detection range.

### 12.3.4 Arduino Mega 2560

The Arduino Mega 2560 should be able to connect to Distance Sensor(s), Vibration Motor(s) and the System Control Module without any issues. It should accept serial data and output digital data.

## 12.4 Integration Testing

### 12.4.1 Hardware Interface Layer

1. Verify that all vibration motors are activated with a unique pattern providing feedback to the user when user turns “On” the system.
2. Verify that the maximum detection range is changed when the user switches maximum range knob.
3. Verify that the correct vibration motor(s) is activated when an object is detected.

### 12.4.2 Warning Layer

1. Verify that distances that are greater than the maximum detection range or lesser than the minimum detection range are disregarded.
2. Verify that the system notifies the user when the system’s battery is low.

### 12.4.3 Notification Layer

1. Verify that the system notifies the user when the system is going to turn “On”.
2. Verify that the system notifies the user when the system has detected an object within the range specified.
3. Verify that the system notifies the user when the system’s battery is low.
4. Verify that the system notifies the user when the system’s battery is critically low.

## 12.5 System Verification Testing

All in all, the system shall be tested using an inside-out approach. We will test the lowest level components and slowly work our way out. After each component is tested individually, we will integrate them with their proper partner and repeat this step until the system is fully integrated. At the end of our integration, we will test the entire system using the black box testing approach based on all requirements where the system needs to respond properly.

## 12.6 Test Cases

|  |  |
| --- | --- |
| **Test Case** | **Expected Result** |
| User turns on the system | All vibration motors should activate with a unique vibration pattern. |
| User turns off the system | System powers off. |
| User changes maximum detection range | Maximum detection range should be changed to correct maximum detection range. |
| Place an object in front of user at 5 feet | System should activate the correct vibration motor(s) with the appropriate intensity based on which sensor module detected the object. |

**Table 12-1: Test Cases**

# 13. Requirements Traceability

## 13.1 Purpose

Team Survivors utilizes requirements mapping in order to verify that our architectural and detailed design satisfies the requirements defined in our System Requirements Specification document.

This section demonstrates how the key requirements of the Sight By Touch System are mapped to the modules that address them. Each matrix shows which modules are responsible for fulfilling which key requirements and which complexities in the overall system.

## 13.2 Requirements Traceability Matrix By Module

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | **Modules** | **Power** | **Range Setting** | **Detection** | **Routing** | **Interpreter** | **Range Adjuster** | **Distance Analyzer** | **Voltage Reader** | **Battery Analyzer** | **On Pattern** | **Object Pattern** | **Low Battery Pattern** | **Critical Battery Pattern** |
| **No.** | **Requirement** |  |  |  |  |  |  |  |  |  |  |  | **Object** |  |  |
| **3.1** | On and Off |  | x |  |  |  |  |  |  |  |  | x |  |  |  |
| **3.2** | Detect Obstructions |  |  |  | x |  |  | x | x |  |  |  |  |  |  |
| **3.3** | Obstruction Notification |  |  |  |  | x |  |  |  |  |  |  | x |  |  |
| **3.4** | Battery Powered |  | x |  |  |  |  |  |  | x | x |  |  | x | x |
| **3.6** | Low Battery Notification |  |  |  |  | x |  |  |  | x | x |  |  | x | x |
| **3.7** | User Friendly |  | x | x |  |  |  |  |  |  |  |  |  |  |  |
| **5.2** | Real-Time Response |  |  |  | x | x | x | x | x | x | x | x | x | x | x |
| **5.3** | Detection Quality |  |  |  | x |  |  | x | x |  |  |  |  |  |  |
| **5.4** | Vibration Intensity |  |  |  |  |  |  |  |  |  |  | x | x | x | x |
| **5.5** | Battery Life |  | x | x | x | x | x | x | x | x | x | x | x | x | x |
| **6.4** | Heat Dissipation |  | x | x | x | x | x | x | x | x | x | x | x | x | x |
| **6.6** | Skin Irritation |  |  |  |  | x |  |  |  |  |  |  |  |  |  |
| **8.3** | Adjustable Range |  |  | x |  |  |  | x | x |  |  |  |  |  |  |

**Table 13-1: Requirements Traceability Matrix by Module**

## 13.3 Hardware Interface Layer Module Mapping

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  | **Modules** | **Power** | **Range Settings** | **Detection** | **Routing** |
| **No.** | **Requirement** |  |  |  |  |  |
| **3.1** | On and Off |  | x |  |  |  |
| **3.2** | Detect Obstructions |  |  |  | x |  |
| **3.3** | Obstruction Notification |  |  |  |  | x |
| **3.4** | Battery Powered |  | x |  |  |  |
| **3.6** | Low Battery Notification |  |  |  |  | x |
| **3.7** | User Friendly |  | x | x |  |  |
| **5.2** | Real-Time Response |  | x | x | x | x |
| **5.3** | Detection Quality |  |  |  | x |  |
| **5.4** | Vibration Intensity |  |  |  |  |  |
| **5.5** | Battery Life |  | x | x | x | x |
| **6.4** | Heat Dissipation |  | x | x | x | x |
| **6.6** | Skin Irritation |  |  |  |  | x |
| **8.3** | Adjustable Range |  |  | x |  |  |

**Table 13-2: Hardware Interface Layer Module Mapping**

## 13.4 Warning Layer Module Mapping

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  | **Modules** | **Range Adjustor** | **Distance Analyzer** | **Voltage Reader** | **Battery Analyzer** |
| **No.** | **Requirement** |  |  |  |  |  |
| **3.1** | On and Off |  |  |  |  |  |
| **3.2** | Detect Obstructions |  | x | x |  |  |
| **3.3** | Obstruction Notification |  |  |  |  |  |
| **3.4** | Battery Powered |  |  |  | x | x |
| **3.6** | Low Battery Notification |  |  |  | x | x |
| **3.7** | User Friendly |  |  |  |  |  |
| **5.2** | Real-Time Response |  | x | x | x | x |
| **5.3** | Detection Quality |  | x | x |  |  |
| **5.4** | Vibration Intensity |  |  |  |  |  |
| **5.5** | Battery Life |  | x | x | x | x |
| **6.4** | Heat Dissipation |  | x | x | x | x |
| **6.6** | Skin Irritation |  |  |  |  |  |
| **8.3** | Adjustable Range |  | x | x |  |  |

**Table 13-3: Warning Layer Module Mapping**

## 13.5 Notification Layer Module Mapping

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  | **Modules** | **On**  **Pattern** | **Critical Battery Pattern** | **Object Pattern** | **Low Battery Pattern** |
| **No.** | **Requirement** |  |  |  | **Object** |  |
| **3.1** | On and Off |  | x |  |  |  |
| **3.2** | Detect Obstructions |  |  |  |  |  |
| **3.3** | Obstruction Notification |  |  |  | x |  |
| **3.4** | Battery Powered |  |  |  |  | x |
| **3.6** | Low Battery Notification |  |  |  |  | x |
| **3.7** | User Friendly |  |  |  |  |  |
| **5.2** | Real-Time Response |  | x | x | x | x |
| **5.3** | Detection Quality |  |  |  |  |  |
| **5.4** | Vibration Intensity |  | x | x | x | x |
| **5.5** | Battery Life |  | x | x | x | x |
| **6.4** | Heat Dissipation |  | x | x | x | x |
| **6.6** | Skin Irritation |  |  |  |  |  |
| **8.3** | Adjustable Range |  |  |  |  |  |

**Table 13-4: Notification Layer Module Mapping**

## 13.6 Analysis of Requirements Mapping

Based on the diagram above, the 3 requirements that every module our system need to address are not directly related to customer’s requirements, but do constrain the design of the system. The first two requirements come from the performance requirements which are the Real-Time Response and Battery life, while the third requirement deals with heat dissipation which explains why the detail design has a very linear flow and a few modules. The diagram also shows that each module is not complex since although each module does help to satisfy more than one requirement, most of the requirements are more related to the performance and safety requirements than to the customer requirements (the actual functionality of the system). The On Pattern, Object Pattern, Low Battery Pattern, and Critical Battery Pattern are the most independent modules as they do not really communicate with each other and there is little communication with other modules from other subsystems. The Routing module is now a critical module in terms of notification requirements since it is responsible for sending vibration patterns to the vibration motors to notify the user.

**14. Acceptance Plan**

**T**his section provides the details of the plan that will be followed to ensure that the Sight by Touch System will be accepted. First, the details of how the system will be packaged and how to install the system will be defined. Second, the details of the acceptance test are given. Finally, the list of acceptance criteria along with the requirements needed to fulfill each one, are listed**.**

## 14.1 Packaging and Installation

The Sight by Touch System will be delivered in a 2ft x 2ft x 2ft cardboard box. The box will include the Sight by Touch System, the System Control Device, a battery, a charger, and a user manual. The software will come preinstalled on the Arduino Microcontroller which will be inside the System Control Device. Vibration motors and Sensors will come preassembled on the Sight by Touch System.

As far as installation goes, the user will only have to insert the battery into the System Control Device. After that, the user will put on the Sight by Touch System which is composed of a top half and a bottom half. Finally, the user will connect the top half of the system to the bottom half via cables. After this quick installation, the user is ready to go.

## 14.2 Acceptance Testing

The Sight by Touch System shall be tested to ensure all required functionality as stated in the System Requirements Specification is present. Once it is confirmed that the system adheres to the specified requirements and guiding principles, the system will be considered complete and acceptable. More details on how the tests will be conducted will be provided in the System Test Plan document.

## 14.3 Acceptance Criteria

The Sight by Touch System must meet the following criteria in order to be complete and acceptable. The four criteria are listed followed by the requirements need to fulfill each criteria.

### 14.3.1 The System Shall Be Intuitive and Accessible.

* 3.1 On and Off: The system shall be able to be turned on/off by the push of a button.
* 3.4 Battery Powered: The system shall be powered by a rechargeable battery.
* 3.7 User Friendly: The system shall be user friendly so that anyone with or without sight can operate it. The functions will be intuitive and easy to learn.
* 5.6 Boot Up Time: The system shall start up in no more than 5 seconds.
* 8.2 Readability: The system shall have Braille next to the on/off button and the sensory range switch to improve utility for the visually impaired.

### 14.3.2 The System Shall Help Visually Impaired Individuals Traverse in both Indoor and Outdoor Environments

* 3.2 Detect Obstructions: The system shall be able to detect the presence of nearby objects that are in front and/or to the sides of the user. The detection area of the system will need to cover at least a 4 feet radius in front of the user including the sides and vertically from the neck to the ankles.
* 3.3 Obstruction Notification: The system shall be able to warn the user of the presence of nearby objects through the use of vibrations. The vibration’s felt by the user will correspond to the direction and the distance of the object(s) that the system has detected.
* 5.2 Real-Time Response: The system shall be able to detect objects and notify the user through vibrations between 30 milliseconds to 100 milliseconds.
* 5.3 Detection Quality: The system shall be able to detect objects less than and equal to 10 feet.
* 5.4 Vibration Intensity: The system shall have a safe vibration intensity of 0.063 m/s2 to 1.15 m/s2. This range ensures that the system can warn the user without harming the user.

### 14.3.3 The System Shall Notify The User That The Battery Is Low

* 3.6 Low Battery Notification: The system shall be able to notify the user when the battery has less than an hour left of charge which will be 20% and again when the battery has less than thirty minutes left of charge which will be 10% and a final notification when the battery is about to be completely discharged which will be at 5%..

### 14.3.4 The System Shall Be Safe To Wear

* 6.1 Wearable Material: The system shall not contain materials that could jeopardize the user’s health including, but not limited to: conductive material, allergic material, sharp objects, rusty material, etc.
* 6.2 Exposed Circuitry Protection: The system shall have no exposed wires or electrical components that will directly come into contact with the user’s skin.
* 6.3 Power Supply Protection: The system shall keep the power supply covered by a material that provides protection from any possible power leaks (ex. chemical, electrical, battery meltdown, etc.)
* 6.4 Heat Dissipation: The system shall dissipate heat produced by the components of the system to prevent overheating.
* 6.5 Water Resistance: The system shall be water resistant to light rain (precipitation rate less than 2.5 millimeters (0.098 in) per hour). This also includes sweat (32-48oz of fluid per hour).
* 6.6 Skin Irritation: The system shall not irritate the skin of the user. Vibrations should be kept at a safe range to avoid harming the skin of the user.
* 6.8 Static Electricity: The system shall not produce static electricity that would harm the user, the system, and anyone who comes into contact with the user.

# Appendix

## Contributed Libraries

Timer1

This library is a collection of routines for configuring the 16 bit hardware timer called Timer1 on the ATmega168/328/2560. This library is primarily used for the timer to read the battery at regular time intervals when the system is in use. See: <http://playground.arduino.cc/Code/Timer1>

## Formulas

### Converting microseconds from HC-SR04 Ultrasonic Sensor to inches

According to Parallax's datasheet for the PING))), there are about 74 microseconds per inch (i.e. sound travels at 1130 feet per second). This gives the distance travelled by the ping, outbound and returns, so we divide by 2 to get the distance of the obstacle. See: <http://www.parallax.com/dl/docs/prod/acc/28015-PING-v1.3.pdf>