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](#_Toc381392380)

[Document Revision History 12](#_Toc381392381)

[List of Figures 13](#_Toc381392382)

[List of Tables 15](#_Toc381392383)

[1. Introduction 17](#_Toc381392384)

[1.2 Purpose and Use 17](#_Toc381392385)

[1.3 Product Concept 18](#_Toc381392386)

[1.4 Product Scope 20](#_Toc381392387)

[2. Architecture Overview 21](#_Toc381392388)

[2.1 Hardware Interface Layer 22](#_Toc381392389)

[2.1.1 System Control Module subsystem 22](#_Toc381392390)

[2.1.2 Sensor Module(s) subsystem 22](#_Toc381392391)

[2.2 Warning Layer 22](#_Toc381392392)

[2.2.1 Battery Level Monitor subsystem 22](#_Toc381392393)

[2.2.2 Detection Filter subsystem 22](#_Toc381392394)

[2.3 Notification Layer 23](#_Toc381392395)

[2.3.1 On Notification subsystem 23](#_Toc381392396)

[2.3.2 Object Notification subsystem 23](#_Toc381392397)

[2.3.3 Battery Notification subsystem 23](#_Toc381392398)

[2.4 Module Decomposition 23](#_Toc381392399)

[2.4.1 Power 26](#_Toc381392400)

[2.4.2 Range Setting 26](#_Toc381392401)

[2.4.3 Detection 26](#_Toc381392402)

[2.4.4 Routing 26](#_Toc381392403)

[2.4.5 Range Adjuster 26](#_Toc381392404)

[2.4.6 Distance Analyzer 26](#_Toc381392405)

[2.4.7 Voltage Reader 26](#_Toc381392406)

[2.4.8 Conversion 26](#_Toc381392407)

[2.4.9 Battery Analyzer 26](#_Toc381392408)

[2.4.10 On Pattern 27](#_Toc381392409)

[2.4.11 Intensity Level 27](#_Toc381392410)

[2.4.12 Object Pattern 27](#_Toc381392411)

[2.4.13 Low Battery Pattern 27](#_Toc381392412)

[2.4.14 Critical Battery Pattern 27](#_Toc381392413)

[3. System Hardware Description 28](#_Toc381392414)

[3.1 Arduino Mega 2560 28](#_Toc381392415)

[3.1.1 Quantity 28](#_Toc381392416)

[3.1.2 Purpose 28](#_Toc381392417)

[3.1.3 Specifications 29](#_Toc381392418)

[3.1.4 Interfaces 29](#_Toc381392419)

[3.2 HC-SR04 Ultrasonic Sensor 30](#_Toc381392420)

[3.2.1 Quantity 30](#_Toc381392421)

[3.2.2 Purpose 30](#_Toc381392422)

[3.2.3 Specifications 31](#_Toc381392423)

[3.2.4 Interfaces 31](#_Toc381392424)

[3.3 SPST High-Current Mini Toggle Switch 32](#_Toc381392425)

[3.3.1 Quantity 32](#_Toc381392426)

[3.3.2 Purpose 32](#_Toc381392427)

[3.3.3 Specifications 32](#_Toc381392428)

[3.3.4 Interfaces 32](#_Toc381392429)

[3.4 DC Vibration Motor 33](#_Toc381392430)

[3.4.1 Quantity 33](#_Toc381392431)

[3.4.2 Purpose 33](#_Toc381392432)

[3.4.3 Specifications 34](#_Toc381392433)

[3.4.4 Interfaces 34](#_Toc381392434)

[3.5 Flat Ribbon Cable Roll 34](#_Toc381392435)

[3.5.1 Quantity 35](#_Toc381392436)

[3.5.2 Purpose 35](#_Toc381392437)

[3.5.3 Specifications 35](#_Toc381392438)

[3.5.4 Interfaces 35](#_Toc381392439)

[3.6 Ribbon Crimp Connector 36](#_Toc381392440)

[3.6.1 Quantity 36](#_Toc381392441)

[3.6.2 Purpose 36](#_Toc381392442)

[3.6.3 Specifications 37](#_Toc381392443)

[3.6.4 Interfaces 37](#_Toc381392444)

[3.7 Push Switch 37](#_Toc381392445)

[3.7.1 Quantity 37](#_Toc381392446)

[3.7.2 Purpose 37](#_Toc381392447)

[3.7.3 Specifications 38](#_Toc381392448)

[3.7.4 Interfaces 38](#_Toc381392449)

[3.8 Resistor 38](#_Toc381392450)

[3.8.1 Quantity 38](#_Toc381392451)

[3.8.2 Purpose 39](#_Toc381392452)

[3.8.3 Specifications 39](#_Toc381392453)

[3.8.4 Interfaces 39](#_Toc381392454)

[3.9 Solderless Plug-in BreadBoard 39](#_Toc381392455)

[3.9.1 Quantity 39](#_Toc381392456)

[3.9.2 Purpose 40](#_Toc381392457)

[3.9.3 Specifications 40](#_Toc381392458)

[3.9.4 Interfaces 40](#_Toc381392459)

[3.10 Rechargeable Battery Pack w/ Charger Cable 40](#_Toc381392460)

[3.10.1 Quantity 41](#_Toc381392461)

[3.10.2 Purpose 41](#_Toc381392462)

[3.10.3 Specifications 41](#_Toc381392463)

[3.10.4 Interfaces 41](#_Toc381392464)

[3.11 Other 41](#_Toc381392465)

[3.11.1 Computer 41](#_Toc381392466)

[3.11.2 Arduino USB Cable 41](#_Toc381392467)

[3.11.3 Custom Encasing 41](#_Toc381392468)

[3.11.4 Body Suit 42](#_Toc381392469)

[3.12 Conceptual Hardware Diagrams 42](#_Toc381392470)

[3.12.1 Voltage Divider Circuit 42](#_Toc381392471)

[3.12.2 Power Supply Connection 43](#_Toc381392472)

[3.12.3 Sensor Module Internal Connection 44](#_Toc381392473)

[4. System Software Description 45](#_Toc381392474)

[4.1 Data Description 45](#_Toc381392475)

[4.1.1 Global Variables 45](#_Toc381392476)

[4.2 System Notification Priorities 45](#_Toc381392477)

[5. Hardware Interface Layer: System Control Module 46](#_Toc381392478)

[5.1 Power 46](#_Toc381392479)

[5.1.1 Prologue 46](#_Toc381392480)

[5.1.2 Interfaces 47](#_Toc381392481)

[5.1.3 External Data Dependencies 47](#_Toc381392482)

[5.1.4 Internal Data Descriptors 47](#_Toc381392483)

[5.1.5 Service Dependencies 47](#_Toc381392484)

[5.1.6 Module Processing (Pseudo-code Algorithm) 48](#_Toc381392485)

[5.2 Range Setting 48](#_Toc381392486)

[5.2.1 Prologue 48](#_Toc381392487)

[5.2.2 Interfaces 49](#_Toc381392488)

[5.2.3 External Data Dependencies 49](#_Toc381392489)

[5.2.4 Internal Data Descriptors 50](#_Toc381392490)

[5.2.5 Service Dependencies 50](#_Toc381392491)

[5.2.6 Module Processing (Pseudo-code Algorithm) 50](#_Toc381392492)

[6. Hardware Interface Layer: Sensor Module(s) 51](#_Toc381392493)

[6.1 Detection 51](#_Toc381392494)

[6.1.1 Prologue 51](#_Toc381392495)

[6.1.2 Interfaces 52](#_Toc381392496)

[6.1.3 External Data Dependencies 52](#_Toc381392497)

[6.1.4 Internal Data Descriptors 53](#_Toc381392498)

[6.1.5 Service Dependencies 53](#_Toc381392499)

[6.1.6 Module Processing (Pseudo-code Algorithm) 54](#_Toc381392500)

[6.2 Routing 55](#_Toc381392501)

[6.2.1 Prologue 55](#_Toc381392502)

[6.2.2 Interfaces 55](#_Toc381392503)

[6.2.3 External Data Dependencies 56](#_Toc381392504)

[6.2.4 Internal Data Descriptors 56](#_Toc381392505)

[6.2.5 Service Dependencies 56](#_Toc381392506)

[6.2.6 Module Processing (Pseudo-code Algorithm) 56](#_Toc381392507)

[7. Warning Layer: Detection Filter 57](#_Toc381392508)

[7.1 Range Adjuster 57](#_Toc381392509)

[7.1.1 Prologue 57](#_Toc381392510)

[7.1.2 Interfaces 58](#_Toc381392511)

[7.1.3 External Data Dependencies 58](#_Toc381392512)

[7.1.4 Internal Data Descriptors 58](#_Toc381392513)

[7.1.5 Service Dependencies 58](#_Toc381392514)

[7.1.6 Module Processing (Pseudo-code Algorithm) 58](#_Toc381392515)

[7.2 Distance Analyzer 59](#_Toc381392516)

[7.2.1 Prologue 59](#_Toc381392517)

[7.2.2 Interfaces 59](#_Toc381392518)

[7.2.3 External Data Dependencies 59](#_Toc381392519)

[7.2.4 Internal Data Descriptors 60](#_Toc381392520)

[7.2.5 Service Dependencies 60](#_Toc381392521)

[7.2.6 Module Processing (Pseudo-code Algorithm) 60](#_Toc381392522)

[8. Warning Layer: Battery Level Monitor 61](#_Toc381392523)

[8.1 Voltage Reader 61](#_Toc381392524)

[8.1.1 Prologue 61](#_Toc381392525)

[8.1.2 Interfaces 61](#_Toc381392526)

[8.1.3 External Data Dependencies 62](#_Toc381392527)

[8.1.4 Internal Data Descriptors 62](#_Toc381392528)

[8.1.5 Service Dependencies 62](#_Toc381392529)

[8.1.6 Module Processing (Pseudo-code Algorithm) 62](#_Toc381392530)

[8.2 Conversion 63](#_Toc381392531)

[8.2.1 Prologue 63](#_Toc381392532)

[8.2.2 Interfaces 63](#_Toc381392533)

[8.2.3 External Data Dependencies 63](#_Toc381392534)

[8.2.4 Internal Data Descriptors 64](#_Toc381392535)

[8.2.5 Service Dependencies 64](#_Toc381392536)

[8.2.6 Module Processing (Pseudo-code Algorithm) 64](#_Toc381392537)

[8.3 Battery Analyzer 64](#_Toc381392538)

[8.3.1 Prologue 64](#_Toc381392539)

[8.3.2 Interfaces 65](#_Toc381392540)

[8.3.3 External Data Dependencies 65](#_Toc381392541)

[8.3.4 Internal Data Descriptors 65](#_Toc381392542)

[8.3.5 Service Dependencies 65](#_Toc381392543)

[8.3.6 Module Processing (Pseudo-code Algorithm) 66](#_Toc381392544)

[9. Notification Layer: On Notification 67](#_Toc381392545)

[9.1 On Pattern 67](#_Toc381392546)

[9.1.1 Prologue 67](#_Toc381392547)

[9.1.2 Interfaces 68](#_Toc381392548)

[9.1.3 External Data Dependencies 68](#_Toc381392549)

[9.1.4 Internal Data Descriptors 68](#_Toc381392550)

[9.1.6 System Dependencies 68](#_Toc381392551)

[9.1.7 Module Processing (Pseudo-code Algorithm) 68](#_Toc381392552)

[10. Notification Layer: Object Notification 69](#_Toc381392553)

[10.1 Intensity Level 69](#_Toc381392554)

[10.1.1 Prologue 69](#_Toc381392555)

[10.1.2 Interfaces 69](#_Toc381392556)

[10.1.3 External Data Dependencies 70](#_Toc381392557)

[10.1.4 Internal Data Descriptors 70](#_Toc381392558)

[10.1.5 Service Dependencies 70](#_Toc381392559)

[10.1.5 Module Processing (Pseudo-code Algorithm) 70](#_Toc381392560)

[10.2 Object Pattern 70](#_Toc381392561)

[10.2.1 Prologue 71](#_Toc381392562)

[10.2.2 Interfaces 71](#_Toc381392563)

[10.2.3 External Data Dependencies 71](#_Toc381392564)

[10.2.4 Internal Data Descriptors 71](#_Toc381392565)

[10.2.5 Service Dependencies 72](#_Toc381392566)

[10.2.6 Module Processing (Pseudo-code Algorithm) 72](#_Toc381392567)

[11. Notification Layer: Battery Notification 73](#_Toc381392568)

[11.1 Low Battery Pattern 73](#_Toc381392569)

[11.1.1 Prologue 73](#_Toc381392570)

[11.1.2 Interfaces 73](#_Toc381392571)

[11.1.3 External Data Dependencies 74](#_Toc381392572)

[11.1.4 Internal Data Descriptors 74](#_Toc381392573)

[11.1.5 Service Dependencies 74](#_Toc381392574)

[11.1.6 Module Processing (Pseudo-code Algorithm) 74](#_Toc381392575)

[11.2 Critical Battery Pattern 74](#_Toc381392576)

[11.2.1 Prologue 75](#_Toc381392577)

[11.2.2 Interfaces 75](#_Toc381392578)

[11.1.3 External Data Dependencies 75](#_Toc381392579)

[11.1.4 Internal Data Descriptors 75](#_Toc381392580)

[11.1.5 Service Dependencies 75](#_Toc381392581)

[11.1.6 Module Processing (Pseudo-code Algorithm) 76](#_Toc381392582)

[12. Quality Assurance 77](#_Toc381392583)

[12.1 Test Plan and Procedures 77](#_Toc381392584)

[12.2 Module/Unit Testing 77](#_Toc381392585)

[12.2.1 Hardware Interface Layer 77](#_Toc381392586)

[12.2.2 Warning Layer 78](#_Toc381392587)

[12.2.3 Notification Layer 78](#_Toc381392588)

[12.3 Component Testing 79](#_Toc381392589)

[12.3.1 Distance Sensor(s) 79](#_Toc381392590)

[12.3.2 Vibration Motor(s) 79](#_Toc381392591)

[12.3.3 System Control Module 79](#_Toc381392592)

[12.3.4 Arduino Mega2560 80](#_Toc381392593)

[12.4 Integration Testing 80](#_Toc381392594)

[12.4.1 Hardware Interface Layer 80](#_Toc381392595)

[12.4.2 Warning Layer 80](#_Toc381392596)

[12.4.3 Notification Layer 80](#_Toc381392597)

[12.5 System Verification Testing 81](#_Toc381392598)

[12.6 Test Cases 81](#_Toc381392599)

[13. Requirements Traceability 82](#_Toc381392600)

[13.1 Purpose 82](#_Toc381392601)

[13.2 Requirements Traceability Matrix By Module 82](#_Toc381392602)

[13.3 Hardware Interface Layer Module Mapping 83](#_Toc381392603)

[13.4 Warning Layer Module Mapping 84](#_Toc381392604)

[13.5 Notification Layer Module Mapping 85](#_Toc381392605)

[13.6 Analysis of Requirements Mapping 86](#_Toc381392606)

[14.1 Packaging and Installation 87](#_Toc381392607)

[14.2 Acceptance Testing 87](#_Toc381392608)

[14.3 Acceptance Criteria 87](#_Toc381392609)

[14.3.1 The System Shall Be Intuitive and Accessible. 87](#_Toc381392610)

[14.3.2 The System Shall Help Visually Impaired Individuals Traverse in both Indoor and Outdoor Environments 88](#_Toc381392611)

[14.3.3 The System Shall Notify The User That The Battery Is Low 88](#_Toc381392612)

[14.3.4 The System Shall Be Safe To Wear 88](#_Toc381392613)

[Appendix A: Arduino Libraries 90](#_Toc381392614)

[Standard Libraries 90](#_Toc381392615)

[Contributed Libraries 90](#_Toc381392616)

[Sensing: 90](#_Toc381392617)

[Timing: 90](#_Toc381392618)

[Formulas 90](#_Toc381392619)

[Converting microseconds from HC-SR04 Ultrasonic Sensor to inches 90](#_Toc381392620)

# 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.

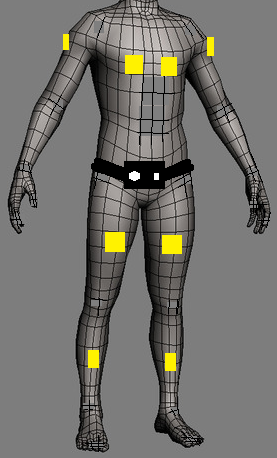
## Product Concept

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.

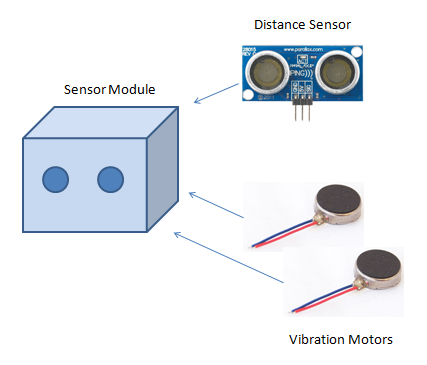
The system will be a full-blown suit with the sensors underneath the clothes and with their wires connected to the microcontroller. The sensors will be detachable to allow for flexibility and cleanliness. The system will focus on the front of the user (180 degrees, at least a 4 foot radius horizontally,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

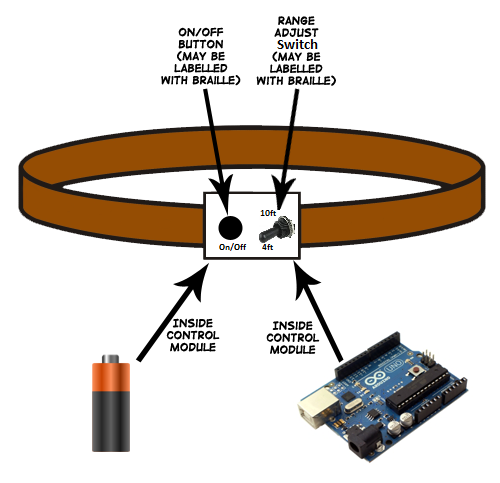
The belt will allow the user to interface with the device. The interface will have an on/off button to turn the sensors on/off. In addition, there will be a switch that allows the user to adjust the range that the sensors will detect. There will be no external elements for all the functions and data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

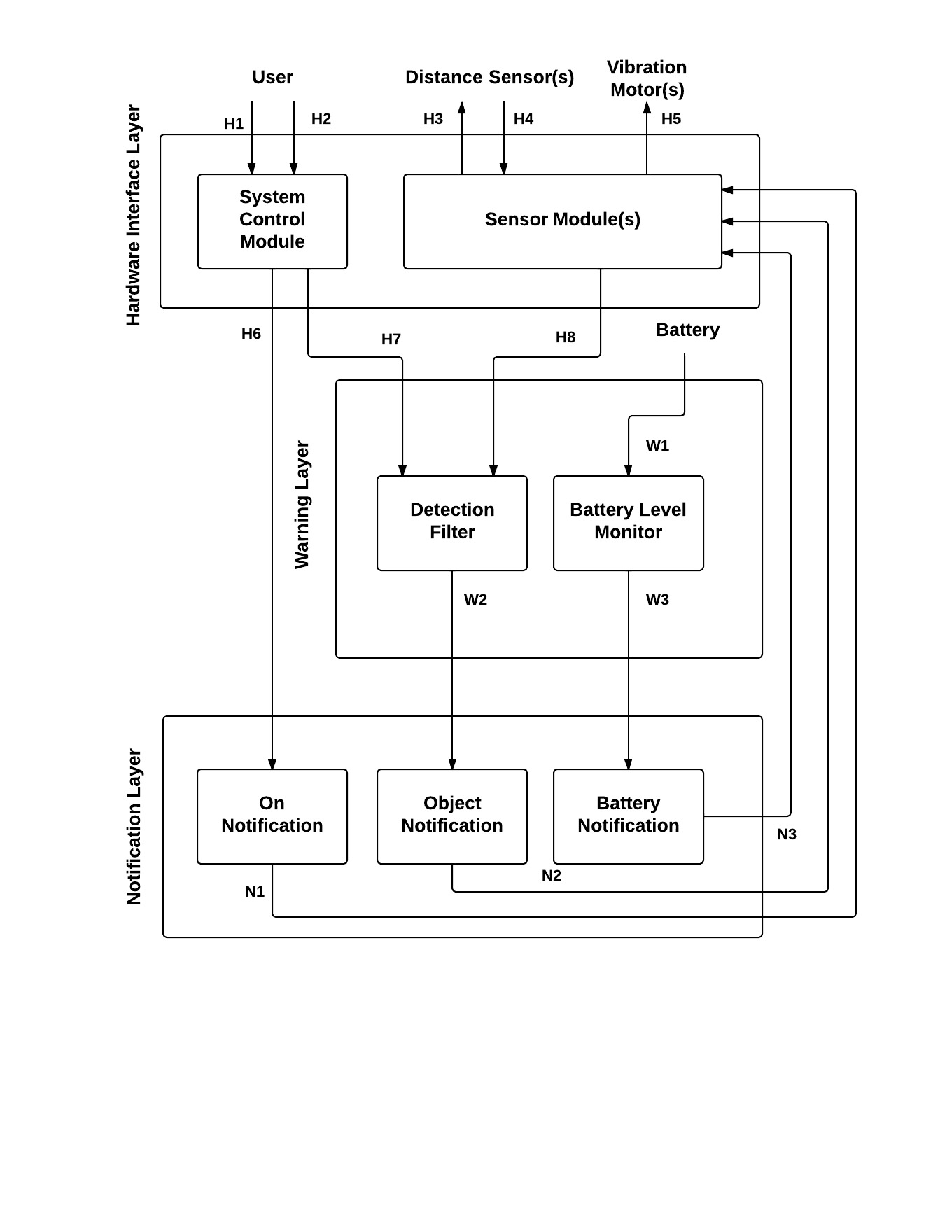
The Sight By Touch system is made to aid visually impaired users through the use of vibrations. Users of the Sight By Touch system will be able to move around and gauge where they are. When a user walks near an object, the system will vibrate in the direction where the object is closest. This warns the 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

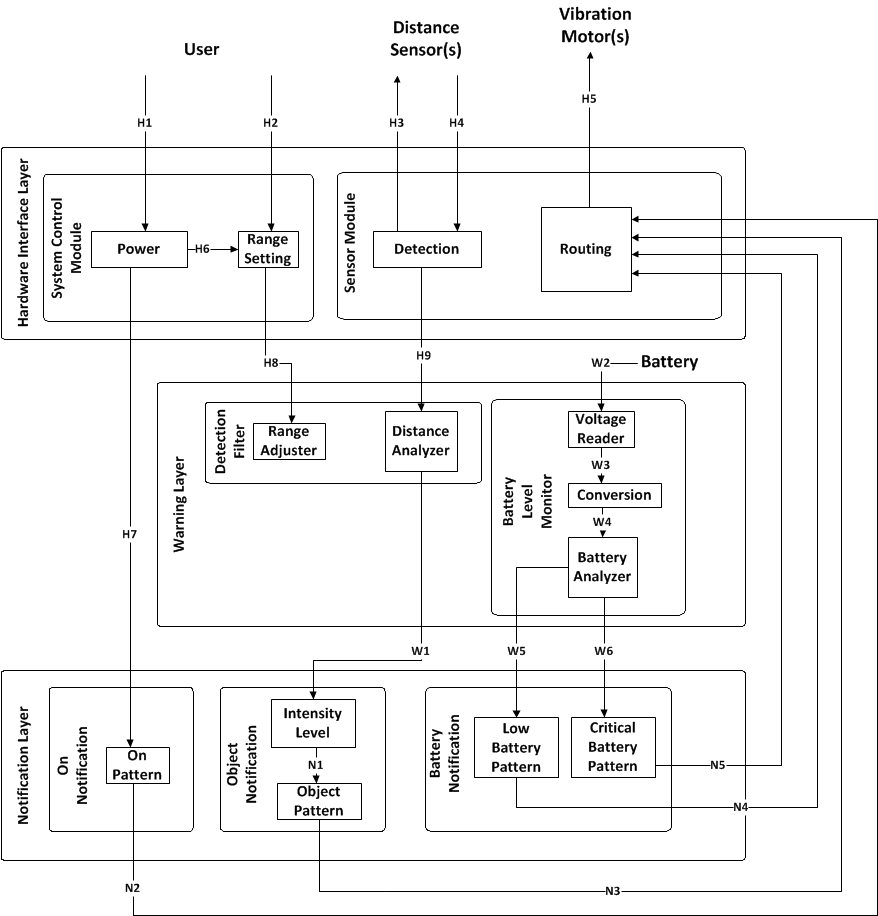
The Object Notification subsystem will receive messages from the Detection Filter subsystem whenever the system has detected a significant object and interpret them into appropriate instructions to relay to the Sensor Module(s) for an Object Detect Notification.

### 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.



**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 signal to the ultrasonic sensor to operate** | | **Detection** | **Distance Sensor(s)** |
| ***H4.*** | **A response that contains the time taken to reach the detected object from the sensor that detected it** | | **Detection** | **Sensor Module(s)** |
| ***H5.*** | **A 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 generate an On Notification to send to the Sensor Modules** | | **Power** | **On Notification** |
| ***H8.*** | **The new maximum detection range** | | **Range Setting** | **Range Adjuster** |
| ***H9.*** | **The ID of the sensor that detected an object and the distance to that object from the sensor** | | **Detection** | **Distance Analyzer** |
| ***W1.*** | **The ID of the vibration motor associated with the sensor that detected an object and the distance to that object from the sensor** | | **Distance Analyzer** | **Intensity Level** |
| ***W2.*** | **A voltage from the battery powering the system** | | **Battery** | **Voltage Reader** |
| ***W3.*** | **The voltage of the battery as digital data** | | **Voltage Reader** | **Conversion** |
| ***W4.*** | **The current voltage provided by the battery as a percentage of its full voltage capacity** | | **Conversion** | **Battery Analyzer** |
| ***W5.*** | **A call to the Low Battery Pattern module to generate a Low Battery Notification to the Sensor Modules** | | **Battery Analyzer** | **Low Battery Pattern** |
| ***W6.*** | **A call to the Critical Battery Pattern module to generate a Critical Battery Notification to the Sensor Modules** | | **Battery Analyzer** | **Critical Battery Pattern** |
| ***N1.*** | **The type of vibration pattern and the vibration motor id associated with the sensor that detected an object** | | **Intensity Level** | **Object Pattern** |
| ***N2.*** | **A unique vibration pattern and the vibration motor id representing all vibration motors for an On Notification** | | **On Pattern** | **Sensor Module(s)** |
| ***N3.*** | **A unique vibration pattern and the vibration motor id associated with the sensor that detected an object for an Object Detected Notification** | | **Object Pattern** | **Sensor Module(s)** |
| ***N4.*** | **A unique vibration pattern and the vibration motor id representing all vibration motors for a Low Battery Notification** | | **Low Battery Pattern** | **Sensor Module(s)** |
| ***N5.*** | **A unique vibration pattern and the vibration motor id representing all vibration motors for a Critical Battery Notification** | | **Critical Battery Pattern** | **Sensor Module(s)** |

**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 range adjuster 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 parsing the vibration pattern contained from the Notification Layer and sending that parsed pattern to the correct vibration motor(s) using the vibration motor id in the notification message 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 range 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 maximum detection range.

### 2.4.7 Voltage Reader

Voltage Reader focuses on acquiring a voltage reading from the battery powering the system.

### 2.4.8 Conversion

Conversion focuses on converting the voltage reading from Voltage Reader into a percentage that can be analyzed.

### 2.4.9 Battery Analyzer

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

### 2.4.10 On Pattern

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

### 2.4.11 Intensity Level

Intensity Level focuses on converting the distance to a corresponding intensity and sending it along with the id of the sensor that detected the object to Object Pattern.

### 2.4.12 Object Pattern

Object Pattern focuses on the actual creation of the pattern using the results of Intensity Level and sends a that pattern along with the id of the vibration motor corresponding to the sensor that detected the object to the Routing to give actual feedback.

### 2.4.13 Low Battery Pattern

Low Battery Pattern will be responsible for creating the vibration pattern that will represent low power left and sends a vibration pattern and the id representing all vibration motors to Routing to give actual feedback.

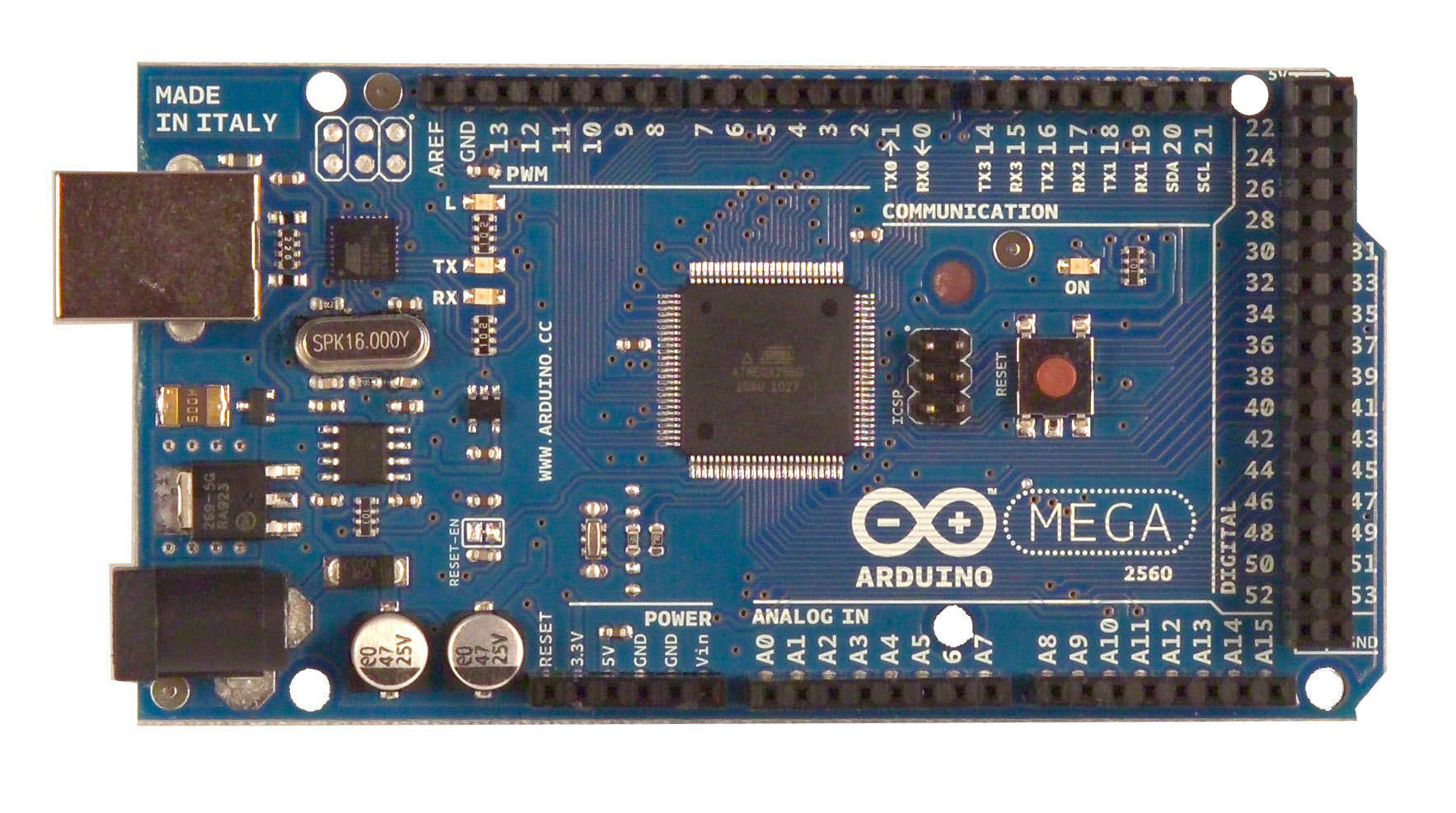
### 2.4.14 Critical Battery Pattern

Critical Battery Pattern will be responsible for creating the vibration pattern that will represent the system turning off from insufficient battery charge and sends it 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:

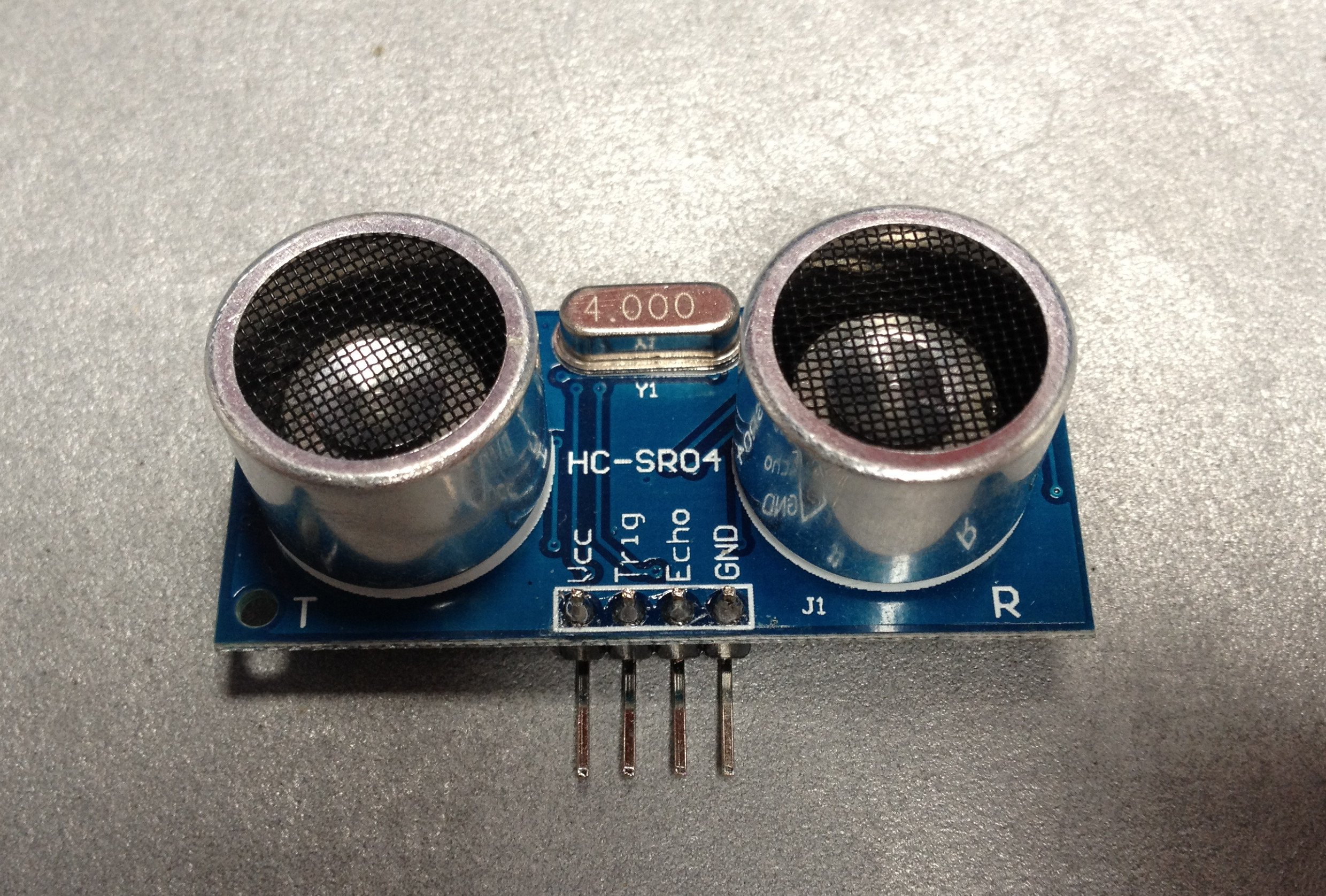
|  |  |
| --- | --- |
| Microcontroller | ATmega2560 |
| Operating Voltage | 5V |
| Input Voltage (recommended) | 7-12V |
| Input Voltage (limits) | 6-20V |
| Digital I/O Pins | 54 (of which 15 provide PWM output) |
| Analog Input Pins | 16 |
| DC Current per I/O Pin | 40 mA |
| DC Current for 3.3V Pin | 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 knob, all distance sensors, all vibration motors, and the external battery. The distance sensors and vibration motors will communicate with the Arduino Mega 2560 through its digital pins. Any digital pin (22 – 53 on the Arduino) may be used as long as it’s calibrated correctly in the software (explained in the System Software Description section). The range knob and external battery will be connected to the Arduino through its analog pin. As with digital pins, any analog pin (A0 – A15 on the Arduino) may be used as long as it’s calibrated correctly in the software (explained in the System Software Description section). The external battery will also be connected to the Arduino’s power jack. The power button will be connected to the Vin pin to allow or stop voltage flowing from the external battery to the Arduino.

## 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. Objects detected will give an echo wave. The sensors will read this echo and convert it into time taken to reach the detected object from the sensor itself in microseconds. Then, the sensors will send this data to the Arduino Mega 2560 for processing. One 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 Min Range 2cm | 40Hz MeasuringAngle 15 degree |
| Max Range | 4m |
| 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 through its digital pins. Any digital pin (22 – 55 on the Arduino) 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 one range setting toggle.

### 3.3.2 Purpose

The purpose of the SPST High-Current Mini Toggle Switch is 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

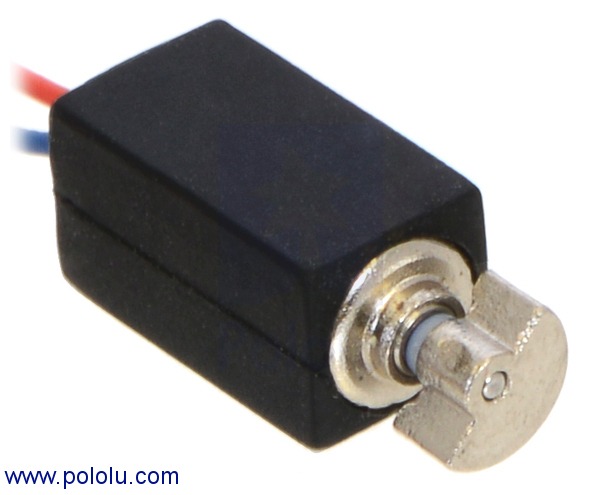
|  |  |
| --- | --- |
| Current Rating | 10 A |
| Voltage Rating | 125 V |
| Mounting Hole Size | ¼ in |

**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 through its digital pins. Any digital pin may be used as long as the corresponding pin is used in the software.

## 3.4 DC Vibration Motor



**Figure 3-4: DC 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 (one for each sensor). More may be used if a stronger vibration intensity is desired.

### 3.4.2 Purpose

The purpose of the DC 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. A DC vibration motor (or more if desired) is part of a Sensor Module.

### 3.4.3 Specifications

The DC Vibration Motor contains the following specifications:

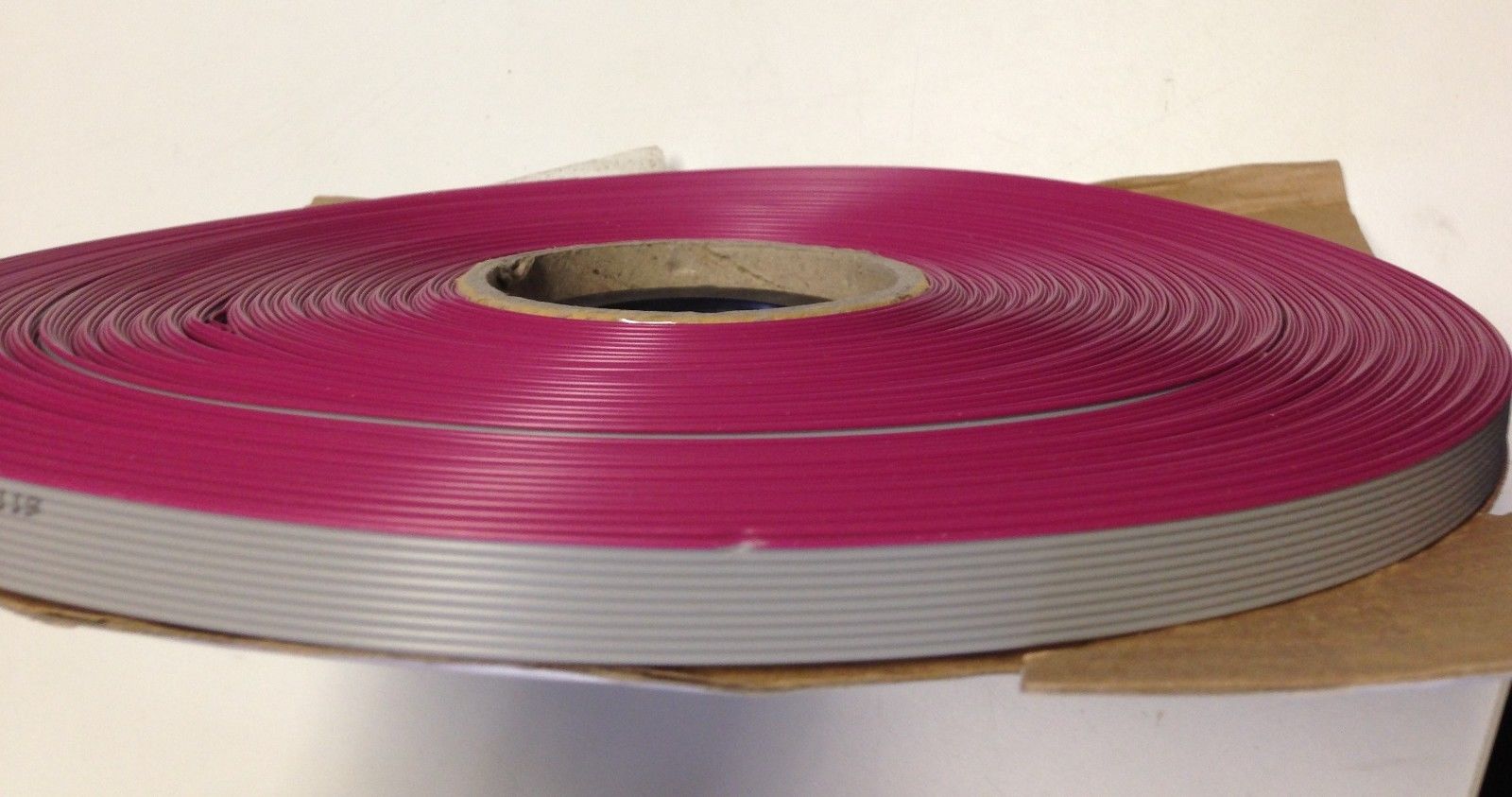
|  |  |
| --- | --- |
| Size | 11.6 × 4.6 × 4.8 mm |
| Weight | 0.8 g |
| Free-run speed @ 3V | 12500 rpm |
| Operating Voltage Range | 2.4 – 3.5 V |

**Table 3-4: DC Vibration Motor Specifications**

### 3.4.4 Interfaces

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

## 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 to the 8 Sensor Modules. It is estimated that a 12ft. long Flat Ribbon Cable Roll is an adequate amount to implement all these connections.

### 3.5.2 Purpose

The purpose of the Flat Ribbon Cable is to provide a method of communication between the Arduino and the Sensor Modules. The design of this component will allow the multiple pins the Arduino uses to connect with the pins of a Sensor Module to be contained in one cable. This will minimize any tangling of wires among other Ribbon Cables. The Flat Ribbon Cables will not directly connect to the hardware components. Smaller wires will be cut off from the Ribbon Cable to connect it to a Breadboard (described later) holding the actual hardware components with the help of a Ribbon Crimp Connecter (also described later). These connections are to supply power to the system from an external battery and to connect the Ribbon Crimp Connector to actual the pins of the internal hardware components such the sensors and vibration motors of a Sensor Module. 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

Pieces of the Flat Ribbon Cable will be used in all connections from the Arduino to the Sensor Modules. Smaller wires cut from the Ribbon Cable will be used to connect the external battery power supply to the Arduino and for connecting specific pins of the cable to the internal hardware components. These components are the vibration motors, sensors, and the actual pins of the Arduino.

## 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 to the Sensor Modules will require a Ribbon Crimp Connector (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 for a more organized way of connecting the Arduino to the Sensor Modules and vice versa through the Flat Ribbon Cables.

### 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 smaller wires used to connect to the Breadboard (described later) holding the actual hardware components.

## 3.7 Push Switch

## http://www.ece.com.tw/upload/image/products/p-list-PushSwitch.jpg

**Figure 3-7: Push Switch**

### 3.7.1 Quantity

The Sight By Touch system will only require 1 Push Switch.

### 3.7.2 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 the Push Switch is 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.

### 3.7.3 Specifications

The Push Switch contains the following specifications:

|  |  |
| --- | --- |
| Size | 8 X 8 mm |

**Table 3-7: Push Switch Specifications**

### 3.7.4 Interfaces

The Push Switch will be connected between the external battery and the Vin pin of the Arduino. When the Push Switch is pressed, it will close the circuit and allow power to flow from the battery to the Arduino. When the Push Switch is pressed again, it will open the circuit and stop the power flow from the battery into the Arduino.

## 3.8 Resistor



**Figure 3-8: Resistor**

### 3.8.1 Quantity

The Sight By Touch system will use a voltage divider to measure the power remaining in the external battery. This requires 2 resistors.

### 3.8.2 Purpose

The purpose of the Resistors is be used in a voltage divider to reduce the voltage of the external battery into the 5V operating range of the Arduino analog pins. This will allow the Arduino to safely read the voltage of the external battery.

### 3.8.3 Specifications

The Resistor contains the following specifications:

|  |  |
| --- | --- |
| Resistance | 10 k ohms |
| Composition | Metal Film |
| Power | 0.25 W |
| Tolerance | ± 5% |

**Table 3-8: Resistor Specifications**

### 3.8.4 Interfaces

The Resistors will be connected between the external battery and an analog pin on the Arduino. Any analog pin (A0 – A15 on the Arduino) can be used as long as the corresponding pin is used in the software.

## 3.9 Solderless Plug-in BreadBoard

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

**Figure 3-9: Solderless Plug-in BreadBoard**

### 3.9.1 Quantity

The 8 Sensor Modules and the Arduino will each require a Solderless Plug-in BreadBoard for connections. So, 9 Solderless Plug-in BreadBoards will be required.

### 3.9.2 Purpose

The purpose of the Solderless Plug-in BreadBoards is to allow all hardware components (vibration motors, sensors, Arduino, external battery, Push Switch, Resistors, wires and Ribbon Cables) to be connected to the Sight By Touch system’s circuitry. The Sensor Modules and the Arduino will use a BreadBoard for their connections.

### 3.9.3 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 3-9: Solderless Plug-in BreadBoard Specifications**

### 3.9.4 Interfaces

The external battery, vibration motors, sensors, Push Switch, Resistors, and wires connected to the Ribbon Cables will be directly connected to a BreadBoard.

## 3.10 Rechargeable Battery Pack w/ Charger Cable

## http://ecx.images-amazon.com/images/I/41nYEtMES4L.jpg

**Figure 3-10: Rechargeable Battery Pack w/ Charger Cable**

### 3.10.1 Quantity

Only 1 Rechargeable Battery Pack is needed to power the Sight By Touch System.

### 3.10.2 Purpose

The purpose of the Rechargeable Battery Pack is to provide a rechargeable power supply to the Arduino to power the Sight By Touch System. A compatible charger cable will be packaged with the Battery Pack to allow the user to recharge the battery with a wall outlet.

### 3.10.3 Specifications

The Rechargeable Battery Pack contains the following specifications:

|  |  |
| --- | --- |
| Voltage | 12V |

**Table 3-10: Rechargeable Battery Pack Specifications**

### 3.10.4 Interfaces

The Rechargeable Battery Pack will be connected directly to the BreadBoard associated with the Arduino where it will connect with the Vin pin of the Arduino to supply power and connect with the voltage divider circuit that will allow the Arduino to measure voltage.

## 3.11 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.

### 3.11.1 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.

### 3.11.2 Arduino USB Cable

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

### 3.11.3 Custom Encasing

Custom Encasing for the Sensor Module (vibration motors, sensors, breadboard) and the System Control Module (Arduino, 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.

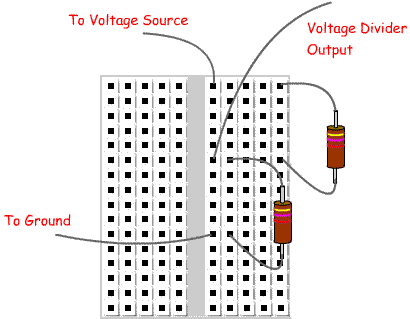
### 3.11.4 Body Suit

A Body Suit is required to allow the user to comfortably wear the Sight By Touch system and experience safe haptic feedback.

## 3.12 Conceptual Hardware Diagrams

This subsection will describe the concepts behind a few of the complicated hardware connections

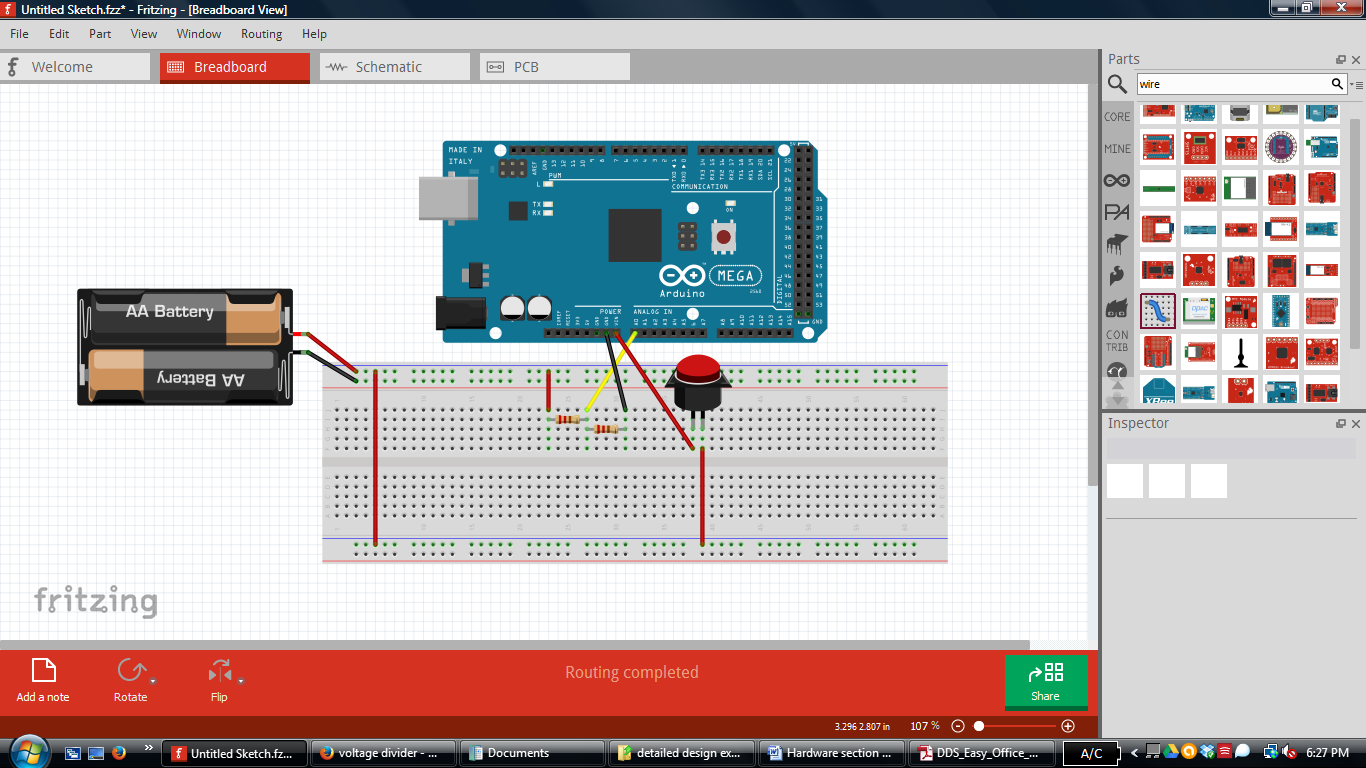
### 3.12.1 Voltage Divider Circuit



**Figure 3-11: 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.

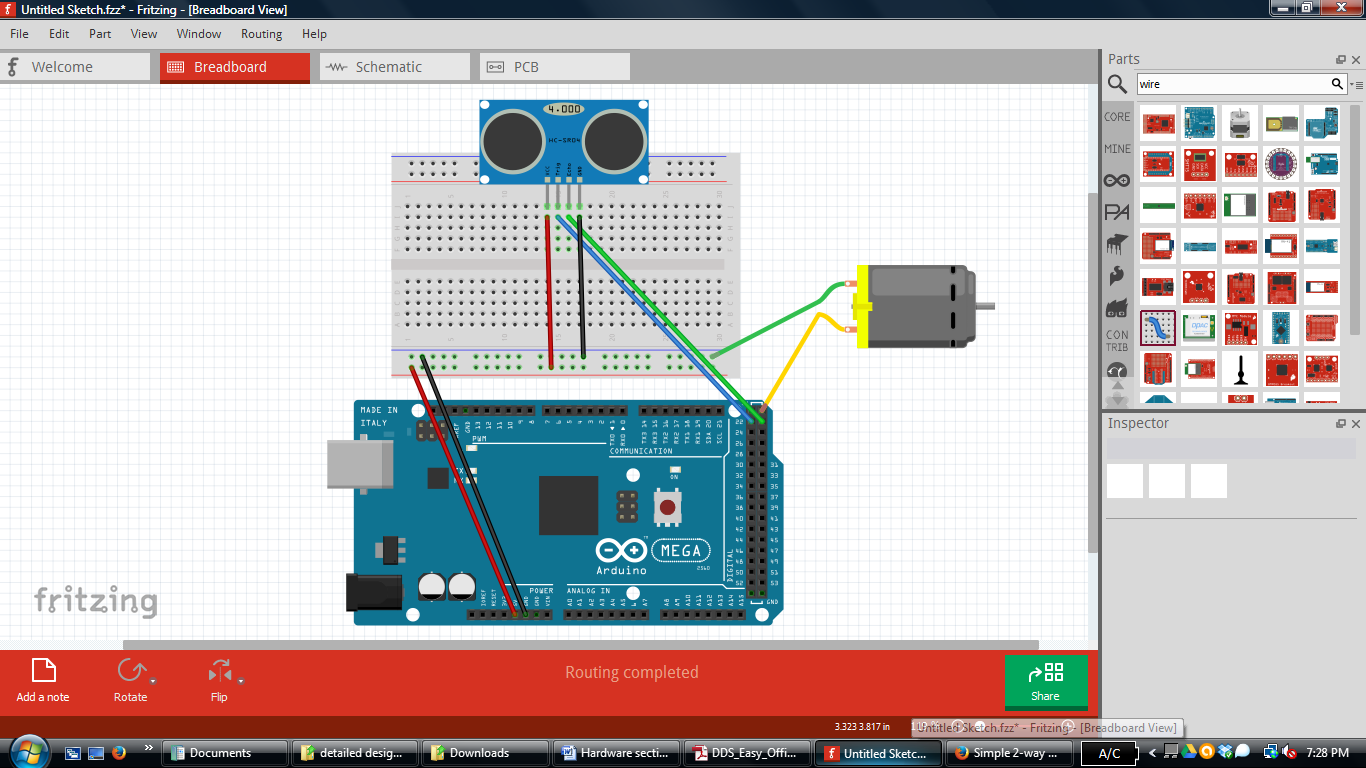
### 3.12.2 Power Supply Connection



**Figure 3-12: 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 resistors comprise the voltage divider circuit. The red wires represent the full voltage flow from the battery to the Arduino 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. 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, 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, turning it off. The yellow wire represent the flow from the voltage divider circuit to an Arduino analog pin. This connection is what allows the Arduino to safely read the voltage from a battery whose voltage exceeds 5V operating range of the Arduino. Finally, the black wires represent the ground flow to complete the circuit connection.

### 3.12.3 Sensor Module Internal Connection



**Figure 3-13: 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 to the BreadBoard are to supply power and ground connections to the BreadBoard, which can be used by the HC-SR04 Ultrasonic Sensor and DC Vibration Motor pictured. The use of these connections are 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 to the sensors are for the Arduino 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 to the DC Vibration Motor is for the Arduino 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.

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

This section will describe the overall software design of the Sight By Touch system. The system will be 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.) and 120 (120 in. =10 ft.) 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.

## 4.2 System Notification Priorities

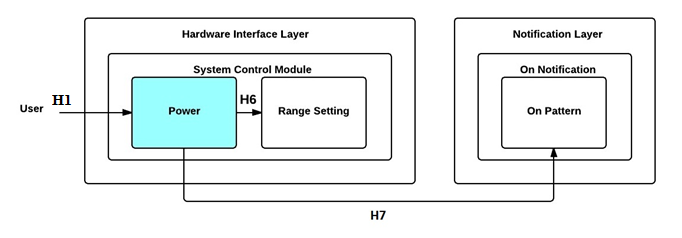
The Sight by Touch System will use interrupts to mimic the importance of each notification.

* 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 preparing the Sight By Touch system for signaling the system is turning on or off 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.

### 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 a “power 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 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 use 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.
* attachInterrupt() – an Arduino function that is use to Specifies a named Interrupt Service Routine (ISR) to call when an interrupt occurs.

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

**void** SystemControlModulePower**(void)**

**{**

//Initialize the global variables

//Initialize battery interrupt

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

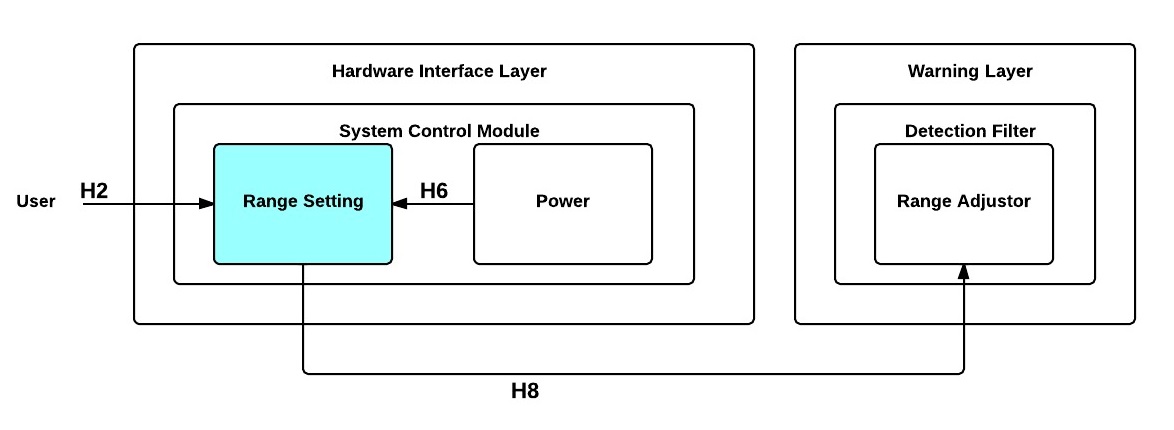
maximum detection range of the system\*/

SystemControlModuleRangeSetting**();**

//call the On Pattern module

**}**

## 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 signal will be generated by the user flips the switch on the control belt to a certain position and sent to Range Setting for translation. |
| Power | Range Setting | A call to update the maximum detection range when the system is turned on. |
| Range Setting | Range Adjuster | The new maximum detection range |

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

### 5.2.3 External Data Dependencies

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| switch\_position | int | An integer value indicating the position of the toggle switch on the control box |

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

### 5.2.4 Internal Data Descriptors

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| new\_max\_range | int | An integer value indicating the new maximum detection range |
| INDOOR\_RANGE | int | An integer constant that represents the indoor maximum detection range of the system in inches. The value of the constant is 48 |
| OUTDOOR\_RANGE | int | An integer constant that represents the outdoor maximum detection range of the system in inches. The value of the constant 120 |
| 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. |
| SWITCH\_PIN | int | The pin number the toggle switch of the control box will be connected to in the Arduino. |

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

### 5.2.5 Service Dependencies

* analogRead() - an Arduino function that reads the value from the specified analog pin.
* map() – an Arduino function that will map a value from a range of numbers to a customized range of numbers for ease of analysis.

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

**void** SystemControlModuleRangeSetting**(void)**

**{**

/\*obtain position value of toggle switch\*/

knob\_position **=** **digitalRead(**SWITCH\_PIN**);**

**if(**knob\_position **==** LOW**)**

**{**

/\*send new maximum detection range for indoors to Detection Filter Range

Adjuster module\*/

DetectionFilterRangeAdjuster**(**INDOOR\_RANGE**);**

**}**

**else** /\*only 2 detection ranges available, this must be the outdoor range\*/

**{**

/\*send new maximum detection range for outdoors to Detection Filter Range

Adjuster module\*/

DetectionFilterRangeAdjuster**(**OUTDOOR\_RANGE**);**

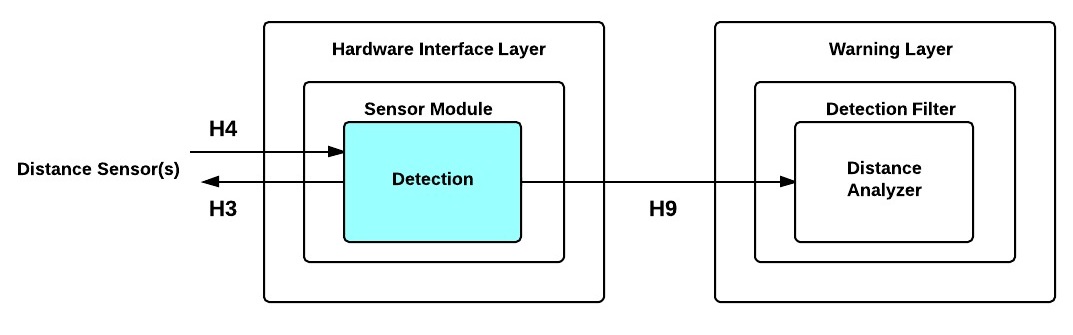
**}**

**}**

# ****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. 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 response that represents the time (in microseconds) it took for the wave to reach the detected object. |
| Detection | Distance Analyzer | The translated time 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** |
| travel\_time | long | The time in microseconds to reach the nearby detected object from the distance sensor that detected it. |

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

### 6.1.4 Internal Data Descriptors

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| trig\_sensor\_ids | int[] | The number of the pin a trigger pin from a distance sensor is connected to in order for the Arduino to send a signal to the sensor to fire an ultrasonic wave.  An array of integers that will hold all of the pin numbers for triggering each distance sensor. |
| echo\_sensor\_ids | int[] | The number of the pin an echo pin from a distance sensor is connect to in order for the Arduino to receive a digital value from the sensor regarding distance to a detected object.  An array of integers that will hold all of the pin numbers for receiving the response of each distance sensor. |
| sensor\_id | int | The id of the sensor that detected the object |
| distance | int | The distance in inches to reach the nearby detected object from the distance sensor that detected it. |
| 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.
* digitalWrite() - an Arduino function that writes a specified value to a specified digital pin.
* 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).
* loop() - an Arduino function that will be used to actively control the Arduino board.

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

**void** SensorModuleDetection**(void)**

**{**

**for** every id in the trig\_sensor\_ids

**{**

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

**digitalWrite(**trig\_sensor\_id**,** LOW**);** /\*clear pin state\*/

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

**digitalWrite(**trig\_sensor\_id**,** HIGH**);** /\*fire ultrasonic wave\*/

**delayMicroseconds(**5**);** /\*give time to make sure wave is fired properly\*/

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

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

travel\_time **=** **pulseIn(**echo\_sensor\_id**,** HIGH**);**

/\*initialize its data members\*/

sensor\_id **=** echo\_sensor\_ids**[];**

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

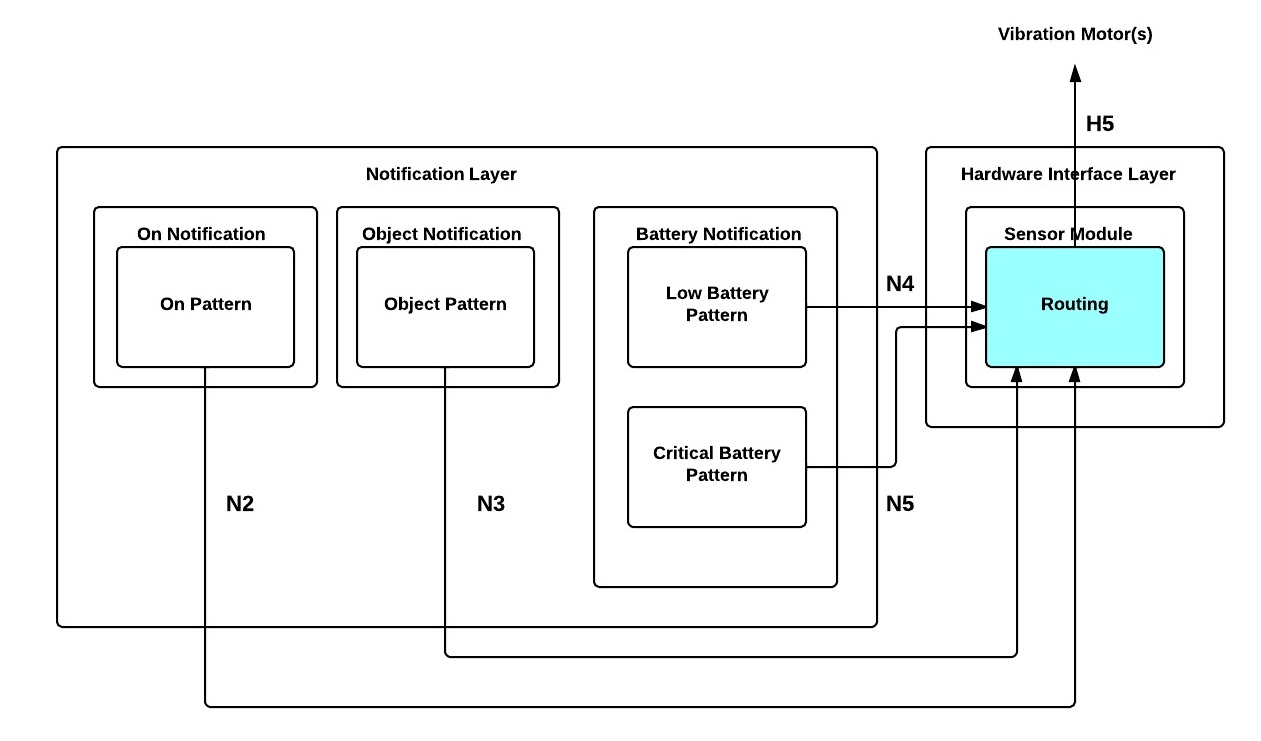
distance **=** travel\_time**/**74**/**2**;**

//send sensor\_id and distance to Detection Filter Analyze module

**}**

**}**

## 6.2 Routing



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

### 6.2.1 Prologue

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

### 6.2.2 Interfaces

|  |  |  |
| --- | --- | --- |
| **Producer** | **Consumer** | **Description** |
| Low Battery Pattern | Routing | The pattern and id of vibration motor |
| Critical Battery Pattern | Routing | The pattern and id of vibration motor |
| 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 | string | The pattern will be a list of numbers representing the delay in microseconds between turning the vibration on and off. The delimiter used will be a semicolon. |
| 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 parameter.
* digitalWrite() - an Arduino function that Writes the value to a specified digital pin.

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

**void** SensorModuleRouting**(**String vibration\_pattern**,** **int** vibration\_id**)**

**{**

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

**{**

//parse vibration pattern

//send parsed pattern to every vibration motor

**}**

**else**

**{**

//use switch statement to find specified vibration motor

//if match, call digitalWrite() to send signal to specified vibration //motor

//default, do nothing

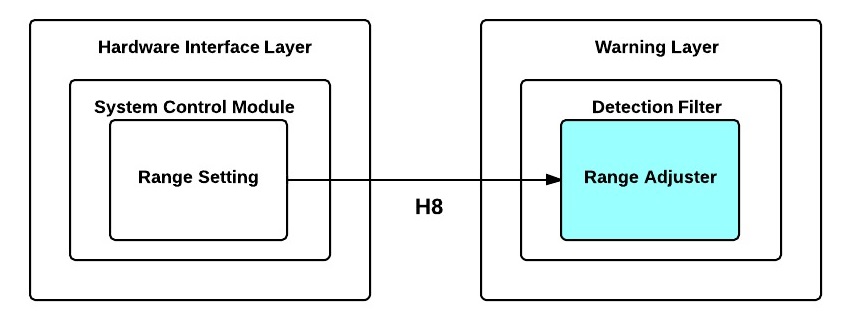
**}**

**}**

# ****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 maximum detection range setting to make decisions. If the distance is within the range, the data will be forwarded to the Warning Manager Obstruction Detection module for processing. The maximum detection range setting 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 | The new maximum detection range 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 | The new detection range that the system will use. |

**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)

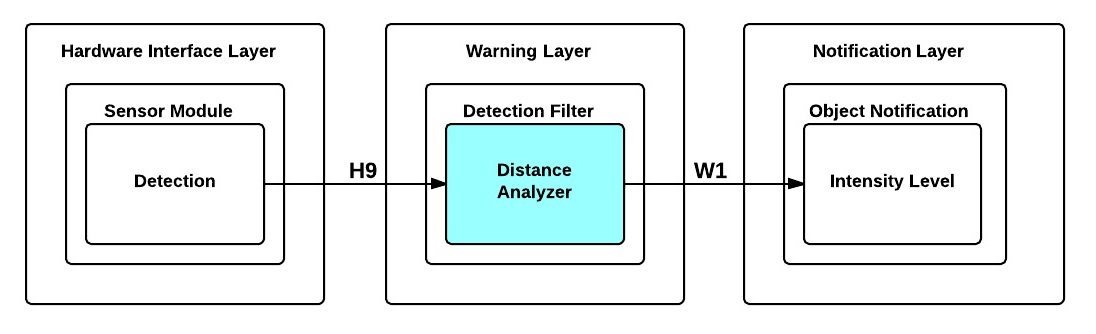
void DetectionFilterRangeAdjuster**(**int new\_max\_detect\_range**)**

**{**

//assign the maximum detection range value from input to (variable)

**}**

## 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 maximum detection range setting of the Sight By Touch system by comparing the distance data sent as input to that current maximum 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 Intensity Level module if the detected object is within the current maximum detection range. Otherwise, it will ignore it.

### 7.2.2 Interfaces

|  |  |  |
| --- | --- | --- |
| **Producer** | **Consumer** | **Description** |
| Detection | Distance Analyzer | The translated time from the Distance Sensor to inches and the id of the sensor that detected the object. |
| Distance Analyzer | Intensity Level | The id of the sensor that detected the object and the distance to the object. |

**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** |
| max\_detect\_range | int | A global variable representing the maximum distance in inches that an object should be from the user. The value could be 48 (if the system is in indoor mode) or 120 (if the system is in outdoor mode). |
| sensor\_id | int | The Id of the sensor that detected the object |
| distance | int | The distance to the object detected in inches. |
| min\_detect\_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** DetectionFilterDistanceAnalyzer**(int** distance**,** **int** sensor\_id**)**

**{**

**if(**min\_detect\_range **<=** distance **<=** max\_detect\_range**)**

**{**

//send distance and sensor\_id to Intensity Level: Obstruction Detection

**}**

**else**

**{**

//ignore detected object

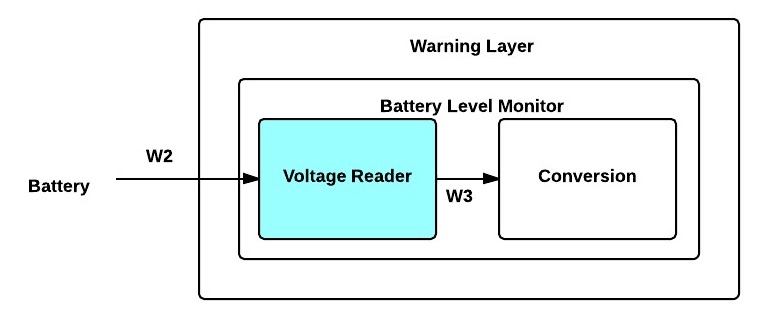
**}**

**}**

# ****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 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, Conversion 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 at regular time intervals. This module takes the voltage of the battery as input. It passes a direct reading of the battery voltage to Conversion for translation.

### 8.1.2 Interfaces

|  |  |  |
| --- | --- | --- |
| **Producer** | **Consumer** | **Description** |
| Voltage Reader | Conversion | A reading of the battery in volts (the voltage range is 0-5V) from an analog pin |

**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. |

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

### 8.1.4 Internal Data Descriptors

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| voltage | int | An integer representing the voltage of the battery |
| analogLow | int | The minimum analog number the battery can read. The value will be 0 |
| analogHigh | int | The maximum analog number the battery can read. The value will be 922 |
| digitalLow | int | The minimum digital number the battery can read. The value will be 0 |
| digitalHigh | int | The maximum digital number the battery can read. The value will be 1200 |

**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.
* map() – an Arduino function that will map a value from a range of numbers to a customized range of numbers for ease of analysis.

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

**void** BatteryLevelMonitorVoltageReader**(void)**

**{**

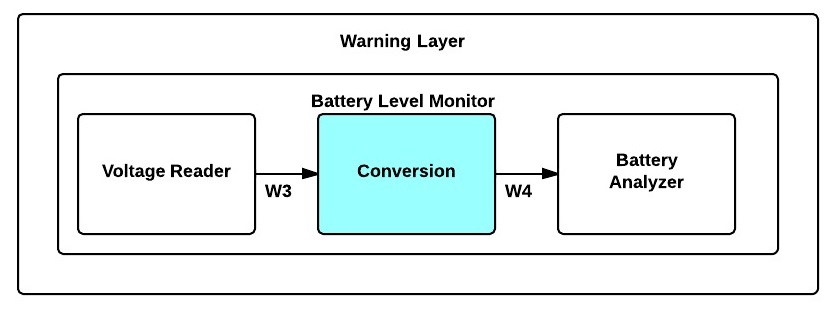
voltage**=map(analogRead(**power\_pin\_number**),**analogLow**,**analogHigh**,**

digitalLow**,**digitalHigh**)**

//send the voltage reading to Battery Level Monitor Convert module

**}**

## 8.2 Conversion



**Figure 8-2: Conversion Module Diagram**

### 8.2.1 Prologue

Conversion is responsible for converting voltage readings from the Voltage Reader into a percentage that can be analyzed. This module accepts the voltage readings from the Voltage Reader as input and passes the voltage reading as a percentage to the Battery Analyzer for analysis.

### 8.2.2 Interfaces

|  |  |  |
| --- | --- | --- |
| **Producer** | **Consumer** | **Description** |
| Voltage Reader | Conversion | A reading of the battery in volts |
| Conversion | Battery Analyzer | A voltage reading as a percentage. |

**Table 8-4: Conversion Module Interfaces**

### 8.2.3 External Data Dependencies

This module does not have any external data dependencies.

### 8.2.4 Internal Data Descriptors

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| Charge\_level | float | A float representing the voltage of the battery as a percentage. |
| Voltage\_reading | int | An integer representing the voltage of the battery. |

**Table 8-5: Conversion Module Internal Data Descriptors**

### 8.2.5 Service Dependencies

This module does not have any service dependencies.

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

**void** BatteryLevelMonitorConvert**(int** voltage\_reading**)**

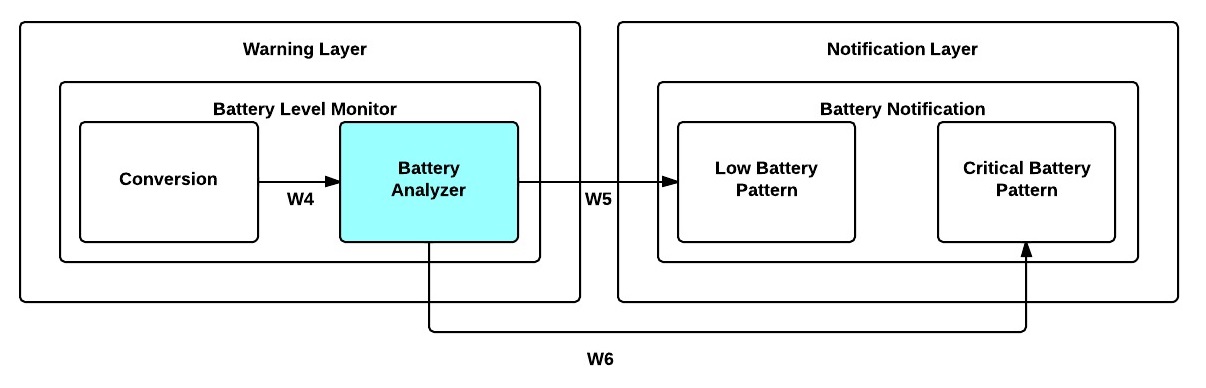
**{**

//convert the voltage reading into a percentage

//send Battery Level percentage to the Monitor Analyze module

**}**

## 8.3 Battery Analyzer



**Figure 8-3: 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 Conversion 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** |
| Conversion | 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 8-6: 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. |
| Voltage\_reading | int | An integer representing the voltage of the battery. |
| CRITICAL\_LOW | float | The minimum battery percentage that the system requires to operate. The value will be .05. |
| BATTERY\_LOW | float | The battery percentage indicating that a recharge is recommended. The value will be .20. |

**Table 8-7: 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** BatteryLevelMonitorBatteryAnalyzer**(float** percentage**)**

**{**

**if((**percentage **<=** BATTERY\_LOW**)** **and** **(**low\_flag**==**0**))**

**{**

low\_flag **=** 1**;**

//call Low Battery Pattern module

**}**

**else** **if((**percentage **<=** CRITICAL\_LOW**)** **and** **(**critical\_low\_flag**==**0**))**

**{**

critical\_low\_flag **=** 1**;**

//call Critical Battery Pattern module

**}**

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

**{**

//do nothing

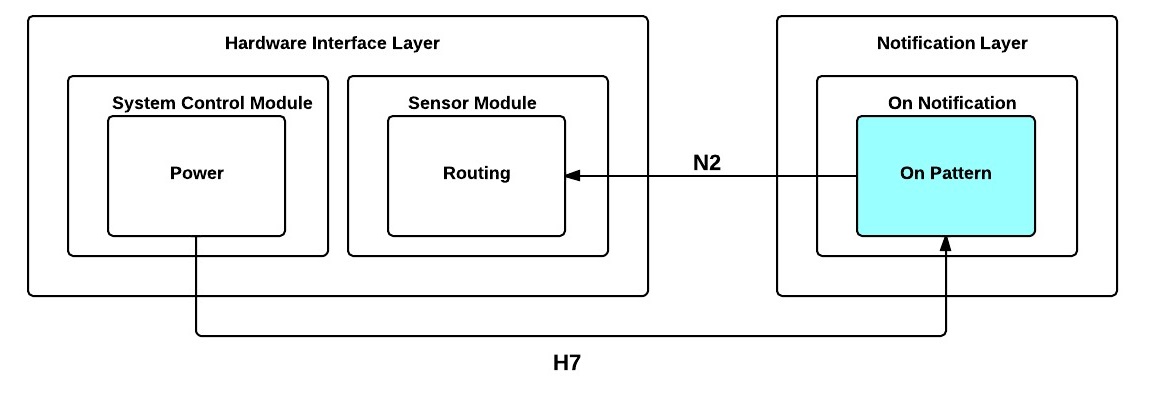
**}**

**}**

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

The On/Off Notification subsystem is responsible for generating the on and off 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 vibration pattern 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 modules that constitute the entire subsystem. These modules are 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 that the system will use to notify the user that the system has been turned on. The vibration pattern will have a predefined activation duration, 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 string 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 a “power on” notification is required. |
| On Pattern | Routing | The vibration pattern and id of the vibration motor for translation. |

**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 | String | The vibration pattern that represents the system turning on. The pattern will contain integers (representing the delay in microseconds) separated by a semicolon. |
| 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. |

**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** OnVibrationPattern**()**

**{**

/\*initialize vibration pattern\*/

**int** Vibration\_id **=** **-**1**;**

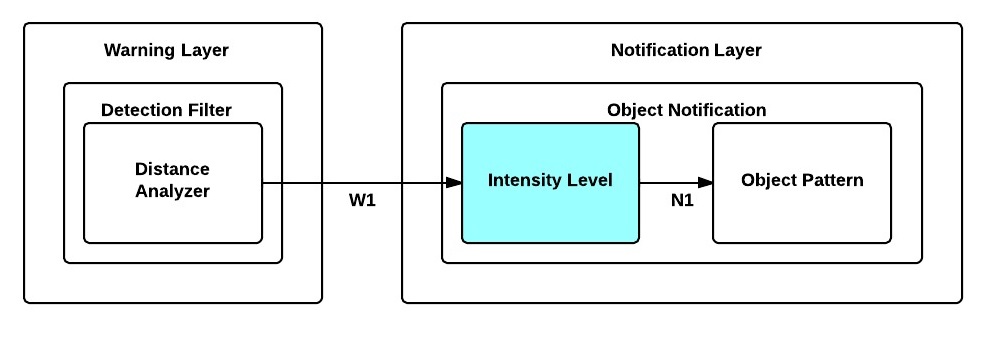
//Send vibration\_id and vibration\_pattern to the Routing module

**}**

# ****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 modules that constitute the entire subsystem. These modules are the Intensity Level and Object Pattern.

## 10.1 Intensity Level



**Figure 10-1: Intensity Level Module Diagram**

### 10.1.1 Prologue

Intensity Level is responsible for converting the distance to intensity. The module will compare the distance that the Distance Analyzer module passed to a list of intensity thresholds that will be predefined. This module accepts the distance and id of the sensor, and passes the distance as an intensity level and sensor id to the Object Pattern module.

### 10.1.2 Interfaces

|  |  |  |
| --- | --- | --- |
| **Producer** | **Consumer** | **Description** |
| Distance Analyzer | Intensity Level | The distance and id of the sensor that detected the object. |
| Intensity Level | Object Pattern | The distance converted to an intensity and the sensor id to specify which vibration motor to activate. |

**Table 10-1: Intensity Level 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** |
| sensor\_id | int | The id of the distance sensor that detected the object |
| distance | int | The distance in inches to reach the nearby detected object from the distance sensor that detected it. This variable is a data member of the object\_detected\_message. |
| vibration\_intensity | int | The intensity required based on the distance |

**Table 10-2: Intensity Level Module Internal Data Descriptors**

### 10.1.5 Service Dependencies

This module does not have any service dependencies.

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

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

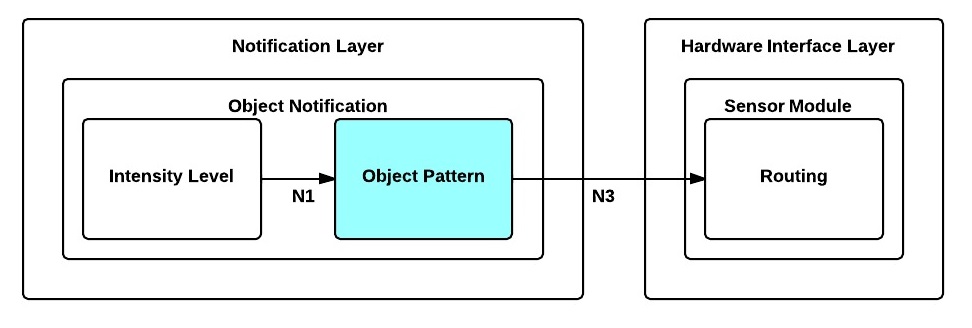
**{**

//convert the distance into an intensity value

//Send vibration\_intensity and sensor\_id to create Pattern

**}**

## 10.2 Object Pattern



**Figure 10-2: Object Pattern Module Diagram**

### 10.2.1 Prologue

Object Pattern is responsible for generating the vibration pattern that the system will use to notify the user that an object is closed to the user. The module will use the intensity to calculate the activation duration that will create the desired vibration pattern which the vibration motor will need to follow. This module accepts the calculated intensity and id of the sensor from the Intensity Level module as input and passes the vibration pattern that needs to be executed by a specific vibration motor and the id of that vibration motor ( based on the sensor id) to the Routing module.

### 10.2.2 Interfaces

|  |  |  |
| --- | --- | --- |
| **Producer** | **Consumer** | **Description** |
| Intensity Level | Object Pattern | The distance converted to an intensity and the sensor id to specify which vibration motor to activate. |
| Object Pattern | Routing | The pattern and id of vibration motor. |

**Table 10-3: Object Pattern Module Interfaces**

### 10.2.3 External Data Dependencies

This module does not have any external data dependencies.

### 10.2.4 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 |
| vibration\_intensity | int | The intensity required based on the distance |
| vibration\_pattern | string | The pattern will be a list of numbers representing the delay in microseconds between turning the vibration on and off. The delimiter used will be a semicolon. |

**Table 10-4: Object Pattern Module Internal Data Descriptors**

### 10.2.5 Service Dependencies

This module does not have any service dependencies.

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

**void** ObjectPattern**(int** vibration\_intensity**,** **int** sensor\_id**)**

**{**

//map intensity to a vibration pattern

//select the vibration pattern

//map the sensor id to the vibration id vibration motor ID

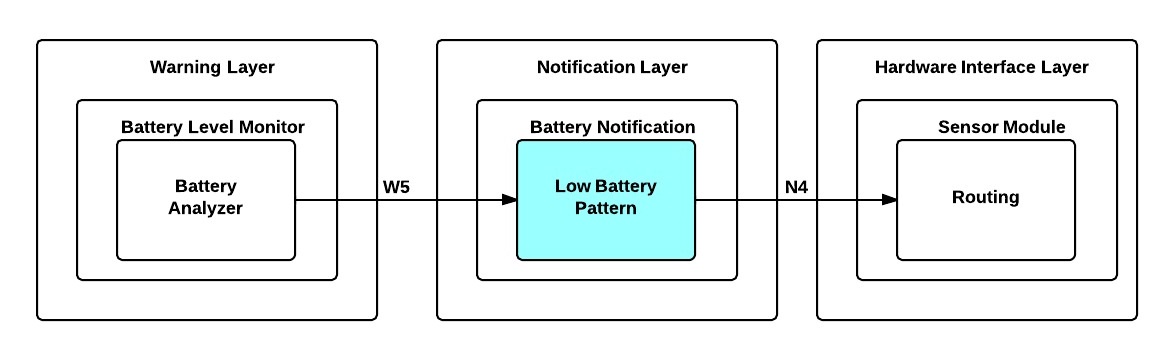
//Send the vibration pattern and vibration\_id to the Routing module

**}**

# ****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 patter 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 that the system will use to notify the user that the battery is low. The vibration pattern will have a predefined activation duration, which each vibration motor will need to follow. This module is called by the Battery Analyzer module and passes the vibration 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 pattern and id of vibration motor |

**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 | String | The vibration pattern that represents the battery is low. The pattern will contain integers (representing the delay in microseconds) separated by a semicolon. |
| 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. |

**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** lowVibrationPattern**()**

**{**

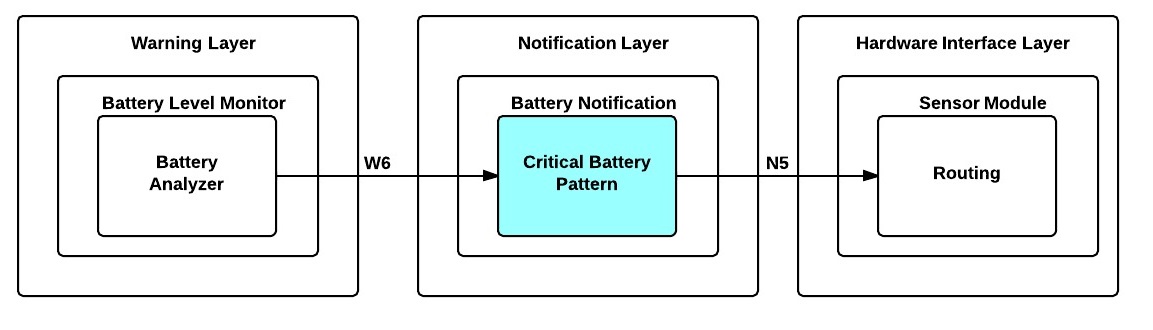
/\*initialize vibration pattern\*/

**int** vibration\_id **=** **-**1**;**

//Send vibration\_id and vibration\_pattern to the Routing Module

**}**

## 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 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 will have a predefined activation duration, which each vibration motor will need to follow. This module is called by the Battery Analyzer module and passes the vibration 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 low battery notification. |
| Critical Battery Pattern | Routing | The pattern and id of vibration motor |

**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 that the battery is about to be completely discharge. The pattern will contain integers (representing the delay in microseconds) separated by a semicolon. |
| 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. |

**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** criticalVibrationPattern**()**

**{**

//initialize vibration pattern

**int** vibration\_id **=** **-**1

//Send vibration\_id and vibration\_pattern to the Routing module

**}**

# 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 System Control 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 the maximum detection range can be changed correctly by printing out the result of the new maximum detection range updated by the Range Setting module.

**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 and sensor 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 Adjustor**

The Range Adjustor module will be sent a test command containing a maximum detection range. 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 maximum 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. **Conversion**

The Conversion module will be sent a test voltage reading. It will be verified by printing out the results of the Conversion module and compare them with the calculated percentage.

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 produce a unique vibration pattern. It will be verified by checking the output produced by this module matches the correct pattern.

.

**12.2.3.2 Object Notification subsystem**

1. **Intensity Level**

The Intensity Level module will receive a test distance and it must be able to convert that distance into the correct intensity. We will verify this by printing out the intensity level that is outputted.

1. **Object Pattern**

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

## 12.3 Component 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 Mega2560 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 Mega2560

The Arduino Mega2560 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.
3. Verify that the system is ordering the messages received correctly.

### 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 | All vibration motors should activate with a unique vibration pattern. |
| 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** | **Conversion** | **Battery Analyzer** | **On Pattern** | **Intensity Level** | **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 | x |  |  |
| **3.4** | Battery Powered |  | x |  |  |  |  |  |  | x | x | x |  |  |  | x | x |
| **3.6** | Low Battery Notification |  |  |  |  | x |  |  |  | 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 | x | x |
| **5.3** | Detection Quality |  |  |  | x |  |  | x | x |  |  |  |  |  |  |  |  |
| **5.4** | Vibration Intensity |  |  |  |  |  |  |  |  |  |  |  | x | x | x | x | x |
| **5.5** | Battery Life |  | x | x | 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 | 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** | **Read Voltage** | **Conversion** | **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 | x |
| **3.6** | Low Battery Notification |  |  |  | x | x | x |
| **3.7** | User Friendly |  |  |  |  |  |  |
| **5.2** | Real-Time Response |  | x | x | x | x | x |
| **5.3** | Detection Quality |  | x | x |  |  |  |
| **5.4** | Vibration Intensity |  |  |  |  |  |  |
| **5.5** | Battery Life |  | x | x | x | x | x |
| **6.4** | Heat Dissipation |  | x | 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** | **Intensity Level** | **Object Pattern** | **Low Battery Pattern** |
| **No.** | **Requirement** |  |  |  |  | **Object** |  |
| **3.1** | On and Off |  | x |  |  |  |  |
| **3.2** | Detect Obstructions |  |  |  |  |  |  |
| **3.3** | Obstruction Notification |  |  |  | x | 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 | x |
| **5.3** | Detection Quality |  |  |  |  |  |  |
| **5.4** | Vibration Intensity |  | x | x | x | x | x |
| **5.5** | Battery Life |  | x | x | x | x | x |
| **6.4** | Heat Dissipation |  | x | 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, Intensity Level, 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 one cable. 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 A: Arduino Libraries

## Standard Libraries

* [**SPI**](http://arduino.cc/en/Reference/SPI) - for communicating with devices using the Serial Peripheral Interface (SPI) Bus
* [**SoftwareSerial**](http://arduino.cc/en/Reference/SoftwareSerial) - for serial communication on any digital pins. Version 1.0 and later of Arduino incorporate [Mikal Hart](http://arduiniana.org/)'sNewSoftSerial library as SoftwareSerial.
* [**Stepper**](http://arduino.cc/en/Reference/Stepper) - for controlling stepper motors
* [**TFT**](http://arduino.cc/en/Reference/TFTLibrary) - for drawing text , images, and shapes on the Arduino TFT screen
* [**WiFi**](http://arduino.cc/en/Reference/WiFi) - for connecting to the internet using the Arduino WiFi shield
* [**Wire**](http://arduino.cc/en/Reference/Wire)- Two Wire Interface (TWI/I2C) for sending and receiving data over a net of devices or sensors.

## Contributed Libraries

Sensing:

* [**Debounce**](http://www.arduino.cc/playground/Code/Debounce) - for reading noisy digital inputs (e.g. from buttons)

Timing:

* [**DateTime**](http://www.arduino.cc/playground/Code/DateTime) - a library for keeping track of the current date and time in software.
* [**Metro**](http://www.arduino.cc/playground/Code/Metro) - help you time actions at regular intervals
* [**MsTimer2**](http://www.arduino.cc/playground/Main/MsTimer2) - uses the timer 2 interrupt to trigger an action every N milliseconds.

## 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>