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: 2/11/2014

# Table of Contents

[Table of Contents 2](#_Toc379953429)

[Document Revision History 10](#_Toc379953430)

[List of Figures 11](#_Toc379953431)

[List of Tables 12](#_Toc379953432)

[1. Introduction 13](#_Toc379953433)

[1.2 Purpose and Use 13](#_Toc379953434)

[1.3 Product Concept 14](#_Toc379953435)

[1.4 Product Scope 14](#_Toc379953436)

[2. Architecture Overview 15](#_Toc379953437)

[2.1 Hardware Interface Layer 16](#_Toc379953438)

[2.1.1 System Control Module subsystem 16](#_Toc379953439)

[2.1.2 Sensor Module(s) subsystem 16](#_Toc379953440)

[2.2 Warning Layer 16](#_Toc379953441)

[2.2.1 Battery Level Monitor subsystem 16](#_Toc379953442)

[2.2.2 Detection Filter subsystem 16](#_Toc379953443)

[2.2.3 Warning Manager subsystem 16](#_Toc379953444)

[2.3 Notification Layer 17](#_Toc379953445)

[2.3.1 On/Off Notification subsystem 17](#_Toc379953446)

[2.3.2 Object Notification subsystem 17](#_Toc379953447)

[2.3.3 Battery Notification subsystem 17](#_Toc379953448)

[2.4 Module Decomposition 17](#_Toc379953449)

[2.4.1 Power 19](#_Toc379953450)

[2.4.2 Range Setting 19](#_Toc379953451)

[2.4.3 Sensor Module: Detection 19](#_Toc379953452)

[2.4.4 Interpreter 19](#_Toc379953453)

[2.4.5 Routing 19](#_Toc379953454)

[2.4.6 Range Adjuster 19](#_Toc379953455)

[2.4.7 Distance Analyzer 19](#_Toc379953456)

[2.4.8 Voltage Reader 19](#_Toc379953457)

[2.4.9 Conversion 19](#_Toc379953458)

[2.4.10 Battery Analyzer 19](#_Toc379953459)

[2.4.11 Message Control 20](#_Toc379953460)

[2.4.12 On Pattern 20](#_Toc379953461)

[2.4.13 Off Pattern 20](#_Toc379953462)

[2.4.14 Intensity Level 20](#_Toc379953463)

[2.4.15 Object Pattern 20](#_Toc379953464)

[2.4.16 Low Battery Pattern 20](#_Toc379953465)

[3. Hardware Interface Layer: System Control Module 21](#_Toc379953466)

[3.1 Power 21](#_Toc379953467)

[3.1.1 Prologue 21](#_Toc379953468)

[3.1.2 Interfaces 21](#_Toc379953469)

[3.1.3 External Data Dependencies 22](#_Toc379953470)

[3.1.4 Internal Data Descriptors 22](#_Toc379953471)

[3.1.5 Module Processing (Pseudo-code Algorithm) 22](#_Toc379953472)

[3.2 Range Setting 22](#_Toc379953473)

[3.2.1 Prologue 22](#_Toc379953474)

[3.2.2 Interfaces 23](#_Toc379953475)

[3.2.3 External Data Dependencies 23](#_Toc379953476)

[3.2.4 Internal Data Descriptors 23](#_Toc379953477)

[3.2.5 Module Processing (Pseudo-code Algorithm) 23](#_Toc379953478)

[4. Hardware Interface Layer: Sensor Module(s) 24](#_Toc379953479)

[4.1 Detection 24](#_Toc379953480)

[4.1.1 Prologue 24](#_Toc379953481)

[4.1.2 Interfaces 25](#_Toc379953482)

[4.1.3 External Data Dependencies 25](#_Toc379953483)

[4.1.4 Internal Data Descriptors 25](#_Toc379953484)

[4.1.5 Module Processing (Pseudo-code Algorithm) 26](#_Toc379953485)

[4.2 Routing 26](#_Toc379953486)

[4.2.1 Prologue 26](#_Toc379953487)

[4.2.2 Interfaces 27](#_Toc379953488)

[4.2.3 External Data Dependencies 27](#_Toc379953489)

[4.2.4 Internal Data Descriptors 27](#_Toc379953490)

[4.2.5 Module Processing (Pseudo-code Algorithm) 27](#_Toc379953491)

[4.3 Interpreter 27](#_Toc379953492)

[4.3.1 Prologue 27](#_Toc379953493)

[4.3.2 Interfaces 28](#_Toc379953494)

[4.3.3 External Data Dependencies 28](#_Toc379953495)

[4.3.4 Internal Data Descriptors 28](#_Toc379953496)

[4.3.5 Module Processing (Pseudo-code Algorithm) 29](#_Toc379953497)

[5. Warning Layer: Detection Filter 30](#_Toc379953498)

[5.1 Range Adjuster 30](#_Toc379953499)

[5.1.1 Prologue 30](#_Toc379953500)

[5.1.2 Interfaces 30](#_Toc379953501)

[5.1.3 External Data Dependencies 30](#_Toc379953502)

[5.1.4 Internal Data Descriptors 31](#_Toc379953503)

[5.1.5 Module Processing (Pseudo-code Algorithm) 31](#_Toc379953504)

[5.2 Distance Analyzer 31](#_Toc379953505)

[5.2.1 Prologue 31](#_Toc379953506)

[5.2.2 Interfaces 32](#_Toc379953507)

[5.2.3 External Data Dependencies 32](#_Toc379953508)

[5.2.4 Internal Data Descriptors 32](#_Toc379953509)

[5.2.5 Module Processing (Pseudo-code Algorithm) 33](#_Toc379953510)

[6. Warning Layer: Battery Level Monitor 34](#_Toc379953511)

[6.1 Voltage Reader 34](#_Toc379953512)

[6.1.1 Prologue 34](#_Toc379953513)

[6.1.2 Interfaces 34](#_Toc379953514)

[6.1.3 External Data Dependencies 34](#_Toc379953515)

[6.1.4 Internal Data Descriptors 34](#_Toc379953516)

[6.1.5 Module Processing (Pseudo-code Algorithm) 35](#_Toc379953517)

[6.2 Conversion 35](#_Toc379953518)

[6.2.1 Prologue 35](#_Toc379953519)

[6.2.2 Interfaces 35](#_Toc379953520)

[6.2.3 External Data Dependencies 35](#_Toc379953521)

[6.2.4 Internal Data Descriptors 35](#_Toc379953522)

[6.2.5 Module Processing (Pseudo-code Algorithm) 35](#_Toc379953523)

[6.3 Battery Analyzer 36](#_Toc379953524)

[6.3.1 Prologue 36](#_Toc379953525)

[6.3.2 Interfaces 36](#_Toc379953526)

[6.3.3 External Data Dependencies 36](#_Toc379953527)

[6.3.4 Internal Data Descriptors 36](#_Toc379953528)

[6.3.5 Module Processing (Pseudo-code Algorithm) 37](#_Toc379953529)

[7. Warning Layer: Warning Manager 38](#_Toc379953530)

[7.1 Message Control 38](#_Toc379953531)

[7.1.1 Prologue 38](#_Toc379953532)

[7.1.2 Interfaces 39](#_Toc379953533)

[7.1.3 External Data Dependencies 40](#_Toc379953534)

[7.1.4 Internal Data Descriptors 40](#_Toc379953535)

[7.1.5 Module Processing (Pseudo-code Algorithm) 41](#_Toc379953536)

[8. Notification Layer: On/Off Notification 42](#_Toc379953537)

[8.1 On Pattern 42](#_Toc379953538)

[8.1.1 Prologue 42](#_Toc379953539)

[8.1.2 Interfaces 42](#_Toc379953540)

[8.1.3 External Data Dependencies 42](#_Toc379953541)

[8.1.4 Internal Data Descriptors 43](#_Toc379953542)

[8.1.5 Module Processing (Pseudo-code Algorithm) 43](#_Toc379953543)

[8.2 Off Pattern 43](#_Toc379953544)

[8.2.1 Prologue 43](#_Toc379953545)

[8.2.2 Interfaces 43](#_Toc379953546)

[8.2.3 External Data Dependencies 44](#_Toc379953547)

[8.2.4 Internal Data Descriptors 44](#_Toc379953548)

[8.2.5 Module Processing (Pseudo-code Algorithm) 44](#_Toc379953549)

[9. Notification Layer: Object Notification 45](#_Toc379953550)

[9.1 Intensity Level 45](#_Toc379953551)

[9.1.1 Prologue 45](#_Toc379953552)

[9.1.2 Interfaces 45](#_Toc379953553)

[9.1.3 External Data Dependencies 45](#_Toc379953554)

[9.1.4 Internal Data Descriptors 46](#_Toc379953555)

[9.1.5 Module Processing (Pseudo-code Algorithm) 46](#_Toc379953556)

[9.2 Object Pattern 46](#_Toc379953557)

[9.2.1 Prologue 46](#_Toc379953558)

[9.2.2 Interfaces 46](#_Toc379953559)

[9.2.3 External Data Dependencies 47](#_Toc379953560)

[9.2.4 Internal Data Descriptors 47](#_Toc379953561)

[9.2.5 Module Processing (Pseudo-code Algorithm) 47](#_Toc379953562)

[10. Notification Layer: Battery Notification 48](#_Toc379953563)

[10.1 Low Battery Pattern 48](#_Toc379953564)

[10.1.1 Prologue 48](#_Toc379953565)

[10.1.2 Interfaces 48](#_Toc379953566)

[10.1.3 External Data Dependencies 48](#_Toc379953567)

[10.1.4 Internal Data Descriptors 49](#_Toc379953568)

[10.1.5 Module Processing (Pseudo-code Algorithm) 49](#_Toc379953569)

[11. Quality Assurance 50](#_Toc379953570)

[11.1 Test Plan and Procedures 50](#_Toc379953571)

[11.2 Module/Unit Testing 50](#_Toc379953572)

[11.2.1 Hardware Interface Layer 50](#_Toc379953573)

[11.2.2 Warning Layer 51](#_Toc379953574)

[11.2.3 Notification Layer 52](#_Toc379953575)

[11.3 Component Testing 52](#_Toc379953576)

[11.3.1 Distance Sensor(s) 52](#_Toc379953577)

[11.3.2 Vibration Motor(s) 53](#_Toc379953578)

[11.3.3 System Control Module 53](#_Toc379953579)

[11.3.4 Arduino Mega2560 53](#_Toc379953580)

[11.4 Integration Testing 53](#_Toc379953581)

[11.4.1 Hardware Interface Layer 53](#_Toc379953582)

[11.4.2 Warning Layer 53](#_Toc379953583)

[11.4.3 Notification Layer 54](#_Toc379953584)

[11.5 System Verification Testing 54](#_Toc379953585)

[11.6 Test Cases 54](#_Toc379953586)

[12. Requirements Traceability 55](#_Toc379953587)

[12.1 Purpose 55](#_Toc379953588)

[12.2 Requirements Traceability Matrix By Module 55](#_Toc379953589)

[12.3 Hardware Interface Layer Module Mapping 56](#_Toc379953590)

[12.4 Warning Layer Module Mapping 57](#_Toc379953591)

[12.5 Notification Layer Module Mapping 58](#_Toc379953592)

[12.6 Analysis of Requirements Mapping 59](#_Toc379953593)

[13.1 Packaging and Installation 60](#_Toc379953594)

[13.2 Acceptance Testing 60](#_Toc379953595)

[13.3 Acceptance Criteria 60](#_Toc379953596)

[13.3.1 The System Shall Be Intuitive and Accessible. 60](#_Toc379953597)

[13.3.2 The System Shall Help Visually Impaired Individuals Traverse in both Indoor and Outdoor Environments 61](#_Toc379953598)

[13.3.3 The System Shall Notify The User That The Battery Is Low 61](#_Toc379953599)

[13.3.4 The System Shall Be Safe To Wear 62](#_Toc379953600)

[Appendix A: Arduino Libraries 63](#_Toc379953601)

[Standard Libraries 63](#_Toc379953602)

[Due Only Libraries 63](#_Toc379953603)

[Esplora Only Library 63](#_Toc379953604)

[Arduino Robot Library 63](#_Toc379953605)

[Arduino Yún Bridge Library 63](#_Toc379953606)

[USB Libraries (Leonardo, Micro, Due, and Esplora) 63](#_Toc379953607)

[Contributed Libraries 64](#_Toc379953608)

[**Communication (networking and protocols)** 64](#_Toc379953609)

[**Sensing:** 64](#_Toc379953610)

[**Displays and LEDs:** 64](#_Toc379953611)

[**Audio and Waveforms:** 64](#_Toc379953612)

[**Motors and PWM:** 65](#_Toc379953613)

[**Timing:** 65](#_Toc379953614)

[**Utilities:** 65](#_Toc379953615)

# Document Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Revision Number | Revision Date | Description | Rationale |
| 1.0 | 2/11/2014 | Initial DDS Draft ready for review | Initial and Review Draft |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# List of Figures

Figure # Title Page #

2-1 Architecture Design Diagram 15

2-2 Module Decomposition Chart 18

# 

# 

# List of Tables

Table # Title Page #

3-1 Power Module Interfaces 21

3-2 Range Setting Module Interfaces 23

4-1 Detection Module Interfaces 25

4-2 Routing Module Interfaces 27

4-3 Interpreter Module Interfaces 28

5-1 Range Adjuster Module Interfaces 30

5-2 Distance Analyzer Module Interfaces 32

6-1 Voltage Reader Module Interfaces 34

6-2 Conversion Module Interfaces 35

6-3 Battery Analyzer Module Interfaces 36

7-1 Message Priorities 38

7-2 Message Control Module Interfaces 39

8-1 On Patten Module Interfaces 42

8-2 Off Pattern Module Interfaces 43

9-1 Intensity Level Module Interfaces 45

9-2 Object Pattern Module Interfaces 46

10-1 Low Battery Pattern Module Interfaces 48

11-1 Test Cases 54

12-1 Requirements Traceability Matrix by Module 55

12-2 Hardware Interface Layer Module Mapping 56

12-3 Warning Layer Module Mapping 57

12-4 Notification Layer Module Mapping 58

# 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 assumptions, 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 batteries, which connects to the sensors and vibration motors. When the user is not using the product and wishes to charge the batteries, the batteries will be removed from the system and put into a docking station to charge.

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 3 feet 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. There can be multiple vibration motors vibrating at the same time and at different intensities depending on where the object is located. 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 knob 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.

## Product Scope

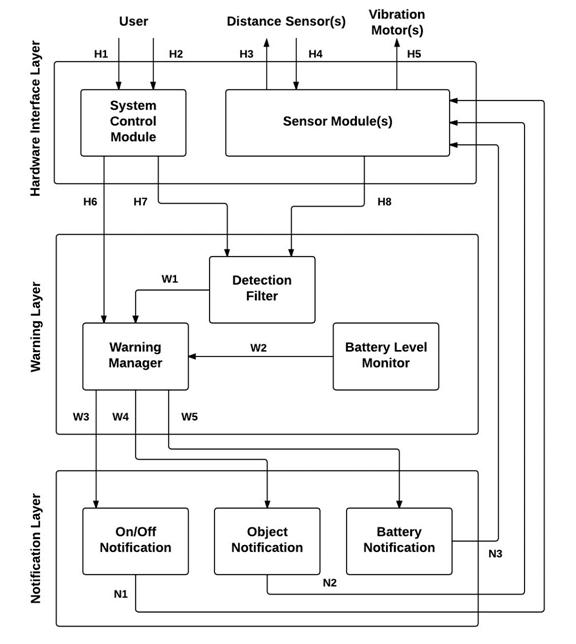
The Sight By Touch system is made to aid visually impaired users through the use of vibrations. Users of the Sight By Touch system will be able to move around and gauge where they are. When a user walks near an object, the system will vibrate in the direction where the object is closest. This warns the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



**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 reformatting it so the Warning Layer can process it. 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. 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 relaying the battery level to the Warning Manager. This will help in notifying the user when the battery level is low.

### 2.2.2 Detection Filter subsystem

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

### 2.2.3 Warning Manager subsystem

The Warning Manager will determine the type of messages that needs to be relayed to the Notification Layer.

## 2.3 Notification Layer

This layer is responsible for sending all the signals from the Warning Layer to the correct sensing modules in order to notify the user of each event. The events include the On/Off Notification, the Objected Detected Notification, and the Battery Notification. The following are the subsystems of the Notification Layer.

### 2.3.1 On/Off Notification subsystem

The On/Off Notification subsystem will receive messages from the Warning Manager and interpret them into appropriate instructions to relay to the Sensor Module(s).

### 2.3.2 Object Notification subsystem

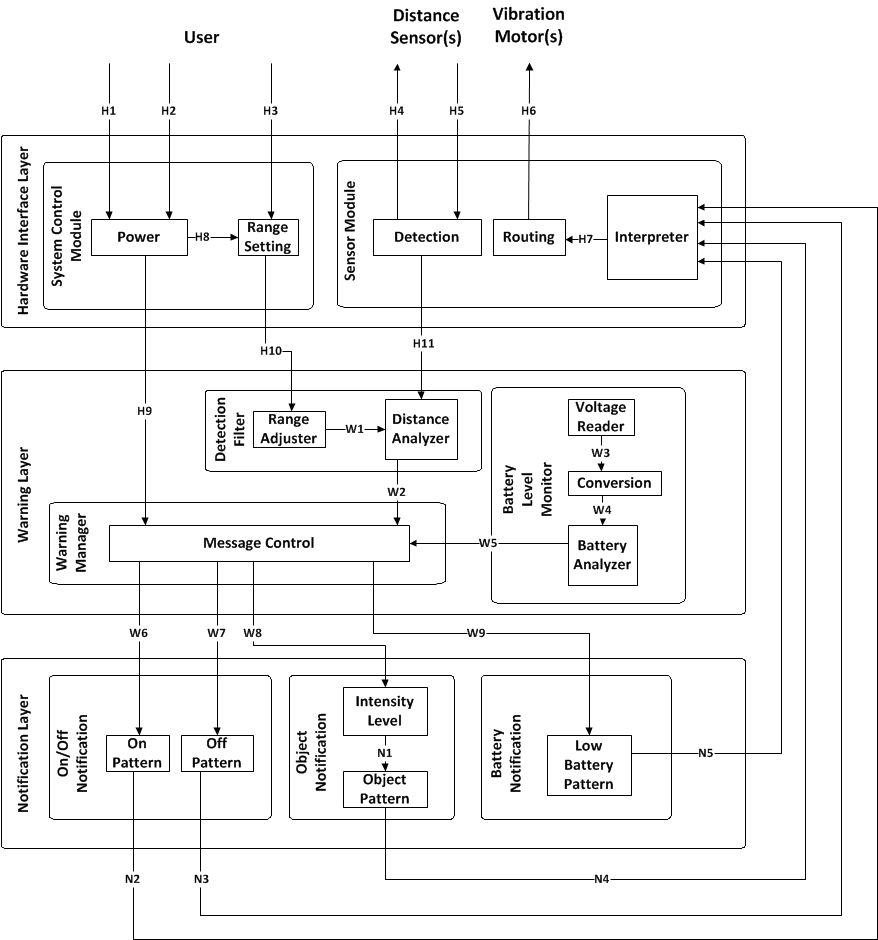
The Object Notification subsystem will receive messages from the Warning Manager whenever the system has detected a significant object and interpret them into appropriate instructions to relay to the Sensor Module(s).

### 2.3.3 Battery Notification subsystem

The Battery Notification subsystem will receive a message from the Warning Manager whenever the system’s battery is running low and interpret them into appropriate instructions to relay to the Sensor Module(s).

## 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 dataflow 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 dataflows needed to fully describe the detailed design.

### 2.4.1 Power

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

### 2.4.2 Range Setting

Range Setting is responsible for handling the use case in which the range adjuster knob’s position changes.

### 2.4.3 Sensor Module: Detection

Detection focuses on detecting objects, acquiring distance to them and passing the distance data to the Warning Layer.

### 2.4.4 Interpreter

Interpreter focuses on converting notification data from Notification Layer into pulses that can be used to operate vibration motors.

### 2.4.5 Routing

Routing focuses on sending a signal to the vibration motors to operate using the pulses created by translates.

### 2.4.6 Range Adjuster

Range Adjuster focuses on updating the range to make filter decisions on using the new range setting from the System Control Module.

### 2.4.7 Distance Analyzer

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

### 2.4.8 Voltage Reader

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

### 2.4.9 Conversion

Conversion focuses on converting the voltage read into a format that can be analyzed.

### 2.4.10 Battery Analyzer

Battery Analyzer focuses on making decisions on the battery level reading created from Conversion.

### 2.4.11 Message Control

Message Control is responsible for managing the incoming alert messages from the Power, Distance Analyzer, and Battery Analyzer modules and outputting messages to the Notification Layer based on priority.

### 2.4.12 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 passes that data to the Sensor Modules to give the actual feedback.

### 2.4.13 Off Pattern

Off Pattern is responsible for creating the vibration pattern that will be used to notify the user that the system is off and passes that data to the Sensor Modules to give the actual feedback.

### 2.4.14 Intensity Level

Intensity Level focuses on analyzing the distance data to determine what kind of pattern is needed.

### 2.4.15 Object Pattern

Object Pattern focuses on the actual creation of the pattern using the results of determine intensity and sends that data to the Sensor Modules to give feedback.

### 2.4.16 Low Battery Pattern

Low Battery Pattern will be responsible for creating the vibration pattern that will represent low power left and sends the data to the Sensor Modules to give feedback.

# ****3. 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.

## 3.1 Power

### 3.1.1 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 doesn’t have any inputs. It outputs data which will be passed to the Message Control module in the Warning Manager subsystem to signify that the system is to going to power on or off and a signal to update the maximum detection range to the Range Setting module in the System Control Module subsystem.

### 3.1.2 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 | Message Control | A translated command that represents “power off” or “power on” to the Message Control module. |
| Power | Range Setting | A signal to update the maximum detection range when the system is turned on. |

**Table 3-1: Power Module Interfaces**

### 3.1.3 External Data Dependencies

This module doesn’t have any external data dependencies.

### 3.1.4 Internal Data Descriptors

This module will utilize integer data types to represent the status of the system and the commands to send to the Message Control module regarding whether the system is turning on or off.

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

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

**{**

/\*system status flag value will change, change value from on to off or

vice versa\*/

**if(**system\_status **==** SYSTEM\_ON**)**

**{**

//generate command representing "power on"

//send command to Warning Manager Message Control module

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

maximum detection range of the system\*/

**}**

**else** **if(**system\_status **==** SYSTEM\_OFF**)**

**{**

//generate command representing "power off"

//send command to Warning Manager Message Control module

**}**

**}**

## 3.2 Range Setting

### 3.2.1 Prologue

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

### 3.2.2 Interfaces

|  |  |  |
| --- | --- | --- |
| **Producer** | **Consumer** | **Description** |
| User | Range Setting | A signal will be generated by the user twisting the knob switch on the control belt to a certain position and sent to Range Setting for translation. |
| Power | Range Setting | A signal to update the maximum detection range when the system is turned on. |
| Range Setting | Range Adjuster | A translated command that represents the new maximum detection range to the Range Adjuster for processing. |

**Table 3-2: Range Setting Module Interfaces**

### 3.2.3 External Data Dependencies

This module doesn’t have any external data dependencies.

### 3.2.4 Internal Data Descriptors

This module will utilize an integer data type that represents the new maximum detection range.

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

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

**{**

//initialize variable to hold signal data

//call function to read signal from knob switch

**if(**signal **==** INDOOR\_RANGE**)**

**{**

//generate command representing the new maximum detection range

//send translated command to Detection Filter Range Adjuster module

**}**

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

**{**

//generate command representing the new maximum detection range

//send translated command to Detection Filter Range Adjuster module

**}**

**}**

# ****4. 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, Routing, and Interpreter.

## 4.1 Detection

### 4.1.1 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 output data which will be passed to the Distance Analyzer module in the Detection Filter subsystem for processing. This process will repeat for all distance sensors in round robin fashion.

### 4.1.2 Interfaces

|  |  |  |
| --- | --- | --- |
| **Producer** | **Consumer** | **Description** |
| Detection | Distance Sensor(s) | A signal to the Distance Sensor(s). These will generate ultrasonic sound waves and send them into the environment to detect nearby objects. |
| Distance Sensor(s) | Detection | Distance Sensor(s) will send responses that were detected to the Detection for translation. |
| Detection | Distance Analyzer | Detection will output a message containing the translated data from the responses sent by the Distance Sensor(s) and the sensor id associated to the Distance Analyzer for processing. |

**Table 4-1: Detection Module Interfaces**

### 4.1.3 External Data Dependencies

This module is dependent on the responses received from the distance sensors, a long representing the time in milliseconds to reach the nearby object.

### 4.1.4 Internal Data Descriptors

This module will utilize numerical data types to represent distance (ex. integer, float, etc.), an integer to identify distance sensors, and a struct data type to hold both of the previous data.

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

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

**{**

//initialize variable to hold distance data from response

//initialize variable to hold response data

int sensor\_id **=** 1**;**

**while(**system\_status **==** SYSTEM\_ON**)**

**{**

/\*call library function to operate a sensor using the sensor id to

specify which sensor and store data return into the variable

that will hold the response data\*/

**if(** is\_response\_valid**(**response**)** **)**

**{**

//extract distance data from response

//initialize struct that will hold distance data and sensor id

//send the struct holding to Detection Filter Analyze module

**}**

**else** //response not valid

**{**

//do nothing

**}**

//update sensor\_id variable to a new distance sensor id for round robin

**}**

**}**

## 4.2 Routing

### 4.2.1 Prologue

Routing is responsible for forwarding the signals sent by the Interpreter module to specified vibration motor(s) to operate. This module will accept a message containing the signal and sensor id (which also specifies which vibration motor(s)) as input and output the signal to the proper vibration motor(s) based on the sensor id.

### 4.2.2 Interfaces

|  |  |  |
| --- | --- | --- |
| **Producer** | **Consumer** | **Description** |
| Interpreter | Routing | Interpreter will send a message containing the signal and sensor id to Routing so it can forward the signal to the designated vibration motor(s). |
| Routing | Vibration Motor(s) | Routing will send a signal to the specified vibration motor (based on sensor id) to work. |

**Table 4-2: Routing Module Interfaces**

### 4.2.3 External Data Dependencies

This module doesn’t have any external data dependencies.

### 4.2.4 Internal Data Descriptors

This module will utilize an integer to identify the vibration motor to operate, a string of numbers that will represent the vibration pattern, and a struct data type to hold the two previous data.

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

void SensorModuleRouting**(**struct output\_message**)**

**{**

/\*use switch statement to check sensor\_id in output\_message against

available ids\*/

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

//default, do nothing

**}**

## 4.3 Interpreter

### 4.3.1 Prologue

Interpreter is responsible for translating commands from the Notification Layer into actual signals the vibration motors need to function. This module will accept a message containing the system command and the sensor id (specifying which vibration motor) associated from the Notification Layer as input and outputs a message containing the translated signal and the sensor id to the Routing module.

### 4.3.2 Interfaces

|  |  |  |
| --- | --- | --- |
| **Producer** | **Consumer** | **Description** |
| On Pattern | Interpreter | On Pattern will send a message containing the system command and sensor id to Interpreter for translation. |
| Off Pattern | Interpreter | Off Pattern will send a message containing the system command and sensor id to Interpreter for translation. |
| Object Pattern | Interpreter | Object Pattern will send a message containing the system command and sensor id to Interpreter for translation. |
| Low Battery Pattern | Interpreter | Low Battery Pattern will send a message containing the system command and sensor id to Interpreter for translation. |
| Interpreter | Routing | A message containing the vibration pattern and vibration id (s) for the vibration motors that need to be activated. |

**Table 4-3: Interpreter Module Interfaces**

### 4.3.3 External Data Dependencies

This module doesn’t have any external data dependencies.

### 4.3.4 Internal Data Descriptors

This module will utilize an integer to represent a sensor id, an integer representing the priority of the message, a string which will represent the pattern, and a struct data type to hold the previous all the previous data mentioned here.

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

void SensorModuleInterpreter**(**struct notification\_message**)**

**{**

/\*translate command in notification\_message into signal using library

function\*/

//initialize struct variable to hold signal and sensor\_id

//send new output message to Sensor Module(s) Routing module

**}**

# ****5. 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.

## 5.1 Range Adjuster

### 5.1.1 Prologue

Range Adjuster is responsible for updating the maximum detection range the Distance Analyzer uses to make decisions. This module will accept the new maximum detection range from Range Setting as input and outputs the new maximum detection range to Distance Analyzer to use for decision making.

### 5.1.2 Interfaces

|  |  |  |
| --- | --- | --- |
| **Producer** | **Consumer** | **Description** |
| Range Setting | Range Adjuster | Range Setting will send the new maximum detection range to Range Adjuster for updating the current maximum detection range. |
| Range Adjuster | Distance Analyzer | The new maximum detection range that the Distance Analyzer will use in the decision making. |

**Table 5-1: Range Adjuster Module Interfaces**

### 5.1.3 External Data Dependencies

This module doesn’t have any external data dependencies.

### 5.1.4 Internal Data Descriptors

This module will utilize an integer to represent the new maximum detection range to use.

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

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

**{**

//extract maximum detection range value from input

//send value to the Detection Filter Analyze module

**}**

## 5.2 Distance Analyzer

### 5.2.1 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 a message containing the distance between the object detected and the distance sensor that detected that object and the sensor id of the distance sensor that detected the object and the new maximum detection range (if sent) as inputs and outputs a message containing distance and sensor id to the Message Control module if the detected range is within the current maximum detection range. Otherwise, it will ignore it.

### 5.2.2 Interfaces

|  |  |  |
| --- | --- | --- |
| **Producer** | **Consumer** | **Description** |
| Detection | Distance Analyzer | Detection will output a message containing the translated data from the responses sent by the Distance Sensor(s) and the sensor id associated to the Distance Analyzer for processing. |
| Range Adjuster | Distance Analyzer | Range Adjuster will forward the new maximum detection range to Distance Analyzer to use in decision making. |
| Distance Analyzer | Message Control | A message containing the distance data and the sensor id to the Message Control module of the object detected within the current maximum detection range. |

**Table 5-2: Distance Analyzer Module Interfaces**

### 5.2.3 External Data Dependencies

This module doesn’t have any external data dependencies.

### 5.2.4 Internal Data Descriptors

This module will utilize an integer to identify the distance sensor that detected the object, any numerical data type that can represent distance (ex. integer, float, etc.) and a struct data type to hold the previous data mentioned here.

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

void DetectionFilterDistanceAnalyzer**(**struct detect\_message**,**

int new\_max\_detect\_range**)**

**{**

**if(**new\_max\_detect\_range**)** //if defined/given

**{**

current\_max\_detect\_range **=** new\_max\_detect\_range**;**

**}**

**else** **if(**0 **<=** detect\_message**.**distance **<=** current\_max\_detect\_range**)**

**{**

//initialize struct to hold the distance data and sensor\_id

//send new message to Warning Manager: Obstruction Detection

**}**

**else**

**{**

//ignore detected object

**}**

**}**

# ****6. 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. To do so, it reads the voltage level of the battery at regular time intervals and converts the readings into a readable data format that can be analyzed. Once the remaining voltage reaches particular low levels, an interrupt will be generated and the voltage level will be reported to the Warning Manager subsystem for processing. 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.

## 6.1 Voltage Reader

### 6.1.1 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 doesn’t have any inputs. It outputs a direct reading of the battery voltage to Conversion for translation.

### 6.1.2 Interfaces

|  |  |  |
| --- | --- | --- |
| **Producer** | **Consumer** | **Description** |
| Voltage Reader | Conversion | Voltage Reader will send a voltage reading to Conversion for translation. |

**Table 6-1: Voltage Reader Module Interfaces**

### 6.1.3 External Data Dependencies

This module doesn’t have any external data dependencies.

### 6.1.4 Internal Data Descriptors

This module will utilize a float to represent the voltage reading.

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

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

**{**

**while(**system\_status **==** SYSTEM\_ON**)**

**{**

//call function to read voltage in the battery

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

/\*insert delay here to have this module check the battery at regular time intervals\*/

**}**

**}**

## 6.2 Conversion

### 6.2.1 Prologue

Conversion is responsible for converting voltage readings from the battery into a readable format that can be analyzed. This module accepts the voltage readings from the battery as input and outputs a readable format of the voltage reading in the battery to the Battery Analyzer for analysis.

### 6.2.2 Interfaces

|  |  |  |
| --- | --- | --- |
| **Producer** | **Consumer** | **Description** |
| Voltage Reader | Conversion | Voltage Reader will send a voltage reading to Conversion for translation. |
| Conversion | Battery Analyzer | Conversion will send a voltage reading in readable format to Battery Analyzer for analysis. |

**Table 6-2: Conversion Module Interfaces**

### 6.2.3 External Data Dependencies

This module doesn’t have any external data dependencies.

### 6.2.4 Internal Data Descriptors

This module will utilize a float to represent a voltage reading and an integer to represent a power percentage.

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

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

**{**

//reformat voltage\_reading into numerical form that can be analyzed

//send reformatted data to Battery Level Monitor Analyze module

**}**

## 6.3 Battery Analyzer

### 6.3.1 Prologue

Battery Analyzer is responsible for analyzing the translated voltage reading to check if the battery level has reached a particular level of power left. If the remaining power does reach such a level, it will generate a system interrupt and report to Message Control for processing. This module accepts the readable voltage readings from Conversion as input and outputs a message to Message Control for processing.

### 6.3.2 Interfaces

|  |  |  |
| --- | --- | --- |
| **Producer** | **Consumer** | **Description** |
| Conversion | Battery Analyzer | Conversion will send a voltage reading in readable format to Battery Analyzer for analysis. |
| Battery Analyzer | Message Control | Battery Analyzer will send a message to Message Control for processing. |

**Table 6-3: Battery Analyzer Module Interfaces**

### 6.3.3 External Data Dependencies

This module doesn’t have any external data dependencies.

### 6.3.4 Internal Data Descriptors

This module will utilize an integer to represent a power percentage, an integer data type that can be used to represent a flag so that the same alert is not generated multiple times, and a struct (containing an integer with value 2 for the priority) representing a message

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

void BatteryLevelMonitorBatteryAnalyzer**(**translated\_voltage\_reading**)**

**{**

//flags used so that alerts are only generated once

int critical\_low\_flag**=**0**;** //initialize to some value that means not set

int low\_flag**=**0**;** //initialize to some value that means not set

**if((**0 **<=** translated\_voltage\_reading **<=** CRITICAL\_LOW**)** and **(**critical\_flag**==**0**))**

**{**

//set critical\_flag to new value

/\*call function to generate system interrupt and send message to Warning Manager: Low Battery module\*/

**}**

**else** **if((**0 **<=** translated\_voltage\_rading **<=** LOW**)** and **(**low\_flag**==**0**))**

**{**

//set low\_flag to new value

/\*call function to generate system interrupt and send message to Warning Manager: Low Battery module\*/

**}**

**else** //nothing worth alerting yet

**{**

//do nothing

**}**

**}**

# ****7. Warning Layer: Warning Manager****

The Warning Manager is responsible for controlling flow of messages for the Sight By Touch system. The Subsystem will receive many messages such as the system is on or off, an obstruction was detected, or the battery is low which will need to be ordered based on priority. This section will cover the Warning Manager in detail by describing the detailed design of the module Message Control.

## 7.1 Message Control

### 7.1.1 Prologue

Message Control is responsible for deciding in which order will be delivered next to the notification layer. The order will be based on the priority of the message and if the message has the same priority then the order will be kept based on the order of the queue. Each message type will have a priority from 1 to 3 where 1 means high priority (will be executed next) and 3 means low priority (will wait until all other message priorities have been sent). The table below shows the priority of each type of message. This module accepts data from the Power, Distance Analyzer, and Battery Analyzer modules as inputs and outputs messages to the On Pattern, Off Pattern, Intensity Level, and Low Battery Pattern modules to determine vibration patterns for the vibration motors.

|  |  |
| --- | --- |
| **Message Type** | **Priority** |
| On/Off | 1 |
| Low Battery | 2 |
| Object Obstruction | 3 |

**Table 7-1: Message Priorities**

### 7.1.2 Interfaces

|  |  |  |
| --- | --- | --- |
| **Producer** | **Consumer** | **Description** |
| Power | Message Control | A translated command that represents “power off” or “power on” to the Message Control module. |
| Distance Analyzer | Message Control | A message containing the distance data and the sensor id to the Message Control module of the object detected within the current maximum detection range. |
| Battery Analyzer | Message Control | Battery Analyzer will send a message to Message Control for processing. |
| Message Control | On Pattern | The message was determined to be an On message and will be sent to the On Pattern module. |
| Message Control | Off Pattern | The message was determined to be an Off message and will be sent to the Off Pattern module. |
| Message Control | Intensity Level | The message was determined to be an Object Obstruction message and will be sent to the Intensity module to determine intensity based on distance detected. |
| Message Control | Low Battery Pattern | The message was determined to be a low battery message and will be sent to the Low Battery Pattern module. |

**Table 7-2: Message Control Module Interfaces**

### 7.1.3 External Data Dependencies

This module doesn’t have any external data dependencies.

### 7.1.4 Internal Data Descriptors

This module will use an array of structs to simulate a queue of messages where each message will consist of a struct that will hold the information of the message. 2 unsigned integers where one will signal a low battery notification, another will signaling a power on or power off.

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

void MessageOrganazier**(**int priority, struct message**)**

**{**

//This method will be called every time there is another incoming

//message

**if(Priority = ”1”)**

**{**

//This signifies an On/Off message

**if(message = ”On”)**

**{**

Power = 1

**}**

**else**

**{**

Power = 2

**}**

**}**

**else** **if(Priority = “2”)**

**{**

Battery = 1

**}**

**else**

**{**

//This signifies an object obstruction message that will be placed at

//the end of the queue

**}**

**}**

void ExecuteNotification**()**

**{**

//This method will take the next message in the queue that needs to be

//sent to the Notification Layer

**while(**system\_status **==** SYSTEM\_ON && Queue != Empty**)**

**{**

**if(Power = ”1”)**

**{**

Power = 0

//Forward message to the On Pattern Module

**}**

**else** **if(Power = “2”)**

**{**

Power = 0

//Forward message to the Off Pattern Module

**}**

**else if (Battery = “1”)**

**{**

Battery = 0

//Forward to the Low Battery Pattern Module

**}**

**else**

**{**

//Forward message to the Intensity Level Module

**}**

**}**

**}**

# ****8. Notification Layer: On/Off 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 Warning Manager whenever the system needs to generate an on or off vibration pattern and then will send that patter to the Sensor Module subsystem. This section will cover the On/Off Notification in detail by describing the detailed design of the modules that constitute the entire subsystem. These modules are the On Pattern and Off Pattern.

## 8.1 On Pattern

### 8.1.1 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 accepts a message from the Message Control module as input and outputs a message containing the vibration pattern to be executed by all vibration motors and the vibration motor id (a.k.a. sensor id) signifying all vibration motors to the Interpreter module for translation.

### 8.1.2 Interfaces

|  |  |  |
| --- | --- | --- |
| **Producer** | **Consumer** | **Description** |
| Message Control | On Pattern | The message was determined to be an On message and will be sent to the On Pattern module. |
| On Pattern | Interpreter | A message containing the vibration pattern that needs to be executed by every vibration motor and the sensor id that represents all vibration motors will be sent to the Interpreter module for translation. |

**Table 8-1: On Pattern Module Interfaces**

### 8.1.3 External Data Dependencies

This module doesn’t have any external data dependencies.

### 8.1.4 Internal Data Descriptors

This module will use a struct to encapsulate the vibration pattern. Each struct will contain an unsigned integer to signal that all vibration motors need to be activated, another unsigned integer that will state the duration of the vibration.

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

void OnVibrationPattern**(**struct message**)**

**{**

//initialize new struct that holds vibration pattern and sensor id

//Send notification message to the Sensor Module(s)

**}**

## 8.2 Off Pattern

### 8.2.1 Prologue

Off Pattern is responsible for generating the vibration pattern that the system will use to notify the user that the system has been turned off. The vibration pattern will have a predefined activation duration, which each vibration motor will need to follow. This module accepts a message from the Message Control module as input and outputs a message containing the vibration pattern to be executed by all vibration motors and the vibration motor id (a.k.a. sensor id) signifying all vibration motors to the Interpreter module for translation.

### 8.2.2 Interfaces

|  |  |  |
| --- | --- | --- |
| **Producer** | **Consumer** | **Description** |
| Message Control | Off Pattern | The message was determined to be an Off message and will be sent to the Off Pattern module. |
| Off Pattern | Interpreter | A message containing the vibration pattern that needs to be executed by every vibration motor and the sensor id that represents all vibration motors will be sent to the Interpreter module for translation. |

**Table 8-2: Off Pattern Module Interfaces**

### 8.2.3 External Data Dependencies

This module doesn’t have any external data dependencies.

### 8.2.4 Internal Data Descriptors

This module will use a struct to encapsulate the vibration pattern. Each struct will contain an unsigned integer to signal that all vibration motors need to be activated, another unsigned integer that will state the duration of the vibration.

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

void OffVibrationPattern**(**struct message**)**

**{**

//initialize new struct that holds vibration pattern and sensor id

//Send notification message to the Sensor Module(s)

**}**

# ****9. 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.

## 9.1 Intensity Level

### 9.1.1 Prologue

Intensity Level is responsible for converting the distance to intensity. The module will compare the distance that the Message Control module passed to a list of intensity thresholds that will be predefined. This module accepts a message from the Message Control module as input and outputs a message with the distance as an intensity level and vibration motor id (a.k.a. sensor id) to be interpreted by the Object Pattern module.

### 9.1.2 Interfaces

|  |  |  |
| --- | --- | --- |
| **Producer** | **Consumer** | **Description** |
| Message Control | Intensity Level | The message was determined to be an Object Obstruction message and will be sent to the Intensity module to determine intensity based on distance detected. |
| Intensity Level | Object Pattern | The message with the distance converted to an intensity and the sensor id to specify which vibration motor. |

**Table 9-1: Intensity Level Module Interfaces**

### 9.1.3 External Data Dependencies

This module doesn’t have any external data dependencies.

### 9.1.4 Internal Data Descriptors

This module will use a struct to encapsulate the message. Each struct will contain an unsigned integer to signal that only one vibration motor will be activated, another unsigned integer for the vibration ID, and another unsigned integer for the intensity.

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

void CalculateIntensity**(**struct message**)**

**{**

//read the distance in message

//convert the distance into an intensity value

//create new struct with intensity value and sensor id

//Send intensity level message to create Pattern

**}**

## 9.2 Object Pattern

### 9.2.1 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 an intensity level message from the Intensity Level module as input and outputs a message containing the vibration pattern that needs to be executed by a specific vibration motor and the id of that vibration motor (a.k.a. the sensor id) to the Interpreter module.

### 9.2.2 Interfaces

|  |  |  |
| --- | --- | --- |
| **Producer** | **Consumer** | **Description** |
| Intensity Level | Object Pattern | The message with the distance converted to an intensity and the sensor id to specify which vibration motor. |
| Object Pattern | Interpreter | A message containing the vibration pattern that needs to be executed by the specified vibration motor and that vibration motor’s id. |

**Table 9-2: Object Pattern Module Interfaces**

### 9.2.3 External Data Dependencies

This module doesn’t have any external data dependencies.

### 9.2.4 Internal Data Descriptors

This module will use a struct to encapsulate the vibration pattern. Each struct will contain an unsigned integer that will state the duration of the vibration and another unsigned integer for vibration ID

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

void PatternGenerator**(**struct intensity\_level\_essage**)**

**{**

//read the intensity in input message

//generate vibration pattern based on the intensity

//create new struct holding vibration pattern and vibration motor ID

//Send the notification message to the Sensor Module(s)

**}**

# ****10. Notification Layer: Battery Notification****

The Battery Notification subsystem is responsible for generating a low battery vibration pattern for the Sight By Touch system. The subsystem will be alerted by the Warning Manager whenever the system needs to generate a low battery vibration patter 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.

## 10.1 Low Battery Pattern

### 10.1.1 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 accepts a message from the Message Control module as input and outputs a message containing the vibration pattern to be executed by all vibration motors and the vibration motor id (a.k.a. sensor id) signifying all vibration motors to the Interpreter module for translation.

### 10.1.2 Interfaces

|  |  |  |
| --- | --- | --- |
| **Producer** | **Consumer** | **Description** |
| Message Control | Low Battery Pattern | The message was determined to be a low battery message and will be sent to the Low Battery Pattern module. |
| Low Battery Pattern | Interpreter | A message containing the vibration pattern that needs to be executed by every vibration motor and the sensor id that represents all vibration motors will be sent to the Interpreter module for translation. |

**Table 10-1: Low Battery Pattern Module Interfaces**

### 10.1.3 External Data Dependencies

This module doesn’t have any external data dependencies.

### 10.1.4 Internal Data Descriptors

This module will use a struct to encapsulate the vibration pattern. Each struct will contain an unsigned integer to signal that all vibration motors need to be activated and another unsigned integer that will state the duration of the vibration.

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

void lowVibrationPattern**(**struct message**)**

**{**

//initialize new struct that holds vibration pattern and sensor id

//Send notification message to the Sensor Module(s)

**}**

# 11. Quality Assurance

## 11.1 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.

## 11.2 Module/Unit Testing

### 11.2.1 Hardware Interface Layer

**11.2.1.1 System Control Module subsystem**

1. **Power**

The System Control Module will power on and off the system. Checking with the Warning Manager that the Message Control Module receives a “power off” or “power on” message 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 set by the Range Setting module.

**11.2.1.2 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 a test signal containing a vibration pattern and sensor id within it. It will be verified that this module can forward this signal to the proper vibration motor(s) based on the sensor id.

1. **Interpreter**

The Interpreter module will be sent a test message containing a system command and sensor id within it. It will be verified that this module can translate this command and forwards this message to the Routing module.

### 11.2.2 Warning Layer

**11.2.2.1 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.

**11.2.2.2 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.

**11.2.2.3 Warning Manager subsystem**

1. **Message Control**

The Message Control module will be sent multiple types of test messages that will include On/Off, Low battery and Object Obstruction. We will verify that it prioritizes these messages in the correct order by printing out the list after it has been ordered.

### 11.2.3 Notification Layer

**11.2.3.1 On/Off 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.

1. **Off Pattern**

The Off 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.

**11.2.3.2 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.

**11.2.3.3 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.

## 11.3 Component Testing

### 11.3.1 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) that proportion to distance. It should be able to connect to the Arduino Mega2560 without issues.

### 11.3.2 Vibration Motor(s)

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

### 11.3.3 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.

### 11.3.4 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.

## 11.4 Integration Testing

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

### 11.4.2 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.

### 11.4.3 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 is going to turn “Off”.
3. Verify that the system notifies the user when the system has detected an object within the range specified.
4. Verify that the system notifies the user when the system’s battery is low.

## 11.5 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.

## 11.6 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 11-1: Test Cases**

# 12. Requirements Traceability

## 12.1 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.

## 12.2 Requirements Traceability Matrix By Module

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | **Modules** | **Power** | **Range Setting** | **Detection** | **Routing** | **Interpreter** | **Range Adjuster** | **Distance Analyzer** | **Voltage Reader** | **Conversion** | **Battery Analyzer** | **Message Control** | **On Pattern** | **Off Pattern** | **Intensity Level** | **Object Pattern** | **Low Battery Pattern** |
| **No.** | **Requirement** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | **Object** |  |
| **3.1** | On and Off |  | x |  |  |  |  |  |  |  |  |  | x | x | x |  |  |  |
| **3.2** | Detect Obstructions |  |  |  | x | x |  | x | x |  |  |  | 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 |
| **3.7** | User Friendly |  | x | x |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **5.2** | Real-Time Response |  |  |  | x | 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 | x |
| **6.4** | Heat Dissipation |  | x | 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 12-1: Requirements Traceability Matrix by Module**

## 12.3 Hardware Interface Layer Module Mapping

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | **Modules** | **Power** | **Range Settings** | **Detection** | **Routing** | **Translation** |
| **No.** | **Requirement** |  |  |  |  |  |  |
| **3.1** | On and Off |  | x |  |  |  |  |
| **3.2** | Detect Obstructions |  |  |  | x | x |  |
| **3.3** | Obstruction Notification |  |  |  |  | x |  |
| **3.4** | Battery Powered |  | x |  |  |  |  |
| **3.6** | Low Battery Notification |  |  |  |  |  |  |
| **3.7** | User Friendly |  | x | x |  |  |  |
| **5.2** | Real-Time Response |  | x | x | x | x | x |
| **5.3** | Detection Quality |  |  |  | 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 |  |  |  |  | x |  |
| **8.3** | Adjustable Range |  |  | x |  |  |  |

**Table 12-2: Hardware Interface Layer Module Mapping**

## 12.4 Warning Layer Module Mapping

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

**Table 12-3: Warning Layer Module Mapping**

## 12.5 Notification Layer Module Mapping

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | **Modules** | **On Pattern** | **Off Pattern** | **Intensity Level** | **Object Pattern** | **Low Battery Pattern** |
| **No.** | **Requirement** |  |  |  |  | **Object** |  |
| **3.1** | On and Off |  | x | x |  |  |  |
| **3.2** | Detect Obstructions |  |  |  | x | x |  |
| **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 12-4: Notification Layer Module Mapping**

## 12.6 Analysis of Requirements Mapping

At a glance, we can see that the modules from all three layers play many critical roles in fulfilling the requirements specified from the System Requirements Specification document. Throughout the requirements, all modules play an important role in fulfilling Requirement 5.5 Battery Life and Requirement 6.4 Heat Dissipation. Without a consistent battery life all modules would not be able to function, and all modules must be able to dissipate heat to prevent the overheating of the system. The modules in the Hardware Interface Layer , Warning Layer and the Notification Layer are imperative in fulfilling Requirement 5.2 Real-Time Response. Without a reliable and consistent real-time response, the user would not be able to accurately avoid obstacles in their direction. In the Notification Layer all of the modules are critical in accomplishing Requirement 5.4 Vibration Intensity because in the Notification Layer, how much a vibration motor should vibrate is determined in this layer. While several modules in the different layers focus on certain requirements, all are critical to the Sight By Touch System.

**13. 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**.**

## 13.1 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.

## 13.2 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.

## 13.3 Acceptance Criteria

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

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

* 3.1 On and Off: The system shall be able to be turned on/off by the click 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 to improve utility for the visually impaired.

### 13.3.2 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 3 foot 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 within 10 feet max.
* 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.

### 13.3.3 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 and again when the battery has less than thirty minutes left of charge.

### 13.3.4 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, rusted 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

* [**EEPROM**](http://arduino.cc/en/Reference/EEPROM) - reading and writing to "permanent" storage
* [**Ethernet**](http://arduino.cc/en/Reference/Ethernet) - for connecting to the internet using the Arduino Ethernet Shield
* [**Firmata**](http://arduino.cc/en/Reference/Firmata) - for communicating with applications on the computer using a standard serial protocol.
* [**GSM**](http://arduino.cc/en/Reference/GSM) - for connecting to a GSM/GRPS network with the GSM shield.
* [**LiquidCrystal**](http://arduino.cc/en/Reference/LiquidCrystal) - for controlling liquid crystal displays (LCDs)
* [**SD**](http://arduino.cc/en/Reference/SD) - for reading and writing SD cards
* [**Servo**](http://arduino.cc/en/Reference/Servo) - for controlling servo motors
* [**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.

## Due Only Libraries

* [**Audio**](http://arduino.cc/en/Reference/Audio) - Play audio files from a SD card.
* [**Scheduler**](http://arduino.cc/en/Reference/Scheduler) - Manage multiple non-blocking tasks.
* [**USBHost**](http://arduino.cc/en/Reference/USBHost) - Communicate with USB peripherals like mice and keyboards.

## Esplora Only Library

* [**Esplora**](http://arduino.cc/en/Reference/EsploraLibrary) - this library enable you to easily access to various sensors and actuators mounted on the Esplora board.

## Arduino Robot Library

* [**Robot**](http://arduino.cc/en/Reference/RobotLibrary) - this library enables easy access to the functions of the Arduino Robot

## Arduino Yún Bridge Library

* [**Bridge Library**](http://arduino.cc/en/Reference/YunBridgeLibrary) - Enables communication between the Linux processor and the Arduino on the Yún.

## USB Libraries (Leonardo, Micro, Due, and Esplora)

* [**Keyboard**](http://arduino.cc/en/Reference/MouseKeyboard)- Send keystrokes to an attached computer.
* [**Mouse**](http://arduino.cc/en/Reference/MouseKeyboard) - Control cursor movement on a connected computer.

## Contributed Libraries

Communication (networking and protocols)

* [**Messenger**](http://www.arduino.cc/playground/Code/Messenger)- for processing text-based messages from the computer
* [**NewSoftSerial**](http://sundial.org/arduino/index.php/newsoftserial/) - an improved version of the SoftwareSerial library
* [**OneWire**](http://www.arduino.cc/playground/Learning/OneWire)- control devices (from Dallas Semiconductor) that use the One Wire protocol.
* [**PS2Keyboard**](http://www.arduino.cc/playground/Main/PS2Keyboard) - read characters from a PS2 keyboard.
* [**Simple Message System**](http://www.arduino.cc/playground/Code/SimpleMessageSystem) - send messages between Arduino and the computer
* [**SSerial2Mobile**](http://code.google.com/p/sserial2mobile/) - send text messages or emails using a cell phone (via AT commands over software serial)
* [**Webduino**](http://code.google.com/p/webduino/) - extensible web server library (for use with the Arduino Ethernet Shield)
* [**X10**](http://arduino.cc/en/Tutorial/X10)- Sending X10 signals over AC power lines
* [**XBee**](http://code.google.com/p/xbee-arduino/) - for communicating with XBees in API mode
* [**SerialControl**](http://www.arduino.cc/playground/Code/SerialControl)- Remote control other Arduinos over a serial connection

Sensing:

* [**Capacitive Sensing**](http://www.arduino.cc/playground/Main/CapSense) - turn two or more pins into capacitive sensors
* [**Debounce**](http://www.arduino.cc/playground/Code/Debounce) - for reading noisy digital inputs (e.g. from buttons)

Displays and LEDs:

* [**GFX**](https://github.com/adafruit/Adafruit-GFX-Library) - base class with standard graphics routines (by [Adafruit Industries](http://www.adafruit.com/))
* [**GLCD**](http://www.arduino.cc/playground/Code/GLCDks0108) - graphics routines for LCD based on the KS0108 or equivalent chipset.
* [**Improved LCD library**](http://web.alfredstate.edu/weimandn/arduino/LiquidCrystal_library/LiquidCrystal_index.html)**fixes** LCD initialization bugs in official Arduino LCD library
* [**LedControl**](http://www.arduino.cc/playground/Main/LedControl) - for controlling LED matrices or seven-segment displays with a MAX7221 or MAX7219.
* [**LedControl**](http://www.wayoda.org/arduino/ledcontrol/index.html) - an alternative to the Matrix library for driving multiple LEDs with Maxim chips.
* [**LedDisplay**](http://www.arduino.cc/playground/Main/LedDisplay) - control of a [HCMS-29xx](http://sigma.octopart.com/23295/datasheet/Avago-HCMS-2973.pdf) scrolling LED display.
* [**Matrix**](http://wiring.org.co/reference/libraries/Matrix/index.html) - Basic LED Matrix display manipulation library
* [**PCD8544**](https://github.com/adafruit/Adafruit-PCD8544-Nokia-5110-LCD-library) - for the LCD controller on Nokia 55100-like displays (by [Adafruit Industries](http://www.adafruit.com/))
* [**Sprite**](http://wiring.org.co/reference/libraries/Sprite/index.html) - Basic image sprite manipulation library for use in animations with an LED matrix
* [**ST7735**](https://github.com/adafruit/Adafruit-ST7735-Library) - for the LCD controller on a 1.8", 128x160 TFT screen (by [Adafruit Industries](http://www.adafruit.com/))

Audio and Waveforms:

* [**FFT**](http://wiki.openmusiclabs.com/wiki/ArduinoFFT) - frequency analysis of audio or other analog signals
* [**Tone**](http://code.google.com/p/arduino-tone/) - generate audio frequency square waves in the background on any microcontroller pin

Motors and PWM:

* [**TLC5940**](http://www.arduino.cc/playground/Learning/TLC5940) - 16 channel 12 bit PWM controller.

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.

Utilities:

* [**PString**](http://sundial.org/arduino/index.php/pstring/)- a lightweight class for printing to buffers
* [**Streaming**](http://arduiniana.org/libraries/streaming/) - a method to simplify print statements