envision today

School of Engineering Science
Simon Fraser University
8888 University Drive
Burnaby, BC
V5A 1S6

March 14, 2013

Dr. Andrew Rawicz
School of Engineering Science
Simon Fraser University
Burnaby, British Columbia
V5A 1S6

**Re: ENSC 440 Design Specification for NaviCane: Navigation-Assisting White Cane**

Dear Dr. Rawicz,

Enclosed with this letter is our *Design Specification for NaviCane: Navigation-Assisting White Cane* for our Engineering Science 440 Capstone Project class. Our design and implementation will provide additional ways for those that are visually impaired to navigate freely in almost all urban areas.

In our design specifications, we will be outlining the information and specifications necessary for the design process of the proof-of-concept model. In addition, we will be indicating any future design improvements for the product even though they will not be implemented in our current stages of development. This document will allow our engineers and managers to follow the requirements and development activities when bringing the product into fruition.

Envision Today is composed of a four-member team of senior engineering students whose knowledge and expertise allows for an effective delivery in products featuring hardware and software solutions. The team, Vincent Guan, Edwin Leong, Raymond Li, and Darren Tong, has a background in systems and computing engineering. If you have any questions or concerns regarding our design specifications, please feel free to contact Raymond Li via email at rla41@sfu.ca.

Sincerely,

[illegible]

Vincent Guan
CEO
Envision Today

Enclosure: *Design Specification for NaviCane: Navigation-Assisting White Cane* envision today

# Design Specification for NaviCane: Navigation-Assisting White Cane

Project Team:
Vincent Guan
Edwin Leong
Raymond Li
Darren Tong

Contact Person:
Raymond Li
rla41@sfu.ca

Submitted to:
Dr. Andrew Rawicz - ENSC 440
Steve Whitmore - ENSC 305
School of Engineering Science

Issued Date:
March 14, 2013

Revision:
1.0

## Executive Summary

The NaviCane, Navigation-Assisting White Cane, is a product that Envision Today is developing to enhance the quality of life for the visually impaired. NaviCane allows their users to program desired locations to their white cane. By following the haptic feedback provided by the cane, the user is able to travel to new locations without relying on further assistance. The NaviCane reduces the concerns arising from current navigational-assisting devices, such as cognitive and economical requirements.

The design specification for the Navigation-Assisting White Cane illustrates the detailed processes for our proof-of-concept model. However, justification for our designs and future improvements for our models will discussed. Our document will mainly be comprised of sections regarding the mechanical, hardware, embedded, and software designs. Test plans are provided and explained to ensure compliance each of NaviCane's components.

## Table of Contents

## List of Figures

## List of Tables

## List of Equations

## Glossary

| | |
|---|---|
| **ADC** | Acronym; Analog-to-Digital Converter |
| **API** | Acronym; Application Programming Interface |
| **CNIB** | Acronym; Canadian National Institute for the Blind |
| **GUI** | Acronym; Graphical User Interface |
| **GPS** | Acronym; Global Positioning System |
| **HTML** | Acronym; HyperText Markup Language |
| **I2C** | Definition; Multi-master serial single-ended computer bus |
| **IDE** | Acronym; Integrated Development Environment |
| **LDO** | Acronym; Low-Dropout |
| **Mbps** | Acronym; Megabits per Second |
| **OS** | Acronym; Operating System |
| **PC** | Acronym; Personal Computer |
| **PWM** | Acronym; Pulse-Width Modulation |
| **UART** | Acronym; Universal Asynchronous Receiver/Transmitter |
| **USB** | Acronym; Universal Serial Bus |
| **XML** | Acronym; Extensible Markup Language |

# 1. Introduction

The NaviCane, Navigation-Assisting White Cane, is a mobility device being developed for the visually impaired. NaviCane allows users to freely program their desired destinations to the cane and it will provide feedback to safely guide them to their target location through an optimized path that consumes the least amount of travel time. The design specification describes the technical details for the design of each component of the NaviCane.

## *1.1 Scope*

This document provides in-depth technical aspects of each component of the NaviCane's design. It will appropriately correspond to the functional requirements provided by *Functional Specification for NaviCane: Navigation-Assisting White Cane*. As a result, the design details will provide a deeper understanding as to how the functional requirements will be implemented. The design specification will only provide the technical points of the proof-of-concept and prototype models; therefore, only the fundamental functional requirements will be discussed.

## *1.2 Intended Audience*

The functional specification will be used by all members at Envision Today. It will allow us to measure the progression of our design during the development phase. Furthermore, this document becomes our guideline for function implementation, which ensures compliance between both design and implementation phases. The testing of the product will also follow the procedures listed below as a final evaluation.

## 2. System Specifications

NaviCane's top-level block diagram is modelled below in Figure 2-1.

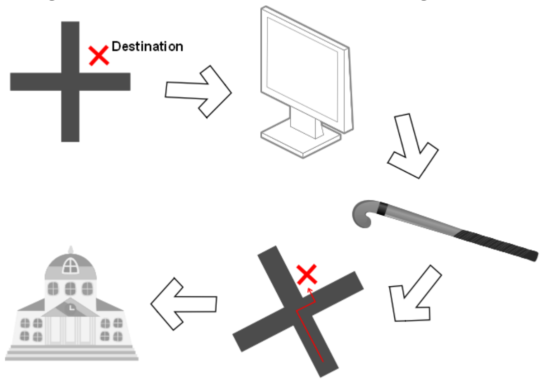


Figure 2-1: Top-level Block Diagram

The NaviCane can be divided into two subsystems: software application and hardware. The software application is where the user will indicate their starting location and destination via computer interface either by way of address or name. The program will return details of the location in speech readable text allowing the user to determine whether the information is correct. The application will then retrieve the longitudinal and latitudinal coordinates of the user-specified destinations as well as routes and nodes demarcating the route from the user's current location to their destination. All required data will then be transmitted to the microprocessor through a USB port located at the top of the handle of the cane. The interface will also detail the current battery charged status of the device in speech readable text. The software application process flow diagram is shown in Figure 2-2.

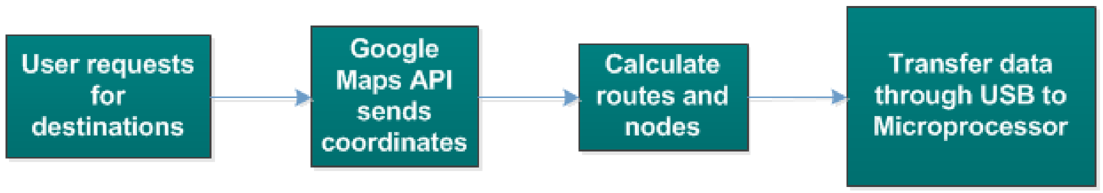

User requests for destinations
Google Maps API sends coordinates
Calculate routes and nodes
Transfer data through USB to Microprocessor

Figure 2-2: NaviCane's Software Application Process Flow Diagram

Once the user is ready to go to their destination, the cane will be detached from its USB connection. Once removed and held, the user can toggle a switch on the cane located near the handle that will activate the haptic feedback that will direct the user in the direction of the next waypoint. The haptic feedback will be produced by two vibration motors located in the handle of the cane; these vibration motors are located opposite one another and as the device has a forward facing portion, are located on the left and right side with respect to this forward facing.

As the cane is rotated in the direction of the next waypoint with respect to the forward facing, the haptic feedback will increase in strength with a specific frequency and intensity indicating to the user the direction of the next waypoint. This direction is determined by a compass which is located inside the handle facing in the forwards position. Once the current waypoint has been reached within a reasonable distance, the next waypoint in the route will become active. Once the final waypoint and thus destination has been reached, a very specific pattern of haptic feedback will be given to indicate to the user that they are in the immediate vicinity of their destination. The top level process flow diagram is shown in Figure 2-3 while a detailed diagram of the location of various features is shown in Figure 2-4.

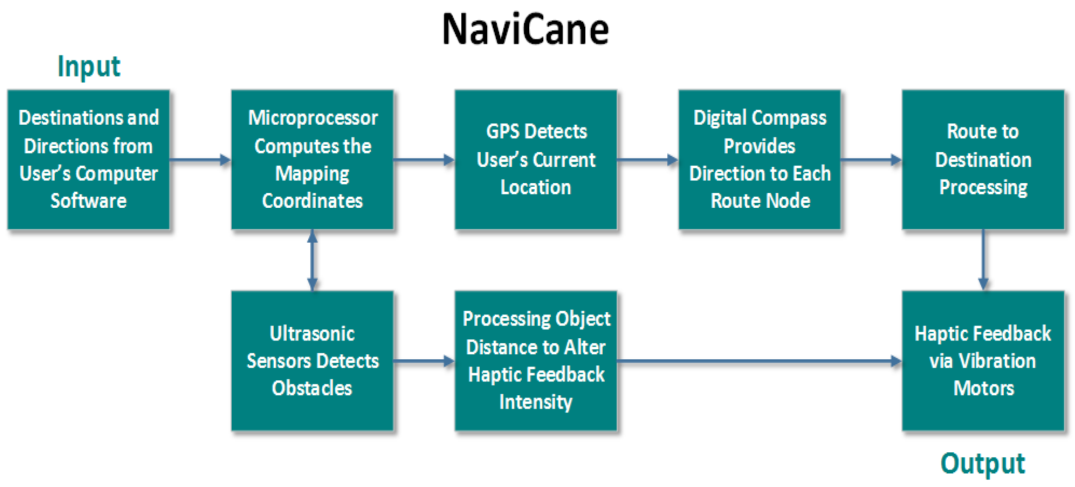

NaviCane
Input
Destinations and Directions from User's Computer Software
Microprocessor Computes the Mapping Coordinates
GPS Detects User's Current Location
Digital Compass Provides Direction to Each Route Node
Route to Destination Processing
Ultrasonic Sensors Detects Obstacles
Processing Object Distance to Alter Haptic Feedback Intensity
Haptic Feedback via Vibration Motors
Output

Figure 2-3: NaviCane's Top-Level Block Diagram

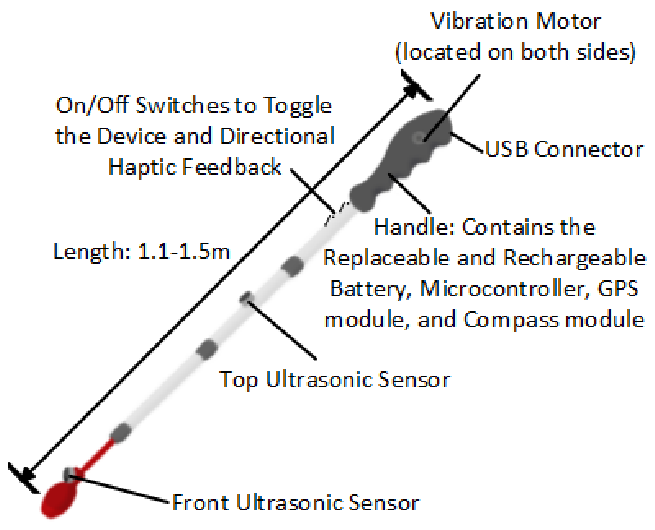

Vibration Motor (located on both sides)
On/Off Switches to Toggle the Device and Directional Haptic Feedback
USB Connector
Handle: Contains the Replaceable and Rechargeable Battery, Microcontroller, GPS module, and Compass module
Length: 1.1-1.5m
Top Ultrasonic Sensor
Front Ultrasonic Sensor

Figure 2-4: NaviCane's Mechanical Enclosure

# 3. Mechanical Design

## *3.1 Inner Enclosure*

The following requirements are met in our proof-of-concept model:

| | |
|---|---|
| **[R3-B]** | The GPS module can be toggled on or off via a switch. |
| **[R4-B]** | The device will be held in a specific way. |
| **[R13-A]** | The device shall communicate and recharge through the USB interface. |
| **[R15-B]** | No electronic components will be exposed to the user. |
| **[R18-B]** | The power button will be mounted on the outside. |
| **[R20-C]** | The mechanical components shall not be physically intrusive. |
| **[R22-C]** | The microprocessor, GPS module, compass module, and battery shall be stored within the handle of the device. |
| **[R46-A]** | The device shall not exceed 3 lbs. |

The inner mechanical enclosure is located within the handle of the NaviCane. This enclosure must provide protection for the microcontroller, GPS module, compass module, and the battery. While designing the inner enclosure, we considered the following:

- Signal interference
- Sustainability
- Weight
- Structural integrity
- Ergonomics and aesthetics

After careful consideration, the inner enclosure will be composed of either composite wood or lego. Wood and plastic certainly outclass metal in terms of not providing interference against our GPS and compass modules. The thickness of the material is also a primary concern for signal interference. With this in mind, the proposed inner enclosure is projected to be approximately 50-60 millimeters in thickness. Since sustainability is an issue, both options of wood and lego are valid. Naturally, wood is biodegradable, but composite wood undergo manufacturing and chemical processing in order to circumvent rotting and biodegradation. Although composite wood is not biodegradable, it can still be recycled. Lego is a great alternative in regards to sustainability because it can be taken apart and recycled. To design the NaviCane handle with the hardware components inside, the dimension of each component is considered. The dimension of each hardware component is shown in Table 3-1.

Table 3-1: Dimensions of Hardware Components

| **Hardware Components** | **Dimensions** |
|---|---|
| LS20031 GPS Receiver | 30 x 30 x 7.2 mm |
| LSM303DLMTR Compass Module | 20.5 x 20.5 mm |
| ATmega32U4 Microcontroller | 18 x 34.3 mm |
| Rechargeable Lithium Polymer Battery | 5.8 x 54 x 60 mm |
| Battery Charger Circuit | 29.4 x 10.8 mm |

The enclosure is also made to be ergonomically-enhanced for the visually impaired, where easy hand placement to hold the handle should be comfortable. The mechanical design is shown in Figure 3-1. The design of the handle is ambidextrous, allowing both left and right handed users to have a good grip on the NaviCane. The design is also made to only allowing one direction of holding the device due to the hardware components. The ultrasonic sensors on the cane require a static position on the handle. Even if it is slightly rotated, the cane will misinterpret the sensor's readings because it is unable to distinguish the position of the cane. Similarly, the GPS module requires satellite communication. If the module is distorted, it may affect the connection between the module and the satellite. Although the compass module has its own tilt compensation, it is still not 100% compensated. Therefore, the inner enclosure is designed to allow the hardware components for maximum efficiency. In addition, the size of the handle is constructed in a way to have all the components fit inside completely without any obtrusive object that can harm the user.

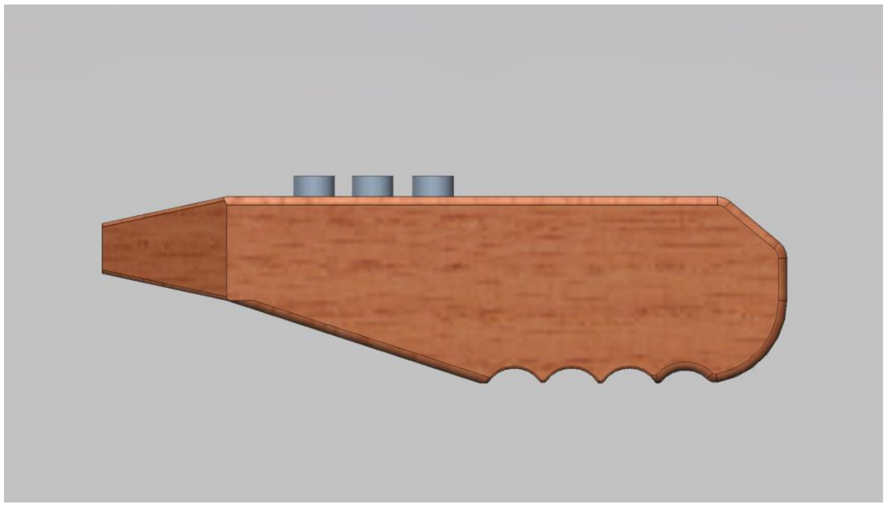


Figure 3-1: The Mechanical Design of the NaviCane Handle

The handle is made to have easy access to the USB interface. The placement of the USB is put on the back to allow quick access in transferring information from the software client to the NaviCane device. The diagram is shown in Figure 3-2 where the USB communication between the software on the computer is interacting with the handle. The USB interface is also used for charging the battery of the device.

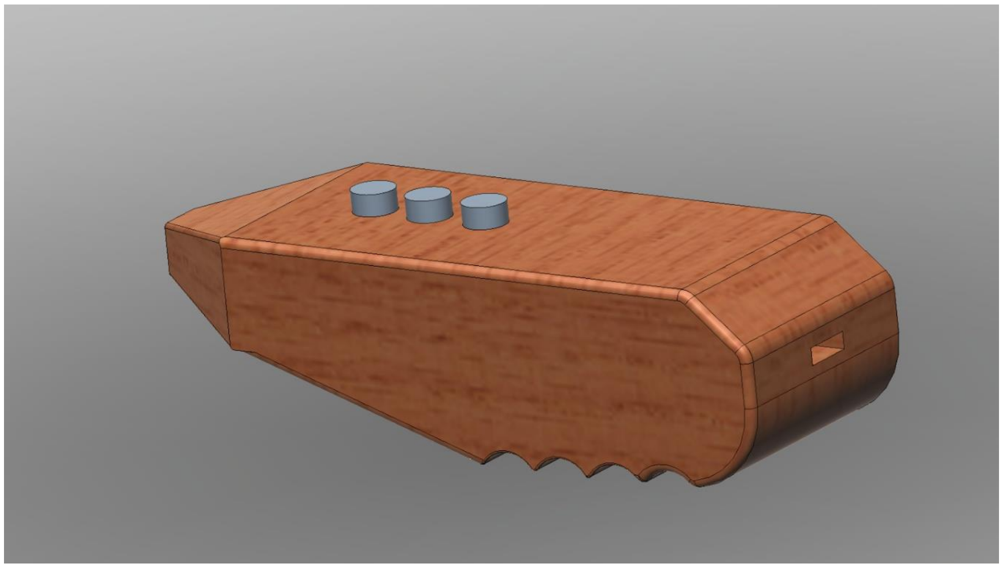


**Figure 3-2: The USB Interface is Located on the Back of the Handle**

The NaviCane handle also takes into consideration of pre-existing white cane user. The handle can be attached to a white cane shown in Figure 3-3. The enclosure is made to be secure enough for tactile feedback ensuring that the user will not damage the device. The hole will vary in size to allow for different types of white cane attachment and not limit the choices of the visually impaired.

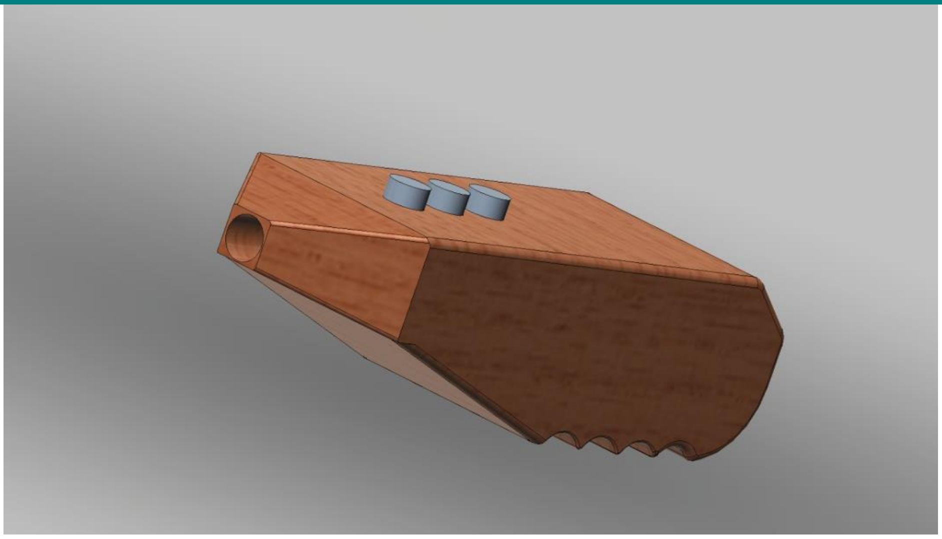


Figure 3-3: Attaches to Pre-Existing Canes

The NaviCane handle with all the components inside should not be heavy enough to be a distraction for the user. The NaviCane handle takes in consideration of the weight on pre-existing white cane and adding it together should not be anymore heavier. The heaviest component is the battery which weighs in 36 grams. The diagram in Figure 3-4 illustrates the components inside the NaviCane handle.

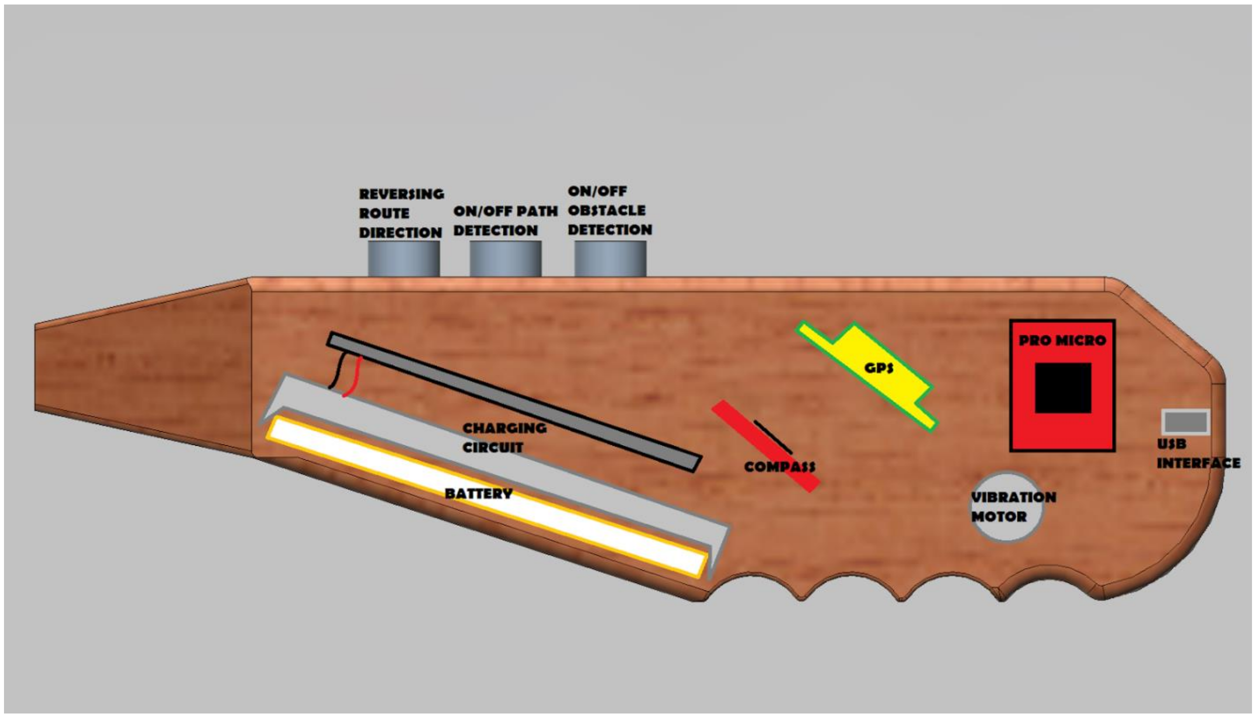


Figure 3-4: The Components inside the NaviCane Handle

The placement of the battery is at the bottom of the handle to allow easy access for replacement. The GPS and the compass are both place slanted at about 45 degrees since the user will be holding it in a down position. This is to compensate for the tilt especially for the compass when guiding the visually impaired to the new location. The compass inside the NaviCane has tilt compensation but to ensure optimal usage, the compass is position in such a way to minimize the error. The GPS is place closer to the top of the handle to minimize signal interfere when getting information from the satellite. The two vibration motors are put on both sides to guide the user to the new destination. By alternating the different pulse and the width, it can be used for both obstacle detection and guiding the visually impaired to the direction they should be heading. The three buttons depicted in figure 3-4 have functionality listed that may or may not be included for proof-of-concept but are shown anyway for completeness.

## 3.2 Cane

**[R47-B]** The device shall operate similarly to a basic white cane

The proof-of-concept model will be using a 42 inch guide cane from CNIB. The guide cane is generally used for tactile feedback by holding it in front and across the user for protection. Unless feeling for obstacles, the cane is not constantly sweeping. Although the proof-of-concept design uses a guide cane, future models will incorporate designs for long canes, identification canes, and support canes. Figure 3-5 shows the proof-of-concept for the cane design.

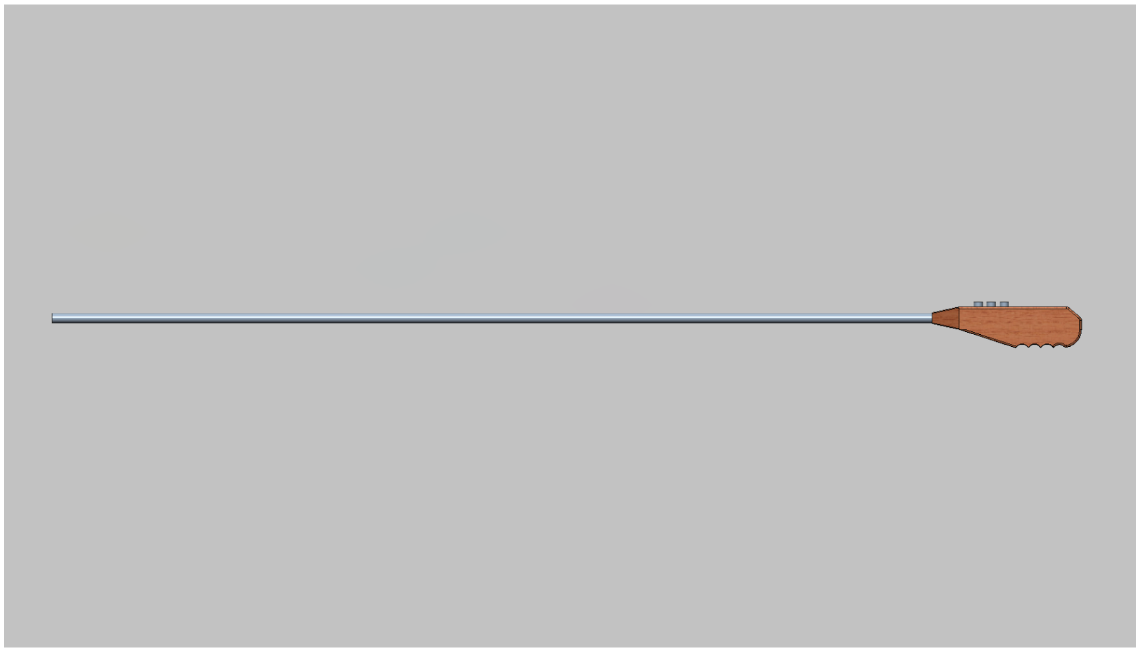


**Figure 3-5: NaviCane Handle Attached to a Cane**

## 4. Electrical Hardware Design

This section explores the hardware design and components used in our proof-of-concept model of the NaviCane. As a proof-of-concept design, the NaviCane currently utilizes modules and PCBs built from third party manufacturers in order to reduce costs and development time. For a production version of the NaviCane, a customized PCB will be utilized to incorporate all of the components, such as the microprocessor, battery charging circuit, GPS module, and the magnetometer, onto a single compact PCB.

The following requirements are met by our proof-of-concept model:

| | |
|---|---|
| **[R11-A]** | The device will operate with a 2000 mAh polymer lithium battery. |
| **[R12-A]** | The battery will operate at a maximum voltage of 4.2V. |
| **[R13-A]** | The device shall communicate and recharge through the USB interface. |
| **[R14-A]** | The device will operate at a typical voltage of 3.3V. |
| **[R16-C]** | The battery must be easily accessible for replacement. |
| **[R39-A]** | The device shall not overcharge or over-discharge the battery. |

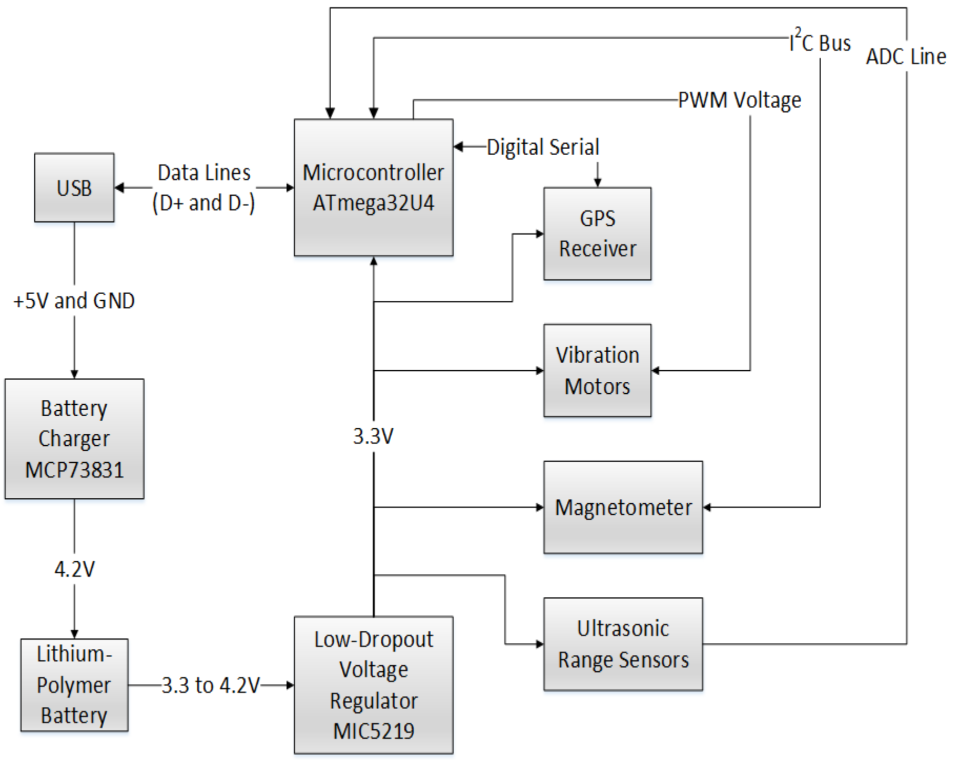


Figure 4-1: NaviCane's Hardware System[10]

## 4.1 Microprocessor

The NaviCane utilizes a low-powered ATmega32U4 microprocessor in its designs. This microprocessor is supported by the Arduino IDE, which provides open-sourced software libraries and tools for powerful and flexible programming. This particular microprocessor was chosen due to its low power consumptions (runs at 3.3V) and tiny form factor. In order to minimize the cost and development time of the proof-of-concept model, we utilized a pre-assembled board that features only the microprocessor as well as a simple voltage regulator, which would be explained in section 4.2.

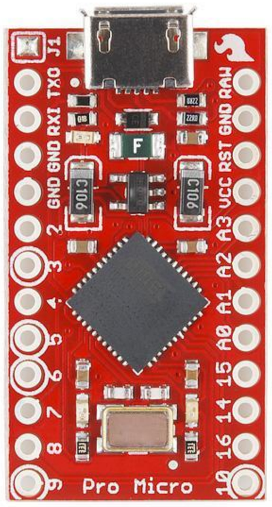


**Figure 4-2: Pro Micro - ATmega32U4 [10]**

The main features of the ATmega32U4 are listed as follows [10]:

- Runs at 3.3V at 8MHz
- 32KB of Flash, 2.5KB of SRAM, 1KB EEPROM
- Four 10-bit ADC pins
- Twelve digital I/0s with five pins cable of PWM output
- I$^2$C bus supported
- Hardware serial (UART) lines supported

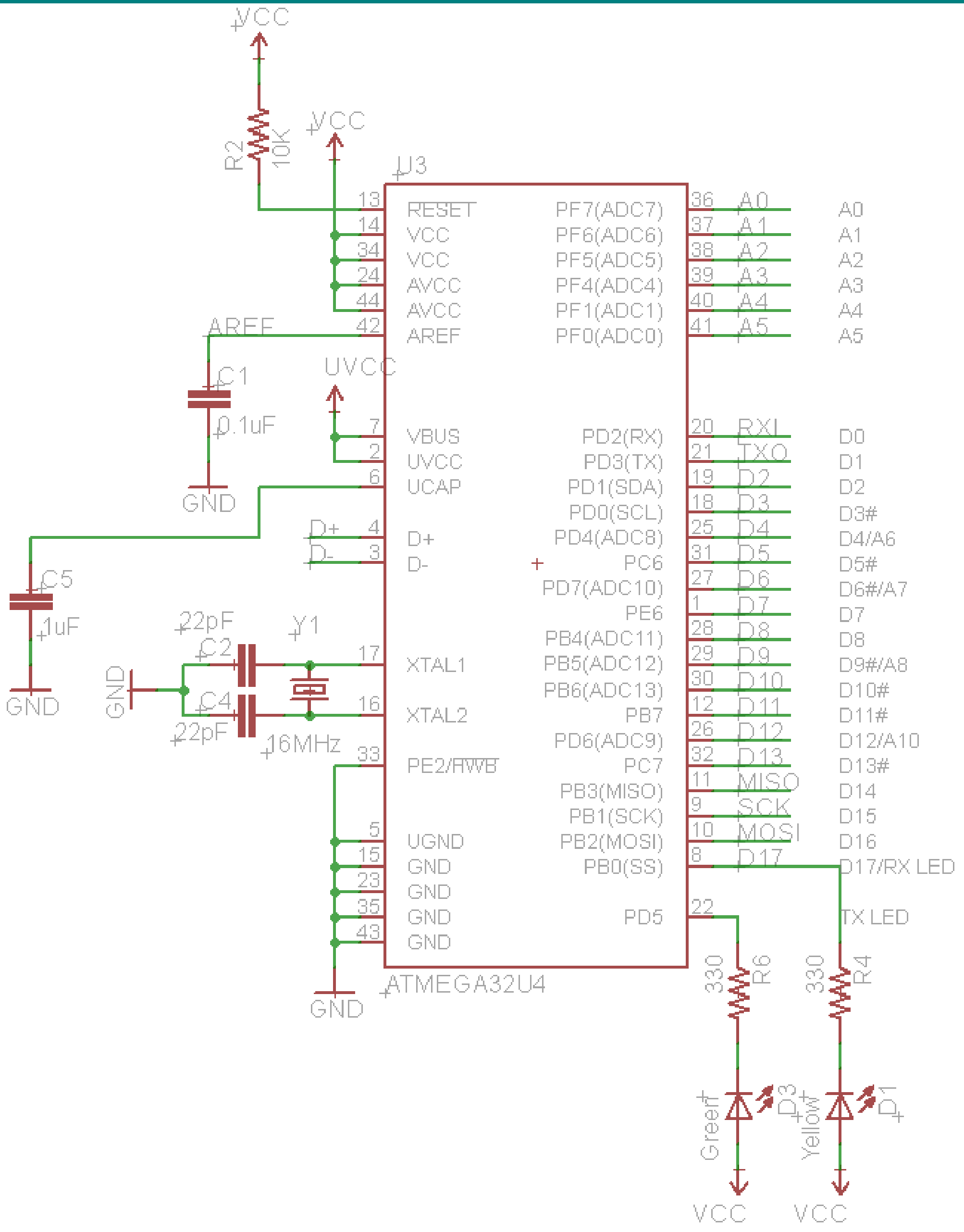


Figure 4-3: Schematic Diagram of the ATmega32U4 [10]

## 4.2 Main Power Supply

To power each of the individual subsystems of the NaviCane, the MIC5219 low-dropout (LDO) voltage regulator will be utilized. This integrated circuit is pre-assembled on our Pro Micro board and will be used to power the microcontroller, vibration motors, GPS module, ultrasonic range sensors, and the magnetometer.

The general specification of the voltage regulator is shown below [2]:

- Input voltage: 2.5V to 12V
- Output voltage: 3.3V
- Maximum 500mA peak current
- Dropout voltage (at output currents of 150mA): 175 mV
- Current and thermal limiting (maximum power dissipation = 455 mW)

**Table 4-1: Maximum power consumption of each subsystem**

| **Hardware Subsystem** | **Maximum Current Consumption (mA)** |
|---|---|
| Microprocessor (ATmega32u4) | <0.5 |
| Two Haptic Feedback Vibration Motors | 85 + 85 |
| GPS Module | 41 |
| Two Ultrasonic Range Sensors | 2 + 2 |
| Magnetometer | <0.5 |
| **Total:** | **216** |

The power dissipated by the voltage regulator is given in the following equation:

$$P_D = (V_{in} - V_{out}) \cdot I_{out} + V_{in} \cdot I_{GND}$$

**Equation 4-1: Power Dissipation of Voltage Regulator**

Since the voltage regulator will be powered by a lithium-polymer battery, the maximum input voltage is 4.2V. Therefore, at a current load of 216 mA, the power dissipation is estimated to be 194 mW, which is less than half the maximum power dissipation of the regulator. Since both the current load and power dissipation is well below the rated maximums, the voltage regulator is expected to be more than adequate for the NaviCane.

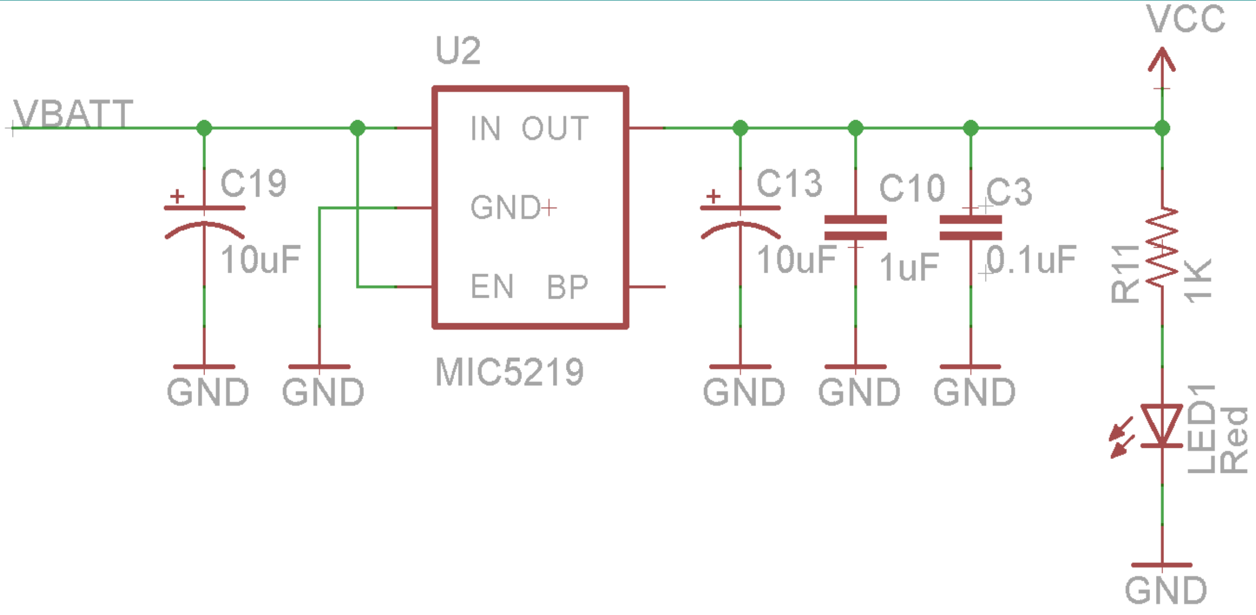


Figure 4-4: MIC5219 Voltage Regulator [10]

The input voltage of the voltage regulator is supplied by a rechargeable battery that is user-replaceable via a JST connector.

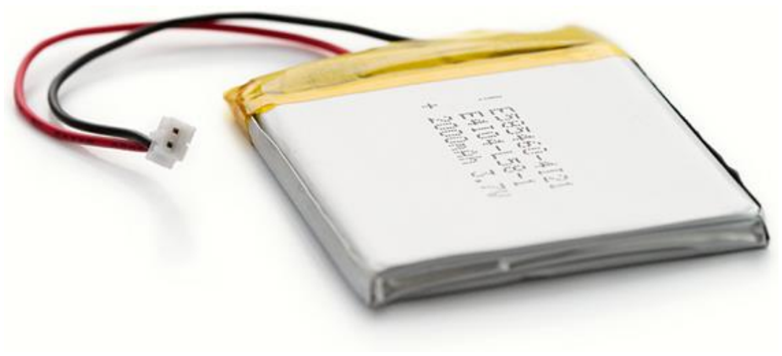


Figure 4-5: Rechargeable Lithium Polymer Battery 2000 mAh [9]

From Table 4-1, it is clear that the 2000 mAh rechargeable battery is able to offer at least 8.87 hours of constant usage. This calculated figure is for the worst case scenario when both vibration motors are continuously vibrating at full intensity, which will not be occurring in real-world applications.

### *4.3 Battery Charging*

As a proof-of-concept model, the NaviCane will support charging via USB interface. In a prototype/production model, we wish to have the NaviCane support the QI wireless inductive charging standard. USB charging is provided through Microchip's Li-Polymer charge management controller (MCP73831). The +5V power line from the USB interface will be used to power this circuit. The output is directly connected to the nominal 3.7V battery.

The main technical features of the charger are listed as follows [3]:

- Input voltage range: 3.75V to 6V
- Regulated output voltage: 4.168V to 4.232V (Typically at 4.20V)
- Charging/Output current: 500 mA
- Reverse discharge protection

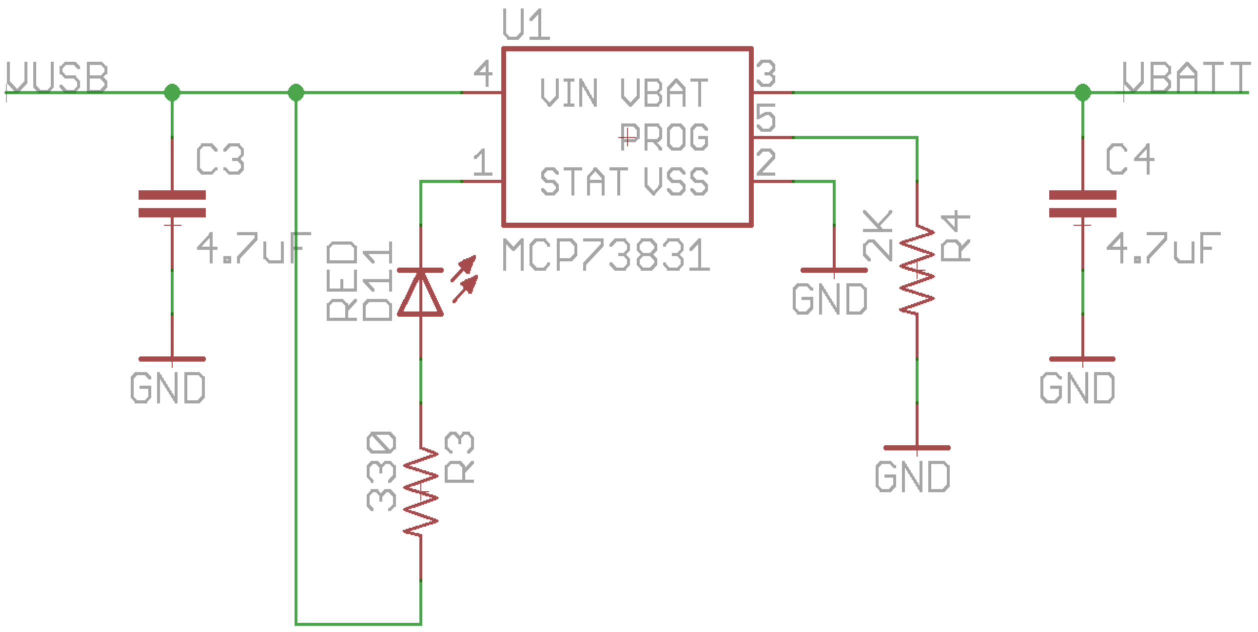


**Figure 4-6: MCP73831 Battery Charger [7]**

Charging at a rate of 500 mA, it is clear that a 2000 mAh battery will take approximately 4 hours to fully charge from an empty state.

### *4.4 Haptic Feedback Vibration Motors*

In order for the NaviCane to provide feedback to the user, we will be relying on vibration motors implemented onto the handle as shown below in Figure 4-7. The Precision Microdrive's vibration motor was selected as part of NaviCane's designs because of its small form factor.

**Figure 4-7: Precision Microdrive's Vibration Motor [5]**

The main technical specifications of the motor are listed as follows [5]:

- Voltage range: 2.5~3.8V
- Rated Current: 75 mA
- Start Voltage: 2.3V
- Start Current: 85 mA
- Terminal Resistance: 75 Ohms
- Vibration Amplitude: 0.8 G

The vibration motor will be controlled by a PWM line supplied by the Arduino microcontroller. This will allow precise control of the vibration intensity and duration of the pulses. A transistor will be used to drive the power to the motor. The 3.3V source powering the motor is supplied by the main voltage regulator described in Section 4.2. A protection diode is added to prevent the high voltage spikes (caused by a sudden change in the magnetic field of the motor when it stops) from damaging the transistor.

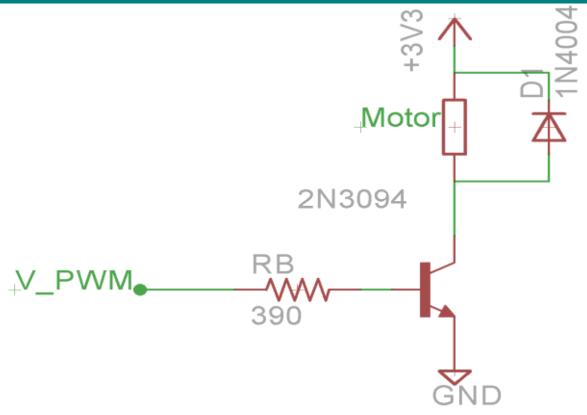


Figure 4-8: Transistor Circuit Driving the Motor

### 4.5 Ultrasonic Range Sensors

In order for the NaviCane to detect obstacles, we have included two ultrasonic sensors onto the cane. As shown in Figure 2-4, one ultrasonic sensor will be situated near the bottom of the cane and another will be placed near the middle. Each of the ultrasonic sensors will transmit its data through an analog signal line to the microprocessor's ADC line. The Maxbotix EZ0 model was selected for the NaviCane because of its small form factor, accurate distance measurements, and low power consumptions.

The main specifications of the ultrasonic sensor are listed as follows [1]:

- Operating voltage: 2.5V to 5.5V
- Analog voltage line yielding ~6.4 mV/inch accuracy when operating at 3.3V
- Low power consumption (typically 2 mA current draw)

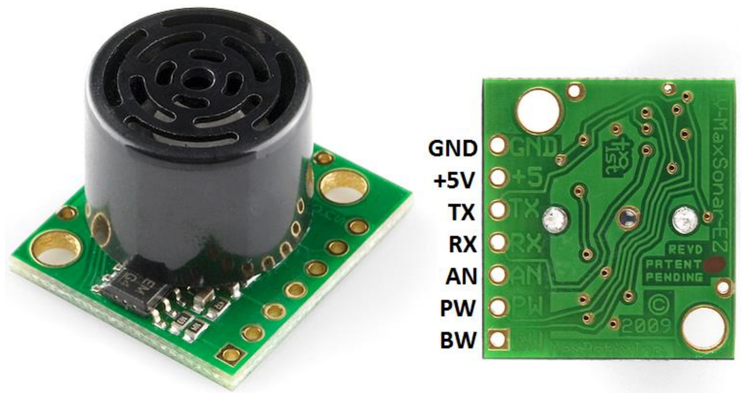


Figure 4-9: Maxbotix EZ0 - Ultrasonic Range Sensors [11]

The ultrasonic range sensors will be powered by the 3.3V line supplied by the main voltage regulator as indicated in section 4-2. Running at 3.3V, the general beam detection pattern is shown below in Figures 4-10 and 4-11.

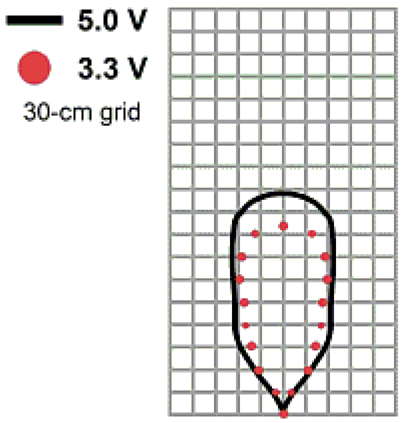


Figure 4-10: General Beam Characteristics [1]

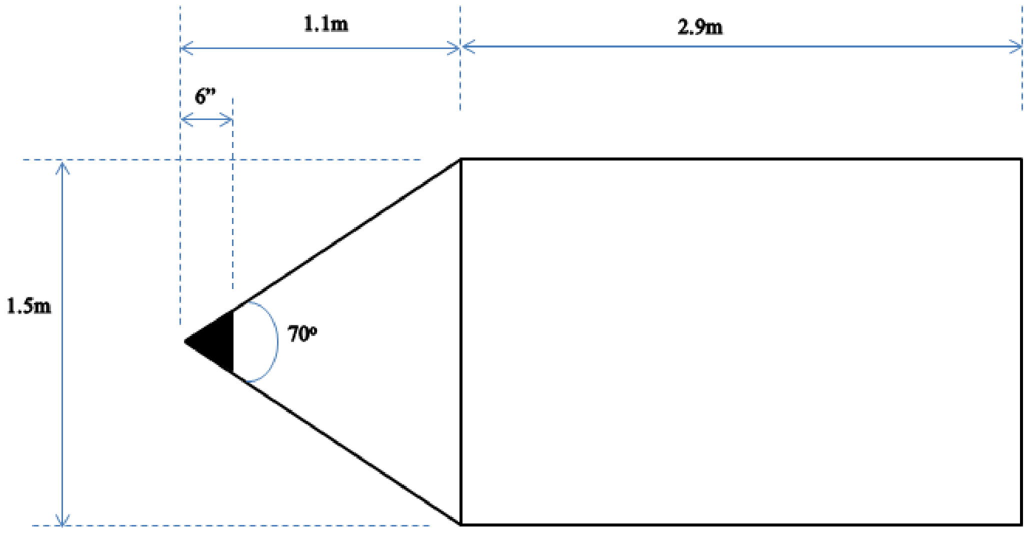


Figure 4-11: General Beam Characteristics [4]

From Figure 4-11, the ultrasonic sensor is unable to reliably detect an obstacle that is 6 inches away from it. Any objects detected from 0 inches to 6 inches will be indicated by the analog signal line as being 6 inches away. This would not affect the NaviCane adversely because any obstacle within a couple feet of the user will already set the intensity of the vibration motors to maximum.

## 4.6 GPS Receiver

The NaviCane will utilize the LS20031 GPS receiver, which will allow quick and precise tracking of the user's location. This specific module was selected because of its accuracy and ability to receive a signal even in dense urban areas.

The general specifications of the GPS receiver includes [6]:

- 5 Hz refresh rate (updates 5 times a second)
- Operates at 3.3V
- Typical current consumption of 41 mA
- Tracks up to 66 satellites
- Embedded antenna
- Requires 35 seconds to determine the user's location on a cold start (completely powered down)
- Requires 2 seconds to determine the user's location on a hot start (GPS receiver powered on from standby mode)

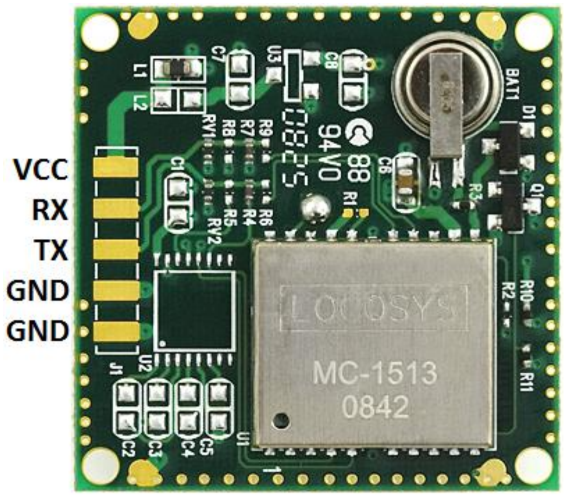


Figure 4-12: GPS Receiver LS20031 [6]

The GPS receiver is powered by the main 3.3V regulator and is connected to the microprocessor via software serial lines.

## *4.7 Magnetometer (Compass)*

In order for the NaviCane to accurately determine the direction of a specified route, it is important to have an accurate magnetometer. The LSM303DLH was selected as part of NaviCane's designs because it incorporates an accelerometer, which allows for tilt compensation. This means that the compass will be able to reliably output data (through the I$^2$C bus connected to the microprocessor) even when the NaviCane itself is tilted at drastic angles. It will be powered by the 3.3V source supplied by the main voltage regulator.

Some of the key specifications of the magnetometer are [8]:

- Input supply voltage: 2.16V to 3.6V
- +/- 1.3 to +/- 8.1 gauss magnetic field full-scale
- Low current consumption: typically 360 µA

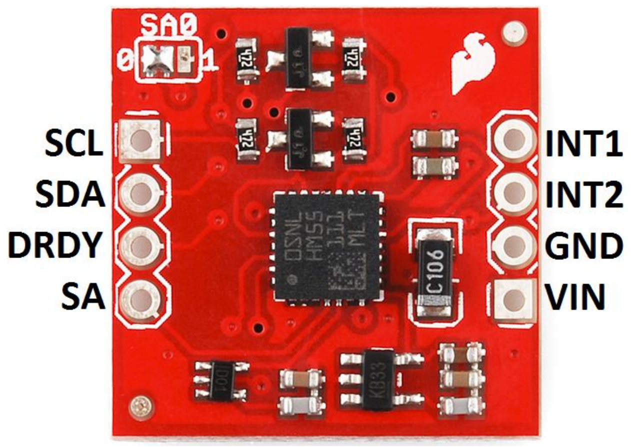


**Figure 4-13: Magnetometer LSM303DLMTR [8]**

# 5. Arduino Embedded Software Design

## 5.1 Microcontroller Logic

| | |
|---|---|
| **[R30-A]** | The GPS signal requires 35 seconds to operate on a cold start. |
| **[R31-A]** | The GPS signal requires 2 seconds to operate on a hot start. |
| **[R40-A]** | The device shall startup and guide the user within 45 seconds. |

The microcontroller, on startup, initializes with its setup function by enabling the communications port for the serial port, enabling the communications port for the GPS, and enabling and setting the calibration values for the magnetometer. It will then delay for approximately 35 seconds before exiting in order to ensure the GPS has time to acquire satellites. Upon exiting the setup function, it will jump into the main loop function. Within the loop function, most of the functionality of the workflow are dealt with. This includes the GPS, the Haptic Feedback Motors, and the Compass. This loop also holds the list of waypoints for the currently loaded path and a pointer which indicates the current waypoint. The looped section currently takes less than 5 seconds to complete a cycle and begin directing the user. Figure 5-1 illustrates the basic workflow.

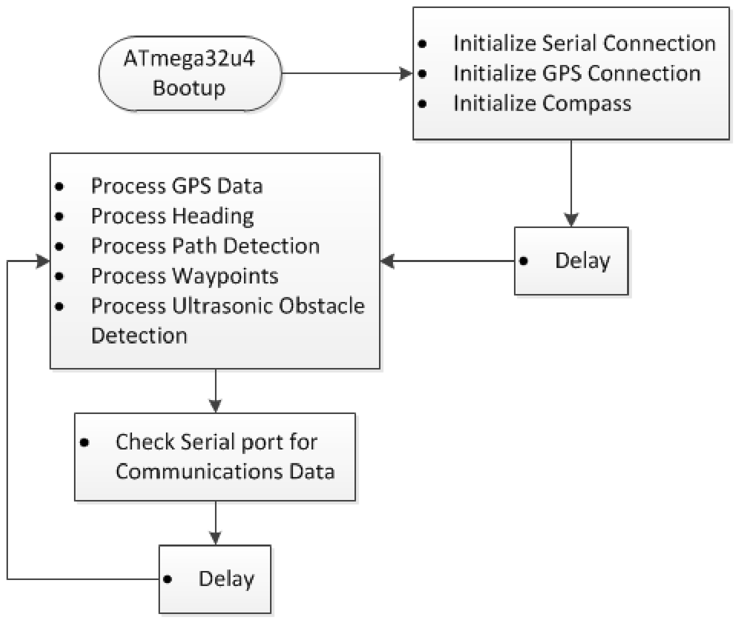

ATmega32u4 Bootup
Initialize Serial Connection
Initialize GPS Connection
Initialize Compass
Delay
Process GPS Data
Process Heading
Process Path Detection
Process Waypoints
Process Ultrasonic Obstacle Detection
Check Serial port for Communications Data
Delay

Figure 5-1: Illustration of Overall Microcontroller Workflow

The Serial Communications between the hardware and the PC software is handled in between loop cycles in serialEvent. The workflow of each individual function will be discussed in the following sections.

## 5.2 Serial Communications Protocol

The communication between the embedded software on the microcontroller and the user interface on the PC is checked after every cycle of the main loop. If data is sent from the serial port on the PC to the Arduino, then the serialEvent will hold up the beginning of the next loop cycle and wait to process more data from the serial port for a period of time. Data sent from the PC Software will be given signifiers denoting to the embedded software which action is being requested. Currently, there are three types of requests; the first type of request contains the number of waypoints in the list of waypoints that the Arduino is to store; the second contains the actual list of waypoints for the Arduino to store in [LAT,LON] format to 6 decimal places. The exact format of the protocol is subject to change. The final packet type will trigger the Arduino to dump its current waypoint list and pointer. If no more data is sent through the serial port for a period of time, the serialEvent handler will exit. Figure 5-2 illustrates this workflow.

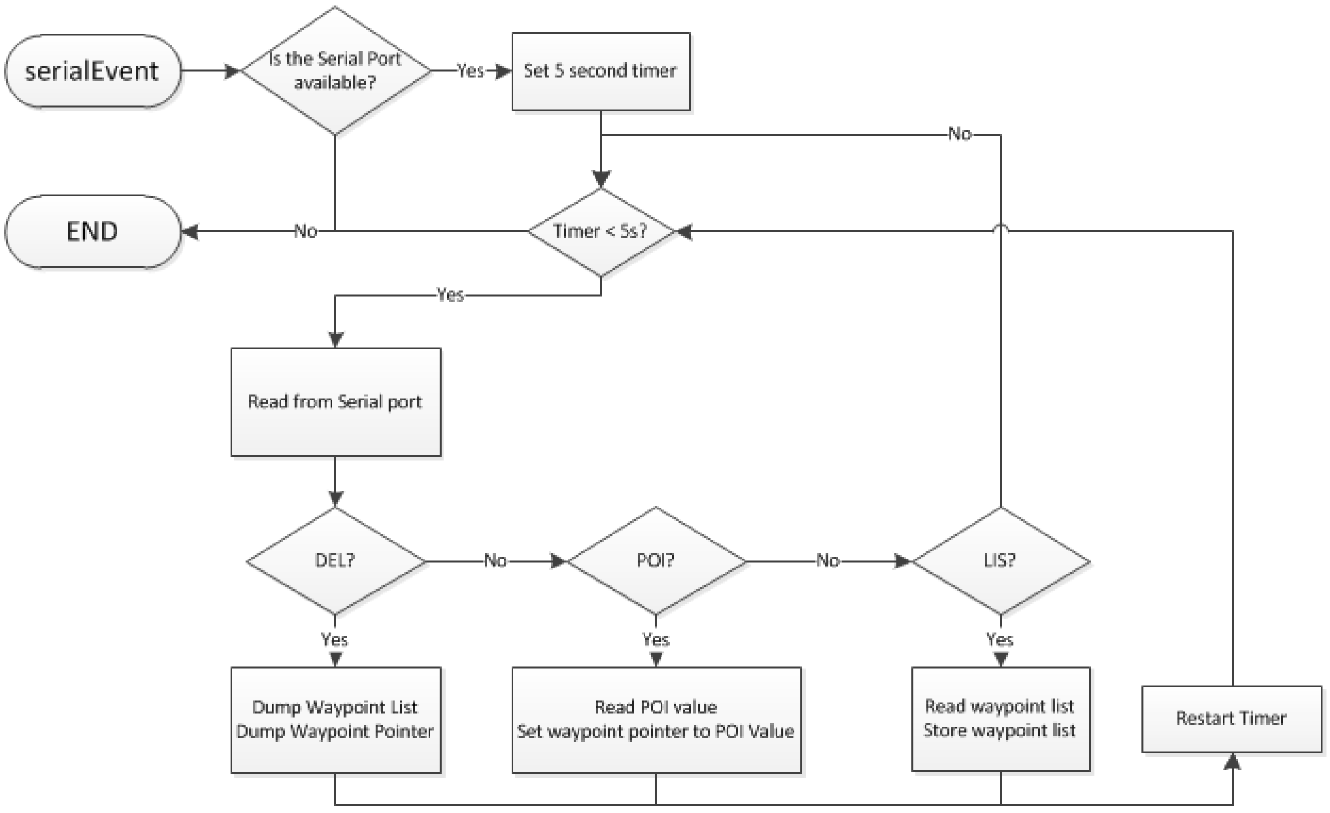


**Figure 5-2: Illustration of Serial Communications Workflow**

### 5.3 GPS Processing

The GPS hardware transmits data at a specific frequency and baud rate and it has been set to 1 hz and 9600, respectively. The process_gps function listens to the communications port hooked into by the transmit pin of the GPS and takes in data for 1 second. The function will then attempt to decode the data sent from the GPS into longitude, latitude, time, and satellite data which is then stored or is overwritten onto the GPS object. This object is then called to return the latitude and longitude data which is then stored separately by the embedded software for use in calculations. This workflow is illustrated in figure 5-3.

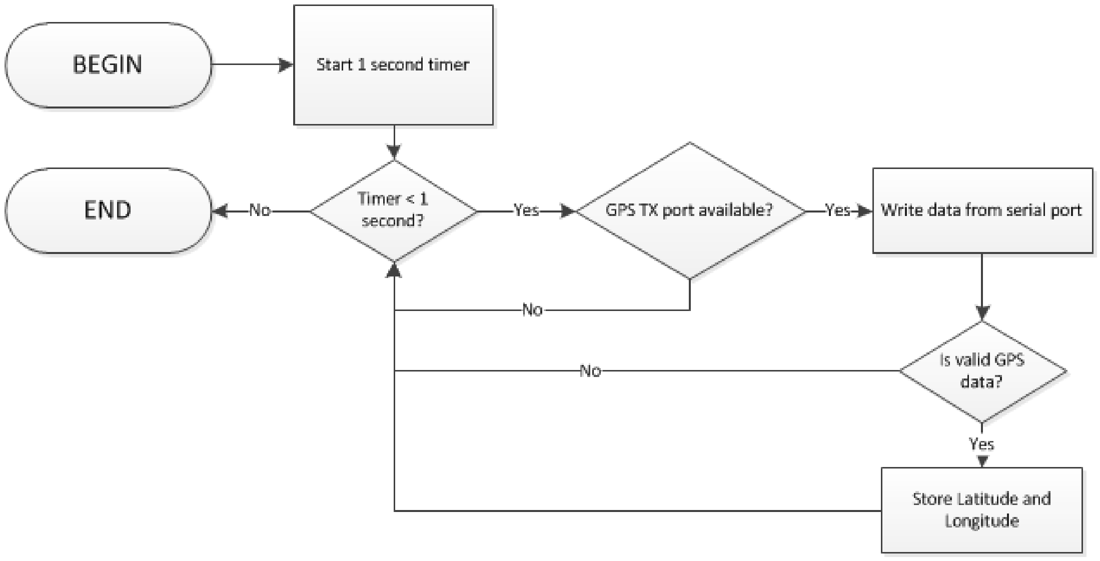


Figure 5-3: Illustration of GPS Processing Workflow

### 5.4 Heading Processing

The calculation of the forward azimuth (correct compass heading towards a destination) given the users current location towards the current waypoint destination is calculated in a function called calc_waypoint_heading. Taking in two sets of latitude and longitude coordinates that represent the current location and the current waypoint location, respectively, this function simply returns the forward azimuth assuming a spherical earth, a magnetic north located at polar north, and using a great circle arc. For small distances far from the magnetic poles, the discrepancy will be small enough to ignore. The formula to calculate this is shown in equation 5-1.

$$\theta = \left( \frac{\boldsymbol{atan2}\ (\boldsymbol{sin}(\boldsymbol{\Delta lon}).\boldsymbol{cos}(\boldsymbol{lat2}), \boldsymbol{cos}(\boldsymbol{lat1}).\boldsymbol{sin}(\boldsymbol{lat2})\ -\ \boldsymbol{sin}(\boldsymbol{lat1}).\boldsymbol{cos}(\boldsymbol{lat2}).\boldsymbol{cos}(\boldsymbol{\Delta lon})\ )\cdot \mathbf{180}}{\boldsymbol{\pi}} + \mathbf{360} \right) \%\mathbf{360}$$

Equation 5-1: Forward Azimuth Calculation

### 5.5 Path Detection Processing

**[R41-A]** The device shall provide sensory vibration feedback at most every second.

The path detection algorithm is contained within the path_detect function and it simply determines the difference between the magnetometer's returned heading and the forward azimuth calculated in the calc_waypoint_heading function. If the absolute difference is far too great of a margin, the first haptic feedback motor will not activate at all. If the absolute difference falls within a certain margin, then the duty cycle of the haptic feedback motor will be set at a value inversely proportional to the difference. The end result is that the duty cycle will be strongest when the calculated forward azimuth and current bearing match. A separate clock is used to pulse the motor at second intervals. The exact values are likely to be in flux as testing goes forward. The workflow of this function is shown in figure 5-4.

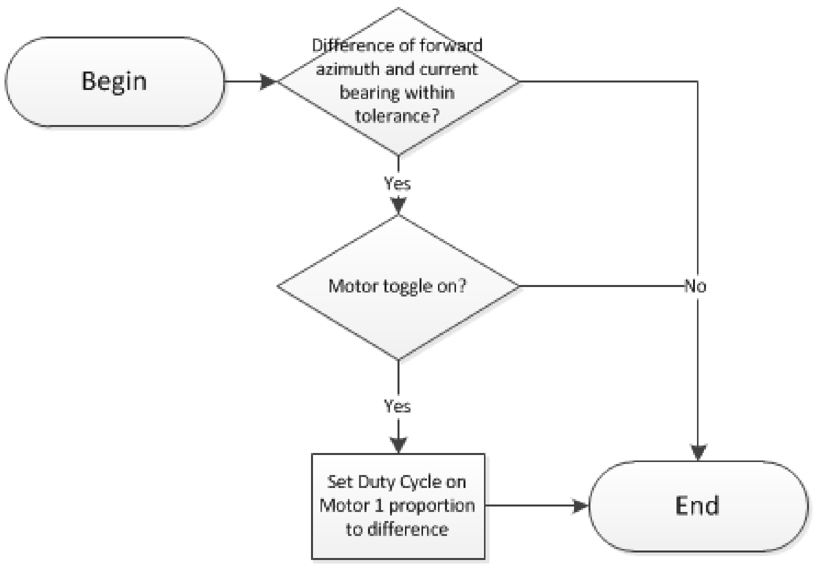

Begin
Difference of forward azimuth and current bearing within tolerance?
Yes
Motor toggle on?
No
Yes
Set Duty Cycle on Motor 1 proportion to difference
End

Figure 5-4: Illustration of Path Processing Workflow

### 5.6 Waypoint Processing

The processing of the current waypoint occurs in the function waypoint_detect. This function determines the distance between the device's current location and the current waypoint utilizing

the spherical law of cosines to calculate the great arc length distance between the two (again, assuming a spherical earth). The formula for this is shown in Equation 5-2.

$$d = acos(sin(lat1) * sin(lat2) + cos(lat1) * cos(lat2) * cos(lon2 - lon1)) * radius\ of\ Earth$$

**Equation 5-2: Waypoint-to-Current Location Distance**

If this distance falls within a testing approved number (potentially 5-20 meters), the pointer for the waypoint list will be incremented to indicate that the next waypoint is being used. This workflow is illustrated in figure 5-5.

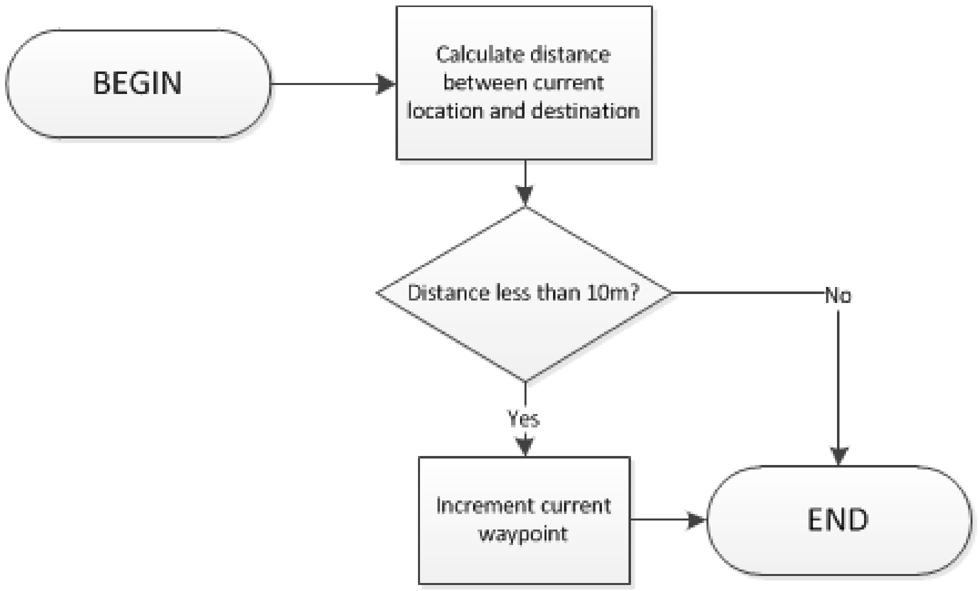

BEGIN
Calculate distance between current location and destination
Distance less than 10m?
No
Yes
Increment current waypoint
END

**Figure 5-5: Illustration of Waypoint Processing Workflow**

## *5.7 Ultrasonic Sensor Processing*

The ultrasonic sensor data process detects obstacles by sending out a signal to determine the voltage level output. The voltage level will determine the distance between the sensor and the obstacle using Equation 5-3.

$$\text{Distance per inch} = \frac{(\text{Vcc}/512)}{\text{inch}}$$

**Equation 5-3: The relationship of Pin Voltage to Distance**

Since the device is operating at 3.3V, the distance will yield ~ 6.44mV/inch. The analog reading of the ultrasonic sensor is very sensitive. In order to reduce the amount of error in a reading, an average of 50 - 60 readings will give a more consistent result. The sensors are always on to detect for obstacles in front and above but the feature can be turned off with a simple button press. The process flow diagram of the ultrasonic sensors is shown in Figure 5.6.

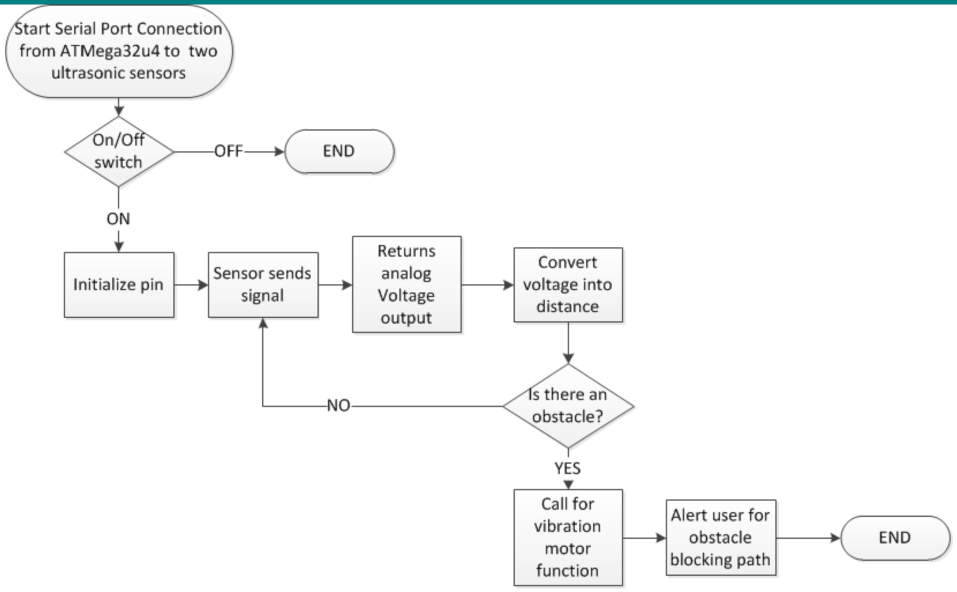

Start Serial Port Connection from ATMega32u4 to two ultrasonic sensors
On/Off switch
OFF
END
ON
Initialize pin
Sensor sends signal
Returns analog Voltage output
Convert voltage into distance
Is there an obstacle?
NO
YES
Call for vibration motor function
Alert user for obstacle blocking path
END

Figure 5-6: Illustration of Ultrasonic Sensor Processing Workflow

### *5.8 Interrupt Handling*

There are three true interrupt cases that the Arduino deals with: a switch that toggles on/off the motor related to the waypoint sensing workflow, a switch that toggles on/off the motor related to the ultrasonic sensor object detection, and a button that switches the route of the user (this function may not be implemented for proof of concept and is not shown). The workflow for these are shown in figure 5-7

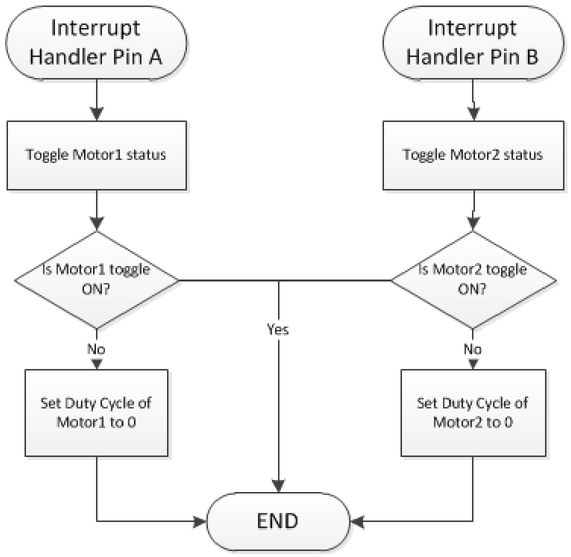


Figure 5-7: Illustration of Interrupt Handling Workflow

## 6. Software Application Design

### *6.1 Software Graphical User Interface (GUI)*

| | |
|---|---|
| **[R55-A]** | Software will be built on Microsoft .NET framework 4.5. |
| **[R60-A]** | The application shall be compatible with Windows OS. |

In the Software Interface section of the *Functional Specification for NaviCane: Navigation-Assisting White Cane*, there are several fundamental requirements for NaviCane's proof of concept model. In order to satisfy these requirements, the interface is generated from Microsoft Visual Studio 2010's C# Windows form features. Unfortunately, Microsoft Visual Studio 2010 only supports up to Microsoft .NET framework 4. In order to support .NET framework 4.5, newer editions of Microsoft Visual Studio will have to be used. Compatibility with Windows OS can be achieved as long as the user's computer supports Microsoft .Net framework 4. For the future models of NaviCane, we will implement the GUI through Microsoft Visual Studio 2012 in order to support Microsoft .NET framework 4.5.

The general design of the NaviCane's software GUI is not meant be attractive and flashy. The main purpose is to allow Text-to-Speech and Speech-to-Text software to conveniently navigate through the program with relative ease. Figure 6-1 illustrates the simplicity of the GUI.

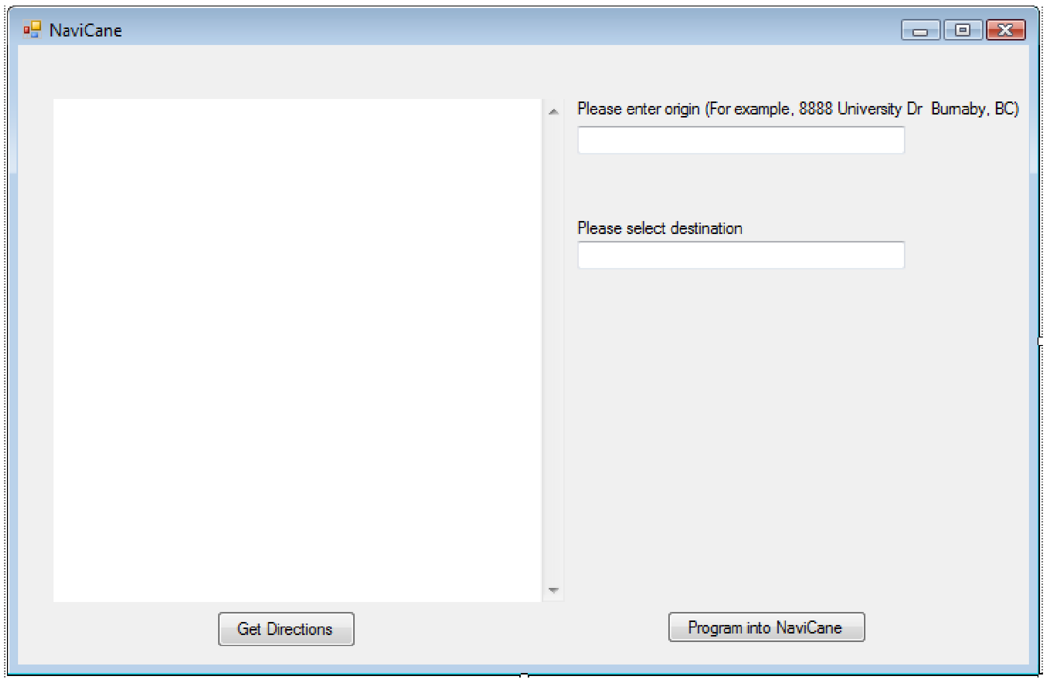
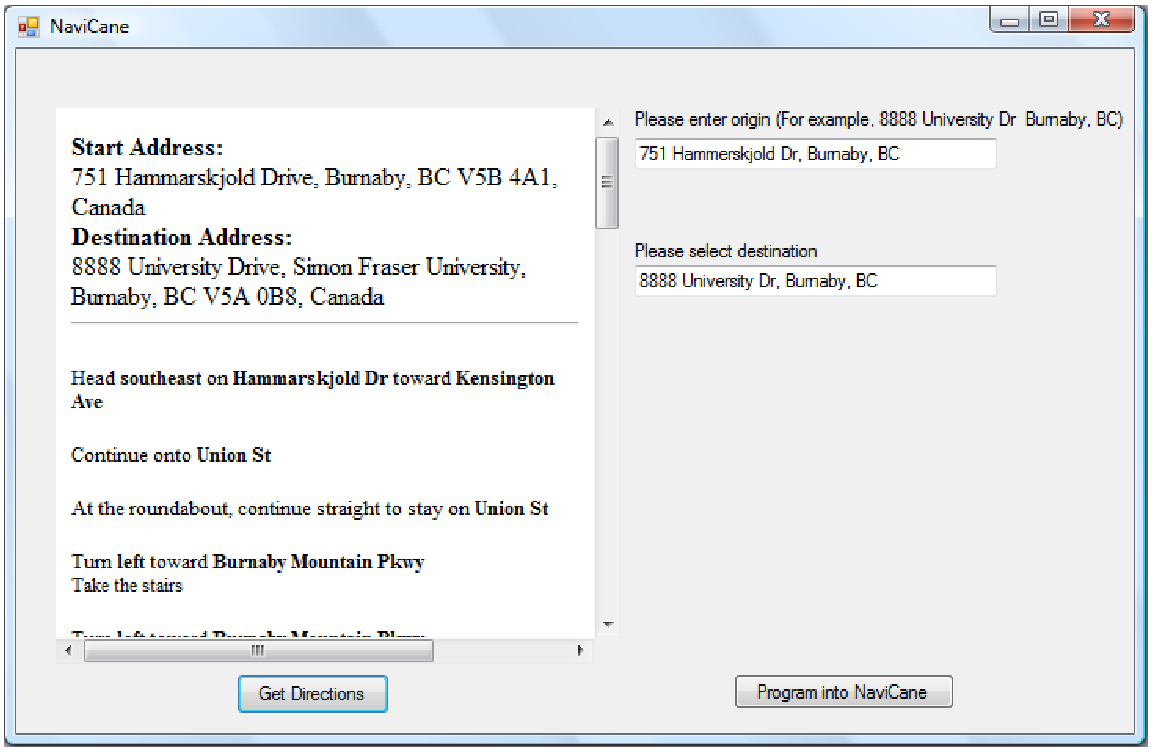

NaviCane
Please enter origin (For example, 8888 University Dr  Burnaby, BC)
Please select destination
Get Directions
Program into NaviCane

**Figure 6-1: NaviCane Software GUI**

After the user clicks on the “Get Directions” button with valid origin and destination addresses in their respective boxes, the GUI will display the information in the html browser box as shown in Figure 6-2.


NaviCane
Please enter origin (For example, 8888 University Dr  Burnaby, BC)
Please select destination
Get Directions
Program into NaviCane

751 Hammerskjold Dr, Burnaby, BC

8888 University Dr, Burnaby, BC

**Start Address:**
751 Hammarskjold Drive, Burnaby, BC V5B 4A1, Canada
**Destination Address:**
8888 University Drive, Simon Fraser University, Burnaby, BC V5A 0B8, Canada

Head **southeast** on **Hammarskjold Dr** toward **Kensington Ave**

Continue onto **Union St**

At the roundabout, continue straight to stay on **Union St**

Turn **left** toward **Burnaby Mountain Pkwy**
Take the stairs

Figure 6-2: NaviCane GUI with directions

After the directions information has been retrieved, the “Program into NaviCane” button will program the routes into NaviCane’s microcontroller. Firstly, the client will establish connection with the microcontroller through the serial communications port. Once connection has been established, it will follow our protocol to ensure that the proper datastream will be sent.

## *6.2 Software Algorithm*

**[R58-A]** The application shall finish data processing within 5 minutes.

The one of the major requirements for the software client’s algorithm is to provide a fast processing time. However, the processing time may differ depending on the user’s internet connection and computer processing power. Although there are concerns regarding the data processing time, a 2.0 GHz processor with 4 GB of RAM running under Windows Vista Home Premium Service Pack 2 and 54 Mbps was able to process the data relatively instantly, which was less than 2 seconds.

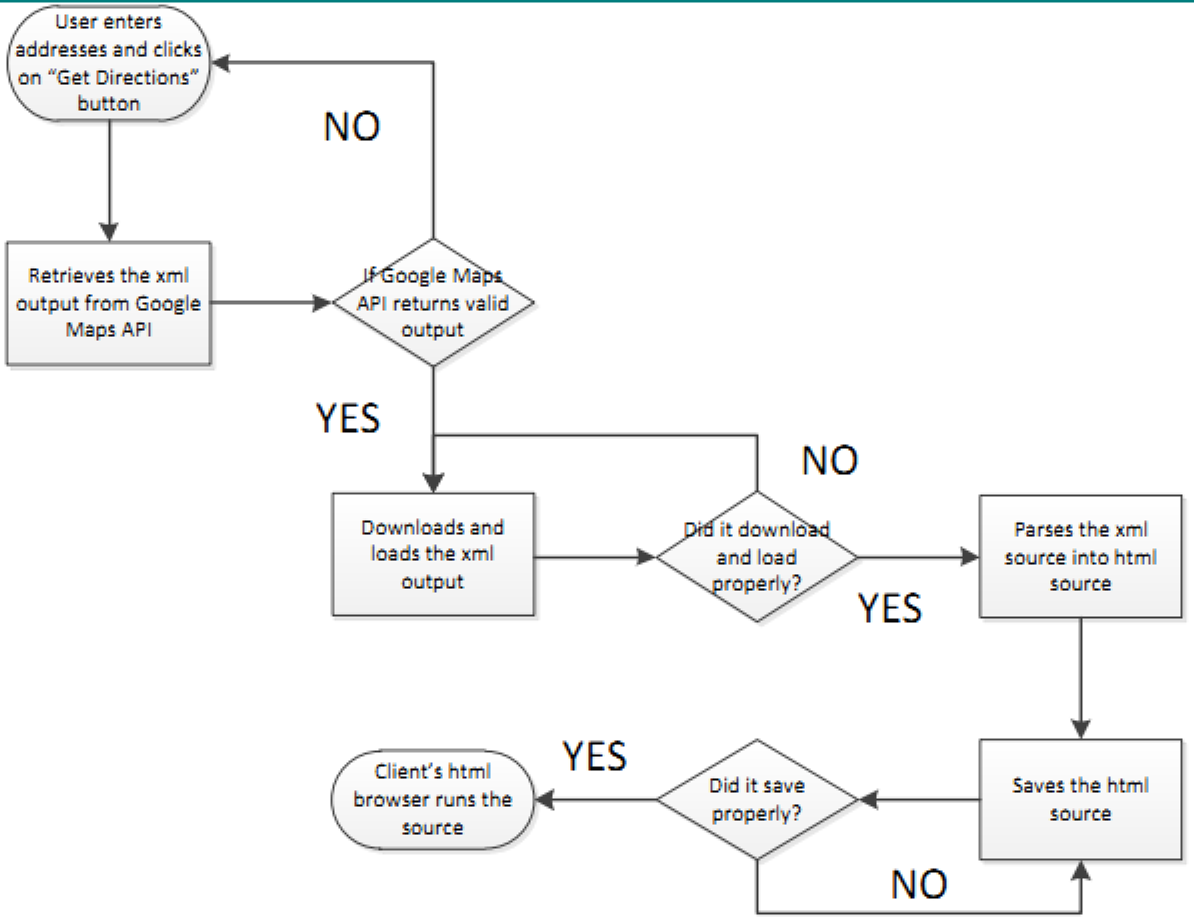

User enters addresses and clicks on "Get Directions" button
NO
Retrieves the xml output from Google Maps API
If Google Maps API returns valid output
YES
NO
Downloads and loads the xml output
Did it download and load properly?
YES
Parses the xml source into html source
YES
Client's html browser runs the source
Did it save properly?
Saves the html source
NO

**Figure 6-3: Software Client Algorithm Flowchart**

Figure 6-3 illustrates the software application's algorithm. The software will retrieve the xml output provided by Google Maps API after the user requests for the directions. If the xml output is valid, it will be temporarily downloaded onto the user's hard disk drive. Once downloaded, the application will load the xml file. It will undergo parsing and translation into an html source file. The html source will be temporarily saved on the user's hard disk drive. The client's html browser will read the html source and provide the directions that Google Maps API provided.

Additional information provided by Google Maps API such as time duration and distance to travel may be implemented in future models of NaviCane. In relation with the distance to be travelled, an odometer may work in conjunction with the GPS module to accurately track the user's location. However, Google Maps API's walking directions is still in beta mode. Because everyone walks at a different speed, the time duration will vary greatly depending on the user. Since this device is geared towards visually impaired users, their rate of travel is definitely slower compared to a non-visually impaired person.

## 7. System Test Plan

The system will be tested in four sections; the first section will test the client software, the second section will test the embedded software on the arduino device, the third section will test the hardware on the cane, and the fourth section will test the full expected operating procedure of the device. Each section will also be tested for its interaction with the other section; mainly, the communications between user interface and hardware device and the protocol utilized. Afterwards, the standard operating procedure of the device will be tested.

### *7.1 Software Application Test Plan*

To ensure that each component of the user interface maintains its functionality, unit tests will be implemented to target specific functions in the.

- The XML parser will be tested against various types of proper and improper XML messages.
- The user input fields will be tested against various proper and improper user inputs
- Google maps API querying will be tested for basic mapping queries
- The device communications protocol that has been established to communicate with the hardware device will be tested to verify that the proper syntax and symbols are sent to the proper USB port.
- All necessary text will be verified to be readable by screen-reading and text-to-speech technology
- The accuracy of the GPS coordinates will tested against the GPS module's coordinates
- Test the application's data processing speed against computers with varying processing power and internet connection

### *7.2 Embedded Software Test Suite*

To ensure that each basic function of the embedded software on the device maintains its functionality, a suite of unit tests are being implemented which target specific functions in our program.

- The communications port will have unit tests functions that verify each aspect of the communications protocol that has been established to communicate with the client. This includes waypoint lists, waypoint list size, and clear commands.
- The magnetometer and its accompanying functions have unit test functions tested to verify the calibration values that had been determined by re-running the calibration routine to find the maximum and minimum pin values.

- The path detection algorithm will have unit tests to verify its proper functionality and for its response to various proper and improper inputs.
- The waypoint detection algorithm will have unit tests to verify its proper functionality and for the response to various proper and improper inputs.
- The waypoint lists on the device will have unit tests to ensure that it tracks and switches waypoints properly upon reaching a new node location.

### *7.3 Hardware Tests*

These tests target specific hardware devices utilized in our design to ensure that they are each in working order including the Arduino itself. Each pin on the Arduino device will be activated in turn to verify that they work

The motors will be individually tested for activation at varying duty-cycles. They will be tested for activation, strength, and consistency within a degree of accuracy.

1. Utilize haptic motor test sketch
2. Turn on the device and charge to full capacity with hardware modules activated and connected to a PC
3. Note the changes in vibrational feedback in relation to the serial monitor data

The ultrasonic sensor will be tested to verify it reports the proper range and voltages.

1. Utilize ultrasonic sensor test sketch
2. Turn on the device and charge to full capacity with hardware modules activated and connected to a PC
3. Place an object at various distances to the ultrasonic sensor
4. Read the values returned to the serial monitor by the Arduino on the serial monitor
5. Note any major differences in values

The magnetometer will be tested to verify the calibration values that had been determined by re-running the calibration routine to find the maximum and minimum pin values. The magnetometer will also be tested for proper heading and accuracy against other compasses.

1. Utilize compass calibration sketch
2. Turn on the device and charge to full capacity with hardware modules activated and connected to a PC
3. Read the values returned to the serial monitor by the Arduino
4. Note any major differences in values
5. With current values, read compass bearing and compare with standard compass and electronic compasses

The GPS will be tested for activation, rate of message frequency, baud rate, positional accuracy, and message accuracy (checksum).

1. Turn on the device and charge to full capacity with hardware modules activated
2. Query GPS for status
3. Query for position
4. Utilize another GPS device and Google maps to verify the accuracy

The battery will be tested for basic functionality (the ability to power the device) and for idle device battery capacity.

Test 1:

1. Turn on the device and charge to full capacity with hardware modules deactivated
2. Leave idle
3. Note Time to Power Failure

Test 2:

1. Turn on the device and charge to full capacity with GPS, Compass, and Ultrasonic Sensors activated
2. Leave Idle
3. Note Time to Power Failure

### 7.4 Operating Procedure Test

This test simulates the simplest full workflow for our device. The tester will follow the steps listed below.

1. Utilize the software user interface on a PC to select a simple route from their current location within walking distance (1-5km)
2. Upload the waypoints to the hardware on the NaviCane
3. Disconnect the device from the PC with a full-charge
4. Follow the haptic-feedback guided waypoints to their destination
5. Specifically note the absence or presence of feedback based on approaching objects and obstacles that fall within the activation region for the ultrasonic sensor detection
6. Note the charge on the battery during (if the battery fails) and after the run through of the procedure

## 8. Conclusion

The design specifications listed in this document detail how the functional specifications of the NaviCane are to be fulfilled as well as having explained the implementation and current functionality of each subcomponent (hardware and software) of the device. The test plan will help to ensure that the design specifications are met and that product is in working order. The prototype is currently being developed and we expect to have a fully functional prototype by the first week of April, 2013.

## 9. References

[1] Maxbotix, “*LV-MaxSonar -EZ0 High Performance Sonar Range Finder*“ [Online]. Available: http://maxbotix.com/documents/MB1000_Datasheet.pdf

[2] Micrel, “*MIC5219 Voltage Regulator*“ [Online]. Available: http://www.micrel.com/_PDF/mic5219.pdf

[3] Microchip, “*MCP73831 Miniature Single Cell, Fully Integrated Li-Ion, Lithium Polymer Charger Management Controller*” [Online]. Available: http://ww1.microchip.com/downloads/en/DeviceDoc/21984a.pdf

[4] Muaz Salah, “*LV_EZ0_BW*” [Online]. Available: https://picasaweb.google.com/lh/photo/qR0TjEvvj99L2by5Nd7b5PSJXZrb1iNroGG0R-7ecD8

[5] Precision Microdrives, “*310-101 10mm Shaftless Vibration Motors*” [Online]. Available: http://www.sparkfun.com/datasheets/Robotics/310-101_datasheet.pdf

[6] SparkFun Electronics “*66 Channel LS20031 GPS 5Hz Receiver*” [Online]. Available: https://www.sparkfun.com/products/8975

[7] SparkFun Electronics, “*LiPo Charger Basic - Micro-USB*” [Online]. Available: https://www.sparkfun.com/products/10217

[8] SparkFun Electronics “*LSM303DLMTR Breakout Board - Tilt Compensated Compass*” [Online]. Available: https://www.sparkfun.com/products/10888

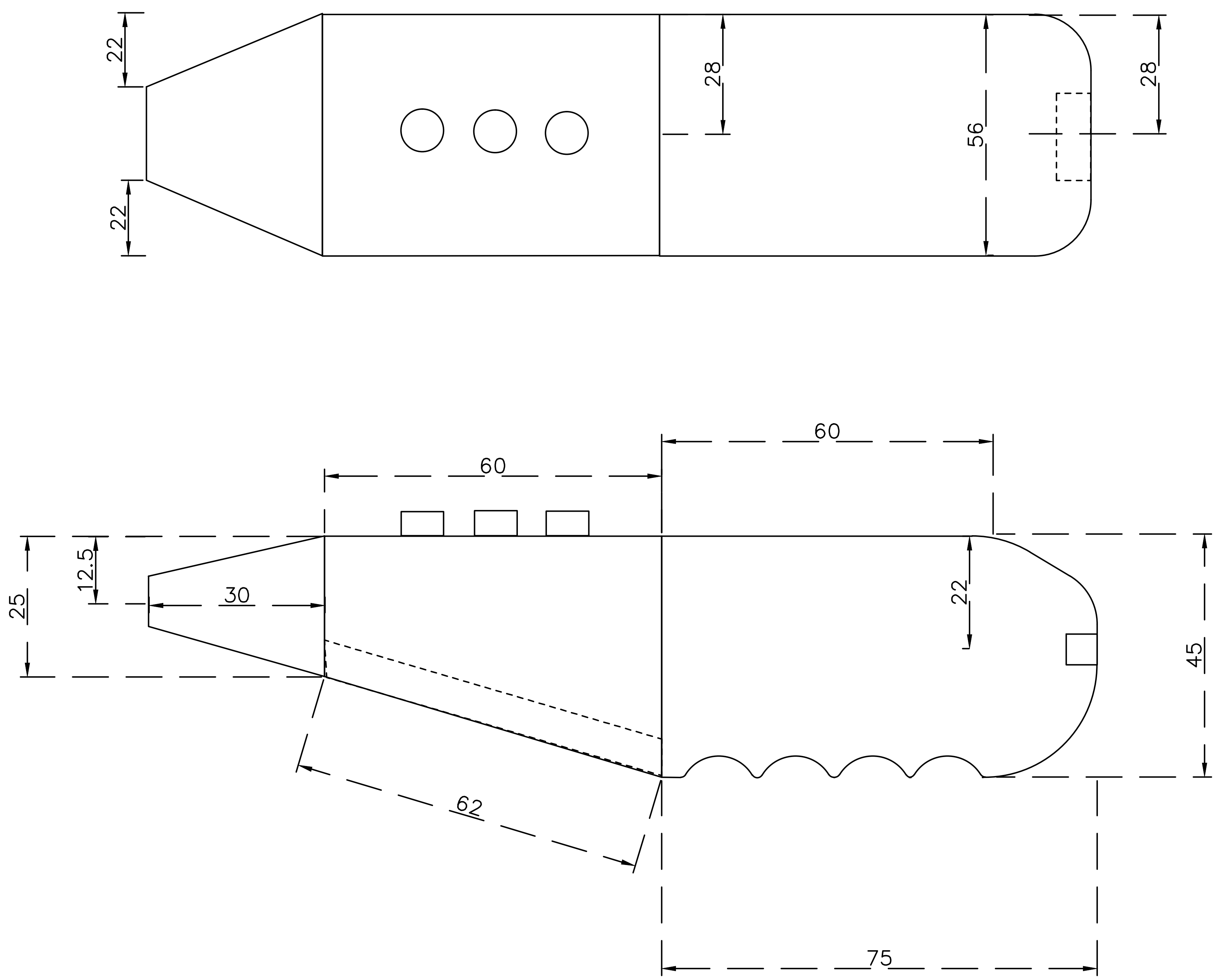
[9] SparkFun Electronics, “*Polymer Lithium Ion Battery - 2000mAh*“ [Online]. Available: https://www.sparkfun.com/products/8483

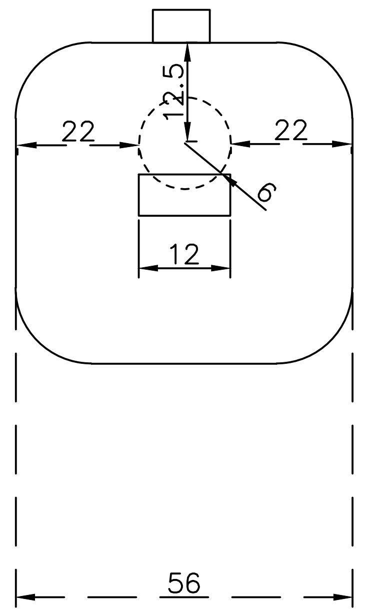
[10] SparkFun Electronics, “*Pro Micro - 3.3V/8Mhz*” [Online]. Available: https://www.sparkfun.com/products/10999

[11] SparkFun Electronics, “*Ultrasonic Range Finder - Maxbotix LV-EZ0*” [Online]. Available: https://www.sparkfun.com/products/8502

| RevNo | Revision note | Date | Signature | Checked |
|---|---|---|---|---|
| | | | | |

# APPENDIX A





| | | | |
|---|---|---|---|
| FILE NAME NaviCane Handle | FSCM NO | SHEET 1 | SCALE n/a |
| SIZE 1:1mm | | PROTOTYPE | |
| DRAWN 03/11/2013 | | NaviCane Handle | |
| CHECK | | | |
| APPR. | | | |
| ISSUED 03/14/2013 | | | |
| REV 1.0b | | DWG NO 1 – 1 | |
| CONTRACT NO rla41@sfu.ca | | | |