UNIVERSITY OF ALASKA ANCHORAGE

CSCE A470

CAPSTONE PROJECT

Audio Programming for Obstacle Detection System

Author:

Evan Taylor Riley

Supervisor:

Prof. James Molic, PhD

Anchorage AK, May 2016

Computer Science & Engineering Department
UNIVERSITY of ALASKA ANCHORAGE

evantriley@gmail.com

Abstract

The purpose of this capstone was to not only create a sensor system which was capable of detecting obstacles in a user's path and notify them, but to also keep the price of the device low enough that feasible anyone could afford it.

The motivation for this project was that while systems such as the one conceived already exist, they can cost upwards of hundreds of dollars. By keeping the cost of our system around or optimistically under one hundred dollars will mean that people who could otherwise not afford the more costly systems could still get the help they need if their vision is compromised enough to need aid.

The final design for the system has a single Arduino control board, three MaxBotix sensors, and an Adafruit waveshield. Two of the sensors will be worn on the users head detecting obstacles that the user may be looking at. The third sensor will be with the Arduino board in the control unit worn on the waist. It will be aimed downwards at an angle to check for drop offs and ledges. The waveshield will allow for auditory warnings to be sent to the user when an obstacle is detected.

Acknowledgements

The biggest thanks I need to give is to my father. He fostered my early love of computers and I learned how to use them by watching him. I probably would never have gotten into computer science without him. I also have to give a big thanks to the rest of my family, for supporting me throughout my schooling.

Table of Contents

1 Introduction 1

1.1 Introduction 1

1.2 Application 3

1.3 Motivation 4

1.4 Recent Development 5

2 System Integration and Modelling / Methodology 6

2.1 Introduction 6

2.2 Testing 6

2.3 System Design/Hardware & Software 7

2.4 Agile Methodology 9

2.5 Gantt Chart 9

3 Design and Testing / User Interface 11

3.1 Introduction 11

3.2 Design Process 11

3.3 Testing Methodology 12

3.4 User Interface 13

3.5 Agile Methods Employed in Production 15

4 User Manual 16

4.1 Welcome 16

4.2 Components 17

4.3 Setup 17

4.4 Instructions 18

4.5 Troubleshooting 18

5 Conclusion 20

Thoughts on Development 20

Final Product and Further Development 20

A Appendix A 22

B **Appendix B** **24**

R **References** **31**

Table of Figures

1

1.1 Snellen eye chart. 2

1.2 Object detection using sensor systems. 3

1.3 Obstacle detection systems in automobiles. 3

1.4 Chart explaining diabetic eye disease.

4

1.5 Similar obstacle detection system. 5

2

2.1 Maxbotix ultrasonic sensor. 7

2.2 Arduino control board. 8

2.3 Adafruit Wave shield for audio output. 8

2.4 Example of how an agile design process is conducted. 9

2.5 Gantt chart showing possible design timeline. 10

3

3.1 The Gorsuch Commons on UAA campus. 13

3.2 Adafruit Wave shield with attached speaker. 14

3.3 Vibration motor possibly added to system. 14

3.4 Picture of what the system looked like during development. 15

4

4.1 Drawing of mythological Greek figure Argus. 16

4.2 Example of low obstacle. 18

4.3 Example of high obstacle 18

4.4 Example of wall obstruction. 18

4.5 Example of drop-off. 19

5

5.1 The completed assembled system. 21

Chapter 1

Introduction

1.1 Introduction

The five senses of sight, smell, touch, hearing, and taste are all vital parts of our everyday lives. From smelling a piece of food and realizing it's rotten, to feeling the touch of a loved on your arm, these senses not only help us perceive the world around us and understand it, but warn us to imminent or lurking dangers. But what happens when one of these senses begins to wane or fails completely? What happens when you can no longer hear an approaching threat such as a car, or you could not smell the odor associated with a methane leak? Over the centuries inventors have tried to solve these problems by creating devices that boost failing or replace lost senses. An example of this might be the simple ear trumpets of the 17th century which made from a variety of materials, amplified noises so a deafened user might better hear them.

But what happens when a person starts losing what many consider to be one of their most important senses, their eye sight. Without reliable eyesight not only could a person no longer be capable of enjoying their favorite activities such as reading, but it also makes them much more susceptible to falling [1] or injuring themselves in other ways. In the past, those with compromised vision have compensated by wearing glasses if the problem is easily corrected, or in more severe cases, using canes to probe for obstacles ahead, or having trained canines lead them. With the use of modern technology though, new options have become possible.

This project will explore the possibility of creating a wearable personal system which could possibly replace these other options. Using a series of connected sensors, worn on the users head and chest, sonar waves will be sent out, and if they detect an obstacle in the user's path, an audio warning could be given to the user so they are made aware of the obstruction. Thus, a user would no longer need to carry a cumbersome walking stick, or need the aid of a trained animal or professional to do simple navigation. The system will be designed with the intention of being used by individuals with low vision, which encompasses those who have what is defined as having moderate to severe visual impairment. This means that the system would be used by individuals who have eye sight in the range of 20/70 to 20/400, meaning they see at seventy or four hundred feet what a healthy person with normal vision would see at twenty. (fig 1.1) [2].

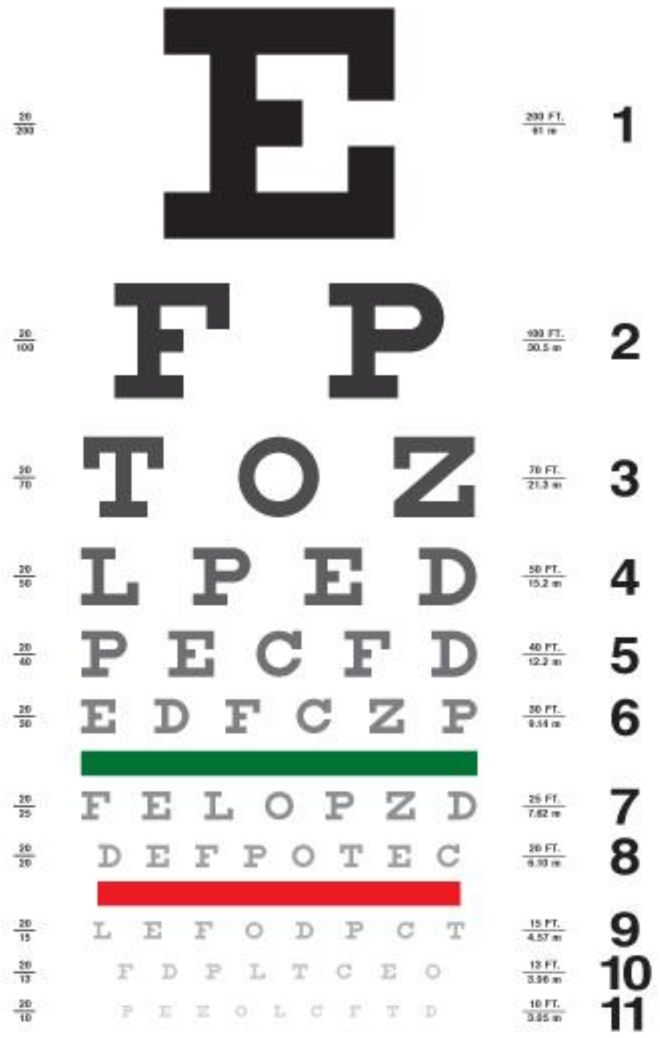


Fig 1.1: A standard English Snellen eye chart used to gauge an individual's eyesight. An individual with moderate to severe visual impairment would only be able to see lines one, two and three.

1.2 Application

Object detection systems have many modern applications. An advanced use for object detection systems, is object recognition. These systems not only check for objects within their sensor range, but then use a variety of different methods to try and identify what the the objects identity or properties (Fig 1.2) [3]. The most common use of sensor systems for public use, can be found in the automobile industry. For example, some cars created by the Nissan Motor Corporation are equipped with multiple sensor systems (fig 1.3) [4]. There sensors in the front and back of the vehicle which check for stationary or moving obstacles that will or are within the cars path as they pull into or out of a parking space.

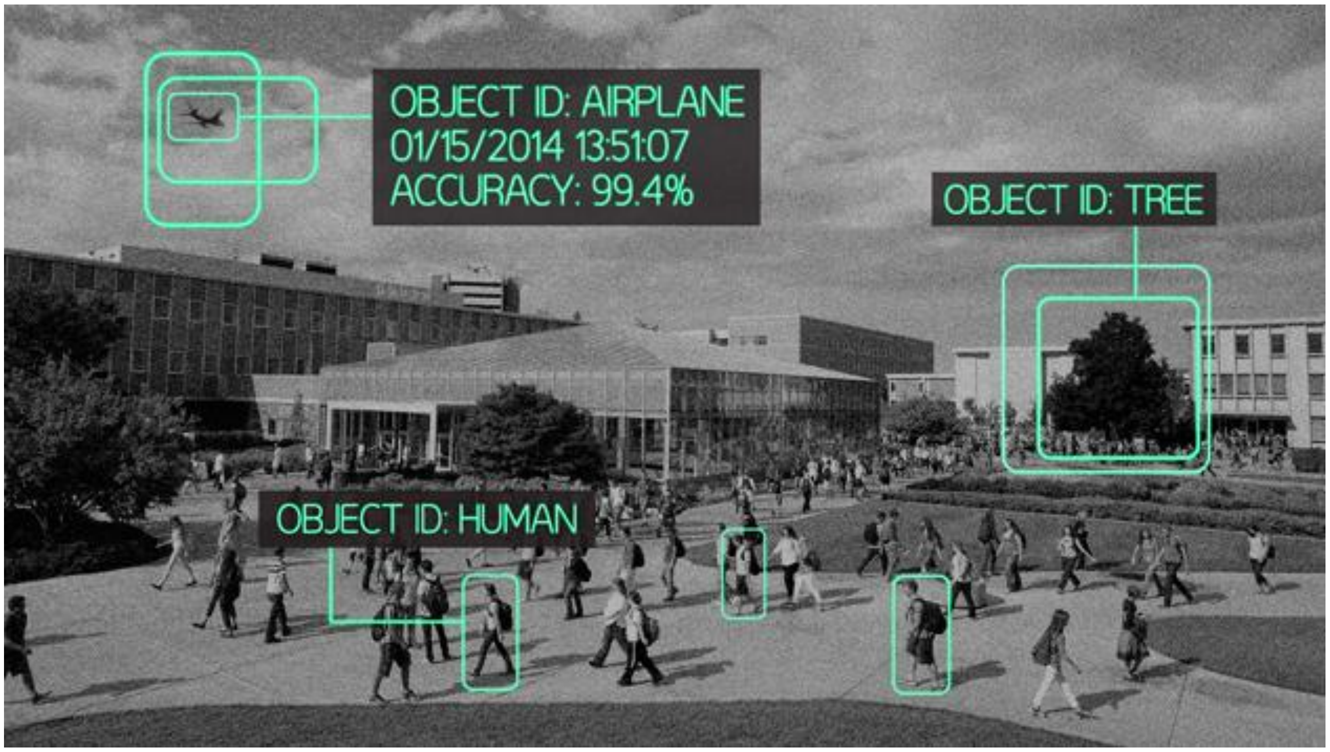


Fig 1.2: An example of how a sensor system utilizing object detection and identification might be able to pick out and define objects.

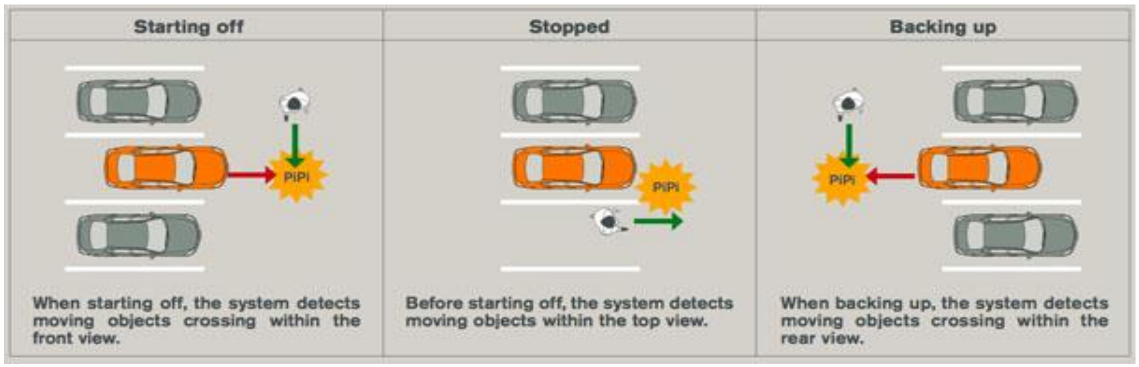

Starting off
Stopped
Backing up
PiPi
PiPi
PiPi
When starting off, the system detects moving objects crossing within the front view.
Before starting off, the system detects moving objects within the top view.
When backing up, the system detects moving objects crossing within the rear view.

Fig 1.3: Several example cases of how a car equipped with object detection systems can warn the driver of obstacles.

1.3 Motivation

The motivation for this project is to hopefully increase the quality of life of visually impaired individuals. In 2014, the World Health Organization estimated that 285 million individuals have some form of visual impairment, with 39 million being legally blind, and the other 246 million having what is defined as low vision [5]. Our system is not being designed for the blind, so while it will not benefit all those individuals with sight impairments, it could still be useful to the vast majority.

There is also a personal motivation too this project for myself. I was diagnosed with diabetes as a child, and because one of the side effects of poorly controlled blood sugar levels, is diabetic eye diseases (fig 1.4) [6]. This disease is actually several eye conditions that people with diabetes are susceptible too including diabetic retinopathy, diabetic macular edema (DME), cataract, and glaucoma [7]. The World Health Organization estimated that 9% of adults eighteen years or older had diabetes in 2014. This means that hundreds of millions of people are at risk of losing their vision due to complications, this system could be of use to them.

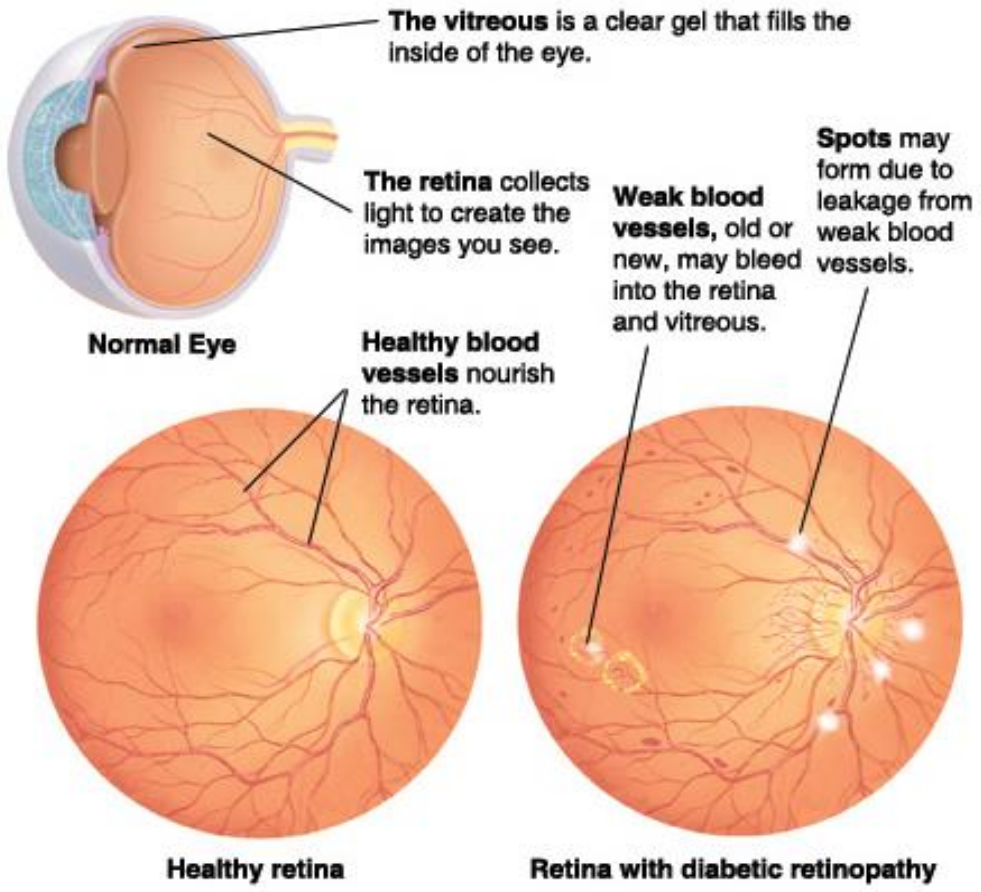

The vitreous is a clear gel that fills the inside of the eye.
The retina collects light to create the images you see.
Normal Eye
Healthy blood vessels nourish the retina.
Weak blood vessels, old or new, may bleed into the retina and vitreous.
Spots may form due to leakage from weak blood vessels.
Healthy retina
Retina with diabetic retinopathy

Fig 1.4: Because it can go unnoticed by the patient, diabetics are advised to get yearly eye exams to check for damage caused by uncontrolled blood sugar levels.

1.4 Recent Developments

Recently, there has been an increased interest in the idea of wearable technology (Examples include systems such as Google Glass and others). This trend has stated to be applied to obstacle detections systems in the past several years. One example of this from several years ago was a project from a team at the University of Alcalá in Madrid, Spain [8] (Fig 1.5). This project involved a central chest unit which tracked both the ground plane in front of the user as well as detecting obstacles on that plane. Feedback to the user was done using audio cues. This system allows for detection of obstacles that are in front of the users path, but not high obstacles such as overhangs.

Another recent development project was done by a team in Switzerland [9]. This system involved having sensors placed on the users chest. However, because of how the sensors were placed, the system could detect high and low obstacles at a distance, but not ones that are near the user. It would use either audio or tactile stimuli to warn the user about obstacles in their path.

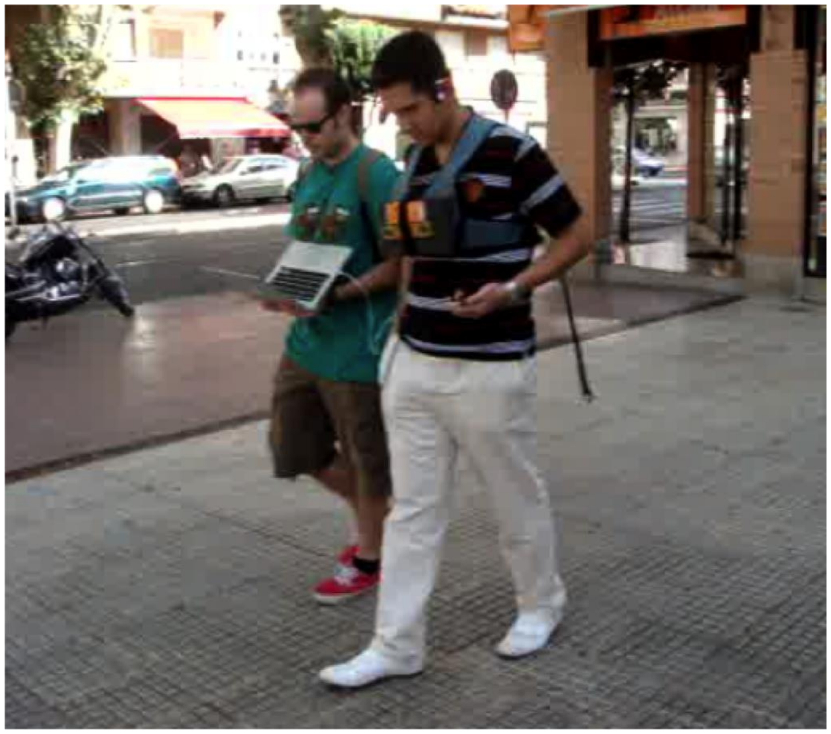


Fig 1.5: Users testing the object detection system created by the team at University of Alcalá in Madrid, Spain

Chapter 2

System Integration and Modelling / Methodology

2.1 Introduction

Using agile methodology, our team will create a system that utilizing several ultrasonic sensors placed at key locations on the wearers body, will be able to alert the user to obstructions in the users path or unknown dangers, such as stairs or drop offs.

The system will be created using an Arduino Uno as the main controller, with an Adafruit wave shield to allow for auditory warning cues to the user. The ultrasonic sensors used will be LV-MasSonar units, either EZ1 or EZ0 models. The programming for the Arduino will be done using their company supported Arduino IDE.

2.2 Testing

Testing for this unit will be done by creating test courses of obstacles to navigate. These courses could include obstructions such as narrow spaces simulating door frames, steps for stairs and drop-offs, and tables for overhangs. After a prototype of the system is completed, and a test course is constructed, volunteers or members of the development team will navigate the course while using the system. It will be optimal if the test user of the system is moderately to severely visually impaired, but if such individuals cannot be found, a method will be found to simulate the impairments such as drunk goggles or blindfolds.

2.3 System Design/Hardware & Software

The system can be divided into two main section. The sensor array of three ultrasonic sensors, and the main control unit which will house the Arduino controller, sound board, and a single sensor. The array will be attached to some form of platform which may be attached to a hat or other head ware. The control unit will be worn at the user's waist attached to a belt loop or waistline.

The ultrasonic sensors used for both the head and waist components will be of the LV-MaxSonar®-EZ™ Series brand (fig 2.1) [9]. These sensors work by emitting an ultrasonic burst, which will bounce off of an object, and back to the sensor. The sensor then takes into account how long it took for the soundwave to return and transmits an electrical pulse back to the control unit. This pulse's strength is based on how long it took for the wave's return and is interpreted by the control unit. The sensors used in the head apparatus will be the EZ1 model. The EZ1 has the second widest beam of the different models, can detect objects up to eight feet away, and offers a balance between picking up objects and rejecting false readings. The sensor bundled with the control unit will be an EZ4. This model of sensor has a much smaller sensor range, only being able to sense objects up to four feet away. This type of sensor is used mainly for finding big objects, or large changes in readings.

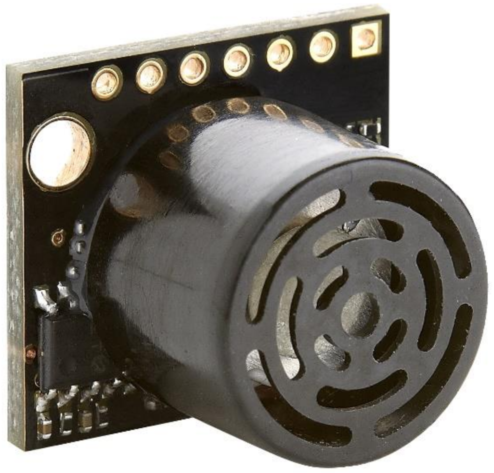


Fig 2.1: A Maxbotix ultrasonic sensor. Depicted is the LV-MaxSonar®-EZ0™ which will be used for the hip component of the system.

The main controller for the system will be an Arduino Uno board (fig 2.2) [10]. The Uno is a microcontroller board, which means that it has a microprocessor, I/O circuits, clock generator, RAM, and can store program memory. This means that after a program is written and loaded onto the board, it can be executed using the connected hardware without needing a desktop. The

Uno board operates with a voltage of five volts, had thirty two kilobytes of flash memory, and a clock speed of sixteen megahertz.

Fig 2.2: The Arduino Uno's small size is a great asset as it allows the control module to be small and unobtrusive.

To allow for audio cues to the user regarding warning or obstacles, an Adafruit wave shield will attached to the Arduino (fig 2.3) [11]. The Adafruit comes in a prepackaged kit which has all the chips and parts for construction. This shield, which is a term for a board which can be stacked on top of an Arduino, is capable of playing uncompressed audio files of any length. The files are read off an SD card which is plugged into the shield, and the volume of the output can be controlled using an attached volume knob. There is a jack where either headphones or a speaker may be plugged in, for our project headphones will be the suggested output method.

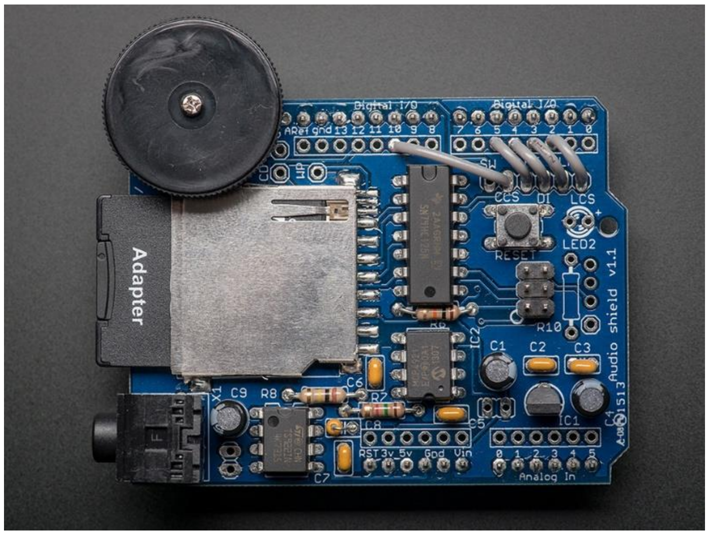


Fig 2.3: An overhead view of the Adafruit wave shield for audio output. Speakers or headphones may be plugged into the jack on the bottom left for user use.

For the writing of the program which will run on the Arduino, their original Arduino IDE will be used. This IDE is open-source and runs on Windows, Mac OS X, and Linux. The language it uses is C with a C++ wrapper, so it is object oriented and is not so different from Java or the other languages taught at UAA.

2.4 Agile Methodology

For this project, the development team will be following the agile software development process. Agile was first introduced in the early 1990's, but was not widely publicized until 2001 [12]. The method is centered on using lightweight development methods, which means completing small to moderate milestones in every iteration of the project or software (fig 2.4). This means that it is easier to show progress to a client, and makes it easier to make changes, add, or eliminate features from a project.

To compare the agile method to another common project development process, waterfall, and some major differences can be seen. In waterfall, clients are generally consulted at the start of the project, and after testing is completed. In agile, they are kept involved in the process, usually involving weekly meetings to show milestones and ask for input on what might be improved/changed. The majority of required features are defined at the start of development in waterfall projects, while in agile, major features are chosen at the start of the project, but because of the smaller milestones and involvement of the clients, they are open to change during development.

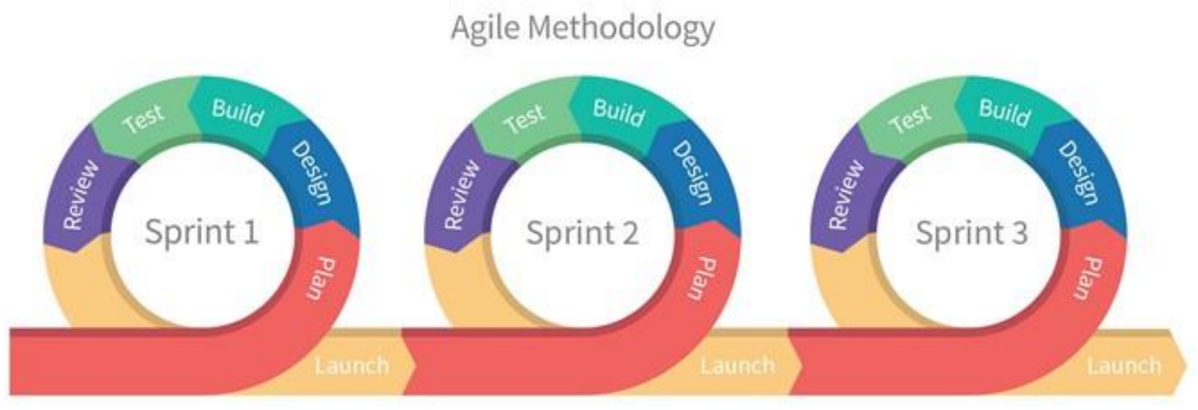
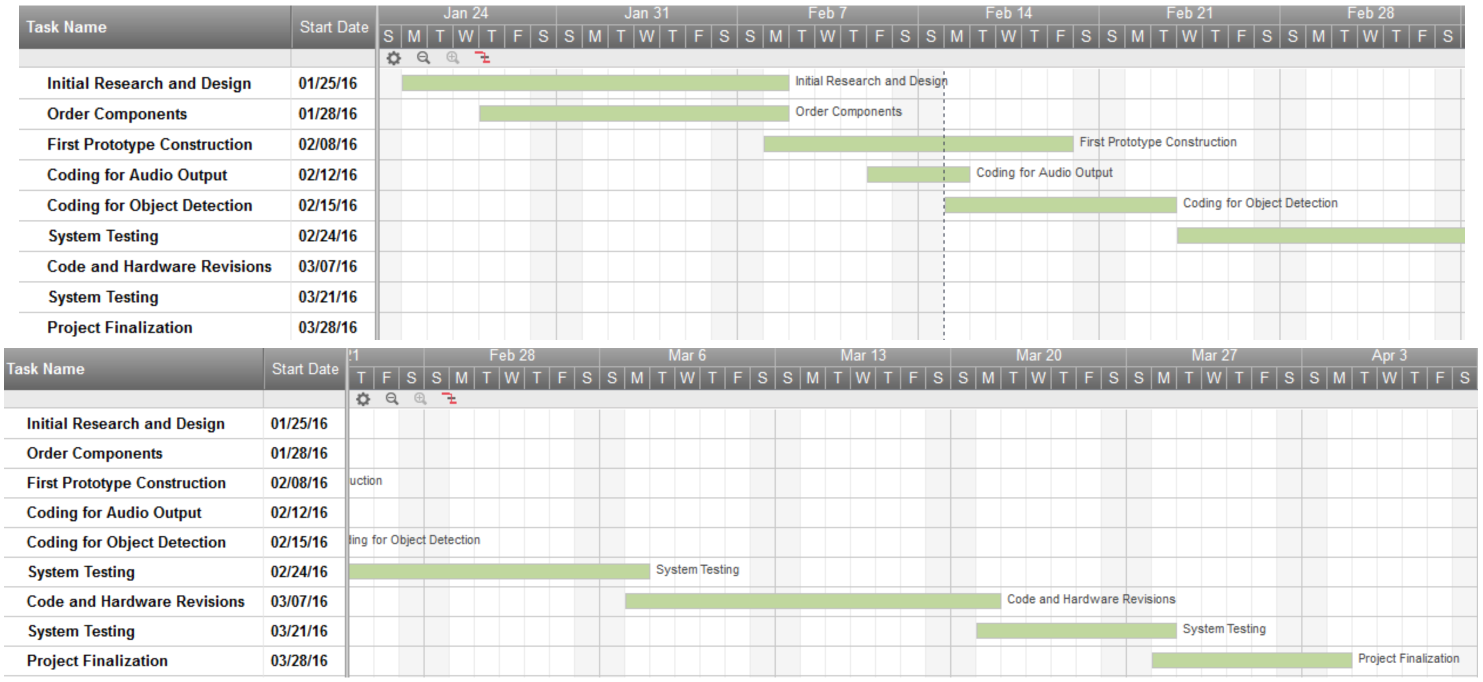


Fig 2.4: A graphical example of how Agile iterations are conducted.

2.5 Gantt Chart

A gantt chart is used by projects to show a theoretical breakdown of when different parts of the development are projected to be finished. Find below an example chart for the project (). To give some context to the chart, the second half of the project is intentionally vauge on what will be completed, instead simply saying "testing". This is because as we get further and further from the present it is hard to know exactly what may happen. Development of software or hardware may take longer or shorter than estimated, but we have given ourselves plenty of extra time with which to finish the project by April 1st, if not before (fig 2.5).



Task Name	Start Date
Initial Research and Design	01/25/16
Order Components	01/28/16
First Prototype Construction	02/08/16
Coding for Audio Output	02/12/16
Coding for Object Detection	02/15/16
System Testing	02/24/16
Code and Hardware Revisions	03/07/16
System Testing	03/21/16
Project Finalization	03/28/16

Fig 2.5: The Gantt chart of when and in what order our team plans to complete work on the project.

Chapter 3

Design and Testing / User Interface

3.1 Introduction

Because this project will be responsible for warning a visually impaired individual of possibly life threatening obstacles, including stairs or sudden drop offs, it is of paramount importance that the system go through rigorous testing and debugging to make sure that it can accurately warn a user of impending obstacles or dangers.

After the initial construction of the system is completed, we plan to carry out several cycles of testing, after each testing period, we will evaluate the results, checking to see if there are any deficits in the design. With the given time frame of approximate four months for development, we plan to carry out two of these cycles.

3.2 Design Process

After the initial idea for the project was decided, an obstacle detection system, we began to brainstorm different ways to implement the various pieces. The original design for the system involved having three sensors attached to some form of head apparatus, such as a band that could be worn over the forehead or attached to a hat. There would be a left, right, and central sensor, with the center one being aimed downwards to check for drop-offs.

Just from looking at drawings of this original design, we saw right away that there would be a definite issue with wires leading to/from the apparatus, we also realized the central sensor would be blocked by the bill of a hat or other headwear and it limited our mounting options for the three sensors. For this reason, we moved the central sensor to be placed on the user's chest. A sensor to be worn on the users hand in some kind of glove device was also proposed but ultimately rejected as it would increase the cost of the system with minimal added functionality.

After working on the project for a little over a month, and examining the amount of time left in the semester, we decided that the system required further refinement and scaling back a bit. We decided to cut the system to two sensors instead of three, and moved the chest sensor to be integrated into the main control unit which would be worn on the user's waist.

3.3 Testing Methodology

Testing the system was done in two different ways. First, controlled sensor tests in the lab, and second, practical tests of the system in a typical indoor environment. We felt that these two types of tests could accurately show us how well the system operated, and if there were any glitches or features that weren't working as intended.

The controlled sensor tests were kept as simple as possible. The main purpose of this type of testing was to check that each sensor was working properly before being integrated into the system, that the chaining of sensors would not cause undue interference, and to find an optimal distance to check for obstacles. For this test, we hooked up an Arduino board to a team member's laptop, and then two sensors to the board. The sensors then repeatedly fire, and their outputs were displayed on the screen. Checking these results to measurement taken by hand, we could verify if the sensor was correct or not.

Field tests were also conducted to check how the system performed when in use. Having a team member hold one sensor at their waist, and another at their eye-line, this simulated where they will be in the final product after mounting. The user then walked down a hallway, around a small seating area (Fig 3.1) [13], and up to a stairwell to make sure the system was accurately giving them warning about obstructions. It was also important that the system was not outputting any false obstruction warnings, or few enough that they did not adversely impact the user's experience.

Fig 3.1: Picture of the Gorsuch Commons at UAA, a good example of an indoor environment that a visually impaired individual would need aid navigating.

3.4 User Interface

The obstacle detection system we are designing will not have any graphical or visible user interface. There are three main reasons for this choice. First, adding a screen to the control unit would require another shield on the Arduino board, increasing the unit's size. Second, it would increase the units power consumption, meaning there is a greater chance for the system to possibly fail while being used. Third, the main goal of this project is to not only create a helpful obstacle detection system, but to also keep the price low enough that it could theoretically be available to a greater number of consumers.

Instead of a GUI we have elected to carry out our user interactions with audio and possibly tactile ques. Using an Adafruit Wave Shield (Fig 3.2) [14], we can output uncompressed audio files from an integrated SD card to a user's headphones or other connected audio device. This allows the user to receive warnings even if they are not currently looking at the control unit, or are in an area with high noise interference. Sample warnings that might be output to the user include "low obstruction detected", "high obstruction detected", or "wall obstruction detected".

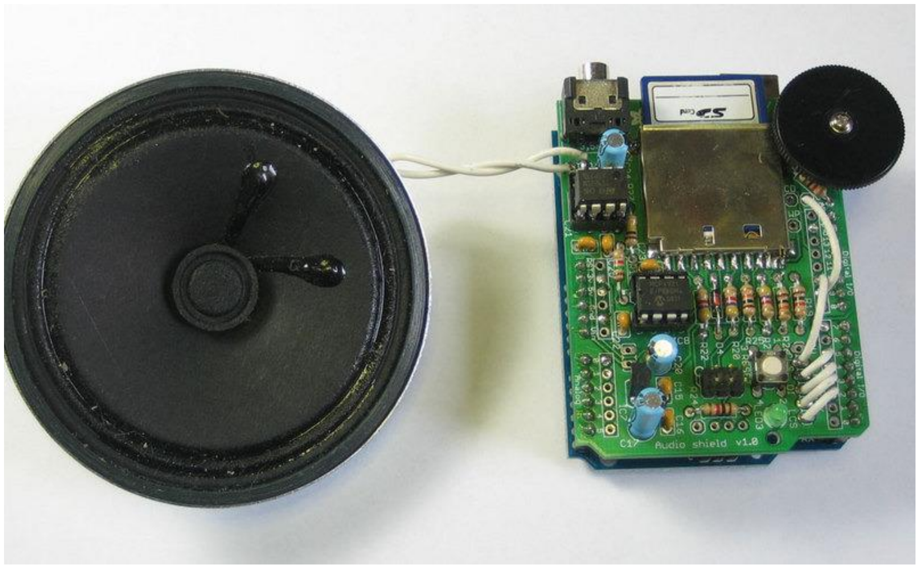


Fig 3.1: Example of the Adafruit Waveshield with attached speaker.

We are also exploring the possibility of adding some form of rumble functionality to the main control unit (Fig 3.3) [15]. Because this form of interface is more ambiguous, it would be more difficult to have a user accurately understand what type of warning they are receiving, but even if they did not have headphones on, they could still recognize that the system had detected some form of obstruction and respond accordingly.

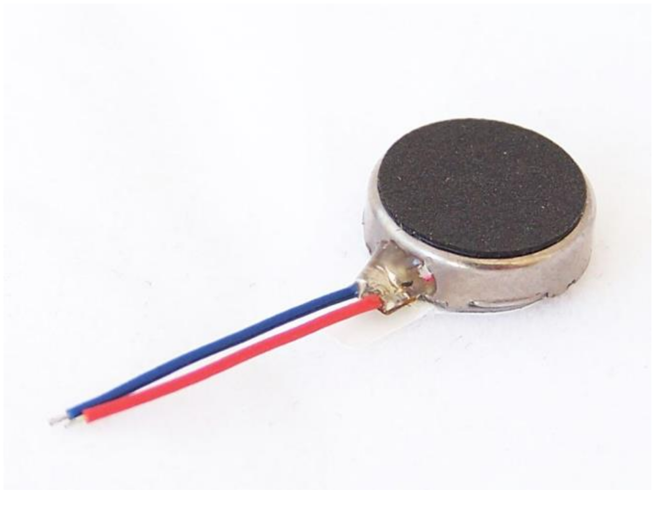


Fig 3.2: Example of a small flat vibration motor that may be integrated into the central unit for tactile feedback.

3.5 Agile Methods Employed in Production

The agile methods that our development team utilized during the systems construction was mostly in regards to trying to keep sprints short and to have either new code, or hardware to test each cycle. Meeting twice a week, the development team would reconstruct the system (Fig 3.4), testing each sensor to make sure it was still operating correctly, and then adding a new feature, such as an additional sensor, or testing new code, which might be fixing a previously found error or trying a new method for differentiating between obstacles.

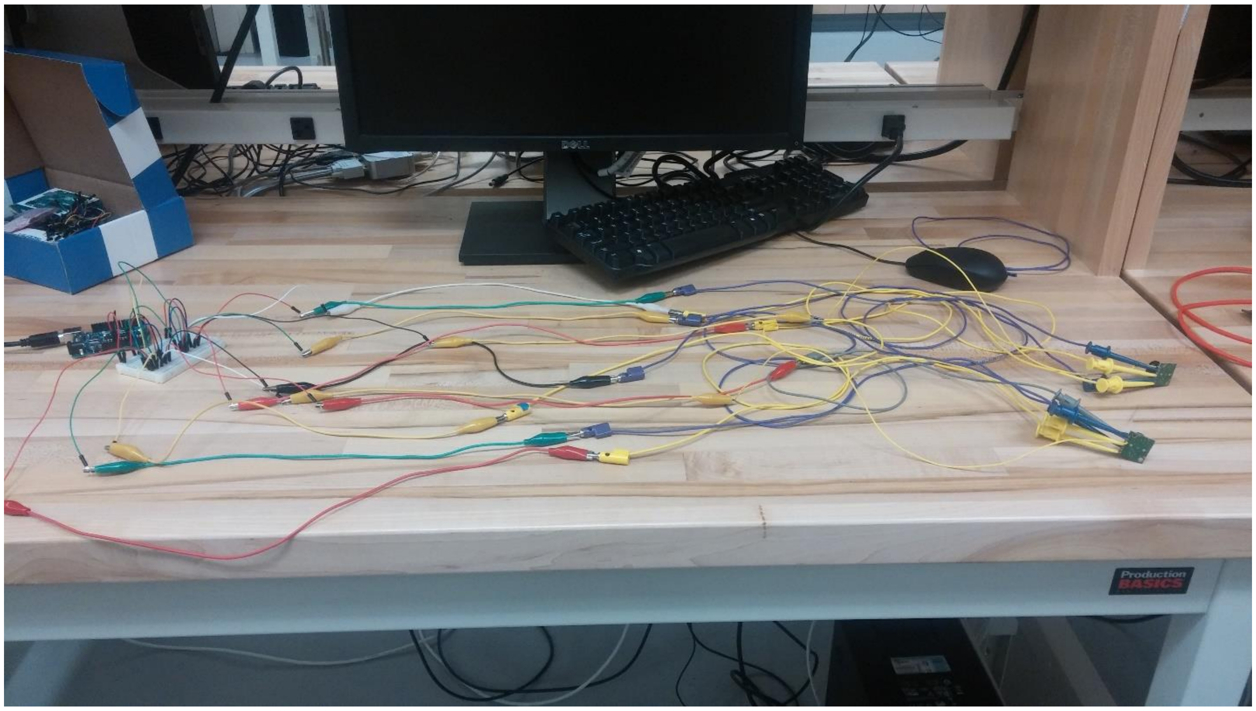


Fig 3.4: The system set up in the lab with two sensors attached. Code was then run testing that each sensor was accurately firing independently of one another.

Chapter 4

User Manual

4.1 Welcome

First off, we the designers of this device, the “Argus”, would like to thank you for choosing our system to help you with your navigation needs. Named after a 100 eyed-giant from ancient Greek mythology, he was described as “all-seeing”, and we hope that this system will live up to its namesake and your expectations.

Fig 4.1: An artists interpretation of what the mythical Argus may have looked like with his hundred eyes. [16]

4.2 Components

The “Argus“ is a rather simple system with only two major components but for a comprehensive understanding all sub components will also be listed:

1 Control Box (Waist unit)
   a. Arduino board (x2)
   b. Maxbotix sensor
   c. Adafruit Waveshield
2 Head Band
   a. Maxbotix sensors (x2)

4.3 Setup

Setup for the “Argus“ is very simple. Following these easy steps you should have your system up and running in a matter of minutes:

1. Open the back of the waist unit and put in 6 AA batterries.
2. Clip the waist unit to your belt or another spot where it can be tilted down to face the floor at an approxomitly 45 degree angle
   (The angle does not have to be exact, but the system will work better if it is close.)
3. Put on the attatched head unit with the two sensors facing forward.
4. Make sure that both the head and waist units are properly adjusted and are not being blocked by anything (shirt, hair, etc.).
5. Plug your choice of headphones into the audio jack on the front of control unit and put them on.
6. While facing towards an extended area of flat space (Ex. Open hallway) Plug the battery pack into the back of the control unit, and wait for approxommitly ten seconds for the system to finish setup.
7. **When you are finished:** Unplug the battery pack.

4.4 Instructions

With you now wearing the system, and the inital setup completed, you are free to move around your environment. When you come within range of an obstacle, the system will alert you with an audio cue regarding the distance and type of the obstacle. Find below images and descriptioins of the different warnings you may receive (Figs 4.2, 4.3, 4.4, 4.5).

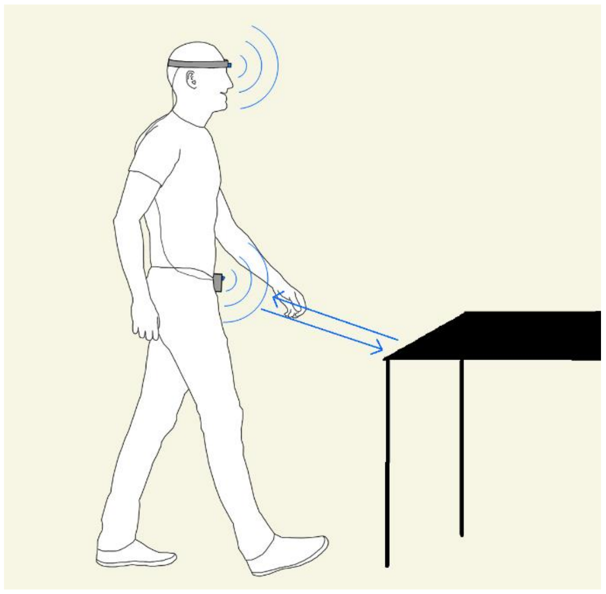


Fig 4.2: Example of low obstacle

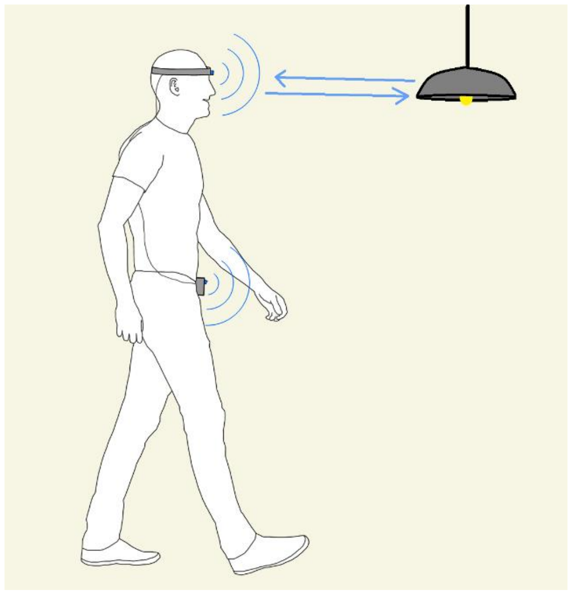


Fig 4.3: Example of high obstacle

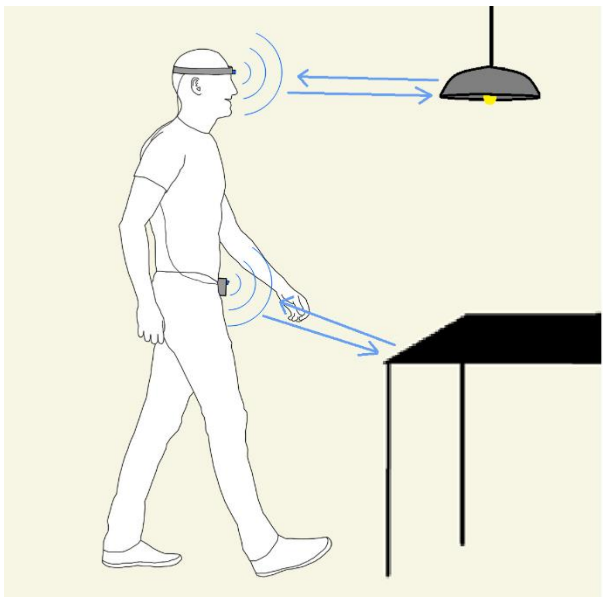


Fig 4.4: Example of wall or total obstruction.

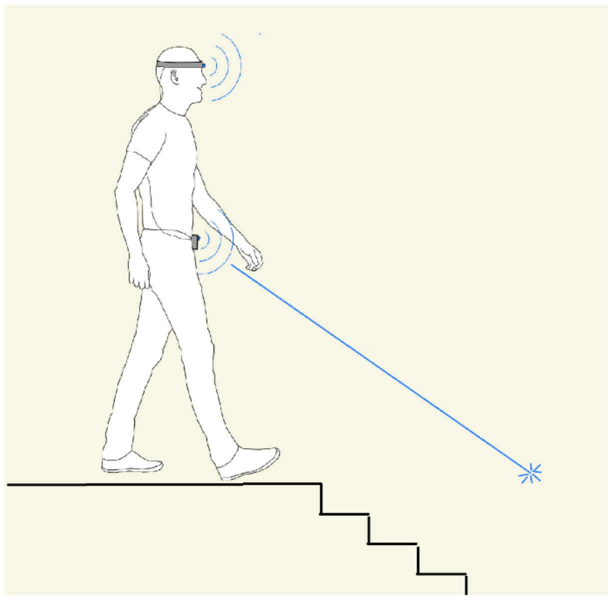


Fig 4.5: Example of drop-off.

4.5 Troubleshooting

Problems may arise while using the system. Hopefully the list of possible issues and solutions below will help you overcome any of these issues.

The system keeps telling me there is a drop-off in front of me when I know there is not.

This is most likely caused by an error during the system's startup. If the waist sensor is pointed downwards at too steep an angle during initialiation it sets the average ground distance to be shorter, and if the unit is then readjusted to a greater angle, it will give a false drop-off warning. To fix this, unplug the power, make sure the waist sensor is at an optimal angle, and then plug it back in.

The system keeps giving false readings, mostly walls.

A great number of false readings are due to the sensors being blocked by something, or an object or person quickly moving through the sensors range. The optimal use for the navigation system is a static environment with few moving obstacles.

Make sure the sensors are not being interefered with by your clothes or hair as well, as these will cause a false warning.

There is a static buzzing when audio cues are being given.

The audio components of the system are exceedingly fragile and it is possible that a chip or connection was damaged. For a replacement unit, please contact the development team.

Chapter 5

Conclusion

5.1 Thoughts On Development

As a software developer, this was my first experience working with a micro controller such as the Arduino. It was interesting to have to take into account how the wiring worked, and figuring out how to have the two units we ended up having to use send signals to each other. This was also only my second project working with a larger group, and it was a different experience working on one specific section of the project. Everyone involved in the project worked very hard and meetings were held twice weekly to iterate on either software or hardware.

5.2 Final Product and Potential Future Development

Looking at the final product we designed, I'm proud of how it (Fig 5.1), but there are definitely some issues I can see right away. For one, the waist unit is somewhat oddly sized, and could definitely use some streamlining. Also, the waist sensor is very temperamental, it's designed to be pointed downwards at exactly 45 degrees, but in practicality this is almost possible to do not only by the naked eye, but keep at this angle for an extended period of use.

The biggest change I would say could be made in development is the use of self-made proprietary parts. By using the pre-made Arduinos, wave shield, and sensors it was hard to fine tune them to exactly what we wanted. Another possible future feature is possibly adding additional sensors, such as a back sensor to check for approaching obstacles from the rear, or another sensor being added to the other side of the users waist so a left/right distinction can be made for low obstacles and drop-offs.

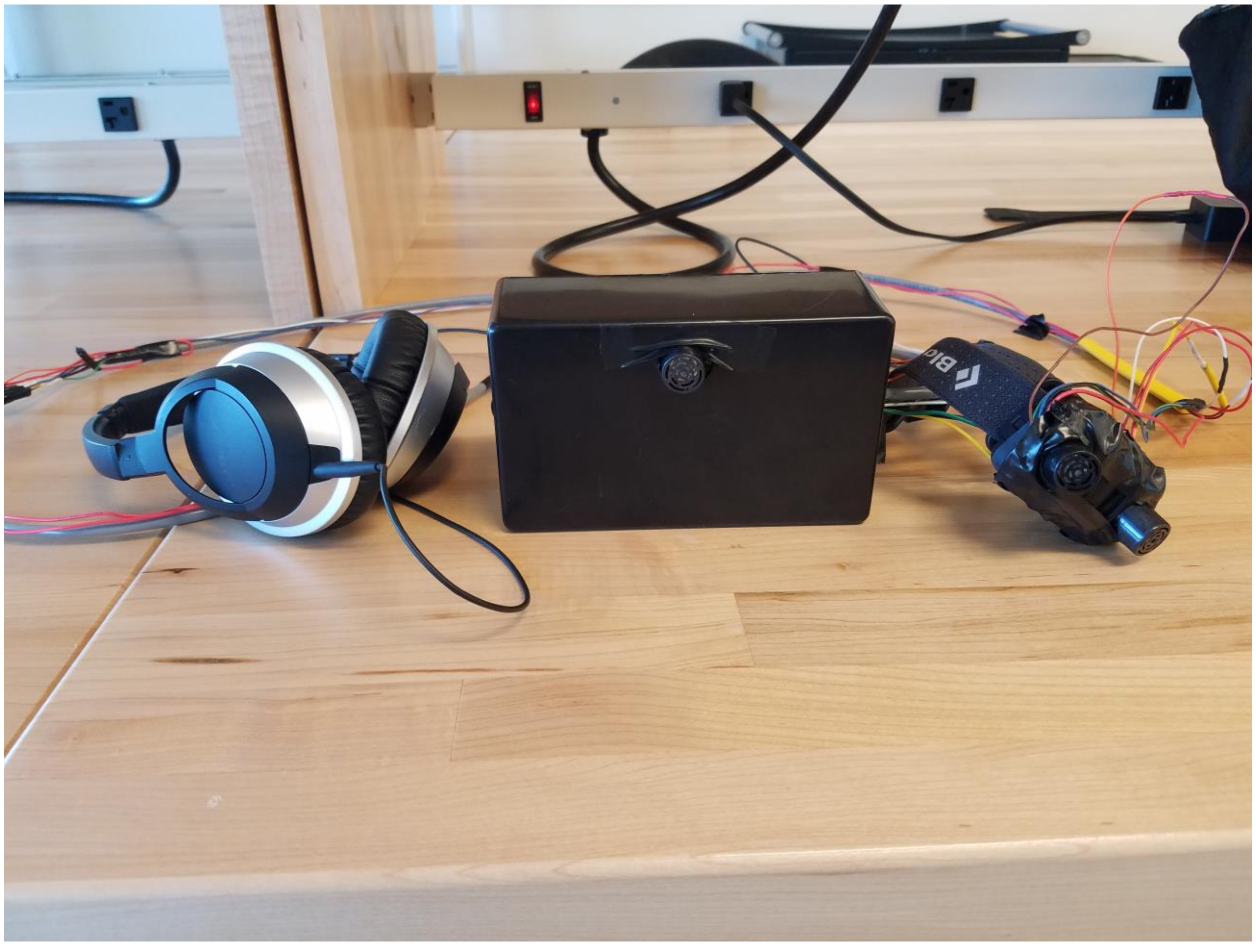


Fig 5.1: The completed obstacle detection system which will be shown to the public.

Appendix A

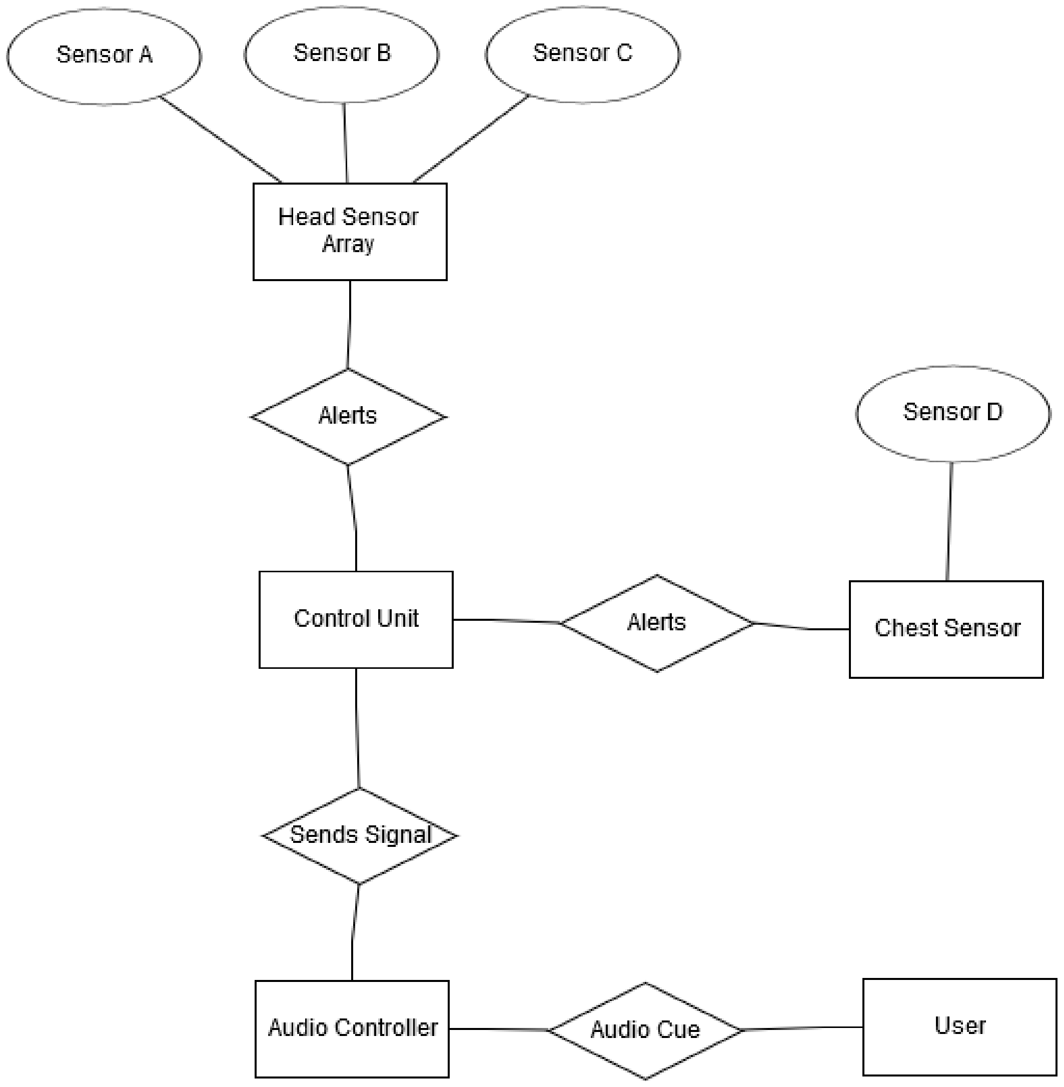


Fig A.1: Initial ER diagram of obstacle detection system.

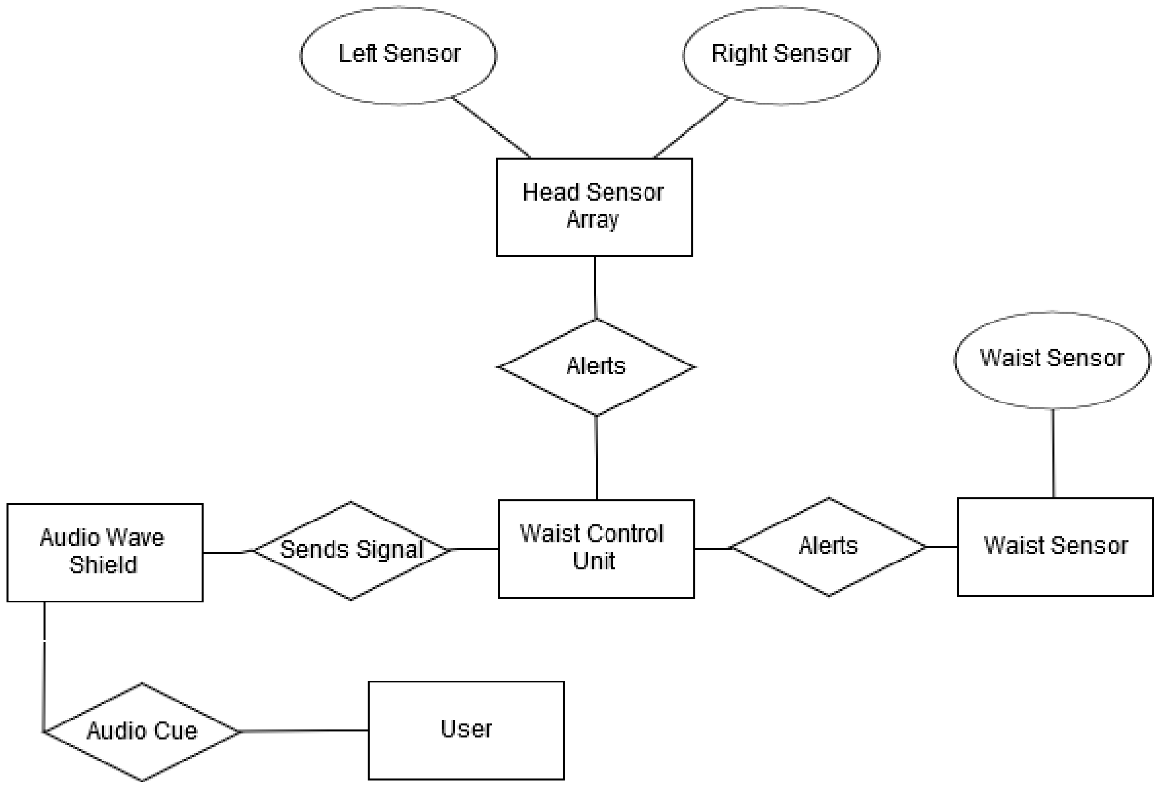


Fig A.2: ER diagram of final system design

Appendix B

For access to code repository: https://github.com/etriley/Capstone-Project.git

Source code:

This section of code is what can be found running on the Arduino equipped with the audio sheild. It receives input from the main control board, sets specific boolean values to true, and then in the main loop plays the appropriate sound file to the user.

```
#include <FatReader.h>
#include <SdReader.h>
#include <avr/pgmspace.h>
#include "WaveUtil.h"
#include "WaveHC.h"
#include <Wire.h>


SdReader card;    // This object holds the information for the card
FatVolume vol;    // This holds the information for the partition on the card
FatReader root;   // This holds the information for the filesystem on the card
FatReader f;      // This holds the information for the file we're play

bool low, medium, high, obstacleHigh, obstacleLow, obstacleWall, obstacleDropOff, ft1, ft2, ft3, ft4, ft5, ft6, ft7,
ft8, ft9, left, right, both, canreceiveevents;

WaveHC wave;      // This is the only wave (audio) object, since we will only play one at a time


// this handy function will return the number of bytes currently free in RAM, great for debugging!
int freeRam(void)
{
  extern int  __bss_end;
  extern int  *__brkval;
  int free_memory;
  if((int)__brkval == 0) {
    free_memory = ((int)&free_memory) - ((int)&__bss_end);
  }
  else {
    free_memory = ((int)&free_memory) - ((int)__brkval);
  }
  return free_memory;
}

void sdErrorCheck(void)
{
  if (!card.errorCode()) return;
  putstring("\n\rSD I/O error: ");
  Serial.print(card.errorCode(), HEX);
  putstring(", ");
  Serial.println(card.errorData(), HEX);
  while(1);
}

void setup() {
  canreceiveevents = false;
  byte i;

  // set up serial port
  Serial.begin(9600);
  Wire.begin(8);
  Wire.onReceive(receiveEvent);

  putstring("Free RAM: ");       // This can help with debugging, running out of RAM is bad
  Serial.println(freeRam());      // if this is under 150 bytes it may spell trouble!

  //  if (!card.init(true)) { //play with 4 MHz spi if 8MHz isn't working for you
  if (!card.init()) {         //play with 8 MHz spi (default faster!)
    putstring_nl("Card init. failed!");  // Something went wrong, lets print out why
    sdErrorCheck();
    while(1);                            // then 'halt' - do nothing!
  }

  // enable optimize read - some cards may timeout. Disable if you're having problems
  card.partialBlockRead(true);

// Looks for a FAT partition
  uint8_t part;
  for (part = 0; part < 5; part++) {
    if (vol.init(card, part))
      break;
  }
  if (part == 5) {
    putstring_nl("No valid FAT partition!");
    sdErrorCheck();
    while(1);
  }

  // Lets tell the user about what we found
  putstring("Using partition ");
  Serial.print(part, DEC);
  putstring(", type is FAT");
  Serial.println(vol.fatType(),DEC);

  // Try to open the root directory
  if (!root.openRoot(vol)) {
    putstring_nl("Can't open root dir!");
    while(1);
  }

  putstring_nl("Ready!");
  canreceiveevents = true;
}

void loop() {
```

```
canreceiveevents = false;
Serial.println("START OF LOOP");
if(left) {
  playcomplete("LEFT.wav");
  left = false;
}
else if (right) {
  playcomplete("RIGHT.wav");
  right = false;
}
else if (both) {
  playcomplete("BOTH.wav");
  both = false;
}

if(obstacleHigh) {
  playcomplete("HIGH.wav");
  obstacleHigh = false;
}
else if(obstacleLow) {
  playcomplete("LOW.wav");
  obstacleLow = false;
}
else if(obstacleWall) {
  playcomplete("WALL.wav");
  obstacleWall = false;
}
else if(obstacleDropOff) {
  playcomplete("DROPOFF.wav");
  obstacleDropOff = false;
}

if(ft1) {
   playcomplete("1FT.wav");
   ft1 = false;
}
else if (ft2) {
   playcomplete("2FT.wav");
   ft2 = false;
}
else if (ft3) {
   playcomplete("3FT.wav");
   ft3 = false;
}
else if (ft4) {
   playcomplete("4FT.wav");
   ft4 = false;
}
else if (ft5) {
   playcomplete("5FT.wav");
   ft5 = false;
}
else if (ft6) {
   playcomplete("6FT.wav");
   ft6 = false;
}
```

```
  else if (ft7) {
     playcomplete("7FT.wav");
     ft7 = false;
  }
  else if (ft8) {
     playcomplete("8FT.wav");
     ft8 = false;
  }
  else if (ft9) {
     playcomplete("FT9.wav");
     ft9 = false;
  }
  Serial.println("TRUE");
  canreceiveevents = true;
  Serial.println("END OF LOOP");
  delay(100);
}

void receiveEvent(int inputLength) {
  char type = Wire.read(); // receive byte as a character
  char distance = Wire.read();
  char direc = Wire.read();
  if(canreceiveevents) {
  Serial.println("    RECEIVING EVENT");
    switch(type) {
      case('H'):
        obstacleHigh = true;
       break;
    case('L'):
        obstacleLow = true;
       break;
    case('B'):
        obstacleWall = true;
       break;
    case('D'):
        obstacleDropOff = true;
       break;
    default:
        obstacleHigh = false;
        obstacleLow = false;
        obstacleWall = false;
        obstacleDropOff = false;
      break;
    }
    switch(distance)  {
      case('1'):
        ft1 = true;
        break;
    case('2'):
        ft2 = true;
        break;
    case('3'):
        ft3 = true;
        break;
    case('4'):
        ft4 = true;
        break;
    case('5'):
        ft5 = true;
        break;
    case('6'):
        ft6 = true;
        break;
    case('7'):
        ft7 = true;
        break;
    case('8'):
        ft8 = true;
        break;
    case('9'):
        ft9 = true;
        break;
    default:
        ft1 = false;
        ft2 = false;
        ft3 = false;
        ft4 = false;
        ft5 = false;
        ft6 = false;
        ft7 = false;
        ft8 = false;
        ft9 = false;
        break;
    }

    switch(direc) {
      case('L'):
          left = true;
          break;
      case('R'):
          right = true;
          break;
      case('B'):
          both = true;
          break;
      default:
          left = false;
          right = false;
          both = false;
          break;
    }
  }
  else {
    Serial.println("    CANT RECEIVE EVENT");
  }
}

void playcomplete(char *name) {
  Serial.println(name);
  playfile(name);
  //    WHILE WAV IS PLAYING DO NOTHING
  while (wave.isplaying) {
  }
}

void playfile(char *name) {
  //    CHECK FOR FILE
  if (!f.open(root, name)) {
    putstring("Couldn't open file "); Serial.print(name);
    Serial.println("");
    return;
  }
  //  VALIDATE FILE IS OF TYPE WAV
  if (!wave.create(f)) {
    putstring_nl("Not a valid WAV");
    return;
  }
  //    BEGIN PLAYBACK
  wave.play();
}
```

References

[1] Central and Peripheral Visual Impairment and the Risk of Falls and Falls with Injury
Patino, Cecilia M. et al.
Ophthalmology , Volume 117 , Issue 2 , 199 - 206.e1.

[2] "Eye Charts." *Eye Charts*. Web. 31 Jan. 2016. <http://www.cascadilla.com/eyecharts/>.

[3] Hollingshead, Todd. "BYU's Smart Object Recognition Algorithm Doesn't Need Humans." *BYU's Smart Object Recognition Algorithm Doesn't Need Humans*. BYU News, 15 Jan. 2014. Web. 31 Jan. 2016.

[4] "Moving Object Detection (MOD)." *NISSAN TECHNOLOGICAL DEVELOPMENT ACTIVITIES*. Web. 30 Jan. 2016.

[5] "Visual Impairment and Blindness." *World Health Organization*. WHO, Aug. 2014. Web. 30 Jan. 2016. <http://www.who.int/mediacentre/factsheets/fs282/en/>.

[6] Gondane, Rahul. "Most Common Age-Related Eye Vision Problems." *LensPick Blog Sunglasses Contact Lens Lens Solutions Eyeglasses*. 18 Mar. 2015. Web. 31 Jan. 2016.

[7] "Diabetes." *World Health Organization*. Jan. 2015. Web. 31 Jan. 2016.

[8] "Assisting the Visually Impaired: Obstacle Detection and Warning System by Acoustic Feedback." Web. 1 Feb. 2016.

[9] C, Sylvain, T, Daniel, V, Frederic "Wearable Obstacle Detection System for visually impaired People". Web 1 Feb. 2016

[10] "LV-MaxSonar-EZ Datasheet." Web. 14 Feb. 2016. <"LV-MaxSonar-EZ Datasheet." Web. 14 Feb. 2016. >

[11] "Arduino Uno." *Arduino - ArduinoBoardUno*. Web. 17 Feb. 2016. <https://www.arduino.cc/en/Main/ArduinoBoardUno>

[12] "Wave Shield." *Overview*. Adafruit. Web. 14 Feb. 2016. <https://learn.adafruit.com/adafruit-wave-shield-audio-shield-for-arduino/overview>

[13] B, Colin. "What Is Agile Development?" *Screenmedia*. Web. 14 Feb. 2016. <http://www.screenmedia.co.uk/blog/2014/08/what-is-agile-development-a-brief-introduction/>.

[14] "Gorsuch Commons." Web. 07 Mar. 2016. <https://www.uaa.alaska.edu/ccs/facilities/commons/>.

[15] "Halloween Pumpkin." *Step 1: Playing Sounds through the Speaker*. Web. 07 Mar. 2016. https://learn.adafruit.com/halloween-pumpkin/step-1

[16] "1027 Flat Vibrating Vibration Motor - Silver (5 PCS)." *DX.com*. Web. 07 Mar. 2016. <http://www.dx.com/p/1027-flat-vibrating-vibration-motor-silver-5-pcs-154245#.VuC3Z-bvasY>.

[17] "Mythological Creatures - Argus Panoptes." *Pinterest*. Web. 23 Mar. 2016. <https://www.pinterest.com/pin/341288477986802686/>