Connected Crosswalk Assistance (CoCA), a Crosswalk Remote Activation Device for Visually-Impaired Individuals

Rohith Yelisetty, Arnav Bhalla, Anirudh Mantha, Adithiya Balaguru, Tanush Kallem

Abstract— Currently, thousands of pedestrian accidents occur every year, and visually impaired individuals face an apparent disadvantage when it comes to crossing streets. Despite there being numerous smart walking sticks already present in the market, they fail to allow for easier activation of crosswalks for visually impaired individuals. Therefore, Connected Crosswalk Assistance or CoCA was developed in order to allow for the remote activation of crosswalks in addition to warning users of potential dangers while on the crosswalk. This includes alerting for oncoming objects while also notifying the user when they are veering off of the crosswalk path. The stick utilizes numerous aspects of Arduino technology including a microcontroller, an RFID reader, and various sensors in order to successfully achieve the goal of the CoCA white cane. CoCA is able to prevent numerous pedestrian accidents and simplify an important aspect of visually impaired individuals' lives.

Keywords— Arduino, Visually-impaired, Crosswalks, Object Detection, White Cane

I. Introduction

According to the World Health Organization, around 2.2 billion people, or 27.6 percent of the world's population are visually impaired or visually impaired[1]. This gives around a quarter of the world a serious disadvantage as according to a survey conducted by the NIH, 73.63 percent of individuals answered that they felt vision was the most important sense[3]. Additionally, 80 percent of a human's impressions and experiences around them occur with the use of their eyes. Visual impairment is a partial or complete loss of vision due to genetic, neurological, or psychological factors[2]. However, there are various types of visually impairments. Partial visual impairment is a condition where an individual has limited vision while complete visual impairment occurs when an individual has a total loss of vision.

The loss of vision poses an immense disadvantage to the visually impaired community. One of the most detrimental of these issues includes navigating streets and crosswalks. visually impaired individuals have no easy and widespread method of knowing when to cross and if they are staying on the crosswalk path. Thousands of pedestrian deaths occur each year, and visually impaired individuals have a significantly increased chance of being affected by these accidents. However, there are numerous aspects of current technology that help to ease the gap in pedestrian travel between the visually impaired community and society.

Currently, the most prominent of these solutions is accessible pedestrian signals (APS) or crosswalk signals that play an audio signal to allow visually impaired individuals to help locate the activation button[4]. However, these methods are relatively inaccessible for individuals residing in rural or suburban regions. Another current technology available is the WeWALK smart walking cane that allows for connection to mapping software and awareness of surrounding objects and places. Yet, a major drawback to this product is the high prices that many visually impaired individuals can not readily afford.

Connected Crosswalk Assistance, or CoCA, is a smart walking stick for visually impaired individuals that will remotely connect to crosswalks, and warn individuals of approaching objects while ensuring they stay within the crosswalk path. There are numerous different aspects to the white cane that allow for the product to accomplish its goals. These include a microcontroller to control the electronics, a Wi-Fi module to connect to the crosswalk server, and an ultrasonic sensor for surrounding object detection. Additionally, the white cane incorporates a gyro sensor to ensure the user remains in the path, and a Radio-Frequency Identification (RFID) sensor along with a pushbutton to access and remotely activate the crosswalk. CoCA can help improve the lives of thousands of visually impaired individuals.

II. Methodology

In order to assess the need for such a product in the market, the team met with two counselors from the Department for the visually impaired and Visually Impaired (DBVI) in Virginia, Ms. Melanie Hughes and Ms. Sariana Marrero Velez who gave the team further insight surrounding daily struggles that visually impaired individuals face and problems that needed to be solved in a potential solution. This included easy access to crosswalks, ensuring that there are no surrounding objects present, and lastly, assisting a user to stay within the crosswalk path while crossing the street. Therefore, assisted by counselors of the visually impaired, the team gave a survey to numerous visually impaired individuals asking them if they face such problems in their daily

Contact data: Rohith Yelisetty, Arnav Bhalla, cocawalk@gmail.com

lives.

Therefore, the CoCA walking cane was developed using numerous aspects of current technology including a microcontroller, an RFID sensor, and various other sensors. The walking stick has a multistep process in order to remotely activate crosswalks while ensuring that the user is safe at all times. The walking stick uses an RFID sensor along with a pushbutton in order to remotely activate the crosswalk. After activating the crosswalk, the ultrasonic and gyro sensors ensure that the user's surroundings are safe and ensure that the user maintains a heading consistent with the crosswalk path. Lastly, a vibration motor with different patterns is used to signify the individual's ability to cross, and the potential dangers in the crosswalk.

Sections:

II.I Microcontroller
II.II Crosswalk Identification and Activation
II.III Surrounding Awareness
II.IV Warning System
II.V Wi-Fi Server Connection

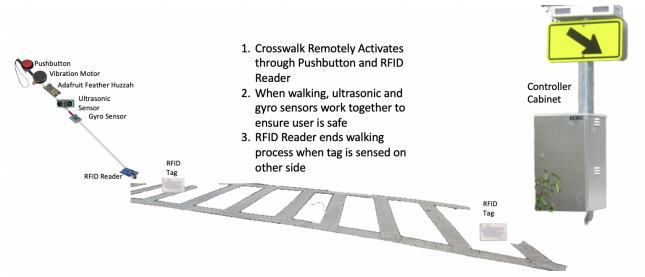

Fig. 1: Diagram of CoCA walking stick process.

a. II.I Microcontroller

The microcontroller serves as the motherboard of the CoCA walking cane and controls all the other sensors and operations. These microcontrollers are programmable using the Arduino IDE which uses a variant of the C++ programming language. Microcontrollers have varying RAM sizes along with different integration modules. When comparing the various possible options for microcontrollers, the team took into account memory size, power methods, and price. Lastly, one major element the team looked for was WiFi module integration to limit the number of external sensors required. After comparing the pros and cons of all the potential microcontrollers, the team was able to narrow the options down from four microcontrollers to two, the Adafruit Feather Huzzah and the NodeMCU. The main difference between the two controllers was the power source with the Feather Huzzah allowing for easier access and more variety while the NodeMCU only worked with an external power supply connected through the VIN pin. Therefore, the Adafruit Feather Huzzah was selected as the microcontroller for the CoCA walking stick. Before being able to use the Adafruit Feather Huzzah's features, it was necessary to import the Arduino libraries in the IDE and select the correct board.

Arduino Library:

#include <Wire.h>

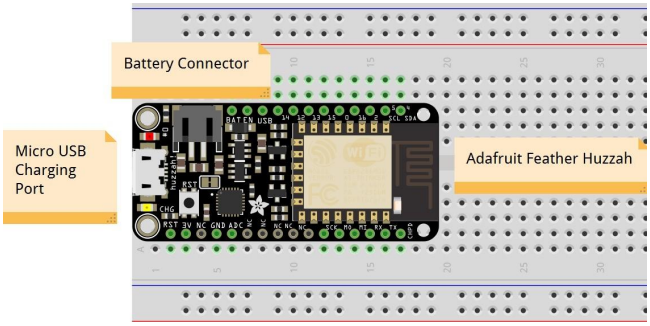

Fig. 2: Adafruit Feather Huzzah with Lipo Battery Connector and Micro USB Charging Port

The Arduino Wire library allows for communication between I2C sensors and devices such as the gyro sensor for the microcontroller through the SDA (data line) and SCL (clock line) ports. Additionally, the Wire library is responsible for containing many of the main functions of the Arduino IDE that make coding more streamlined and easier to understand. The stick uses the Wire library in order to communicate with the gyro sensor by sending and receiving sensor-specific data.

b. II.II Crosswalk Identification and Activation

The main purpose of the CoCA walking stick is to allow for the remote activation of crosswalk signals. Therefore, identifying the crosswalk and knowing when the individual wants to cross the street is an essential aspect of the product. The team decided to implement two important components: a Radio-Frequency Identification (RFID) sensor and a pushbutton.

Parts:

RFID Sensor and Tags
Pushbutton

Part 1: RFID Sensor and Tags

The first aspect of crosswalk identification involves locating where the crosswalk signal and the crosswalk area are. RFID tags are an easy and effective way to transmit data between a wireless and powerless chip, often in the form of a sticker, and a powered reader that can read the ID of the specified tag and the message it carries. Although RFID tags are generally used for tracking packages, the team decided to use these tags in order to generate separate IDs for each crosswalk. After researching various types of tags, the team was able to narrow it down to three RFID tags: Low-Frequency, High-Frequency, and Ultra-High Frequency. The main difference between all the tags was the identification range or the distance that the tag ID would be recognizable with the Low-Frequency tags having the lowest at around 5 inches and the Ultra-High Frequency tags having the highest at around 30 meters. The team decided that it would be optimal to have a tag that has a short identification range as the user should be able to precisely pick which crosswalk they would like to activate.

Implementation Steps:

1. Import Necessary Libraries
2. Establish Sensor Pins and Initialize the RFID Sensor Object

3. Allow for RFID Sensor to Set Up
4. Assessing if the RFID Sensor is Near an RFID Tag
5. Accessing the Tag ID

Step 1: Import Necessary Libraries - Allow for access to sensor-specific methods throughout the code such as when reading the RFID tag ID.

```
#include <SPI.h>
#include <MFRC522.h>
```

The Arduino SPI library enables communication between the microcontroller and various SPI devices, and in this case, the RFID reader. Similar to the Wire library, this library allows for the sending and receiving of sensor-specific data. The MFRC522 library is the RFID reader library that gives access to sensor-specific methods that the stick can use in order to read certain RFID tag IDs. These two libraries in combination allow for the microcontroller to communicate with the RFID reader and for the RFID reader to identify RFID tags and their specific tag IDs.

Step 2: Establish Sensor Pins and Initialize the RFID Sensor Object

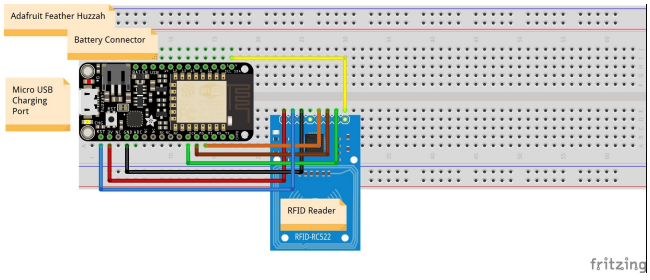

Fig. 3: Wiring diagram of RFID reader with Adafruit Feather Huzzah

#define SS_PIN 4 - **SDA (Data Line) Pin**
#define RST_PIN 0 - **Reset Pin**
MFRC522 rfid(SS_PIN, RST_PIN);

This initializes the RFID sensor object and defines which pins to access when trying to reset or communicate through I2C.

Step 3: Allow for RFID Sensor to Set Up

```
for (uint8_t t = 4; t > 0; t–) {
  Serial.printf("[SETUP] WAIT %d... n", t);
  Serial.flush();
  delay(1000);
}
```

This for-loop allows for the Serial port to refresh while also giving time for the various sensors, specifically the RFID and ultrasonic sensors to initialize. This occurs in the setup phase of the code which happens once at the beginning of the program.

Step 4: Assessing if the RFID Sensor is Near an RFID Tag

The RFID sensor has numerous specific functions that allow the sensor to identify if there is an RFID tag in proximity. The first step involves activating the SPI communication to allow for communication with the sensor. Additionally, this step includes activating the RFID sensor to allow for the sensing of near-range RFID tags.

```
SPI.begin();
rfid.PCD_Init();
```

After initializing SPI communication and the RFID sensor, the sensor is able to identify whether there is a tag present. RFID readers work by emitting radio waves through a mini-antenna that activates the tag which, in turn, sends a confirmation signal back to the antenna. Additionally, there are two different types of RFID tags: active tags, and passive tags. Active tags contain their power source such as a battery and allow for the slightly quicker transmission of signals while passive tags do not require a power source and receive power from the RFID reader antenna. Passive RFID tags also ensure precision when a user is picking which crosswalk they wish to cross. Therefore, a passive RFID tag was used for a cheaper and more effective solution.

```
rfid.PICC_IsNewCardPresent();
```

This method is a boolean which returns true if there is an RFID tag present in the sensor's vicinity. It is used in the loop after initializing the RFID sensor in order to continuously check for updates and for new RFID tags. After identifying the tag (*Part 5: Accessing the Tag ID*), the stick will ensure that the pushbutton is being pressed (View *II.II Part 2: Pushbutton* for more information) and will then activate the crosswalk. If already walking, the stick will end the walking process, and restart the loop. After checking for tags and identifying their tag IDs, the SPI communication is turned off in order to optimize efficiency for the other processes in the code.

Step 5: Accessing the Tag ID

The last aspect involving the RFID reader includes accessing and storing the specific tag ID whenever one is found. This involves reading the individual IDs from the radio waves that are emitted by the passive tags. Whenever the sensor finds a tag in proximity, the RFID reader automatically stores the unique identifier (UID) in a public field that is accessible. However, this value is unreadable when initially stored. Therefore, the code converts the UID from hexadecimal into a string to allow for easier storage.

```
String rfidTag;
for (int i = 0; i < rfid.uid.size; i++) {
  rfidTag.concat(String(rfid.uid.uidByte[i] < 0x10 ? "0" : ""));
  rfidTag.concat(String(rfid.uid.uidByte[i], HEX));
}
```

As mentioned, the string concatenates individual bytes from the tag UID in order to make it easier to work with and store.

Part 2: Pushbutton

The second aspect of crosswalk identification and activation involves the push button in order to identify whether the user wishes to cross or not. When analyzing and comparing different activation devices, the team focussed on two main possible ideas: a hands-free motion device and a pushbutton. The hands-free motion device would use similar technology to an ultrasonic sensor where it would activate the crosswalk when a tag was present along with the motion device being obstructed, signaling the user wishes to cross. On the other hand, the pushbutton would only activate when fully pressed and uses much more simplified technology. Therefore, the team chose to utilize the button as it was more effective, ensured that the user wanted to cross, and was cheaper.

Implementation Steps:

1. *Establish Button Pins and Initialize Button Object*
2. *Check if the Button is Pressed while RFID Tag is within Reader Range*

Step 1: Establish Button Pins and Initialize Button Object

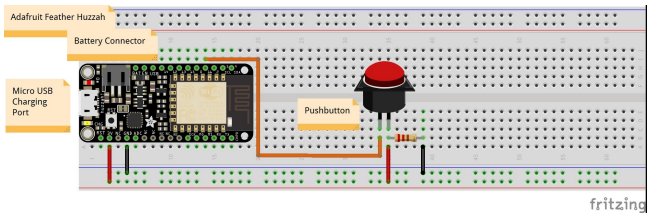

Fig. 4: Wiring Diagram of Pushbutton with Adafruit Feather Huzzah

#define BUTTON_PIN 16 - **Pushbutton Pin**

pinMode(BUTTON_PIN, INPUT);

The first line assigns a global variable that sets the button pin to GPIO Pin 16 which can be used throughout the rest of the program. The next line is present in the setup phase of the Arduino program which defines the pushbutton as a sensor that provides data for the microcontroller to use as an INPUT option.

Step 2: Check if the Button is Pressed while RFID Tag is within Reader Range

The main aspect of the pushbutton involves checking whether the pushbutton is pressed while there is a tag in the RFID reader's vicinity at that specific time. Therefore, the code checks for the button state after finding an RFID tag. If one is found, the pushbutton will call a get request to the server in order to officially activate the crosswalk (View *II.V WiFi Server Connection* for more information). The code uses the built-in *digitalRead();* command in order to read the input from the pushbutton. When the input is found to return HIGH, signaling that the pushbutton is being pressed, while there also is an RFID tag present, the walking stick is able to identify that the user wishes to cross at a certain street crosswalk.

c. II.III Surrounding Awareness

After identifying and activating the crosswalk, the crosswalk will send a signal back to the CoCA walking stick notifying the user that it is safe to cross the street (II.V WiFi Server Connection). The next aspect of CoCA involves ensuring that the visually impaired individual remains safe while crossing the street or crosswalk. Therefore, the team used two individual pieces of technology to maintain awareness of surroundings: an ultrasonic sensor in order to warn users of incoming objects or objects that are in front of them along with a gyro sensor to ensure that the visually impaired individual does not veer off of the crosswalk path.

Parts:

1. *Ultrasonic Sensor*
2. *Gyro Sensor*

Part 1: Ultrasonic Sensor

The first aspect of ensuring that the user stays safe in the crosswalk involves making sure that the user's surroundings are clear. Therefore, the team decided upon using a type of object-detection device that is able to detect objects at any point in front of the given sensor's transmitter and receiver. There were four main types of object-detection sensors: electro-mechanical, pneumatic, photoelectric, and ultrasonic sensors. The main difference between all of the sensors was the methods to detect objects in front, with the electro-mechanical and ultrasonic sensors being the most effective. After factoring in the costs of the sensor, it became evident that the ultrasonic sensor was the best possible option for an object-detection sensor.

Implementation Steps:

1. *Import Necessary Library*
2. *Define Ultrasonic Sensor Pins and Initialize Object*
3. *Check for Objects in the Ultrasonic Sensor's Range*

Step 1: Import Necessary Library

#include <HCSR04.h>

This library gives the Arduino code access to the specific ultrasonic sensor (HCSRO4) library and sensor-specific methods. The HC SR04 ultrasonic sensor is a sensor that measures distance using ultrasound waves. The sensor emits these waves through the transmitter and measures the amount of time it takes to return these waves to the receiver in order to calculate the distance between the sensor and the objects in front.

Step 2: Define Ultrasonic Sensor Pins and Initialize Object

#define echoPin 2
#define trigPin 14
HCSR04 hc(2, 14);

The first two lines define the echo pin (transmitter) and the trig pin (receiver) of the ultrasonic sensor in order to allow for access later in the program. The last line initializes the ultrasonic sensor by defining which pins to access for measuring the distance between the sensor and specific

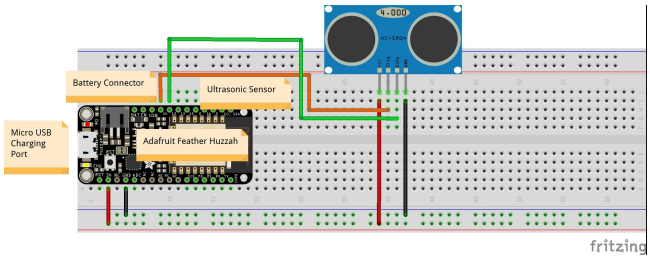

Fig. 5: Wiring Diagram of Ultrasonic Sensor with Adafruit Feather Huzzah

objects.

Step 3: Check for Objects in the Ultrasonic Sensor's Range

After initializing the object, the ultrasonic sensor continuously checks if there are objects in front of the device in order to ensure that the visually impaired individual's surroundings are clear. The code uses the ultrasonic sensor command .dist(); which returns the distance of the closest object in centimeters. While the user is utilizing the CoCA stick, the individual will be warned every time an object is detected within 25 centimeters (10 inches) using a vibration motor (View II.IV Warning System for more information). The ultrasonic sensor functions not only when the user is on the crosswalk, but also when on a normal sidewalk in order to ensure maximum safety of the users.

Part 2: Gyro Sensor

The second aspect of ensuring safety includes verifying that the user maintains a straight heading while on the crosswalk or street. Therefore, the team decided upon using a gyro sensor which would be able to determine the degree to which the user is traveling based on how much the stick is turning. After analyzing various types of gyro sensors, the team was able to narrow in on three main gyro sensors: the Arduino GY-521, the Adafruit-ADXL335, and the Adafruit-LIS3DH. Since there were very few differences when simply comparing the properties of each of the sensors, the team compared the effectiveness of each sensor and found that the Arduino GY-521 takes too long to calibrate and often loses its calibration within 30 seconds. The Adafruit-ADXL335, although more effective, still lost its calibration frequently. Therefore, the team decided on using the Adafruit-LIS3DH as it was the most effective and reliable sensor with a relatively low price.

Implementation Steps:

1. Import Necessary Library

2. Instantiate Gyro Sensor Object and Allow for Calibration of Sensor

3. Check if the User is Veering Off of the Crosswalk

Step 1: Import Necessary Library

#include <Adafruit_LIS3DH.h>

This library gives the Arduino code access to sensor-specific methods that allow for the usage of the gyro sensor. The Adafruit LIS3DH communicates using I2C communication which uses the Wire library and is able to measure the orientation of the gyro sensor through three axes (X, Y, and Z).

Step 2: Instantiate Gyro Sensor Object and Allow for Calibration of Sensor

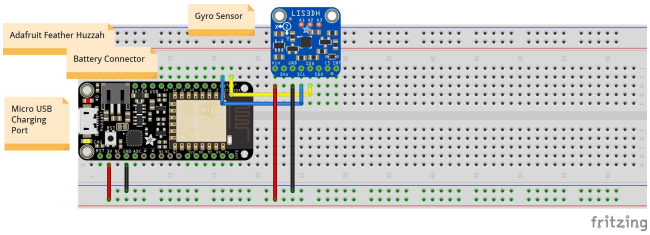

Fig. 6: Wiring Diagram of Gyro Sensor with Adafruit Feather Huzzah

Adafruit_LIS3DH gyro = Adafruit_LIS3DH();

This first line instantiates a gyro sensor object and creates the object that will communicate with the microcontroller.

Wire.begin(12, 13);

The line above sets the SDA (Data Line) and SCL (Clock Line) pins for I2C communication. After establishing the gyro sensor pins, the program allows for the sensor to calibrate itself before reading the axis values. Additionally, the sensor sets the rate and the speed at which it recalculates its degree based on a variety of external factors.

Step 3: Check if the User is Veering Off of the Crosswalk

The last step in using the gyro sensor involves reading the certain axis' value. Converting the gyro sensor's raw values to degrees is a mathematical process that is a sensor-specific method. However, when testing this method, the team found that it was very inaccurate and decided upon testing its own values. The team discovered that a raw value of 2000 equals around 30 degrees. Therefore, whenever the walking stick knows that the user is in the middle of a crosswalk while also maintaining a degree of more than 30 degrees, the vibration motor will buzz in a specific pattern (View *II.IV Warning System* for more information).

d. II.IV Warning System

The CoCA walking cane establishes a series of warning announcements in order to notify users of various dangers such as when to cross, when there are objects in front of the user, and when the user is veering off of the crosswalk. There were various options for warning devices, however, two main sensors were mainly debated upon: a speaker and a vibration motor. After analyzing the differences between the two sensors, the team was able to decide upon using the vibration motor as it relied more on the individual's sense of touch rather than hearing, as it would be difficult to hear the speaker in crowded city streets. The vibration motor works by sending vibrations through the stick (essentially vibrating the entire stick) through different pre-established patterns in order to warn users of various dangers.

Implementation Steps:

1. Establish Motor Pin and Instantiate Motor Object

2. Set Various Patterns for Different Warnings

Step 1: Establish Motor Pin and Instantiate Motor Object

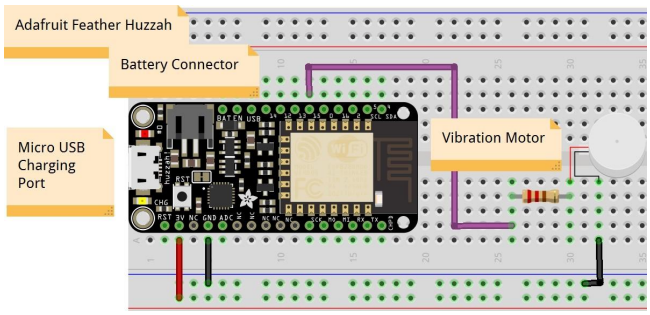

Fig. 7: Wiring Diagram of Vibration Motor with Adafruit Feather Huzzah

#define MOTOR_PIN 15 - **Vibration Motor Pin**
pinMode(MOTOR_PIN, OUTPUT);

The first line sets the vibration motor pin to a global variable, GPIO pin 15 while the second line establishes that the motor is an output device which means that the microcontroller will send commands to the vibration motor for it to execute.

Step 2: Set Various Patterns for Different Warnings
The CoCA walking cane sets different buzzer patterns in order to establish various warning signals for the user. The Arduino code does this by setting brief periods where the vibration motor is turned on and brief periods where the vibration motor is turned off through Arduino delays. Three vibration patterns are sent out by the vibration motor in order to warn and notify the users of various aspects.
Patterns:
1. Notifying User When They Can Cross:
- 1 Second Buzz – 1 Second Off – 1 Second Buzz
2. Warning User of Object In Front:
- ½ Second Buzz
3. Warning User of Veering Off of Crosswalk:
- 2 Second Buzz

e. II.V WiFi Server Connection

The last and most important aspect of the CoCA walking cane includes connectivity to the WiFi server to activate and receive signals from the crosswalk signal. Crosswalks have a control panel known as a controller cabinet that directs the movements based on the push of a button at the crosswalk. Additionally, each crosswalk is attached to an overarching server which stores a unique ID for each signal. The purpose of the CoCA walking stick is to adapt and improve this server by placing an RFID tag at each signal in order to allow for remote activation. However, for testing purposes, a miniature server was created where it simply stored the unique RFID tag IDs and an ID for the crosswalk's WiFi module. Additionally, the tests consisted of connecting to the local WiFi rather than utilizing cellular service through a chip that will be present in a future iteration of the walking stick.
Parts:
1. Nodejs WiFi Server
2. Arduino Server Connection

Part 1: Nodejs WiFi Server
The WiFi Server works by parsing arguments passed through the URL when calling the server in order to figure out which crosswalk the stick is acting upon in addition to which action the stick wishes to make. In regard to the walking cane, there are two methods that it mainly calls from the server: logging a request with the server and checking to see if the server is currently turned on. The Nodejs server is a backend system that utilizes JavaScript in order to create a database of all of the crosswalk statistics in addition to allowing for the stick to receive data from certain aspects of the server.
Implementation Steps:
1. Import Necessary Libraries
2. Create Hashmap for Crosswalk Data Storage
3. Create GET Methods for Walking Stick to Access

Step 1: Import Necessary Libraries

```
const express = require('express');
const app = express();
const url = require('url');
```

The first and second libraries allow for the server to be a web server that runs on localhost for testing purposes. The third library imports a version of jQuery which allows the code to break apart the arguments passed through the URL from the stick's Arduino code.

Step 2: Create Hashmap for Crosswalk Data Storage
The most important aspect of the WiFi server involves creating a data set through a hashmap that stores each crosswalk's essential data. These include the corresponding RFID tags, the crosswalk ID, the last time it was requested to turn on, and the last time that the crosswalk was turned on. All of these data points make it easier for the server to know when to turn on the crosswalk signal and when to notify the user that the crosswalk that they specifically requested is turned on.

Step 3: Create GET Methods for Walking Stick to Access
The last aspect of the WiFi server involves creating the GET methods that the stick calls on through its Arduino code. The first method that the stick accesses involves logging a request to turn the crosswalk on. The server parses the URL arguments in order to get which crosswalk the stick would like to turn on based on the tag ID given by the stick. Next, if the last request time value for that specific ID is null in the hash map and the crosswalk is not currently turned on, it saves the current time for the button press. The second method that is used by the stick in the server includes checking if the crosswalk is currently on after previously logging a request with that specific crosswalk. The server checks if the last start time value in the hashmap is null and if it allows for enough time to cross safely before notifying the stick that it is allowed for the user to cross. Other methods that are available in the test server make it easier for the traffic light and crosswalk signal to know when to turn on.

Part 2: Arduino Server Connection

The second part of the connection involves the CoCA walking cane accessing the different methods of the localhost WiFi server. The walking cane will call upon the two different GET methods described above based on when it is needed. The Adafruit Feather Huzzah utilizes its built-in WiFi module in order to access the server. However, one main difference between a real-world version of the walking cane and the test version built was that the test stick utilized the local WiFi instead of using cellular service present near the crosswalks.

Implementation Steps:

1. *Import Necessary Libraries*
2. *Initialize WiFi Object and Connecting to WiFi*
3. *Accessing GET Methods from WiFi Server*

Step 1: Import Necessary Libraries

```
#include <ESP8266WiFi.h>
#include <ESP8266WiFiMulti.h>
#include <ESP8266HTTPClient.h>
#include <WiFiClient.h>
```

These libraries allow for the Adafruit Feather Huzzah and the rest of the sensors to establish communication with the WiFi module in addition to the module being able to access the internet and access various URLs that the team tested (View *III.Results* for more information). For the CoCA walking cane, the WiFi module is used in order to access the test local host server.

Step 2: Initialize WiFi Object and Connecting to WiFi

```
ESP8266WiFiMulti WiFiMulti;
WiFiClient client;
```

The WiFi module is an ESP8266 module that is able to access URLs on the internet and receive data returned from each website. Additionally, the WiFiClient object allows the module to connect to the WiFi.

```
WiFi.mode(WIFI_STA);
WiFiMulti.addAP("******", "**********");
```

The two lines above set the WiFi type while also allowing the WiFi module to connect to the local network's SSID and password (Blurred out for privacy).

Step 3: Accessing GET Methods to WiFi Server

The last aspect of connecting the WiFi module and the CoCA walking cane with the server involves calling these methods when necessary. The first GET method that is accessed logs a request with the server asking for the specific crosswalk signal to be turned on based on the tag ID read by the RFID sensor. This method is called when an RFID tag is found in addition to the pushbutton being pressed at the same time. It calls a method created in the walking cane's Arduino code that uses the URL parameter in order to call the GET method and stores whatever data is given back in a variable that is returned. For example, *"GET /turnOn?tagId="* + *tagID* + *" HTTP/1.1"* requests the server to turn on the corresponding crosswalk to the specific RFID tag ID given.

The second GET method continuously checks with the server if the crosswalk signal that was previously requested is currently on and is allowing the user to cross. This method is automatically called by the Arduino code after the *turnOn* GET method is called until it is allowed for the user to cross. This method calls the *isItOn* GET method in the server in the same way as mentioned above in order to receive the status of the crosswalk.

III. Results

Several tests needed to be conducted to ensure the accuracy and efficiency of all the components in the CoCA walking cane. These tests involved connecting the Adafruit Feather Huzzah to the Arduino software and recording the graphical data in the Arduino Serial Plotter and the text output in the Arduino Serial Monitor. The first test measured the position and angle of the gyro sensor, which was important to make sure that the sensor accounted for any major variation while tracking the movement of the user on the sidewalk. The first part of this test held the gyro sensor still to detect the significance of the tilt the sensor had while remaining in a stable position (Figure 8). Secondly, the team steadily rotated the gyro sensor across the x-axis, to confirm that the sensor could capture a sudden change in angle (Figure 9).

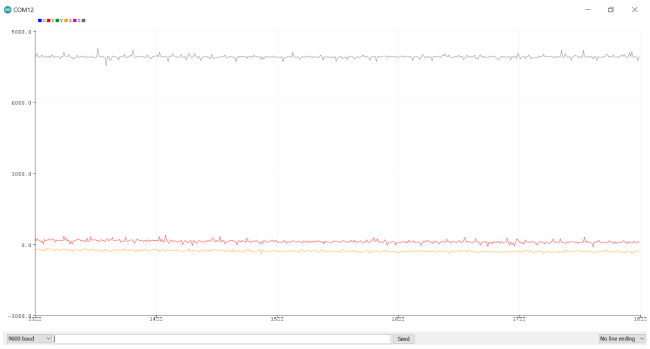

Fig. 8: Gyro Sensor Kept Still; X-Axis: Red; Y-Axis: Orange; Z-Axis: Black;

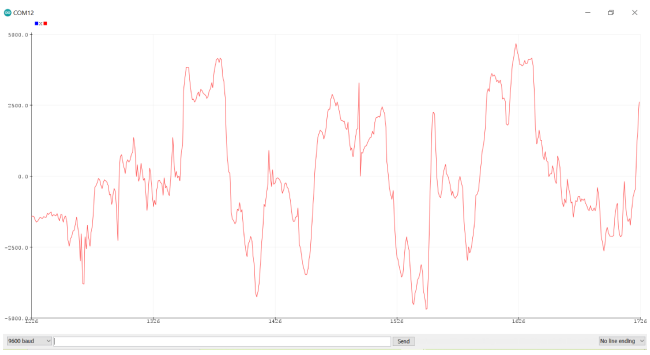

Fig. 9: Gyro Sensor Reading X-Axis when turning the gyro; Note the Y-Axis (-5000 degrees – 5000 degrees)

The next test performed measured the accuracy of the ultrasonic sensor. As one of the most important components of the walking cane, the ultrasonic sensor must be able to immediately warn the user of any imminent objects approaching. In the first section of this test, the team put the ultrasonic

sensor against a wall and calculated the variance in distances measured over a period of time (Figure 10). The second portion of this test required the team to approach the ultrasonic sensor from a distance (Figure 11). This was necessary to ensure that the ultrasonic sensor could repeatedly update the proximity of objects as they came near the walking cane.

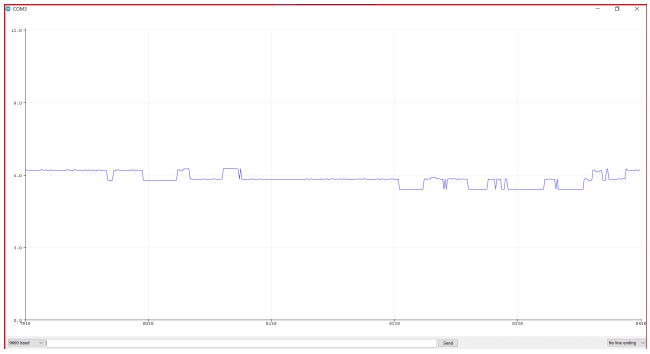

Fig. 10: Ultrasonic Sensor with Hand in front; Distance: Blue; Note the Y-Axis (3 cm – 9 cm)

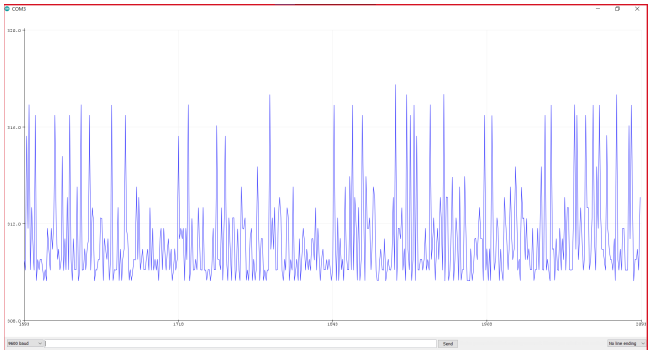

Fig. 11: Ultrasonic Sensor against a wall; Distance: Blue; Note the Y-axis (308 cm – 320 cm)

The third test conducted was the most important for the overall goal of the walking cane to remotely activate the crosswalk signal. In this test, whenever the RFID tag was detected, the RFID reader was required to print out the ID of the tag in the Arduino Serial Monitor (Figure 12). The final test involved connecting the WiFi module to a default server and a local server that was created by the team solely for this test (Figure 13). The two servers were necessary to confirm that the module would function regardless of the connection (Figure 14). The outputs of this test were also displayed in the Arduino Serial Monitor.

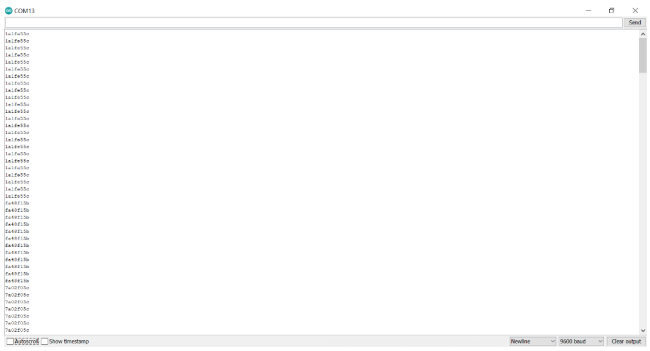

Fig. 12: RFID Tag Reader printing tag ID

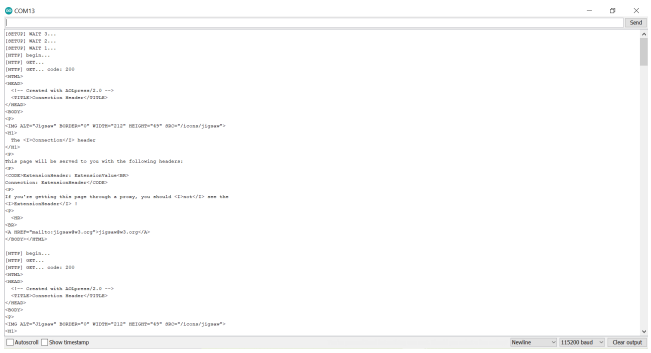

Fig. 13: Wi-Fi module connecting to http://jigsaw.w3.org/HTTP/connection.html

The survey sent out to the visually impaired individuals with the help of counselors from the DBVI provided important details to the team, such as the fact that these individuals find pressing the crosswalk button difficult (Figure 15). Additionally, 75 percent of the survey respondents said that they found APS and parallel traffic unreliable for their safety (Figure 16). Finally, a significant percentage of the visually impaired individuals at the DBVI responded that they often bump into others while on the streets (Figure 17), and that they would use a smart walking cane if accessible (Figure 18).

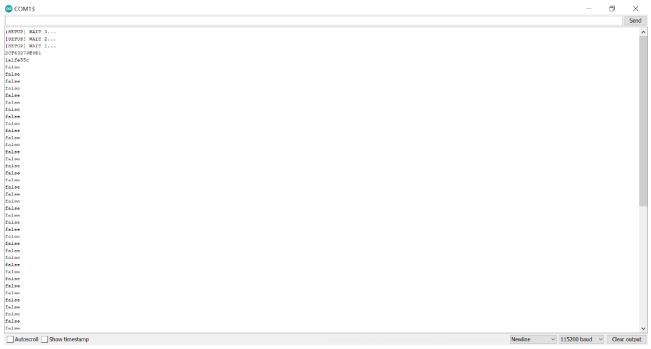

Fig. 14: Wi-Fi Module connecting to local server created by team

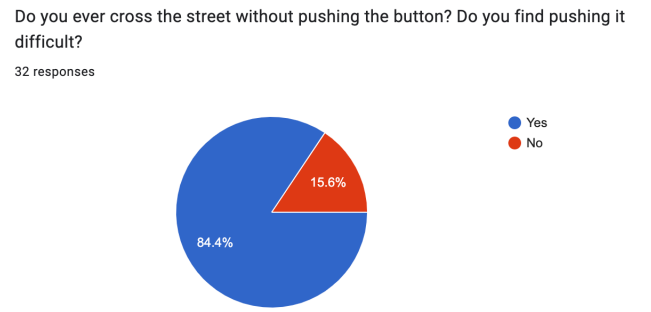

Fig. 15: First survey question

IV. Discussion

Throughout the development of the CoCA walking stick, a few problems arose that had a quick fix, such as the ultrasonic and gyro sensors drifting. However, other issues still need refinement, such as the adaptation of crosswalk controller cabinets. One problem when conducting our tests was the RFID reader that was only able to detect the first tag when

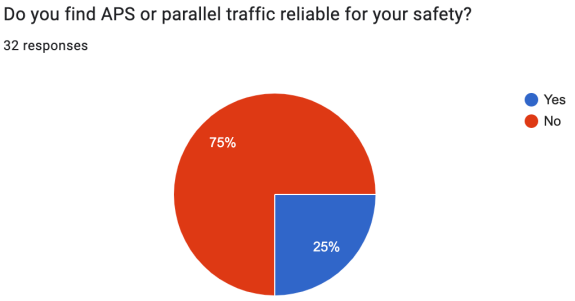

Fig. 16: Second survey question

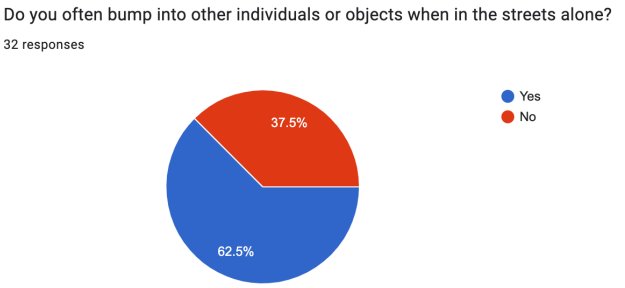

Fig. 17: Third survey question

approaching the crosswalk. After scanning the initial tag, it could not recognize any other tag, most likely because of the drift within the sensors. The data gathered by the sensor varies significantly, which in turn prompts the microcontroller to gather false inputs and activate the vibration motor at irregular moments. A solution to this issue would be to recalibrate the sensors immediately after detecting the RFID tag on the first end of the crosswalk, which requires further research and testing.

Additionally, the team encountered issues with the frequency of the RFID tags which required the user to be at most two inches from the crosswalk platform. The visually impaired individual must set the stick on the podium with nearly complete contact, if not, the RFID reader would give faulty data. Alongside this obstacle, the focus of the ultrasonic sensor would occasionally drift in an obscured direction. To combat this issue, the team decided to replace velcro straps with an L-bracket to tightly grasp the ultrasonic sensors and not skew the output. When integrating the ultrasonic sensor, the Adafruit Feather Huzzah only supplies 3.3v, which was another reason for a skew in the readings. Methods such as modifying the current sensor to run off 3.3 volts and using a new sensor for 3.3 volts were ineffective. Connecting the sensor to the high voltage pin gave readings and showed no signs of error at all.

The MPU 6050 gyro sensor that was used does not measure the instantaneous angle, rather, it measures the delta in the start and end angle, therefore it goes back to zero after no motion. This implies that the gyro is insufficient for proper measurements from the start to the end of the crosswalk, which has the hazardous risk of sending false outputs to the individual. To fix this, the team used a LIS3DH instead. Unlike the MPU 6050, this measured the instantaneous angle.

CoCA needs modifications to a normal crosswalk setup to be effective in the real world. First, to get the RFID functioning, tags must be placed under the crosswalk pads, to locate

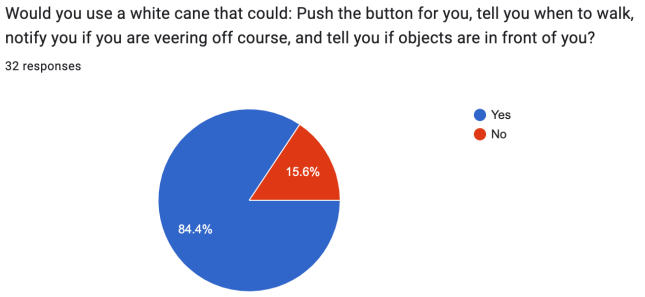

Fig. 18: Fourth survey question

the crosswalks and get authorization. To inform people about a crosswalk's status, there needs to be a connection between the stick and the controller box of that crosswalk. This modification to the controller box must allow a stable connection with the server. However, the gyro sensor and the ultrasonic sensors have both been tested, and modified to be consistent. This ensures that CoCA would still play a pivotal part in the user's safety even in the chance that the current crosswalks do not get modified.

After talking to multiple VDOT (Virginia Department of Transportation) employees, the team received feedback on the set process to integrate the sticks within an existing crosswalk. A crosswalk controller board is needed for tampering to find an efficient way to connect to CoCA, which can be accomplished through VDOT electric shops with open source equipment. After an integration with CoCA is tested and works, the idea must approach the VDOT Central Office Traffic Engineering Department, which will evaluate and test our product in certain controlled environments. The VDOT employees also explained more about their limited used APS (Accessible Pedestrian Signals) and how it functions. To activate APS, there is an infrared sensor that has audio cues to notify the visually impaired user about the outgoing request, and authorization to cross, etc. On the other side of the crosswalk, a locator tone will play to guide the user in which direction the crosswalk is. Additionally, APS has sound pollution and may disturb people going about their day-to-day lives.

Other than APS, multiple solutions are aimed at visually impaired people when navigating around the streets. Many of these solutions focus on a cane, focusing on an ultrasonic sensor. The WeWalk stick, for example, incorporates voice activation into an ultrasonic sensor that gives the visually impaired user the distance they are from an object through vibrations. Although this solution is innovative, it has no coherence with road intersections and the voice activation aspect can easily be tampered with street noises. The CoCA team examined our product alongside other products with the same motive, and with a few of our technical mistakes to feedback the CoCA team have received from others, there is room for improvement in which this product can be taken to the next level.

V. Conclusions

The CoCA or Connected Crosswalk Assistance Smart Walking Stick is a solution to solve the issue of visually impaired pedestrian accidents. The product acts as a remote activation device for crosswalks, warns the user if an object is nearby,

and alerts the user if they are veering off of the crosswalk path. The various tests conducted regarding each component of the walking cane allowed data and feedback to be collected. Thus, improvements were made to the CoCA walking stick, and technical errors allowed more solutions to be developed.

Future technologies introduced to the market mean that updates to the CoCA smart cane will be necessary. Most importantly, smaller technologies are required to be implemented in the walking cane as this will allow for the compactness of all components. Additionally, smaller devices will help with the functionality of the product. The first step that the CoCA team would like to take to improve the walking cane is to replace the ultrasonic and gyro sensors with a camera that incorporates a Computer Vision (CV) algorithm and uses Lidar technology. This camera will more efficiently capture the user's surroundings and ensure that the user remains safe while walking on the crosswalk and sidewalks.

Additionally, the team would like to adapt current wheelchairs with abilities similar to the CoCA white cane to ensure that individuals who are unable to walk have access to these technologies and safety features. Finally, the team will get the walking cane tested and patented in an environment controlled by the Virginia Department of Transportation (VDOT). Getting this product tested is an important step in adapting the current crosswalk cabinets as well as working with visually impaired individuals to make their lives better.

REFERENCES

[1] "Blindness and vision impairment," 2021.

[2] F. Hutmacher, "Why Is There So Much More Research on Vision Than on Any Other Sensory Modality?" *Frontiers in Psychology*, 2019.

[3] "Blindness and Vision Loss," *MedlinePlus: National Library of Medicine (US)*, 2020.

[4] "Blindness Statistics," *National Federation of the Visually Impaired*, 2019.

[1] [2] [3] [4]