Wearable SONAR for the Blind

Tarun Thathvik
thathvik@nyu.edu

Smrithi Thudi
srt381@nyu.edu

Vedant Desai
vbd223@nyu.edu

Abstract

In the spirit of helping the blind, a wearable device was built around the functionality of a traditional cane and potentially replace it. This device vibrates based on the feedback from the SONAR to perceive the distance of the objects in the surrounding. Several calibration features were incorporated for user convenience. This device runs on an 8-core Parallax Propeller micro-controller.

1 Inspiration

Hugh Herr, a famous American rock climber who has shattered the limitations of his disabilities, is a strong believer of the philosophy that technology could help disabled persons to live a normal life. In one of his TED talks, Herr said "Humans are not disabled. A person can never be broken. Our built environment, our technologies, is broken and disabled. We the people need not accept our limitations, but can transfer disability through technological innovation". These were not just words but the ideology he lived by; today he uses Prosthetic legs and claims to live a normal life. So yes, technology can not only improve the quality of human life but also neutralize human disability; and we aim to tackle the visual impairment with simple SONAR technology and a controlled vibration to perception. This could potentially replace a cane.

2 Research and Survey

In order to make a device that will be very accessible to the blind, information was gathered during a survey done by Vedant Desai in the course of his undergrad study on blind people and their daily life and problems. It was found that the most used assistive device, cane comes with a lot of limitations. Amongst these limitations, the following few were addressed in this project:

- **Fixed Length of the cane:** When open, canes are long and take up a lot of space when used, and make it really uncomfortable sometimes in narrow spaces. Moreover, this rigid structure makes it impossible to reduce the length of the cane as per the users desire.
- **Need to hold a cane:** It is needless to say that using a cane requires one to hold it with a hand, and this restricts them from using this hand fully and normally.

For a bio-inspired solution, the case of bats was studied, to implement the age old technology SONAR for this device. An Ultrasonic sensor was used for this. To convert depth perception from the sonar into an actuation that can be perceived by the blind, a sensation of touch and sound were considered. However, using sound would require the user to use headphones which again restricts them hearing the other sounds around them, since it is amongst one of the most used sensory feedback to perceive their surrounding. So a method of actuation through touch was the only way to go, as there is a large surface area of sensory reception for touch throughout a human body, therefore vibration was chosen for this purpose. An increase in vibration to indicate an object getting closer is quite intuitive for perception.

Additionally, during the same survey it was found that a blind person is very well aware of their personal space. This was useful information which was needed to implement a calibration mode into the device.

3 Electronics

For this project, to meet the requirements, we had used a **Parallax Propeller** as the micro-controller. The peripherals used for the project are listed below:

1. Ultrasonic Sensor [HC-SR04] [SONAR]
2. Vibration Motor [A00000117]
3. Potentiometer [Generic]
4. Piezo Buzzer [Generic]
5. Push Buttons [x2] [Generic]

4 Implementation

Since the goal of this project is to assist a visually impaired person to get a better sense of their surroundings by identifying the distance of the surrounding objects using an ultrasonic sensor and converting this data into a perceivable output, that is vibration. This whole set up was placed on a glove, as shown in the fig(1), for wearability.

Ultrasonic Sensor, one of the most common distance sensing sensors out there, emits short high frequency (ultrasonic waves) at regular intervals, and the data received will have travelled up to the object, reflected back to the sensor. From the time taken to receive the data after transmission can help us find the distance of the object from the sensor.

A Vibration motor will be used to implement the physical actuation, for perception. The closer the object, higher the intensity of the vibration. The vibration motor used for this project has a voltage rating of 3V, so the precautions of capping the voltage output from the pin was done in the code. Theoretically, after a minimum threshold, a vibration motor should have a linear relation between the voltage input and the intensity of the vibration, however, the perceivable change in vibration intensity, for the vibration motor used in the model, ranged within 0.8V and 1.3V. For this reason, the operation of the vibration motor was restricted to this range. With a different Vibration Motor, this range will be different, however the maximum voltage output from a Parallax Propeller GPIO pin is 3.3V, so the maximum vibration is limited to 3.3V even though a certain vibration motor is capable of greater vibration, in this particular circuitry.

The range of the ultrasonic sensor HC-SR04 is 2-400cm, but for usability only a certain range with the threshold for maximum distance is taken. This threshold can be changed via a special mode called, **Distance Calibration Mode**. Similarly, the Maximum intensity of the vibration motor can also be changed by the user through **Vibration Calibration Mode**. The functionality of these **Calibration modes** is explained below.

1. **Distance Calibration Mode**
   This feature was added in the light that blind people are aware of their personal space, and allows them to change the maximum distance observable.
   (a) **Activating the mode**
   To activate the mode, the user will need to *press the distance button until it beeps*.

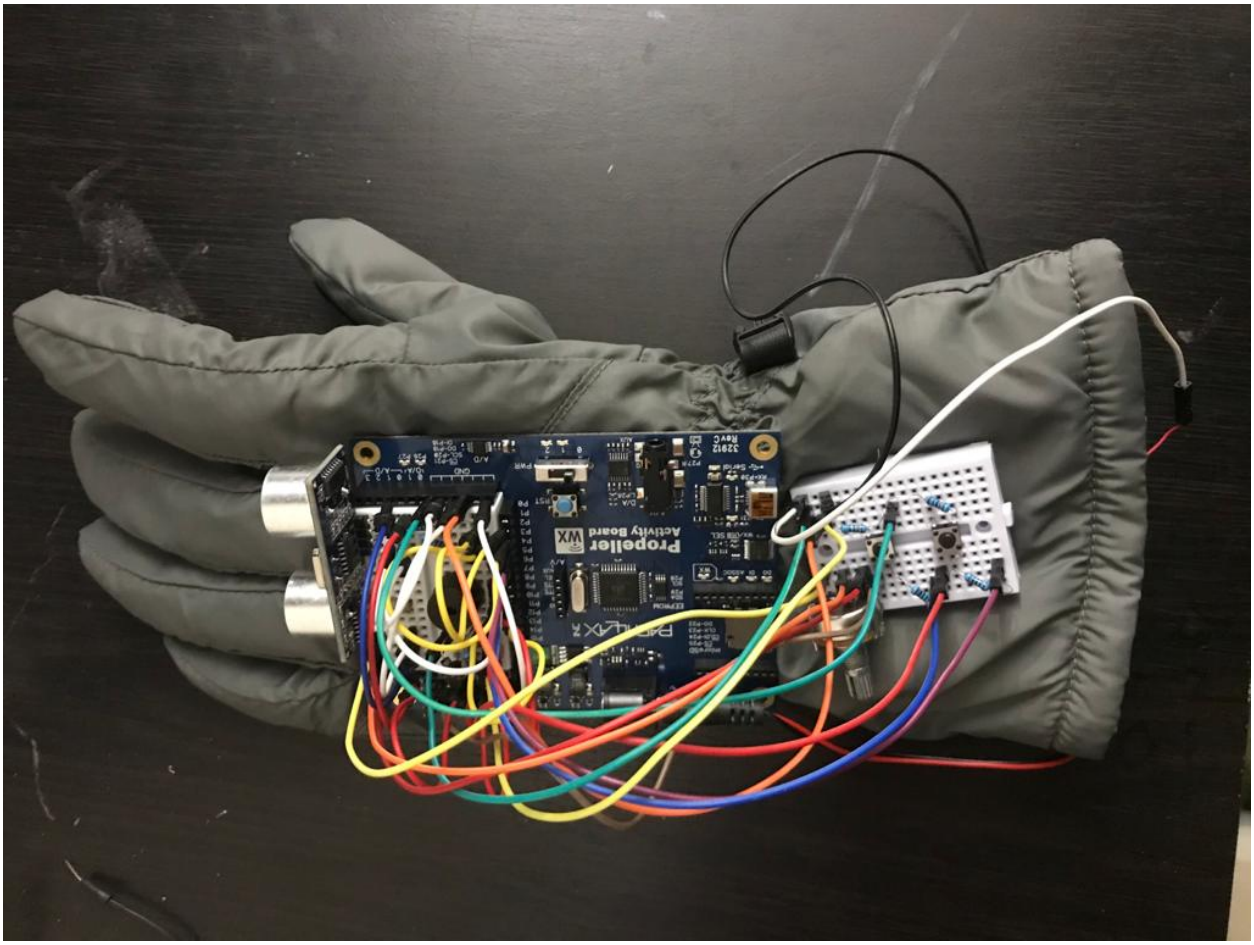

Figure 1: Photo of the product

(b) **Calibration**
The user will need to stand a comfortable distance[1] away from a wall, towards the end, this distance will be calibrated as the maximum distance observable. Once the mode is activated, the user will need to turn the potentiometer until a buzzer noise is heard, and then *press the distance button to set the distance*. *To cancel* the calibration, at any point before pressing the calibration button, the user will need to *press the vibration button*. At the tone of the buzzer, the device is out of this mode.

2. **Vibration Calibration Mode**
The user is given the option to change the intensity of Vibration felt while operating the device.

(a) **Activating the mode**
To activate the mode, the user will need to *press the vibration button until it beeps*.

(b) **Calibration**
Once the mode is activated, the user will need to turn the potentiometer proportion-

[1]The maximum distance cannot be greater than 300cm, due to the soft limits set

ally the user can feel the intensity of vibration change[2]; at the desired intensity, *the vibration button needs to be pressed to set* maximum intensity of vibration. *To cancel* the configuration, at any point before pressing the vibration configuration button, the user will need to *press the distance button*. At the tone of the buzzer, the device is out of this mode.

Exploiting EEPROM on the Parallax Propeller, the device does not require calibration every time it is powered down. The values set during both the calibration modes will be stored in the EEPROM and is reused all the time.

5 Circuit

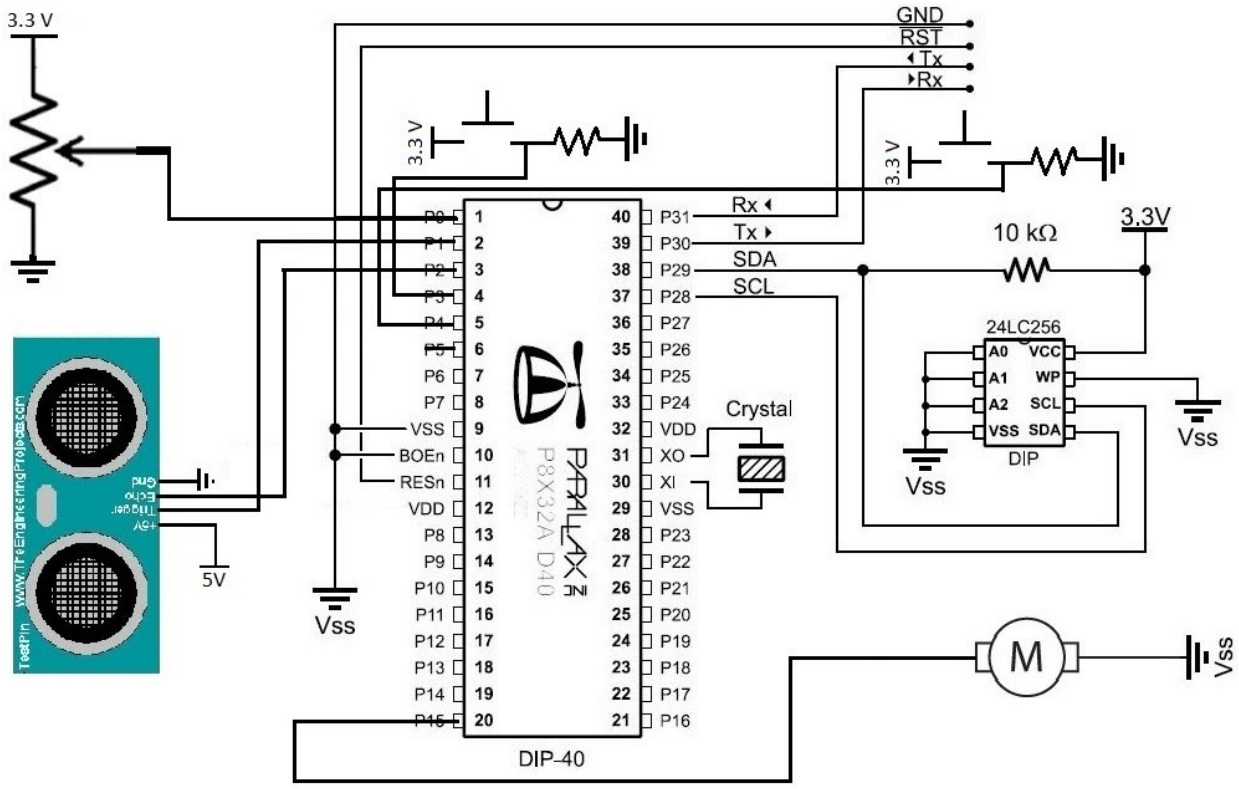

Figure 2: Circuit Diagram

6 Code

```
1
2 #include "simpletools.h"                      // Include simple tools
3 #include "adcDCpropab.h"                      // Include adcDCpropab
4 i2c *eeBus;
5 #define Trig 1                               //Trigger for Ultrasonic sensor
```

[2]Depending on the Vibration Motor, it maximum vibration cannot be changed past a point, so for usability, it is suggested to use it at less than the maximum perceivable vibration

```
#define Echo 2                                  //Echo pin for Ultrasonic sensor
#define Pot  0                                  //Potentiometer input

#define VIB 9                                   //Vibration motor output
#define buzz 15                                 //Buzzer output

#define cfg_but 3                               //Configure mode button (push button)
#define vib_but 4                               //Vibration configuation mode button (
    push button)

int ping(int TrigPin, int EchoPin);
int ping_cm(int TrigPin,int EchoPin);
int ping_inches(int TrigPin,int EchoPin);
int map(int var,int v_l1,int v_l2,int set_l1,int set_l2);
int vibration_intensity(int dist, int min_d,int  max_d);

void ultrasonicDistance(void *pauseTime);
void potentiometerReading(void *par);
void button_monitor();
void config_vibrate();
void configure();
void main_fun();
void button_m();
void firstTime();
static volatile int distance,potentiometer,pot_map_max_dist, pot_map_max_vib, max_dist
    , vib_max, config_button, vibrate_button;
static volatile int cog_ultra, cog_pot_met, cog_buttons, cog_vib, cog_cnfg, cog_main,
    cog_button;
static volatile float step_dist;
unsigned int stack1[40+25];
unsigned int stack2[40+25];
unsigned int stack3[40+25];
unsigned int stack4[40+25];
unsigned int stack5[40+25];
unsigned int stack6[40+25];
unsigned int stack7[40+25];
static volatile int eeAddr = 0b1010000;
static volatile int memAddr_max_dist = 32768;
static volatile int memAddr_vib_max = 32773;
static volatile int memAddr_firstTime = 32778;
int check_val = 0;
int main()                                      // Main function
{
  eeBus = i2c_newbus(28,  29,   0);             // Set up I2C bus, get bus ID
  adc_init(21, 20, 19, 18);
  int pauseT = 60;
  i2c_in(eeBus,eeAddr,memAddr_firstTime,2,(char*) &check_val,4);
  if (check_val!=1)
  {
    firstTime();
  }

  i2c_in(eeBus, eeAddr, memAddr_max_dist,2,(char*) &max_dist,4);
  i2c_in(eeBus, eeAddr, memAddr_vib_max,2,(char*) &vib_max,4);

  cog_ultra = cogstart(&ultrasonicDistance, (void*)pauseT, stack1, sizeof(stack1));
  cog_pot_met = cogstart(&potentiometerReading, NULL, stack2, sizeof(stack2));
  cog_buttons = cogstart(&button_monitor, NULL, stack3, sizeof(stack3));
  cog_main = cogstart(&main_fun, NULL, stack6, sizeof(stack6));
```

```
62
63    while(1)                                    // Loop repeats indefinitely
64    {
65      putChar(HOME);                            // Cursor -> top-left "home"
66      print("potentiometer value = %d%c\n",potentiometer,CLREOL);
67      print("distance measured = %d%c\n",distance,CLREOL);
68      print("max dist set= %d%c\n",max_dist,CLREOL);
69      print("max vib set= %d%c\n",vib_max,CLREOL);
70      print("Distance Button = %d%c\n",config_button,CLREOL);
71      print("Vibrate Button = %d%c\n", vibrate_button,CLREOL);
72      print("Vibration = %d%c\n", map(distance, 2, max_dist,vib_max,67),CLREOL);
73      pause(100);                               // Wait 1/10 s
74    }
75  }
76
77  void firstTime()
78  {
79    int val =50;
80    i2c_out(eeBus,eeAddr,memAddr_max_dist,2,(char*) &val,4);
81    while(i2c_busy(eeBus,eeAddr));
82
83    val = 150;
84    i2c_out(eeBus,eeAddr,memAddr_vib_max,2,(char*) &val,4);
85    while(i2c_busy(eeBus,eeAddr));
86
87    val = 1;
88    i2c_out(eeBus,eeAddr,memAddr_firstTime,2,(char*) &val,4);
89    while(i2c_busy(eeBus,eeAddr));
90  }
91
92  void main_fun(){
93    while(1){
94      if(distance <= max_dist){
95        dac_ctr(VIB, 0, map(distance, 2, max_dist,vib_max,67));
96      }
97      else{
98        dac_ctr(VIB, 0, 0);
99      }
100   }
101 }
102
103
104
105 void ultrasonicDistance(void *pauseTime)
106 {
107   int pauseT = (int)pauseTime;
108   while(1)
109   {
110     distance = ping_cm(Trig,Echo);
111     pause(pauseT);
112   }
113 }
114
115 int ping(int TrigPin,int EchoPin)
116 {
117   low(TrigPin);
118   pulse_out(TrigPin, 10);
119   return pulse_in(EchoPin, 1);
120 }
```

```
121
122 int ping_cm(int TrigPin,int EchoPin)
123 {
124   long tEcho = ping(TrigPin,EchoPin);
125   int cmDist = tEcho / 58.0;
126   return cmDist;
127 }
128
129
130
131
132
133 int map(int var, int v_l1, int v_l2, int set_l1, int set_l2)
134 {
135   float y;
136   float a = 1.0* (set_l1 - set_l2)/(v_l1 - v_l2);
137   float b = 1.0* (set_l1 - a*v_l1);
138   y = a* var + b;
139   return (int)y;
140 }
141 int vibration_intensity(int dist, int min_d,int  max_d)
142 {
143   step_dist = (max_d - min_d)/3.0;
144   int  intensity = 80;
145   if (dist*1.0 <= min_d*1.0+step_dist)
146   {
147     intensity = 150;
148   }
149   if ( dist*1.0 <= min_d*1.0+2.0*step_dist && dist*1.0 > min_d*1.0+step_dist)
150   {
151     intensity = 77;
152   }
153
154   if (dist*1.0 <= max_d && dist*1.0 > min_d*1.0+2.0*step_dist)
155   {
156     intensity = 68;
157   }
158   return intensity;
159
160 }
161
162
163 void potentiometerReading(void *par)
164 {
165   while(1)                                    // Loop repeats indefinitely
166   {
167     potentiometer = adc_in(Pot);
168     pot_map_max_dist = map(potentiometer, 0,2650 ,30 ,300);
169     pot_map_max_vib = map(potentiometer, 0, 2650, 80, 230);
170     pause(10);
171   }
172 }
173
174
175
176 void configure(){
177
178   cogstop(cog_main);
179   dac_ctr(VIB, 0, 0);
```

```
  freqout(buzz, 500, 3000);
  int flag = 0;
  int i = 0;
  while(config_button == 1);
  int count = 0;
  while(1){
    if( (distance < pot_map_max_dist + 3) && (distance > pot_map_max_dist - 3))
    {
      flag = pot_map_max_dist;
      dac_ctr(VIB, 0, 75);
      freqout(buzz, 10, 5000);
    }
    else
    {
      dac_ctr(VIB, 0, 0);
    }
    if(config_button == 1)
    {
      max_dist = flag;
      freqout(buzz, 500, 3000);
      i2c_out(eeBus,eeAddr,memAddr_max_dist,2,(char*)&max_dist,4);
      while(i2c_busy(eeBus,eeAddr));
      freqout(buzz, 500, 3000);
      break;
    }
    if(vibrate_button == 1)
    {
      freqout(buzz, 500, 4000);
    }
   }

  cog_main = cogstart(&main_fun, NULL, stack6, sizeof(stack6));
  cogstop(cog_cnfg);
}



void config_vibrate(){
  cogstop(cog_main);
  dac_ctr(VIB, 0, 0);
  freqout(buzz, 500, 3000);
  int flag = 0;
  int i = 0;
  while(vibrate_button == 1);
  int count = 0;
  while(1){


    dac_ctr(VIB, 0, pot_map_max_vib);


    if(vibrate_button == 1)
    {
      vib_max = pot_map_max_vib;
      freqout(buzz, 500, 3000);
      i2c_out(eeBus,eeAddr,memAddr_vib_max,2,(char*) &vib_max,4);
      while(i2c_busy(eeBus,eeAddr));
      freqout(buzz, 500, 3000);

```

```
      break;
    }
    if(config_button == 1 ) {
      freqout(buzz, 500, 4000);
      break;
    }
   }

  cog_main = cogstart(&main_fun, NULL, stack6, sizeof(stack6));
  cogstop(cog_cnfg);
}



void button_m(){
  while(1){
  config_button = input(cfg_but);
  vibrate_button = input(vib_but);
  }
}



void button_monitor(){
  cog_button = cogstart(&button_m, NULL, stack7, sizeof(stack7));

  while(1){
  if(config_button == 1){
    for(int i = 0; i <= 100; i++){
     pause(10);
     if(config_button != 1){
      break;
     }
    }
    if(config_button==1){
      cog_cnfg = cogstart(&configure, NULL, stack4, sizeof(stack4));
    }
  }
   if(vibrate_button == 1){
    for(int i = 0; i <= 100; i++){
     pause(10);
     if(vibrate_button != 1){
      break;
     }
    }
    if(vibrate_button==1){
      cog_vib = cogstart(&config_vibrate, NULL, stack5, sizeof(stack5));
    }
  }

  }
}
```

7 Budget

The Budget is clearly summarized in the table(1).

Label #	Components	Cost
1	Parallax Propeller Activity Board	$ 79.0
2	Ultrasonic Sensor [HC-SR04]	$ 4.20
3	Vibration Motors [A00000117]	$ 12.99
4	Piezo Buzzer [A14121600UX0159]	$ 2.19
5	Potentiometer, push button, and other components	$ 10.00
	Total	$ 107.28

Table 1: Bill of Materials

8 Conclusion and Future Work

During this project, the cog functionality of the Propeller was explored to a great extent by utilizing seven cogs to perform the said task. Using I^2C protocol, the data from the RAM was stored in the EEPROM on the Propeller to remember the personalized calibration settings. Several hardware limitations like the perceivable vibration for a range of voltage input was discovered first hand, and precautions to work around them. The framework to replace the vibration motor on the device is already set, in the future iterations, a vibration motor with a more diverse range can be used to better perceive the distances, and potentially increase the comfortable distance range.