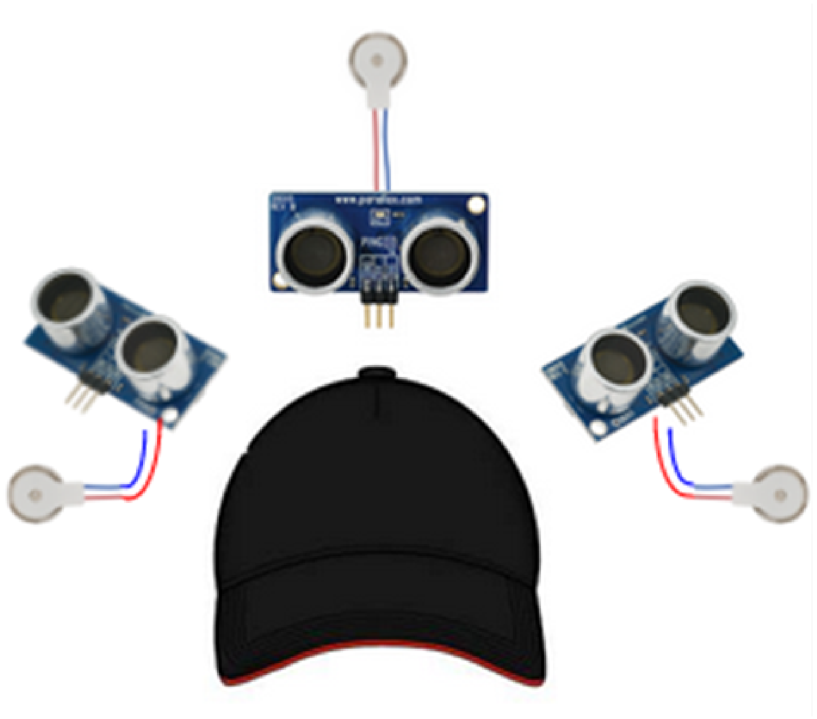
ITMT 492 Embedded System final project

Clothing Based Proximity Sensors for the Visually Impaired

Youjung Chio
Yelim Kim
Changyul Oh
Neil Okhandiar

Contents



1. Abstract

If you are blind, how can you avoid obstacle? There are so many obstacles in everywhere to blind people. So we want to make a sensing machine to avoid obstacles for them. It need to cheap, small, convenient in order to use real-life. So we want to make a wearable product using hat and small sensors.

In this paper, we explain components, software, hardware of our product. In components, each sensor, attached the hat in front, left, right, is composed by an Ultrasonic and vibrator. If these sensors detect obstacle, It will begin vibrate. Software operates each sensor using Arduino program. Because this product is wearable product, we had to use conductive thread and sew it through the hat to connect each component.

2. Introduction(name and purpose of device)

Our product name is Clothing Based Proximity Sensors for the Visually Impaired. This is a wearable device that we tried to develop that to help the visually impaired. This way they can dodge obstacles without relying on a stick.

Through ultrasonic waves can detect the distance between user and obstacles. And then, if some obstacle is detected the vibrator will be used to inform to user about both the direction and the distance. By putting on three vibrations and ultrasonics at other directions we can avoid to obstacle from front and both sides. And then we set different vibration power according to distance. If they close from obstacles, the power will be stronger. The next component was the wiring itself. We sowed steel thread throughout the hat instead of using wire because it better integrates with the cap.

Our expectations for our project are to help solve the problem of blind obstacle detection and to extend the functionality of wearable devices. We also emphasize the use of environmental energy from the sun as it is more sustainable in the long run.

3. Components

Components	The number
Arduino micro	1
Ultrasonic range detection sensor	3
Vibrating mini motor disc	3
Flexible solar panel	1
Solar Lithium charger	1
3.7V Lithium Battery	1
Power Boost	1
Touch Control of electronic device	1
Stainless Conductive Thread	2

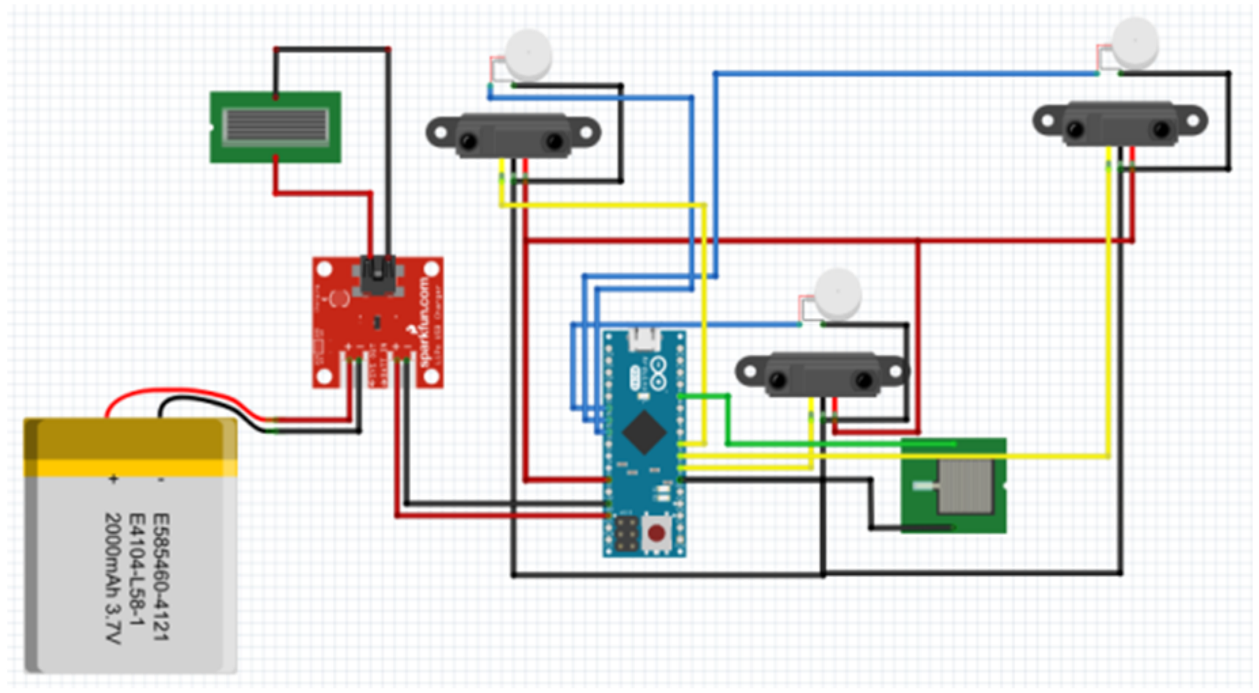
We are making a wearable product, so we need smaller than the Arduino Uno. We first planned on using the Flora, which was designed for sowing into clothes, but this lacked the necessary number of digital pins we required. So we used the Arduino Micro. It has both a small size and the same number of pins as the Uno, so it was perfect for our purposes.

In input/output sensors, we used ultrasonic range detection sensor. It can calculate the distance by echo pulse width, detecting the reflected ultrasonic. Also we used mini vibrator. When the Ultrasonic detects a distance less than 70 cm, this motor will begin vibrate.

In batteries, flexible solar panel use light energy from the sun. So when just sunlight is available, it will be used to charge the battery for our system. And we used charger, 3.7 voltage battery, power boost. Arduino needs 5 voltage, but we just had 3.7 battery. So we have to use power booster.

And we used touch panel and thread. The thread is made with conductive stainless thread. It functions just like the normal wire, so we sowed it through the hat to connect the hardware with the Arduino.

4. Circuit



5. The code
The arduino logic for this program fundamentally depends on <SimpleTimer> library. The SimpleTimer library allows us to implement simple timer functions, which repeatedly run functions every x milliseconds through interrupts. This allows us to poll various devices throughout the duration of the arduino's runtime, in a simple and reliable manner, and then operate based on the data received, without forcing the arduino to wait for polling to complete before executing further code.

Another significant operation of the arduino code is the averaging function. In order to account for random noise that might incur during normal operation, we created a 2D array of size [3][20], storing the last twenty distances received by the ping sensor, for each sensor. Using a counter to keep track of the last data added, every time the ping is polled, we add the distance to the array and increment the counter. When counter exceeds the array size (20), then it starts at the beginning and simply replaces the old distance. We then calculate the new average using a simple for loop, that just iterates through the array and totals the values, then divides it by the size of the array.

Then, if the average distance is less than 70 cm (our soft-coded max distance detected), we’ll want to turn on the vibrator at a certain pulse rate to reflect the distance detected.

The pulsing system is where the greatest complexity with this program arises. Because the vibrator can only be turned on or off, in order to vary how strong the vibrations are in relation to the distance detected, we had to construct the vibrator function as a kind of PWM function. On top of this, we wanted a step-function to relate the pulse to distance, incrementing or decrementing every 10 cm detected. To do so, we used a SimpleTimer that goes off every x milliseconds, with x being the shift in pulse frequency. The current code is set to 200 milliseconds, but this needs to be modified with further testing. As it stands, the user can no longer feel the pulse after 30cm.

The SimpleTimer calls a function which attempts to set off all the vibrators. The code for pulsing the vibrator works by having a “distance level” set by ping, when the average distance is less than 70. The distance level is simply averageDistance (cm) % 10 (cm), where 10 is the distance between step changes. The vibrators are each given a variable that begins at 0, and increments by 1 every time they are called by the SimpleTimer. If the variable is 0, then the vibrator turns on. If the variable is equal to, or greater than the assigned “distance level”, the variable resets to 0.

Thus, the vibrator is turned on for 200ms, then paused for 200X milliseconds, where X = averageDistance % 10. The pulses might look something like
___IIII________IIII_____________IIII________________IIII___
Where IIII is the vibrator being on, when the user gets further from a detected object. The pulse itself always remains the same, while the pause varies based on distance.

The touchpad also works by a SimpleTimer. It is simply polled by SimpleTimer every 250ms. The touchpad just checks how much pressure is being applied to it. With a 10k resistor, a returned value of >200 implies that, at the very least, the pressure of a finger has been applied. Random noise generally remains under ~50. By only polling every 250ms, we handle both double-detection (our press-time being much longer than the speed at which arduino can poll, it is possible to set off the device multiple times accidentally) as well as relatively long-presses (but the code doesn’t handle presses that last for more 250ms “beautifully”, repeatedly switching the relevant from true to false, and vice versa, until the user removes his finger.)

The touchpad, when detected as pressed, just switches a boolean variable. The boolean value, if set to false, causes the arduino to simply skip over the ping-detection code, and turn off any currently active vibrators (but the touchpad is still polled every 250ms!). When set back to true, the arduino resumes normal activity.

The ping detection itself operates on the default code given by adafruit. It turns the ping off, then on, then waits, then off and on once more. It now checks the time between sending and receiving the ultrasonic pulse, and calculates the cm based on that time. Finally, it returns this value, and everything else begins.

The last notable aspect of the code is that nearly every global variable was instantiated as an array of size [3], so that we could maintain different values for each of the three different Ping | Vibrator pairs in use.

NOTE: The function pulseIn is a standard library function used by the ping to get the time between sending and receiving a pulse. This function, copied from adafruit, takes the (pin, HIGH). However, this function is overloaded. The overloaded function takes (pin, HIgh, timeout). If you use the first function, timeout is set to 1,000,000 microseconds. (1 second) This means that, if nothing is being detected, arduino will wait a full second before moving on. For our purposes, timeout would be sufficient at 4060 microseconds, which would mean a detection range of 70. This usually doesn't matter all that much, but when our Pings were failing to work properly, and detecting nothing, our code began to run obscenely slow due to this. Our not being aware of this property gave us a great deal of unnecessary stress.

6. Operation Principle

First we considered the size of the Arduino to choose. Because our product is a wearable item, we needed a small, nonintrusive chip. This is why we decided on the Arduino Micro as our controller.

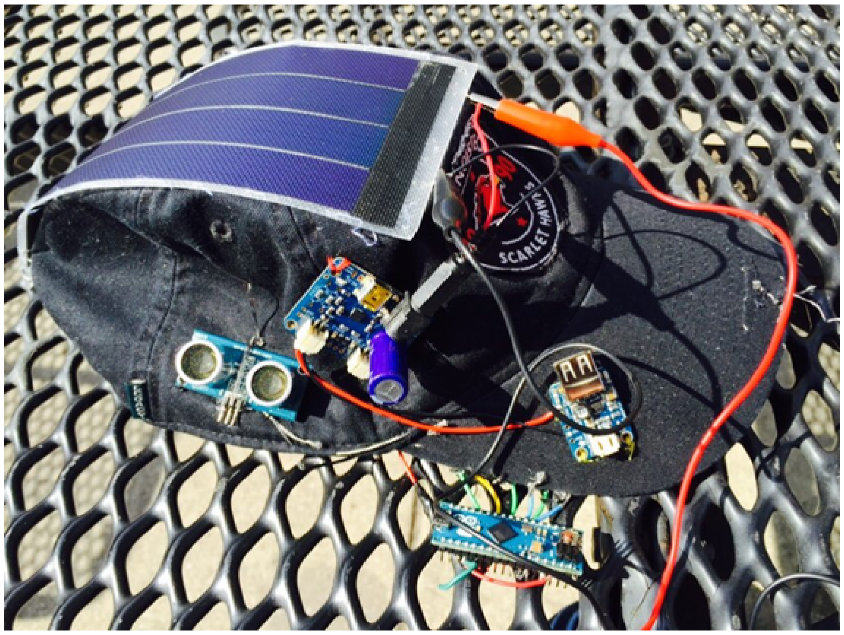
And then we attached 3 Vibrations and Ultrasonics at 3 different directions. If in any direction there is a obstacle, you will be informed and can then avoid it.

When the user meets an obstacle, the ultrasonic sensor will detect its distance. We then perform some mathematical operations to transform the distance in milliseconds, and use this as the frequency for our vibrator. As the user comes closer to the obstacle, the vibrator pulses at a faster rate, increasing the strength felt by the user.

If the user wishes to turn off the cap, such as when talking to another person, he can press the pressure embedded into the front of the cap. This will shut down all functionality of the hat. Another press, and the hat will resume instantly.

Finally When you are outside at daytime, you can charge the battery via solar panel which is placed on top of a cap. The solar panel outputs up to 8 volts of electricity, which must be lowered to 3.7 volts to safely charge the battery. Then, power is drawn from the battery to the arduino. However, we require 5 volts to power the ultrasonic sensors, so a step-up device was required between the battery and the arduino. Thus, we go from 8V to 3.7V to 5V.

7. Result



This product what we made can detect obstacles within 70cm as program code. This distance can be modified simply by setting different distance as user's need. It can charge battery with solar energy or usb cable. User can recognize surrounded object by sensing vibrations with 0.5 sec delay(Ultrasonic bounce time).

8. Conclusion(Problems in building it, operational issues, did it work)

The hat what we made works properly with all components, 3-side ultrasonic sensor, vibrators which have work frequency as distance, and solar panel to charge battery. However, we had one problem that we couldn't make it exactly same as what we first intended because of thread problem. We had iron thread as electric wire. The threads are too sophisticated to put it in small space. When we put all components in hat with iron thread, it showed kind of short problem. So we decided to make one side with normal electric cables. We have to make sure about electric line is compact and efficient. The thing we left is optimize electronic components, and make it have water resist for washing as it is wearable product.