**Harris Haptic Device:**

**Navigation for the Visually Impaired**

Villanova University

Department of Electrical and Computer Engineering

ECE 4970-001

Technical Executive Summary

By Savannah Benbrook, Tim McCormick, and Brian Sakhuja

Faculty Advisor: Professor Edmond Dougherty

12 December 2016

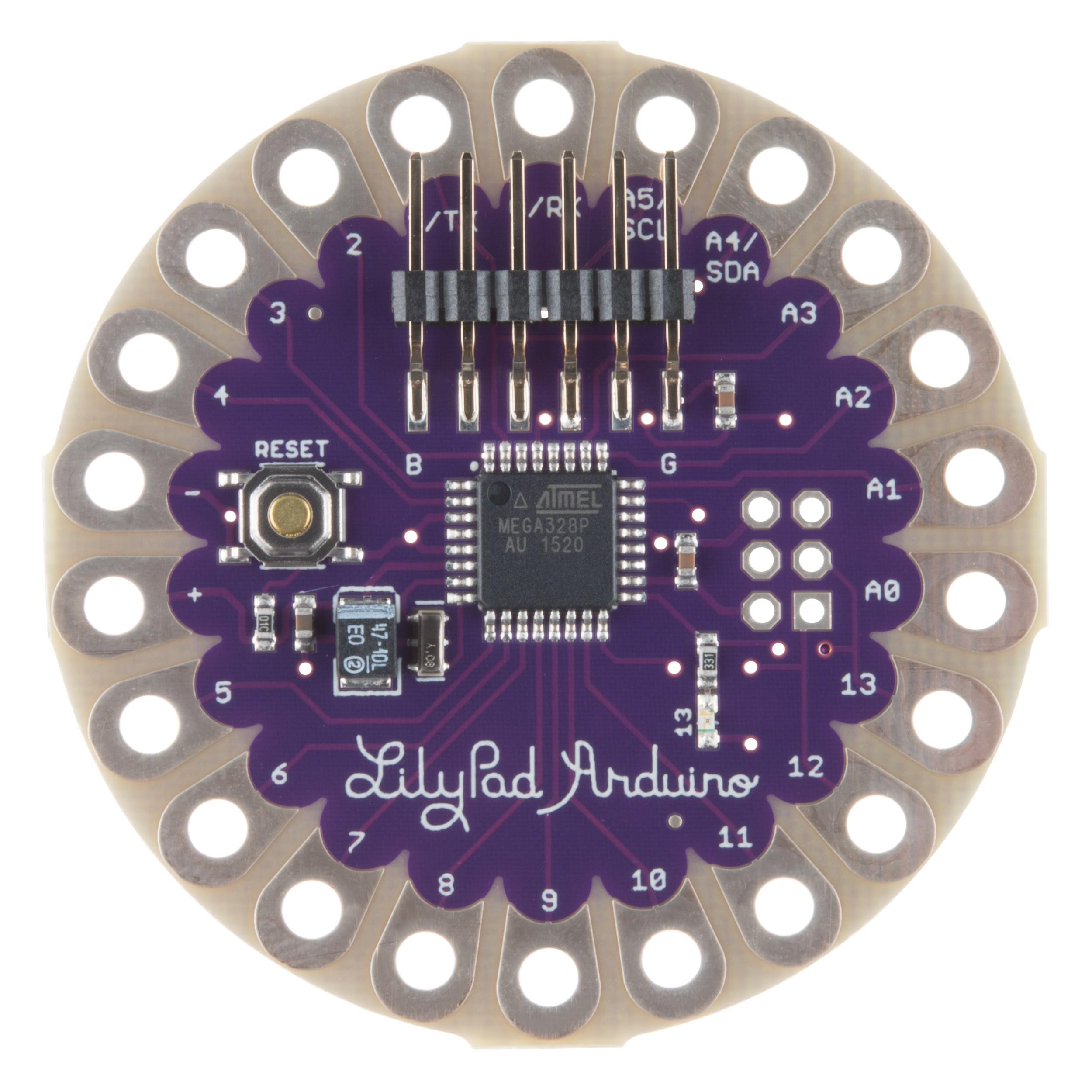
**Abstract/Overview**

This technical executive summary contains the technical specifications and details for the Harris Haptic Device: Navigation for the Visually Impaired project. The purpose of this project is to design a navigational device for the visually impaired using haptic feedback. Functional and technical specifications for the device are detailed, including a block diagram, a description of the the behavior of the code, and a list of the products used in the final implementation of the navigation device. The device provides the following navigation assistance: detection of objects in direct line of sight at waist height or above in the direction of the user’s head. The final, wearable design is implemented on a hat using an Arduino LilyPad, two vibration motors, one ultrasonic sensor, a battery, and a switch to turn the device on and off.

**Specifications**

*Functional Specifications*

Our device is implemented as a hat. As per the Harris specifications, the device detects objects in the direct line of sight of the user. The two motors pulse more rapidly the closer an object is, cutting off at a distance of under five inches from the user. Similarly, the two motors vibrate less frequently the further away the object is from the user, with the maximum distance set to fifteen feet. An Arduino Lilypad was used to control two vibrating motors on the left and right side of the hat located near the temple region. The motors were controlled based on the distance readings from one MaxSonar ultrasonic sensor in the center on the brim of the hat, a 2000mAh battery, and an on/off switch to toggle between turning the device on and charging the battery with a mini USB.



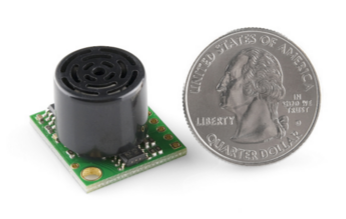
**Figure 1:** Arduino LilyPad



**Figure 2:** 2000mAh battery



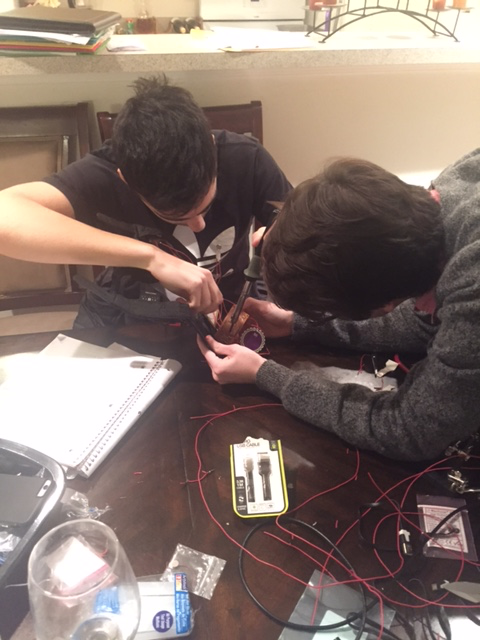
**Figure 3:** Vibration motor



**Figure 4:** Ultrasonic Proximity Sensor



**Figure 5:** Final product



**Figure 6:** Tim and Brian soldering the breadboard



**Figure 7:** Tim and Savannah hot-glueing sensors into hat

*Block Diagram*

final flow chart.png

*Technical Description of System Operation*

The user flips the switch on the hat to power the device using the battery. The LilyPad runs the code, which has the following behavior based on object proximity (which can also be seen in the block diagram above):

For objects in the range of 180”-120”, the motors are on for 100ms and off for 900ms. For objects in the range of 120”-100”, the motors are on for 100ms and off for 800ms. For objects in the range of 100”-80”, the motors are on for 100ms and off for 700ms. For objects in the range of 80”-60”, the motors are on for 100ms and off for 600ms.

For objects in the range of 60”-50”, the motors are on for 100ms and off for 500ms.

For objects in the range of 50”-40”, the motors are on for 400ms and off for 400ms.

For objects in the range of 40”-30”, the motors are on for 300ms and off for 300ms.

For objects in the range of 30”-20”, the motors are on for 250ms and off for 250ms.

For objects in the range of 20”-15”, the motors are on for 200ms and off for 200ms.

For objects in the range of 15”-10”, the motors are on for 150ms and off for 150ms.

For objects in the range of 10”-5”, the motors are on for 100ms and off for 100ms.

The closer an object is, the more frequent or intense the vibrations are. Because of the properties of the delay() function we used to create the motor pulses, the closer an object is also means that the ultrasonic sensor reads the distance more frequently. The converse is true too: the further an object is, the less frequent the sensor reads the data. This helps conserve the battery.

**Achievements**

* Uno prototype successfully worked to detect objects and relay distance through a single motor.
* ProMini didn’t work as we couldn’t successfully communicate/upload code to the device - this was likely a problem with the board.
* LilyPad was difficult to communicate with as the FTDI needed us to install a package that didn’t initially work.
* Instead of using the accelerometer to detect when the user is in motion, our team simplified to an on/off switch due to concerns the device might be powered on by the accelerometer detecting motions of the head that do not necessarily correspond to mobility that requires object detection.
* Tweaked code so that there were many different (10) distance thresholds to have different vibration patterns.