Piezoelectric Pressure Sensing Shoe Insole

ECE 445 Individual Progress Report

Team 47: Alan Lee

TA: Kyle Michal

Spring 2019

1. Introduction

Our project is a shoe device that captures foot pressure distribution data through pressure sensor pads. It is equipped with a rechargeable battery, piezoelectric generators, and a small Bluetooth enabled microcontroller to stream data periodically to a mobile app. Throughout our product design, much of it has been pair design as our team is only consisted of two. I have been active in designing all subsystems including the power, sensors, control system, and mobile app platform. Although design was mainly done in a pair, the main subsystems that I have been responsible for researching are the control system and the mobile app platform.

The control system connects to the mobile app platform as well as the sensor system. It consists of a microcontroller and a Bluetooth module that is connected to the power subsystem. The mobile app platform consists of an android app that will have displays of foot pressure data over select one hour intervals. This only interacts with the control systems microcontroller through a Bluetooth interface to receive periodic pressure data. The data is sourced from the pressure sensors attached to the control system. The overall objective that these two subsystems satisfy in conjunction is the requirement that the shoes stream dynamic foot pressure data from 0 to 120 kPa per sensor in a 50 - 60 KB data set once per hour of operation and display foot location mappings on a mobile app.

As of now, we have selected and ordered all relevant parts regarding the control system and have begun prototyping on an Arduino. Currently the main focus is to enable the Arduino to properly relay foot pressure data. This involves verifications of the sensor subsystem as well as writing the system code that our device will operate on. After this step is complete, the remaining tasks at hand for these two subsystems include:

- Implementing a streaming capability from the Arduino
- Building a mobile app for user interface
- Finalizing the code so that we may undock the ATmega328P microprocessor and mount it on a final PCB

Once these are completed, the data streaming and processing requirement will be done.

2. Design

During the original design of the product, there was difficulty in choosing an adequate microcontroller that would satisfy the compatibility of a Bluetooth module as well as allowing us to program a relatively complex program. The status of control system development has been substantial since the design document. Originally, the design used the Silicon Labs C8051F50x. It was found to be extremely complex to program this microcontroller through conventional means as it did not have a USB port. The only way to program this microcontroller was to build a significantly complex resistor circuit and send a signal to a GPIO pin. This would then flash the onboard memory to allow us to program the control system software. Therefore after this realization, we changed our design to use the ATmega328P. We found this through trying to emulate the ease of programming that the Arduino uses. To program this, we can utilize an existing Arduino as the ATmega328P is the same microcontroller that the Arduino uses.

As of now, we know how to program our control system. Currently I am working on researching how to integrate the Bluetooth functionality to stream data periodically to the mobile app. On the mobile app side, there has still been minimal development. The decision was made to use an android OS and we are currently exploring how to build a simple UI that can display our desired interface.

The method we are using to program our microcontroller is by using another Arduino UNO. We can build a proto Arduino on a breadboard and load a bootloader onto it so that we can have easier programming through a USB interface. We will first build a circuit similar to the one below.

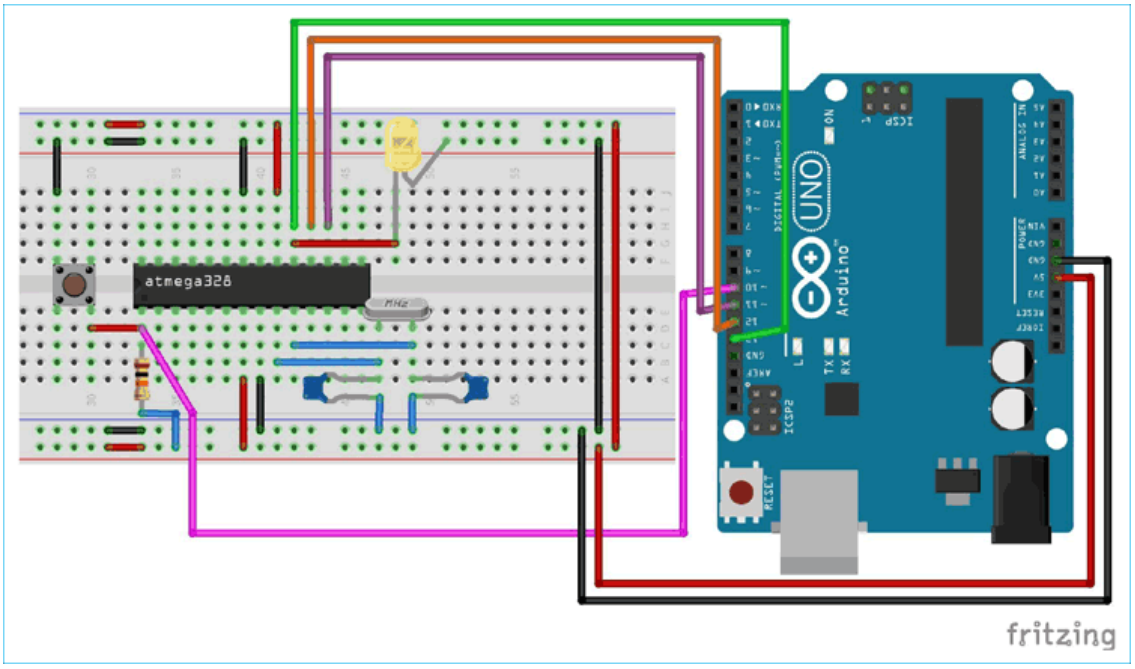

Figure 1: Microcontroller programming setup [2]

The circuit is comprised of a few simple LED's, Resistors, capacitors, and a push button. We then burn the given boilerplate Arduino bootloader code given from the Arduino foundation onto the ATmega328 chip by utilizing our existing Arduino. We can program our existing Arduino simply through a USB interface. After we have the bootloader on the ATmega328, we can then modify our circuit to look like:

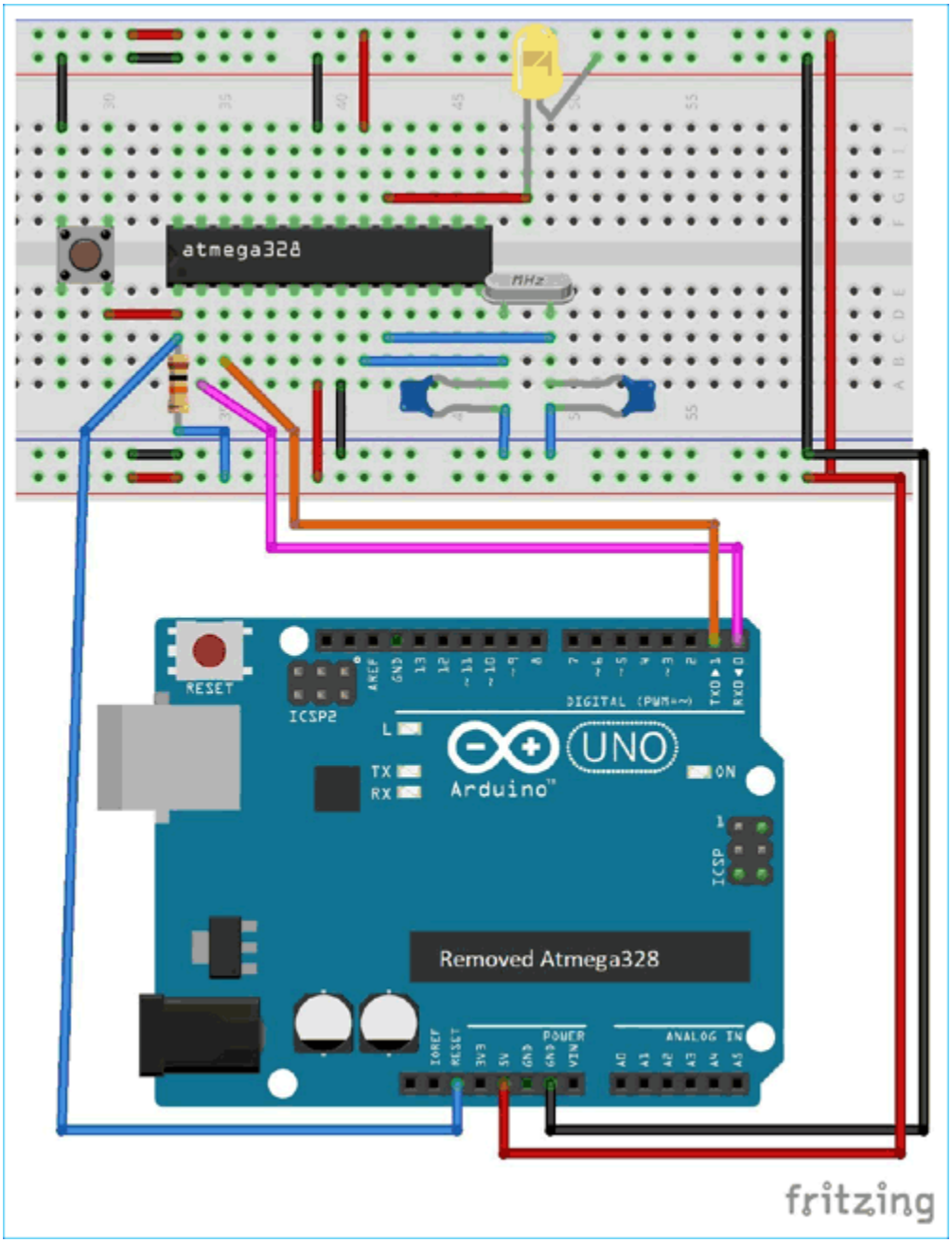

Figure 2: Bootloader programming [2]

Once we have built this circuit, we can upload any program we choose just as if it were a regular Arduino.

On the system side code, we will have pseudocode similar to the below code.

```
int Pressure;
void setup() {
  Serial.begin(9600); //Begin serial communication
}
void loop() {
  Pressure = analogRead(A0); //reading analog value from pin A0
  Serial.write(map(Pressure, 0, 1023, 0 ,255)); //Mapping the value
  delay(100);
}
```

What still remains to be fully fleshed out is the processing code so that we may display a feed of aggregated walking data. Currently the pseudocode design will take the approach of mapping serial read in values to a 2d matrix to represent a foot pressure sensor pad. Then by filling in time averaged values in row matrix order, we can get a map of pressure values. Then we can map a RGB scale the matches certain numerical values which will give us a pressure map of the area of the applicable pressure sensor. The concept design for the app will be similar as shown below.

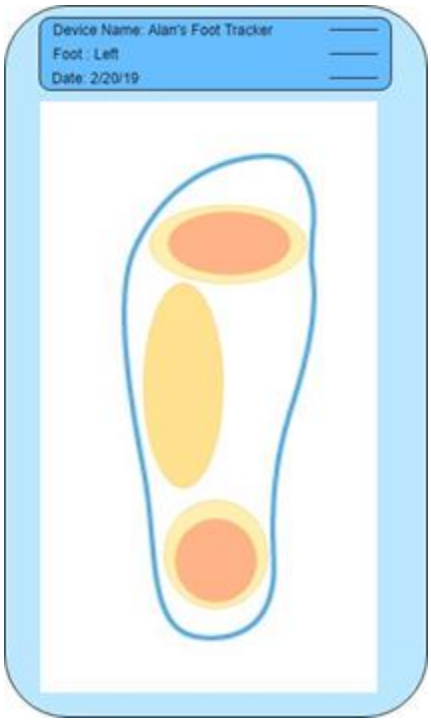

Figure 3: Mobile App GUI

As for the memory needed to store and transmit a steady stream of data it has been calculated below:

$$8\,Hours * 60\frac{mins}{hr} * 60\frac{s}{min} * 1\,\frac{sample}{s} * 2\frac{bytes}{sample} * 1000\frac{bytes}{kb}$$
$$= \sim 58KB\ required\ for\ 8\ hours\ of\ data\ storage$$

3. Verification

Mobile App Testing + Verification

The mobile app platform will serve as the main user interface for the patient/user. It will be comprised of a mobile app screen that will capture the sensor feeds and visualize it into an aggregated pressure map of the user's foot. We have not performed any of the below tests yet as the mobile app has not been written yet.

Requirement	Verification
• Must be able to connect to the Bluetooth adapter on the insole	A. Call startDiscovery() within the app and then use getAddress() to receive the MAC address associated with the discovered device. B. Initiate the connection by calling connect and check to make sure it does not throw a C. Wait for a push notification popping up indicating that the connection is successful.
• Must have a foot display for pressure distribution	A. Verify app has foot pressure UI that is displays data taken from the insole after a gather data button is pressed.
• Must display accurate pressure sensor data	A. Check for malformed data by applying pressure on each sensor individually and inspecting the display in the app. Pressure should appear in the corresponding areas with a level that matches the weight.
• Must have Android OS compatibility	A. Download the app to an actual android device B. Go through previous verification steps to ensure app works on hardware.

Microcontroller Testing + Verification

The microcontroller handles the data from the three pressure sensor feeds. It will intake these inputs and utilize the Bluetooth transceiver to stream the data to the Mobile App Platform. As we are still in the process of building our rev 1 prototype, we are prioritizing instantiating the microcontroller platform first. None of the tests below have been performed as of now.

Requirement	Verification
• Must be powered by a 1.8v - 5.25v +-15%, 19mA +-10mA power supply for 8 hours	A. Verify figures on data sheet. B. Must see the microcontroller turn on while measuring voltage input is within expected range. C. Let the device run for 8 hours. Check back to ensure there is still power at this point by checking battery voltage.
• Must have sensor data input ports and support the I2C protocol for communication with the Bluetooth Transceiver	A. Verify the microcontroller has sensor inputs on the datasheet/website.

Bluetooth Transceiver

The transceiver will be the component that allows the microcontroller to communicate with the mobile app platform. The Bluetooth transceiver is chosen with specs that meet our streaming requirement. As this is dependent on the microcontroller's functionality, none of the tests below have been performed as of now.

Requirement	Verification
• Must be compatible with our microcontroller and the I2C protocol	A. Verify transceiver has I2C capability in datasheet

• Must operate on 2.4GHz bandwidth	A. Verify the transceiver operates on Bluetooth's standard bandwidth of 2.4GHz on the datasheet
• Must operate on a range of 30 meters	A. Separate device and insole by 10m B. Verify connection exists by checking app C. Repeat these steps for 5m increments up to the maximum 30 meters. D.
• Must transfer data losslessly	A. Verify that data transfer of pressure sensors is lossless through visual inspection of data set.

4. Conclusion

The timeline for completion our team has is within the next three weeks (by 4/14/19), we plan on having working finalized designs of both of the subsystems above in conjunction with the other sensor and power subsystems.

3/30	Build microcontroller platform that can be loaded with system code, write systems code to capture pressure data
4/6	Verify Bluetooth connection and integrity of data. Test transmission and visualization on mobile platform
4/13	Integrate with other systems while finalizing on product ready PCB. Prep for demo. Test in field to get data to show for the demo/presentation.
4/20	Final demo.
4/27	Final presentation.

So far, our project has gone according to schedule within the guidance of our project manager. As a team, Gerald and I work quite fluidly and effectively. As we are a small team of two, most of our work is done collaboratively and division of labor only occurs when required. All design has been done as a group. The main instances where individual work has been divided has been ordering specific parts pertaining to the systems we have designated each other to research. As a whole, I would estimate that we have an equal investment of time and effort in the success of our product.

One possible concern is that we are slightly behind on schedule on actual prototyping and product development. We hope to make up for our lack of progress within the coming 3 weeks. I believe the expectation that we can build a working product is very

realistic as long as there are no major issues with loading the microcontroller with system code and utilizing the Bluetooth transceiver.

Some of the concerns our project has is that it gathers and processes potential Protected Health Information (PHI), which is requires us to ensure confidentiality, integrity, and availability of said data [1]. We would not include diagnosis within the app to avoid violating #3 in the IEEE code of ethics[3] by potentially misdiagnosing a user. Because this product and the data it provides is not comprehensive enough, it should not be used as the sole piece of evidence for diagnosing physiological diseases. Primarily the data would be in the hands of a medical doctor who could utilize it accurately. It's important to note that this product will not be doing any analysis on the data, instead giving a professional means to conduct that analysis themselves.

5. Citations

[1] HIPAA Journal, 'What is Protected Health Information?'. [Online]. Available: https://www.hipaajournal.com/what-is-protected-health-information/. [Accessed: 6-Feb-2019].

[2] Circuit Digest, 'How to Burn Arduino Bootloader in Atemga328 IC and Program it using Arduino IDE'. [Online]. Availible: https://circuitdigest.com/microcontroller-projects/how-to-burn-bootloader-in-atmega328p-and-program-using-arduino-ide . [Accessed 25-Mar-2019]

[3] IEEE, 'IEEE Code of Ethics'. [Online]. Available: https://www.ieee.org/about/corporate/governance/p7-8.html. [Accessed: 25-Mar-2019].