CS 437 IOT Capstone Final Project:

Foot Monitor

Michael Potts

Mjpotts2

December 6th, 2021

Github link:

https://github.com/michael-j-potts/IOT/tree/main/Capstone%20project

Video demonstration:

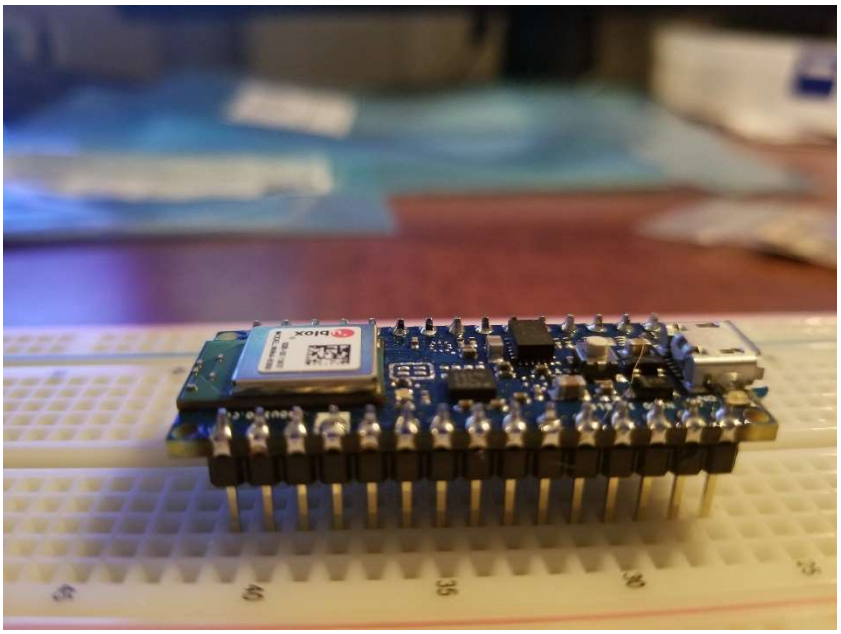
https://uofi.box.com/s/hahxcqqwhvj0ro06ac4kba55ii9cvdg2

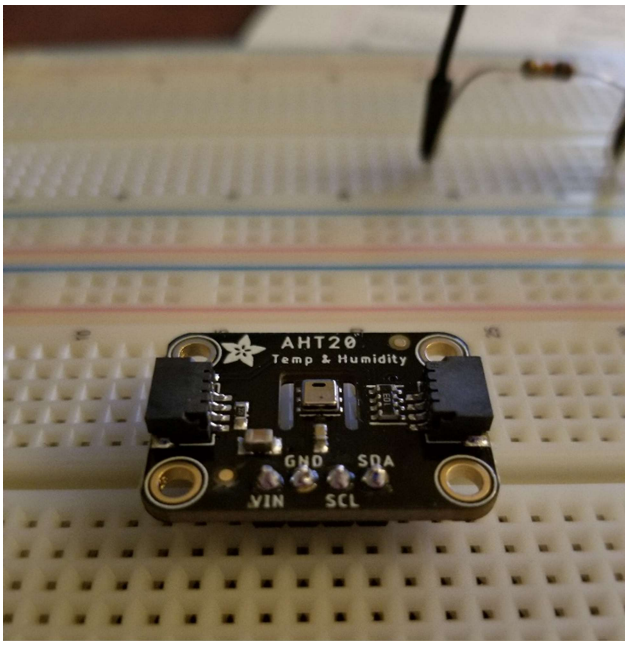
Contents

Design and Build:

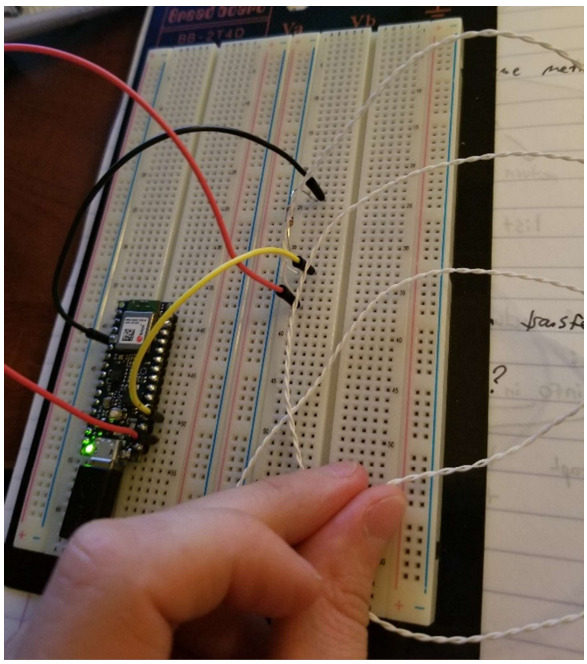
Most of the skills within this project were new to me and required a larger detail of study before being implemented.

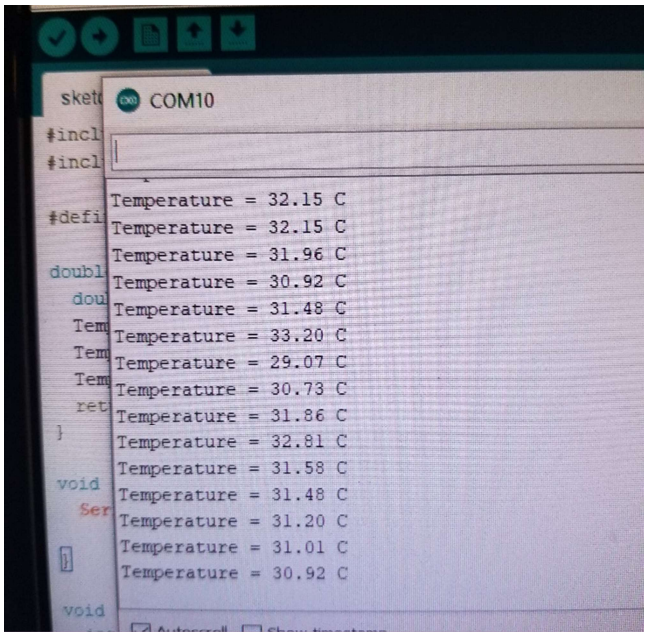
Both the AHT20 and Arduino Nano BLE 33 arrived without the header pins soldered (per my choice to try a new skill). The AHT20s pins were much smaller and therefore I was not as successful but overall, both components work well!

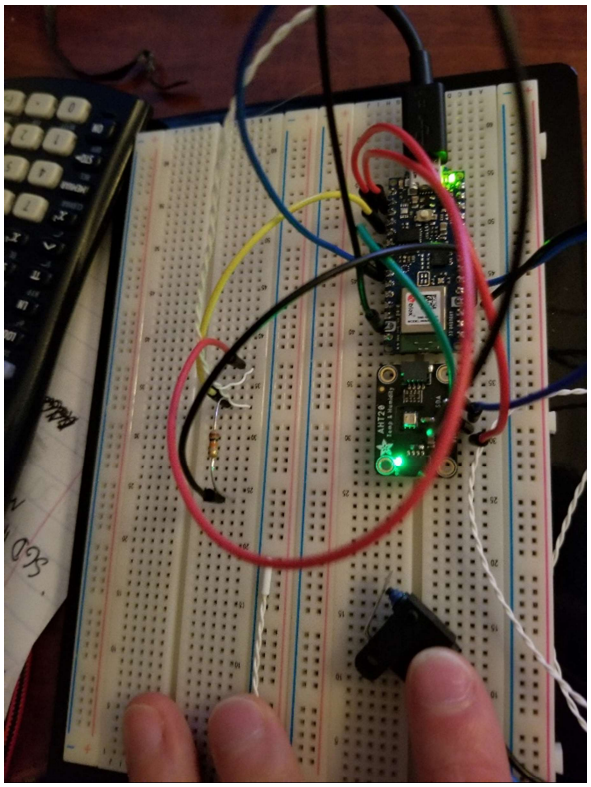


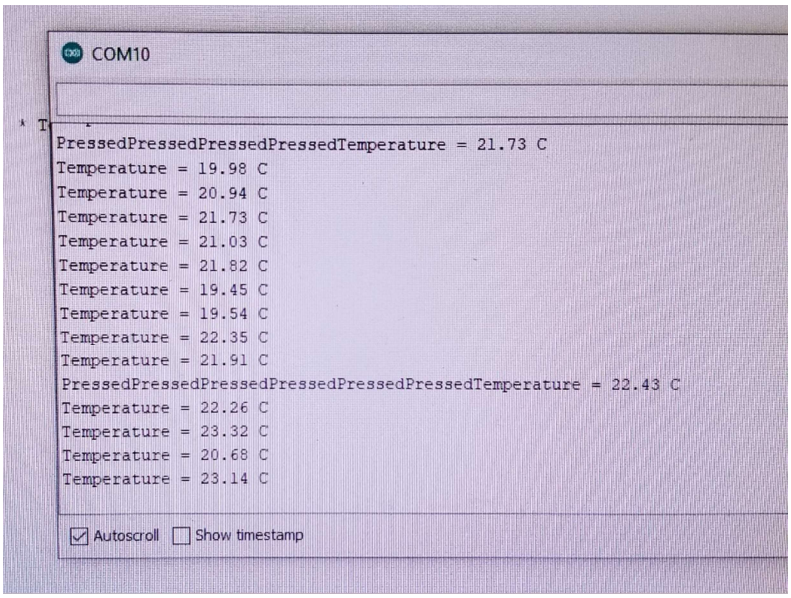


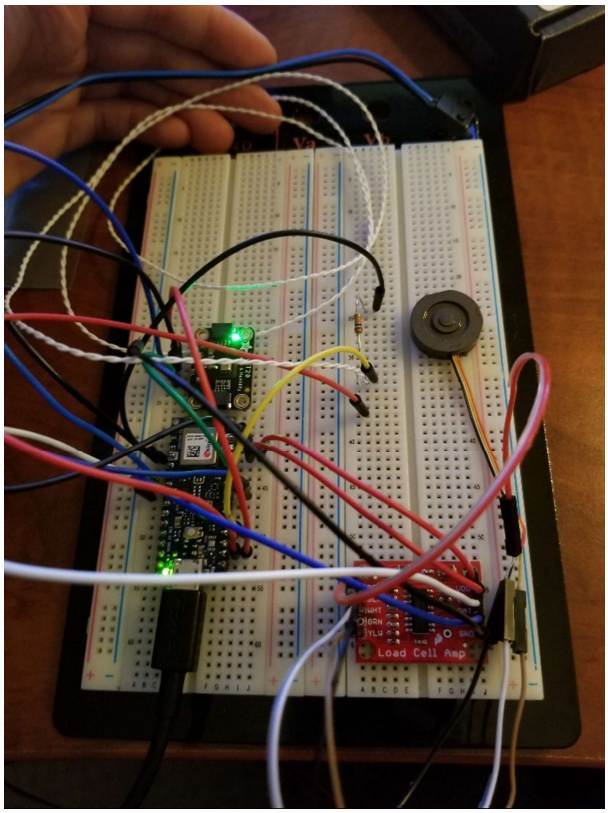
Next, I started with the thermistor, first installing, then testing the component to evaluate its use. (cite Temperature Measurement with a thermistor and an Arduino Gerald Recktenwald).





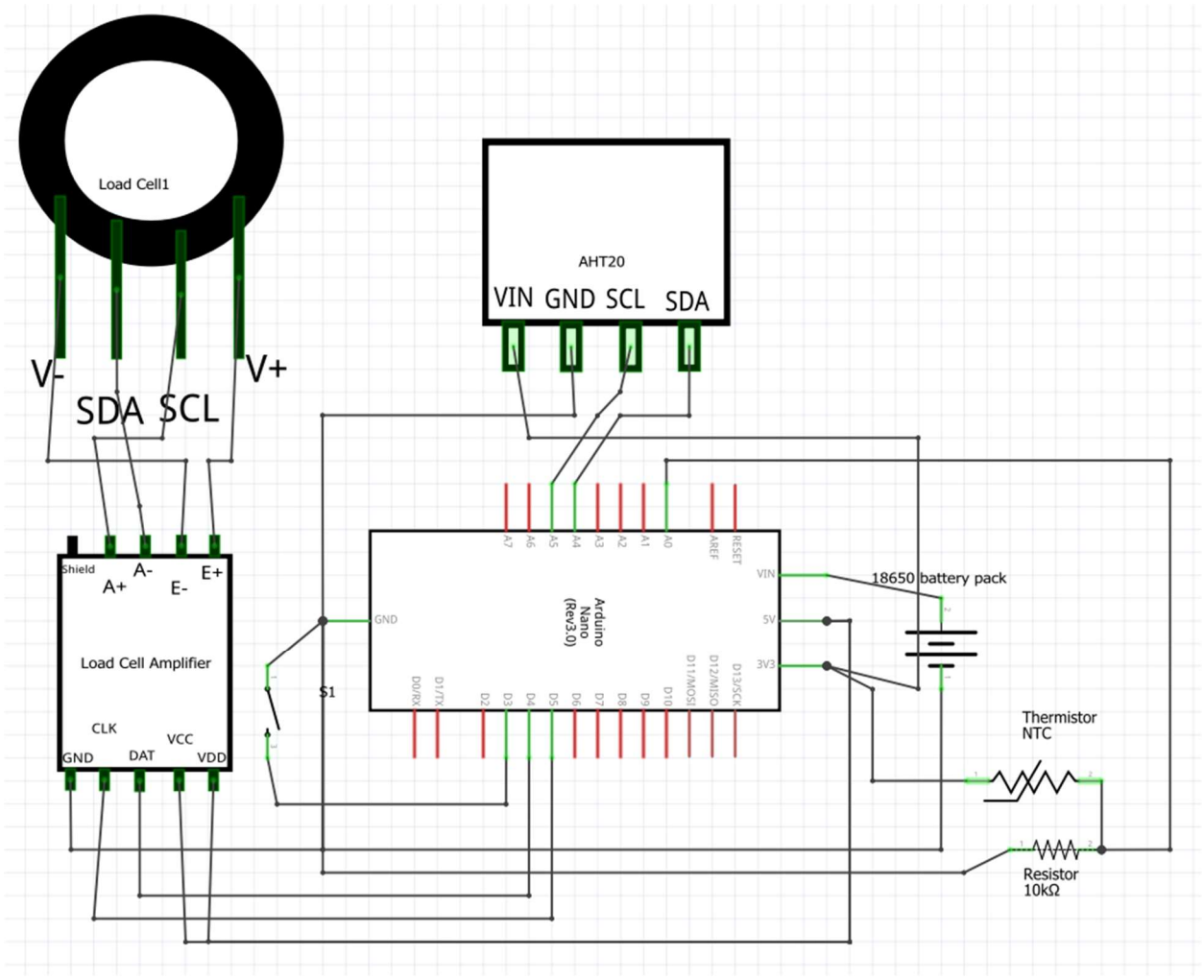




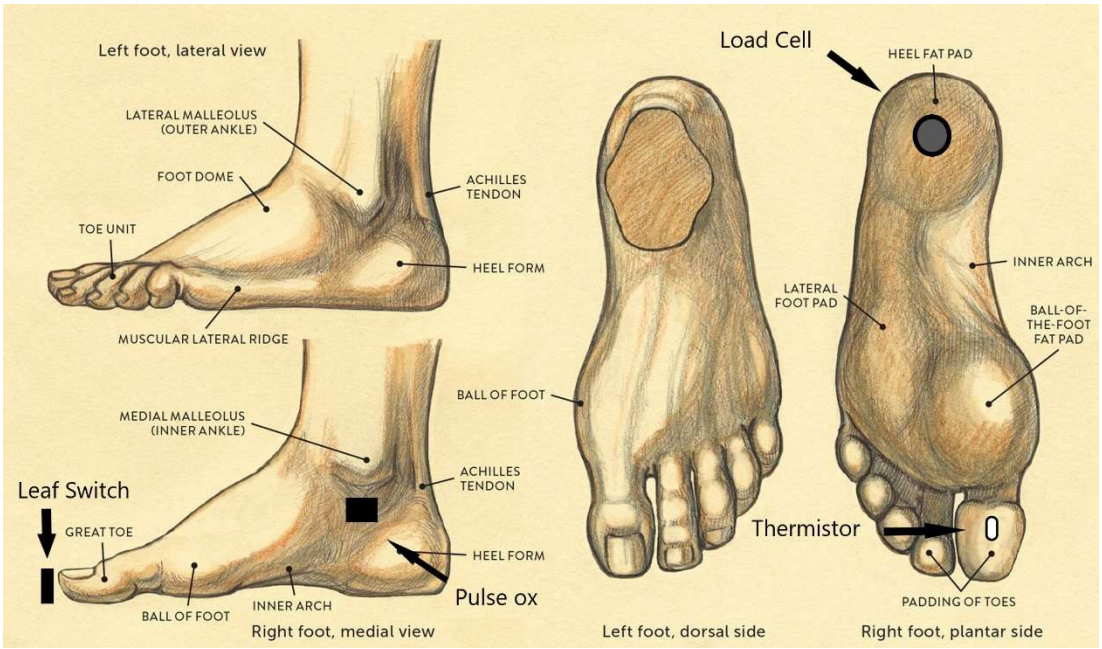


At this point in my build I ran into a few issues. I originally did not realize that the load cell would require an amplifier, so I ordered and waited for it to arrive, choosing to work on the programming. Once the amplifier arrived and I tested the load cell, the amplifier appeared to work well, but the load cell did not seem to register data. With that in mind, I have left the load cell programming in place, to be used when a working load cell is installed.

Finally, I received the last component, the pulse oximeter and integrated it into the ciruitry. Unfortunately again, the code for the oxymeter was programmed only for arduino uno's. Without significant revision and time, I was unable to install it to my arduino nano ble.



The rough placement of components within a shoe:



Arduino Programming:

Working with the Arduino ide, I found the process to be quite straightforward. Some decisions about the nanos data I made were: if the impact switch is triggered, we will return a Boolean value (since we don't need any further data, just the knowledge that it happened), and strings for the remaining data). A few notable problems I encountered were:

1. while(!Serial) - I found that after following an Arduino tutorial, I was unable to run my Arduino. However, every time I reuploaded and ran it on the ide, it worked perfectly fine? Deleting while(!Serial) solved this problem, as I no longer needed the serial monitor to run my nano.
2. Initiating and getting used to the Arduino nano ble Bluetooth.
    a. Services and characteristics – I debated between making one service with all the characteristics or several services with their respective services. I realized that adding multiple services, allows for greater customization, if a user would prefer to only connect to one service and its respective characteristics at a time.
    b. Nanos lack of pairing – The Arduino nano ble does not inherently pair with Bluetooth devices, and instead only loosely connects to devices. This would be a primary problem that would plague much of my project. That being said, it would appear that there are methods around this issue, notably using an Arduino nano, or utilizing an external Bluetooth module (which makes the nano ble redundant)
    c. Returning data – At first it was difficult to know what to expect for data reception. I used a helper app called nRF connect that would connect but not pair with an Arduino. This allowed me to visualize the data it was sending. Originally, it would only transmit hex values that made no sense. I eventually stumbled across an example that set the characteristics as BLEStringCharacteristics instead of my original double values. Further, I needed to add a character limit to the object, and in this case, I set the limit to 13 since I would only ever send 5, but wanted to allow some room. This allowed my nano to transmit string data!

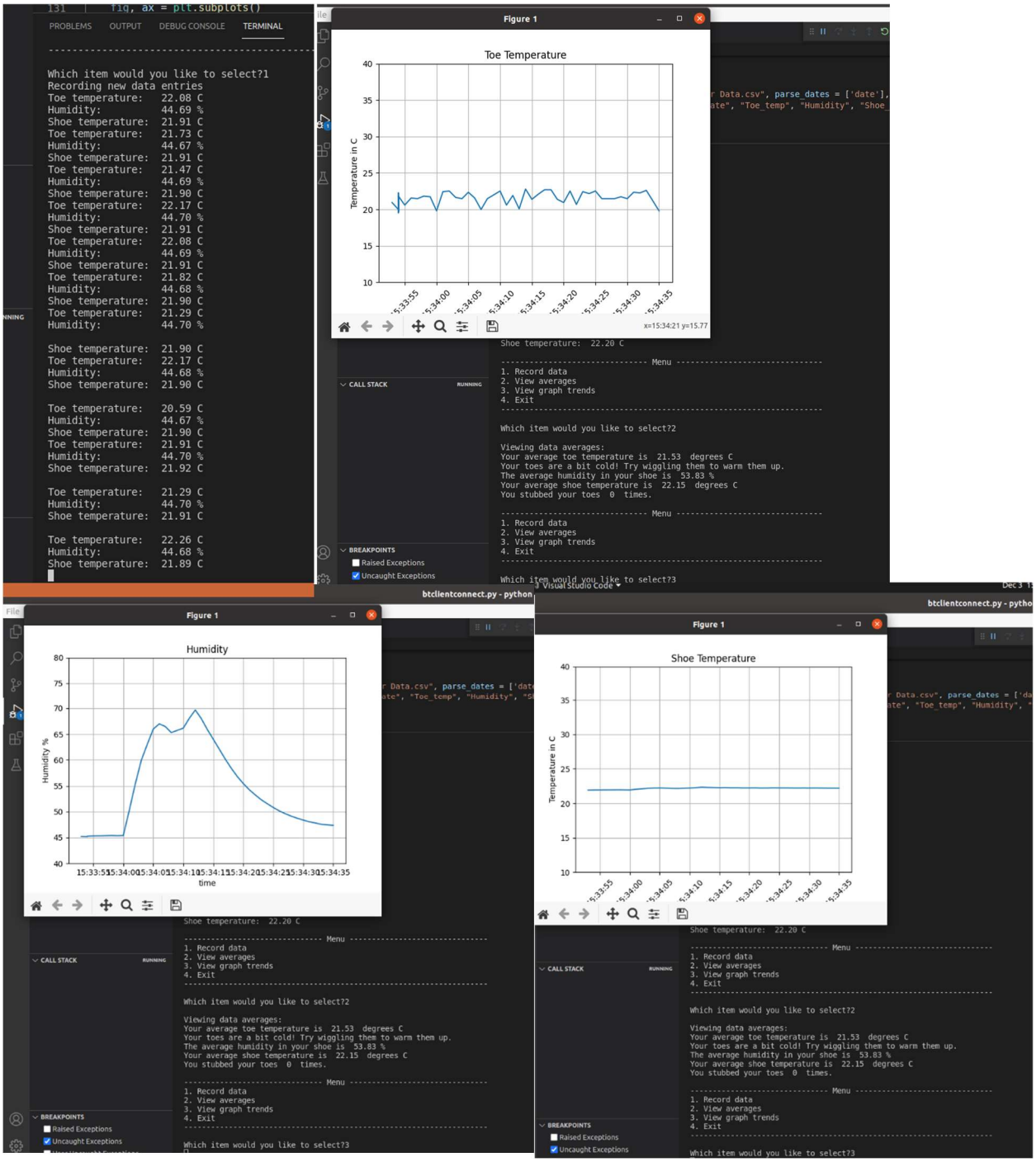
    The components that I programmed were, a leaf switch that delivers true when activated, a thermistor to measure temperature

Client Programming:

The BTClient.py program is the current program that connects and adds functionality to my Arduino nano board. Walking through the code, we have our import statements, a class, and some definitions. After this, we first want to set our device MAC address, look for/create a .csv file to store data (and prepare it with some headers), and then call our delegate class from bluepy to connect to the Arduino nano. At this point, once connected, we manually connect to the nanos list of services and characteristics and then enter our menu loop. Once in the menu, we want to select #1 first, as we would like to populate our empty .csv file with some data. This method will record 20 data items, specifically from notifications from the Arduino. This method so far has some timing issues between the program and Arduino, and so may record a few extra entries (nothing that breaks anything in the program though). Once we have finished recording, we close our file to save our data.

Next, we would like to select 2/3 from our menu. 2 will display the averages from our data, which, while not helpful over months of time, can be expanded to view more specific information like

daily averages etc. Further #2 will also display the number of times, the patient has stubbed their toe via the leaf switch engaging. #3 will allow us to print trend graphs of our data over time, which will provide more detailed analysis of the data the Arduino is collecting. So far, I have set up graphs for Toe temperature, humidity, and the humidity module's temperature (shoe temperature). Once further components are integrated, I will implement more graphs. As you can see for humidity, I breathed on the sensor to display its ability to capture. Further for the thermistor (Toe temperature), we can see it does bounce up and down a little bit, but that can be managed through setting a 5 second average within the Arduino or within the python file if desired.



Difficulties, Considerations and Future Design Changes:

Arduino build:

1. One Future design change I would implement would be to build the Arduino set without the header pins, as it will allow for a flatter build that can be built into smaller spaces. As this was my first build, I wanted to use the pins. This decision would also allow me to utilize a breadboard which helped with development.
2. Unfortunately, during the building process my load cell was either damaged during building or arrived not working. My load cell amplifier did not successfully receive any signal from the load cell, even though the amplifier module worked well and returned results. The code was left in as a replacement load cell will be able to fill the gap easily, just not within the allotted course time unfortunately.

App Programming:

One major issue during the programming of the android app, was that Kivy and python for android do not reliably have a method to connect to BLE devices. The app was programmed up to and including connecting to previously paired devices. However, since Arduino Nano BLE 33 does not allow true pairing, only one solution would have worked in my situation. A program called “Bleak” – (github link) would have solved my problem completely, but unfortunately, I ran into a bug currently without a solution (https://github.com/hbldh/bleak/issues/688). In its place, I instead chose to focus on a text interface and modified my Arduino to record information.

Prototype -> fully functioning device:

Due to time constraints and realizing that this project is much, much larger than originally thought/intended, I decided away from trying to make a real world device for the time being (implemented in a shoe), as I can demonstrate the overall idea well in this form. Since I would like to continue to work on it, I will need to make sure I don’t break anything and would like to have a platform to continue to develop it on (the bread board). Since my final design will be encapsulated into a small unit, it will be difficult to continue testing and altering in that state.

(unused) App programming:

To begin, during the summer of 2021, as a way to become more comfortable with python and prepare for beginning the masters program, I decided to create a hospital administration system that could add/remove patients, record patient data, and display the information in a text system. After having created that, I decided to make it an app for no particular reason. This led me to learning about kivy, a python app development language. I only finished about half of the work on the app before school started and it was relegated to a hobby project for another day.

It seemed to fit perfectly as an idea for my final project, so I decided to reopen the project in October/November. By early – mid November, I had finished off the skeleton of the app, and took care of the cosmetic issues bothering me. I also had a few methods of how I would record and access the data available (my main idea was a table of linked lists with the table being each week of the year, and the linked lists being the set of nodes for said week, as this would improve data organization and speed of lookup.)

When I had finished building my Arduino and had most of the Bluetooth in place for it, I then decided to start implementing Bluetooth for my app: It was difficult from day 1. Unfortunately, there are not many resources for python for android Bluetooth, and even fewer for BLE. I ended up finding an excellent resource for using Bluetooth in kivy, and implemented it hoping it would solve my problems. However, this would lead to the impasse I am still stuck at. The solution I found, only works with Bluetooth devices that have previously (and completely) paired with your android device. As covered previously, Arduino nano ble currently does not support full pairing, and instead only supports a much lesser version of pairing that is not recognized as secure by Bluetooth adapters and therefore fails. Another dead-end. I ended up joining the discord community "kivy" to ask for support and while they gave me limited help, ultimately, I was unable to find any helpful resources. After several more days of unsuccessful searching, I finally stumbled upon bleak! (Incredibly ironic name for how I felt at the time…). I ran into a bug at this point that the developer helped me solve. Specifically, my computer is missing firmware and I needed to add a specific adapter variable to my call (https://github.com/hbldh/bleak/issues/693). I was hoping this would be the only bump, but as this project has taught me, many more bumps can be experienced still. Specifically, after changing my adapter, the bleak scanner started working and even pairs(?) with my device. I am unsure if it is true pairing, as I am now receiving a timeout error, but the orange LED lights up and stays lit upon my Arduino. Bleak may be a solution, but I am unsure if it currently supports pairing for devices. Using the Bleak scanner, I was successful in stably connecting to my Arduino, but the BleakClient() method appears to have a timeout error, which is a known issue and being reported on (within the last few days even - https://github.com/hbldh/bleak/issues/688). While ultimately, I would like to fix this, it seems for now, I have to decide between three options:

1. I could wait and hopefully bleak will become available. The issue is that there may not be a fix.
2. I could redesign my nano, add some additional memory storage, and collect data (much like the zebranet system). This issue with this approach is that it takes the device out of the patients/consumers hand, and while it may work for some patient care, it would inherently still have problems. Further, the greatest benefit of this device would be for long term patients over years, not during hospital stays. This being know, it may work in outpatient and community centers but again, will be limited as it would not contain active feedback about moving position or check toes after stubbing your shoe.
3. I could also look at using an Arduino uno, as, to my knowledge, the Arduino uno with an attached Bluetooth device, may not have the same pairing issues. This approach will of course deviate further than option two from the original idea for this project. The device needs to be small enough to be mostly unnoticeable and either wear on a shoe or be implanted within a shoe.

All this to say, I would love to continue this project upon hearing back from the bleak team, and hopefully I will be able to make a working prototype.

Documents, links, and references:

Soldering guide: https://www.instructables.com/How-to-Solder-Pins-Header/

Button instruction guide: https://arduinogetstarted.com/tutorials/arduino-button

Thermistor paper: “Temperature Measurement with a Thermistor and an Arduino”, Gerald Recktenwald, may, 2013.

AHT20 Humidity sensor: https://cdn-learn.adafruit.com/downloads/pdf/adafruit-aht20.pdf

HX711 load cell amplifier: https://github.com/sparkfun/HX711-Load-Cell-Amplifier/tree/V_1.1

Pulse ox: https://www.maximintegrated.com/en/design/reference-design-center/system-board/6300.html

Pulse ox cont: https://github.com/MaximIntegratedRefDesTeam/RD117_ARDUINO

Interrupts guide: https://www.arduino.cc/reference/en/language/functions/external-interrupts/attachinterrupt/

Various Bluetooth implementations (all android implementations so far unsuccessful, bluepy = PC):

Bluetooth Guide: https://ladvien.com/arduino-nano-33-bluetooth-low-energy-setup/

Bleak: https://github.com/hbldh/bleak

Python for android: https://pypi.org/project/python-for-android/

Bluepy: https://github.com/IanHarvey/bluepy

Pyjnius: https://pyjnius.readthedocs.io/en/stable/

Kivy documentation: https://kivy.org/doc/stable/