Technology and Concept Report

Team Omicron

CE 4000 004

October 5th, 2018

Mitchell Larson

Gunther Huebler

James Windorff

Zachary Oberbroeckling

Brandon Reed

Contents

[Introduction 4](#_Toc526529111)

[General Approach 4](#_Toc526529112)

[General System Requirements 4](#_Toc526529113)

[General Block Diagram 4](#_Toc526529114)

[Component Technologies 4](#_Toc526529115)

[Form Factor 5](#_Toc526529116)

[Options 5](#_Toc526529117)

[Other Form Considerations 6](#_Toc526529118)

[Form Factor Conclusions 6](#_Toc526529119)

[Sensors 7](#_Toc526529120)

[Options 7](#_Toc526529121)

[Sensors Conclusions 8](#_Toc526529122)

[Power 8](#_Toc526529123)

[Options 8](#_Toc526529124)

[Power Conclusions 10](#_Toc526529125)

[Data Collection and Transmission 11](#_Toc526529126)

[Microcontroller Options 11](#_Toc526529127)

[Microcontroller Conclusions 13](#_Toc526529128)

[Receiver 14](#_Toc526529129)

[Data Receiving 14](#_Toc526529130)

[Data Transmission 14](#_Toc526529131)

[Receiver Conclusions 16](#_Toc526529132)

[Data Storage 17](#_Toc526529133)

[SQL Databases 17](#_Toc526529134)

[NoSQL Databases 18](#_Toc526529135)

[TimescaleDB 18](#_Toc526529136)

[Data Storage Conclusions 19](#_Toc526529137)

[Data Processing 19](#_Toc526529138)

[Mobile Phone 19](#_Toc526529139)

[Web Server 20](#_Toc526529140)

[Costs 21](#_Toc526529141)

[Web Server Conclusions 21](#_Toc526529142)

[Data Presentation 22](#_Toc526529143)

[Mobile App 22](#_Toc526529144)

[Web Application 22](#_Toc526529145)

[Data Presentation Conclusions 26](#_Toc526529146)

[Web Hosting Overview 27](#_Toc526529147)

[Pricing 28](#_Toc526529148)

[Web Hosting Conclusions 29](#_Toc526529149)

[Block Diagram with Available Component Technologies 29](#_Toc526529150)

[Final Design Approach 29](#_Toc526529151)

[Final Block Diagram 30](#_Toc526529152)

[System Requirement Satisfaction 30](#_Toc526529153)

[Modularity 30](#_Toc526529154)

[Total Cost 31](#_Toc526529155)

[Conclusion 31](#_Toc526529156)

[Glossary of Technical Terms 32](#_Toc526529157)

[References 32](#_Toc526529158)

# Introduction

The purpose of this document is to document possible approaches for Team Omicron to successfully implement the LogiSteps project, as proposed in the formal project proposal document. Team Omicron first theorized a general data flow for the system, identifying key components and aspects of the design. In-depth analysis of market solutions and compatible technologies that could be used for successful development of the project was performed, helping to identify the best technologies to be used in the system design. In this document, the general system (without specific hardware and software components) will be discussed, familiarizing readers with the general structure of the project. After Identifying the basic project requirements, component technologies that help achieve the respective system requirements will be discussed in depth, ending with a conclusion of the best available option. The possible modularity of this project’s design makes it possible to choose many technologies independent of other components in the project. Lastly, the final approach will be discussed; this will be the optimal design using the selected component technologies based on team Omicron’s research.

# General Approach

## General System Requirements

LogiSteps, as introduced in Team Omicron’s proposal, is a full stack product designed to track user’s fitness activity. To be invisible to the user, providing a seamless user experience, the fitness tracking device is going to be embedded into a user’s shoe. This device will use sensors tracking the pressure between a user’s foot and the ground, feeding data into a data collection device that is also located on the user’s shoe. In addition to the sensors, whether provided by the sensors or not, the data collection device will require enough power to collect sensor data and also transmit the data off the device. The data collection device will need to transmit data to a receiving device, whether that be in close proximity to the data collection device or not. Upon receiving data, the receiver will need to store and the process the data for long term retrieval, indexed by time. A final system component will be required to present data to a user. In short, the general requirements of the system, regardless of actual implementation and technology selection, are those shown in figure 1.

## General Block Diagram

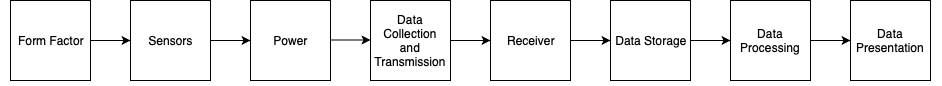


Figure 1 - -General System Requirements for achieving LogiStep's functionality, as described in the project proposal.

# Component Technologies

Each component shown in figure 1 is high-level, only describing the general function of the specific stage in LogiStep’s design approach, satisfying the high-level requirements of the project. By drilling further into each stage, specific technologies can be identified which may provide the tools needed to achieve successful completion of the project. In the following sections, several hardware and software technologies will be discussed, accompanied with deep analysis and comparisons to other alternative technologies. After discussing the advantages and disadvantages of varying technologies as they pertain to design of LogiSteps, a technology that best suits the need of the specific stage in the design will be chosen. Using the conclusion of each technological component, a final system design will be discussed.

## Form Factor

The form factor of the device is one of the most important aspects. Ideally it will be an insole that requires no maintenance and is thin enough so as to not reduce comfort. Additionally, the material will need to be flexible and temperature/pressure resistant to not be damaged under normal use. The best option for this seems to be some type of rubber. The material also needs to be malleable enough easily shape and work with to fit other design requirements.

Other than the material used, the characteristics of the device need to be considered. In a perfect world, the insole will be as flat as possible and contained entirely in the shoe. However, due to other components required for the device, it may be necessary to have a thicker sole with the main electrical components attached outside of the shoe.

Options

#### Silicone Rubber

Silicon rubber can be cast and cured into any shape (including thin ones); it is also very robust making it a desirable choice. The rubber would be used to help activate the sensors and house the electrical components. Activating the sensors may require deformation which can be easily done with a shape that is producible with silicone rubber. Housing the electrical components requires a minimum amount of stress on them and that they remain safe from static. This can be achieved using silicone rubber since it is both an insulating material and can absorb much of the shock/stress put one them.

##### Cost

The cost of silicone rubber depends on the quality, quantity, and form it comes in. The ideal form is a type of castable liquid. A sizeable amount of this costs $20-$50.

#### 3D Printed Flexible Rubber

The main benefit to this approach over silicone rubber is that it can be designed with more precision. It has all the same benefits as silicone rubber as well. A drawback is that it requires a more complex design since it would not be trivial to get the necessary electrical components inside.

##### Cost

Assuming free access to a 3D printer, the cost of this would be the cost of the rubber filament itself. There are a few options for an ideal filament and all are within $30-$40.

The labor cost of this should be noted as well. It is a more complex process and design resulting in potentially a significant amount more time devoted.

#### Polycarbonate/Plastic Sheets

This option is the thinnest of them all. The concept here would be that nothing other than the flat sensors are contained in the shoe and the electrical components are housed outside the shoe. The benefit to this is a maximum amount of comfort since the insole is paper thin. The major drawback however is the second part of the device outside of the shoe that is required as well.

##### Cost

There are many options to the type of material, vendors, sizes, and cost. This option would cost as little as $10 and no more than $20.

### Other Form Considerations

The layout of the sensors and electrical components needs to be considered in any form factor. The sensor layout depends largely on the number of sensors to be used, which depends on power requirements and size of electrical components. But the simplest and most desired concept will be the fully insole version of the device, with at least two sensors on the heel and forefoot to get pressure differences between those areas. The electrical components will need to be housed in an area that has the least amount of pressure; this would be the bridge of the foot. A concept of this layout is shown in figure 2.

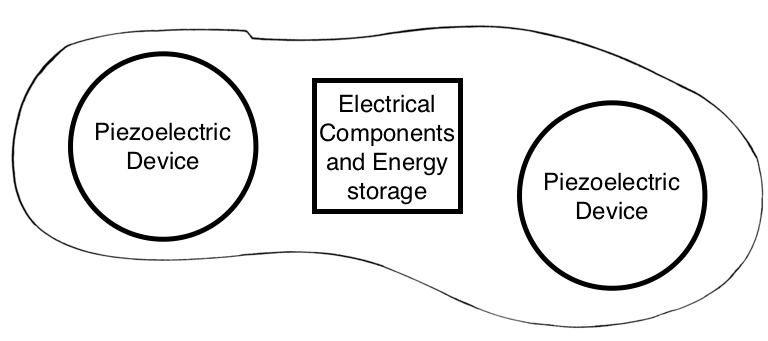


Figure 2 - Theorized physical form of the embedded LogiSteps application.

### Form Factor Conclusions

The most attractive approach seems to be the 3D printing of flexible rubber. It allows detailed design which may be important for sensor development, and while it is more complex than silicone rubber, the increased quality control may end up saving time and effort.

The thin polycarbonate /plastic material would the most different design. It would require housing the electrical components somewhere other than the insole, which would most likely be outside of the shoe entirely. It would be the simplest option because it requires no complexity in design and it also allows more freedom with electrical component sizes but less freedom with sensor choices. Because of these things, the reason to go with this option would most likely be due to other factors of the sensors and electrical components.

## Sensors

The most important requirement for the sensors is that they remain thin and robust. Ideally, power would be obtained from these sensors as well, removing the need to charge the device. If possible, the sensors would be capable of detecting specific amounts of pressure so that factors such as weight can be determined.

### Options

#### Piezoelectric Material

Piezoelectric sensors appear to be a great choice; they’re thin, scalable to meet system requirements, produce power when stress is applied, and certain types of piezoelectric material are very robust. The material used for this would ideally be PVDF (polyvinylidene difluoride) since it is the thinnest type of piezoelectric material that is commonly used and can handle the most stress. The only problem with PVDF is that it is generally expensive compared to the next best option, piezoceramic. Piezoceramic is capable of producing more power, it’s cheaper and most easily accessible, but it is quite brittle. Either option will require the form factor to be one capable of deforming the material significantly enough to produce power but not enough to damage the sensors. The largest downside to piezo material, besides that cost, is that it requires extensive work to calibrate the sensors well enough to detect specific pressure.

##### Cost

The best cost option for the sensors would be piezoceramic deposited on copper. This can be as cheap as $1 per sensor up to $3 per sensor. This would also make prototyping very easy since they can be damaged without too much loss.

The best option functionality-wise would be a piezo PVDF film. This option is ideal because it is extremely thin (μm range) and can be layered for voltage and material strength. This option comes at a large price however - the best options seem to be around $133 per 8’’x11’’ sheet.

#### Pressure Dependent Resistive Sensors

Pressure dependent resistive sensors are the common approach to detecting pressure. These sensors have a bassline resistance that changes based on the pressure applied to them. This would allow for very specific measurement of pressure and there are far more choices (most are cheaper then piezo material). The downside to the resistive sensor approach is that power cannot be harvested from them. Instead, it would require additional power to measure them. Another benefit to this approach would be that the sensors would not require an activator, meaning the insole could be entirely thinner.

##### Cost

Reasonable quality sensors start at around $5-10 each but can be up to $100 each for very high-quality ones. The high-quality ones are usually made for industrial type applications so those are not only out of the price range, but unnecessary for the application. $5 sensors should be suffice.

### Sensors Conclusions

The PVDF film is the top choice for the sensors due to its power harvesting ability. The goal of the LogiSteps is to never require a charge and force resistive sensors would require the use of a battery with significant energy density (far from ideal). On the other hand, the cost of piezoelectronics, as well as their complexity, means that other power options need to be explored in case as a backup if piezoelectronic sensors fail. Based on this research however, Team Omicron concludes piezoelectronic sensors to be the best technology for measuring fitness data.

## Power

A goal of the device is to never need to be charged. This requires either a battery or supercapacitor capable of sustaining the device for its entire lifetime, or the ability of the system to harvest energy from human locomotion. The harvesting of energy can only be done with piezoelectric materials. Other types of energy harvesting were considered such as triboelectricity, thermal energy harvesting, and even RF energy harvesting, but the investigation into those options proved them to not be viable.

### Options

#### Battery

The battery option is the least desired since batteries tend to not do well under pressure or when exposed to heat. This would require the form factor of the device to be partially outside of the shoe to avoid danger risks. The largest benefit of the battery however is that the device could potentially be powered for its entire lifetime on a single charge. At a rate of sending 20+ sensor values every second, using a low footprint battery, the device could last several years before needing to be recharged.

##### Cost

Thin film Lithium Ion Polymer batteries cost as little as $5 each and don’t get much above $20 each. The number of options and sizes here are quite significant as well.

The cost of design needs to be mentioned as well. Designing the battery power supplying system would be nearly trivial compared to the piezoelectric power delivery system, so it may be the better option based simply on time constraints and labor required.

#### Supercapacitor

The main benefit to a super capacitor over a battery is that a super capacitor does perfectly fine under pressure, is not significantly affected by heat, and there are less risks. The main drawback is that they have a much lower energy density. A supercapacitor would only be able to power the device for a matter of months. A small drawback is the fewer amount of supercapacitor options, but another benefit is that charging the device would take only a matter of seconds.

##### Cost

Thin film, high capacity supercapacitors are generally all around $5-$8. The DMHA14R5V353M4ATA is currently the best seeming option due to its size, capacity, and voltage range; They’re $7 each. Due to their size, it’s possible to even have multiple in each shoe, potentially extending the battery life to lifetime of device capable ranges, but at a higher cost.

#### Piezoelectric Energy Harvesting

By far the most attractive option because this system never requires a charge, nor any type of costly or dangerous components. It is however, also by far, the most difficult system to develop. When piezoelectric material is deformed, it produces a voltage. That voltage can be harvested to power the rest of the device.

##### Piezoelectricity Harvesting Design

Since energy needs to be controlled to power the device at proper and consistent voltage/current limits, other components will be needed. These other components would be diodes, capacitors, and a voltage regulating chip. The diodes and capacitors depend largely on the voltage and amount of energy in general that the piezo material can produce. A good option for a voltage regulating chip this is the LTC3588-1; this is a silicon-based chip that can maintain proper voltage limits while being extremely efficient. The design of the energy harvesting circuit is shown figure 3.

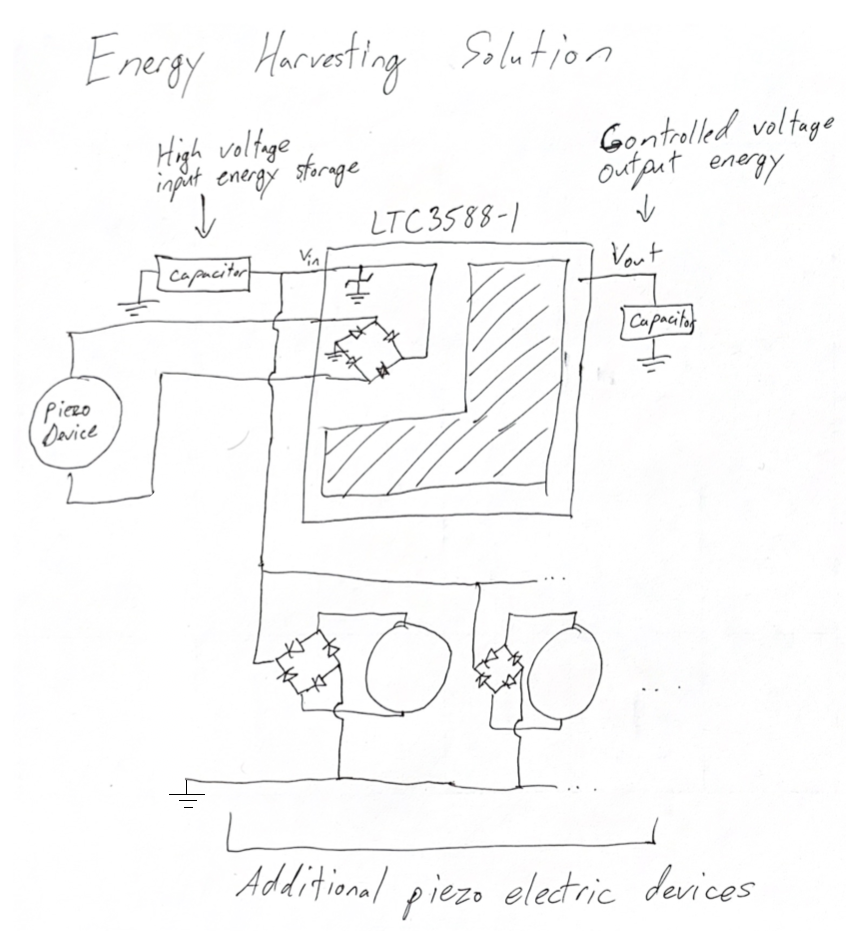


Figure 3 - Prototype circuit design for the energy harvesting circuitry.

The voltage generated by the piezo devices will alternate, thus full bridge rectifiers will be used to polarize the voltage, so it can be stored on a capacitor. The uncontrollable nature of the voltage requires that a buck regulator or similar type of voltage controller be used so a voltage level adequate to power a microcontroller can be output. The LTC3588 has voltage output controlling components, and high voltage energy can be stored on the input capacitor to be used to power the output capacitor at a controlled voltage. This system requires two capacitors and a full bridge rectifier circuit for every additional piezo device used, giving the ability to scale the system to meet requirements as long as the additional components can fit on the shoe.

The piezoelectric device would also require an activator. This activator would need to have the ability to deform the material to produce a voltage. This activator would be built into the form factor of the insole. An example of an activator is shown in figure 4.

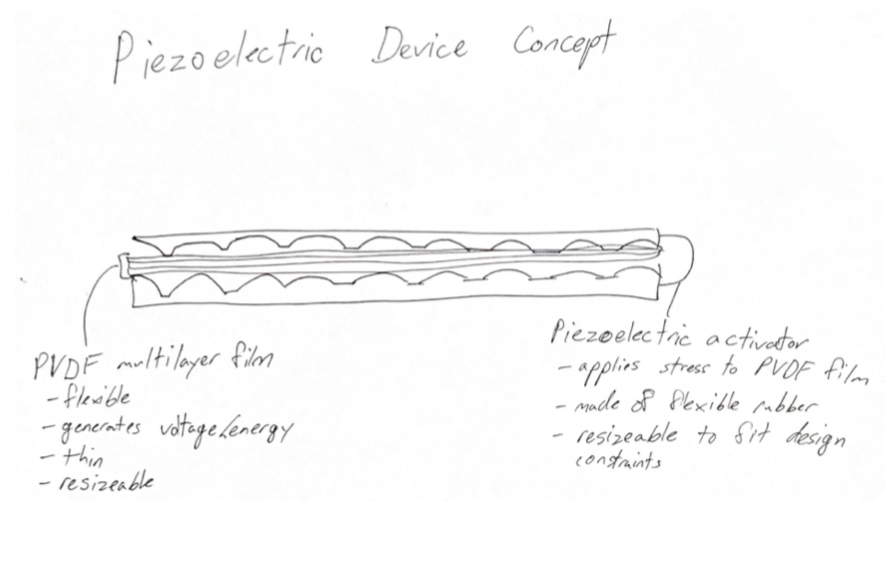


Figure 4 - Theorized design for the piezoelectric sensors. Note the rigid edges in the design, enhancing voltage output.

##### Cost

The piezo material itself is the costliest aspect of this approach, which was previously. The LTC3588-1 is the next most costly component at about $7 each. The diodes and capacitors are all in the range of tens of cents, so until mass production begins, their cost will be considered negligible.

## Power Conclusions

The piezoelectric energy harvesting option is the most desirable for many reasons, including cost, but comes at the price of difficult development. The supercapacitor option is the next most preferable due to its robust nature and scalability. The battery is the least desirable option because of danger risks, but it is also the one that requires the least development, so it is still a viable alternative.

It is also possible to combine the supercapacitor and piezoelectric ideas. If a supercapacitor is used, it can supply the device with power for a number of months, and when combined with piezoelectric energy harvesting, is much more likely to be able to supply the device with power for its entire lifetime. This approach would be taken if the energy harvesting alone is found to be incapable of supplying enough energy to fully power the device but may be enough to extend the lifetime of the device when supplied with a significant enough amount of initial energy. The only drawback to this approach seems to be that a supercapacitor as well as the energy harvesting circuit adds to the cost and footprint of the system.

## Data Collection and Transmission

A microcontroller will be used to collect sensor data and transmit to a receiver. Since there are many options for microcontrollers to use, research was performed to find a microcontroller that had the necessary components to read sensors and transmit data over Bluetooth. Specific requirements the microcontroller include: ADC count, power, size, cost, lead time, programmability, and documentation.

After researching microcontrollers, a handful of microcontrollers with Bluetooth capabilities embedded in them were found. It was decided to choose a microcontroller with embedded Bluetooth rather than using an external component, which would add additional complexity to the system.

### Microcontroller Options

By researching, a handful of microcontroller options were found that could be used for the project. Each microcontroller is embedded with Bluetooth and meets each of the specific requirements mentioned above. The first microcontroller option is the CC2540 chip from Texas Instruments. The second microcontroller option is the nRF52832 chip from Nordic Semiconductor. The third option is the ESP32 chip from Espressif Systems. Each microcontroller option has its own advantages and disadvantages, and they will be explored by reviewing the specific requirements mentioned above for each option.

#### ADC count

The ADC count of the microcontrollers is important because there needs to be enough channels to read each sensor input. Both the CC2540 and the nRF52832 have 8 channels for a 12-bit ADC. The ESP32 excels in ADC count, having 18 channels for a 12-bit ADC. Although it seems that the ESP32 is the best choice in terms of ADC count, a simpler design in terms of sensors is desired. Therefore, the microcontroller won’t require more than a few ADC channels, so the large number of channels with the ESP32 is not very important.

#### Power

The power requirement for the microcontroller is one of the more important requirements, since the hope is to be able to power the microcontroller with the voltage created by the sensors. The CC2540 has a supply voltage range of 2 – 3.6 volts, the ESP32 has a recommended supply voltage between 1.8 – 3.3 volts, and the nRF52832 has a supply voltage range of 1.7 – 3.6 volts.

Overall, the nRF52832 chip has the best power options. Not only does it have the smallest voltage requirement, it also provides a low power mode, and allows the microcontroller to drift in and out of a powered state.

#### Size

Since the space for housing electrical components will be limited, small size for any microcontroller option is essential.

SparkFun provides development platforms for microcontrollers and provides a platform for both the nRF52832 and the ESP32 microcontrollers. Both development boards are the same size: slightly longer and less wide than a quarter. The CC2540 development kit is a lot larger than the SparkFun development platforms.

Overall, the nRF52832 and ESP32 development boards are both very size efficient. They would be small enough to use in the system design, while also being large enough to properly work with and connect the sensor inputs.

#### Cost

During development, it was determined that it would be best to have multiple microcontrollers to work on at the same time, if necessary. Also, it would be best to have an extra microcontroller in case there are any problems or malfunctions with the ones that are worked with. Therefore, cost is an important factor for the project. The development boards mentioned in the size section both sell for $19.95, while the CC2540 development kit sells for around $50.

In terms of cost, the development boards provided from SparkFun Electronics are the cheaper options.

#### Leadtime

Leadtime is an important aspect of this development process, since the project is expected to be finished by the end of spring 2019. Therefore, there is a need to choose a microcontroller that is both readily available and ships out quickly. Both SparkFun Electronics development boards are in supply and can be shipped within 3 – 11 business days (or faster with a cost increase). The development kit for the CC2540 microcontroller is also readily available and will ship in 4-7 business days.

Overall, the availability and lead time for each microcontroller chip is about the same, so it isn’t a very important deciding factor.

#### Programmability

Programmability is one of the most important aspects of a microcontroller for this project, since the preference is to have a microcontroller that is as simple to work with as possible. The programmability for each microcontroller can be determined by looking at documentation provided for the development boards that would be worked with.

The CC2540 programming kit provides a datasheet for the microcontroller kit. It provides information about the microcontroller itself like a regular datasheet while also providing information about software available to use. It doesn’t provide much in terms of program examples or troubleshooting. The nRF52832 and ESP32 development boards both provide a hookup guide to get started with their board. They both provide example circuits and programs that can be used to ensure that the microcontroller and board is working too. The difference between the two is the documentation for the microcontroller itself. The ESP32 provides a well-documented IDF along with a few example applications. The Nordic website provides numerous example applications as well as an open forum for continuous support when working with the nRF52832 microcontroller.

In terms of programmability, the provided example applications and code for the ESP32 and the nRF52832 are extremely helpful for programmers looking to use the product. Beyond sample applications, the nRF52832 microcontroller has an extensive amount of support and application examples for it, making it most likely the easiest product to work with.

#### Documentation

The documentation for a microcontroller is an important aspect for anyone wanting to use it.

The smallest datasheet is the CC2540 datasheet with 33 pages. It is mostly made up of schematic diagrams and characteristic information. The next smallest datasheet is the ESP32 datasheet with 43 pages. It is mostly made up of pin definitions and peripheral information. The largest datasheet is the nRF52832 datasheet with 555 pages. This is far and away the most detailed and most helpful datasheet out of the microcontroller choices. It contains a very detailed table of contents that makes it easy to navigate to a certain topic of interest.

In terms of documentation, the nRF52832 microcontroller has the best documentation provided for users. The datasheet covers nearly every possible topic of interest, making it very easy to find answers for any questions that need to be answered. Not only is the datasheet much better than the other choices, but the Nordic Semiconductor website provides an extensive amount of helpful documentation including starting guides, product specifications, and software user manuals.

### Microcontroller Conclusions

After looking at the specific requirements for each microcontroller option, it could be seen that each would be a viable option for this project. The ADC count, power requirement, size, and lead time for each microcontroller is acceptable for what this project would need. Therefore, choosing a microcontroller came down to the programmability, cost, and documentation.

Our group decided that the Nordic Semiconductor’s nRF52832 microcontroller is the best option. Not only does SparkFun Electronics provide an incredibly useful development kit with very detailed instructions, but the Nordic website provides much more documentation and support than the other microcontroller options. The nRF52832 should be the most accessible microcontroller for this project and should be the least challenging to work with.

The cost of purchasing a single nRF52832 microcontroller development board is around $27 including shipping. The cost of purchasing multiple microcontrollers to work with (and have one as backup) would be around $70 total.

## Receiver

How data is transported through LogiSteps, and how the user views it quickly, is what makes the product useful. Using a mobile app on a mobile phone is not the only way to transport the data to the server, but it minimizes the amount of power needed to get data from the microcontroller to the server. Some other advantages of using a mobile phone with a lightweight app to relay fitness data to the server are as follows:

* **Power Saving –** The mobile app allows the of use Bluetooth on the LogiSteps device. Bluetooth is much lower power than the alternative which is LTE. This will save power on the device which further enable the device to be passively powered.
* **Ease of Use –** The mobile phone is something everyone has. Almost everyone carries it with them everywhere. This allows for ease of use and no need to store data on the LogiSteps device or the phone (provided the phone has LTE turned on or is connected to WIFI).
* **Small Application –** Because the web server is storing the data, and is the primary method of displaying the data, the app only needs to transfer the received data to the web server. It also means that the app does not need to do any heavy processing of the data. This will keep the app small for those who have limited space on their phones.
* **Updates –** The use of the app store allows the pushing of updates to be seamless. Yes, the application needs to be pushed out to anyone who owns LogiSteps. However, the update does not have to come directly from development. The update can be developed outside of the store, then pushed when ready.
* **Additional data** – The mobile phone can provide additional data points such as GPS data.

There can be two parts to the receiver - the aspect that receives data from the microcontroller over Bluetooth, and the aspect that relays the data to the web server if data processing and UI is not being supplied by the mobile phone.

### Data Receiving

The method for receiving data from the microcontroller has a few different options: LTE or Bluetooth. Between these options, only one is acceptable for the project’s constraints: Bluetooth. Bluetooth has a low energy mode, so it does not take up too much power. Power is a large constraint for LogiSteps to be passively powered. LTE has much higher power requirments, ruling it out

Android has a library already in place for Bluetooth connections. It also has protocols in place to store the key for the product, so a connection can be done automatically after the pairing is done. The libraries are stored in three different permissions. These libraries are very well documented on the android developer page.

### Data Transmission

The LogiSteps data, if it is assumed to be sent to a server, sent straight from the embedded device to LTE, or from the Android device and onto the internet, will need to encapsulate the data in a data protocol. There are many data protocols that can be used to send data over the internet, either over LTE from the embedded system, or from a mobile phone app.

#### MQTT

MQTT is a Message Queueing Telemetry Transport. It is a lightweight (low bandwidth), publish/subscribe protocol that runs over TCP/IP. It is ideal for one-to-many information delivery. MQTT offers three QoS, or qualities of service, for its messages. At most once, the information might be lost. At least once, duplicate messages might occur. And Exactly once, no loss or duplicity. So, there is low bandwidth usage, medium, and high. Of note is its message header, which is a fixed 2 bytes in length. With these features MQTT was designed for constrained, low bandwidth devices, while assuring reliability in its messages. MQTT can also ensure the security of its data using SSL, or other data encryption methods.

Advantages of MQTT for LogiSteps are low bandwidth, reliability, and low power consumption since it is lighter weight than HTTP and will consume less of the user’s phone battery.

A Stephen Nicholas performed a power profiling comparing HTTPS and MQTT. What he found was that for a connection receiving sporadically MQTT used about 70% as much power as HTTPS over 3G, and over WIFI only used about 10% as much power. Sending as fast as possible saw MQTT using 2% less battery/hour over 3G. For sending data he found that MQTT used 1% less battery/hour over 3G, and 2% less battery/hour over WIFI. MQTT also had a better rate of successful packets. So, for LogiSteps, MQTT’s clear advantage is it will use less energy. The expected usage of LogiSteps and its app could lead to a considerable saving of battery life for the user, were MQTT to be the chosen data protocol.

For MQTT on android there are a few libraries out there that can be used. There is the Moquette library on GitHub which has almost one-thousand stars. Moquette also uses Netty for encoding/decoding. This library would be fairly easy to begin using. There is also an MQTT client by IBM using either IBM®MessageSight or IBM WebSphere® MQ, which has a lot of tutorial/development help. It also has a C library, Mosquitto, which would enable development directly on the embedded system.

#### CoAP

The next protocol of note is CoAP, or Constrained Application Protocol. Like MQTT CoAP is designed for IoT. CoAP is defined by RFC7252, written by the Internet Engineering Task Force (IETF). CoAP’s big points are that it is built for nodes with small amounts of memory, and networks with high error rate, which can be translated to a phone app environment, and it is built for IoT and many, many nodes. Its lightweight attributes include a small header of 4-bytes and requiring as little as 10KB of RAM and 100KB of code space. CoAP is also built on the REST model, and uses a request/response, so being very similar to HTTP can easily interface with HTTP services. And of course, CoAP data can be secured, with the default being 3072-bit RSA keys. It should also be noted that that the power saving aspects of MQTT will likely also apply to CoAP, maybe even more so, which would be important for any battery-operated device using the protocol.

Like MQTT, CoAP has some open libraries that could be used for LogiSteps. The first is Californium. Californium is under the Eclipse Foundation and is well documented and supported. It implements CoAP (RFC 7252), the Observe draft (RFC 7641), the block wise transfers draft (RFC 7959), and DTLS1.2 for security. It provides CoAP-HTTP cross-proxy support. And it is designed as a scalable model for IoT. Another implementation is SpitFireFox on GitHub, which is notable for, like Moquette of MQTT, encapsulating CoAP in Netty, the asynchronous, event-driven framework. This would be a fairly easy implementation to set up and begin using for LogiSteps. CoAP also has a C library, libcoap, that would enable development of CoAP directly on the embedded system.

#### HTTP

Of course, there is HTTP, HTTP/2, and HTTPS. They are the standards of internet communication. HTTP, Hyper Text Transfer Protocol, used by the Internet. HTTP is a stateless protocol, meaning there is no knowledge of what previously occurred on the network, that uses request-response protocol to communicate between client and server, and it operates on TCP/IP. HTTP is not designed with constraints in mind and has very large and complex headers. Another fault is HTTP is not a secure method of data transfer and communication.

However, there are the HTTP upgrades HTTPS and HTTP/2. Both of them are improvements and widely used. HTTPS secures HTTP by using either Transport Lay Security (TSL), or the Secure Sockets Layer (SSL). This and adds protection and privacy to the data, which is something LogiSteps will most likely want for its data. HTTP/2 was created with several goals to improve HTTP. It adds the ability to allow clients and servers to choose their data protocol. It maintains compatibility with HTTP 1.1. And has improved upon the large overhead of HTTP, thus reducing latency. Of note is that HTTP/2 itself is not secure but does support HTTPS using TLS.

As HTTP is the world standard for internet communication there are many more libraries and clients for Android that LogiSteps could utilize for itself. For example, there is the Google HTTP Client Library for Java, supporting Android 1.5 and higher. There is OkHTTP which works as an HTTP & HTTP/2 client for android and has almost 30,000 stars on GitHub. And there is Retrofit which turns the HTTP API into a Java interface. Retrofit also has almost 30,000 stars on GitHub. Due to the wide spread usage of HTTP and all the libraries there are for it.

#### Data Protocol Conclusions

MQTT and CoAP have their advantages as communication protocols for IoT. However, HTTP is a more versatile protocol, and due to being the global standard there are many libraries that make using HTTP very simple and easy to use. Should LogiSteps remove the Android phone app from the design, and instead send the data directly from the embedded device to the web service over LTE, CoAP would be the optimal choice for its IoT benefits and HTTP similarity. However, as LogiSteps does not require the IoT benefits provided by MQTT and CoAP specifically for communication from an Android device, LogiSteps will use HTTP on the Android application due to the benefits provided by HTTP’s widespread use, documentation, and libraries.

### Receiver Conclusions

While receiving data from the microcontroller embedded into the shoe could theoretically be done using LTE or wired connections, a mobile application receiving data over Bluetooth offers significant advantages in terms of power savings and usability. Based on the system that processes and presents the data, a mobile app receiving data could then relay the data to a final destination using higher level communication protocols.

## Data Storage

The LogiSteps product will generate data pertaining to a user’s movement originating from their shoe. Data will be relayed from the shoe’s sensors, to a microcontroller, where actions will need to be taken to move data into long term memory for analysis and presentation. The medium of the long-term storage could be a database able to perform efficient storage and querying on timeseries data, or the data couple simply be placed into a hard drive on a web server or mobile device. Application data is being considered timeseries data since it will likely be indexed and queried using time.

Choosing the right medium for long term storage of application data is highly dependent on the structure of the data, it’s rate of generation, how the data is being used, and other factors such as the ability to quickly access and perform analysis on the data. For LogiSteps, the data storage medium needs to scale well as the number of rows increases with little or no deletions. Additionally, the LogiSteps application will need to query large amounts of data based on the time that the data was generated. A good storage medium for the LogiSteps application is one which allows for easy, efficient querying of data indexed optimized for timeseries data.

Some popular storage technologies that exist for long term storage are databased such as MongoDB, MySQL, PostgreSQL, and simpler technologies like hard drives. Several options for long term storage of LogiSteps application data are compared in the following sections, followed by a conclusion on the best storage technology for LogiSteps.

### SQL Databases

SQL databases, such as MySQL and PostgreSQL, are advantageous in that they make use of powerful, long established standards for querying and manipulating data. As a result, users of the database can access and view subsets and supersets of the database, helping answer analytical questions and present applicable data to a user. Additionally, SQL databases are accessed using a standard SQL language, which can make it easier to access data without having to write a lot of code. While SQL databases can excel in querying and keeping consistent data, they often do not scale well. An experiment done by a TimeScale engineer showed that the insertion rate for a PostgreSQL database decreased at a near linear rate when increasing the number of records in the database, beginning to level out near 400 million rows. This could present a problem as LogiSteps scales up, increasing the number of users logging data into the database. Another disadvantage of using a SQL database is the inability to store complex data structures such as objects; a SQL database requires all columns to be a single scalar attribute. While this likely won’t impact LogiSteps, the inability to scale with dataset size presents a problem.

#### Cost

Pricing can be complicated and depends largely on the database being used and it’s use. For development, most SQL database options are free, and only require payment for commercial uses. As a result, the cost of using a SQL database for senior design would be free. Cost would then be dependent on hosting options if the application were hosted on a web server. Even then, most hosting services provide a free development period.

### NoSQL Databases

A popular database alternative that scales well are noSQL databases such as MongoDB. noSQL databases have taken off in popularity in previous years due to their various advantages over SQL databases such as

* Ability to handle well structured, semi-structured, and unstructured data
* Quick prototyping and development (structure does not need to be well defined)
* Ability to store objects
* Efficient, scale-out architectures

While these advantages can be beneficial, LogiSteps only needs the ability to scale up, as well as quick development and prototyping. In addition to the advantages of a noSQL database,noSQL present several disadvantages as well. Most notably, noSQL databases lack many of the business intelligence and analytic features that SQL databases are able offer or paired with. While MongoDB provides a service for timeseries data, it lags behind more specialized solutions designed for timeseries data.

#### Cost

Like SQL databases, most noSQL databases such as MongoDB provide a free version for developers. Once a product is taken public, MongoDB will charge a price, depending on the needs of the platform. For the length of senior design, noSQL databases should remain free to develop with.

### TimescaleDB

Timescale is a SQL based database which is designed specifically for timeseries data. Specifically, Timescale is built on PostgreSQL, and as a result, is compatible with any tools that work with PostgreSQL. Being built on a SQL database allows Timescale data to benefit from the same advantages of traditional SQL databases, but it is designed to scale like a noSQL database. Essentially, Timescale attempts to bridge the gap and offer some of the most notable benefits of a traditional SQL database and a noSQL database. To achieve scalability similar to that of a noSQL database, Timescale partitions data into time-based chunks which allows for faster performance at scale. Some of the most notable advantages of this database are

* SQL database designed to be compatible with SQL compatible tools
* Scales like a noSQL database
* Comes with built in timeseries specific analytical functions
* Abstracts data as one continuous table for simplified queries.
* Can be managed like a traditional PostgreSQL database
* TimescaleDB integrates directly into the PostgreSQL query planner and execution engine

#### Cost

Timescale offers an installation that does not involve payment. This means that TimescaleDB can be used to develop for free. Additionally, TimescaleDB can be installed on most of the popular dynamic hosting services such as Google Cloud Platform and AWS. TimescaleDB does offer an Enterprise addition of its database, which provides custom deployment assistance and proprietary functions. For its use in senior design, the free install of TimescaleDB should suffice.

### Data Storage Conclusions

LogiSteps aims to provide a user interface for customers to interact with and view their fitness data. As a result, a powerful query language will be needed that will make it possible to easily calculate aggregate data and find subsets of their data based on certain parameters. Additionally, the storage medium used to store user data will likely need to interface with a third-party visualization library or an object relational mapper. Such tools are often designed to interface with SQL databases due to their standardization. These needs are well suited by a SQL database; however, the SQL database will likely not scale well for LogiStep’s timeseries data. A noSQL database would likely be best for scalability but would sacrifice the benefits of a SQL database. To fulfill both requirements, TimescaleDB appears to be the best option for LogiSteps. Timescale DB is a database designed specifically for timeseries data that is built to scale, but is built on, and takes advantage of, the SQL properties in PostgreSQL. Other timeseries databases with similar features, such as influx, may be used if needed.

## Data Processing

Data processing could potentially be performed either on a web server, centralizing all processing. Another option for LogiSteps is to perform all data processing on the mobile phone of each user.

### Mobile Phone

Data processing on a mobile phone has a few advantages, but many disadvantages. Using a mobile app to perform processing of fitness data has a few advantages:

* Always on you - the data will always be there and able to be processed, even when not on LTE
* Clean data – the data sent to a server would already be formatted
* Display faster – the app would not have to pull data to display general information

There are many disadvantages as well. The mobile app could prove to be detrimental in these ways:

* Battery use – processing the data on the mobile app will use more of the user’s phone battery
* Slowdown app – processing the data in the app would slow down the app being running
* App size – the app would greatly increase in size if processing of the data was on it
* Updates – the updates are not mandatory, therefore not doing it could cause problems
* Development effort – Team Omicron would need to develop the same app for multiple platforms

These factors alone may drive users to not use or not want LogiSteps. Those who have older phones and need a charge every few hours due to a bad battery would more than likely not use LogiSteps at all. Additionally, those who do not have a lot of storage space may be unable to download the app.

### Web Server

Processing all data using a web server offers several advantages over performing all data processing on a mobile phone. Many of these advantages are similar to the advantages of using a web interface for presenting data to users. By processing data on a server, LogiSteps could take advantage of the following:

* Single Platform – no need to develop for multiple platforms
* Installation-less – Users do not need to install an application on their phone
* Updates – Updates only need to be published to LogiStep’s web servers
* Administrative overhead – No need to gain approval from companies such as apple
* More powerful hardware – Not limited to processing power of mobile device

To assist with developing back-end servers for processing data in web applications, several back-end frameworks exist to aid in development. A few of the best options available for LogiStep’s project are discussed.

#### Express

Express is a framework that uses NodeJS for development. NodeJS is an event driven, non-blocking I/O language, which makes it ideal for applications driven by user interaction and events. LogiSteps is predicted to be mostly a data driven application, with I/O events occurring in regular, infrequent intervals. Due to this, NodeJS is most likely not an idea language for writing the LogiSteps web application.

Express is also a lightweight framework that comes with support for both SQL and NoSQL databases, allowing easier integration of data for applications. Additionally, Express offers features such as routing, view caching, and middleware chaining. Essentially, express is a minimalist web framework that has as little functionality as possible, while providing a series of middleware function calls for executing code, changing request and response objects, ending the request/response cycle, and calling the next middleware in the pipeline. This gives applications using Express a log of flexibility, without bogging down an application in express specific code. Additionally, by using express and NodeJS, developers have access to NPM – the largest open source library in the world.

#### Django

Django differs from express in a multitude of ways, providing more structure, and utilizing a completely different programming language. Django is a framework that helps build back-end applications using Python, a common language used for data driven applications. While Express was a framework with little requirements for structure, Django enforces rigid application structure, following the MVC pattern for representing objects and state. While a stricter structure requirement can limit the possibilities of an application, Django provides several features that frameworks such as Express and Flask do not provide. In particular, Django has a relational database interface that is built into the framework. This provides access to a build in object-relational mapper, support for SQL database managers, the ability to quickly switch between different DBMS, and allows the application to be closely coupled with the backend database.

Additionally, Django advertised several more features that frameworks such as Express and Flask do not advertise. Some of these features include built in Security (SQL injection, cross-site scripting, cross-site request forgery, clickjacking), user authentication, administrative controls, scalability, and more. Django provides an all-in-one solution, and this helps assist in quicker application development. In frameworks such as Express and Django, these features may only be available through third party plugins and libraries. Django is designed for building data driven applications, such as the web application for LogiSteps.

#### Flask

Flask is a back-end framework for applications written in Python (similar to Django), but with limited functionality (similar to Express). Flask was designed to provide a web framework that is focused on simplicity, minimalism, and fine grain control. As a result, the Flask framework has a smaller community than Django, but is less restrictive. Further, due to Flask’s simplicity, it does not contain an object-relational mapper, which allows for greater flexibility, but more work implementing data driven applications. While Flask is considered a “microframework”, it still provides some of the fundamental features that web frameworks provide, such as minor security, routing, middleware, and scalability.

### Costs

All the mentioned frameworks are free of cost.

### Web Server Conclusions

Performing data processing on a web server offers the best user experience for users of LogiSteps, and significantly reduces development work for team Omicron. By completing processing on a server, LogiSteps will have minimal impact on a user’s phone, and development can be focused on a single platform, with better performance. When considering the best backend framework to use for the development of LogiSteps, several factors were considered. Express is built on NodeJS and is designed for efficient event driven programs. Express also integrates well with NoSQL databases such as MongoDB, which may be beneficial for storing complex data structures. Django and Flask, on the other hand, are built on Python, which offers a core API much more equipped for data processing. While Django and Flask share a common underlying programming language, they are quite different. Django offers a "batteries included" approach, providing several tools for quick and efficient prototyping. Additionally, Django specializes in providing an interface for integrating relational databases into an application. This reduces the effort required for performing CRUD operations and querying for data. Flask on the other hand, offers a bare-bones, minimalism approach. This offers inherit benefits, such as increased freedom and size, but can make it more difficult to rapidly prototype and develop. This provides increased flexibility for the type of database and management systems being used for storing and querying data, but increases the effort required for integrating databases into the application. This project will mostly be a data driven application, serving data visualization views to clients. Because of this, it makes more sense for the back-end technology to utilize the Django framework, which assists in easy data mapping and database querying using the relational object mapper. Using Django, relational databases can be easily integrated into the application, Python can be used for efficient data manipulation, and other Django features such as user authentication, security, and more can be used to support development.

## Data Presentation

### Mobile App

One option for displaying user data is implementing a full UI on a user’s mobile phone. Displaying all the data on the phone offers a few advantages.

* The application has easier access to personal user data
* The user does not need multiple views to see all their data; it is all in one place

Displaying all the data also offers multiple disadvantages.

* The application size goes up
* The application slows down
* The user is not required to update. Therefore, if the format of the data changes, the user view could break due to the server having a different format
* User interfaces must be designed for dozens of different types of mobile phones.

Rather than building a full feature user interface on users’ mobile devices, another option is to display a very basic UI, displaying only a few basic statistics and connection status. This will:

* Help the user save recourses on their phone
* Greatly reduce the development needed.

The only disadvantage to this presentation style would be that the user would have to go online to view their complete data.

### Web Application

An important aspect to LogiSteps is the ability to portray user data generated from their shoes in a clean, easy to use manner. While there may be several approaches for displaying the data (mobile application, native application, etc.), one of the most dynamic and flexible options is using an HTML based web user interface - doing so presents numerous advantages over other display options. While the advantages are numerous and vary from use case to user case, a few important advantages to consider are the following:

* **Cross Platform** – by serving user interfaces from a web server, it allows the application to be designed without worrying about compatibility across Windows, MacOS, and Linux. UX code will render and run in the browser.
* **Installation-less** – Since the user interface is being served from a web server, there is no need to design software for installing and booting the user interface. End users only need to know the right URL for accessing the application.
* **Updates** – Updates can be easily deployed to a centralized location without the need to push updates software to numerous distributed users.
* **Administrative Overhead** – by deploying the end user application to a web server, this project will be able to mitigate much of the administrative overhead placed by companies such as Apple.
* **Fast Prototyping/Development** – The abundant availability of front-end frameworks help get the project into deployable state without needing to design UI components from the ground up.

Of course, there are more advantages than the ones considered, however they help illustrate the case for using a web application for presenting data to users.

While there are numerous advantages to this approach, one must also consider valid disadvantages, and factor them into the decision-making process. The use of a web server for processing and displaying user data lacks behind mobile and native apps when considering the following issues:

* **Processing Overhead** – Using front end and back end frameworks for building an interactive application for a user comes with a lot of extra files. Web applications often require an extensive number of files for rendering and processing data, much of which may not be used if the user does not navigate to that respective part of the application. Additionally, since the application is abstracted from the system, it often runs slower than native desktop and mobile apps. The speed of the application also becomes dependent on the web browser being used, an aspect of design that the developer cannot control.
* **Programming Language** – Since the front-end portion of the application is being run in a browser, development is limited to JavaScript, HTML, and CSS. Additionally, the back-end server will also be limited to a language supported by web frameworks and web servers – typically NodeJS, Python, Ruby, etc.
* **Resources** – The application will not have access to many of the client’s system resources. This will somewhat limit the design flexibility for displaying information to the user.
* **Security** – Resources will be more viable to attack and probe from outside sources. Additionally, data being transmitted over the public internet may be vulnerable to spies. Security becomes a much larger problem when using the public internet to serve and present resources to users.

One last thing to note that by presenting information to users using a web application, the application becomes decoupled, with a distinct separation of client side and server-side software. This can be an advantage if designed correctly, but the server and client may not be able to easily access resources that could be if the application were closely coupled (like a native mobile/desktop app).

Choosing a front-end web framework for a web application can be a difficult task due to the vast multitude of available frameworks – each claiming to be better than the next. After doing initial technology research, team Omicron discovered that three of the most popular front-end web frameworks available to developers are Angular, ReactJS, and VueJS. Other notable front end frameworks worth mentioning are Ember, Elm, knockout, and more.

When considering front-end frameworks, their attributes were closely examined to help determine the framework that would best suit LogiSteps. Each of the comparable attributes are examined in finer detail below.

#### Documentation

Angular, React, and Vue are three of the most popular front-end frameworks in the world. As a result, each has an extensive amount of rich, detailed documentation and tutorials. All three frameworks have the necessary documentation to begin development and reference if there is any trouble in development. Of all three, it is difficult to choose which framework has the best documentation, but it appears that React may lag behind both Angular and Vue in documentation, with simple explanations of the software. This claim appears to be backed up by several members of the software development community, with many claiming that React lacks in some aspects of its documentation. Having little experience in web development, documentation is an essential attribute for Team Omicron to consider when choosing a proper front-end framework.

#### Architecture

Architecture of a framework has a large impact on how flexible a developer’s application can be. Angular is highly opinionated on how a developer’s application should be structured, with React and Vue providing a much more flexible approach, allowing developers to structure their application in a manner that best fits it’s needs. This has much to do with the fact that React and Angular are only View layer libraries, while Angular uses a model, view, view model structure with structured components and services.

Having a strict/strongly opinionated framework can make design work simpler by limiting the developer’s decisions, but it limits the flexibility of the application, while also adding overhead that is often unnecessary. This makes Angular a much heavier framework than Vue and React, introducing a lot of complexity that is likely unneeded. On the other hand, having a structure to develop from can be beneficial in implementing an application. Due to this, both Vue and React have templates and third-party plugins for modeling client-side data and structuring an application. This makes it possible to integrate model layers into a front-end application without requiring an application to adhere by a strict set of rules (as it is in Angular). The benefits of structure can be exploited without dependency on it.

#### Data Binding

Data binding is a property that maps underlying data to the view layer of the application. This attribute relates to the architecture of the framework being used. In frameworks that are primarily the view layer of an application, data modeling is typically one way, from the model to the user interface. Vue and React follow this pattern; data has a downward flow, and in React child elements don’t have any effect on parent data. This can be advantageous because it keeps the application logic simple – data and the state of the data lives in only one place. The disadvantage of this attribute is that it makes it more difficult to directly change the state of data from user interaction. Other tools and features of the framework must be exploited to update the underlying data.

Angular differs from Vue and React in that it offers two-way data binding. Two-way data binding is a powerful feature that allows actions by the user to directly update and change the state of underlying data models. This becomes possible in Angular because of its model and view model layers underneath the view layer of the framework. Although this feature can be powerful in applications, it can lead to bugs that are difficult to test, find, and fix. It also adds extra logic to the application. For applications where data is primarily viewed, with no great need for user manipulation, this feature of the framework is less useful.

Since LogiSteps will be using the user interface to primarily display user data, two-way data binding is likely an unnecessary feature. One-way data binding will most likely provide a cleaner conduit for conveying data to the end user.

#### Performance and Load Times

A noticeable metric for an end user of a web application is the performance of the web page and how quick the initial load time is. John Hannon did a study on the performance and load times for various front-end web frameworks, and he found that for keyed implementations of the frameworks, Vue beat out Angular and React significantly in both typical operations as well as initial load times. Angular and React appeared to perform similarly for typical operations, with React slightly edging out Angular in most metrics. When comparing initial load times, React was significantly faster than Angular, but still significantly slower than Vue.

The web application for LogiSteps will not be an enormous web application performing complex algorithms. Due to this, there is an expectation that the web page should be fast and respond to users quickly. The page should also load quickly when a user navigates to the page to view their statistics. For this metric, Vue beats both Angular and React.

#### Syntax

Syntax is an important aspect of a framework, as it directly correlates to the learning curve associated with the framework; the more unfamiliar a syntax is, the longer it will take to begin development of the application. Angular and React present syntaxes unlike others found in other languages and frameworks. React represents all UI elements using JSX and rendering functions. JSX is an XML like syntax that is used within JavaScript. While JSX can be powerful, allowing JavaScript to be interwoven with UI, it means that using React would require learning a new language for expressing UI elements, presenting a steep learning curve to all members of Team Omicron. Vue, on the other hand, supports JSX, but allows UI elements to be expressed with standard HTML, CSS, and JavaScript.

Angular, like React, provides a rich syntax that is complex in nature. Additionally, Angular makes use of Typescript rather than JavaScript. This means developers using Angular may need to be learn the Angular specific syntax for element expression, as well as Typescript for data modeling. This presents a large learning curve to developers on Team Omicron. Vue keeps syntax simple, using HTML, CSS, and JavaScript for UI element expression, while also picking a select few expressions used in AngularJS for more powerful UI expression.

Vue provides a simple syntax for its core functionality, while expanding upon the strengths of other frameworks to allow for more powerful element expression and HTML based templates.

#### Community Support

An important aspect of a front-end framework is the amount of third party and community support that the framework has. This metric was checked by team Omicron by comparing GitHub statistics. Angular, React, and Vue have the following community activity.

|  |  |  |  |
| --- | --- | --- | --- |
| Metric | Angular | React | Vue |
| Github Stars | 40,903 | 111,824 | 114,636 |
| Github Contributors | 731 | 1,241 | 193 |
| Stack Overflow Tags | 132,505 | 103,665 | 23,029 |

Chart 1 - Community Support Statistics for Angular, React, and Vue.

From the metrics collected in table 1, Angular and React appear to have massive following in the developer community. Angular has a significantly lower GitHub star count than the others, but this is likely a consequence of Angular migrating versions. Both Angular and React also have a large amount of questions on stack overflow, meaning that many potential questions likely have answers available on stack overflow. Vue has a significantly lower number of GitHub contributors and stack overflow questions, possibly making it more difficult to find answers to questions during the development process.

#### Other

Angular, React, and Vue all provide several other features as well which have been considered when performing technology research. All three frameworks provide view templating routing, and expressive element expression. When considering framework transfer size, Vue wins, beating both React and Angular based on a study performed by Jacek Schae.

#### Costs

All three options (Angular, React, and Vue) are free to use and develop with.

### Data Presentation Conclusions

Presenting user data using a web-based interface offers the most benefit to Logisteps. Not only does it simplify the development effort required, but it minimizes the amount of resources that the application would require from a user’s phone. A web-based app makes it easier to build cross-platform interfaces, mitigate much of administrative tasks to publish an app to an app store, develop for multiple screen sizes, and more.

The best front-end framework for development of the web-based user interface was decided using the following criteria. All 3 front end frameworks provide similar speed performance, however, Angular typically has a longer initial load time. In terms of learning curve, both Angular and React have steep learning curves, with Angular having complex syntax, and React using JSX for UI development. Vue offers the smallest learning curve by using native JavaScript, CSS, and HTML to implement rich, expressive user interfaces. Vue also has advantage in terms of size - due to Angular's heavy use of libraries, complex syntax, and other features, it becomes bloated. Both Vue and React are much smaller in size, with libraries being included as they are required. Angular provides more structure, but this structure can be restrictive at times; React and Vue allow for much greater flexibility. The flexibility of Vue and React can sometimes make it more difficult to start projects, but Vue offers a powerful CLI that helps setup projects with various configuration settings available. Vue is currently less popular than Angular and React - leading to a smaller supporting community and a smaller collection of libraries, but Vue is quickly growing in community size and popularity. In the end, Vue likely offers the best option for the project due to its small size, and smaller learning curve. Team Omicron’s front-end UI is meant to be simple, providing a portal for viewing statistics. The added size of Angular, and complexity of React and Angular make them a less attractive option for the project.

## Web Hosting Overview

Some of the options for processing, storing, and presenting data to users would rely on a publicly hosted website. A critical piece for creating the web applications is hosting the application publicly so that any user across the world can access it. While it is technically feasible to host a dynamic website on a development PC or server, it would be vulnerable to a multitude of security concerns and would not be able to scale well. To solve this issue, there are a multitude of web hosting services that allow a developer to host their database, web server, and website publicly. Some of the top competitors in this industry are Google Cloud Platform, Amazon Web Services, and Heroku. All 3 of these options offer free versions for active development.

Google Cloud Platform and Amazon Web Services are similar in that they offer infrastructure for hosting web applications as a service. This allows a developer to choose the correct plan based on the needs of the application and offers limited free trials for initial development. Amazon web services have been around the longest of all 3 options (Heroku actually uses Amazon Web Services), but often it can be difficult to predict the price for hosting a web application on Amazon. Additionally, based on experimentation, Amazon Web Services appear to be the most difficult of the three options for navigating and finding all features. AWS claims to have free tier with the following features:

* Amazon Cognito – Mobile user identification and synchronization
* Amazon DynamoDB – 25 GB NoSQL database
* Amazon EC2 – 750 hours of cloud compute capacity per month
* AmazonMQ – 750 hours of broker service for Apache ActiveMQ per month
* AmazonRDS – 750 hours of Managed relation database service for MySQL, PostgreSQL, etc.
* Elastic Load Balancing – 750 hours of traffic distribution

The Google Cloud Platform also has a free tier that gives a 12-month $300 credit to developers. Google Cloud offers many of the same features as AWS, but with a more intuitive interface for developers. One of Google Cloud’s most valuable features is its Compute Engine, which provides scalable, high performance virtual machines. Using this, custom software can be installed and used to host a web application. Google Cloud also offers services for load balancing applications, removing a necessary piece of deployment. Additionally, Google Cloud offers several predefined production environments that a web application can be deployed to, removing the necessary administrative tasks involved with a custom web stack. While Google offers a $300 credit to its customers, the following services are always free

* Google App Engine – platform for building scalable web applications
* Highly-scalable NoSQL databases
* Google Compute Engine – scalable, high performance virtual machines (web hosting too)
  + 30 GB of HHD
  + 1 f1-micro instance per month
* Google Cloud Pub/Sub – real-time and reliable messaging and streaming data
* Cluster Management
* Google Stackdriver – monitoring, logging, and diagnostics for applications on Cloud Platform
* GCP Marketplace – pre-configured free production grade solutions

### Pricing

Google cloud will likely be free for the duration of Senior Design due to the requirements of the web application. If LogiSteps were scaled up, pricing would be heavily dependent on the number of users. A server with 4 cores and 15GB of memory would cost $97.09 per month, and a server with 1 core and 3.75GB memory would cost $24.27 per month.

Heroku is a cloud application platform that supports building, deploying, and managing apps. Heroku allows apps to be run inside of what they call dynos, which are fully managed runtime environments for applications. Heroku is designed to be as easy as possible for developers to easily, quickly, and frequently deploy applications; Heroku then provides a dashboard for managing all applications. Heroku helps handle scale, similar to Amazon Web Services and Google Could Platform. Heroku offers several different database solutions such as Heroku PostgreSQL but requires extensions and add-ons to work with other database systems, making it slightly more difficult to deploy data driven applications. Heroku is likely an excellent source for active development but appears to lack the extensive functionality that can be achieved using a Google Cloud Compute virtual machine. Heroku also sleeps after inactivity for free accounts, leading to large latency for requests made when the dyno is sleeping.

* Free - $0
  + Core platform features
  + Sleeps after 30 minutes of inactivity
  + 512 MB RAM | 1 web/1 worker
* Hobby Level - $7/dyno/month (pay for time used to the second)
  + Core platform features
  + Never sleeps
  + Free SSL & Automated Certificate Management
  + 512 MB RAM | 10 Process Types
* Standard - $25 - $500/dyno/month (pay for time used to the second)
  + All hobby features+
  + Horizontal scalability
  + Threshold alerts
  + Preboot
  + 512 MB RAM or 1 GB RAM
* Performance - $25 - $500/dyno/month (pay for time used to the second)
  + All standard features+
  + Dedicated
  + 2.5GB RAM or 14GB RAM
  + Infinite process types

### Web Hosting Conclusions

For the development of LogiSteps, Google Cloud Platform will likely be the best option. GCP not only offers a free plan to begin development, but it also offers a robust administrative web interface for managing the deployment. Google Cloud also presents a more straightforward pricing model which coincides with a pricing calculator to help determine cost of hosting, depending on the needs of the web application. Additionally, Google’s compute Engine allows a customized web stack to be developed and deployed without having to worry about compatibility. This presents a huge advantage over services such as Heroku. It is worth noting that there are several other web hosting services which appear to be extremely cheap up front (such as hostgator), but these services often attempt to lock customers into long term deals and offer limited hosting options, severely limiting the web stack.

## Block Diagram with Available Component Technologies

Analyzing how the different approaches to each section may affect each other, a more complex block diagram has been developed to help visualize possible permutations of each technology that would satisfy design requirements. This diagram is shown in figure 5

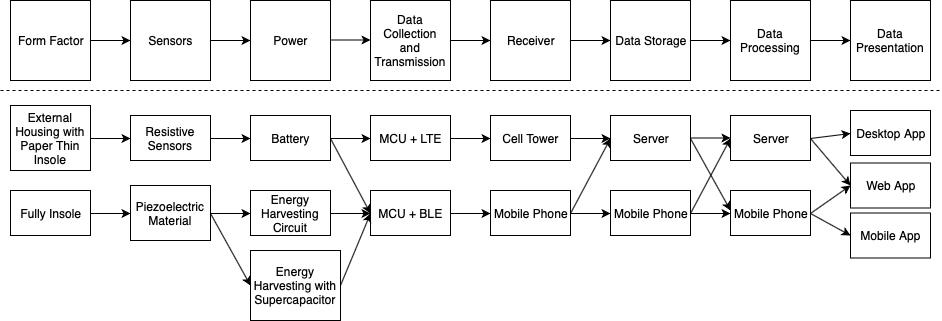


Figure 5 - General system requirements with technological components that make it possible to achieve system requirements. Several options are shown for each system requirement.

# Final Design Approach

The final design has taken on a very modular approach, allowing for some freedom between design choices should changes need to be made down the road. Some of the choices affect others, such as resistive sensors requiring an energy dense power source rather than an energy harvesting solution, or LTE to server data storage technique making a mobile app no longer useful. Overall, the top choices in each category work together and are most likely to offer the best solution, and issues to be encountered won’t affect other choices in a breaking manner. A full stack view of the system using the preferred approaches for each system requirement is shown in figure 6.

## Final Block Diagram

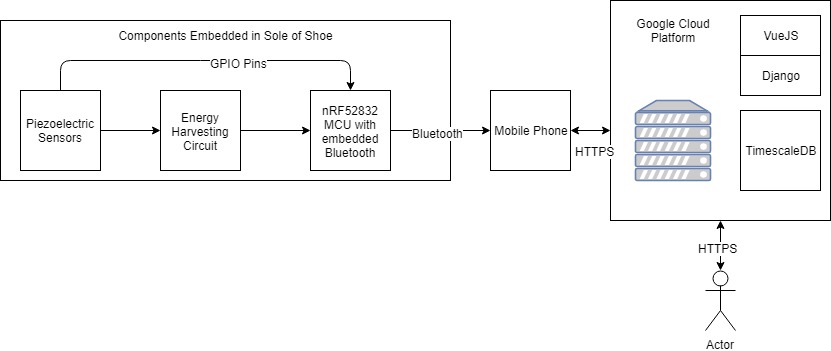


Figure 6 - Final design approach, using the technologies discussed in previous sections of this report that were concluded to be the best options for meeting system requirements.

## System Requirement Satisfaction

The solutions that have been chosen to be pursued will entirely meet design requirements as previously specified. The piezoelectric sensors will be able to provide useful data pertaining to the users' steps, including pressure sensitive readings. Those type of sensors will also be able to power the system through an efficient energy harvesting technique, and with the thin, fully insole form factor the user experience will be as seamless as possible. The MCU and BLE capabilities will allow the data to be collected and transmitted from the shoe to a receiver in real time. The phone acting as the receiver allows the system to require no more than the insole alone, and it enables a simple solution to transmit data to a server for permanent storage and processing. The timescale database solution allows for efficient permanent storage of user data over time. Choosing Django as the server backend allows for efficient processing of data and choosing VueJS as the front end lets the processed data be professionally and simply presented to the user.

## Modularity

Rather than highlighting 3 specific approaches that the project could have, the project design was split into a more modular view. Since the project contains a lot of different modules working together, it was decided it would be better to give information about the various options for each module.

Having a modular design for the project will help as prototyping is begun, and progress made with the project. If any problems are encountered with the design and a need to change the approach arises, it will be able to be switched to another option for that module without having to drastically change the rest of the project design. Since the project has a lot of pieces working together, it was desired to make it easy to change the individual approaches if necessary.

## Total Cost

The development costs were estimated using the minimum and maximum costs determined by the analysis of each approach. The preferred approach costs were determined using the preferences for each module's solution, in addition to conservative prototyping plans.

|  |  |  |  |
| --- | --- | --- | --- |
| **Development** | **Min ($)** | **Max ($)** | **Preferred**  **Approach ($)** |
| Form Factor | 40 | 70 | 70 |
| Sensors | 30 | 400 | 133 |
| Energy Components  (pure harvesting) | 20 | 20 | 20 |
| Energy Components  (general) | 10 | 40 | 40 |
| Microcontroller/BLE | 60 | 150 | 60 |
| Server Costs | 0 | 0 | 0 |
| Solution Totals | 160 | 680 | 323 |
| Labor | 55,500 | 111,000 | 111,000 |

Table 2 - Summary of development costs for LogiSteps.

Should this product be taken to market, production costs would need to be estimated as well. The component costs would go down significantly in each category per device due to whole sale prices, but some additional costs would arise such as licensing and hosting fees. The hosting fees are difficult to determine due to the plethora of options available, more would need to be known about the customer base and specific service requirements to get a good estimate of necessary processing power and storage needs. The Bluetooth branding fee is estimated to be $8000. The Google App store license fee is $20. The cost of mass production also needs to be considered, this would require outsourcing the production of the final product design to a manufacturer and distributor.

# Conclusion

In conclusion, the final system approach discussed in the previous section satisfies all system requirements. In addition, thorough research has been performed for each component of the system, so if an approach is found to be unfeasible, the Team Omicron will be able to quickly change one piece of the system design, without needing to perform a complete system redesign. Each aspect of the final approach is defended by the research documented in the component technologies of this report. Team Omicron is optimistic that using the technology and concepts discussed in this report will result in a switch, and successful prototype and design cycle.

# Glossary of Technical Terms

**Piezoelectric** – electricity or electric polarity due to pressure especially in a crystalline substance.

**Bluetooth** – a standard for the short-range wireless interconnection of mobile phones, computers, and other electronic devices.

**Server** – a computer or computer program that manages access to a centralized resource or service in a network.

**Cadence** – the total number of steps you take per minute.

**Wearables** – an item that can be worn.

**AD converter** – A circuit that translates an analog signal into a digital binary format.

**Microcontroller** – A small computer or integrated circuit that contains memory and input/output peripherals.

**Web application** – an application that is stored on a remote server and served to a user over the internet in a web browser.

**Embedded Systems** – A combination of hardware and software designed for a specific application, often small in size.

**e-textiles** – fabrics that enable digital components and electronics to be embedded in them.

**Android** – An open-source operating system for mobile devices.

**C** – General purpose programming language ideal for development on embedded systems.

**Java** – High level, object-oriented programming language with high portability using the Java Virtual Machine

**Python** – high level general purpose programming language.

# References

Amazon. (2018). *Amazon EC2 Pricing.* Retrieved from https://aws.amazon.com/ec2/pricing/on-demand/

Amazon. (2018). *AWS Free Tier.* Retrieved from https://aws.amazon.com/free/?awsf.Free%20Tier%20Types=categories%23featured

Android. (2018). *Bluetooth overview.* Retrieved from https://developer.android.com/guide/topics/connectivity/bluetooth

Android. (2018). *Connect to the network.* Retrieved from https://developer.android.com/training/basics/network-ops/connecting

Angular. (2018). *What is Angular?* Retrieved from https://angular.io/docs

Django. (2018). *Django Homepage.* Retrieved from https://www.djangoproject.com/

Eclipse. (2018). *Eclipse Californium (Cf) CoAP Framework.* Retrieved from https://projects.eclipse.org/projects/technology.californium

Eclipse. (2018). *Eclipse Mosquitto™.* Retrieved from https://mosquitto.org/

eSun. (2018). *eSUN eLastic TPE 85A 1.75mm Flexible 3D Printer Filament Natural 1kg (2.2lbs) Spool, Natural White.* Amazon. Retrieved from https://www.amazon.com/eSUN-eLastic-Flexible-Printer-Filament/dp/B01A4WP4AY/ref=sr\_1\_1\_sspa?ie=UTF8&qid=1538631121&sr=8-1-spons&keywords=esun+elastic&psc=1

Express. (2018). *Express Homepage.* Retrieved from https://expressjs.com/

Flask. (2018). *Welcome to Flask.* Retrieved from http://flask.pocoo.org/docs/1.0/

Google. (2018). *Google Cloud Platform Free Tier.* Retrieved from https://cloud.google.com/free/

Google. (2018). *Google Compute Engine Pricing.* Retrieved from https://cloud.google.com/compute/pricing

Google. (2018). *Using the Google HTTP Client Library for Java on Android.* Retrieved from https://developers.google.com/api-client-library/java/google-http-java-client/android

Hannah, J. (2017). *Which JavaScript Frameworks Are the Fastest?* Javascript Report. Retrieved from https://javascriptreport.com/js-frameworks-fastest/

Heroku. (2018). *Simple, flexible pricing.* Retrieved from https://www.heroku.com/pricing

IBM. (2018). *Introduction to MQTT.* Retrieved from https://www.ibm.com/support/knowledgecenter/en/SSFKSJ\_7.5.0/com.ibm.mm.tc.doc/tc00000\_.htm

IETF. (2014). *The Constrained Application Protocol (CoAP).* Retrieved from https://tools.ietf.org/html/rfc7252

Instruments, T. (2019). *2.4-GHz Bluetooth low energy System-on-Chip.* Texas Instruments. Retrieved from http://www.ti.com/lit/ds/symlink/cc2540.pdf

Kiefer, R. (2017). *TimescaleDB vs. Postgres for time-series: 20x higher inserts, 2000x faster deletes, 1.2x-14,000x faster queries.* Timescale. Retrieved from https://blog.timescale.com/timescaledb-vs-6a696248104e

Kiefer, R. (2018). *How to store time-series data in MongoDB, and why that’s a bad idea.* Timescale. Retrieved from https://blog.timescale.com/how-to-store-time-series-data-mongodb-vs-timescaledb-postgresql-a73939734016

Kleine, O. (2018). *CoAP Client for Android based on nCoAP.* Github. Retrieved from https://github.com/okleine/spitfirefox

Linear Technology. (n.d.). *Nanopower Energy Harvesting Power Supply.* Retrieved from http://www.analog.com/media/en/technical-documentation/data-sheets/35881fc.pdf

MatterHackers. (2018). *Taulman Black PCTPE - 1.75mm (0.45kg).* Retrieved from https://www.matterhackers.com/store/l/taulman-black-pctpe-filament-175mm/sk/MF1SV5MP?rcode=GAT9HR&gclid=Cj0KCQjw0dHdBRDEARIsAHjZYYDA6VmY7QY3KDw9Un3oK\_78S63tapgvDIWmhVeBvXaZuuQeOh6J078aAq30EALw\_wcB

MongoDB. (2018). *Advantages Of NoSQL.* Retrieved from https://www.mongodb.com/scale/advantages-of-nosql

MQTT. (2018). *MQTT Homepage.* Retrieved from https://mqtt.org/

Nicholas, S. (2012). *Power Profiling: HTTPS Long Polling vs. MQTT with SSL, on Android.* Retrieved from http://stephendnicholas.com/posts/power-profiling-mqtt-vs-https

Nordic Semiconductor. (2018). *Online Power Profiler.* Retrieved from https://devzone.nordicsemi.com/power/

PUI Audio, Inc. (2018). *BUZZER ELEMENT PIEZO 1.2KHZ 50MM.* Digi-Key. Retrieved from https://www.digikey.com/product-detail/en/APS4812B-LW100-R/668-1190-ND/1738483

PUI Audio, Inc. (2018). *PIEZO CERAMIC BENDER 30V 1300HZ.* Digi-Key. Retrieved from https://www.digikey.com/product-detail/en/AB4113B-LW100-R/668-1409-ND/4147333

React. (2018). *React Homepage.* Retrieved from https://reactjs.org/

Schae, J. (2018). *A Real-World Comparison of Front-End Frameworks with Benchmarks (2018 update).* Medium. Retrieved from https://medium.freecodecamp.org/a-real-world-comparison-of-front-end-frameworks-with-benchmarks-2018-update-e5760fb4a962

Semiconductor, N. (n.d.). *nRF52832 Product Specification v1.3.* Nordic Semiconductor. Retrieved from https://www.mouser.com/datasheet/2/297/nRF52832\_PS\_v1.3-1117956.pdf

Shareef, A., Goh, W. L., Gao, Y., & Narasimalu, S. (n.d.). *Synchronous Electric Charge Extraction for Low.* IEEE. Retrieved from https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7829690&tag=1

Silicone Solutions. (2018). *SS-8112 Fast Cure Liquid Silicone Rubber - Per Pound - GE RTV 8112 Offset.* Retrieved from http://siliconesolutions.com/ss-8112.html?\_vsrefdom=adwords&gclid=Cj0KCQjw0dHdBRDEARIsAHjZYYB4GZ9MGousYpa0Q5rTiDGB3IViZbp3X2PF8dFdV0rzuUL9G7smgIMaAhqfEALw\_wcB

Song, H., Huang, X., Jiang, X., & Wang, J. (n.d.). *The synchronized switch harvesting circuit on.* IEEE. Retrieved from https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5768727

Song, J., Zhao, G., Li, B., & Wang, J. (2017). *Design optimization of PVDF-based piezoelectric energy harvesters.* Heliyon. Retrieved from https://www.sciencedirect.com/science/article/pii/S240584401731263X#fig0005

Sparkfun. (2018). *SparkFun ESP32 Thing.* Retrieved from https://www.sparkfun.com/products/13907

Sparkfun. (2018). *SparkFun nRF52832 Breakout.* Retrieved from https://www.sparkfun.com/products/13990

Square. (2018). *OkHttp.* Retrieved from https://square.github.io/okhttp/

Systems, E. (2016). *ESP32 Datasheet.* Retrieved from https://cdn.sparkfun.com/datasheets/IoT/esp32\_datasheet\_en.pdf

TE Connectivity Measurement Specialties. (2018). *PIEZO FILM SHEET.* Digi-Key. Retrieved from https://www.digikey.com/product-detail/en/te-connectivity-measurement-specialties/1-1004347-0/223-1320-ND/5277280

TE Connectivity Measurement Specialties. (2018). *SENS PIEZO PLYCBL.* Digi-Key. Retrieved from https://www.digikey.com/product-detail/en/te-connectivity-measurement-specialties/1005801-1/223-1223-ND/4862192

Texas Instruments. (2014). *CC253x System-on-Chip Solution for 2.4-GHz User's Guide.* Retrieved from https://www.ti.com/lit/ug/swru191f/swru191f.pdf?HQS=TI-null-null-mousermode-df-pf-null-wwe&DCM=yes

VueJS. (2018). *Getting Started.* Retrieved from https://012.vuejs.org/guide/

Zhao, J., & You, Z. (2014). *A Shoe-Embedded Piezoelectric Energy Harvester for Wearable Sensors.* NCBI. doi:10.3390/s140712497