

6-14-2018

# Pet Watch

JB Anderson

*Santa Clara University*, jbanderson@scu.edu

Rachel Hale

*Santa Clara University*, rhale@scu.edu

Follow this and additional works at: [https://scholarcommons.scu.edu/cseng\\_senior](https://scholarcommons.scu.edu/cseng_senior)



Part of the [Computer Engineering Commons](#)

---

## Recommended Citation

Anderson, JB and Hale, Rachel, "Pet Watch" (2018). *Computer Engineering Senior Theses*. 115.

[https://scholarcommons.scu.edu/cseng\\_senior/115](https://scholarcommons.scu.edu/cseng_senior/115)

This Thesis is brought to you for free and open access by the Engineering Senior Theses at Scholar Commons. It has been accepted for inclusion in Computer Engineering Senior Theses by an authorized administrator of Scholar Commons. For more information, please contact [rscroggin@scu.edu](mailto:rscroggin@scu.edu).

**SANTA CLARA UNIVERSITY**  
**DEPARTMENT OF COMPUTER ENGINEERING**

Date: June 12, 2018

I HEREBY RECOMMEND THAT THE THESIS PREPARED UNDER MY SUPERVISION BY

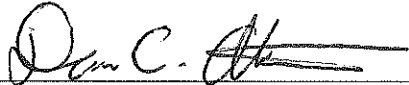
**JB Anderson**  
**Rachel Hale**

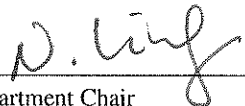
ENTITLED

**Pet Watch**

BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

BACHELOR OF SCIENCE IN COMPUTER SCIENCE AND ENGINEERING

  
\_\_\_\_\_  
Thesis Advisor

  
\_\_\_\_\_  
Department Chair

# **Pet Watch**

by

JB Anderson  
Rachel Hale

Submitted in partial fulfillment of the requirements  
for the degree of  
Bachelor of Science in Computer Science and Engineering  
School of Engineering  
Santa Clara University

Santa Clara, California  
June 14, 2018

# **Pet Watch**

JB Anderson  
Rachel Hale

Department of Computer Engineering  
Santa Clara University  
June 14, 2018

## **ABSTRACT**

This paper outlines our project of building Pet Watch. Pet Watch is a device similar to a Fit Bit except that it tracks your pet's activity instead of your own. You can then access this data on our website. This paper defines our requirements, how the system works, and how we built this system.

## **Acknowledgements**

First and foremost, we have to thank our advisor Dr. Darren Atkinson. Without his assistance and dedicated involvement in every step throughout the process, this project would have never been accomplished. We would also like to thank Dr. Behnam Dezfouli and Dr. Andrew Wolfe. The conversation we had with Dr. Dezfouli was invaluable to getting the wifi to work.

We want to acknowledge the school for funding this project and teaching us so much over the past four years. We especially wanted to thank the folks at the maker lab for making it so easy to use the 3D printers. We also wanted to express our gratitude towards Matt Belford, Will McMullen, and Keith Dorais who helped design our case.

We can't forget to mention our pet volunteers: Ellie, Stevie, and Lucy. Getting to play with you three made building this device worth it.

# Table of Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Development</b>	<b>1</b>
2.1	Planning and Specification . . . . .	1
2.2	Prototype 1 . . . . .	3
2.3	Prototype 2 . . . . .	4
<b>3</b>	<b>Testing</b>	<b>5</b>
<b>4</b>	<b>Collar Design</b>	<b>6</b>
4.1	Temperature Sensor . . . . .	6
4.2	Accelerometer . . . . .	7
4.3	Wifi Controller . . . . .	7
4.4	Integration . . . . .	8
<b>5</b>	<b>Server Design</b>	<b>8</b>
5.1	Collar Server . . . . .	8
5.2	Front End Server . . . . .	8
<b>6</b>	<b>Societal Issues</b>	<b>9</b>
6.1	Ethical . . . . .	9
6.2	Social . . . . .	9
6.3	Political . . . . .	9
6.4	Economics . . . . .	9
6.5	Health and Safety . . . . .	10
6.6	Manufacturability . . . . .	10
6.7	Sustainability . . . . .	10
6.8	Environmental Impact . . . . .	11
6.9	Usability . . . . .	11
6.10	Lifelong Learning . . . . .	11
6.11	Compassion . . . . .	11
<b>7</b>	<b>Conclusion</b>	<b>11</b>
<b>A</b>	<b>Source Code</b>	<b>14</b>
A.1	Collar code . . . . .	14
A.1.1	Collar Controller . . . . .	14
A.1.2	Wifi controller . . . . .	15
A.1.3	Temperature Sensor . . . . .	18
A.1.4	Accelerometer . . . . .	20
A.2	Collar webservice code . . . . .	21
A.2.1	Connection . . . . .	21
A.2.2	log . . . . .	23
A.2.3	database configuration . . . . .	23
A.2.4	database setup . . . . .	24
A.3	Frontend User Webpages . . . . .	24
A.3.1	Home Page . . . . .	24
A.3.2	owner login . . . . .	26
A.3.3	owner register . . . . .	27
A.3.4	pet register . . . . .	28
A.4	Backend User Scripts . . . . .	30
A.4.1	owner login . . . . .	30
A.4.2	owner register . . . . .	31

A.4.3	pet history	31
A.4.4	pet register	33
<b>B</b>	<b>Specification</b>	<b>34</b>
B.1	Functional Requirements	34
B.2	Non-Functional Requirements	35
B.3	Design Constraints	35

## List of Figures

1	Owner Activity . . . . .	2
2	Use Cases . . . . .	2
3	Architecture Diagram . . . . .	3
4	Development Timeline . . . . .	4
5	collar wiring . . . . .	6

## List of Tables

1	Risk Analysis . . . . .	3
2	Final Technologies Used . . . . .	5
3	Material costs for petwatch . . . . .	10



# 1 Introduction

In today's society, people care about their pets more than ever before. However, people have responsibilities and cannot spend their entire day monitoring their pets. Furthermore, owners are not trained to be dog or cat whisperers, and consequently do not always know when a dog or cat is not getting enough exercise or is overheating during hot summer days.

Many mobile devices designed to monitor people have recently exploded on the market. Fitbit is one of the most popular health watches on the market [1]. The L.A. Rams placed GPS chips in practice pads to monitor and fine tune athletes practices. The Rams believe this decision is one of the reasons they had a very healthy team in 2016 in a sport where injuries occur every day [2]. Although these designs are useful, they are designed for people, not pets. Consequently, the activity monitoring algorithms employed by these devices will not be accurate for pets. Recently, several startup companies have started implementing smart technology in dog collars, but these products are still in their infancy. Also, they can be very buggy, expensive, many require monthly service fees, and some are too heavy for small dogs. The FitBark Dog Activity Monitor provides a lightweight, waterproof, solution with a long battery life, but it only tracks one health statistic: a dog's movement. In addition, the use of bluetooth requires the owner to be within 30 feet of their pet, limiting an owner's ability to monitor their pet from a distance [3].

We will develop an IoT device to monitor the health and wellbeing of pets, and dogs in particular. This design will eliminate the need for monthly service and subscription fees that many pet owners face when purchasing a smart device for their pets. Our solution will implement multiple sensors to monitor the pet's activity including temperature and eating habits. This data will be uploaded to the cloud periodically, which will enable owners to check up on their pets using their phone or a computer while away from the house. This will solve one of the primary concerns with devices such as the FitBark, which cannot provide data to pet owners throughout the day. We will consult veterinarians in the design of our monitoring algorithms to ensure we can accurately measure and assess the health of the pets. We want to take the raw data we receive from the device and transform it into meaningful visuals that owners can take action on to improve the health and happiness of their pet. This design mimics the human monitoring capabilities of devices like the Fitbit, while being pet friendly. Owners will find peace of mind knowing that they can monitor the health and safety of their pets from their workplace or home.

To meet these goals, we decided on several crucial objectives to guide our work throughout development. These include initial planning of specifications, implementation of each individual component of the Pet Watch, creating a back and front end and solidifying the data pipeline, and finally testing our design on pets and owners. With these objectives in mind, we hope to successfully implement a Pet Watch that satisfies the goals we have outlined for our product.

## 2 Development

### 2.1 Planning and Specification

Before we began any implementation of our Pet Watch, we needed to come up with a plan. We began with a rough idea of what we wanted: a device to track a pet's activities throughout the day, and easily provide this information on a website for pet owners to view. We needed to create a well defined concept of what we wanted our Pet Watch to be, including requirements, use cases and activity diagrams, conceptual models, and an idea of the architecture and technologies we would use. In order to organize these ideas, our first objective was to create our specifications in the form of a design report.

After deciding on a general concept of our design, we solidified our requirements. The formal list of our requirements can be viewed in Appendix B. This was essential to ensuring that we had a well thought out design with well-defined requirements whereby to measure our success. Once we had our requirements outlined, we were able to develop the activity diagrams and use case diagrams shown in **Figure 1** and **Figure 2** to understand how our device would be used.

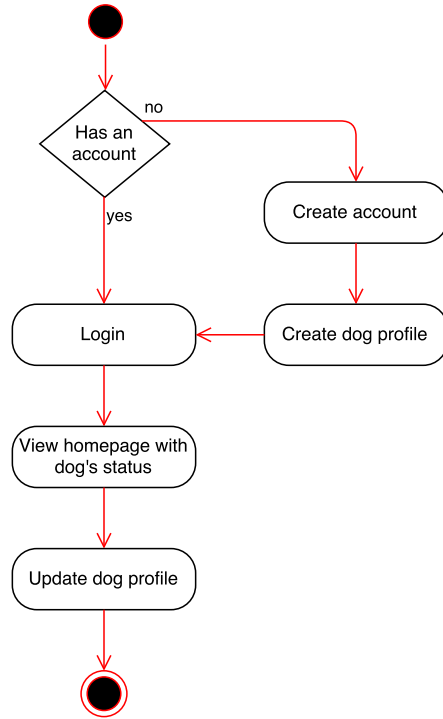


Figure 1: Owner Activity

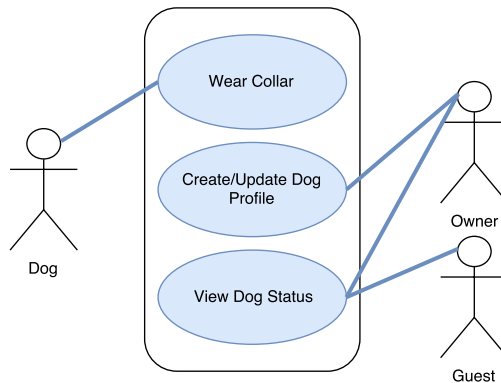


Figure 2: Use Cases

Using these diagrams as a reference, we had a more clear understanding of the uses of our Pet Watch, and could decide on which technologies we would need. First we created our architectural diagram[4, p. 132-137], providing us with an idea of the communication strategies we would implement between our devices, shown in **Figure 3**.

Since we were making a web application, we knew we would need a lot of the typical web technologies. We decided to use HTML[5, p. 15,28], CSS[5, p. 90], and JavaScript[5, p. 462] to build the front-end since we already knew how to use these technologies to set up web pages in the design center[6]. We decided to store all the information in a MySQL database[5, p. 465] on a server in the Engineering Computing Center, since we were familiar with managing SQL databases through PHP scripts[5, p. 28] and the school provides us with a free account. MySQL also abstracts some of the details of implementing a database.

As for the dog collar, we decided to rely heavily on Arduino technology because we had experience working with Arduinos [7]. This meant using an Arduino board for the brains of the collar and attaching multiple Arduino compatible

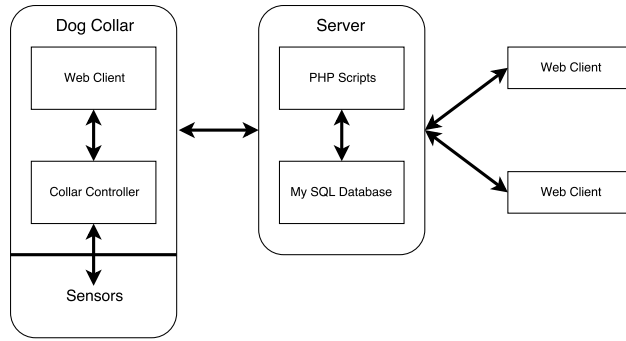


Figure 3: Architecture Diagram

Table 1: Risk Analysis

Risk	Consequences	Probability	Severity	Impact	Mitigation
Work Stoppage	We will have to sacrifice our scope, quality, or work overtime	25%	1-8	25-200	Save work on Github Communicate clearly and often
Users don't find our product useful	Nobody will use our product	10%	2	20	Involve dog owners in the development process ask for user feedback constantly
Device breaking	We will have to replace the parts, which increases our budget and takes time to ship	50%	1-4	40-200	Ensure case durability before pet testing
Insufficient Technical Skills	Either learn the skills or pivot to use more known skills	40%	1-4	40-160	Plan for less known technologies having longer development time. Start on them early
Device too bulky	Only larger pets will be able to use the device	25%	5	125	rapidly prototype collar designs early on
Going overbudget	We will need to find project sponsors or go into our own wallets	10%	2	20	Follow our already approved budget proposal

components: a temperature sensor, accelerometer, and a Wifi component. We decided to use SPI as opposed to I2C for our accelerometer communication protocol[8, p. 16] since it provided a more reliable connection and faster data transfer. It is also full-duplex, allowing for simultaneous communication. We decided to use a 3-D printer to build our case for the Pet Watch so we could customize the design of the collar and iterate through collar versions quickly.

Once we had a more complete concept of our final design and understanding of the scope of the projet, we performed risk analysis on our project. Our analysis of potential risks is shown in **Table 1**, including the probability of each risk occuring, and the severity and consequences if it did occur. Some of our severity is based on a range, since the risks could have a range of impacts. We multiplied our probability and severity to get an impact score, and developed mitigation strategies for each risk. This allowed us to plan ahead and reduce the likelihood of certain risks occurring as well as minimize the damage if a certain risk did occur.

Finally, we created a timeline for our development to help us stay on track based on minimizing the identified risks. In order to effectively organize the development of our project, we used a Gantt Chart highlighting the timeline and completion of each deliverable. Our development timeline, shown in **Figure 4**, includes all of the stages of our project that needed to be completed, as well as the deadline for completing each one. All of these deliverables from the planning phase of our design process helped us to plan out and implement our project in a more successful and organized fashion.

## 2.2 Prototype 1

When we finally began our implementation of our device, it was with the hope of having a working prototype by spring break in order to spend the time testing it on one of our dog volunteers, Lucy. In accordance with the timeline we created in our planning phase, we ordered our parts so that we could begin to implement and test the various components of our device. We also created a simple website to display the data we received from our device.

Upon recieving the parts, we realized that the NFC controller was much too large to fit on our device if we wanted the Pet Watch to be usable for different sizes of pets. At that moment, we decided to focus our efforts on the temperature

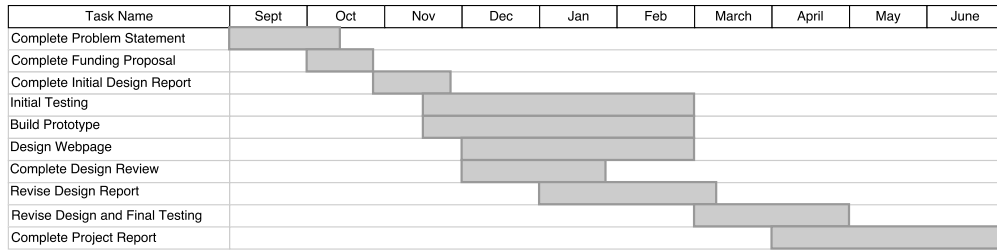


Figure 4: Development Timeline

sensor and accelerometer instead, leaving the problem with the NFC controller for future work if we had time. We also noticed that the Arduino UNO board was significantly heavier than we had anticipated, but decided to carry on with our initial plans for implementing our first prototype.

We began by working on our temperature sensor, wiring it to our Arduino UNO board and creating basic code to receive the temperature and humidity and store it in our database. We created a simple website so that we could display this information and ensure that our database was updating properly. Our initial results were very good with the temperature sensor, so we decided to move on to the accelerometer.

The accelerometer proved to be much more difficult to use than the temperature sensor had been. We initially connected it to our Arduino UNO, and encountered numerous issues transmitting our data back to our database in a usable format. Once we were finally able to receive data from the accelerometer, we had to convert our data to the number of steps taken by the pet, rather than simply their change in position. Our calculations on what constitutes a step for a pet at this point were rough estimates, and required future calibration and testing.

Once we could successfully display both temperature and accelerometer data on our simple website, we began to integrate our Huzzah wifi component to allow us to transmit data remotely. We immediately encountered issues communicating between the UNO and Huzzah, which required us to use a separate communication protocol to transmit the information from our sensors via wifi. At this point we rethought our design, and consulted Professor Dezfouli on the best method of transmitting this data. He advised us to eliminate the UNO board all together, and simply transmit our accelerometer and temperature data directly to the Huzzah.

We reconnected our temperature sensor and accelerometer to transmit through the Huzzah instead of the Arduino UNO and began testing once again. We encountered issues with the unavailability of the appropriate pins on the Huzzah to connect our accelerometer, but resolved these issues quickly to ensure that we could begin testing over spring break on our first prototype. Before we were able to do any testing, we connected the 150 mAh lithium ion polymer battery to power our device. This allowed us our device to be unplugged from our laptop so that the pet could wear it and transmit data.

## 2.3 Prototype 2

We began our second prototype with minimalism in mind, taking into consideration the results of our initial testing. Our first prototype worked, but was too heavy and bulky for a variety of pet sizes. We needed to redesign several components to cut down on weight and bulk. In addition, we needed to create a protective case for our Pet Watch, and implement a more user-friendly version of our website.

The first change we made on our new prototype was to move the temperature sensor the bottom side of our board. We made this change for two reasons: to reduce the bulk on the top side of the board, and to ensure that our temperature sensor could get more accurate readings by being closer to the body of the pet. This quickly improved our accuracy by reading the pet's body temperature relative to the exterior temperature, instead of simply returning the exterior temperature.

Table 2: Final Technologies Used

Collar	Website
Feather Huzzah Board	HTML
LIS3DH Accelerometer	CSS
DHT22 Temp Sensor	JavaScript
Arduino IDE	PHP
3D Printer	MySQL

We also needed to reduce the size of our protoboard to make sure that our device was less bulky. We cut the board to the smallest size we could while still being able to mount all of the necessary components. This decreased our weight and bulk, and gave us the opportunity to design a smaller case for the Pet Watch than we previously had in mind.

Earlier in the design process we had made a conceptual model for the Pet Watch case that would hold all of the components and protect them from wear and tear. In our new prototype, we made a small opening on the bottom side of the case to allow our temperature sensor to protrude and be flush with the body of the pet. In addition, we reduced the overall size of the case significantly, and after several iterations of 3-D printing, produced a much smaller and lighter case for the Pet Watch.

As we began integration testing on our new prototype, we also began to increase the capabilities of the website. We allowed for multiple users to login and enter information about multiple pets they have. We also improved the appearance of the user's home page, where information about the pet and their daily activities are displayed. In accordance with our requirements, we created a separate guest login where pet information could be viewed, but not changed. As we continued testing, we improved the layout of our website as well as the accuracy of our step counting algorithm, resulting in our final working prototype.

### 3 Testing

Our final objective was to perform testing on our Pet Watch to ensure it worked accurately. Once we finished implementation of each part of our design, we tested the component on its own to ensure it was working properly. This phase included manually collecting data with each sensor, then displaying the data on our simple website to check the accuracy of our readings. Our prototype 1 readings were not very accurate, leading us to change the position of the temperature sensor on the board for prototype 2, and alter our step counting algorithm for our final product.

After we completed integration of our components, we began integration testing. For our first prototype, we were still having difficulties with the wifi component, but were able to do limited testing on our mid sized dog volunteer, Lucy. She wore the collar for a period of time, and we were able to evaluate the accuracy of the temperature and accelerometer readings we received from the Pet Watch. Once we completed our second prototype, we performed more thorough testing of our design on our larger dog volunteer, Ellie. She wore the collar for an extended period of time and traveled out of the wifi range. This allowed us to test whether we were storing data on the device properly when it could not constantly send data to the database. We found that our prototype 2 effectively stored the data for later transmission, and recorded the steps taken by our dog volunteer over the period of time she wore the Pet Watch with reasonable accuracy.

In addition to the quantitative results generated by our testing, we also found that our pets seemed to enjoy wearing the collar, and that it was not a source of discomfort to them. The collar also held up extremely well when the pets wore it while playing and rolling in the grass.

Once we were able to reliably record and display data from the Pet Watch, we performed testing on our website. This consisted of allowing owners to use our website, and provide us feedback on the available features. We also tested our website with the W3C online validation services to ensure our code conformed to standards, and used WebAIM's accessibility evaluation tool to ensure that our website was accessible to as many users as possible.

Overall, the testing we conducted helped us to improve our design for both users and pets. We improved the usability of our website, the comfort of the collar, and the accuracy of our sensor measurements, resulting in an more useful and consistent final design.

## 4 Collar Design

We decided a component-based architecture made the most sense for the collar, where each sensor is implemented as a separate component. Arduino allowed us to easily build our device component by component. We created a separate script to handle each component and a script to integrate all of them. Arduino’s IDE allowed us to easily load these scripts and the relevant external libraries from a computer to a board. Each of these scripts is described in the sections below. In addition to creating the code for the collar, we also had to wire the hardware components together and create a case for them. **Figure 5** shows how the components are wired together.

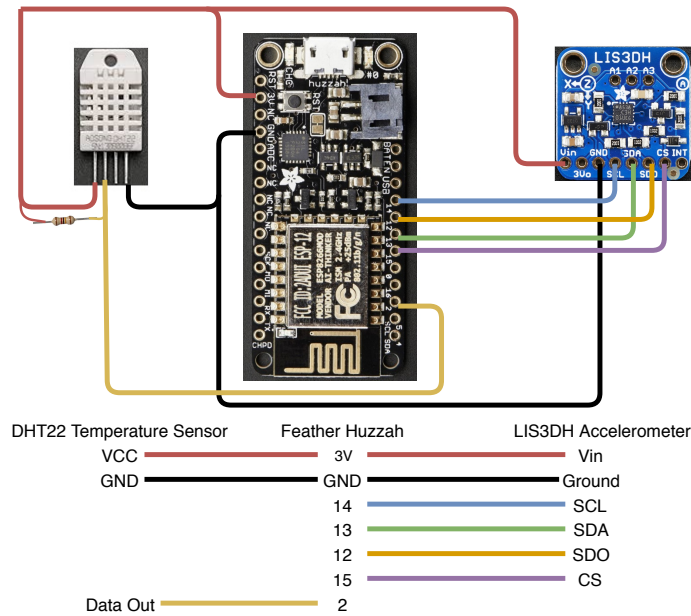


Figure 5: collar wiring

### 4.1 Temperature Sensor

The temperature sensor was the easiest component to implement in the entire project. We used the DHT22 temperature and humidity sensor[9]. It had just three wires: power, ground, and data. We hooked up these wires to the controller as described in **Figure 5**. We used a 10KOhm resistor as a pullup between the data and power lines. In addition, Adafruit’s library for the sensor contained functions that made it simple to collect and properly translate the data from the sensor.

The code for the temperature sensor can be found in Appendix A.1.3. It is composed of two data structures and four functions: `setupTemp`, `measureTemp`, `calculateTemps`, and `printTemps`. One data structure contains a current state and the other is a storage container. `setupTemp` runs at the start of the collar and sets up the variables for this component. The `measureTemp` function runs every minute and reads and stores the temperature data to the temporary data structure. Every five minutes, it also calls the `calculateTemps` function. This function averages the data from the past five minutes and stores it in the storage structure. `printTemps` is a helper function that prints the contents of the storage structure.

## 4.2 Accelerometer

The accelerometer was difficult to implement. We decided to use the LIS3DH triple-axis accelerometer from Adafruit[10]. This device contains eleven pin outs, but we only used six of them. Two were for power and ground. The other four operated the SPI protocol[8, p. 109-110]. The SPI is a communication protocol that connects a master device to one or more slave devices. We only had one slave device, the accelerometer. In our final design, we used the Feather Huzzah board as the master device. This protocol uses one wire to create a clock to synchronize the two components. Another wire is used to select which slave device to communicate to. Data is sent on the last two wires called MOSI and MISO. MOSI (Master Out Slave In) sends data from the master device to the slave. MISO (Master In Slave Out) sends data from the slave device to the master.

The greatest difficulty of this project was converting raw acceleration data to an accurate step count. We finally decided to model our algorithm off of basic mechanical pedometers. However, every approach faced the same large obstacles. The device secures on to the collar. However, collars frequently rotate on pets necks. Especially on walks when a leash is attached to the collar. This makes it difficult to correlate the raw acceleration data with the direction its happening in. Additionally, pets usually have 4 legs. This makes their movement mechanics drastically different than humans. Pets also have different leg lengths that affect the force exerted by each step.

The accelerometer script counts the number of steps the pet takes in five minute intervals. It contains similar components to the temperature sensor and can similarly be found in Appendix A.1. It has 2 data structures: a current state and a storage container. The current state contains the number of steps in the current interval and information on the current state of the accelerometer. The storage data structure stores the number of steps taken in each five minute time frame. The script also contains 3 functions: `setupAccel`, `measureAccel`, and `totalAccel`. `setupAccel` setups up communication with the accelerometer and sets variables to their default values. `measureAccel` checks whether a step was taken. It does this by reading from the accelerometer. It then converts the raw acceleration data into a G-force. The system determines steps by looking for spikes in acceleration above a certain G-force threshold. If the current reading is above this threshold, the function sets the current state to taking a step. Once the current state is taking a step, the function looks for a drop in acceleration below the threshold. This indicates the end of a step. The system then counts a step and resets the state. Every five minutes this function calls `totalAccel`. `totalAccel` simply stores the data on the past five minutes to the storage container.

## 4.3 Wifi Controller

The wifi implementation changed drastically over the course of the project. We eventually settled on using Adafruit's Feather Huzzah board[11]. This board comes with an ESP8266 wifi chip built in. We also used this board as the collar controller.

The Wifi controller script can be found in Appendix A.1.2. The script is composed of 2 main functions (`setupWifi` and `sendInfo`) and 5 helper functions: `initialData`, `accelData`, `tempData`, `recievePacket`, and `sendPacket`. The `initialData` function packages all the variables that are sent in every message. This includes global timing variable and logic variables to indicate if the package contains acceleration or temperature data. `acceldata` and `tempData` package all the data that has been stored in our sensor storage structures. `sendPacket` takes in a string of data as input and sends that data to our server. `recievePacket` simply recieves a packet and prints it to the serial port. We decided to create all of these helper functions to organize the packaging of the data and to make it easier to restructure sensor data. When we did modify the datastructure of a sensor, we only had to modify the corresponding helper function to extract the data properly again. `setupWifi` runs when the controller is turned on or reset. It connects to the designated wifi, then the server, and finally sends an initial message. The `sendInfo` function checks and reconnects with the wifi if needed. It does the same thing with the server connection. It then sends then sends all of the available data to the server.

## 4.4 Integration

We integrated all of our components using a controller script. This script can be found in Appendix A.1.1. This script contains several timing variables and the classic arduino functions: setup and loop. Setup runs once every time the board starts up. It simply calls each components setup function. Loop is constantly running. The loop function keeps track of the time. It runs the temperature sensor every minute, the accelerometer every 50 milliseconds, and sends the data every 5 minutes.

# 5 Server Design

## 5.1 Collar Server

The Collar Server was easy to initially implement, but hard to perfect. The scripts for this server are in Appendix A.2. We decided to run it on the schools linux webservers and use the same MySQL database we had used for previous classes. We created two helper scripts that we found useful when we created previous MySQL databases: the database configuration and log script. The database configuration script contains several variables that make logging in to the database easier. This script is included in all of our files that access our database. The Log script contains a function that logs events to a file and a function to redirect urls easier. This file is included in almost all our files to log any special events and make debugging the whole server easier.

The Connection script processes all of the messages sent by the collars to the server. It was easy to create the script and collect the messages from the collar. The difficulty came in extracting the data from the message and processing it into the database in the proper formats. This was not a static problem. We had to modify the processing logic and variables as they changed on the collar's web agent and the database. Even after we had solidified the collar and the database architecture, we still had to fix several errors from edge cases. The logic of the connection script is fairly simple at the high level. It first connects to the database, then it checks if this is the first packet number sent by the collar. The first packet number indicates a collar reset and additional code runs to set up the time offset between the collar and the database. The next block of code gets the time offset for the collar. The script processes the accel data then the temperature data if there is any to process. These are done very similarly. The script first checks if there is data to process. It then extracts the data from the message sent by the collar into php arrays. Each data point in the arrays is then inserted into the database. The time the data was collected is also calculated and then stored in the database.

## 5.2 Front End Server

In order to produce a useful product, we required a front end consisting of a website to display the pet data to the owner. The code for this portion of our implementation is included in Appendix A.3 and A.4. We ended up using HTML and CSS to develop and style our webpage, as well as Bootstrap [12] for easier styling. We also used jQuery [13] for dynamic features of our site, and PHP to communicate with our database.

To implement this portion of our design, we created a login on our website that initially requests the user's first and last name, their email address, and a password. Upon registering, this data is stored in our database, and the new user can login. Once logged in, they can then select "Add Pet", where they will be prompted to add some basic information about their pet, along with the serial number of their Pet Watch. This information is stored seperately in our database, so that multiple pets can be associated to one user.

Once the user has their account setup with their new Pet Watch registered, we begin storing the pets data collected from the collar in the database associated with their specific device. We perform some processing on the data, then display it on the user's home screen. The necessary processing includes converting the accelerometer data, which is recorded as change in position, to steps taken by the pet. This code is modified based on the size of the pet, since larger



pets have a different algorithm to register a step than a smaller pet. Once the data is in a usable format, the owner can view it on the website. While the site itself is fairly simple, having a successfully implemented front end made our design practical, since we were able to not only read a pets data, but display it for use.

## 6 Societal Issues

### 6.1 Ethical

The goal of this project is to increase the communication between a pet and it's owners. This is an ethical issue because an increasing number of people are adopting dogs and other pets. When they adopt a pet, they now have an ethical imperative to take care of that pet to the best of their abilities. This project is designed to increase the communication so the owner can be more informed about their pet and can make better decisions. personal story about how hard it is to manage a dog with a large family

Like many systems that run on the internet today, our system has the capability to collect information on the users of our system and their pets. We have a responsibility as engineers to protect our users data, collect only what data we need, and provide transparency regarding how their data will be used.

### 6.2 Social

The Pet Watch will help the pet owning community to provide better care for their animals. Since owners are able to actively track the health of their pet, they can ensure that their pets are receiving all of the care they need, and adjust their routine if necessary. This feature of the Pet Watch has the potential to reduce unintentional animal neglect, by giving owners the ability to monitor their pets health. The data collected by the Pet Watch can also be shared with veterinarians, improving the vets ability to detect changes in the health of the pet. The Pet Watch will not help pets whose owners choose to abuse them or neglect them, but owners who forget to regularly exercise their pet, for instance, can learn to provide a healthier lifestyle for their pet. We do not anticipate any negative social impacts of the Pet Watch, and hope that it will bring positive change to the pet owning community.

### 6.3 Political

We do not believe that the Pet Watch will have any political impact, since it is designed as a means for owners to track the health of their pets. We do anticipate public officials having any concerns over our design, outside their own personal interest in tracking their pets health.

### 6.4 Economics

As we were doing initial research for this project, we realized that several similar products were already on the market, but they cost a lot. This shows the market viability for our project, but also a need to stand out. During our project, we made it our goal to make our product more affordable than the current solutions in order to target a more cost conscience market demographic. The materials for the second prototype cost \$43.80. The breakdown of those costs can be seen in table **Table 3** as well as a projected cost of scaling up to just 100 devices. These low materials cost means we can beat our competitors prices. The fitbark 2 is strong low cost competitor in the market at \$69.95[3]. We can sell our device for \$60 and still have a over 40% profit margin.

Table 3: Material costs for petwatch

Materials	Cost of prototype 2 (\$)	Cost per device to scale to 100 (\$)
Feather Huzzah Board[11]	16.95	13.56
LIS3DH Accelerometer[10]	4.95	3.96
DHT22 Temp Sensor[9]	9.95	7.96
150mAh LiPoly battery[14]	5.95	5.36
protoboard[15]	4.50	3.60
3D Printed case	1.00	1.00
wires	0.50	0.50
Total	43.80	35.94

## 6.5 Health and Safety

The Pet Watch does not present any health or safety concerns to the owner or guest monitoring their pet, but we have not yet gathered enough data to assess health and safety risks to the pet. We do not believe that our device can harm a pet as it is designed, but if the case broke, the sharp components inside could potentially cut the pet. We also do not know the potential health risks to the pet of having an electronic device on their body at all times. We know that electronic devices can be harmful to humans, but not enough research has been conducted on the potential harm to pets. The Pet Watch needs to be a very safe device, since the whole purpose of it is to increase the health of pets. Therefore, we believe that we would need to do more testing of the Pet Watch before being comfortable allowing pets to wear it full time.

## 6.6 Manufacturability

We believe that our prototype can be built, but there are many improvements that can be made in the design. Going into this project, We wanted to make sure we created a design that is small enough to be worn by any sized dog or other pets, but also affordable. The materials for our final prototype cost \$45, but this cost would go down if we scaled to manufacturing. Initially our device cost over \$100 and was much bulkier. We found an easier way to build this product without the most expensive component: the arduino board. Cutting this component reduced our device size and weight as well. We additionally cut the nfc board from our design to reduce the device size and to limit the scope to only core features.

We layed out our hardware on breadboards in many different configurations and eventually decided upon a layout that minimized empty space. We 3-D printed multiple cases for our product, but We ran into a time crunch while developing the 3-D case. The design of the case could be refined more. We did not have experience with the necessary computer aided design programs coming into the project, so we had to learn as we went. This lack of knowledge led to us creating a simple design for the case. Additionally our design was impaired by the accuracy of our 3-D printers available. If we were to manufacture our product, we would scale our process to use industrial grade printers, which would allow us to be more precise and thus create a smaller and more refined design.

## 6.7 Sustainability

The Pet Watch is very sustainable in the sense that it can be easily repaired if it fails, and parts can be replaced if necessary. We hope to make the case waterproof as well, which would reduce the risk of serious failure of the device. We can also update the software to fit new needs, which we believe will arise as the demand for a product to track the health of pets continues to increase. These features allow the Pet Watch to last for a long time, and we believe the need for such a device will continue to grow. Unfortunately, the components required to build the Pet Watch are not manufactured locally, nor are they easily recycled. These concerns have a negative impact on the sustainability of the product, as the Pet Watch does not effectively use the required resources throughout its lifecycle.

## **6.8 Environmental Impact**

The environmental issues of the Pet Watch primarily stem from its use of electronic components that are not easily recycled, nor manufactured locally. This increases the production of e-waste, which leads to severe environmental and health concerns for those working in the informal e-waste recycling sector. Additionally, the waste produced in transporting the components necessary to build the Pet Watch negatively impacts the environment. While the Pet Watch does have a significant negative environmental impact, it uses very little power, and the case is made from a biodegradable plastic that can be composted at an industrial facility within 1-3 months.

## **6.9 Usability**

We tried to make our product as simple and user friendly as possible. For our website, we followed a fairly typical approach to the login and site navigation, allowing users of all abilities to easily access their account and view the desired information. Our home page for the owner clearly displays the pet's daily data, with an easy to access tab at the top for changing information about the pet or the user account. They can also easily access the pet's history with the "Pet History" tab, allowing simple access to all of the pet's information.

Similarly, with the device itself, we tried to make the design as compact and intuitive as possible. The Pet Watch can simply be looped through the pet collar, which is then attached to the pet as usual. We also took into account the user-friendliness for the pet, considering that they will be the one wearing the device. We made sure that our design was lightweight and comfortable, ensuring that the pet would not be bothered by it. This increases usability for owners as well, since they would not want to use a device that makes their pet uncomfortable.

## **6.10 Lifelong Learning**

This project helped show us the path to becoming a lifelong learner. We actually considered learning when selecting our technologies. We chose arduino because of the community around it. This community has created great documentation and support forms for it. We knew these documents would make learning anything we didn't know about arduino easier than alternative technologies. We were already familiar with arduino technology, but we expanded our knowledge because we needed to use different hardware and perform different tasks. Just because we know a technology, doesn't mean we can't learn more about it.

## **6.11 Compassion**

One of our primary goals in designing the Pet Watch was to relieve the suffering of pets, as well as provide peace of mind to pet owners. Even the most loving pet owners may not know that their pet needs more exercise than it is currently receiving, or may not realize that the temperature of their home is not suitable for their four legged companions. The Pet Watch is designed to help owners make sure that their pet is healthy and comfortable, by providing quantitative data directly on the owners phone or laptop, wherever they are. This ensures that owners dont have to worry about the comfort of their pet while they are away at work, because they can just monitor the pet from their device. If they see that their pet needs more exercise or a different temperature, they can address these issues and ensure that their pet is not suffering unnecessarily.

## **7 Conclusion**

Over the past school year, we were able to successfully implement our Pet Watch design. While there were certain features that we had to modify or leave out entirely, we are very pleased with the final product, and proud of the work

we did.

There were a lot of lessons we learned in fully designing and implementing a product, first of all, that things will always take longer than we expect. While implementing certain aspects of our design was relatively simple, other parts that seemed simple ended up being extremely arduous. Whether it was struggling to find a simple bug in the code, or redesigning an entire component due to compatibility issues, we encountered slow downs on almost every aspect of our design. This brings me to a second significant lesson we learned: spending more time in the planning phase leads to less time in implementation. If we had performed more thorough research on each of our components, we could have avoided some of the integration issues we encountered. While it may seem faster to begin on implementation right away, it certainly pays off to have thorough knowledge on all of your components, and a concrete plan on how to design the product.

While we are pleased with the work we did, if we were to do this project over, we would have started planning and research earlier in the school year, allowing more time to develop our design before implementing it. In addition, we would have allocated more time for implementation of each component, allowing us to have a more robust design and fewer bugs in integration.

Designing the Pet Watch allowed us to bring together our experience from various classes at Santa Clara into a working product. While we were not able to accomplish everything we had hoped, we learned so much about product development, as well as many new technical skills. Overall we are very proud of how our Pet Watch turned out, and hope to continue improving our design and adding the features we had been forced to eliminate.

## References

- [1] Craig Smith. 37 amazing fitbit statistics, 2017. URL [expandedramblings.com/index.php/fitbit-statistics/](http://expandedramblings.com/index.php/fitbit-statistics/).
- [2] Alden Gonzalez. Counting the steps: How rams use player tracking to optimize availability, 2017. URL [www.espn.com/blog/los-angeles-rams/post/\\_/id/35171/counting-the-steps-how-rams-use-player-tracking-to-optimize-availability](http://www.espn.com/blog/los-angeles-rams/post/_/id/35171/counting-the-steps-how-rams-use-player-tracking-to-optimize-availability).
- [3] Home, 2017. URL [www.fitbark.com](http://www.fitbark.com).
- [4] Frank Tsui, Orlando Karam, and Barbara Bernal. *Essentials of Software Engineering*. Jones and Bartlett Learning, 2018. ISBN 978-1284106008.
- [5] Mike O’Kane. *A web-Based Introduction to Programming*. Carolina Academic Press, Durham, North Carolina, 2011. ISBN 978-1594608445.
- [6] 2018. URL <http://wiki.helpme.engr.scu.edu/index.php/Webpage>.
- [7] What is arduino, 2018. URL <https://www.arduino.cc/en/Guide/Introduction>.
- [8] J. Edward Carryer, Matthew Ohline, and Thomas Kenny. *Introduction to Mechatronic Design*. Pearson, 2010. ISBN 978-0131433564.
- [9] Dht22 temperature-humidity sensor. URL <https://www.adafruit.com/product/385>.
- [10] Lis3dh triple-axis accelerometer. URL <https://www.adafruit.com/product/2809>.
- [11] Adafruit feather huzzah with esp8266. URL <https://www.adafruit.com/product/2821>.
- [12] Mark Otto. About bootstrap, 2011. URL <https://getbootstrap.com/docs/4.1/about/overview/>.
- [13] jquery api. URL <https://api.jquery.com>.
- [14] Lithium ion polymer battery - 3.7v 150mah. URL <https://www.adafruit.com/product/1317>.
- [15] Adafruit perma-proto half-sized breadboard pcb - single. URL <https://www.adafruit.com/product/1609>.

## A Source Code

### A.1 Collar code

#### A.1.1 Collar Controller

```
//collar controller code. Contains the arduino setup and loop functions.
//Calls all the other collar components.

#include <Wire.h>

//communication protocol for accelerometer
#include <SPI.h>

//libraries for the temperature sensor
#include <Adafruit_LIS3DH.h>
#include "DHT.h"

//general library for all adafruit sensors
#include <Adafruit_Sensor.h>

//timing variables. keeps track of the current time and time since last used different components
unsigned long Time;
unsigned long prevTemp;
unsigned long prevAccel;
unsigned long prevTrans;

//controls the amount of data we can store. each datapoint is 5 minutes
//right now it is configured to hold 4 hours of data
const int storageLimit=12*4;

void setup(void) {

    Time=millis();
    Serial.begin(9600);

    //component setup functions
    setupAccel();
    setupTemp();
    setupWifi();
}

void loop() {

    Time=millis();
```

```

//run temperature check every minute
if(Time-prevTemp>1000*60UL)
    measureTemp();

//run accel check every 50 milliseconds
if(Time-prevAccel>50UL)
    measureAccel();

//send data every 5 minutes
if(Time-prevTrans>1000*60*5UL)
    sendInfo();

}

```

### A.1.2 Wifi controller

```

//library for wifi chip
#include <ESP8266WiFi.h>

//fill in with proper wifi network and password
const char* ssid = "network_name";
const char* password = "network_password";

const char* host = "students.engr.scu.edu";
String url="/~janders2/initialConn.php";
WiFiClient client;
const int httpPort=80;

void setupWifi(){
    prevTrans=Time;

    //connect to wifi
    WiFi.begin(ssid,password);
    Serial.println("Connecting to WiFi");
    while(WiFi.status() != WL_CONNECTED){
        delay(500);
        Serial.print(".");
    }
    Serial.println("");
    Serial.println("WiFi connected");
    Serial.println("IP address: ");
    Serial.println(WiFi.localIP());
    Serial.print("Netmask: ");
    Serial.println(WiFi.subnetMask());
    Serial.print("Gateway: ");
    Serial.println(WiFi.gatewayIP());

    //connect and send initial packet to server
    if(!client.connect(host, httpPort)){

```

```

    Serial.println("connection failed");
    return;
}

String data=initialData(0,0);

sendPacket(data);

delay(500);

recievePacket();

Serial.println();
}

void sendInfo(){
    prevTrans=Time;

    //checks the wifi and server connection and reconnects with them if they get disconnected
    if(WiFi.status() !=WL_CONNECTED){
        Serial.println("wifi disconnected");
        while(WiFi.status() !=WL_CONNECTED){
            delay(500);
            Serial.print(".");
        }
        Serial.println("wifi reconnected");
    }
    if(!client.connected()){
        Serial.println("client disconnected");
        while(!client.connect(host, httpPort)){
            Serial.println("connection failed");
            return;
        }
        {
            Serial.println("client reconnected");
        }
    }
}

//calls functions to package all of our data and then sends it
String data=initialData(1,1)+accelData()+tempData();
sendPacket(data);
delay(500);
recievePacket();
}

/*
 * returns a string with the initial data set for our data packet
 * the two variables set indicate if we are sending our data from
 * the sensors or not (1 for yes, 0 for no)
 */
String initialData(int accel,int temp){
    String data=String("prevTrans=")+ prevTrans;
}

```



```

    if(accel){
        data+ "&accelData=1";
    }
    else{
        data+ "&accelData=0";
    }
    if(temp){
        data+ "&tempData=1";
    }
    else{
        data+ "&tempData=0";
    }
    return data;
}

//packages the accel data
String accelData(){
    //Serial.println("packaging accel data");

    String data=String("&accelIndex=")+accelIndex;
    for(int i=0; i < accelIndex;i++){
        data+=String("&accel["+i+"]="+steps[i]+&time["+i+"]="+accelTime[i];
    }
    accelIndex=0;

    return data;

}

//packages the temp data
String tempData(){
    //Serial.println("packaging temp data");

    String data= String("&tempIndex=")+storedIndex;
    for(int i=0; i < storedIndex;i++){
        data+=String("&temp["+i+"]="+avgTemp[i]+&hum["+i+"]="+avgHum[i]
        +"&HI["+i+"]="+avgHI[i]+&ttime["+i+"]="+timeStamp[i];
    }
    storedIndex=0;

    return data;

}

//recieves packet and prints it to the serial port
void recievePacket(){
    while(client.available()){
        String line=client.readStringUntil('\r');
        //Serial.print(line);
    }
    Serial.println("done reading");
}

```

```

//packages the data and sends it out in a post message
void sendPacket(String data){
  String message=String("POST ") + url + " HTTP/1.1\r\n" + "HOST: " + host + "\r\n"
  +"Accept: *" + "/"+"*\r\n"+ "Content-Length: "+data.length()+"\r\n"
  + "Content-Type: application/x-www-form-urlencoded\r\n"+ "\r\n" + data+ "\r\n\r\n";
  client.print(message);
  Serial.println(message);
}

```

### A.1.3 Temperature Sensor

```

//define temperature pin and sensor type
#define DHTPIN 2 //green wire use pin 0 for prototype 1 and pin 2 for prototype 2
#define DHTTYPE DHT22
DHT dht(DHTPIN, DHTTYPE);

//stores an average temperature of the past five minutes. can hold up to the storage limit

float avgTemp [storageLimit];
float avgHum [storageLimit];
float avgHI [storageLimit];
unsigned long timeStamp [storageLimit];
int storedIndex;

//stores the temporary data to calculate the averages
float Temps[5];
float Hums[5];
float HIs[5];
int tempIndex;

//tester function to print all the data stored in the temperature datastructure
void printTemps(){
  for(int i =0;i<storageLimit;i++){
    Serial.println(avgTemp[i]);
    Serial.println(avgHum[i]);
    Serial.println(avgHI[i]);
    Serial.println(timeStamp[i]);
    Serial.println();
  }
}

//sets up temperature controller
void setupTemp(){
  dht.begin();
  tempIndex=0;
  storedIndex=0;
  prevTemp=0;
}

```

```

//calculates the average temperature and puts data in datastructure
void calculateTemps(){
    if(storedIndex==storageLimit)
        return;
    float temp=0;
    float hum=0;
    float hi=0;

    //averages the temperatures
    for (int i = 0; i<5;i++){
        temp+=Temps[i];
        hum+=Hums[i];
        hi+=HIs[i];
    }
    temp/=5;
    hum/=5;
    hi/=5;

//stores the data
    avgTemp[storedIndex] = temp;
    avgHum[storedIndex] = hum;
    avgHI[storedIndex] = hi;
    timeStamp[storedIndex] = Time;

    storedIndex++;
}

//measures the temperature every minute
void measureTemp(){
    prevTemp=Time;

    //gets the hum and temp reading from sensor
    float hum = dht.readHumidity();
    float temp = dht.readTemperature(true);
    if(isnan(hum)||isnan(temp)){
        //Serial.println("Falied to read temp sensor");
        return;
    }
    else{
        //calculates the heat index and stores the data to temporary datastructure
        float hi = dht.computeHeatIndex(temp,hum);
        Temps[tempIndex]=temp;
        Hums[tempIndex]=hum;
        HIs[tempIndex]=hi;
    }
    //calculates the temp if we have 5 minutes worth of data
    if(tempIndex==4){
        calculateTemps();
    }

    tempIndex = (tempIndex+1)%5;
}

```

#### A.1.4 Accelerometer

```
// defining accelerometer pins. Important that these pins are hooked up
// using the designated pins SPI communication
#define LIS3DH_CLK 14 //blue SCL
#define LIS3DH_MISO 12 //yellow SDO
#define LIS3DH_MOSI 13 //green SDA
#define LIS3DH_CS 15 //blue CS

// global accelerometer variables
Adafruit_LIS3DH lis = Adafruit_LIS3DH(LIS3DH_CS, LIS3DH_MOSI, LIS3DH_MISO,LIS3DH_CLK);

//stores the total movement of 5 minute intervals up to the storage limit
int steps[storageLimit];
unsigned long accelTime[storageLimit];
int accelIndex;

float movement;
int stepCount;
bool isStepping;
unsigned long prevAccelStore;

//g force threshold to take a step
const float gthreshold=1.45;

//sets up accelerometer and sets accelerometer variables
void setupAccel(void){

    Wire.begin(14,13);
    if(! lis.begin(0x18)) { // change this to 0x19 for alternative i2c address
        Serial.println("Couldnt start accelerometer");
    }
    Serial.println("accelerometer found!");

//sets the range of sensitivity for the accelerometer measured in G forces
lis.setRange(LIS3DH_RANGE_4_G); // 2, 4, 8 or 16 G!

    movement=0;
    prevAccel=0;
    accelIndex=0;
    stepCount=0;
    isStepping=false;
    prevAccelStore=0;
}

//stores the number of steps of the past 5 minutes
void totalAccel(){
if(accelIndex==storageLimit)
    return;
    steps[accelIndex]=stepCount;
```

```

    accelTime[accelIndex]=Time;
    accelIndex++;
    stepCount=0;
    prevAccelStore=Time;
}

//checks whether a step was taken
void measureAccel(){
    prevAccel=Time;
//reads data from accelerometer
    lis.read();
    sensors_event_t event;
    lis.getEvent(&event);

    //converts accelerometer variables into a measure of G force
    movement= (event.acceleration.x*event.acceleration.x+event.acceleration.y
    *event.acceleration.y+event.acceleration.z+event.acceleration.z)/(9.8*9.8);

//determines if the pet is taking a step. based on the current and previous g forces
    if(isStepping){
        if(movement<gthreshold){
            isStepping=false;
            stepCount++;
            Serial.println("is this a step");
        }
    }
    else{
        if(movement>gthreshold){
            isStepping=true;
        }
    }

    if(Time-prevAccelStore>1000*60*5UL)
        totalAccel();
}

```

## A.2 Collar webserver code

### A.2.1 Connection

```

<?php
//files with helper variables and functions
    require 'db_config.php';
    require 'log.php';

    //connects to database
    $conn = new mysqli($db_host, $db_user, $db_pass, $db_name);
    if(!$conn){
        echo("connection to server failed");
        die('Could not connect: ' . mysql_error());
    }

```

```

}
    //if connection is good, execute rest of script
else{
$time=$_POST['prevTrans'];
var_dump($_POST);
//echo "<p> Time: " . $time . "</p>";
//file_put_contents('display.html',$Write);

//sets up database and timing if this is the first packet between the collar and website
if($_POST['PacketNumber']==0){
$stmt="DELETE FROM timing WHERE id=".$_POST['SN'].>";";
$result=$conn->query($stmt);
$sentTime=time()-($_POST['prevTrans']/1000);
$stmt="INSERT INTO timing Values($_POST['SN'].",".$sentTime.");";
$result=$conn->query($stmt);
if(!$result){
logToFile('query failed: '.$stmt);
}
}

//gets the time offset for this collar
$stmt="SELECT startTime FROM timing WHERE id=".$_POST['SN'].>";";
$result=$conn->query($stmt);
$row=$result->fetch_assoc();
$timeOffset=$row['startTime'];
echo $timeOffset;

//checks if we have accel data to process
if($_POST['accelData']==1){
    $index=$_POST['accelIndex'];
    if($index!=0){

        //extract data from post request
        $accel=$_POST['accel'];
        $timeArray=$_POST['atime'];

//processes accel data
        for($i = 0;$i<$index;$i++){
            $time=(int)(((int)$timeArray[$i])/1000+$timeOffset);
            $stmt="INSERT INTO accelData Values (".$time. ", " . $accel[$i].");";
            echo $stmt;
            $result=$conn->query($stmt);
            if(!$result){
logToFile('query failed: '.$stmt);
            }
        }
    }
}

//checks if we have temp data to process
if($_POST['tempData']==1){
    $index=$_POST['tempIndex'];

```

```

if($index!=0){

    //extract data from post request
    $temp=$_POST['temp'];
    $hum=$_POST['hum'];
    $HI=$_POST['HI'];
    $timeArray=$_POST['ttime'];

//processes temp data
    for($i = 0;$i<$index;$i++){
        $time=(int)(((int)$timeArray[$i])/1000+$timeOffset);
        $stmt="INSERT INTO tempData Values (".$time." , "
            . $temp[$i].", ".$hum[$i].", ".$HI[$i].");";
        echo $stmt;
        $result=$conn->query($stmt);
        if(!$result){
logToFile('query failed: '.$stmt);
        }
    }
}
}
}

?>

```

## A.2.2 log

```

<?php
//helper function to log events and errors
function logToFile($statement){
    $file = 'php-logs.txt';
    $current = file_get_contents($file);
    $current .= "$statement\n";
    file_put_contents($file, $current);
}

//helper function to redirect urls
function redirect($url){
    ob_start();
    header('Location: '.$url);
    ob_end_flush();
    die();
}

?>

```

## A.2.3 database configuration

```

<?php

```

```
//several variables to make logging in to database easier.
//to get the login info for your database go to
//http://wiki.helpme.engr.scu.edu/index.php/MySQL-5
$db_host = "dbserver.engr.scu.edu";
$db_user = "<username>";
$db_pass = "<accessCardID>";
$db_name = "sdb_<username>";

?>
```

#### A.2.4 database setup

```
<?php
//sets up the basic database for the website

require 'db_config.php';

$conn = new mysqli($db_host, $db_user, $db_pass, $db_name);
if(!$conn){
    die('Could not connect: '.mysql_error());
}
$userTable = "CREATE TABLE Users ( user_id INTEGER, first_name VARCHAR(255),
last_name VARCHAR(255), email VARCHAR(255), password VARCHAR(255) );";
$result = $conn ->query($userTable);

$testTable = "CREATE TABLE Ellie ( log_time INTEGER, accel INTEGER, temp INTEGER,
humid INTEGER, tag_id INTEGER, tag_time INTEGER);";
$result = $conn -> query($testTable);

$testTable = "CREATE TABLE Pets ( pet_id INTEGER, user_id INTEGER, name INTEGER,
weight INTEGER, height INTEGER, age INTEGER, type VARCHAR(255));";
$result = $conn -> query($testTable);

?>
```

### A.3 Frontend User Webpages

#### A.3.1 Home Page

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Welcome</title>
    <meta charset="utf-8"/>
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.min.css"
integrity="sha384-1q8mTJOASx8j1Au+a5WDVnPi2lkFfwEAa8hDDdjZlpLegxhjVME1fgjWPGmkzs7"
crossorigin="anonymous">
    <link href='https://fonts.googleapis.com/css?family=Roboto:300,400,700'
rel='stylesheet' type='text/css'>
</head>
```



```

<body>
<header class = "container">
<nav class="col-sm-4 col-sm-offset-8 text-right">
    <a class="btn btn-nav" href="index.html" role="button"><b>HOME</b></a>
<a class="btn btn-nav" href="about.html" role="button"><b>ABOUT</b></a>
    <a class="btn btn-nav" href="contact.html" role="button"><b>CONTACT</b></a>
</nav>

<div class = "row">
<h1 class = "col-sm-16">PetWatch</h1>
</div>
</header>

    <section class="jumbotron">
        <div class="container">
            <div class="row text-center">
                <h2>Welcome to Pet Watch!</h2>
                <a class="btn btn-primary btn-lg" href="ownerLogin.html" role="button"
                height = 50px width = 100px><b>Owner Login</b></a>
                <a class="btn btn-primary btn-lg" href="guestLogin.php"
                role="button"><b>Guest Login</b></a>
                <br><br>
<p>Don't have an account? Sign up now!</P>
<a class="btn btn-success" href="ownerRegister.html">Register</a>
<br>
</div>
        </div>
    </section>
<footer class = "container">
<div class = "row">
<p id = 'foot' class = 'col-md-4'>The content of these web pages is not generated by
and does not represent the views of Santa Clara University or any of its departments
or organizations.</p>
<ul class="col-sm-8">
    <li class="col-sm-1">
        <a href="https://twitter.com
        /SantaClaraUniv?ref_src=twsrc%5Egoogle%7Ctwcamp%5Eserp%7Ctwgr%5Eauthor">
            
        </a>
    </li>
    <li class="col-sm-1">
        <a href="https://www.facebook.com/SantaClaraUniversity/">
            
        </a>
    </li>
    <li class="col-sm-1">
        <a href="https://www.instagram.com/santaclarauniversity/?hl=en">
            
        </a>
    </li>
</ul>

```

```

        </ul>
</div>
</footer>

</body>
</html>

```

### A.3.2 owner login

```

<!DOCTYPE html>
<html lang = "en">
<head>
<title>Owner Login</title>
<meta charset = "utf-8"/>
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.min.css"
      integrity="sha384-1q8mTJOASx8j1Au+a5WDVnPi2lkFfwwEAa8hDDdjZlpLegxhjVME1fgjWPGmkzs7"
      crossorigin="anonymous">
  <link href='https://fonts.googleapis.com/css?family=Roboto:300,400,700'
        rel='stylesheet' type='text/css'>
</head>

<body>
<header class = "container">
<nav class="col-sm-4 col-sm-offset-8 text-right">
  <a class="btn btn-nav" href="index.html" role="button"><b>HOME</b></a>
<a class="btn btn-nav" href="about.html" role="button"><b>ABOUT</b></a>
  <a class="btn btn-nav" href="contact.html" role="button"><b>CONTACT</b></a>
</nav>

<div class = "row">
<h1 class = "col-sm-16">PetWatch</h1>
</div>
</header>

<section class = "jumbotron">
<div class = "container text-center">
<h2>Owner Login</h2>
<div class = "row text-center tranbox">
<form name="ownerLogin" method="post" action="ownerLogin.php">
<div class="row">
<p class="col-sm-4 col-md-offset-2">
<label="email"><b>Email</b></label><br>
<input name="email" id="email" type="text" placeholder="Enter Email" required>
</p>
<p class="col-sm-4">
<label="password"><b>Password</b></label><br>
<input name="password" id="password" type="password" placeholder="Enter Password" required>
</p>
</div>
<button name="Submit" id="submit" class="btn btn-success" type="submit">Log In</button>
<br>
</form>

```

```

<br><br>
<p>Don't have an account? Sign up now!</P>
<a class="btn btn-success" href="ownerRegister.html">Register</a>
<br>
</div>
</div>
</section>

</body>
</html>

```

### A.3.3 owner register

```

<!DOCTYPE html>
<html lang = "en">
<head>
<title>Owner Registration</title>
<meta charset="utf-8"/>
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.min.css"
    integrity="sha384-1q8mTJOASx8j1Au+a5WDVnPi2lkFfwwEAa8hDDdjZlpLegxhjVME1fgjWPGmkzs7"
    crossorigin="anonymous">
    <link href='https://fonts.googleapis.com/css?family=Roboto:300,400,700'
        rel='stylesheet' type='text/css'>
</head>

<body>
<header class = "container">
<nav class="col-sm-4 col-sm-offset-8 text-right">
    <a class="btn btn-nav" href="index.html" role="button"><b>HOME</b></a>
<a class="btn btn-nav" href="about.html" role="button"><b>ABOUT</b></a>
    <a class="btn btn-nav" href="contact.html" role="button"><b>CONTACT</b></a>
</nav>

<div class = "row">
<h1 class = "col-sm-16">PetWatch</h1>
</div>
</header>

    <section class = "jumbotron">
        <div class = "container">
<div class = "row text-center">
<h2>Owner Registration</h2>
<form class = "col-md-6 col-md-offset-3 form-horizontal tranbox" name = "ownerRegister"
    method="post" action = "ownerRegister.php">
<p>Register for your account! </p>
<label class = "control-label" for = "first_name"><b>First Name</b></label>
<input name="first_name" id = "first_name" type= "text" placeholder = "First Name" required>
<br/>
<label class = "control-label" for = "last_name"><b>Last Name</b></label>
<input name="last_name" id = "last_name" type= "text" placeholder = "Last Name" required>
<br/>

```

```

<label class = "control-label" for = "email"><b>Email</b></label>
<input name="email" id = "email" type= "email" placeholder = "Email" required>
<br/>
<label class = "control-label for= "password""><b>Password</b></label>
<input name="password" id = "password" type="password" placeholder = "Enter Password" required>
<br/>
<button name="Submit" id = "submit" class = "btn btn-success" type="submit">Register</button>
<br/>
</form>
</div>
</div>
</section>
</body>
</html>
<style>
label{
width: 80px;
margin: 5px;
}
button{
margin: 30px;
}
</style>

```

#### A.3.4 pet register

```

<html lang = "en">
<head>
<title>Add Pet</title>
<meta charset = "utf-8"/>
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.min.css"
integrity="sha384-1q8mTJOASx8j1Au+a5WDVnPi2lkFfwwEAa8hDDdjZlpLegxhjVME1fgjWPGmkzs7"
crossorigin="anonymous">
<link href='https://fonts.googleapis.com/css?family=Roboto:300,400,700'
rel='stylesheet' type='text/css'>
</head>

<body>
<header class = "container">
<nav class="col-sm-4 col-sm-offset-8 text-right">
<a class="btn btn-nav" href="index.html" role="button"><b>HOME</b></a>
<a class="btn btn-nav" href="about.html" role="button"><b>ABOUT</b></a>
<a class="btn btn-nav" href="contact.html" role="button"><b>CONTACT</b></a>
</nav>

<div class = "row">
<h1 class = "col-sm-16">PetWatch</h1>
</div>
</header>
<div class = "petData">
<section class = "jumbotron">
<div class = "container text-center">

```

```

<h2>Add Pet</h3>
<form class = "form-horizontal col-md-6 col-md-offset-3 tranbox" name = "ownerRegister"
      method="post" action = "petRegister.php">
  <p>Enter some information about your pet!</p>
  <label class = "control-label" for "name"><b>Pet's Name</b></label>
  <input align = "right" name="name" id = "name" type= "text" placeholder = "Pet Name" required>
  <br/>
  <label class = "control-label" for "serial"><b>Collar Serial Number</b></label>
  <input name = "serial" id = "serial" type = "text" placeholder = "Serial Number" required>
  <br/>
  <label class = "control-label" for "weight"><b>Weight</b></label>
  <input name="weight" id = "weight" type= "text" placeholder = "Weight" required>
  <br/>
  <label class = "control-label" for "height"><b>Height</b></label>
  <input name="height" id = "height" type= "text" placeholder = "Height" required>
  <br/>
  <label class = "control-label" for "age"><b>Age</b></label>
  <input name="age" id = "age" type= "age" placeholder = "age" required>
  <br/>
  <label class = "control-label" id = "odd" for = "sel"><b>Select the type of pet</b></label>
  <select id = "sel" class = "form-control" style="width:150px;"required>
  <option name = "type" id = "type">Dog</option>
  <option name = "type" id = "type">Cat</option>
    <option name = "type" id = "type">Other</option>
  </select>
  <br/>
  <button name="Submit" id = "submit" class = "btn btn-success" type="submit">Save</button>
  <br/>
</form>
</div>
</section>
</div>
</body>

<style>
select {
display: block;
margin: 0 auto;
margin-bottom: 1%;
}
label{
width: 140px;
margin: 5px;
margin-left: 0px;
}
button{
margin: 10px;
}
#odd{
width: 150px;
}
</style>
<!--.norm{
width: 10em;

```

```
margin-right: 3px;
text-align: right;
}-->
```

## A.4 Backend User Scripts

### A.4.1 owner login

```
<?php
session_start();

require 'db_config.php';
require 'log.php';

$conn = new mysqli($db_host, $db_user, $db_pass, $db_name);

if(!$conn){
die('Count not connect: ' .mysql_error());
}

//Check login
if(!empty($_POST['email']) &&!empty($_POST['password'])){
$email = $_POST['email'];
$formPassword = $_POST['password'];

//Checking password and ID via cookie
if($stmt = $conn->prepare("SELECT user_id, password FROM Users A WHERE A.email=?")){
$stmt->bind_param("s", $email);
$stmt->execute();

$stmt->bind_result($user_id, $password);
$stmt->fetch();

logToFile($user_id);
logToFile($password);

//Check the password and redirect if wrong
if(password_verify($formPassword, $password)){
logToFile('Successful Login');
$_SESSION['login_user'] = $user_id;
setcookie("owner", $user_id, time() + (86400 * 30), '/');
if(isset($_SESSION['login_user']))
redirect('ownerHome.php');
}
else{
logToFile("Login incorrect");
redirect('ownerLogin.html');
}

$stmt->close();
}
}
```

```
mysqli_close($conn);
?>
```

#### A.4.2 owner register

```
<?php

require 'db_config.php';
require 'log.php';

$conn = new mysqli($db_host, $db_user, $db_pass, $db_name);
if(!$conn){
die('Could not connect: ' .mysql_error());
}

$sql = "SELECT MAX(user_id) AS id FROM Users";
$result = $conn->query($sql);
if($result->num_rows > 0){
$row = mysqli_fetch_assoc($result);
$highestID = $row["id"];
$highestID++;
}

$hash = password_hash($_POST['password'], PASSWORD_BCRYPT);

//prepare to insert and then execute
$stmt = $conn->prepare("INSERT INTO Users (user_id, first_name, last_name, email, password)
VALUES (?, ?, ?, ?, ?)");
$stmt->bind_param('issss', $id, $first_name, $last_name, $email, $password);

$id = $highestID;
/*$stmt2 = $conn->prepare("INSERT INTO Pets(user_id) VALUES (?)");
$stmt2->bind_param('i', $id);
*/
$first_name = $_POST['first_name'];
$last_name = $_POST['last_name'];
$email = $_POST['email'];
$password = $hash;

$stmt->execute();
$stmt->close();

redirect('ownerLogin.html');

mysqli_close($conn);
?>
```

#### A.4.3 pet history

```
<?php
session_start();
```

```

require 'log.php';

if(!isset($_COOKIE["owner"])){
logToFile('Cookie not set');
redirect('ownerLogin.html');
}
?>

<html lang = "en">
<head>
<title>Pet History</title>
<meta charset = "utf-8"/>

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.min.css"
integrity="sha384-1q8mTJOASx8j1Au+a5WDVnPi2lkFfwwEAa8hDDdjZlpLegxhjVME1fgjWPGmkzs7"
crossorigin="anonymous">
<link href='https://fonts.googleapis.com/css?family=Roboto:300,400,700'
rel='stylesheet' type='text/css'>

<script>
$(document).ready(function(){
var len = 0;
if(len >= 1) {
    $('.noPets').hide();
}else{
$('.noPets').show();
}

});
</script>
</head>

<body>
<header class = "container">
<nav class="col-sm-4 col-sm-offset-8 text-right">
    <a class="btn btn-nav" href="home.html" role="button"><b>HOME</b></a>
<a class="btn btn-nav" href="about.html" role="button"><b>ABOUT</b></a>
    <a class="btn btn-nav" href="contact.html" role="button"><b>CONTACT</b></a>
</nav>

<div class = "row">
<h1 class = "col-sm-16">PetWatch</h1>
<nav class="col-sm-4 text-left">
    <a class="btn btn-nav" href="petRegister.html" role="button"><b>Add Pet</b></a>
<a class="btn btn-nav" href="petHistory.php" role="button"><b>Pet History</b></a>
<!--      <a class="btn btn-nav" href="contact.html" role="button"><b>CONTACT</b></a>
--></nav>
</div>
</header>
<div class = "petData">
<section class = "jumbotron">
<div class = "container text-center">

```



```

<h2>Pet History</h3>
<div class = "row text-center tranbox">
<p class = "noPets">You don't have any pets yet. Add a pet!</p>
<a class = "noPets btn btn-primary" href="petRegister.html" role = "button" >Add Pet</a>
<?php

require 'db_config.php';

$conn = new mysqli($db_host, $db_user, $db_pass, $db_name);

if(!$conn){
logToFile("connection test failed");
die('Could not connect: ' . mysql_error());
}

//display results
$user = $_SESSION['login_user'];
$sql = "SELECT * FROM Pets WHERE user_id = '$user'";

$result = $conn->query($sql);

if($result->num_rows > 0){
echo "<table align = 'center' cellpadding = '10' id = 'pet'><tr><th>Pet Name </th>
    <th>Age </th><th>Weight </th><th>Height </th></tr>";
while($row = $result->fetch_assoc()){
echo "<tr><td>".$row["name"]."</td><td>".$row["age"]."</td><td>".$row["weight"]."</td><td>".
    ".$row["height"]."</td><td>".$row["type"]."</td></tr>";
}
echo "</table>";
}

mysqli_close($conn);

?>
</div>
</div>
</section>
</div>
</body>

<style>
table{
border-collapse: separate;
border-spacing: 5px;
}
</style>

</html>

```

#### A.4.4 pet register

```

<?php

```

```

session_start();
require 'db_config.php';
require 'log.php';

$conn = new mysqli($db_host, $db_user, $db_pass, $db_name);
if(!$conn){
die('Could not connect: ' .mysql_error());
}

$sql = "SELECT MAX(pet_id) AS id FROM Pets";
$result = $conn->query($sql);
if($result->num_rows > 0){
$row = mysqli_fetch_assoc($result);
$highestID = $row["id"];
$highestID++;
}

//prepare to insert and then execute
$stmt = $conn->prepare("INSERT INTO Pets (pet_id, user_id, name, weight, height, age, serial)
VALUES (?, ?, ?, ?, ?, ?, ?)");
$stmt->bind_param('iisssss', $id, $_SESSION['login_user'], $name, $weight, $height, $age, $serial);
/*
$stmt = $conn->prepare("UPDATE Pets SET pet_id = '$id' name = '$
*/

$_SESSION['serial'] = $serial;
$id = $highestID;
$name = $_POST['name'];
$weight = $_POST['weight'];
$height = $_POST['height'];
$age = $_POST['age'];
//$type = $_POST['type'];
$serial = $_POST['serial'];

$stmt->execute();
$stmt->close();

redirect('ownerHome.php');

mysqli_close($conn);
?>

```

## B Specification

### B.1 Functional Requirements

- An owner and guest login screen
- High website reliability and availability
- Sensors on a dog collar to track a pet's:
  1. Movement with accelerometer

2. Temperature with temp sensors
  3. Eating/drinking habits with NFC chips
  4. Bathroom/door waiting habits with NFC chips
- Send sensor data to the website to store

## **B.2 Non-Functional Requirements**

- Provide owners (and interested parties) with updated statistics about their dog
- Easy to understand interface for the statistics
- Provide long battery life for the collar
- make the collar durable
- ensure pets enjoy wearing the collar

## **B.3 Design Constraints**

- website hosted in the Engineering Computing Center
- website works across common web client platforms
- collar can be worn by multiple sized pets