Group A
EEL 4915
12/07/21

Pet Pal: The Smart Pet Collar

Pet Pal
Keeping Your Companion Happy and Healthy

University of Central Florida
Department of Electrical Engineering and Computer Science

Dr. Lei Wei
Dr. Samuel Richie
Senior Design I/II

Group A
Andres Graterol - Computer Engineering
Zac Kirksey - Computer Engineering
Hanh Le - Electrical Engineering
Hunter Miller - Electrical Engineering

TABLE OF CONTENTS

List of Figures

List of Tables

1. Executive Summary

The domestication of animals is one of the crowning achievements of the human race. This helped us survive in even the most difficult eras and cemented our place as the apex of the animal kingdom. As the need for animals to assist us in many of our daily tasks has mostly been replaced with machinery, animals as pets or companions has become far more common. For the most part, gone are the days of plowing fields with oxen, and now exists a time where animals walk alongside humans. As the importance of our pets has grown in society, there are whole markets dedicated to the care and health of one's pets. This project aims to bring a new level of security and confidence to the owner of a pet.

Smart Collars as an industry have a lot of room to grow and expand on the number of features that are available. They are quite useful and make the consumer's life easier at times, but the cost of many of these smart collars makes them not worth the number of features that the customer is receiving. In our design, we explore and implement multiple features for the benefit of the consumer while keeping the total cost to the customer rather low. These features will include monitoring the health and general wellbeing of the animal and the location from GPS, and all of this data will be gathered and displayed on a mobile application.

The system will consist of more than just a smart collar. An application will be developed in addition to the smart collar. This will create a centralized and convenient environment for the owner to monitor the health and location of their animal. The goal of this is to make the process as easy and stress free for the consumer as possible.

2. Project Description

A pet is not just an animal, it is a good friend and a trust-worthy family member. Thus, the need of treating the best companion of humans is significant and has become a necessary part of our lives. However, people who own pets can completely understand the struggle of time consumption and high expenses, such as vaccinations, food, startup supplies, preventative medicine, … etc. For example, a dog owner is responsible for walking their dog three or four times a day to allow them to go to the bathroom and do physical exercise. Our goal is using advanced technology to create a low-cost smart pet collar with features such as a GPS tracker, a heart rate monitor, and a calming monitor. These features will make owning a pet less challenging for people.

As we're all aware, dogs and cats are playful pets, there is always a possibility that they will go missing for several days, or worse, a few years while they have their potty breaks. For that reason, when people decide to adopt a pet, they are offered to insert a microchip in the pet. The basic microchip can give the information of the owner when it is scanned but it can't send the location of the missing pet to its owner. On the other hand, the thought of inserting a tiny electrical device under your pet's skin is terrifying to a lot of people. Therefore, our approach to this problem is to create a removable GPS collar which gives the pet owner full control of their pet's location. Of course, it will come with an application interface that is user-friendly and can be downloaded to any android device. In addition, there will be an alarm system to notify the pet owner if their pets' location is moderately far away from their device. This feature will allow people to let their pets go potty in their yard by themselves without worrying about the pets running away.

Beside the anxiety of losing pets, pets' health is also a significant concern of the pet owners. According to statistics, about 10% of dogs have heart disease (1) which gives them only 6 to 14 months left to enjoy their lives (2). Hence, it is very important to spot the early symptoms so that the owners can help their pet to have appropriate treatments and to extend their lives. Nevertheless, the pet owners have to pay a high price to buy a pulse monitor for their pets or they have to detect their pets' heart rate using their hands, of course, this method takes time and requires medical knowledge. Our smart collar has a pulse sensor that can measure the pet's heart rate and send the data to the mobile device. This way, pet owners will easily be able to discern if their pets have irregular heart rate. Moreover, the temperature sensor is also a part of our collar. Too high or too low temperature is extremely dangerous for the pets since it can damage the pets' internal organs or can be fatal (3). The temperature sensor informs the owner of the pet's condition, as the result, the owner can fully take care of their pets and comfort them whenever they need it.

As mentioned above, our goal is bringing convenience to the people who own pets. We believe with our smart collar, people can spend time with their family while they are still able to fully look after their pets. The distance between the pet owners and their pets has been narrowed down using our low-cost smart collar.

2.1 Motivation

Engineering is a major that does not only involve math and physics, it also requires discipline, teamwork, analysis, high accuracy, and high precision. We have been through four to five years in the university to acquire from basic knowledge to advanced technical concepts in order to understand tasks that we need to do as an engineer. Through the Senior Design course, we are pushed to work as a team, do research, and apply what we have learned from the past to create a project that is useful in real life. At the beginning, our group listed 15 ideas using the question “What can we do to improve the quality of life?”. One out of the 15 ideas was chosen, the Smart Pet Collar.

According to the National Pet Owners Survey, there are about 67% of US households owning at least one pet (8) and in a year, 47% of dogs are rehomed. We realize one of the reasons for giving up furry companions is that not everyone has a good quality time to spend with their lifelong friends. Thus, our Smart Pet Collar is created as a solution for people who want to carefully take care of their pet as well as the pet abandonment problem.

Although in the pet industry, there are calming collars, GPS collars, and different devices which are used to measure a pet's heart rate or temperature, collars that can do multiple tasks are sparse and mostly expensive. For that reason, the pet owners don’t look at those devices as their solutions. We want our Smart Pet Collar to provide entirely services to the users with an affordable cost as possible. Besides having a potential market, the Smart Pet Collar has a lot of room for improvement and can become a device that widely used in the future

2.2 Goals

As mentioned above, our Smart Pet Collar will contain GPS, speaker system, heart rate and temperature monitors. The collar will be controlled and observed using an application on Android devices. This is a project that we believe evenly distributes tasks between Electrical Engineering and Computer Science.

The main goal of this project is creating a smart collar that can do multiple tasks including tracking, recording heart rate and temperature, and playing calm music or the owner’s voice. By combining different separate roles in one product, we hope that our customers can experience the convenience while they are taking care of their pets. We will find and consider various parts for each task. Parts will be chosen based on the given specifications or if there is a chance, we will choose the parts that potentially improve our product. While working on our project, we always have to keep in mind that the internal structure of animals, particularly dogs, is far different from the body structure of a human. GPS and speakers are common functions that we can easily do research, study, and design. On the other hand, the heart rate monitor and temperature monitor need to be placed at specific places on the dog’s neck in order to collect the correct data.

The second goal that we want to achieve is that all of the mentioned functions can be controlled using a friendly-user application on an Android device. The microcontroller will receive the data from the heart rate monitor, temperature monitor, and the GPS module. Then, all of the information will be sent to the Android device through a bluetooth module. By doing this, our users can easily have a quick update on their pets' health as well as location. The microcontroller and the bluetooth module are chosen carefully in order to gain the best signal transmission possible.

The size of the collar is also a target that we aim for. Since we are trying to combine multiple functions in one collar, it is obvious that it might have a large and weighty size of system. As we are aware, this product will be used by medium or large dogs, an enormous collar carried on a dog's neck is definitely not a prospect we want to see. In addition, our initial intention when we design our smart pet collar is that the owner and their pets both can feel comfortable and convenient when they use the collar daily. Thus, besides choosing appropriate parts which fit to our specifications, we especially pay attention to the size of these parts, the smaller parts are, the better. We endeavour to minimize the size of the collar in order to make sure that our product is useful and practical for daily use.

Another goal that we want to reach in this project is producing a low-cost product. A collar with a speaker system that plays calming music for dogs is about $30 to $50. A GPS collar that can track pets is over $40. Eventually, a regular collar that combines multiple tasks is over $120, and if it looks trending fancy, it can cost up to $800. Price is one of the reasons why smart pet collars are not widely used. We believe we can reduce the product cost while designing a product with the same quality.
Working as a team is one of the most important skills that we want to achieve through this project. Because of miscommunication between engineers, there were disasters causing millions or even billions dollar damage. One of the well-known cases we want to mention is the Mars Climate Orbiter disaster. In that incident, two engineers in a team were using different units while working on the orbiter which led to the disintegration of the spacecraft. This project costs 327.6 million total for the orbiter and lander (9). We are separate individuals with different time schedules and different ideas as well as knowledge. Not only that, each of us also has our own perspectives and self-esteem, However, when we are working together, we need to respect our teammates by listening and understanding their opinions. Our team shared ideas and each person has responsibility for the parts of the project that they said they will take care of. Without team working, we couldn't go this far and our project can be barely completed.

Pressure management and time management are necessary skills that we all have to face when we have a job. More than ever, as students, we experience working under pressure for years. We have several deadlines for assignments which are assigned mostly every week. Sometimes, there are assignments that are easy to do because we were provided with full materials and background information. However, this project is completely different from our usual homeworks. First, we need to work as a team to find and agree on a project idea. Then, we have to do a lot of research to learn how to design our project. We have to gather all the theoretical knowledge we have learned before to apply to create a final product. In addition, there are students who are taking 4 classes at the same time. Thus, if we can't effectively organize our time and have a specific plan, we won't be able to finish our assignments on time.

Through this project, we expect that we can gain some engineering skills as well as accomplish a practical product. Our group has precise specifications which we hope that we can achieve during Senior Design 1 and Senior Design 2.

2.3 Related Works

Pet Pal is not the only product that helps people to take care of their pets. In fact, there are a lot of pet care products on the market right now. These products have their own advantages and disadvantages that we can learn and use to improve our product. Most of the relevant products do not offer as much as features like Pet Pal. The particular reason for that is it may lead to high production cost and time consuming. A smart collar that has only one feature can cost up to $100. We also found products that are in the designing phase, this means creating a complete smart pet collar which is comfortable for pets to wear and easy for the owner to use are not simple. Below are some of the products that have similar technologies to Pet Pal.

2.3.1 Pet Pace

Pet Pace is a competitor smart sensing dog/cat collar. It wirelessly collects the pet's vital signs and behaviour patterns. This system monitors: body temperature, heart rate, respiratory rate, activity, body posture, calories burned, and heart rate variability. It sends all of this data to the "PetPace Gateway" which then transfers the data to a cloud database. On the application side, the data is pulled from the cloud database. Here you can check for health alerts, perform instant check-ups, and add reminders. In addition you can add your pet's veterinarian to the application as well. However, there is a catch. In order to use the health analytics on the application, you must also purchase a yearly subscription service. Without this subscription service, the collar and application are pretty much rendered useless. Upon purchasing, you are forced into buying a bundle consisting of the collar and a 1 year subscription; a package priced at $200. After the year has passed, renewing your subscription is $100 per year(12).

Figure 2.1
Pet Pace Smart Collar

2.3.2 Fi

The Fi collar is specifically designed for tracking only. Unlike PetPace, this collar does not offer any health tracking services or analytics. It has GPS tracking technology and communicates with the GPS with LTE-M technology in order to maintain power-efficiency. This system also comes with a corresponding application that has maps integration to show you things such as the path traversed by your dog and last seen location. The application will also send notifications to your phone when the pet has left the designated "geofence" area. The collar also comes with a red LED light to assist in locating your pet. This collar is priced at $150^{(13)}.

Figure 2.2
Fi Smart Collar

2.3.3 Link

This smart collar provides a combination of both tracking and health alerts. This collar uses the geofence feature to set safe areas that the dog is allowed to traverse in. It also has real-time location service data and a map inside of its corresponding application. On the health side, it has features to monitor activity and send temperature alerts whenever the dog's environment gets too hot or too cold. A unique feature to this collar are the training controls: chimes and vibrations that positively reinforce behaviors for training purposes(15). This system also requires a service plan to access all of its mentioned features. Currently there is no way to purchase this system as the website states that they are currently developing improvements.

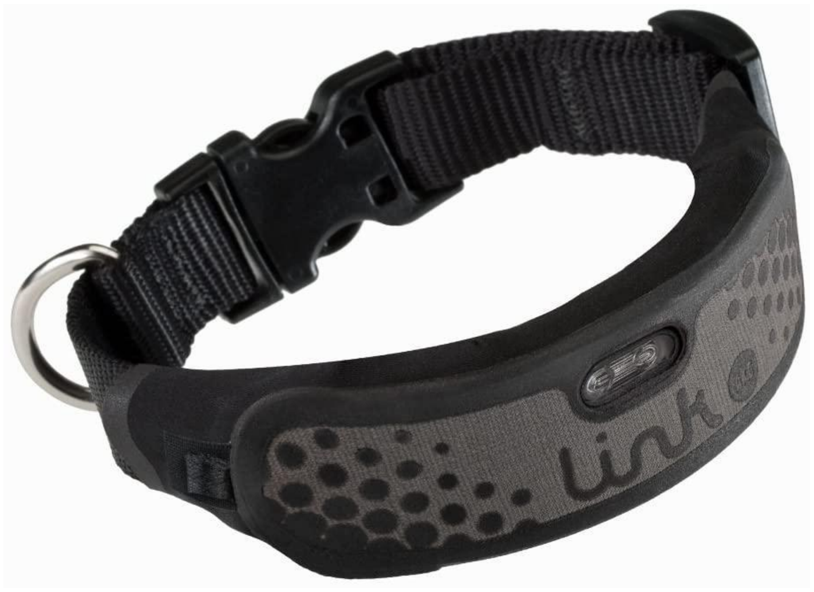


Figure 2.3
Link Smart Collar

2.3.4 ThunderEase

Does not have many functions like other smart pet collars, the main purpose of ThunderEase is to calm the pet in multiple situations. Instead of using advanced technologies, such as playing relaxing music, or noise canceling through a speaker controlled by mobile devices, ThunderEase contains a chemical sense called pheromones which is activated by pet's body heat. Pheromones is a chemical smell that animals produce which can change the behavior of surrounding animals. There are four different types of pheromone, release pheromone, primer pheromone, signaler pheromone, and modifier pheromone. Each type of pheromone will have its own features. For example, modifier pheromone is usually found on newborn animals which can make them feel safe, comfortable, and reduce stress, which is the type of pheromone used for ThunderEase.

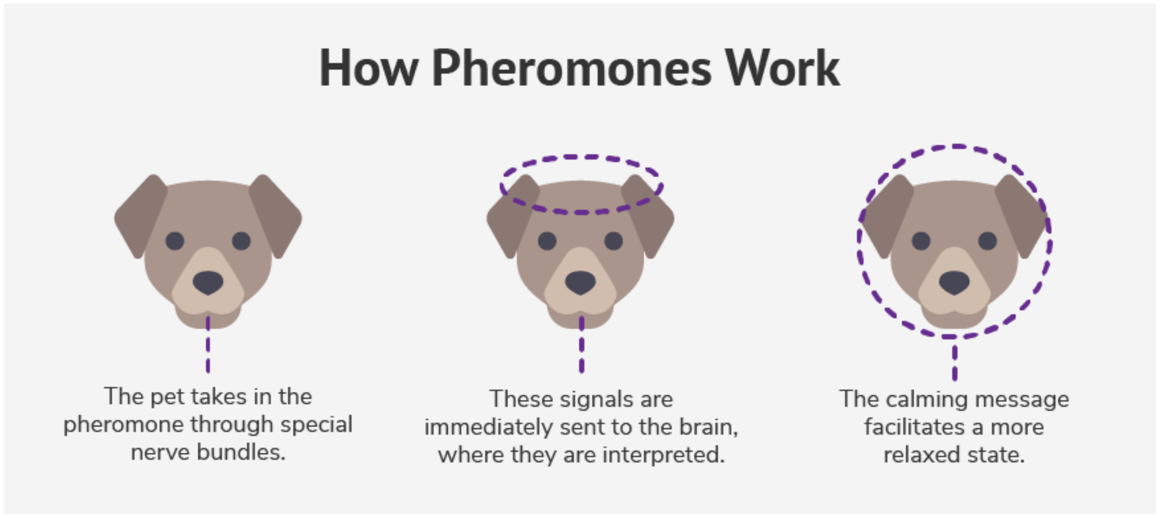

How Pheromones Work
The pet takes in the pheromone through special nerve bundles.
These signals are immediately sent to the brain, where they are interpreted.
The calming message facilitates a more relaxed state.

Figure 2.4
The Effect of Pheromones

ThuderEase, as the manufacturer claims, can reduce problem barking, destructive chewing, uneasiness in a new home or environment, fear of loud noises like thunder or fireworks, vet anxiety, anxiety when boarding, and separation anxiety. This seems like a reasonable product for the owners who do not want to use technologies. Nonetheless, this collar needs to be replaced every month because after four weeks, the collar will run out of the pheromone, which makes the collar unusable. There is no way to charge or reuse the old collar, this will be a serious problem to people who care about the environment and thus, this product will not be supported by a large number of customers. Moreover, the price of this collar is 22 dollars, which is pretty high for a single feature collar.

Figure 2.5
ThunderEase Smart Collar

2.3.5 How our System Competes

Pet Pal is a combination of all the systems described above. It will boast accurate real-time GPS location services with Google Maps integration on our android application. It will also come with built-in speakers, so that the user can request that an alert sound be played through the collar. This alert ping will assist the owner in pinpointing the exact location of the pet once they are within range. On the health side of things, Pet Pal will be able to poll the heart rate and temperature sensors on command to give the user an accurate representation of how their pet is feeling at the time. All this data will be stored inside of a database. All of this will be done without the need for a subscription-based service! All features of Pet Pal will be ready to use out of the box right after the user creates their account and registers their pet and collar with the system.

	Pet Pace	Fi	Link	ThunderEase	Pet Pal
Body Temperature	✓		✓		✓
Heart Rate	✓				✓
GPS		✓	✓		✓
Calming Pet				✓	
Price	$200 + subscription $100/ year	$150	Currently developing improvements	$22	$65

Table 2.1
Related Works Comparison

2.4 Relevant Technologies

In this section, we will list some of the theoretical concepts which are used in our design. Although we have learned most of the concepts from other courses before, it is necessary for us to do research and have a quick review to understand more about the technologies that we will apply in our project. The below contents are just summaries of basic knowledge of different technologies. In order to practically apply these technologies in our product, we have to look at the specific datasheets of the parts and consider their capability. Each technology has its own key variables which helps us to choose the appropriate components for our product

2.4.1 MOSFET

MOSFET stands for the metal-oxide-semiconductor field-effect transistor, is the kind of transistor that we will use for a lot of applications in our design. There are two types of MOSFETs, N channel MOSFET and P channel MOSFET. Each type of MOSFET has its own characteristics so we need to understand how a MOSFET works in order to select a suitable MOSFET for our product.

The figure below displays the structure of N channel MOSFET. It will have a p-type substrate, on top of the substrate, there are two n-type regions and an oxide is placed between those two n-type regions, this oxide will ensure that two n type regions are separative. Finally, a layer of metal is put on top of the oxide to create a complete n channel MOSFET. On the other hand, the P channel MOSFET will also have an oxide insulator and a metal on top of it. However, instead of two n type regions, p channel MOSFET will have two p type regions, and instead of a p-type substrate, it will have a n-type substrate.

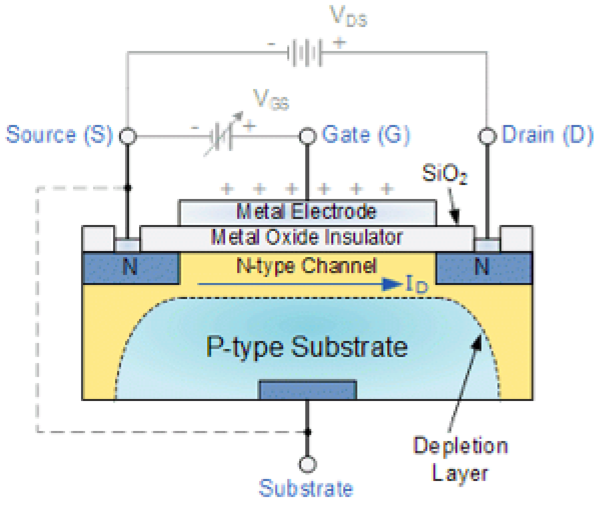
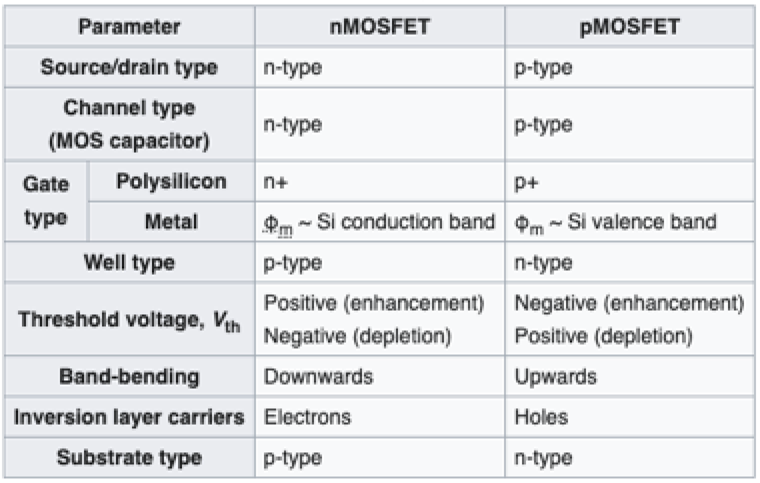


Figure 2.6
N-channel MOSFET Structure

In total, a MOSFET has three terminals, the gate, source, and drain. One of the n-type regions will be assigned to the drain terminal and another n-type region is assigned to the source terminal. The gate terminal is actually the metal which is on top of the oxide. As we have learned, the P-type substrate contains holes while n-type regions contain electrons. When we supply a positive voltage for the gate terminal, the holes that are between two n-type regions pushed down, create an empty area between two n-type regions. When the voltage difference between the gate terminal and the source terminal is higher than the threshold voltage of the MOSFET, that is when the MOSFET is turned on and there is a current flow from the drain terminal to source terminal.

In our project, we will need to use both N-channel and P-channel MOSFET. Thus, it is important to tell the differences between two types of MOSFETs.



Parameter		nMOSFET	pMOSFET
Source/drain type		n-type	p-type
Channel type (MOS capacitor)		n-type	p-type
Gate type	Polysilicon	n+	p+
	Metal	Φ_m ~ Si conduction band	Φ_m ~ Si valence band
Well type		p-type	n-type
Threshold voltage, V_{th}		Positive (enhancement) Negative (depletion)	Negative (enhancement) Positive (depletion)
Band-bending		Downwards	Upwards
Inversion layer carriers		Electrons	Holes
Substrate type		p-type	n-type

Figure 2.7
Comparison between N-channel and P-channel MOSFETs (49)

The table above demonstrates the differences between NMOS and PMOS. For example, NMOS will have n-type source, n-type drain and n-type channel while PMOS have p-type source, p-type drain and p-type channel. In addition, it also shows us what type of substrate NMOS and PMOS have. One of the most important characteristics of the MOSFET is its threshold voltage. Different types of MOSFETs will have different values of threshold voltage and we must know the threshold voltage of a MOSFET before using it in the design. Threshold voltages also change depending on the mode of the MOSFET. A MOSFET has two modes, enhancement and depletion modes. Nevertheless, we just only focus on enhancement mode for now because we do not need just depletion mode in our project. As shown in the comparison table, for enhancement mode, NMOS will have a positive threshold voltage and PMOS will have a negative threshold voltage. This threshold voltage will be used to compared to the voltage difference between gate terminal and source terminal, which is calculated using the formula:

$$V_{GS} = V_G - V_S$$

The voltage difference between gate and source terminal must be bigger than the threshold voltage in order to activate the MOSFET. For instance, assume that the voltage at the gate terminal is 0 volt. It is required that the source voltage must be negative in order to activate a N channel MOSFET, or positive in order to turn on a P channel MOSFET in enhancement mode.

2.4.2 Voltage Regulator

The purpose of using a voltage regulator is to get a desired constant output voltage regarding input voltage. We can easily build a linear voltage regulator using six resistors, two capacitors, an amplifier, and a transistor. However, the weakness of this design is that it does not have any short circuit or thermal protection. It also takes time to buy components and design them on Eagle. There is a simpler way to build a linear voltage regulator which is using a voltage regulator chip. A voltage regulator chip can produce either negative or positive output voltage depending on what voltage regulator family that we are using. For some of the voltage regulator family, we can tell the values of the output voltage that is produced by looking at two last numbers of the voltage regulator's name. For example, below is the schematic of the 7805 or 7812 voltage regulator, which means 7805 regulator will produce 5 volts, while 7812 regulator will produce 12 volts.

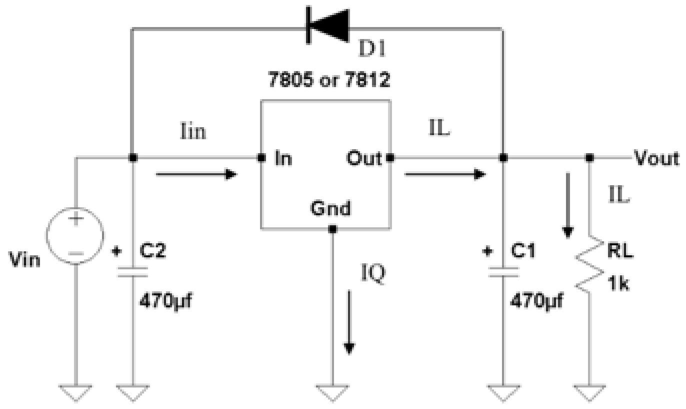


Figure 2.8
Example of Schematic of Linear Voltage Regulator

There are always at least two capacitors connected to the voltage regulator, one at the input port and another one at the output port. The reason why we need to use capacitors is that we want to eliminate as much noise from the input voltage as possible. Recall that this is a linear voltage regulator, which will produce an unchanged DC voltage, so our final target is to have an output voltage without any variation. The schematic above also shows us how to safely operate a three terminal voltage regulator. As we know, the 7805 voltage regulator will produce an output voltage of at least 5 volts. However, if the input voltage has a smaller value compared to the expected output voltage, it may cause heavy damage to the voltage regulator. One of the important things that we need to pay attention to in our design is the values of two capacitors. As mentioned above, we do not want to have an input voltage that is smaller than the output voltage, and if the value of capacitor C1 is larger than the value of capacitor C2, the circumstance where the input voltage is smaller than the output voltage may occur. In addition, we definitely want to avoid the reverse polarity of the input voltage. The schematic above uses a diode to protect the regulator from the situation where the output voltage is larger than the input voltage. This diode will prevent the current from flowing through the voltage regulator when Vout is higher than Vin.

We can use the same voltage regulator to produce a higher desired output voltage just by connecting two more resistors. One resistor R1 is placed between the output pin and ground pin of the voltage regulator. Another resistor R2 is connected from ground pin to ground. This is called the adjustable voltage regulator. The formula we use to find the values of two resistor based on the desired output voltage is:

$$Vout\ = Vreg(\frac{R1}{R2} + 1)$$

Vreg is the regulator output voltage, and each of the voltage regulators will have its own Vreg. Moreover, the dropout voltage is also an important information of the voltage regulator that we need to know. Both the regulator output voltage and the dropout voltage of the regulator can be found in the datasheet. For our project, since we use a pretty low input voltage, it is superior that we can find a voltage regulator with a low dropout voltage.

2.4.3 Conductive Threads

Conductive threads carry current in the same way that wires do. These conductive threads fall into two separate categories: silver-coated threads, and stainless-steel threads. Silver-coated threads are exactly what they sound like - they are nylon threads with a silver coating which promotes conductivity. The resistance of these threads can vary a great deal, so it is useful to have a multimeter on hand to ensure that the resistance of the thread is not interfering with the current flow. Due to the interior nylon threading, this type of thread is easily damaged by nearby soldering as the nylon core is too susceptible to the heat required to solder.. Silver-threading works best in small, contained projects where the acceptable resistance is high. On the other hand, stainless steel threads do not have a nylon thread core. Instead, they are completely made out of stainless steel. They may be a little difficult to sew with but they have relatively low resistance. Another plus for stainless-steel threading is that it is resistive to soldering. It can easily withstand the temperatures of nearby soldering without damaging the thread(16).

2.5 Requirements Specifications

The American Heritage Dictionary describes a specification as “a detailed and exact statement of particulars, a statement fully describing something to be built”. All of the requirements that have been specified in *Table 2.1* below must be satisfied in order for our Pet Pal design to be considered successful and to fully meet the expectations set by a potential customer or end user. The requirements specified below were discussed and agreed upon by the development team behind this project, Group A. All of the following engineering requirements below meet the four properties set by IEEE [IEEE Std. 1233-1998]. They are abstract, as in they specify what the system will do rather than how it will be implemented. They are verifiable, as you can see the specification column has a list of measurements that will be used to verify whether or not the corresponding requirement has been met. They are unambiguous. And lastly, they are traceable. In addition to these four properties, our design requirements meet a fifth property - they are considered realistic.

Req #	Requirement	Specification
1.1	Power requirement	Max of 10 W
1.2	Battery powered (replaceable), or rechargeable	15 V max
1.3	Bluetooth Module	50 ft Range, 3.3 V, 2 W
1.4	GPS Module accuracy	6 feet
1.5	Heart Rate Monitor	90% accurate
1.6	System Casing dimensions	2 inch width
1.7	Collar System weight	< 2 pounds
	DEMONSTRABLE	
1.8	Temperature Sensor	90% accurate
1.9	Speaker	3.3 V, 2 W, 50 dB
1.10	Mobile Application communication	< 30 sec response time

Table 2.2
Engineering Requirement Specifications

As can be seen above, all of the above key features of our system pertain to the sensors and therefore the functional features of our system. Non-functional requirements were not given priority consideration when determining this table. When it comes to the demonstrable section (the items that will be showcased in the final demonstration in Senior Design 2) these were carefully selected to be the best representation of whether or not the system works as intended. These three features will be showcased through the android studio application that is being designed for use with the Pet Pal smart collar.

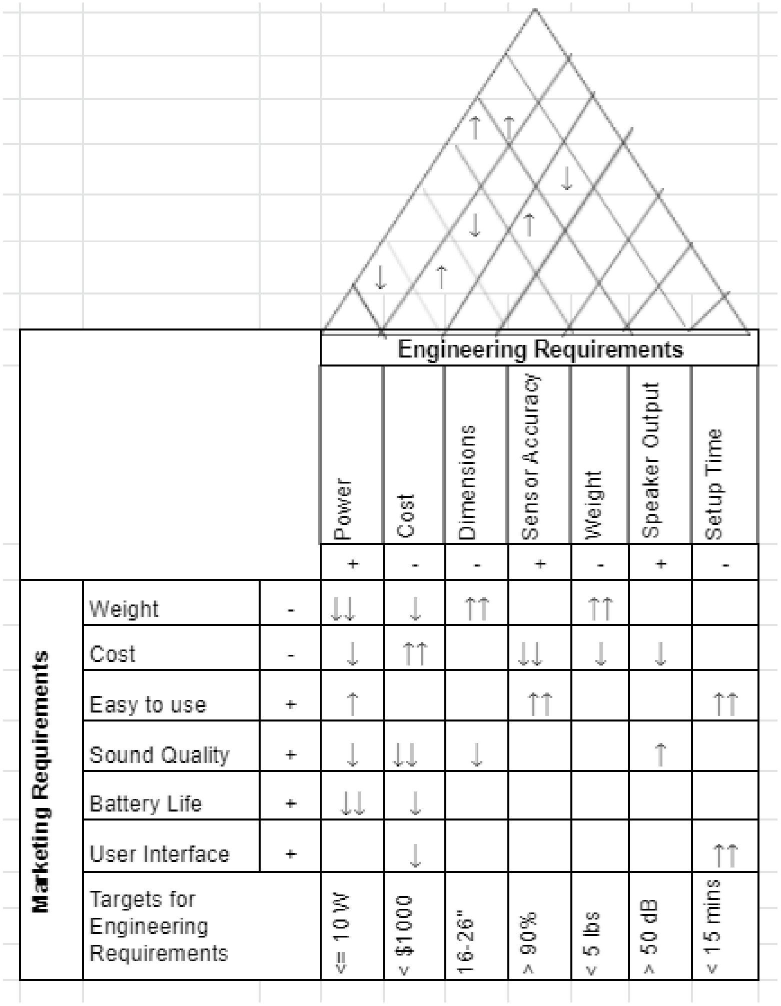
Stretch Requirements:

2.1	Solar cells	5V

Table 2.3
Stretch Goals

The "Stretch Goals" table shown above represents a list of goals that we have in-mind for our Pet Pal smart collar system but that might not be necessarily in-reach for two semesters of work. These requirements were moved to their own table in order to differentiate from the main system goals and to also meet the property of realism as described above.

The Engineering-Marketing Trade Off Matrix, more commonly known as the House of Quality is meant to be a visual representation of the relationships between our marketing and engineering requirements. Every marketing (row) and engineering (column) requirement is assigned a polarity (positive or negative) in accordance with whether we want to maximize or minimize that certain quality. The entries in the body of the table determine the type of correlation between the intersection of engineering and marketing requirements. The legend found below *Table 2.3* shows more explicitly what each arrow type represents. When it comes to the "roof" of the house of quality, it shows the correlation between the engineering requirements themselves. For example, power and cost have a negative correlation - increasing the power (a positive polarity trait) supplied to the system is going to increase the cost of the system (a negative polarity trait) which we do not want.



			Engineering Requirements						
			Power	Cost	Dimensions	Sensor Accuracy	Weight	Speaker Output	Setup Time
			+	-	-	+	-	+	-
Marketing Requirements	Weight	-	↓↓	↓	↑↑		↑↑		
	Cost	-	↓	↑↑		↓↓	↓	↓	
	Easy to use	+	↑			↑↑			↑↑
	Sound Quality	+	↓	↓↓	↓			↑	
	Battery Life	+	↓↓	↓					
	User Interface	+		↓					↑↑
	Targets for Engineering Requirements		<= 10 W	< $1000	16-26"	> 90%	< 5 lbs	> 50 dB	< 15 mins

Table 2.4
House of Quality

Legend

- + Positive Polarity
- - Negative Polarity
- ↑ Positive Correlation
- ↑↑ Strong Positive Correlation
- ↓ Negative Correlation
- ↓↓ Strong negative Correlation

2.6 Block Diagrams

This project overview block diagram shown in *Figure 2.10* describes the hardware part of the Pet Pal smart collar system. The direction of the arrows in between the blocks specifies in which direction the data flows. Each arrow is also accompanied by a brief description of how the corresponding systems interact with each other. Whenever applicable, the communication protocol is also specified. The block labeled "Mobile Application" is further expanded upon in the following section and in *Figure 2.10*.

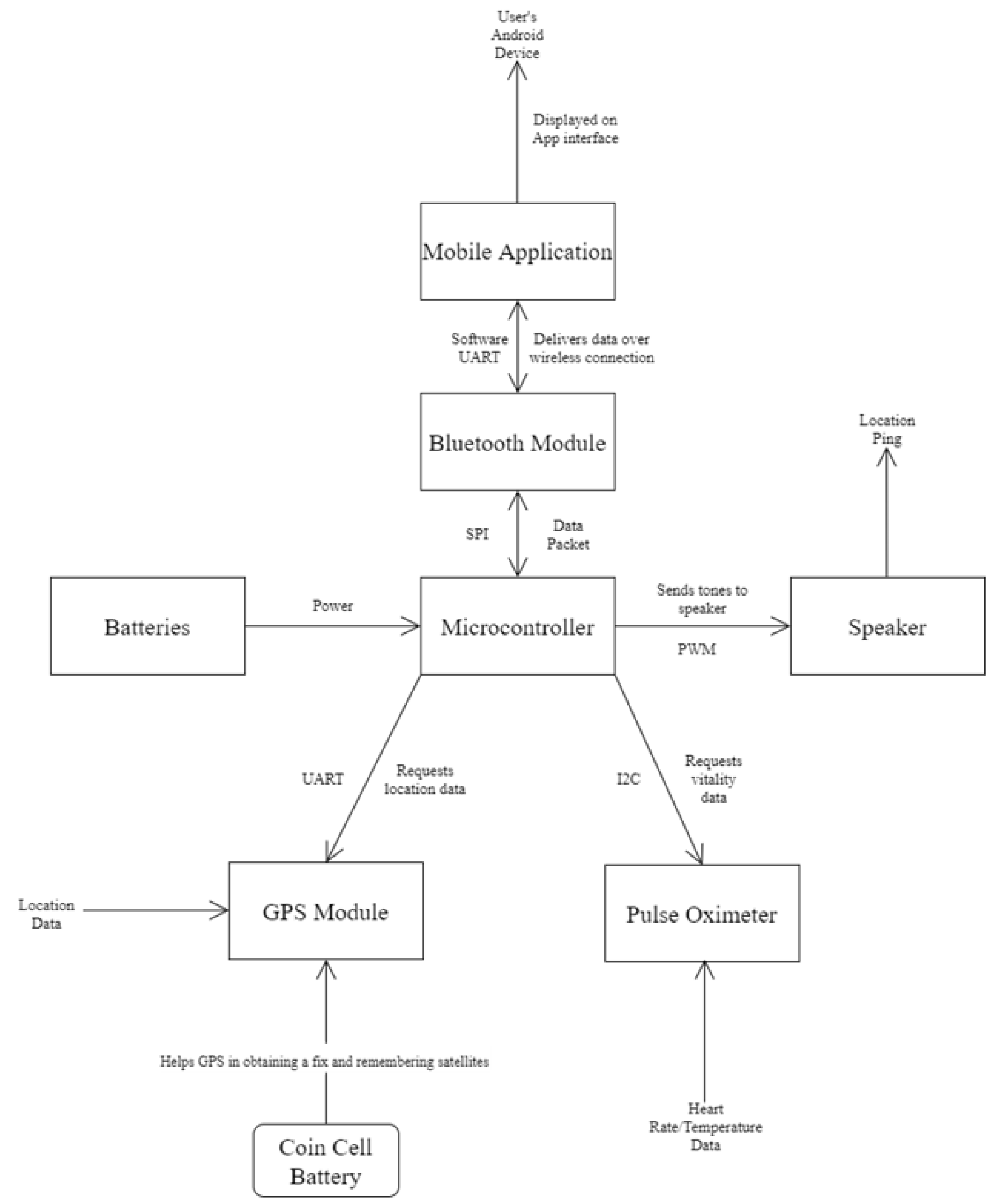


Figure 2.9
Hardware Block Diagram

This next section describes the software side of the smart collar system. More specifically, the mobile application that we are going to develop for Android Phones. The mobile application is what is going to be controlling all of the sensors that are a part of our system. The mobile application will connect to the microcontroller via bluetooth. From there, the microcontroller will request data from the sensors in accordance to what the user requests on the application side. For example, if a user travels to the health page and requests to check their pet's current heart rate, the microcontroller will request the data from the heart rate sensor at that specific moment and display it on the application's user interface. All of the data that is requested from the microcontroller and in turn the sensors will be stored inside of a database for analytic purposes. This software diagram describes the flow of the application as well as the different pages that will be available for the user to traverse through. For a more in-detail look at the software architecture, please refer to the *Software Implementation* section of this document.

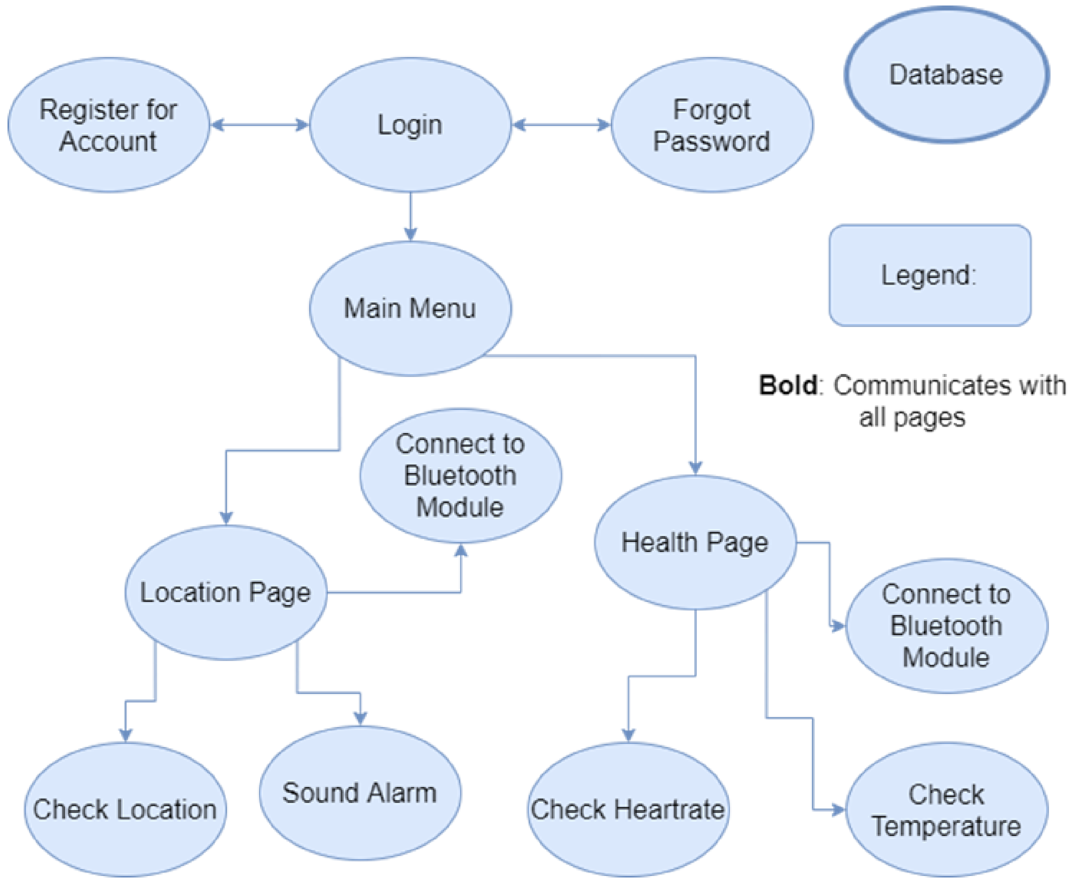


Figure 2.10
Diagram of the Software Flow

3. Research and Part Selection

In order to achieve the goal of a low-cost and effective smart collar, the parts we select and subsequently use are especially important. While having a low total cost is important, what is of paramount importance is to have a well functioning device. Picking the correct sensors to achieve maximum functionality while still retaining a low cost will be a primary method of attaining this goal. To make this idea more concrete, low-cost components which explicitly fulfill our requirements are what we search for.

To narrow down the massive list of potential usable components, we use our predefined requirements. The communication protocol we choose, power usage, as well as sensor accuracy will be the main criteria by which to judge our components. In addition to these requirements, there are a couple other considerations to keep in mind: number of available pins on our microcontroller, the necessary voltages to power our various devices on the board, and the number of measurements taken by each of the sensors per second.

The sensors on the board will be outputting data to the microcontroller, and it is important that this data be quickly and readily available for the microcontroller to prepare to broadcast to the mobile application through the use of a bluetooth module. The communication protocol which we choose to use is based mostly on two factors: bandwidth and data transmission rate. The speed at which the sensors relay data to the microcontroller, and in turn to the Bluetooth module, must not exceed the bandwidth of our chosen communication protocol for our sensors. There are three main protocols to choose from: UART, SPI, and I2C. Each has significant pros and cons, but for the purposes of this project, the protocol that needs the fewest number of buses will take precedence. This is not only more simple, but allows for us to make use of the maximum amount of sensors without compromising our design. It is worth noting that these do not have a significant impact on the price of each sensor. SPI has the highest rate of data transfer, so likely this will be the protocol we will use when sending data to the bluetooth module to be broadcast to the mobile application. Another advantage of SPI is that it has a very high bandwidth of close to 60 Mbps. This is more than enough bandwidth than would ever be used by these sensors which would allow for future improvements in the design if we so desired. When possible, our preferred protocol will be UART. This is because of the ease of use in addition to the fact that it requires very few buses when compared to SPI which requires three to four buses. I2C, while only requiring two buses, would become too complex for our design because of the multi-slave, multi-master design. In addition, UART does not require the use of a clock unlike SPI. When compared to I2C, UART also uses less space on the PCB because of the relatively simple design. Many Microcontrollers have another communication protocol called USART. USART uses asynchronous/synchronous timing to make sure the receiver and sender of the data are properly clocked. Also, USART supports more standards of communication. While not vastly different, these differences could be useful during our implementation and shall be considered. USART can be run in the same fashion as UART, and in our implementation we shall be treating it as such.

The power usage of our devices is also an important consideration. A couple of the main aspects to pay attention to here is the total current draw and wattage that the components need. This data will, in turn, provide a more concrete power requirement for the system allowing us to pick a suitable battery. The components will also generate heat which we will need to be conscious of when fabricating our board. In order to avoid the need for extra circuitry on the board, the sensors shall each have the same necessary input voltage. Most components have a range of power that is accepted. Since 3.3 volts is both more common and much more cost effective, the range for the power requirement we will be using when selecting our components will include 3.3 volts. The overall power consumption will be important to keep in mind in order to extend the life of the battery. Whenever possible the sensors will be in a low-power mode so as to conserve power. Only when actively taking measurements will the sensors need to be operating at full capacity. To circumvent certain issues in our implementation, the power circuitry that is designed is important. Our design will likely be solely battery powered and, as a result, there need to be certain fail safes in place in order to prevent any failures in the application. The first of these is protection against reverse polarity. Since the design is dependent on the user replacing the battery in the system, the possibility for installing the battery backwards exists. This would be catastrophic if the effects of the reverse polarity were to go unchecked. To prevent this, a protection circuit will be put in place which will prevent any reverse polarity voltage from passing. Another issue is the sensitivity of the microcontroller and sensors. The voltage that these components receive should be consistent and stable so as to have the best results. Having voltage that is too high or too low will cause problems with these sensors and the microcontroller. To implement this, there will be a voltage regulator to keep the voltage at 3.3 volts and 5 volts.

Sensor accuracy will likely be one of the defining characteristics of the collar. This will also be the most challenging requirement to fill. This is a result of the inverse relationship between sensor accuracy and cost. In order to offset this, we apportioned a generous amount of our budget to the purchase of our sensors and would ideally like to achieve a minimum of 90% accuracy for each of them. The microcontroller will also have a built in error detection system. This involves identifying erroneous measurements and eliminating them before they are broadcast to the mobile application. The microcontroller is one of the most crucial parts of the system. In selecting a microcontroller, the main things to consider are the number of I/O pins, the supported communication protocols, power usage, and ability to efficiently process data. Using these metrics as a guide, selecting a microcontroller from the 10,000s on the market becomes much easier. One of the roadblocks in this project is that while many microcontrollers can support multiple peripherals, having the capability to support at least five peripherals can be difficult to find.

3.1 Microcontrollers

The microcontroller is the core of the PCB. This section describes the parts that were considered for our design. In addition to this, described below are many of the features and requirements for the microcontroller which will be implemented in the final design and demo. Some of the relevant and related technologies are also discussed in this section to assist in picking the best microcontroller for our implementation.

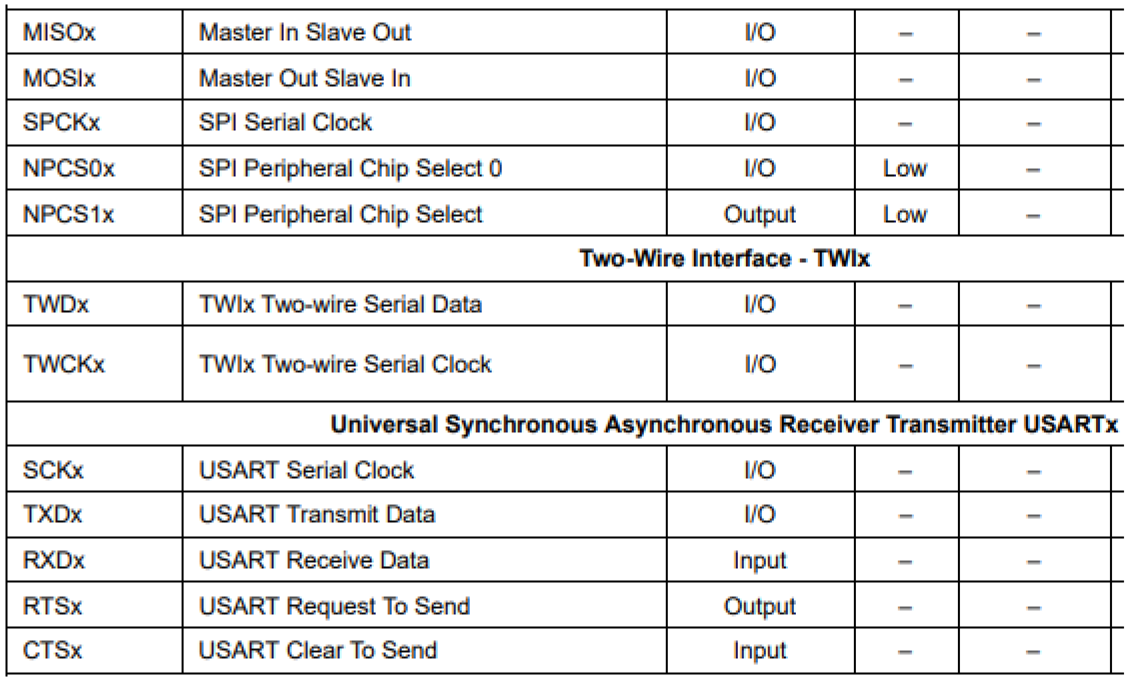
3.1.1 Microcontroller Considerations

When selecting the proper microcontroller to be at the center of our design, many aspects need to be taken into consideration. As previously mentioned, the communication protocols will be of paramount importance. As a result, the microcontroller that is selected must be able to support all three of these and must support multiple of each in order to allow flexibility in the selection of our other components. Another consideration is the operating voltage. To keep our design as simple as possible, our requirement is that our microcontroller and other components will be powered by a Lithium Ion battery which will supply 3.3 volts to the system. Thus, our microcontroller must fit this 3.3 volt requirement. The size of our selected MCU is important as well for two reasons. As this design must fit on a dog collar with a relatively small width, we must consider the dimensions in our design and how complex our implementation of the system can be based on this. Having the capability to add a USB as well will be important. While the USB will not be present in our final design, the capability to test and verify that our software is functional will be very important.

3.1.2 ATSAMG55J19A-AUT Microcontroller

The ATSAMG55J19A-AUT microcontroller is a good choice for use in our design for many reasons. It is relatively small at 7.5 millimeters in length and width. It supports all three of our desired communication protocols as well as allows for a USB port to be added to the design. The USB will not be present in the final design as it will be extraneous, but during the course of this project will be invaluable as a resource. This will allow for testing from an outside computer and allow us to view inputs and outputs in real time to verify and test our designs. The number of each communication protocol is also of paramount importance in the design. The ATSAMG55J19A-AUT supports up to eight communication protocols simultaneously. This is made possible by the high number of I/O pins available on the microcontroller (64). This is more than enough to support the devices currently planned to be connected. The input power required of 1.62-3.6 volts also falls in line with our current desired 3.3 volts. As seen in *Figure 3.2*, each communication protocol is supported. This also gives a proper idea of the functions of many of the pins and how many pins will be needed in order to implement each communication protocol.

Figure 3.1
ATSAMG55J19A-AUT Microcontroller



MISOx	Master In Slave Out	I/O	–	–
MOSIx	Master Out Slave In	I/O	–	–
SPCKx	SPI Serial Clock	I/O	–	–
NPCS0x	SPI Peripheral Chip Select 0	I/O	Low	–
NPCS1x	SPI Peripheral Chip Select	Output	Low	–
		Two-Wire Interface - TWIx		
TWDx	TWIx Two-wire Serial Data	I/O	–	–
TWCKx	TWIx Two-wire Serial Clock	I/O	–	–
		Universal Synchronous Asynchronous Receiver Transmitter USARTx		
SCKx	USART Serial Clock	I/O	–	–
TXDx	USART Transmit Data	I/O	–	–
RXDx	USART Receive Data	Input	–	–
RTSx	USART Request To Send	Output	–	–
CTSx	USART Clear To Send	Input	–	–

Figure 3.2
Available Communication Protocols

3.1.3 ATMEGA328PB-ANR

The ATMEGA328PB-ANR microcontroller is a powerful microcontroller which some of the popular Arduino boards are based on. This component is very good for our application because there are a lot of resources out there at our disposal. As previously mentioned, Arduino is a very popular platform for electronics and referencing other designs could be extremely useful for implementation. Something of paramount importance to our design is the available peripherals on the device as well. This microcontroller has 2 of each of the communication protocols. This would be enough to support all our sensors and other components on our board. In addition to the peripheral support, this microcontroller has 27 programmable I/O pins as well. The operating voltage range is 1.8-5.5 which is perfect for our system operating voltage of 3.3 volts.

3.1.4 ATMega2560

This microcontroller boasts 256 KB ISP flash memory, 8 KB SRAM, and 86 GPIO lines. When it comes to communication protocols, this microcontroller comes with PWM, four USARTs, and TWI. This perfectly supports our system's requirements, as we can run I2C and SPI on the GPIO lines. In addition, this microcontroller has a JTAG interface for debugging. This microcontroller is also Arduino-based, meaning we can come up with prototype code using the Arduino Mega Rev3, which is a development board with the ATMega2560 mounted. This device operates between 4.5 and 5.5 volts.

3.1.5 Part Selected

	ATSAMG55J19	ATMega2560	ATMega328P
Operating Voltage	1.62 – 3.6V	4.5 – 5V	1.8 - 5V
Flash Storage	512 KB	256 KB	32 KB
Pin Count	64	100	32
Cost	$4.96	$13.31	$2.45

Table 3.1
Microcontroller Comparison

The part we selected was the ATMega328P microcontroller. This is the best choice for our design for a few reasons; however, it also comes with at least one drawback. The MCU is great for our design because of the number of available GPIO pins and the number of communication protocols that can be handled. In addition, as the majority of our components will be operated at 3.3 volts, this microcontroller will be perfect as it will also be operated at 3.3 volts. One of the downsides of using this MCU is that compared to the other options, it has low memory, which means that we must do our best to be efficient with our code implementation. In addition to this, the embedded programming will be fairly simple as the Arduino IDE makes the writing and testing of code much easier and more efficient for the programmers. This embedded programming environment is actually a large deciding factor in our choice as the alternative microcontroller, the ATSAMG55, uses the MPLab IDE which comes with a large learning curve.

3.2 Temperature Sensor

The temperature sensor is one of the core peripherals in our design. This section describes the criteria for selection, some considerations for us to be mindful of, and the part selection. This also includes the relevant technologies that exist in the common types of temperature sensors on the market.

3.2.1 Temperature Sensor Considerations

The temperature sensor is of particular interest because the current existing components on the market use a variety of techniques to measure temperature. In our application, oftentimes the coat of fur that exists on most household pets poses a problem. This makes it significantly more difficult to obtain an accurate skin temperature reading. Another consideration that makes the selection of a temperature sensor more difficult is the problem of thermal energy from other components potentially interfering with the reading of the sensor. The heat generated from the particularly power hungry sensors on the PCB is not an easy problem to combat. This allows for thermal isolation between the components so that there is no interference.

In order to mitigate the problem of heat transfer, there are a few avenues to explore. The first being partitioning the PCB and placing the sensor farther away on the board from the high heat generating components. Another useful technique for reducing the heat transfer between components is creating a router trace in the PCB. This allows for the ambient air surrounding the component to prevent very much thermal heat transfer. The heat transfer is reduced because the thermal conductivity of air is very low compared to the materials in most PCBs.

There are many types of temperature sensors available for us. The first are thermocouples. The principle of these is that two pairs of dissimilar metals create junctions. Each of these junctions has a different function: one junction represents a reference temperature, and the other is where the temperature will be measured. This configuration works when a temperature difference causes a voltage to be generated. Some of the advantages of this particular variant of sensor are that they have good performance at extremely high temperatures and they measure their own temperature so the temperature reading can be calculated based on the relative temperature. A couple of the weaknesses of thermocouples are that these sensors will commonly give inaccurate readings after long and continuous use and, since they are electrical conductors, they cannot be allowed to contact another source of electricity.

Thermistors are another type of temperature sensor. They use semiconductor materials with a resistivity that is sensitive to temperature. This resistance decreases as the temperature increases and the change in resistivity is predictable allowing for accurate readings. It should be noted that this application is not suited for very high temperature applications. The advantages of Thermistors is that they are inexpensive and very adaptable - having many uses for a variety of applications.

Resistance temperature detectors are sensors that use a resistor which changes its resistivity concurrently with changes in temperature. There are two main types of RTDs: thin-film and wire-wound. The thin-film variant uses a thin layer of platinum. This is then trimmed into a specific pattern which gives a specific resistance. Wire-wound, on the other hand, are coils of wire that are either packaged or wound around a non-conductive material. The wire-wound variety is usually used in higher temperature applications and is one of the oldest designs still used today in terms of resistors.

Infrared Sensors are generally used to measure surface temperatures. They work by sensing electromagnetic waves in a certain wavelength range. By converting the thermal energy that is emitted from an object in a specific wavelength range into an electrical signal, a temperature reading can be made. This electrical signal is directly proportional to the amount of infrared energy emitted by an object. This is made possible by photodetectors that convert this thermal energy into an electrical signal. These sensors are very useful as they have the capability to obtain readings from an object at a distance as well as moving objects.

3.2.2 DS18B20

The DS18B20 temperature sensor is a simple 3-pin device. This IC is made to be used for applications with multiple sensors. Communication with this sensor is done through a singular 1-wire bus. This is quite different from the traditional communication protocols that were previously discussed. This bus is bi-directional, allowing for two-way communication with the central microcontroller. The operating voltage of this device is 3.3 volts which makes it ideal for our design. The temperature readings are very accurate being within .5 degrees Celsius for -10 to 85 degrees Celsius which is well within the acceptable temperatures described in our requirements. The temperature data output by the sensor will be in units of Celsius. As our application is based on Fahrenheit units, there will have to be a conversion protocol put into place, most likely on the microcontroller side. In addition to this, it also has the capability to run in what is called Parasitic mode. In parasitic mode, the component will draw power from the data line instead of an external power supply. This however, does require additional components in order to properly use this configuration. In order to properly receive data from the 1-wire data bus of this temperature sensor, the data line and VCC pin have to be connected by a pull-up resistor valued at 4.7 kΩ. The pin connections of this sensor are shown in *Figure 3.3*.

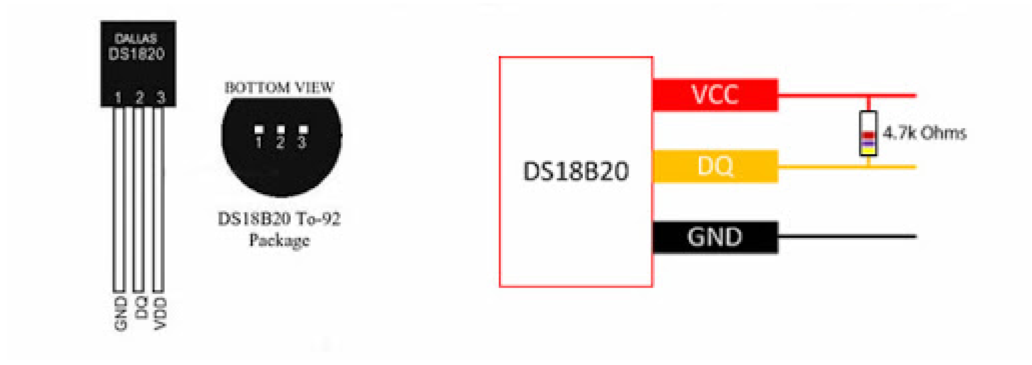


Figure 3.3
DS1820 Pins

3.2.3 TMP235A4DBZT4

The TMP235A4DBZT temperature sensor made by Texas Instruments is a simple 3 pin analog output device. This device has short circuit protection and high accuracy of .5 Celsius for a temperature range of 0 to 70 degrees Celsius. This sensor outputs a voltage proportional to the temperature that is sensed. The slope of this gain is 10mV/C which can be converted on the MCU end to give an accurate temperature reading. Before this can be done, there would need to be an Analog-to-Digital conversion circuit implemented to make the output readable by the microcontroller. Included in the datasheet is a table which gives the temperature reading for a given output voltage. This sensor has a wide input voltage range of 2.3 - 5.5 volts which is well within our constraints. This sensor also uses 1 wire to output data similar to the above sensor. This is, however, a one way directional output as it only outputs a voltage.

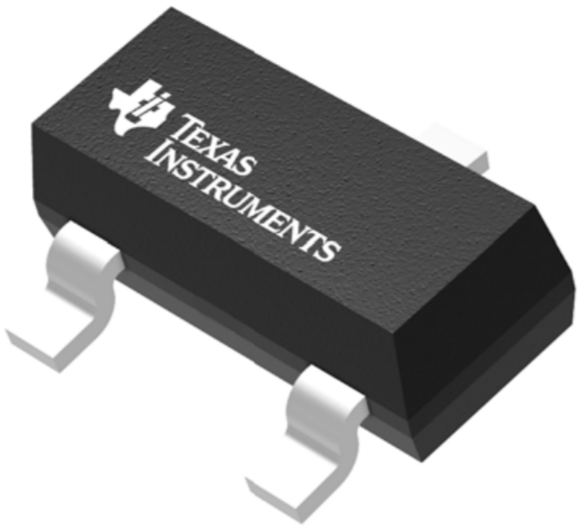


Figure 3.4
TMP235ADBZT Overview

3.2.4 MCP9808

The MCP9808 is a highly accurate temperature sensor made by Adafruit. As this product is made by Adafruit, it is also highly compatible with the Arduino Uno board that we are using to test our components. The main feature of this sensor is its accuracy, which measures to 0.25 ℃ accuracy between -40 ℃ and 125 ℃ with a guaranteed accuracy of 0.5 ℃ at all other temperatures. Another advantage is that the sensor is on a PCB board with 8 pins, so it is much easier to use and manipulate than the 3 pin sensors. In addition, the MCP9808 supports I2C communication with 8 connections for a single address, due to its three I2C address pins. Therefore, this sensor will be useful should we decide to use I2C to communicate with our microcontroller.

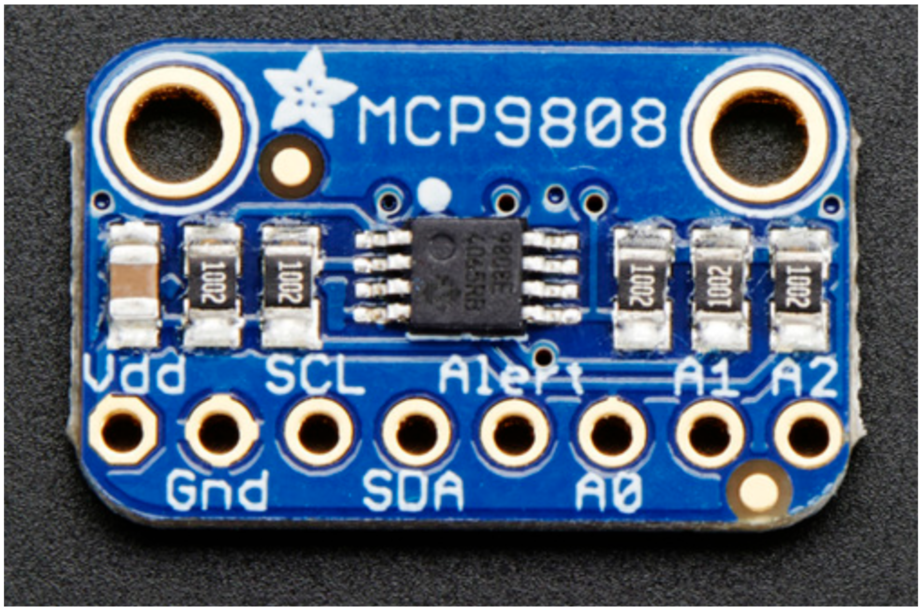


Figure 3.5
MCP9808 Overview

3.2.5 Comparison and Summary

	DS18B20	TMP235A4DBZT	MCP9808
Operating Voltage	3.3 V	2.3 - 5.5 V	2.7 - 5.5 V
Analog/Digital	Analog	Analog	Digital
Parasitic Mode?	Y	N	N?
Number of Pins	3	3	8
Data Protocols	1-pin analog	1-pin analog	I2C
Dimensions	Insignificant	Insignificant	Significant
Price	$5.91	$0.76	$4.95

Table 3.2
Temperature Sensor Attribute Comparison

We chose the DS18B20 temperature sensor mainly due to its simplistic design (1-wire communication). In addition, it operates at the exact voltage that our system is set at - 3.3 Volts - and also has the option to operate in parasitic mode. In addition to all of these features, this temperature probe also comes in a waterproof version if we choose to have this as a feature in our smart collar.

3.3 Heart Rate Monitor

This section describes our part selection of the heart rate monitor. This includes the considerations for the monitor, as well a description of relevant technologies in commonly used heart rate monitors. By taking all these things into account, the best monitor for our design can be implemented in our design.

3.3.1 Heart Rate Monitor Considerations

Similar problems arise with the heart rate monitor as with the temperature sensor. The fur that covers the skin of the animal will be our main issue to work around. In terms of the sensors to choose from, there are two main types. There are sensors which use electrical impulses to measure heart rate and optical sensors which are commonly used in applications such as smart watches.

Optical heart rate monitors use a method called photoplethysmography or PPG to measure heart rate. At a simple level, this works by shining light onto the skin of the user and measuring the scattered light from the blood flow.

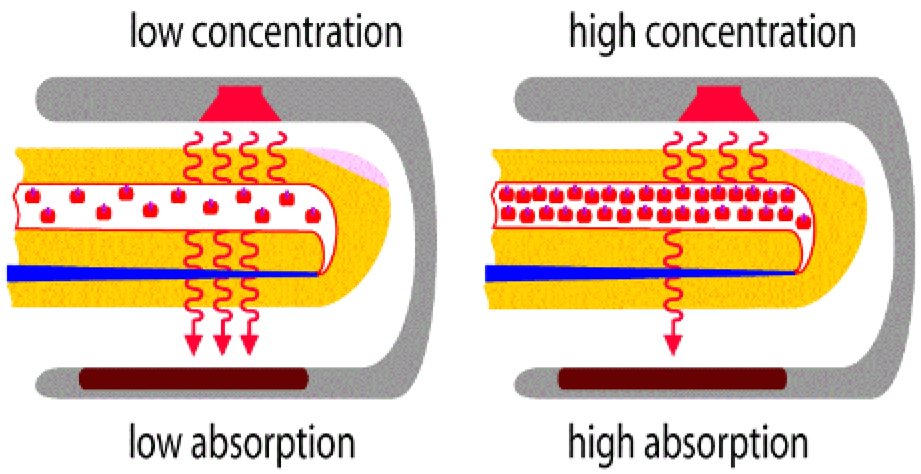


Figure 3.6
Photoplethysmography Demonstration

These applications generally use a few sensors to gather and subsequently convert the data into a more usable form. A few of the issues that arise from the optical based monitors are that there can be interference from optical noise, problems that arise from movement which cause some applications to measure the step rate instead of the heart rate, and in addition, the location of the heart rate monitor is also a consideration as not all body parts can be effectively read and give adequate data.

Electrical heart rate monitors use a technique which measures the electrical impulses from the heart to measure the users heart rate. Electrocardiography or ECG/EKG is the name of the technique that these monitors use. Generally these applications are quite complicated using sensors on multiple areas on the body to get a very accurate reading. This will likely not be feasible for our application as they require the sensors to be on multiple areas of the body to give an accurate reading.

In order to ease the computation needed to be done by the heart rate sensor, the sensor will detect heart rate for 15 seconds, and then multiply this number by 4 in order to get the beats per minute. This process will then be repeated every minute after a calculation has been done in order to provide a constant stream of information to the user.

However, the heart rate naturally increases whenever the dog is engaging in physical activity. If this is not accounted for, then the user could end up getting unwanted and incorrect notifications that the heart rate sensor is out of tolerance. Since there are multiple sensors that are going to be interacting with this system, we could use the correlation of the data from the temperature and heart rate sensors in order to judge if the dog is engaging in physical activity. For example, if the dog's heart rate and temperature spike in the same time frame, then the application would choose not to send the alert since all is normal with the dog. Another solution to this problem would be to add an accelerometer that would be communicating with the microcontroller to tell it to silence the notifications from the heart rate sensor for a set period of time when it detects movement above a certain speed. However, this method of solving this problem seems unlikely to be implemented since we would be adding an entirely new sensor. This would be disadvantageous in several ways such as: possibly putting us over budget, taking up space on the physical collar, and adding another set of code that would need to be written. Alternatively, we could simply implement a boolean inside the application itself that would allow the user to temporarily silence all sensor notifications from the collar. This boolean could be disguised as a "playtime" button for example. The user would press this button whenever they are about to begin exercise. Given the simplicity of this solution, this is the implementation that we will be striving toward.

Typically, the resting heart rate range for a dog is from 50 to 120 beats per minute. However, heart rate range also depends on the breed, age, and medical status of the animal. Due to this, the user should be able to enter parameters on the user interface in order to calculate the individualized resting heart rate. Upon first registering an animal, the user will be asked to enter these parameters:

1. Weight

< 30 Pounds	> 30 Pounds
120 - 180 BPM	60 - 140 BPM

Table 3.3
Appropriate Heart Range Based on Weight

2. Age

Dog's heart rates will not typically change over time. The only exception is when the dog is a puppy. A dog is considered a puppy when it is less than 15 months old. At the puppy stage, the dog's heart rate may go as high as 220 BPM.

3.3.2 MAX86141

This is an optical pulse oximeter and heart-rate sensor for wearable health. This is an ultra-low-power, completely integrated, optical data acquisition system. This sensor has two optical readout channels that can operate simultaneously. This device also supports a standard SPI compatible interface and fully autonomous operation(17). It also has an extremely low sample rate error being less than 2%. Its operating voltage is 1.7 - 2 volts which is not ideal for our configuration.

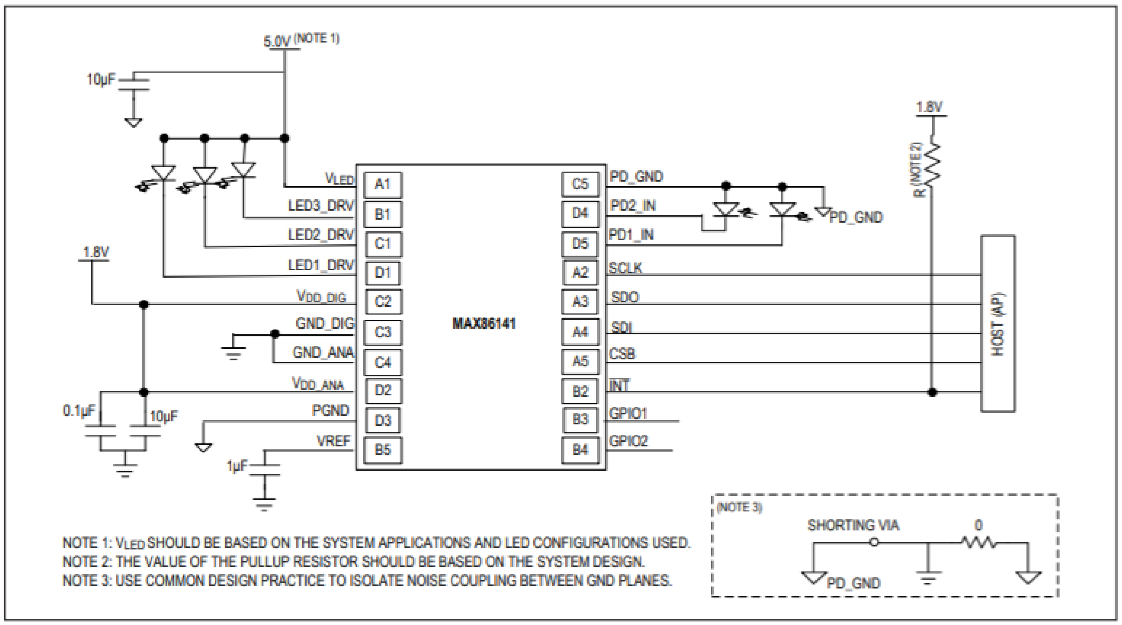


Figure 3.7
MAX86141 Schematic
Corresponding Sensor Pin Connections (copyright request shown in appendix)

3.3.3 MAX30100

Compared with the sensor from before, this sensor offers three groups of communication pins: SPI, UART, and I^{2}C. This sensor is also an integrated pulse oximeter and heart-rate sensor. It combines two LED's, a photo detector, optimized optics, and low-noise analog signal processing to detect pulse and heart-rate signals. It operates from 1.8V and 3.3V power supplies(39).

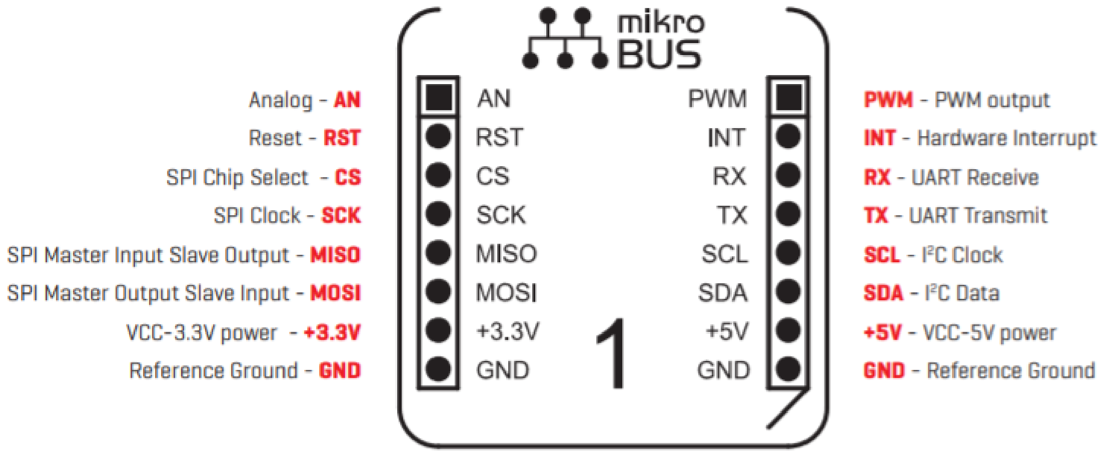


Figure 3.8
MAX30100 Sensor Pin Connections

3.3.4 MAX30102

This is a popular heart-rate sensor that is used in Samsung Galaxy devices. It provides a complete system solution(18). Communication is done through an I2C compatible interface. The module can be shut down through software through software with zero standby current, allowing the power rails to remain powered at all times.

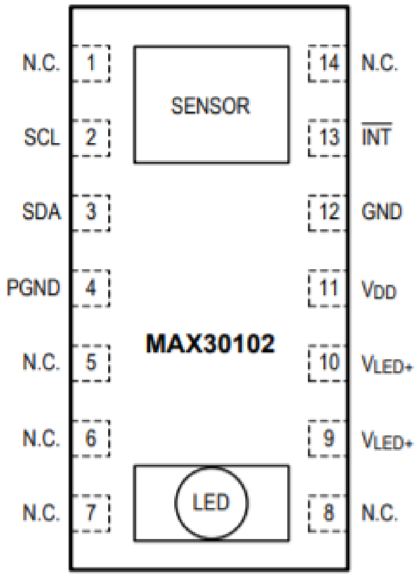


Figure 3.9
MAX30102 Sensor Pin Connections

3.3.5 Comparison and Summary

Sensor Name	MAX30100	MAX30102	MAX86141
Voltage	3.3 - 5 V	1.8 - 5 V	1.7 - 2 V
Size (L x W)	28.6 x 25.4 mm	0.5 x 8.5 mm	2.048 x 1.848 mm
Cost	$19.00	$13.99	$3.52

Table 3.4
Heart Rate/Temperature Sensor Comparison

Despite its relatively large size, we decided to choose the MAX30102 for our system. We made this decision because of its wide availability of communication protocols… it can communicate with UART, SPI, and I^2C. In terms of receiving data, it is also more innovative than the other sensors. The other sensors required a clip or a strap to be attached to the earlobe or the finger in order to start receiving accurate heart rate data. For obvious reasons these sensors were not going to work for our dog collar implementation. The MAX30102 functions by just having the sensor on the board touch the physical skin to get a reading. This works perfectly for the dog collar as we can just put the sensor on the underside of the collar flush with the rest of the leather.

Upon further inspection, our team discovered that this sensor comes with a temperature setting. So as to not overcomplicate the design, we chose to use the pulse oximeter sensor for both temperature and heart rate readings. This also allowed us to only cut out one hole in the bottom of the system casing as opposed to two to get the necessary vitality readings.

3.4 GPS

GPS is a crucial part of our Pet Pal system, as one of the core features of our product is the ability to track the user's pet location through the use of the Pet Pal mobile companion application. Following the requirement specifications that we have set in the Project Description chapter of this document, the GPS component must be small enough to be encapsulated on a PCB board, and have an accuracy of at least 6 feet. In order to test the GPS feature before placing a module on our final PCB board, a breakout board will be used with an arduino testing board in order to ensure that it works properly and fits the needs of our system.

3.4.1 GPS Module

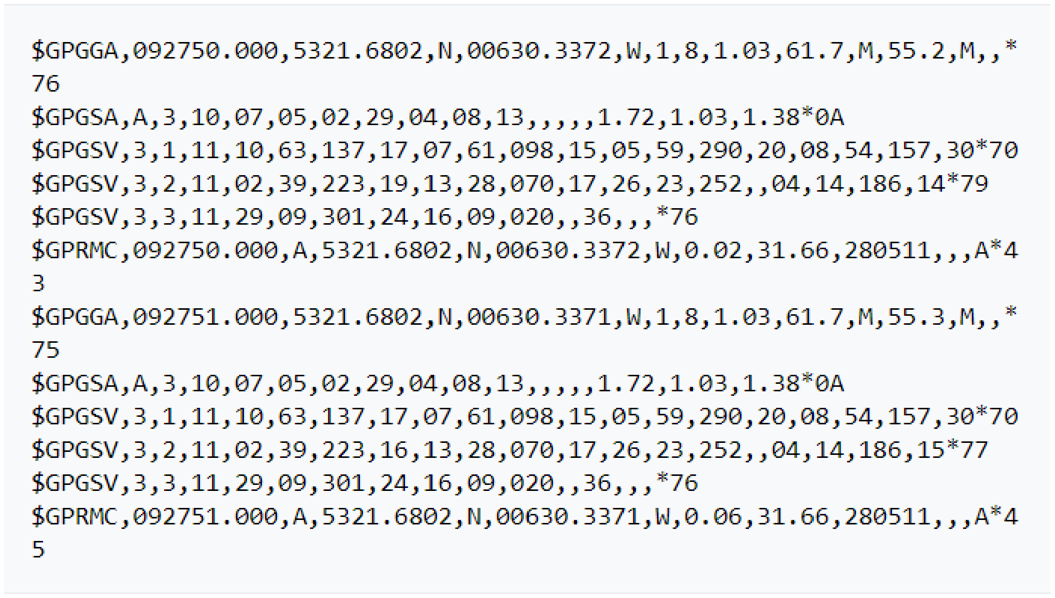
The GPS module will add location functionality to the Smart Collar. This is relevant because we want the product owner to be able to track the location of their pet accurately to six feet as specified in *Table 2.2*. The GPS module shall be attached to the microcontroller, and will interact directly with the user application. Important things to consider when selecting a GPS module are power consumption, antenna, update rate, and sizing so as to ensure that it fits comfortably within our casing.

When it comes to update rate, we may want to consider using a GPS module with over a 1 Hz update rate due to the speed at which some pets can move. If the update rate is not fast enough, the GPS module will have a hard time portraying the exact location of the pet (19).

Relating to the antenna, we can either choose to use a receiver with a pre-soldered surface mount antenna or a receiver with compatibility for an external antenna connection. The external antenna might prove useful since we want the receiver to be concealed within the collar's casing.

3.4.2 Receiving GPS Data

In order to receive GPS data, the GPS module must point directly upwards with a clear view of the sky. Things such as tall buildings, obscured weather, and RF noise can all deter the GPS module from getting a fix with a satellite. This will obviously prove to be an issue for our system, as the GPS module will be contained inside of the collar's casing. In order to minimize errors with this, we can resort to using a device with a powerful external antenna. Most of the time, data received by the GPS receiver follows the NMEA 0183 standard. This standard uses an ASCII serial communications protocol to define how data is received (20). Since this standard typically uses a sampling rate of 4800 baud, it is fairly simple for the microcontroller to handle this data using UART communication. A sample file that can be outputted by the GPS receiver is shown below:



```
$GPGGA,092750.000,5321.6802,N,00630.3372,W,1,8,1.03,61.7,M,55.2,M,,*
76
$GPGSA,A,3,10,07,05,02,29,04,08,13,,,,,1.72,1.03,1.38*0A
$GPGSV,3,1,11,10,63,137,17,07,61,098,15,05,59,290,20,08,54,157,30*70
$GPGSV,3,2,11,02,39,223,19,13,28,070,17,26,23,252,,04,14,186,14*79
$GPGSV,3,3,11,29,09,301,24,16,09,020,,36,,,*76
$GPRMC,092750.000,A,5321.6802,N,00630.3372,W,0.02,31.66,280511,,,A*4
3
$GPGGA,092751.000,5321.6802,N,00630.3371,W,1,8,1.03,61.7,M,55.3,M,,*
75
$GPGSA,A,3,10,07,05,02,29,04,08,13,,,,,1.72,1.03,1.38*0A
$GPGSV,3,1,11,10,63,137,17,07,61,098,15,05,59,290,20,08,54,157,30*70
$GPGSV,3,2,11,02,39,223,16,13,28,070,17,26,23,252,,04,14,186,15*77
$GPGSV,3,3,11,29,09,301,24,16,09,020,,36,,,*76
$GPRMC,092751.000,A,5321.6802,N,00630.3371,W,0.06,31.66,280511,,,A*4
5
```

Figure 3.10
Sample Data Retrieved by a GPS Receiver

The strings shown above are just some examples of sentences that can be interpreted from the GPS receiver. For a full list of all $GPxxx sentence codes and short descriptions of each you can visit: http://aprs.gids.nl/nmea/#allgp. Now to interpret the strings associated with each line beginning with the *$* symbol above:

1. GPRMC - Global Positioning Recommended Minimum Coordinates
    - Recommended minimum specific GPS/Transit data. It contains the time of fix, the state of the receiver, latitude, longitude, speed (knots), and the fix type.

2. GPGSV - GPS Satellites in view
    - Contains total number of messages of a certain type per cycle, message number, total number of SV (space vehicles) in view, the SV PRN (pseudorandom noise code) number, elevation in degrees, azimuth, and SNR (signal-to-noise ratio).

3. GPGSA - Global Positioning Active Satellites
    - GPS DOP (dilution of precision) and active satellites. Also tells us the type of fix (2D or 3D) for the satellites.

4. GPGGA - Global Positioning System Fix Data
    - This provides full coordinate data in addition to extra accuracy data as well as altitude. This one requires a bit more interpretation, as we will see in *Figure 3.11*.

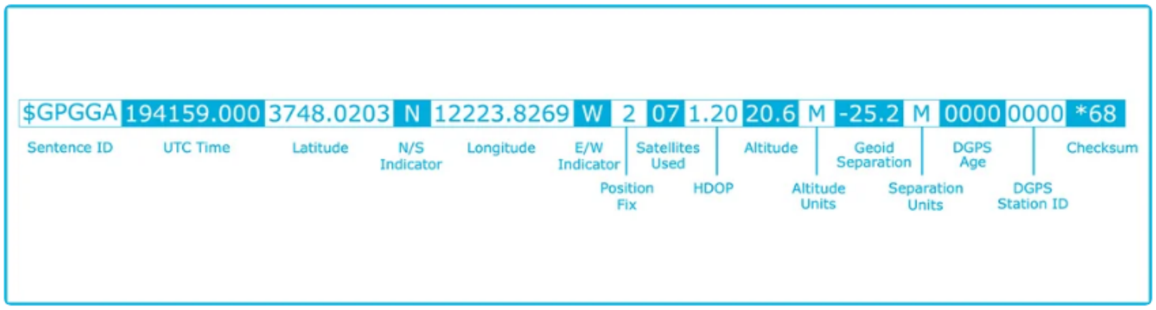


Figure 3.11
Manual Parsing of the GPGGA String

3.4.3 Extracting Data

In order to make our job easier and avoid parsing all these strings by hand or in overly-complicated code, there are several GPS libraries that will do this for us.

1. AdaGPS
    - This is a library provided by Adafruit for handling GPS data.

2. TinyGPS++
    - This is a GPS/NMEA Parser for Arduino. It provides methods for extracting position, date, time, altitude, speed, and course from GPS devices(21).

3. Location Services API for Android
    - In order to allow the application to use location data, location user drivers must be added. First, the required permission for the driver has to be added to the application's manifest file. Then, a driver has to be created. This is done by first creating an instance of GnssDriver. After, register this driver with the UserDriverManager. Lastly, unregister the driver when location events are no longer required (22).

    - In order to get a visual representation of location data, the Google Maps Android API allows for the inclusion of maps in the application. In order to set this up, you have to set up the Google Play services SDK in your application development project (23).

3.4.4 GPS Module Selection

Both of the following GPS receiver boards use the PA6H module with the MTK3339 chipset. The GPS module is the PA6H with the MTK3339 chipset. This module can track up to 22 satellites on 66 channels. The position is accurate up to 1.8 meters. The PA6H module boasts the industry's highest level of sensitivity (-165 dBm) and the lowest power consumption with a refresh rate of up to 10 Hz.

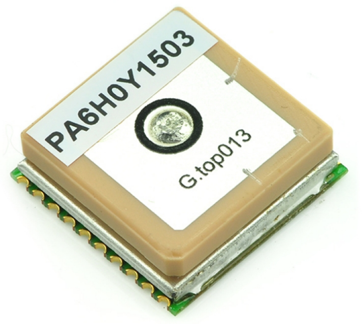


Figure 3.12
The PA6H Module with the MTK3339 Chipset

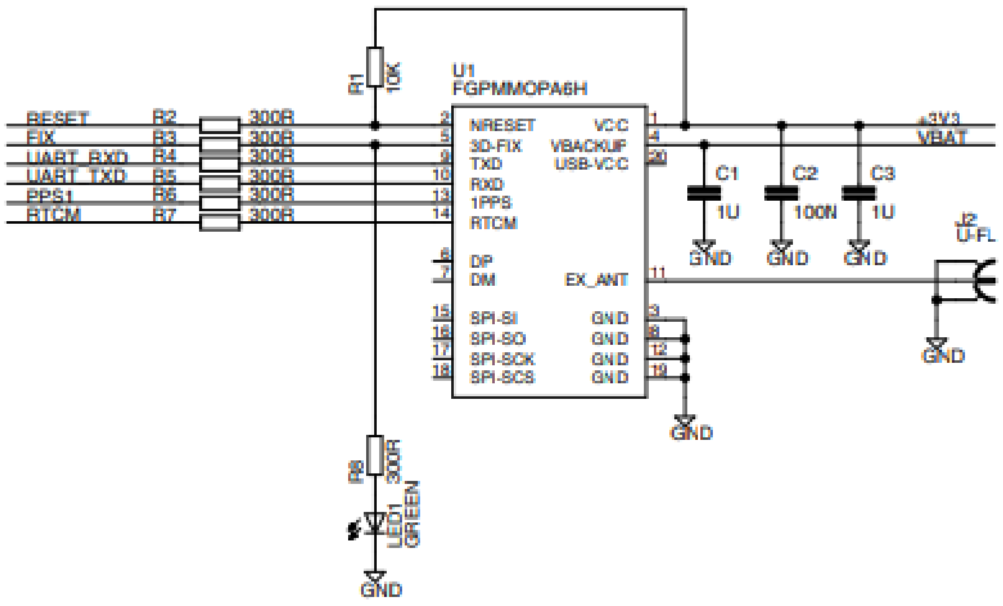


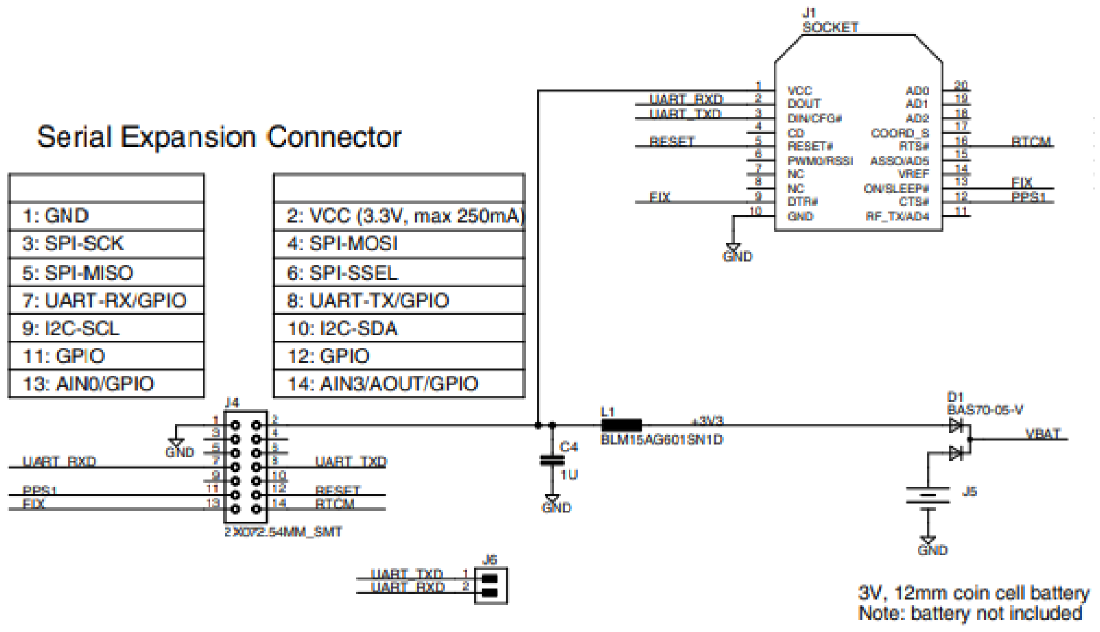
Figure 3.13
PA6H GPS Module Schematic (copyright request in the appendix)

3.5 GPS Receivers

These receivers are boards that host the PA6H GPS module and provide it power as well as maintain communication with it.

3.5.1 EA-ACC-023

This GPS receiver board hosts a high quality GPS module with a built-in antenna. The board communicates with the GPS module with the UART interface. A 3V, 12mm coin cell battery is required to power this receiver board. This last requirement may prove to be less than ideal, as a dead battery on the receiver board would result in the complete loss of GPS communication.



Serial Expansion Connector

1: GND	2: VCC (3.3V, max 250mA)
3: SPI-SCK	4: SPI-MOSI
5: SPI-MISO	6: SPI-SSEL
7: UART-RX/GPIO	8: UART-TX/GPIO
9: I2C-SCL	10: I2C-SDA
11: GPIO	12: GPIO
13: AIN0/GPIO	14: AIN3/AOUT/GPIO

Figure 3.14
EA-ACC-023 Schematic

3.5.2 Adafruit Ultimate GPS Featherwing

This receiver board comes with a standard ceramic patch antenna but it has external antenna functionality. You are able to attach any 3V active GPS antenna using the uFL connector. The module will automatically detect the antenna and switch over (24). This receiver does have a slot for a coin battery, however it is not required! This is simply if you want a backup power supply to the sensor. However, this comes at a price: this receiver board is quite a bit larger than the EA-ACC-023 described above.

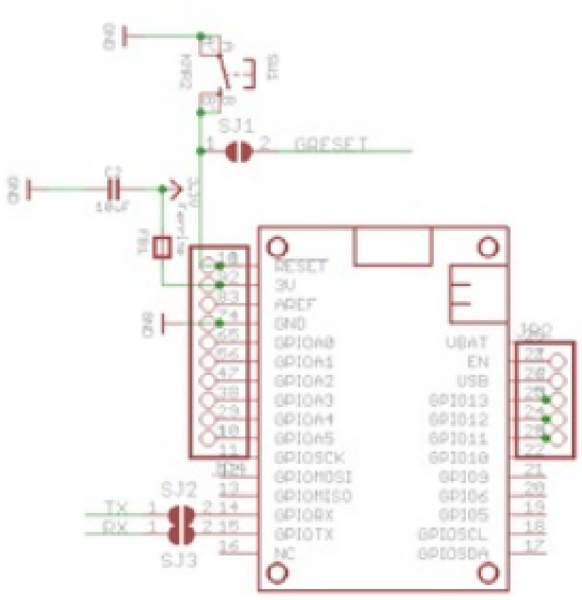


Figure 3.15
Featherwing Schematic

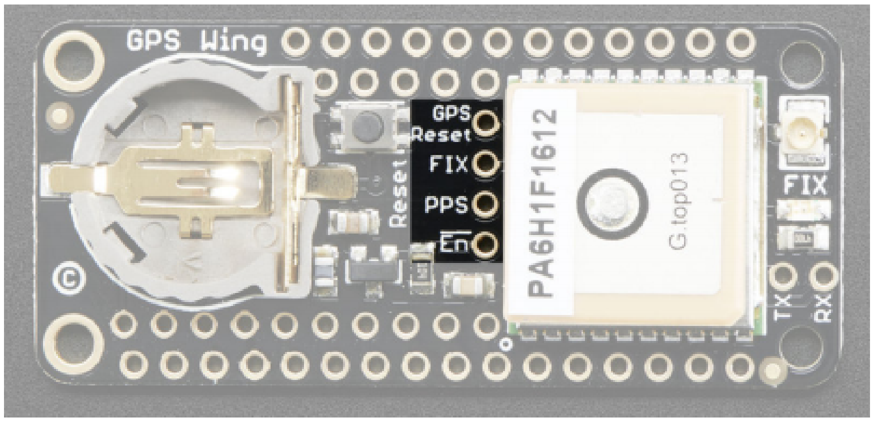


Figure 3.16
GPS Breakout Pins

Now we will go more in depth on extra pins that we can use on the GPS module:

- <u>GPS Reset</u> - This is used to manually reset the GPS module. There is a jumper included on the back if you wish to connect the GPS reset to the microcontroller reset line. This jumper is shown in *Figure 3.17*.

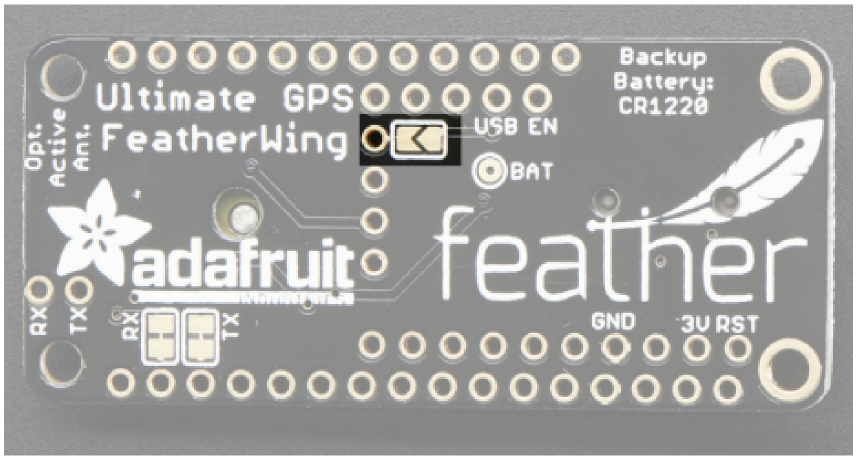


Figure 3.17
Jumper for Connecting to Microcontroller Reset Line

- FIX - This is an output pin. It functions to drive the red FIX LED. The pin will pulse between high (3.3 V) and low (0 V) once every second when there is no GPS fix. When a fix is acquired, the pin will remain low, with a 200 millisecond high pulse occurring once every 15 seconds.
- PPS - A “pulse per second” output. Most of the time it is at ground and then it pulses high once a second. The pulse lasts for 50-100ms so it should be easy for a microcontroller to sync up to it.
- EN - This is a “power disable” control line that can be used to completely cut power to the GPS module. This proves to be especially helpful if you need to run at ultra-low-power modes. You must pull high to disable the GPS since by default this is pulled low.

3.5.3 Flora Wearable GPS

This GPS receiver is part of the Adafruit wearables electronics collection. It is compatible with the FLORA microcontroller board mentioned above in the microcontroller section. To connect this, use either conductive threading or solder wires between the two. The connections should go as follows: attach the 3.3V pads together, connect RX to TX and TX to RX and then ground to ground(25). This receiver communicates via the UART communication protocol. A figure depicting the connection with conductive threads is shown below.

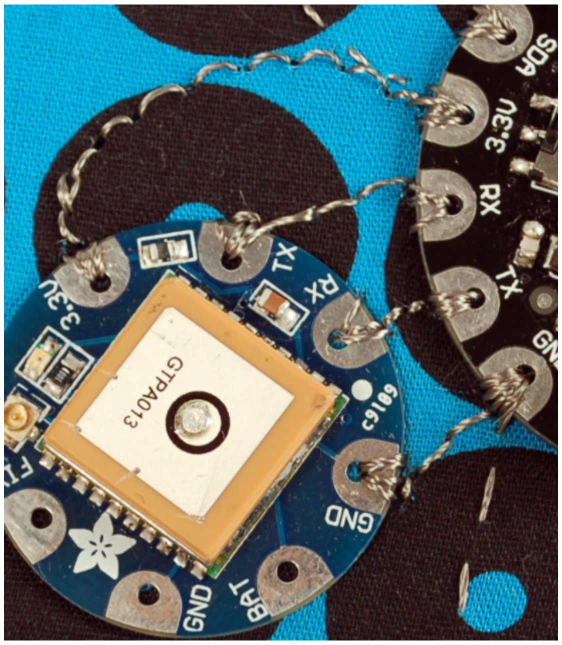


Figure 3.18
Sample FLORA Connection

This IC also comes with an antenna connector to allow for an external antenna to be attached to boost the sensitivity. In addition it allows for a coin cell battery to be connected to the BAT and GND pads on the IC. The addition of a coin cell battery allows the GPS module to get quicker satellite fixes, as well as helping it remember which satellites it has connected to. Ultimately, this leads to faster boot-up times, which will lead to fast response times on the app end. Recall that response time on the app is crucial, as one of our demonstrables is that the app gets responses from its peripherals in 30 seconds or less. The CR2032 coin cell battery that we used in our design is shown in the following section: *Battery Part Selection*.

3.5.4 Datalogging

The GPS data that is received (in the form of NMEA sentences) is handy when using tools such as Google Maps or Google Earth which can take in this raw data and convert it to useful location services. In the context of our application, the smart collar will send this GPS data to the corresponding phone application. This application will then be able to view the location of the collar in real-time.

In order to store location data you can choose out of two options. The first option includes wiring up an SD card holder to the board's SPI bus. This first option proves to be a bit challenging in our case considering the size constraint of our collar - we can't afford to add any more components that aren't entirely necessary. That leaves us with the second option: storing data in the board's internal file system.

3.5.5 External Antenna

Since we might attempt to contain all of the smart collar system components in a casing around the collar, we might need to add more sensitivity to the GPS module in order to receive a fix on the satellites and also to receive error-free location data. This can be done by adding an external antenna to the GPS module. Most external antennas work off of SMA connection, so in order to add this to a board that works off of uFL connection, a uFL-SMA adapter is necessary.

3.5.6 Comparison and Summary

Sensor Name	EA-ACC-023	Featherwing	Flora GPS
Sensitivity	-165 dBm	-165 dBm	-165 dBm
Voltage	3V - 4.3V	3V - 5.5V	3V - 4.3V
Antenna	On-Board, Patch	Patch, or uFL connection	Patch, or uFL connection
Size (L x W)	16 x 4.7 mm	50.8 x 22.86 mm	30.5 diameter x 5.98 mm thick
Cost	$38.75	$39.50	$40

Table 3.5
GPS Receiver Comparison

Despite having the highest cost, we decided to go with the Flora wearable GPS receiver. This is due to its relatively small size and its ability to be sewed into the collar itself and other peripherals using conductive threading. The PA6H module that it supports on the breakout has a uFL connector that allows an external antenna to be connected to add higher sensitivity. This receiver board also boasts built-in data logging capability.

3.6 Battery Part Selection

Since our design will be battery powered and our design will have small dimensions, the battery that is chosen for our design must be small. As defined in our requirements, the battery must output a minimum of 9 volts and a maximum of 15 V. In addition to this, the battery must store a sufficient amount of power in order to power the design for a reasonable amount of time. Our plan is to have our microcontroller and components run in low-power mode for as long as possible to prolong the life of the battery in question.

The battery will not be mounted onto our designed PCB itself, instead it will be off board and connected to the PCB via power circuitry that will be added in order to ensure that the proper amount of voltage that is needed by our system is supplied at all times. There is also a possibility of adding the capability to power the device through a USB connection, but this idea has not been explored and is considered extraneous and unnecessary.

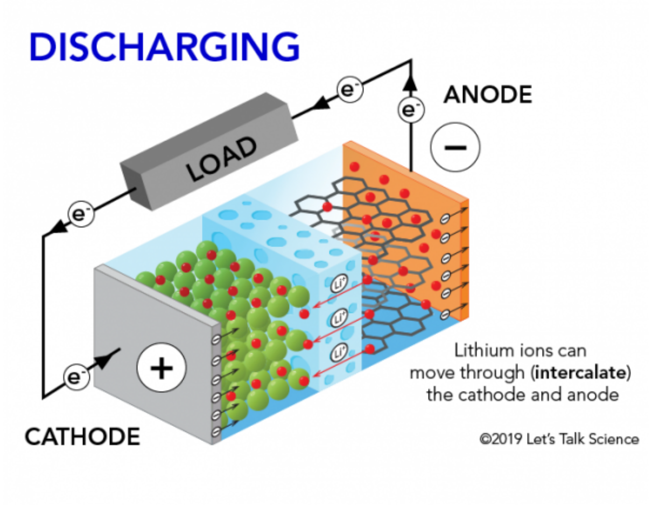

DISCHARGING
LOAD
ANODE
CATHODE
Lithium ions can move through (intercalate) the cathode and anode
©2019 Let's Talk Science

Figure 3.19
Lithium-Ion Diagram

Lithium-Ion batteries are one of many types of batteries on the market. They come in many forms but the most useful for our design will be the coin style. Coin batteries are small and compact making them ideal for implementing in our design, since space will be of paramount importance when considering which components to include. Like the name suggests, the battery operates off of lithium-ions. The anode and cathode store the lithium. The electrolytes in the battery carry the positively charged lithium-ions from the anode to the cathode and the free electrons in the anode generate a charge at the positive collector(10). Lithium-Ion batteries generally have some of the highest energy densities in the range of 100-265 Wh/Kg(11). For all the positive attributes of these batteries they do have a tendency to overheat, which is an important consideration. In addition to this, they also age fairly poorly. However, this does not affect our design. These batteries often are meant to be recharged and that is the reason for the aforementioned drawback. In our design, however, the batteries are intended to be replaced when they run low on power. This adds another consideration to our design. We have to make access to the battery simple enough so that the average user can go in and replace the collar at their own convenience.

3.6.1 Coin Cell Holder

Since we are holding the battery for this system separately, we need to have a way to hold it and connect with the main system. For coin cell batteries, adafruit has developed a 20 mm coin-cell holder with a built-in on/off switch. This battery holder is specific to the non rechargeable CR2032.

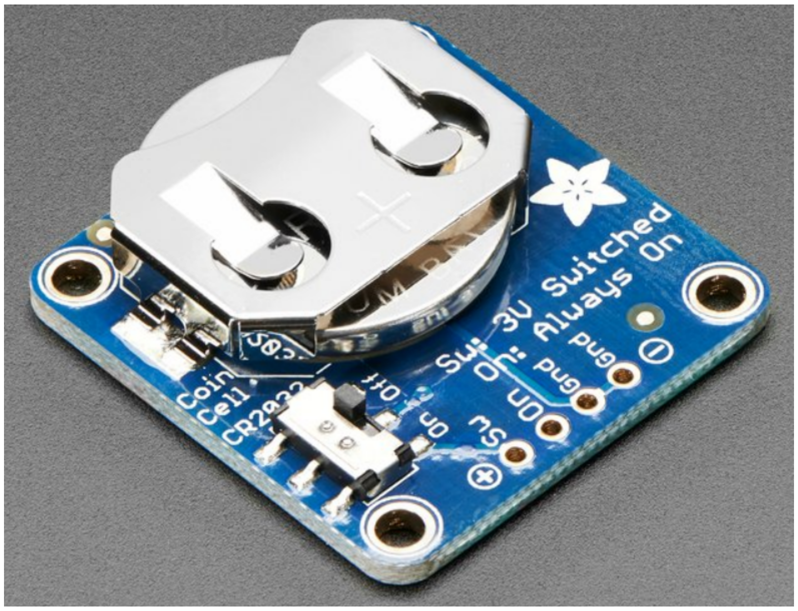


Figure 3.20
Coin Cell Breakout with CR2032 Placed Inside

As you can see in the picture above, this board includes two ground pins, one “switched” power pin, and one pin that is always on. The “switched mode” is a handy mode to have if we decide to implement an external on and off switch that will allow the user to power off the collar. This board also has holes for sewing into the collar that we will be using.

3.6.2 CR2032

This is a coin-cell 3 V lithium battery. This battery is immensely popular in the industry and can be purchased at nearly any retail store. It’s inexpensive price (about $2.00) also makes it a valid competitor. The size is small, about the size of a quarter (hence the name) which is preferable for our design since we want all of our components to be encased in a relatively light-weight container on the collar. One thing to note is that this battery is non-rechargeable.

3.6.3 AFPG804

This is another 3 V coin-cell lithium battery. This battery differs from the previous one listed in terms of termination style. This battery comes with a wire lead connector, which would lead to needing an extra component (port) in our final integration. The size is smaller than the CR2032 but by an amount that is not enough to yield any benefits (a difference of 0.5 mm). It also has a capacity that is less than the CR2032. For a full list of comparisons refer to the *Battery Attribute Comparison* table below.

3.6.4 TL-5930/S

The TL-5930/S is a D Lithium Thionyl Chloride non-rechargeable Battery. This battery is not a coin-cell like the previous entries in this section. Instead, it is a size D, which is quite large considering the proportions of the rest of our design. In addition, this battery would need a battery pack with wire leads in order to supply power to our designed PCB. On the plus side, this battery has a capacity of 19 Ah which is beneficial to our design because it would not be ideal for the user to constantly be opening the collar to change the battery. Though non-rechargeable, this battery's lifetime makes up for it. It supplies a voltage of 3.6 V which would fit our Another thing to consider is the availability of this battery; if the user wanted to replace the battery themselves, it would prove quite difficult to find a replacement since they are not easily found in most retail stores. This battery also proves to be quite expensive, with the price of one battery coming in at $22.24.

3.6.5 LITH-18

The LITH-18 is a non-rechargeable, Lithium Manganese Dioxide battery that can be used to power our circuitry. It provides 6 V of DC voltage, which should be enough to power the Pet Pal electronics. One advantage to this battery is its capacity at 1.3 Ah. When compared to smaller batteries, the LITH-18 has a higher capacity, which increases the lifetime of the battery and in turn saves users money from constantly buying replacements. In addition, the price is very cheap at $7 when compared to other alternatives that are upwards of $20. However, one disadvantage to this battery is that it requires a battery holder, which will increase the components that we will need to include in our PCB design. Also, this battery is considerably bigger in dimension than alternatives, which restricts space for other components in our PCB.

3.6.6 MIKROE-2759

This battery differs from the previous entries because this one is a rechargeable lithium-polymer battery. It has a rated voltage of 3.7 V, which is enough to power all of the sensors that will be on-board. No holder is required for this battery, instead, it has a wire leads termination style. This is beneficial for the development team because this means that we do not have to install an external battery holder on the dog collar. This would add a lot of unnecessary weight to the system and would make the overall dimensions of the collar unsatisfactory.

3.6.7 Energizer L522

The Energizer L522 is a lithium battery with a weight of 33.9 grams which can provide an input voltage of 9 volts. By using this type of battery, we do not need to use multiple batteries at the same time, only one L522 is enough to supply power for the whole system. On the other hand, the audio amplifier requires an input voltage from 7 volts to 12 volts. Thus, the audio amplifier can get an input voltage source directly from this battery without using any voltage regulator

3.6.8 Comparison and Summary

Battery Name	CR2032	AFPG804	TL-5930/S	LITH-18	MIKRO E-2759	Energizer L522
Capacity	225 mAh	165 mAh	19 Ah	1.3 Ah	190 mAh	
Voltage	3 V	3 V	3.6 V	6 V	3.7 V	9 V
Size (L x W x H)	20 x 3.2 mm	20 x 2.5 mm	32.9 x 61.5 mm	35.6 x 20.3 x 33 mm	24.5 x 23.5 x 4.2 mm	26.50 x 17.50 x 49
Termination Style	Requires holder	Wire leads with connector	Button Top (Extending)	Requires holder	Wire leads with connecto r	Snap connector
Cost	$1.97	$23.00	$22.24	$6.99	$7.00	$8.50

Table 3.6
Battery Attribute Comparison

Out of the options that we had for our power source; we chose the LITH-18. This was for a variety of reasons - besides its low price and rampant availability, this lithium battery supplies 6 V to the system which is satisfactory to power all of our circuitry. All of the sensors that we were using were tested on a 3.3 V system perfectly, but the audio circuitry requires at least 4 V hence the decision we made. Though non-rechargeable, its large capacity means a sufficient battery life. The battery will have to be placed in a holder connected to the circuit, which will add weight to our system, but in this case it is necessary. An installed battery holder will also make replacement easy.

3.6.9 Battery Connection Considerations

Alternatively, we could opt to use multiple smaller voltage traditional cylindrical shaped batteries such as the TL-5930/S. However, we would need to find a way to combine multiple batteries in order to produce the correct power that we wish to supply to the system. This approach can be done in two different ways each with their tradeoffs. In order to determine if the battery holder being used is set up for a series or parallel connection, an analysis of the holder itself is necessary. If you see a wire connecting the positive and negative terminals, then the battery is in series connection(53). It is also important to note that for safety reasons, you should never set up a battery by connecting the leads so that a short circuit is created. The batteries must be attached to a load!

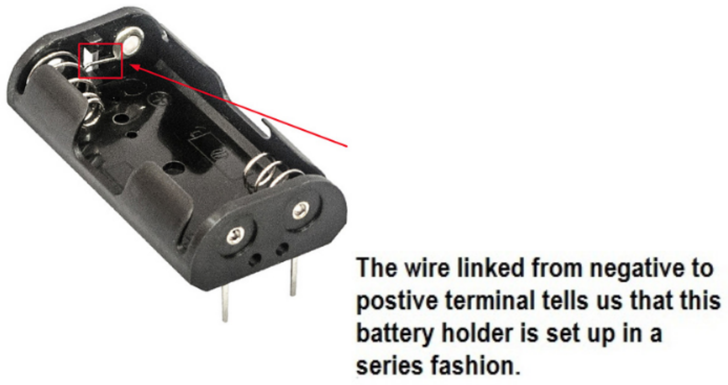

The wire linked from negative to postive terminal tells us that this battery holder is set up in a series fashion.

Figure 3.21
A Series Connected Battery Holder

3.6.10 Series Connection

Series connection battery holders are the most common type. When connecting batteries in series, the positive terminal of one battery should be connected to the negative terminal of another battery and so on. When batteries are connected in series the flow of electrons is the same anywhere in the circuit, this can be represented by the equation: $I_{total} = I_1 = I_2$ where I_1 and I_2 are the currents of battery 1 and battery 2, respectively. However, connecting the batteries in this manner will lead to an increase in the total voltage. This voltage increases every time that a battery is added in series. This can be represented by the equation: $V_{total} = V_1 + V_2$ and so on(52).

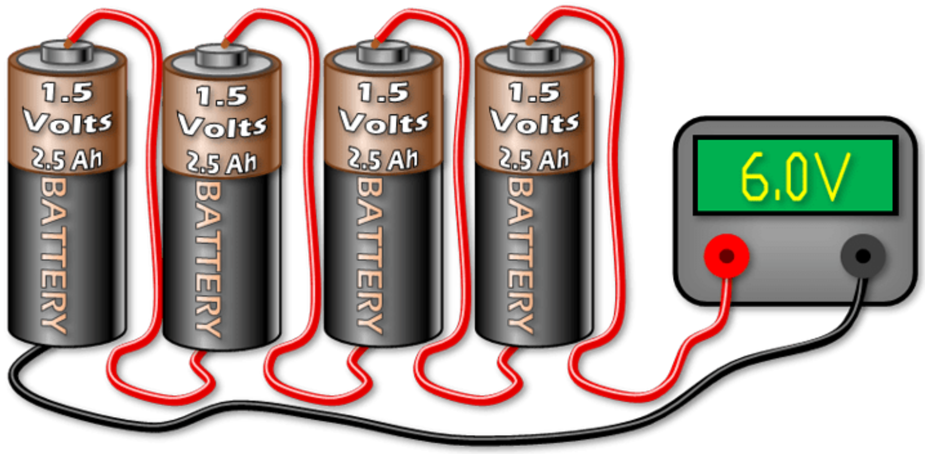


Figure 3.22
Batteries Connected in Series

3.6.11 Parallel Connection

When it comes to connecting batteries in parallel, you can accomplish this by joining all of the positive terminals together with a single wire to one part of the circuit and then connecting all of the negative terminals together in the same fashion. When batteries are connected in this manner, the voltage found in the circuit is the same as the voltage found in an individual battery. This can be represented by the equation: $V_{total} = V_1 = V_2$ where V_1 and V_2 are the voltages of battery 1 and battery 2, respectively. This is because individual electrons can only pass through one of the paths and therefore one battery at a time. However, parallel connections will provide you with an increase in current flow that follows the formula: $I_{total} = I_1 + I_2$. It is worth noting that it is advised to not connect batteries in this way. The voltages between batteries are never quite the same even if they have the saem rated value between them. The batteries' low internal resistance causes large current flow from one battery to another, and if this cycle continues, it could damage the batteries or the circuit and cause them to die prematurely.

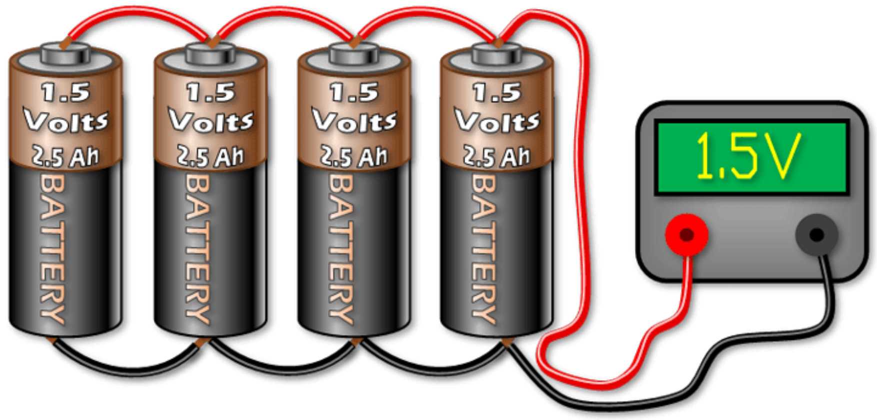


Figure 3.23
Batteries Connected in Parallel

3.7 Dog Collar Selection

The dog collar will allow us to attach all of our electronic components onto a wearable item for intended animals. From our requirement specification for the collar in *Table 2.2*, our goal is for the collar to fit a neck size of 16 - 26" which is equivalent to a medium or large sized dog. In addition, we need our collar to be strong enough to hold the electronics that we will attach to it, while making sure that the dog is comfortable and that the collar is stylish for added market value. Information on specific collars can be found in the following sections.

3.7.1 Tuff Pupper Lifetime Classic Collar

The Tuff Pupper Lifetime Classic Collar is a strong, versatile pet collar that is made from ballistic nylon. The collar comes in three sizes: small with an adjustable length of 11 - 15", medium with a length of 15 - 19", and large with a length of 19 - 23". The medium and large sizes fall within our projected collar length of 16 - 26" as shown in *Table 2.2.* This collar is advantageous due to its durability and customizability. The collar comes in 6 different colors, with the red variant shown below in *Figure 3.24*. However, one downside of this choice is the price, which is $15-16 for our target sizes. In addition, the

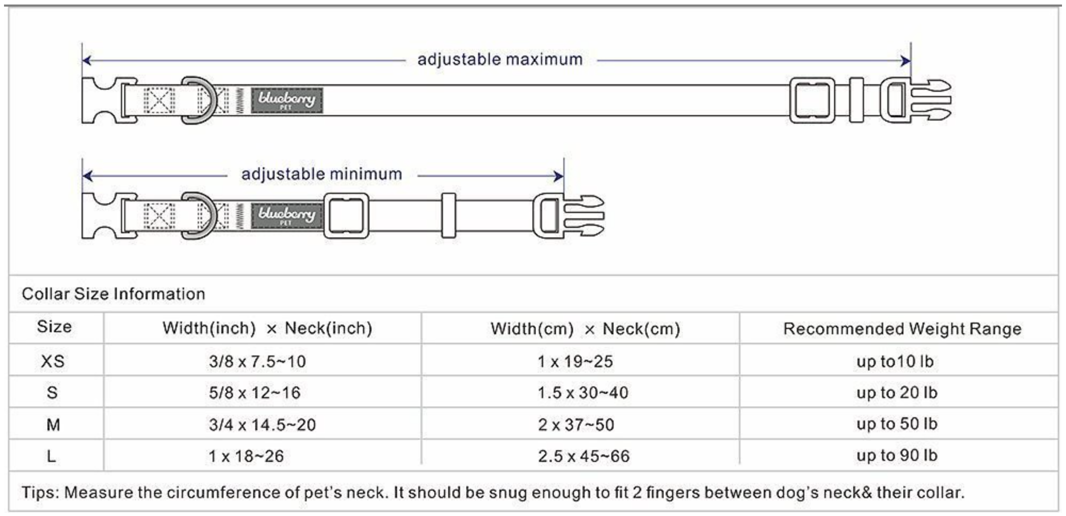
Tuff Pupper collars have a smaller adjustable range than other competing collars. The specifications for each collar size can be found below in *Table 3.7*.

Figure 3.24
Tuff Pupper Lifetime Classic Dog Collar

3.7.2 Blueberry Pet Classic Solid Nylon Dog Collar

The Blueberry Pet Classic Solid Nylon Dog Collar is a popular nylon dog collar that is strong and customizable. As shown in *Figure 3.25*, the collar comes in four sizes: XS, S, M, and L. Specifically, the medium (14.5 - 20") and large (18 - 26) variations fit our engineering requirement from *Table 2.1*, which states that the collar must be within a range of 16 - 26". When compared to the Tuff Pupper collar, the Blueberry collar has a larger range of adjustability, which allows the collar to be more robust for varying sizes of pets. In addition, this collar has more customizability than the Tuff Pupper collars, with four different sizes and 8 bright colors. However, this collar does have some downsides. For example, since the collar is made out of nylon, it is not as durable or strong as the Tuff Pupper collar, which is made with ballistic nylon. In addition, the price is not very different from other target collars, with the price of $13 being only 3 dollars cheaper than the large Tuff Pupper collar. The collar's specifications for each size can be found in *Figure 3.26*.

Figure 3.25
Blueberry Pet Classic Dog Collar


adjustable maximum
adjustable minimum
blueberry PET

Collar Size Information

Size	Width(inch) × Neck(inch)	Width(cm) × Neck(cm)	Recommended Weight Range
XS	3/8 x 7.5~10	1 x 19~25	up to10 lb
S	5/8 x 12~16	1.5 x 30~40	up to 20 lb
M	3/4 x 14.5~20	2 x 37~50	up to 50 lb
L	1 x 18~26	2.5 x 45~66	up to 90 lb

Tips: Measure the circumference of pet's neck. It should be snug enough to fit 2 fingers between dog's neck& their collar.

Figure 3.26
Blueberry Size and Length Specifications

3.7.3 Black Rhino - Hybrid Collar

The Hybrid Collar is a heavy duty dog collar made by Black Rhino. The collar, shown in *Figure 3.27*, comes in two sizes: medium (12” - 18”) and large (15” - 23”). According to our collar length specification in *Table 2.2*, the large size is the optimal size as it has the closest range to the desired adjustable collar range. However, the medium size will be useful for smaller animals such as smaller dogs, cats, or birds in later iterations of the Pet Pal. Other features for the Hybrid Collar include a soft neoprene padding on the inside of the collar and four different color variations: red, pink, blue, and aqua. In addition, the outside is lined with nylon and features a hybrid plastic and metal clasp, which is more durable than a standard plastic clamp. While this collar may not be as strong as competitors that use materials such as ballistic nylon, its use of neoprene makes our product comfortable for the pet to wear without compromising the durability that the nylon layer provides.

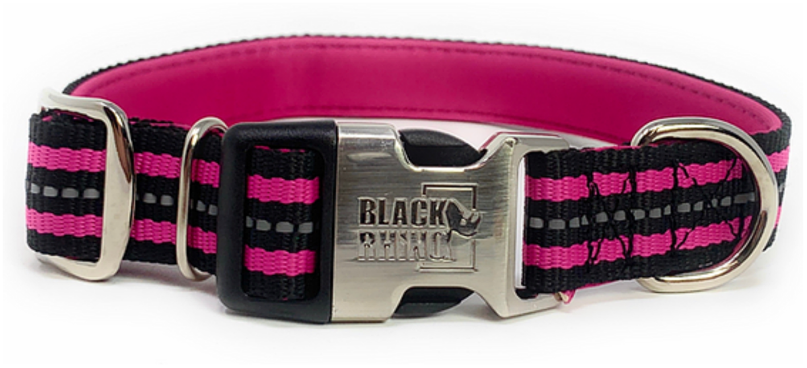


Figure 3.27
Black Rhino Hybrid Collar

3.7.4 Selection of Collar for Final Design

In the selection of the collar, our team considered many different specifications to compare our options. Each collar has benefits and downfalls, In the selection of the collar, its strength is the most important specification to our design. The Pet Pal will have no functionality if the collar does not have durability, since electronic components will have no way to reflect the dog's health statistics. All three options are made of strong materials, but the Tuff Pupper Lifetime Classic Collar is made of the strongest material with ballistic nylon. However, this collar has the shortest range at four inches, while the other two collars have a range of eight inches. The range is another important factor, as a longer range allows us to offer our product to a larger variety of pets. After all specifications were considered for each collar, our choice was narrowed down to the Tuff Pupper or Blueberry Pet collars. We ruled out the Black Rhino collar due its price, lack of color variety, and lack of consistent material. When comparing our final two options, we decided that the range of the collar is more important than its durability, since we would rather offer our collar to more pets. Therefore, we chose the Blueberry Pet Classic Solid Nylon Dog Collar. In addition to a larger length range, its range matches closely to our target length, it has a cheaper price, and it offers the largest variety of colors.

	Tuff Pupper	Blueberry	Black Rhino
Length	19 - 23"	18 - 26"	15 - 23"
Material	Ballistic Nylon	Nylon	Nylon/Neoprene
Colors	6	8	4
Price	$15.96	$12.99	$24.95

Table 3.7
Engineering, Market, and Economic Specifications for Collar Selection

3.8 Bluetooth Module

The Bluetooth module will allow us to send data between our microcontroller and our mobile Android application. This is helpful because we want to be able to display data from the GPS, heart monitor, and temperature sensor, which will be sent using the Bluetooth module. The data from each of these devices will be sent to the microcontroller, where it will then be sent over Bluetooth and decoded by the mobile application. In addition, according to the requirements specification in *Table 2.2*, we want to find a Bluetooth module with over 50 ft of range so that the user will have a useful range of functionality for the device. It is also important to consider the Bluetooth version to ensure that data can be sent to the mobile application, the voltage and power to satisfy circuit requirements, and size to ensure that the collar is not too bulky.

3.8.1 Bluetooth Communication

The Bluetooth module will be the API for the mobile application to communicate with the microcontroller. The module will be connected to the MCU such that data can be sent using the UART or SPI protocols. This communication will be first defined by Arduino code to test our Bluetooth communication with an Arduino microcontroller. We will then use that code as a guideline to program the communication between the MCU and the Bluetooth module in the final product design(26). In addition to interfacing with the MCU, our Bluetooth module will need to communicate with our mobile application. To do this, we will use the Bluetooth API that is compatible with Java in Android Studio. The application will request heartbeat, temperature, and GPS data from the Bluetooth module, which will then get the requested data from the MCU and send it to the application. In addition, the application will send sound data over Bluetooth to play sounds on the collar's speaker. This process, along with example code, is explained in the section on the mobile application.

The Bluetooth connection will be formed with the collar on the Bluetooth page of the Pet Pal companion mobile application. A mockup of the formatting of this page can be found in the Software Implementation section of the Design chapter. The connection will be formed by entering the Bluetooth module's Bluetooth MAC address into the application. A Bluetooth MAC address is a 48-bit value that uniquely identifies a Bluetooth device. In the Bluetooth specification, it is referred to as BD_ADDR(51). Bluetooth addressing can be divided into two main types which will be discussed in detail in the following section.

3.8.2 Bluetooth Addressing

Bluetooth addressing can be divided into two main groups: Public and Random addresses. However, these two groups are divided into subgroups, and a Bluetooth device can sometimes use both Public and Random address types. Both of these types are 48 bits in length.

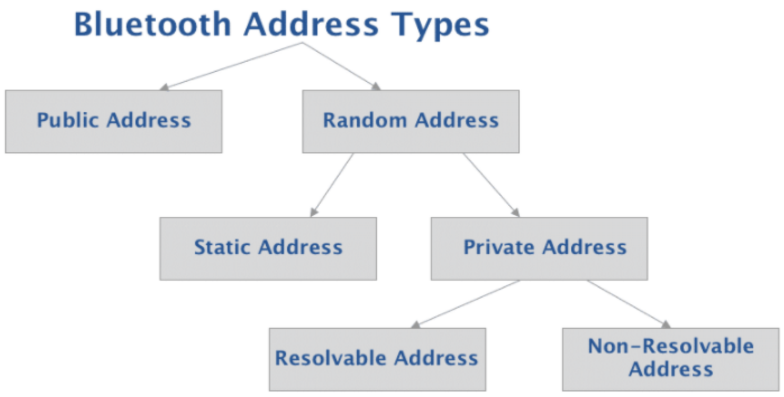


Figure 3.28
Bluetooth Address Types

1) Public Address

A Public Bluetooth address is a global fixed address that is registered with the IEEE and given a EUI-48. An EUI-48 is simply a 48-bit Organizationally Unique Identifier. This address never changes and is guaranteed to be unique per Bluetooth device. In this type of addressing, the most significant 24 bits are known as the "Company ID" field and they correspond to the publicly assigned portion of the address by the IEE. The least significant 24 bits are known as the "Company Assigned" field and they correspond to the internally assigned ID.

2) Random Address

In comparison to public addresses, random addresses do not need to be registered with the IEEE. Due to this, they are the more popular option out of the two. This type of identifier is either programmed into the device or generated at runtime. Another benefit of this addressing type is that it can be allocated and fixed throughout the device's lifespan. It can also be altered at bootup but not during runtime(50).

3.8.3 Privacy

Setting up privacy and protection measures for Bluetooth devices are important as to ensure that the Bluetooth device's MAC address isn't being tracked by untrusted parties. For this reason, Random Private Addresses are implemented to hide the identity of the devices and avert device tracking. Resolvable Random Private Addresses are used to safeguard from Bluetooth tracking. This is accomplished by the sharing of an Identity Resolving Key (IRK) with a trusted pair device.

3.8.6 HC-06 Bluetooth Module

The HC-06 Bluetooth Module is one of the most popular Bluetooth modules. The module is also highly compatible with Arduino, so this module would be useful if we choose to use an Arduino microcontroller for initial testing. In addition, the module can send data back and forth to a microcontroller using UART communication. This is helpful because other modules in our project require UART communication. In addition, this module does not use a lot of power, so it can be battery powered if needed. However, the bluetooth version is outdated (2004), so it is not scalable to devices that use newer versions. Below is a table with the electronic specifications and a schematic overview of the module connected to a microcontroller.

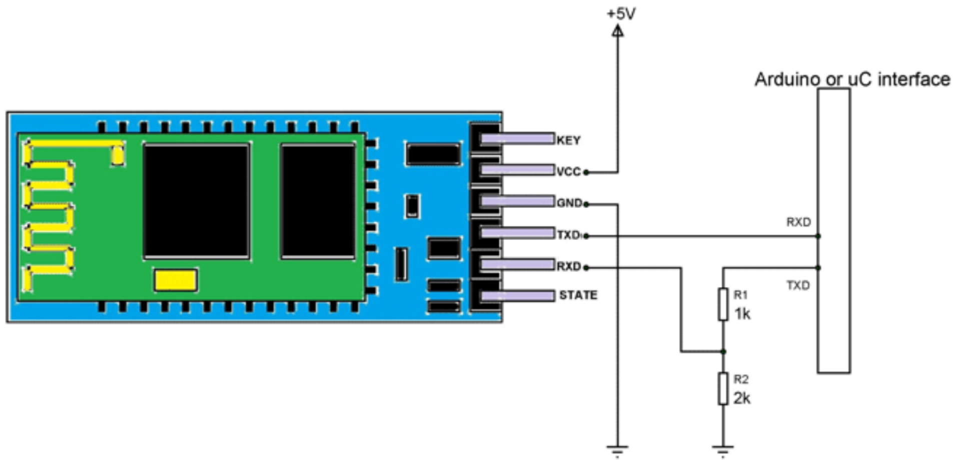


Figure 3.29
HC-06 Schematic Overview with Microcontroller Interface

3.8.5 RioRand Bluetooth Module

The RioRand Bluetooth module is another popular Bluetooth module. Like the HC-06, this module supports UART data communication. However, data can also be transferred using the SPI protocol. In addition, this module is more scalable than the HC-06, because it uses the Bluetooth V4.0 protocol (2010). However, it does have a few downsides. The RioRand module costs much more than the HC-06 module and has a shorter range. Below is a table of the module's electronic specifications, as well as an overview of the physical chip.

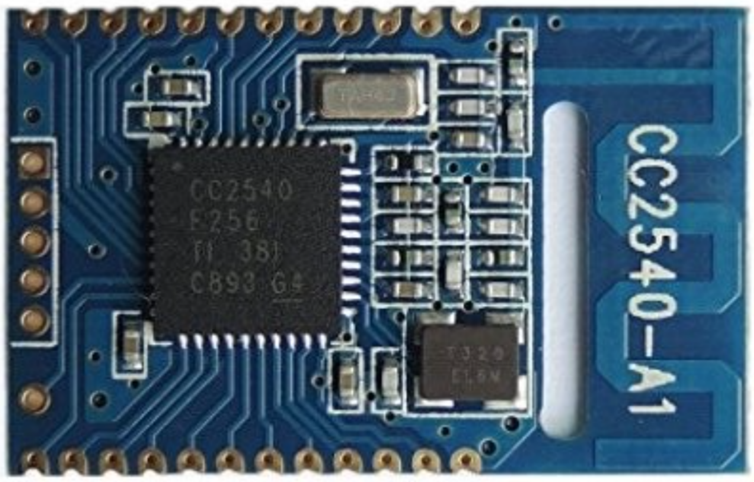


Figure 3.30
RioRand Bluetooth Chip Overview

3.8.6 Flora Wearable Bluefruit LE Module

This bluetooth module is a part of adafruit's wearable technologies set of electronics. It can be sewn into the dog collar and connected to the flora microcontroller through conductive threading. Alternatively, they can be connected in the traditional manner or soldering wires. Communication is done through the UART service. The main advantage of choosing this bluetooth module over the other options listed above is that this module is the most compact in size and can be paired with other peripherals of the same adafruit flora family.

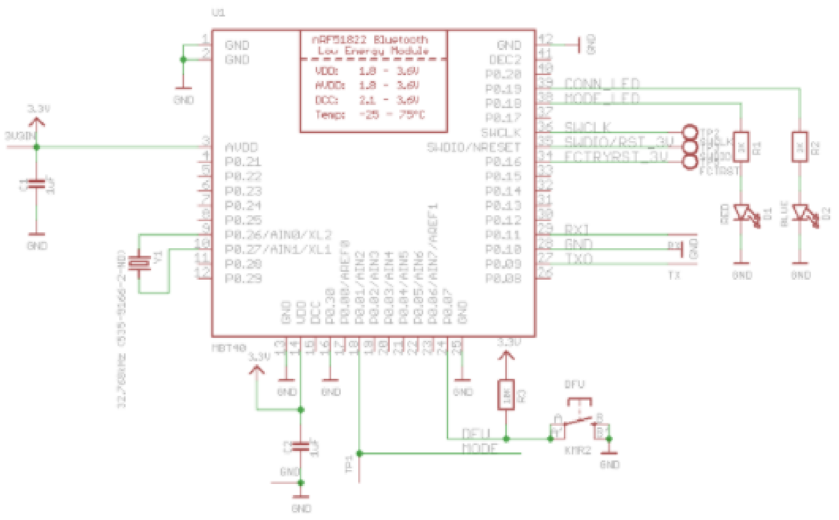


Figure 3.31
Bluefruit LE Module Schematic

3.8.7 Comparison and Summary

Module Name	HC-06	RioRand	Bluefruit LE
Bluetooth Version	V2.0	V4.0	V4.0
Voltage	3.3V - 6V	2.0V - 3.6V	1.8V - 3.6V
Range	100 meters	70 meters	80 meters
Size (L x W)	27.9 x 15.2 mm	24 x 14 mm	18 x 10 mm
Cost	$1.40	$14.57	$17.50

Table 3.8
Flora Bluetooth Module Attributes Comparison

We chose this module mainly because it was the only module listed that had the ability to communicate with the SPI communication protocol. This is important to us when we consider the big picture of how all of the peripherals in this system are going to interact. Since the microcontroller can support multiple peripherals communicating with SPI it makes SPI the efficient choice in this case. The RioRand was also taken into consideration as the documentation suggested that it would be easy to pick up and use. However, due to no availability and long lead times, we ultimately decided against using this module as it would put too much of a bottleneck on our scheduling.

3.9 Photoreceptors Cell Selection

Using solar energy is one of the special features that we want to include in our product. There are a few specifications that we need to know before choosing the suitable solar panel for our project:

1. Maximum power: The maximum output power that the solar panel can produce
2. Open-circuit Voltage: The voltage that the solar panel can produce without any loads. We want to use solar energy to supply at least 5 volts for our product
3. Short-circuit Current: The current flows from the positive side to negative side of the panel without any loads. This parameter tells us how fast a panel can charge the battery.
4. Operating Temperature: The temperature range that the solar panel can operate at. In USA, the temperature can go up to 56.7˚C in summer
5. Dimension: The size of the solar panel. It is very important that the device is small enough to be attached to the collar.
6. mQuantity needed: The number of solar panels needed for our project. If the device is small enough, we can connect multiple panels in series to increase the output voltage
7. Price: Price of the solar panel.

3.9.1 KXOB25-05X3F-TR

This is one of the solar panels that we would like to use for our product because of its appropriate size. 05X3F-TR has a dimension of 23 millimeters long, 8 millimeters wide, and 1.8 millimeters high, which will perfectly fit on a pet collar. Although we expect to build a solar system that can supply at least 5 volts for our product, the open-circuit voltage of this panel is only 2.07 volts, and the maximum power is 30.7 milliwatts. In order to achieve our specification, we can use multiple 05X3F-TR to achieve our specification. Its price is $2.22 per panel and its operating temperature is from -40˚C to 90˚C.

3.9.2 KXOB25-01X8F-TB

This is another product of ANYSOLAR Ltd which has an ideal size of (22 mm)L x (7 mm)W x (1.8 mm)H. This solar panel is not only smaller than the 05X3F-TR but it also has a higher open-circuit voltage which is 5.53 volts. Since 5.53 volts is more than the desired voltage for solar cells in the specification table, only one 01X8F is enough for our project. One weakness of this solar cell is that its maximum power is 24.5 mW. Thus, 01X8F-TB might take more time to charge compared to the 05X3D-TR.

3.9.3 FIT0333

Not as same as other solar panels, the solar cell flex panel is literally bendable. In addition, it is waterproof, thus, it is convenient and much easier for us to install the FIT0333 the way we want. Even though its open-circuit voltage is only 2 volts, its maximum power and short-circuit current are 0.5 watts and 420 mA, respectively. With a length of 165.1 millimeters and a width of 38 millimeters, FIT0333 is not as small as other miniature solar panels, so only one of the FIT0333 will fit the collar. Plus, high price is one of the reasons we consider not to choose FIT0333.

3.9.4 Comparison and Summary

	KXOB25-05X3F-TR	KXOB25-01X8F-TB	FIT0333
Maximum Power	30.7 mW	24.5 mW	0.5 W
Open-circuit Voltage	2.07 V	5.53 V	2 V
Short-circuit Current	19.5 mA	5.9 mA	420 mA
Operating Temperature	-40˚C – 90˚C	-40˚C – 90˚C	-40˚C – 80˚C

Dimension	Length = 23 mm
	Width = 8 mm
	Height = 1.8 mm
		Length = 22 mm
		Width = 7 mm
		Height = 1.8 mm
			Length = 165.1 mm
			Width = 38 mm
			Height = 0.5 mm

Quantity needed	2	1	1
Price	$2.22	$2.49	$40.98

Table 3.9
Solar Cells Selection Comparison

3.9.5 Solar Cell Selection

With regards to the solar cells, this is a difficult choice. The price on all of the above components is very reasonable and does not put any constraints on our budget. The main issue concerning our design with respect to the solar cells is the size. In addition, the voltage that is output will be important. The output voltage of the solar cell will need to be regulated and monitored in real time as well. This will mean that extra power circuitry will need to be added to the design. Given all of this, the KXOB25-05X3F-TR seems like the best choice. The maximum power for the size and voltage output will be a good fit for our design. In addition to this, the size dimensions are small enough to fit the constraints that exist in our solution. The cost is very low as well, so we could feasibly add multiple of these devices to the collar given proper placement and implementation on our end. This would allow for the battery to be charged regardless of the position of the sun as there would be multiple cells placed in various locations on the collar.

3.10 Speaker Selection

As mentioned at the beginning, a speaker will be used to play relaxing music or output the owner's voice in order to calm the pets down. Since the speaker is attached to the collar, one of the most important features must be an appropriate dimension.

3.10.1 Miniature speaker Visaton K15S

Miniature speaker K15S is a product of Visaton GmbH & Co. KG which has a diameter of 15 millimeters and a height of 4 millimeters. On the other hand, the maximum power that the speaker can take is 0.5 Watt, and its impedance is 8 Ohms. Low power consumption is convenient for our product because the less power consumed, the smaller size of the battery we need, and the final product will be lightweight. A K15S speaker also has a common frequency range which is from 800 Hz to 20 kHz, especially this is a waterproof miniature speaker, which is definitely needful in case of rain or swimming. However, 80 millimeters cable and wire leads termination are weaknesses of this speaker since we expect all of our parts to be soldered.

3.10.2 Miniature speaker Visaton K20

This is another potential product of Visaton GmbH & Co. KG with a diameter of 20 millimeters and a height of 3.6 millimeters, which is a good dimension because it is insignificantly larger than the K15S speaker. Besides being waterproof, the K20 speaker is also dust tight. With these two features, our smart collar becomes even more practical and user-friendly. As same as the K15S, a K20 speaker has an impedance of 8 Ohms, but its max power is 1.5 Watts, which is three times higher than the K15S. Nevertheless, it has a considered high efficiency of 73 dB, and it can properly operate under rough temperatures ranging from -40˚C to 85˚C. One of the main reasons why we consider using the K20 speaker is because it has the solder pad, which will be easier for us to place the speaker at a suitable location. Although we expect to pay $5 for a speaker, we can adjust our budget to buy the K20 speaker for the price of $6.36 per unit if it is our best choice for the speaker

3.10.3 SP micro speaker SP-1511L-2

Unordinary shape is one of the strengths of this product. While we were looking for a suitable speaker, we realized most of the miniature speakers on the market have round shapes. Of course, a circular shape will create a good environment for the air movements when the speaker operates. Nonetheless, the round structure combining the thin edge is fragile compared to the rectangular shape. The SP-1511L-2 is a rectangular speaker which is 15 millimeters long and 11 millimeters wide, its height is 4.2 millimeters. The maximum power it can take is 1 Watt and it has a large range of frequency 100 Hz to 20 kHz, which is moderately better than the K15S speaker. In addition, the SP-1511L-2 has a high efficiency of 91 dB with a pretty low price of $1.9. There is no information about waterproof or dustproof of this device, but we can easily test its durability if we decide to use this part in our project. On the other hand, we can use the SP-1511L to test with our system before installing a higher price speaker.

3.10.4 PS1240P02BT Buzzer

Completely different to the other speaker, this is a product of TDK corporation that is not an usual speaker but it is a buzzer which used to play the beeping sound, or tones sounds. The main purpose of this device is to alert the users not playing music. However, we have done some research and found out a possible way to play music with a buzzer, for that reason, we put this device into our list of considerations. With a dimension of 12.2 millimeters and a height of 6.5 millimeters, it has a smaller size compared to the other speaker that we mentioned before. Its efficiency is about 60dBs, which is not as high as other speakers probably because it is a buzzer. The PS1240P02BT buzzer has a pretty low price of 61 cents per unit. Beside using it as a speaker, we can also add this to our project and use it as an alert system. Below is a recommended configuration from the PS1240P02BT's datasheet. In addition, we should use the 1kΩ resistor in order to get the best efficiency.

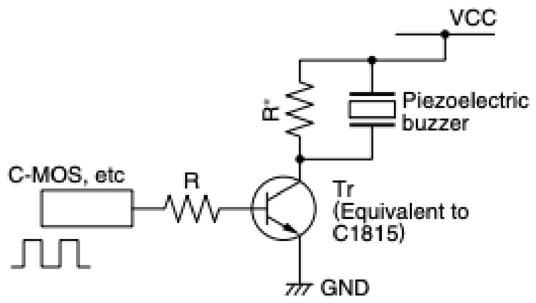


Figure 3.32
Operating Circuit for the PS1240P02BT Buzzer

3.10.5 Comparison and Summary

In general, we have in total four speakers to consider. Each type of speaker has their own pros and cons, for that reason, we have to choose our speaker based on what we prioritize. Size of the device is the higher-priority that we need to achieve, next will be the frequency range, and the power that the device will consume. Looking at the table below, we can see that the PS1240P02BT buzzer has an ideal size, but it has low efficiency as well as short frequency range, so we will exclude PS1240P02BT from our speaker list. The Visaton K15S speaker has a diameter 3 millimeters larger than the PS1240P02BT buzzer. Nonetheless, it has higher efficiency and wider frequency range. The Visaton K15S also consumes less power than other speakers, and it is waterproof. Although Visaton K20 and SP-1511L-2 speakers have higher efficiency and wider frequency range compared to the Visaton 15S speaker, their sizes are more bulky and consume more power. Thus, our final decision for the speaker is the Visaton K15S.

	Visaton K15S	**Visaton K20**	**SP -1511L-2**	**PS1240P02BT**
Maximum Power	0.5 W	1.5 W	1 W	-
Impedance	8Ω	8Ω	8Ω	-
Frequency Range	800 Hz - 20 kHz	600 Hz - 20 kHz	100 Hz - 20 kHz	2 kHz - 10 kHz
Efficiency	70 dB	73 dB	91 dB	60dB
Terminations	Wire Leads	Solder Pads	Wire Leads	Through hole
Shape	Round	Round	Rectangular	Cylinder
Dimension (Diameter x Height)	15 mm x 4 mm	20 mm x 3.6 mm	15 mm x 4.2 mm	12 mm x 6.5 mm
Protection	Waterproof	Waterproof Dust tight	-	-
Price	$4.84	$6.36	$1.9	$0.61

Table 3.10
Speaker Selection Comparison

3.10 Testing Boards

Once our team made a decision on which parts we will use for the different modules in the Pet Pal, we decided that we needed to test the modules using a microcontroller board before we start designing and ordering our PCB. This will ensure that all modules work as we intend. In addition, we ordered multiple different parts for some modules, so we will use the testing board to determine the best fit for these modules. When testing the modules, the microcontroller board also allows us to write preliminary embedded code, which can be used as a basis for our final embedded code design.

3.11.1 Arduino Uno Rev3

The Arduino is a very popular platform for designing and creating integrated circuits. As such there are many available resources at our disposal to reference. This board will mostly be used for testing our component and initially configuring our design. There are many advantages to this board as well. First, it is very inexpensive at about $23 for the full kit, not putting any constraints on our budget. In addition, it also comes with an on board USB port. This will be very beneficial for testing purposes and being able to monitor outputs and inputs in real time as they are sent and received. The board is 68.6 mm by 53.4 mm making it just large enough for us to test our components on. This board has an operating voltage range of 7 - 12 volts which, being much higher than intended for our battery powered design, will not be an issue due to the board being primarily a testing platform.

The microcontroller on the board is the powerful ATMEGA48A. This MCU supports all three of our chosen communication protocols as well as a configurable low-power mode. The number of I/O pins is not of concern since the board itself has 14 digital I/O pins as well as 6 analog I/O pins for our testing. The on board flash memory of the ATMEGA48A is 32KB which will be more than enough to implement the code, receiving and converting the code to a usable form.

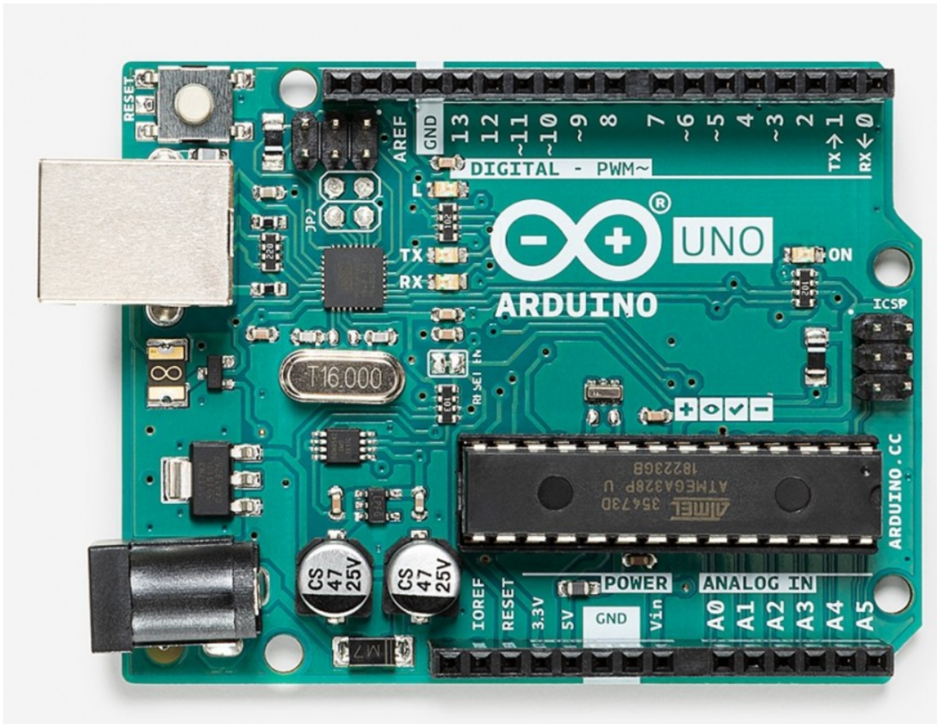


Figure 3.33
Arduino Uno Overview

3.11.2 Texas Instruments MSP-EXP430G2ET

Texas Instruments (TI) is one of the most popular manufacturers of microcontroller development boards for designing integrated circuits. Therefore, one option for a development board is the MSP-EXP430G2ET, which is one of TI's most popular boards. One of our team's main goals is to stick to our budget and underspend if possible. Thankfully, this board would not cause any strain on our budget as all parts can be purchased for a total of only $10. These parts include the Launchpad development kit, MSP430G225IN20 microcontroller, and a micro USB cable for programming. On the launchpad, there is a micro USB port to program the microcontroller. This is advantageous as it allows us to program the microcontroller using Code Composer Studio (CCS), which is an IDE made by TI that is highly compatible with the MSP430 microcontroller. In addition, our team has experience using CCS with MSP430 microcontrollers due to previous projects and curriculum.

As stated before, the microcontroller for this board is the MSP430G225IN20. Some desirable characteristics of this MCU include support for SPI, UART, and I2C as well as 5 different low-power modes. This means that the MCU will be able to communicate with all of our components, and we will be able to configure the MCU to extend the battery life of our system should we use it in our final PCB design. Also, the MSP430G2 has 24 GPIO pins, which will be plenty of pins to connect the sensors that we need to test. In addition, this MCU has an operating voltage of 1.8 V to 3.6 V, which is in the expected voltage range for our battery powered design. For storage, the MSP430G2 has

16 kB of non-volatile memory, which should be enough space to store all of our embedded code.

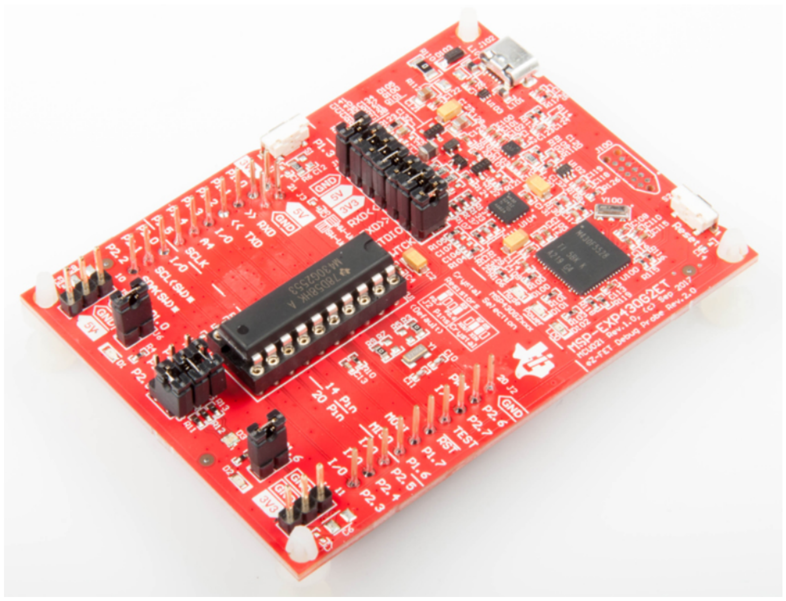


Figure 3.34
MSP-EXP430G2ET Overview

3.11.3 Comparison and Summary

Although the Texas Instruments MSP-EXP430G2ET development board has many advantages over the Arduino Uno Rev3 board, we ultimately decided to use the Uno due to its compatibility with the other modules. The MSP430 board leads the Uno in almost every category in *Table 3.11*, but is not as compatible with the Adafruit parts that we decided to use in our final PCB design. In addition, we decided that the Arduino IDE is much better for our testing environment than Code Composer Studio, which would be required if we went with the TI development board. Ultimately, the specifications for each board are somewhat similar, so we decided that compatibility and ease of IDE use were more important for our testing environment. Many of the components we have selected are arduino friendly and make the testing process much easier when paired with the Arduino board. While in the final product there will be some changes with regards to embedded programming and connectivity, this board will give us the ability to find a clear direction for our project and allow us to get a better understanding of the processes involved with embedded programming.

	Arduino Uno Rev3	TI MSP-EXP430G2ET
UART Connections	1	1
SPI Connections	1	2
I2C Connections	1	1
GPIO Pins	14	24
ADC Channels	6	8
Operating Voltage Range	7 - 12 V	1.8 - 3.6 V
Price	$23	$10

Table 3.11
Testing Development Board Comparison

4. Design Constraints and Standards

Throughout the process of development for the Pet Pal, there are many standards and constraints that we must consider to produce a successful product. The constraints will define which economic, time, manufacturability, sustainability, social, environmental, etc. factors restrict our team from coming up with a more innovative solution to our project's problem. Meanwhile, standards are predefined solutions to many different problems that are used by a wide majority of the technical community. This will save us from extraneous work in each component for which we can use a standard. In this chapter, we will lay out all of the standards and constraints that we believe are related to our Pet Pal design. By defining these parameters, we are able to better understand the characteristics of our design so that we can estimate our development timeline.

4.1 Standards

As our design has many aspects to it, there are many standards associated with the technology that we have implemented. Standards, generally, are documents which describe specifications and procedures to design and implement certain technologies or other products for common usage. These have a multitude of purposes ranging from safety and easing the design and implementation process. These exist in pretty much all walks of life and everyone uses many in their day-to-day life. Standards we will be using will be gathered from the IEEE Standards Association and other sources. These will not only allow us to work more efficiently within the given standards, but they also allow us to gain knowledge and insight into some of these processes that we may be less familiar with. While there are a wealth of organizations that provide standards, IEEE will be the best one for our use because of its base in electronics. There are standards for GPS, Bluetooth, communication protocols, as well as standards associated with the sensors in our designs.

Bluetooth generally follows the NMEA 0183 standard. Devices that follow this standard are designated as either *Talkers* or *Listeners*, with some devices having the capability to be both of these. The data that is sent by the talkers in the form of "sentences" which are indicated by the presence of the character "$" at the start of the sentence. There are also multiple types of sentences including proprietary and query sentences. This standard also describes many different identifiers which indicate the type of device which is broadcasting.

Standards for app development also exist in addition to other technical standards. IEEE 830 describes recommended practices for software development. It is based on a model which assists in accurately describing what software customers wish to obtain, assisting the software developers understand what the customer wishes to obtain, and to help accomplish the following goals: "develop a standard software requirements specification (SRS) outline for their own organizations, define the format and content of their specific software requirements specifications, and develop additional local supporting items such as an SRS quality checklist, or an SRS writer’s handbook”(54). The developed SRS will have many additional benefits including: “establishing the basis for agreement between the customers and the suppliers on what the software product is to do, reduce the development effort, provide a basis for estimating costs and schedules, provide a baseline for validation and verification, and serve as a basis for enhancement.”(54) These answer fundamental questions in the software development lifecycle and give clear guidelines on how to proceed with development. To successfully follow these guidelines, the standard dictates some of the considerations for the production of a good SRS.

1. Nature
    - An SRS, at its core, should address a few issues: Functionality, External Interfaces, Performance, and Design Constraints.
        - Functionality
            - In our implementation, we have a clear vision of the features desired by our potential “customers”.
            - This includes interfacing with the Microcontroller and successfully synthesizing the information we receive from our sensors.
        - External Interfaces
            - This software directly interfaces with mobile devices as a mobile application.
        - Performance
            - The goal of our application is to be able to successfully receive and use the data obtained from the microcontroller.
        - Design Constraints
            - This will be described in a later section.
2. Environment
    - Since the software is only half of the larger system we are developing, it is important that the developed SRS does not go too far out of scope of the application. This means clearly defining our software requirements, not describing any design aspects of the project - only implementation, and not imposing any additional constraints on the project.
3. Characteristics
    - The SRS shall be correct, unambiguous, complete, and consistent so as to allow for the most efficient development.
4. Joint Preparation
    - The SRS should be jointly developed with the customer and developer agreeing on what the software should do. As the customer does not generally have knowledge on the technical aspects, and the developer often does not fully understand the problem the consumer has, the SRS being developed jointly has the best result.
5. SRS Evolution
    - Whenever changes need to be made, it is important that there is a clearly defined process for changes being made.
        - Making sure that there is an accurate and complete trail of changes.

  - Permit both the customer and developer to review the changes.
6. Prototyping
   - This is a useful process for both customer feedback on initial iterations as well as being a time for any issues to be sorted out by the developer.
7. Embedding Design in the SRS
   - While a specific design may be requested by the customer, it is important to note that this poses constraints on the implementation.
   - Design should not be the focus, instead, should be on the services performed by the desired technology.

By following these standards, we have a clear path on how to successfully and efficiently develop our application.

4.1.1 C Programming Standard

In our implementation, embedded coding of the microcontroller was a large task. In order to streamline and effectively implement our design, the use of the ISO/IEC 9899 standard will be quite useful. This standard specifies the representation, syntax and constraints, semantic rules, and representation of input and output data. The standard as a whole applies to the entire C programming language, but we used specific elements that apply to our application.

Any programming language has a myriad of syntactical elements which are necessary for any program written in the C language to run. These are included in annex A of the standard. In addition to this, there are many types and identifiers that are useful when programming in C. Identifiers can denote an object, a function, a tag or a member of a structure, union, enumeration, a typedef name, a label name, a macro name, or a macro parameter(14). These identifiers have a defined scope which is determined by the placement of its declaration. The scope of the identifiers can fall into one of four categories: function, file, block, and function prototype(14). Each of the scopes is clearly defined and described in the standard.

4.1.2 Battery Standard

In the Pet Pal design, the battery is one of the most essential modules, as it powers all of the circuitry. Therefore, it is important for us to follow any standards associated with Lithium-Ion batteries to ensure that power is safely supplied to the circuit. IEC standard 61960-3:2017 describes the performance for “Secondary lithium cells and batteries for portable applications", or more specifically “Prismatic … secondary cells and batteries”. This standard should help us to better understand how to implement our batteries in circuitry, and what performance should be expected from our batteries.

4.1.3 Bluetooth Standard

For our project, the Bluetooth module is very important to the flow of data from the collar to the application. Due to this, it is important for us to follow all standards involved with implementing Bluetooth communication. IEEE standard 802.15.1-2002 defines the lower transport layers in a Wireless Personal Area Network (WPAN), specifically Bluetooth wireless communication. This includes the physical layer (radio) and the Medium Access Control (MAC) layer. In addition, the standard states that communication must operate at 2.4 GHz and must support the dual transmission of voice and data. Standard 802.15.1-2002 was published on June 14th, 2002, but has since been superseded 802.15.1-2005 and is no longer maintained. Although this standard is not maintained by IEEE, it is still widely used in Bluetooth communication today(29).

4.1.4 Microcontroller Standard

For the different functionalities of our collar, the microcontroller is the most important subsystem due to its control of all data. Therefore, it was important to follow the standards involved in data communication between the microcontroller and other components. There are a few different standards that define the physical communication between pins as are listed below. In addition, each module has its own standards that must be followed through the module's data sheet. It was important for our team to follow these data sheet standards to ensure correct communication.

4.1.5 UART Standard

The Universal Asynchronous Receiver Transmitter (UART) is one of the most common data communication standards due to its simplicity in nature. Our team plans on using UART for communication between the microcontroller and multiple modules. As far as scalability, UART defines communication as one-to-one or transmission between two devices only. Each device has a Transmitter signal (Tx) and a Receiver signal (Rx), which uses two wires for each device. This design aspect means that UART uses full duplex communication. Meanwhile, data is sent asynchronously, which means there is no clock wire to synchronize the devices. This is done by adding a start and stop bit to the data packet to define transmission periods. This data is sent in a data package with bits for the start bit, the data frame, the parity bit, and the end bit(s). To start communication, the voltage level of the initiating device's Tx is pulled from high to low for one clock cycle. This first bit in a UART data packet is called the start bit. Once the start bit occurs, the receiving device's Rx signal will be ready to accept the desired data. The next 5 to 8 bits (9 bits can be used if there is no parity bit) are the data frame, which can be specified as starting with the Least Significant Bit (LSB) or the Most Significant Bit (MSB). The data frame specifies the actual data that is being sent between the devices, aside from the bits that support the communication aspect of the standard. Next comes the parity bit, which is used to check if data is sent correctly and is not present in all data packets. A bit of 0 (even parity) means that the bits equal to 1 in the packet should be equal to an even number. Conversely, odd parity means that the logic high bits in the packet should be equal to an odd number. If the bit does not align with the packet's parity, there was an error in the packet and it must be resent. The final bit is the stop bit, which consists of one or two bits. Converse to the start bit, the stop bit pulls the Tx line of the sender device from low to high to signify the end of the packet(31).

4.1.6 SPI Standard

Serial Peripheral Interface (SPI) is another popular data communication standard between microcontrollers and peripheral modules. This standard will be used to communicate between our Bluetooth module and our microcontroller due to its large data throughput. Unlike UART, which can only send 5-9 bits of data at a time, SPI can send as many data bits in a row as desired. This is due to SPI's synchronous behavior, which takes away the need for data checking, start, and stop bits. In addition, SPIs chip select wire allows many different devices to be connected to a single device, called the master. This allows data communication between multiple devices, which is much more beneficial to the ease of development than one-to-one protocols such as UART. However, one downside to this behavior is that SPI uses 3 or 4 communication wires, which results in a more complicated system of data transfer. For our project, we will be using the 4 wire version of SPI. In this version, the four communication wires are as follows: Clock (SCLK), Chip Select (CS), Master In Slave Out (MISO), and Master Out Slave In (MOSI). As can be inferred from the wires, there is a master and a slave in communication. The master in our case is the microcontroller, while the slaves are any peripherals that transfer data using SPI. The CS wire specifies the slave that is currently communicating with the master. Meanwhile, SCLK sends a clock signal to synchronize the master and the slave. MISO is the wire that transfers data from the slave to the master, while MOSI transfers data conversely.

4.1.7 I2C Standard

I2C is a common communication protocol which is used widely in the field of electronics. It was originally developed by Philips Semiconductors (now NXP Semiconductors) as a simple, bidirectional, 2-wire bus. The two required lines are SDA and SCL. SDA stands for serial data line and SCL represents serial clock line. In I2C communications, each device is software addressable and a master/slave relationship exists at all times between 2 devices which use this protocol. There are also 3 defined data transfer speeds in the I2C standard: standard-mode, up to 100kbits/s, Fast-mode, up to 1 Mbits/s, and High-speed mode, up to 3.4 Mbits/s. Unidirectional transfers can also be made and can be at speeds up to 5 Mbits/s. Since devices connected to the I2C bus are software addressed, as many devices as the user wants can be connected.

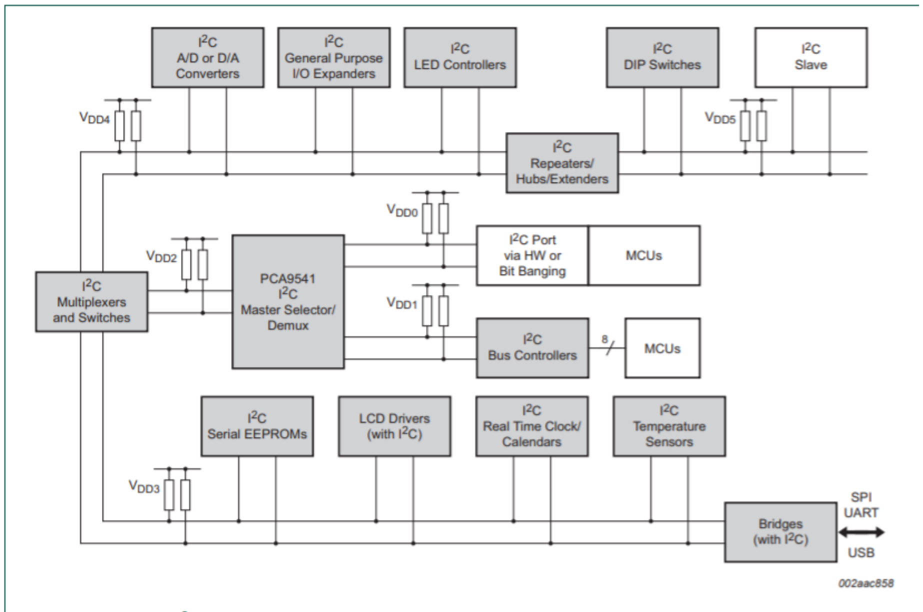

I2C A/D or D/A Converters
I2C General Purpose I/O Expanders
I2C LED Controllers
I2C DIP Switches
I2C Slave
VDD4
VDD5
I2C Repeaters/ Hubs/Extenders
VDD0
I2C Port via HW or Bit Banging
MCUs
VDD2
PCA9541 I2C Master Selector/ Demux
I2C Multiplexers and Switches
VDD1
I2C Bus Controllers
8
MCUs
I2C Serial EEPROMs
LCD Drivers (with I2C)
I2C Real Time Clock/ Calendars
I2C Temperature Sensors
VDD3
Bridges (with I2C)
SPI
UART
USB
002aac858
Fig 1. Example of I2C-bus applications

Figure 4.1
I2C Bus with Multiple Devices

	UART	SPI	I2C
Max Throughput	115200 bps with standard rate of 9600 bps	10 Mbps	5 Mbps
Communication wires	2	4	2
Masters/Slaves	1/1	1/Infinite (bound by # of pins)	Infinite/1008
Clock Signal	No	Yes	Yes
Data Check Method	Parity Bit	None	ACK

Table 4.1
Communication Comparison

4.1.8 Animal Health Standards

One goal of the Pet Pal is to provide users with their pet's vitals. In addition, we used an alert system with our application to notify the user when their pet's vitals enter dangerous ranges. As a result, we needed a standard to define the range of healthy vitals, so that we could define when the collar should send notifications to the user. For an average dog, the normal temperature range for the average dog is 100 - 102.5 degrees Fahrenheit. In addition, veterinarian professionals recommend seeking medical attention if a dog's fever goes below 99 or above 104, so we planned on using these bounds to send an urgent alert to the user. Meanwhile, if the dog's temperature was not in the average or alert range, we will send a warning notification to the user.

Other than the temperature, we planned on monitoring the pet's heartbeat. Therefore, we needed to follow a standard to determine when to notify the user of their pet's heart rate. However, unlike a dog's temperature, its standard heart rate is not uniform for all sizes and breeds. Therefore, the Pet Pal's application was initially designed to send notifications for different ranges, depending on the size of the dog. According to veterinarian professionals, small dogs have a heart rate between 100-140 beats per minute, while large dogs have a heart rate between 60-100 beats per minute. These ranges would have served as the operating ranges of the heart rate monitor, meaning that notifications would be sent to the user when outside of these ranges.

4.1.9 PCB Standard

The IPC-2221B standard describes and establishes requirements for the design and manufacturing of printed boards. The standard goes into depth on many requirements of PCBs including rigidity, material, and interconnections. This standard gives us a reference for what is available on the market. It allowed potential for this design to evolve and change in ways that could potentially transform our design. However, this could also add extra constraints to the design. This standard revealed that there is room for the rigidity of the board to change. This was helpful as generally these boards are perfectly flat, but, quite obviously, collars curve around the neck of an animal. There was potential to explore this as an avenue to improve our design.

4.2 Constraints

In this section, we will talk about the constraints on our project design. When defining constraints, the first step when considering a constraint is to make sure that it is realistic. If we are not careful, we could impose a constraint on our product that is not realistic and therefore does not actually affect the development of the Pet Pal. Therefore, it is important to make sure that all constraints are realistic and strongly related to our development process. Although constraints seem negative when thinking about them on a surface level, there are many benefits to understanding them. For example, once we define the constraints of our project, we will know the limitations to our creativity in the design of the Pet Pal. The project's constraints set boundaries that will narrow down the scope of our project and help us define the obstacles to overcome that we need to overcome to have success in our final product. In addition to setting boundaries, constraints require us to be more creative to overcome the resultant shortcomings. In addition, research on constraints can lead to finding new standards, which can be related to a variety of different standards. Prevalent categories of constraints include economic, manufacturability, sustainability, time, ethical, health, safety, environmental, social, and political. Although this list is quite extensive, there are other categories of constraints that can be applied in a variety of projects.

4.2.1 Economic and Time Constraints

Currently there are other designs for similar products on the market. Our goal for this project is to both create an improved implementation, but also to create a lower cost solution with more functionality. Our total budget for this project is $800, but our goal is keep this number as low as possible while maintaining as much functionality as possible all in a timely manner. To keep this design low cost, sometimes lower quality components will still be used which can still maintain the proper function.

One unique time constraint that we endured over the course of this project is due to the pandemic caused by the COVID-19 virus. The state of Florida and UCF both recommend social distancing, so group gatherings are highly frowned upon. This has limited the amount of time that our team can meet in person to work on testing, prototyping, and general meetings. If not for recommended social distancing, we would have used UCF or our apartments as primary meeting spots weekly, since we all agree that meeting in person is beneficial to the success of our project. However, we have communicated almost primarily through voice and video chat platforms such as Zoom and Discord. Since we have the same timeframe as Senior Design students from previous semesters who did not experience this constraint, our team has less flexibility compared to previous teams when it comes to communication. In addition, we had to limit the amount of times that we can meet to work on the electronic design, so our team has to be very organized and thorough at these meetings to prevent additional excessive meetings. If we did not have this constraint, we would meet in person often and would likely have a smoother design process.

The timeline for this design ended in the Senior Design 2 at the end of the Fall semester of 2021. This could pose a problem as, due to the tumultuous times, ordering components could be an issue. On many distribution websites there is a standard manufacturer lead time. This lead time can be very short or very long due to backlog or surplus. Because of the events of the past year, many components which are required by major companies cannot be acquired in a timely manner. This would be a major constraint causing our design change in major ways on the component level. At the end of Senior Design 1, there are seven to eight months for us to successfully develop, test, and implement our product. By following our predetermined milestones, we can easily develop and implement our design in time for the final demo this coming december.

4.2.2 Manufacturability and Sustainability Constraints

For the manufacturability of the Pet Pal, it is important to consider the weight of the collar. Since our product contains many different modules, the collar can become heavy if we are not careful with our part selection. However, our project goal is to allow a pet owner to maintain their pet's health and safety, while also providing the basic features of a pet collar. One of these features is that the collar is lightweight relative to the pet's weight, so that the pet does not feel uncomfortable when wearing the device. This could also cause future health problems if the collar is too heavy and the pet wears it too often. Therefore, we would like to produce a product that is lightweight so that the collar does not affect the animal in a negative way.

Another important constraint to consider in the design of our project is the Bluetooth range. Since our collar will use Bluetooth communication to send data between the mobile application and our product, we must consider the range of our Bluetooth module for collar functionality. Therefore, it will be beneficial for us to make a user's guide to indicate the range of functionality for the collar's communication with the application.
Regarding the sustainability of our product, it is important to consider the durability of the electronic casing and collar as a whole. For the Pet Pal to be successful for an extended period of time, it is important to protect the electronic parts so that the circuits are not altered by outside elements. Ultimately, we would like our design to be waterproof so that animals can wear the device in a larger variety of situations, but our time constraint makes this feature difficult to achieve. Therefore, our minimum constraint for sustainability is to make sure that the collar does not fray easily and that the casing fully contains all internal electric parts.

One manufacturing constraint to consider is the time it will take for use to receive our parts. Since we are college students, we do not have extensive access to electronics manufacturing companies to expedite our parts. We must order them online and hope that parts do not get put on a backlog, which would greatly increase the time it takes for the manufacturing company to ship our parts. In addition, we cannot guarantee efficient manufacturing processes and durable materials for our parts since we cannot oversee the production of the products.

One sustainability constraint of our product is related to the lifetime of the batteries that we will use in our power circuitry. Since the power for the Pet Pal comes from batteries, we must include functionality in our design so that the user will be able to change the batteries when the device is low on power. In addition, we must create a way to notify the user that the batteries are low, which will likely be done through our application. However, another important edge case to consider is when the phone is not connected to the collar via Bluetooth. To solve this case, we also need to add functionality to our circuitry to trigger some sort of alarm or LED when the batteries are low. It will also be beneficial for our product to come with a set of batteries so that the user does not have to buy any from a third-party source when they first receive the Pet Pal. In addition, we should specify the type of battery in our user's guide so that they know which batteries to buy when the originals run out of power.

Although we will aim to create a durable product, we understand that there may be some unforeseen technical faults that could arise over time. This is due to our lack of experience with developing products such as the Pet Pal. Therefore, in order to market our product, we need to offer a warranty to protect any cases when projected or unforeseen faults arise. In addition to customer satisfaction, we can have the user send the faulted product to our team so we can determine what caused the fault and how to fix the fault in future versions.

4.2.3 Ethical, Health, and Safety Constraints

For our project, there are many different ethical constraints that we must follow to have a successful final product. One class of ethical constraints are UCF's University Syllabus Statements. Specifically, UCF's ethics statement contains the following: "presenting another's ideas, arguments, words or images as your own … contradict(s) the educational value of these exercises"(38). Therefore, we must make sure to cite all sources that we use in our documentation, while making sure that we come up with original ideas for our design. However, when it comes to citing a part schematic figure, we must first contact the part's manufacturer to ask for permission to use the schematic in our documentation. This special case takes extra steps due to the importance of the schematic to the manufacturer's design of the product. In addition, we must adhere to specialized constraints due to the pandemic caused by the COVID-19 virus. When we chose to meet on UCF's campus to utilize their study areas and hardware tools, we needed to adhere to social distancing guidelines.

Pertaining directly to our product, another class of ethical constraints deals with concerns caused by testing the Pet Pal. Specifically, the Animal Welfare Act (AWA) states that in order to use an animal in testing, one must register for a license to show that the animal is given proper care. Although our team would be able to create a safe environment to test our collar, obtaining a license can be quite time-consuming and animal testing is not necesssary to ensure that our product works as intended. In addition, since the Pet Pal contains multiple electronic components, the test animal would be in danger of harm from electricity. Therefore, we plan on testing our product using more empirical methods, while testing the temperature and heartbeat sensors on ourselves.

4.2.4 Environmental, Social, and Political Constraints

One important social constraint to consider for the Pet Pal is the design's appearance. In the current market, it is important for wearable products to be stylish and socially appealing. Many consumers consider the look of a collar when purchasing a product similar to the Pet Pal. Therefore, it is important for our team to produce a collar and casing that is aesthetically appealing in addition to its functionalities. Overall, we believe this constraint increased the marketability of our product.

Meanwhile, an important environmental constraint to consider is the GPS module's operating parameters. Many GPS modules have trouble getting an exact location when inside a building, especially smaller modules like the one we will use in our final PCB design. Therefore, we need to make a warning for users that the GPS functionality may be spotty when inside a building.

Another important environmental constraint is the effect of the environment on the functionality of our circuitry. Electronic parts are highly vulnerable to environmental factors such as dirt and water. One goal for the Pet Pal is that we want users' pets to be able to wear the collar outside as a normal pet collar. However, pets can be dirty and come into contact with many environmental factors that can affect our circuitry. Therefore, it is important for us to create casing for our electronics that will be able to withstand these factors. In addition to preventing our circuitry from failing, the casing will ensure that the pet cannot be electrocuted by malfunctioning electronics.

5. Design

In this section, we will talk about the design of the Pet Pal including circuitry and the mobile application. The design of the product is arguably its most important aspect, as it will determine the effectiveness and longevity of the product. In addition, the design is very important to the marketability of the product, as innovative design is attractive to potential customers and gives our team a better chance to impress our project's review panel in the Senior Design 2 Showcase.

One major aspect of the design is the hardware portion, which is known in our project as the Pet Pal. One goal in our hardware design is to ensure that all components are connected to a central microcontroller so that we can program the flow of data in the device. However, each component has different specifications which creates the need for circuitry such as voltage regulators, amplifiers, and other logic. Therefore, we must consider all additional circuitry needed to ensure that all components can receive power and are compatible with the microcontroller. Once the components are connected, they will receive commands from the microcontroller, which are received from the mobile application via a Bluetooth module. The components will then return the specified pet data (or play a sound in the case of the speaker) to the microcontroller, which will then send the data to the app with Bluetooth.

The other major subsection of our design is the software portion, which consists of the Pet Pal's mobile application and the corresponding embedded programming for the microcontroller. The main concern for the software is the transfer of data from the Pet Pal to its mobile application. More specifically, we will have to come up with a protocol for determining when to poll our sensors and how to store this data to provide useful information to the user. In addition, we will have to translate the different signals that our microcontroller receives from each sensor and provide a protocol for message and data packets to send to the mobile application. For our application, we would like to provide a clean and satisfying UI so that users have a good experience. In addition, we will have real time updates (including push notifications) on pets vitals and location so that users do not have to worry about their pet's safety.

5.1 Hardware

As shown in the block diagram below, our system has a GPS module, a heart rate and temperature monitor, and a bluetooth module, all of these devices will need an input voltage of 3.3 volts to supply the power. In the meantime, our microcontroller, ATMega328P, needs an input voltage of 5 volts to properly operate and the audio amplifier used for the audio circuit needs an input voltage in the range of 7 to 12 volts. Each block in the shown diagram has multiple ways to demonstrate. In this part, we will show the configurations and explain steps that are used to design the system.

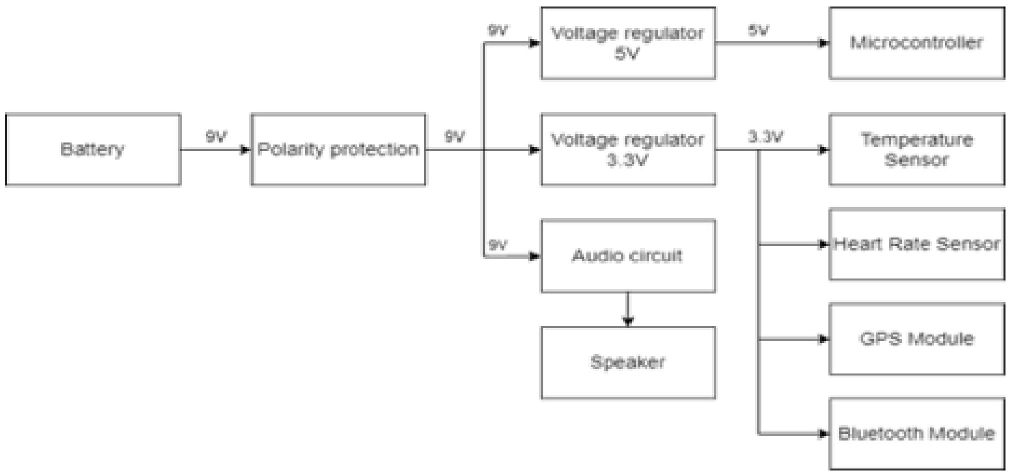


Figure 5.1
Overall Power Circuit Flow Chart

5.1.1 Polarity Protection

A replaceable battery with the minimum voltage of 9 volts is used to supply power for the whole system. Since we use replaceable batteries for our smart pet collar, there is always a chance that the negative polarity and positive polarity are accidentally connected inversely. When the polarity is reversed, anything can happen depending on what components are used in the circuit. Obviously, those polarised components will be more or less destroyed. In order to prevent the circuit from being damaged by inserting incorrect polarity, we create a simple polarity protection.

a) Diode

Our first solution was using a diode connected in series to the positive terminal of the input power. As we all know, the diode is a semiconductor component. There are multiple different types of diode, but in general, we have the zener diode and the ordinary diode. A diode has two terminals, positive terminal, known as anode, and negative terminal, known as cathode. For the ordinary diode, the anode terminal must have a higher voltage compared to the cathode terminal in order to turn on the diode. Each diode has its own turn on voltage, the difference between anode terminal and cathode terminal have to be larger than the turn on voltage in order to turn the diode on.The characteristic of a diode is when it is turned on, it will allow current to flow through in only one fixed direction that is from anode to cathode, and when it is turned off, it will stop the flow of the current going through. On the other hand, a zener diode let current flow through in both directions depending on which terminal has a higher voltage. If the anode terminal has lower voltage, the current will pass through from cathode to anode. By using a diode, connected in series with the source, when the negative terminal is accidentally plugged in, the diode will automatically act like an open switch which blocks the current flowing through. Conversely, the diode will become a short circuit when the positive terminal is connected, the current then passes from the source to the circuit to power the whole system.

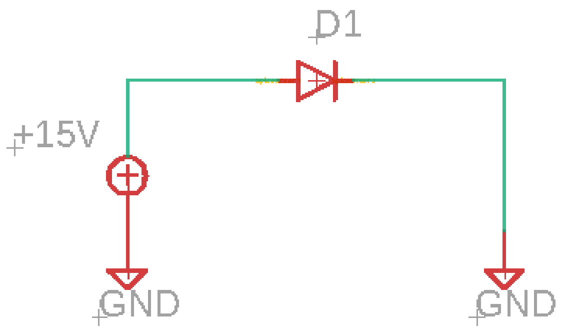


Figure 5.2
Polarity Protection with Diode

Although using a diode can theoretically protect the circuit from reverse polarity, heat increasing due to the current flows through the diode is the situation we want to avoid. All of the monitors and the microcontroller will be attached to the colar, for that reason, we need to make sure not only that the system must be light-weight, but it must also bring comfort to the pet. Heat can be created by the power that is applied to the diode. The formula for the power is:

$$P = V_F * I$$

Where V_F is the voltage drop of the diode and I is the current flowing through the diode.

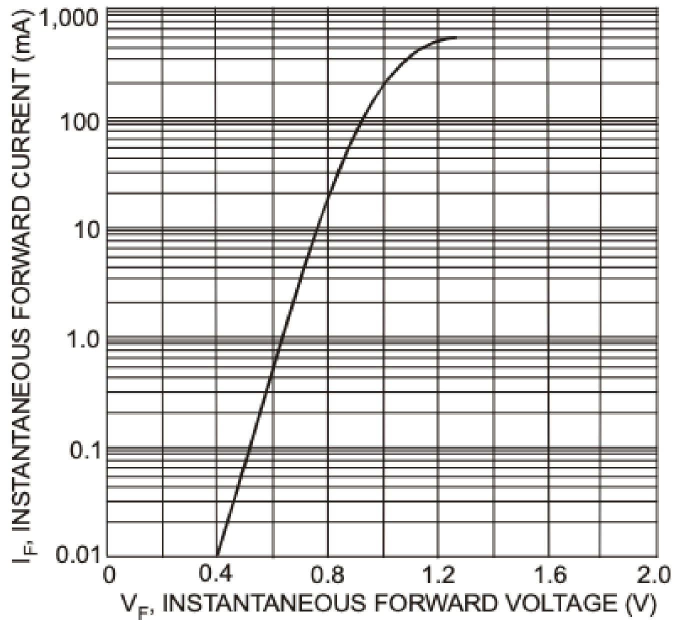


Figure 5.3
Typical Forward Characteristics

For example, if we are using the 1N4148 diode with 500 mA passing through, then based on the plot above, $V_F$equals approximately 1.2 volts. Thus, the power that diode waste is:

$$P = V_F * I = 1.2 * 500 = 600\, mW$$

Overall, using diodes in our design will always generate more or less heat.

b) P-channel MOSFET

MOSFET is a type of transistor, which has 2 types P-channel and N-channel. MOSFET stands for Metal oxide Semiconductor Field Effect Transistor. A MOSFET has three distinct terminals which are gate terminal, source terminal, and drain terminal. Thus, it is important to tell the differences between those terminals to avoid wrong polarity connection. Even though the bipolar junction transistor is also a type of transistor, MOSFET operates completely differently from the BJT. The MOSFET is on or off based on the voltage between gate terminal and source terminal. For N-channel MOSFET, we supply a positive voltage to the gate terminal, if the gate terminal's voltage is larger than the threshold voltage, the MOSFET will allow current flow through from the drain terminal to the source terminal . On the other hand, the gate terminal is usually connected to ground for P-channel MOSFET. In order to operate the MOSFET, we need the voltage between gate terminal and source terminal to be negative, which means

$$V_G - V_S = V_{GS} < 0$$

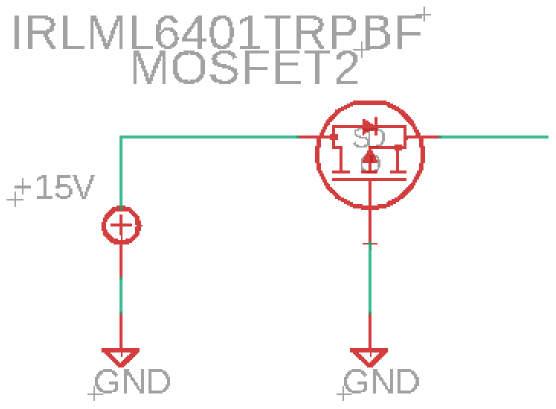


Figure 5.4
Polarity Protection with P-channel MOSFET

As shown from the schematic above, the drain terminal of the P-channel MOSFET is connected to the source, and the gate terminal is connected to ground. Therefore, the gate terminal $V_G$will always equal 0 volts. If the positive polarity is wired to the drain terminal just like above figure, we will have:

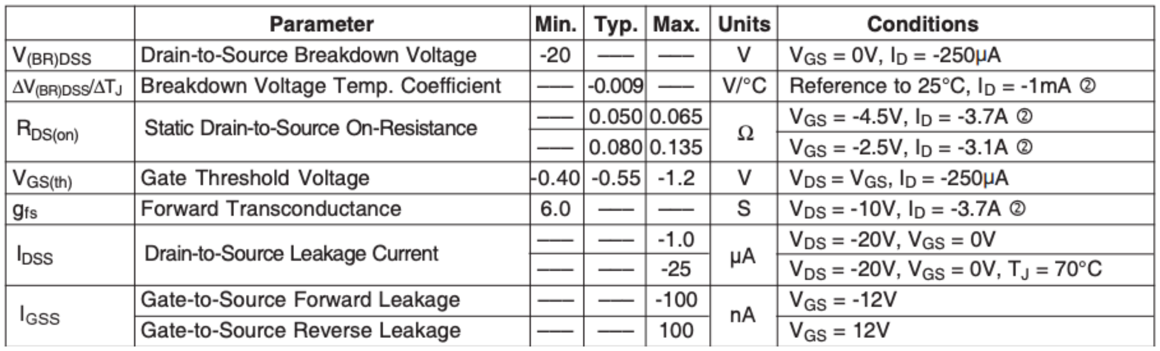
$$V_D \approx V_S = 15V => V_{GS} = V_G - V_S = 0 - 15 =- 15V$$

With the difference voltage of -15 volts, the MOSFET then will be turned on and the current can flow from the drain to the source terminal. Inversely, if we connect the negative polarity to the drain terminal we will have:

$$V_D \approx V_S =- 15V => V_{GS} = V_G - V_S = 0 - (- 15) = 15V$$

This is a positive voltage which doesn't satisfy the condition to turn on the MOSFET.

In fact, to turn on the MOSFET, the voltage between gate terminal and source terminal V_{GS} need to be lower than a given voltage which is called threshold voltage. Each type of MOSFET will have its own threshold voltage and we need to look it up using the datasheet before designing our system.



	Parameter	Min.	Typ.	Max.	Units	Conditions
$V_{(BR)DSS}$	Drain-to-Source Breakdown Voltage	-20	——	——	V	V_{GS} = 0V, I_D = -250μA
$\Delta V_{(BR)DSS}/\Delta T_J$	Breakdown Voltage Temp. Coefficient	——	-0.009	——	V/°C	Reference to 25°C, I_D = -1mA ②
$R_{DS(on)}$	Static Drain-to-Source On-Resistance	——	0.050	0.065	Ω	V_{GS} = -4.5V, I_D = -3.7A ②
		——	0.080	0.135		V_{GS} = -2.5V, I_D = -3.1A ②
$V_{GS(th)}$	Gate Threshold Voltage	-0.40	-0.55	-1.2	V	V_{DS} = V_{GS}, I_D = -250μA
g_{fs}	Forward Transconductance	6.0	——	——	S	V_{DS} = -10V, I_D = -3.7A ②
I_{DSS}	Drain-to-Source Leakage Current	——	——	-1.0	μA	V_{DS} = -20V, V_{GS} = 0V
		——	——	-25		V_{DS} = -20V, V_{GS} = 0V, T_J = 70°C
I_{GSS}	Gate-to-Source Forward Leakage	——	——	-100	nA	V_{GS} = -12V
	Gate-to-Source Reverse Leakage	——	——	100		V_{GS} = 12V

Figure 5.5
IRLML6402PBF MOSFET Electrical Characteristics

When the MOSFET is operating, there is a very small resistance between the drain terminal and source terminal which is used to calculate the power loss of the MOSFET. For example, we are planning on using IRLML6402PBF MOSFET, based on its datasheet, we can find its threshold voltage is about -1.2 voltages. On the other hand, we know the static drain-to-source on-resistance of this MOSFET equals 0.065 ohms. Assume that a current of 500 mA is flowing through the MOSFET, then the power loss is:

$$P = I^2 * R = (500 * 10^{-3})^2 * 0.065 = 0.01625\, W = 16.25\, mW$$

So if we use a MOSFET instead of diode, the power loss is 37 times lower, which is much more efficient compared to using diodes.

5.1.2 Voltage Regulator

Based on the requirement specifications, all three components including Bluetooth module, GPS module, heart rate and temperature sensor require a power supply of 3.3 volts. In addition, the microcontroller needs an input voltage of 5 volts. Thus, the objective of this part is to create a 3.3 volts voltage regulator as same as a 5 volts voltage regulator using either the operational amplifier or the voltage regulator, such as the low noise voltage regulator, SPX3819R2-L/TR, or the low quiescent current regulator, LM2936Z

The main purpose of a regulator is to produce a constant DC output voltage despite varying value of the load.

a) Operational Amplifier

The first approach to build a voltage regulator is using an operational amplifier combined with a NPN transistor. The configuration of this technique can be found in the lab manual of electronics II course. Below is the schematic that we used

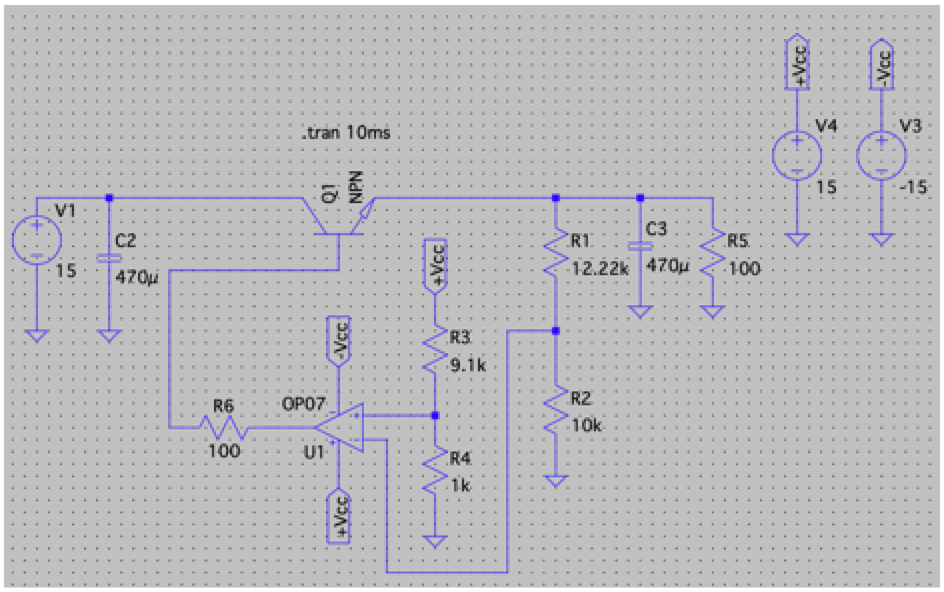


Figure 5.6
Simple Linear Regulator using Amplifier and Transistor

As we can see, there are six resistors and two capacitors used in the design. In addition, the amplifier is powered by two constant DC voltages with the value of 15 volts. The input voltage for the amplifier on the positive terminal is also 15 volts, and the input voltage on the negative terminal is our final output voltage. Each amplifier's terminal is connected to two different values of resistors, which helps us to obtain the expected output voltage using the formula:

$$Vout \ = Vcc * \frac{R1+R2}{R2} * \frac{R4}{R3+R4}$$

Since the output voltage is determined by resistor R1 and R2, we define resistor R3 and R4 as 9.1kΩ and 1kΩ, respectively. Two capacitors with the values of 470 µF are used, one at the input voltage and one at the output voltage in order to fill out the AC signal and guarantee that we can get the output voltage as pure as possible. As mentioned above, beside the main input voltage, there are three more DC voltages used to supply power for the amplifier, two of the DC voltages equal 15 volts and the other one is -15 volts. Replacing the values that we just defined into the equation above, we can find the values of resistor R1 and R2 which are 12.22kΩ and 10kΩ, respectively.

$$Vout = 3.3V = 15V * \frac{R1+R2}{R2} * \frac{1}{9.1+1}$$

$$=> R1 = 1.222\,R2$$

This method is practical since it will give us an output of 3.3 vols. However, there is no short circuit or thermal protection used. Moreover, we will need a lot of components and it might be affect to our PCB size

b) The SPX3819R2-L/TR voltage regulator

The SPX3819 is an ideal regulator for this part because it can produce an output voltage with low noise and low dropout voltage. With the load current in the range of 100 µA to 500 mA, this regulator has the dropout voltage varying from 10 mV to 700 mV. In a nutshell, this regulator needs at least 0.01 volt to be turned on. Below is a recommended configuration from the voltage regulator's datasheet.

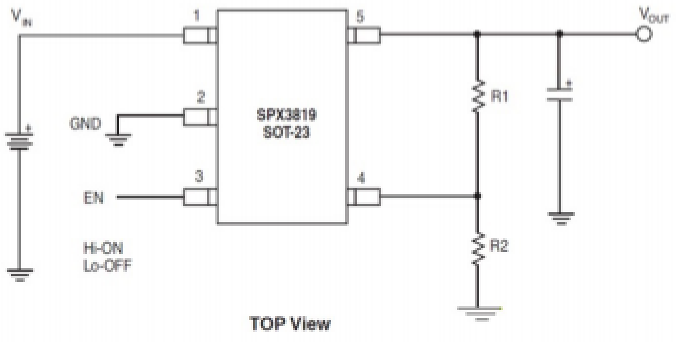


Figure 5.7
Typical Adjustable Output Voltage Configuration

From the configuration, we can only see five pins, however, the voltage regulator that we use in the schematic has eight pins in total. Three of the pins are not assigned, the other two pins number seven and nine are connected to ground. Pin number one is used to control input voltage which is directly connected to the output of the polarity protection and also connected to pin number three, Vin. Although there is just one capacitor connected to output voltage of the regulator from the recommended configuration, we add one more bypass capacitor wired between pin number three and ground in order to eliminate the AC signal from the input voltage, which ensures the voltage goes through the regulator is DC voltage. There is also a bypass capacitor connected between output voltage and ground to fill out the AC impedance just as the configuration. In addition, pin number eight is specifically used in this case since we need to make an adjustable voltage regulator. There are two resistors connected between the pin used for output voltage and pin number eight in order to obtain the desired output voltage. One resistor acts like a bridge between two pins, and another resistor connected to ground. From the configuration, our initial purpose is to create an output voltage of 3.3 volts, so the value of Vout is 3.3V. We use the formula given in the regular's datasheet to find out the value of resistor R1 and R2:

$$Vout = 3.3V = 1.235V * (1 + \frac{R1}{R2})$$

1.235 volts is the value of the adjustable output voltage for the SPX3819 regulator. It is required that the value of the resistor which is connected to ground must be larger than 10 kΩ, for that reason, we use 18 kΩ for resistor R2 and 30 kΩ for resistor R1.

$$Vout = 1.235V * (1 + \frac{30k}{18k}) = 3.2933V$$

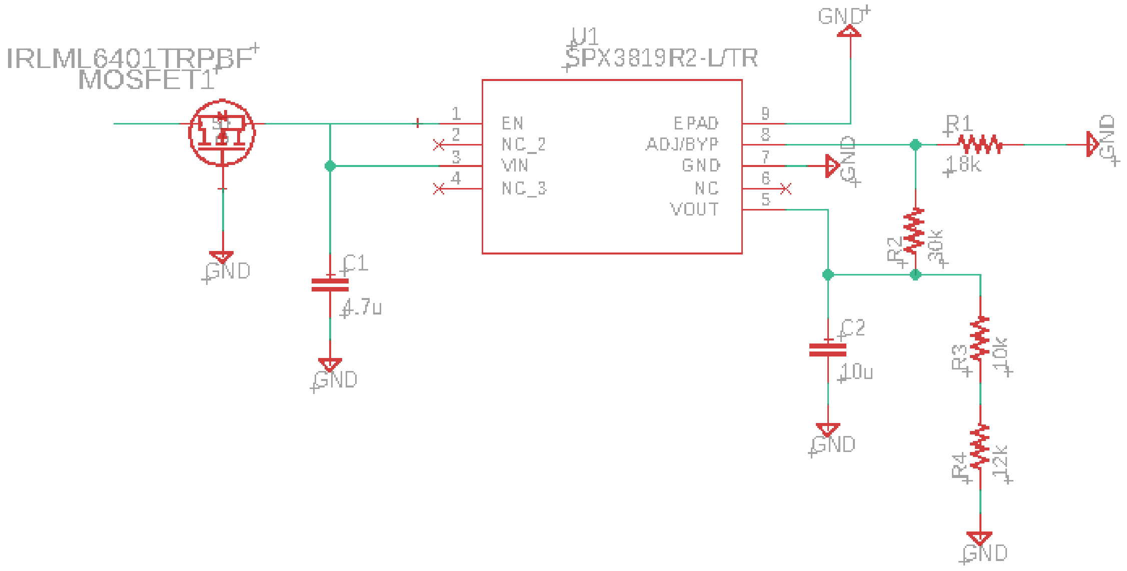


Figure 5.8
3.3V Voltage Regulator Schematic using the SPX3819R2-L/TR Regulator

c) The LM2936Z voltage regulator

After doing a few experiments, our team was aware that we need to find a voltage regulator that draws less current than usual. The LM2936Z regulator has the quiescent less than 50mA for both types LM2936Z-3.3 and LM2936Z-5. Moreover, it has a really simple design so that we don't have to use too many components which optimizes the size of the product.

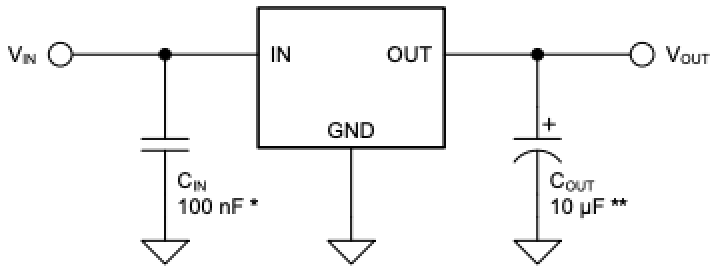


Figure 5.9
Simplified Schematic of the LM2936Z regulator

Overall, we used a 9 volts replaceable battery to supply power for the schematic above. The battery is first connected to the polarity protection. If the battery's polars are reversed, the polarity protection will stop the current flowing through, and there will be nothing happening to the circuit. On the other hand, the current goes through, enables the regulator and supplies power for the 3.3 volts regulator and 5 volts regulator.

5.1.3 Audio Circuit

Before designing the circuit for the speaker, we need to understand how a speaker operates. As shown in *Figure 5.10*, a speaker is a combination of multiple parts, such as cone, surround, dust cap, spider, pole piece, magnet, voice coil, etc…. The dust cap helps to protect the voice cole from dirt, and the spider is attached to the voice coil in order to make sure that the voice coil will be kept at the center of the magnet. It is not just simple that we supply a DC voltage to the speaker and connect the speaker to the microcontroller to make the speaker play music. A speaker requires a wire wrap around the voice coil, and an AC voltage is used to supply power through that wire. When the power changes from negative voltage to positive voltage or vice versa, its current will change the directions, which creates a magnetic field surrounding the voice coil. A magnet under the voice cole will then push or pull the voice coil depending on the polar of the magnetic field. The movement of the voice coil will affect the surround and create sound through it.

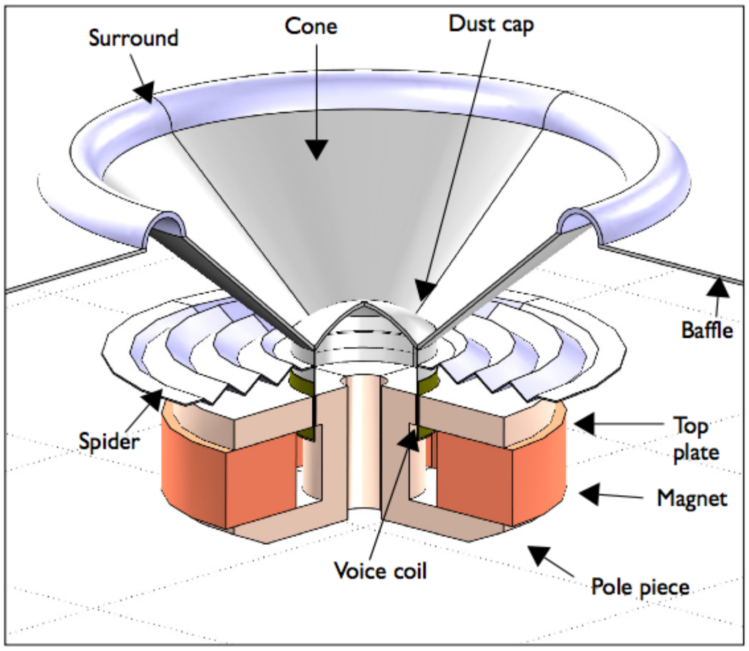


Figure 5.10
Structure of a Speaker

Thus, an AC voltage source is necessary in our circuit design to ensure that the speaker will normally operate. From the above sections, we have designed a 3.3 volts linear voltage regulator. Nevertheless, our audio circuit demands an AC voltage of at least 7 volts.

a) Simple Audio Circuit

The LM386 is an operational amplifier that is commonly used for designing audio applications. It has 8 pins in total and each pin has its own roles which mostly improve the quality of the audio. Nevertheless, we do not need a very good sound quality for our product. In other words, the sound quality is a feature that we can build on later when we have time. For now, we were trying to focus on making the speaker properly operate.

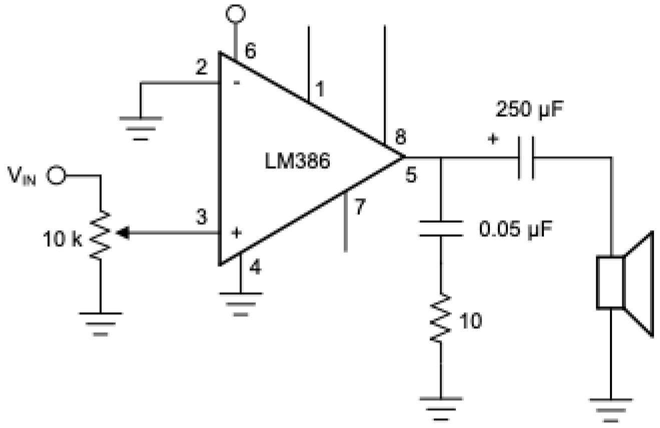


Figure 5.11
Simple Audio Amplifier Schematic

Inside the LM386 chip, there is an operational amplifier whose negative and positive terminals are assigned to pin number two and three, respectively. The output voltage of this chip is assigned to pin number five. In order to activate the LM386 chip, we need an DC voltage source connected to pin number 6. Recall that this amplifier only accepts a voltage source that is higher than 7 volts and lower than 12 volts. The LM386 will get the voltage source directly from the battery which is 9V. The negative terminal of the amplifier is wired to ground. As shown in the schematic above, the positive terminal of the amplifier is connected to a resistance of 10 kΩ and there is also an input voltage. This schematic is actually used for designs where the speaker has a jack plugging into an electronic device, such as a cell phone or a computer. The speaker will receive data from the electronic device and play sound as controlled by the device. The resistance 10 kΩ is where the volume controller is placed. Nonetheless, we do not need the volume controller on our product, and the microcontroller is the device that will control the output of the speaker, for that reason, instead of an audio jack and a volume controller, the positive terminal of the amplifier is simply connected directly to the microcontroller. There are two capacitors used at the output of the amplifier. One of the capacitors connected to ground has a really small value because it is used to filter out the noise from the input voltage and the microcontroller. Another capacitor is connected to the speaker, it is better if this capacitor has a high value since it is used to prevent the direct current flowing through the speaker, which causes damage to our device.

Notice that the above schematic does not guarantee the best sound quality. We will change some values of the capacitors and resistors in order to fill out the noise from the input voltage as much as possible to ensure a clear output sound. We also studied more about the applications of pin number one, seven, and eight of the LM386 chip to produce a better sound quality.

b) Improved Audio Circuit

As mentioned above, the first design of the audio circuit has a few things that are unnecessary for our product. In this part, we changed the connection of the audio amplifier to produce a better sound.

Pin 1 and 8 are still left open because these two pins are the gain setting pins, leaving them open means we did not want to set any specific gain for the audio amplifier but to use the default gain of the amplifier. We also added larger values of capacitors and resistors in order to fill out the noise of the audio signal and the amplifier.

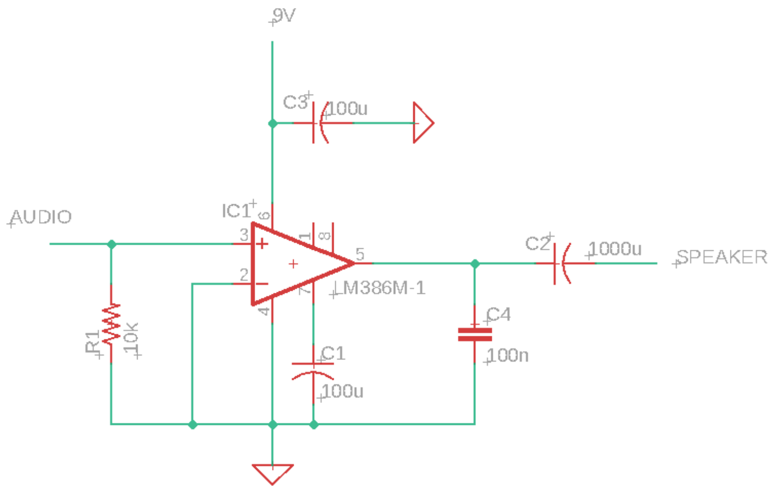


Figure 5.12
Final design of the Audio Circuit

5.1.4 Level Shifter

At the beginning, we decided to use the ATSAMG55J19 as our microcontroller. We realized that beside those components that take 3.3 volts as their power supply, the heart rate and temperature monitor uses 1.8 volts, which makes the connection between heart rate and temperature monitor and the microcontroller more complicated. The reason for the complication is because the microcontroller is expecting the vitality sensor to send its data at 3.3 volts while the vitality sensor is sending information at 1.8 volts. The only way that the microcontroller can receive data from the heart rate and temperature monitor is a circuit that can increase 1.8 volt to 3.3 volts and convert 3.3 volts to 1.8 volt. Our first approach is using an amplifier to boost 1.8 volts and using a voltage divider to get 1.8 volts from 3.3 volts. We realized it is impossible to do that because the wires that connect the heart rate monitor to the microcontroller are bidirectional. Thus, we started to design a level shifter, also known as voltage level translator.

A level shifter is a circuit that converts one voltage to another, and vice versa. Thus, this is a suitable solution for our bidirectional system. The concept of the voltage level shifter is pretty simple. We just need to use one N channel MOSFET, two resistors, and two DC voltage sources in order to create an efficiency level shifter. As we all know, a MOSFET will have three terminals, gate, source, and drain. The voltage difference between gate and source terminal must be larger than the threshold voltage, which is given based on the type of MOSFET, in order to turn on the transistor. When the transistor is on, current will flow from the drain terminal to the source terminal. there will be a small voltage dropped between those terminals. Inversely, when the transistor is off, two terminals drain and source are disconnected. The heart rate monitor will be connected to the source terminal and the microcontroller will be connected to the drain terminal. For the two DC voltage reference sources, one DC voltage is set to 1.8 volt and another one is set to 3 volts. These two DC voltage sources will not be changed because of the input. The 1.8 volt supplies power for the gate terminal while the 3.3 volts is wired in series with a resistor that is connected to the drain terminal. Thus, the voltage at the gate terminal is always 1.8 volt. When the heart rate monitor sends data, the output voltage is placed at the end of the drain terminal and the input voltage at the source terminal will vary from 0 volt to 1.8 volt. For the input voltage of 1.8 volt, this will make the difference voltage between gate and source terminal equals 0V, keep in mind that N channel MOSFET always has a threshold voltage higher than 0V. Therefore, the transistor will be turned off, and the output voltage equals the high reference voltage which is 3.3 volts. For the case where the input voltage is 0 volt, the voltage difference between gate and source terminal is 1.8 volt which will guarantee that the transistor is on. The source and drain terminals are now connected and will have the same voltage of 0 volts.

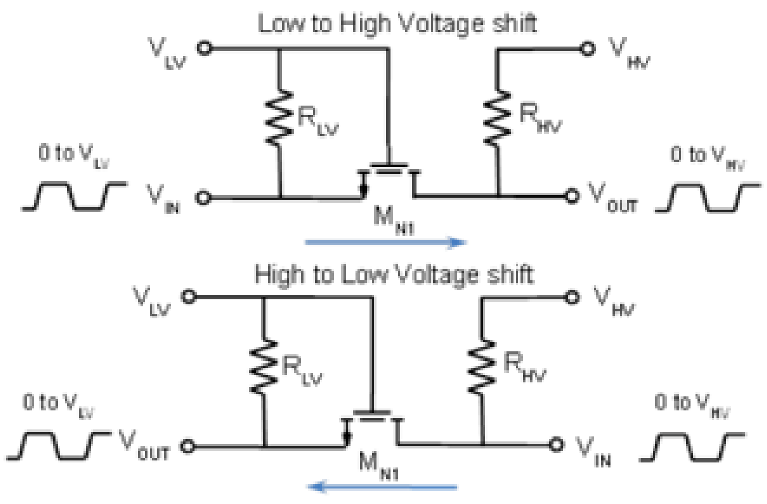


Figure 5.13
Voltage Level Shifter

Overall, the heart rate monitor sends a signal from 0 volt to 1.8 volts to the microcontroller, and the microcontroller will receive a signal from 0 volt to 3.3 volts, vice versa. We believe this is a good solution for us in order to read the data from the heart rate monitor. In addition, there are a lot of level shifters that are selling with a low price. These level shifters not only come with a datasheet with connection instructions but they also can be used to convert voltage for multiple devices, which is extremely convenient and easy to use. Nevertheless, we only need a voltage converter for a heart rate monitor, instead of buying a level shifter, we should build one by ourselves in order to optimize the size of our final product.

In senior design 2, we changed the microcontroller from the ATSAMG55J19 to the ATMega328P DIP, for that reason, the level shifter was eliminated from the final design.

5.2 Software Implementation

Our product will be launching with an Android Application to interact with the Smart Collar itself. As core features, the application allows the user to register an account with the system. The user's email, password, and user information is stored into database collections so that all this information is saved and immediately available as soon as the user logs in. The application also has a location page where you are able to get the position of the collar at all times. System architecture diagrams as well as additional information about the software implementation of this project follows below.

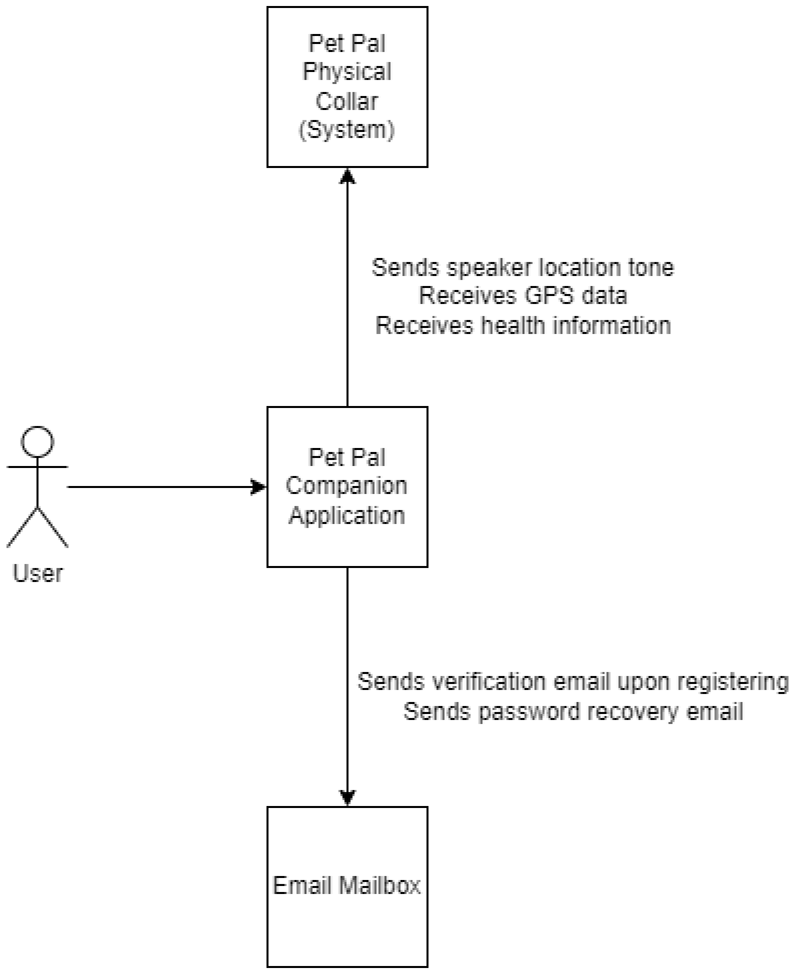

Pet Pal Physical Collar (System)
Sends speaker location tone
Receives GPS data
Receives health information
User
Pet Pal Companion Application
Sends verification email upon registering
Sends password recovery email
Email Mailbox

Figure 5.14
System Context Diagram

This figure includes information about how our system to be developed interacts with other pre-existing systems. As seen above, the Android application will need to have access and send information to the Smart Dog Collar, as well as the user's email mailbox.

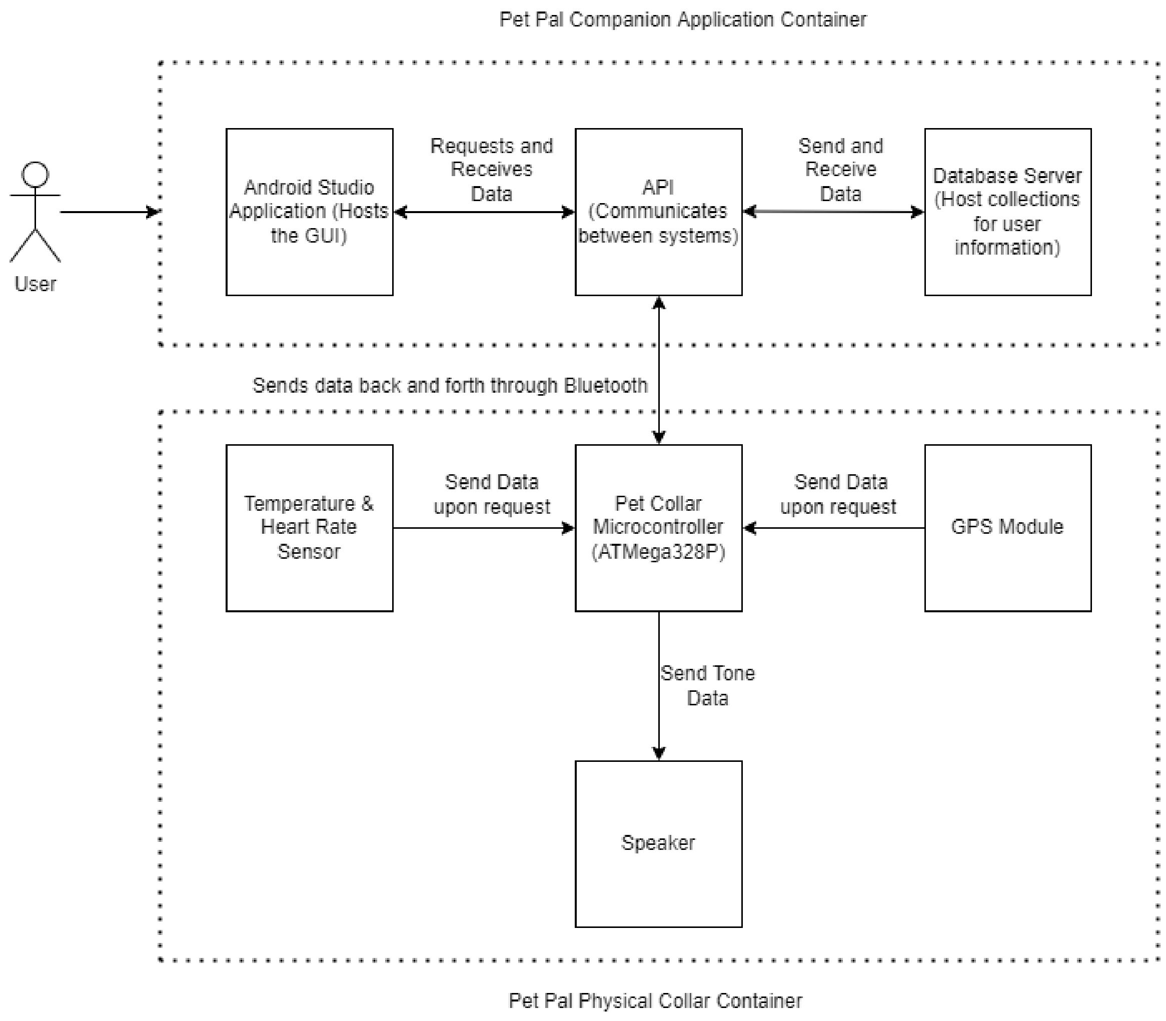

Pet Pal Companion Application Container
User
Android Studio Application (Hosts the GUI)
Requests and Receives Data
API (Communicates between systems)
Send and Receive Data
Database Server (Host collections for user information)
Sends data back and forth through Bluetooth
Temperature & Heart Rate Sensor
Send Data upon request
Pet Collar Microcontroller (ATMega328P)
Send Data upon request
GPS Module
Send Tone Data
Speaker
Pet Pal Physical Collar Container

Figure 5.15
Container Diagram: Smart Collar (Application and Physical)

This container diagram takes a deeper look at both the Smart Dog Collar system and the Android application system as described in the above context diagram. Each of these containers run in their own process space independent of each other. As seen in the figure above, the two systems interact with each other through the API on the application side, and the bluetooth module on the physical collar. The physical collar's container shows all of the modules and sensors that interact with the microcontroller. The application's container shows how the Android studio based application interacts with the API which then in turn interacts with the database server.

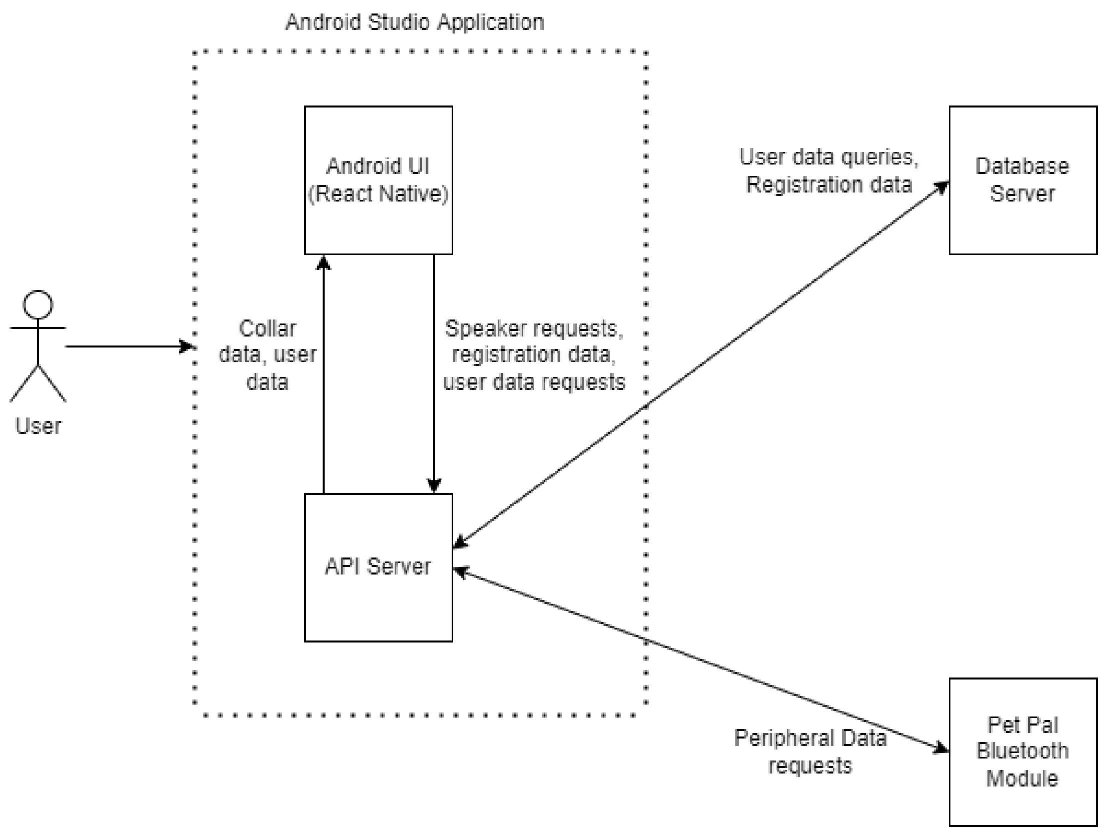

Android Studio Application
Android UI (React Native)
User
Collar data, user data
Speaker requests, registration data, user data requests
API Server
User data queries, Registration data
Database Server
Peripheral Data requests
Pet Pal Bluetooth Module

Figure 5.16
Component Diagram: Android Studio Application

A component is typically defined as "a grouping of related functionality encapsulated between a well-defined interface" (33). All components inside a container typically execute inside of the same process space. In this figure, we are diving into the Android Studio Application that was specified in the container diagram above. Here we can see how the front-end graphical user interface interacts with the API server. The API server then in turn interacts with the database server and the dog collar bluetooth module.

5.2.1 Application

In order to build the GUI and the application as a whole, our team has decided to use Android Studio. This is an IDE that you can download on your computer that includes a visual layout editor, flexible build system, and real-time profilers. These are all valuable tools that would assist us greatly in our coding of the application(36).

1. Visual Layout Editor

   Allows us to build the UI in a drag-and-drop style instead of writing XML by hand. This editor also allows us to preview our built UI on different android devices and versions so we can ensure compatibility across all fields.

2. Build System

   The build system on Android Studio allowed us to build multiple build configurations, meaning that we could have branching paths for our project to expand based on our wants and needs. Using the Gradle plugin, we were able to run and use Android Studio through the command line and even on devices that do not have Android Studio installed.

3. Android Profiler

   These tools provided real-time data on CPU, memory, network, and battery usage.

4. Android Emulator

   Unfortunately, our team does not all have Androids, so some of us had to develop the application without being able to test it on their own phone. Thankfully, Android Studio provides a tool to emulate any Android phone, which was useful for those of us who were not able to test application changes on a personal device. Since we were able to test the application on specific phones, we were confident that our code would work on a real Android device.

5.2.2 Android GUI Frameworks

React Native

React Native is a popular front-end framework for iOS and Android devices. This framework closely resembles the Node.js framework for web applications, and as a result is written in Javascript. One advantage to this framework is that our team has experience developing an Android application with React Native. In addition, we have found that this framework is easy to use when compared to other alternatives. However, React Native is a bit outdated, which restricts the creativity in our GUI design in comparison to more modern alternatives. Another advantage of this framework is the use of Javascript to design the application. Javascript is very extendable to many different libraries and databases, which simplifies the API and database development. In addition, React Native allows our Javascript code to be used for both Android and iOS versions of our application.

Flutter

Flutter is a new framework from Google that is gaining popularity among mobile applications. However, one disadvantage of this framework is that it is implemented with Dart, which is a programming language that our team has not used before. In addition, our team does not have experience with Flutter as a whole, so this framework could cause a steep learning curve. Also, since Flutter came out in December 2018, there are still some bugs due to the limited time that developers have had to improve the framework.

However, Flutter allows for greater flexibility when compared to alternatives. In addition, it is scalable to multiple operating systems, which is useful for development teams using different platforms.

Kotlin

Kotlin is a mobile application development language that is rapidly gaining popularity among developers. Kotlin is a language that can run on the Java virtual machine, and is very similar to Java in its syntax. In addition, Kotlin is interoperable with Java code, making it easier to learn with Java experience than other languages. However, one downside to this framework is our team's experience with the language. Although our team has used Kotlin before, we do not have extensive knowledge because we have used it with React Native. Therefore, this framework would cause a bit of a learning curve, which could increase the development time. However, this framework is advantageous due to Java's age as a programming language. Kotlin gives the user lots of flexibility to come up with a creative design, while Java's libraries are much more outdated. Another advantage of Kotlin is that there is no need for external frameworks. Like Android development with Java, Kotlin can handle all of the aspects of the application, which includes the front-end GUI, the API, and the database.

Selection

When considering our options for front-end frameworks, we first ruled out Kotlin. We decided that ultimately the learning curve would be too steep and that other frameworks are more flexible. In addition, we found that using Kotlin for all components of our software stack could cause some concern for security since the database would be in the source code. When left with Flutter and React Native, we ultimately chose React Native due to our experience with the framework. Although Flutter is highly flexible and would allow us to be creative in our design, we would have to learn Dart as we have no experience with the language. In addition, there is concern for bugs since it is less than three years old at this time. Meanwhile, some team members have developed an application using React Native and have found that learning a new mobile framework can be time consuming and sometimes difficult. In addition, a huge advantage to this framework is that any changes we make to our React Native code in Javascript can be used for both Android and Apple devices.

5.2.3 Application/Microcontroller Connectivity

Regarding achieving connectivity between the application and the microcontroller, there are several methods that can be used:

1. <u>Bluetooth Module</u>

   Bluetooth is an API that allows data to be sent between devices with Bluetooth capabilities such as mobile devices, computers, speakers, and much more. This is useful to our design because our front-end frameworks have libraries that can communicate directly with a Bluetooth module to send data back and forth. In addition, this method disregards the use of C/C++ programming in addition to learning the JNI module, so there is likely less scope involved with this method. However, we will have to add a Bluetooth module to our microcontroller to communicate with the mobile application in this way, which will increase the cost of our project. Also, our team does not have extensive experience with Bluetooth, so communication syntax may cause some learning curve issues.

   <u>Communication</u>

   For our Android application, we will use Android Studio as the primary IDE. In addition, Android Studio has the "android.bluetooth" library to create an API between the application and our collar's Bluetooth module. The first step in starting communication between Bluetooth is to discover the devices within the phone's range. To read these devices, a BroadcastReceiver must be set up to read the address of the collar's Bluetooth module. Once the address is read, the devices are connected using client-server architecture. In Bluetooth protocol, each device acts as an open server to allow any other device to initiate connection. After connection, data can finally be sent using a byte buffer. This buffer will be used to send commands to the microcontroller, as well as to receive data from the collar's modules. In addition, the mobile application will act as the server when the collar detects that the pet's vitals are not in the recommended range.

2. <u>JNI</u>

   Java Native Interface (JNI) is an API for Java that can call native C and C++ library functions. This is useful for our project because our team has experience with microcontroller data communication through the C language. In addition, if we write our Android application in Java, this option would cater to many of our programming strengths. This is due to our experience using Java and C in many other projects and in previous curriculum. However, our team does not have extensive experience with JNI, so using this API may result in a large learning curve.

5.2.4 Selection

When choosing between APIs to communicate with the Pet Pal microcontroller, we first needed to decide which framework to use for the front-end of the application. Ultimately, we chose React Native, which is written in Javascript. Therefore, we ruled out JNI since it requires Java to connect to the microcontroller. In addition, JNI is an outdated and unpopular API that would likely cause a large learning curve and would restrict the flexibility of our API design. This leaves Bluetooth as the communication method with the MCU. To reflect this change, we added a Bluetooth module to our PCB design.

5.2.5 Microcontroller Programming Design

For our application to send data, we must program our microcontroller to send messages to the Bluetooth module, which will then send to the application. To do this, we use microcontroller IDEs to define the behavior of the microcontroller's data flow. A list of these IDEs are listed below.

Code Composer Studio

Code Composer Studio (CCS) is a commonly used IDE to communicate with Texas Instruments (TI) microcontrollers. If we decide to use a TI microcontroller in our design, this is one of the preferred IDEs. CCS contains a C/C++ compiler that can be used to design data communication. This is advantageous to our team because we have experience with C, C++, and CCS in general. Also, this compiler is free, unlike some competitors that would cause a large increase in our budget. However, CCS can be slow and unintuitive, which could add time to our application development. In addition, CCS is not extendable to non-TI microcontrollers.

IAR Embedded Workbench

IAR Embedded Workbench (IAR) is a popular embedded IDE that can be used to program a wide variety of microcontrollers from different companies. It is a very intuitive IDE that is much easier to learn and use when compared to other IDEs. Like many other embedded IDEs, IAR uses a C/C++ compiler to program a microcontroller. However, one major downside to this IDE is the cost of the different extensions of the workbench. There are only a few 1 month free trials for the different microcontroller extensions, while the other options require the user to submit a quote, which is estimated to be around $2000. However, there is little documentation on the internet about the price of these workbenches other than the general consensus that it is expensive. If we decide to use this IDE, we will likely have to finish our embedded programming within the one month trial so that we do not exceed our budget of $1000 from *Tabel 7.1*.

Arduino IDE

Arduino IDE is a commonly used embedded compiler for Arduino microcontrollers. Like most other embedded IDEs, the data flow is designed using C/C++ libraries. One downside to Arduino is that our team does not have experience with the IDE or Arduino microcontrollers in general. However, Arduino has good documentation and tutorials, so the learning curve shouldn't be too big. In addition, since our team has experience with C and C++, we will not have much to learn. Another advantage of this IDE is that it is free, which will save costs in our budget for other expenses. However, this IDE is not extendable to microcontrollers that are not produced by Arduino, so it would only be used for Arduino products.

MPLAB X

MPLAB X is a popular embedded IDE for Microchip Technology microcontrollers. Like the Arduino and Code Composer Studio IDEs, MPLAB X is designed to work exclusively with its company's microcontrollers. Therefore, we cannot use this IDE unless we use a Microchip Technology microcontroller in our PCB design. In addition, like many other embedded IDEs, MPLAB X code is written in C and C++, which means there shouldn't be too big of a learning curve if we use this software. Some other advantages of this IDE include auto-completion of header files, a live syntax checker, and customization of the workspace. Also, MPLAB X is free to download, so using this software would not cause any strain on our budget unlike some other alternatives.

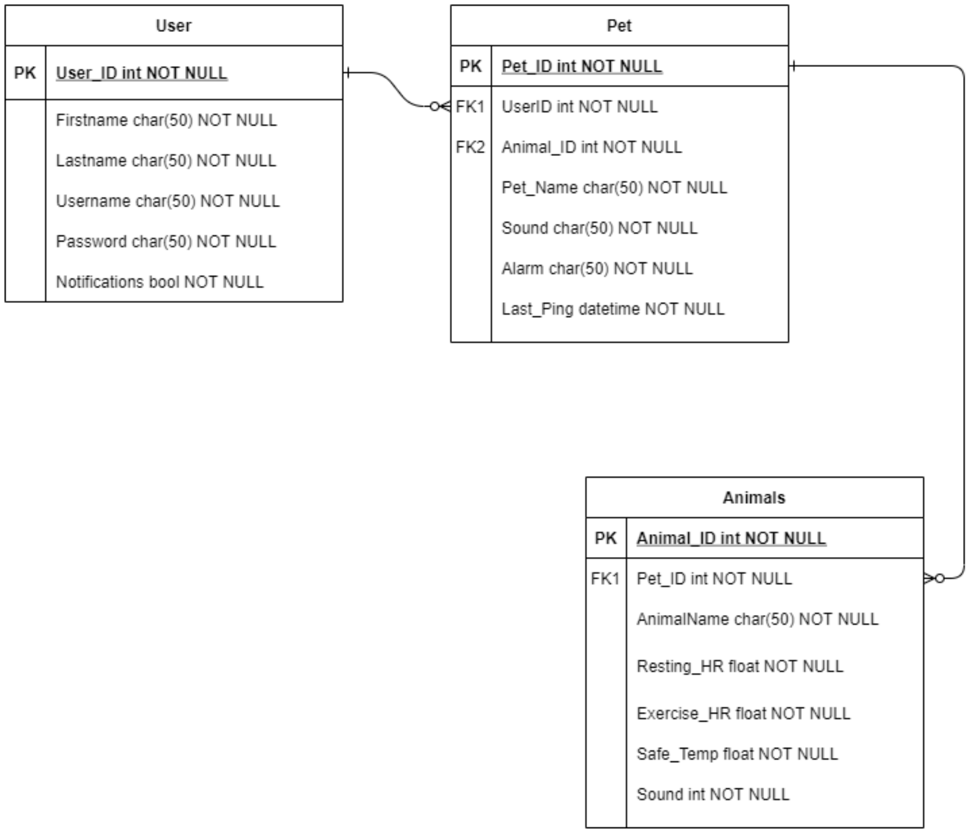
Selection

When choosing which embedded programming IDE to use for our microcontroller, the main factor in our decision was the microcontroller in our final PCB design. Once we decided on the ATSAMG55J19A-AUT, we were able to rule out the Arduino IDE and Code Composer Studio due to being compatible with only Arduino and Texas Instruments microcontrollers, respectively. Therefore, our decision was between IAR Embedded Workbench and MPLAB X. Although IAR Embedded Workbench is intuitive and easy to use, our team had a large concern with the cost of the software. We would have to complete the embedded programming in a month to avoid this concern, but we decided that IAR's advantages are not worth the time crunch that we would have to take on. Therefore, we decided on MPLAB X. This IDE will be advantageous to our development since the ATSAMG55J19A-AUT is a microcontroller produced by Microchip Technology, and therefore has built-in support for the MCU. In addition, initial research has shown our team that MPLAB X will be easy to learn and has many tools that will be useful for debugging.

5.2.6 Database

Entity-Relationship Diagram

We modelled the relationships between the different aspects of our application in the Entity-Relationship Diagram (ERD) below. We used this ERD as a baseline for our database design.



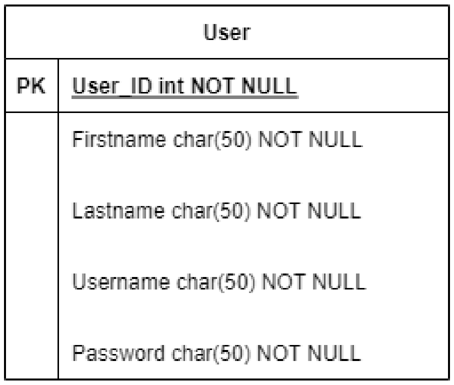
User
PK
User_ID int NOT NULL
Firstname char(50) NOT NULL
Lastname char(50) NOT NULL
Username char(50) NOT NULL
Password char(50) NOT NULL
Notifications bool NOT NULL
Pet
PK
Pet_ID int NOT NULL
FK1
UserID int NOT NULL
FK2
Animal_ID int NOT NULL
Pet_Name char(50) NOT NULL
Sound char(50) NOT NULL
Alarm char(50) NOT NULL
Last_Ping datetime NOT NULL
Animals
PK
Animal_ID int NOT NULL
FK1
Pet_ID int NOT NULL
AnimalName char(50) NOT NULL
Resting_HR float NOT NULL
Exercise_HR float NOT NULL
Safe_Temp float NOT NULL
Sound int NOT NULL

Figure 5.17

Entity-Relationship Diagram

Senior Design 2 Update:

This ERD does not reflect our final database, since we were unable to implement the Pet and Animals tables due to time constraints. In addition, the User table does not contain a notifications variable. The updated ERD is shown below.



User
PK
User_ID int NOT NULL
Firstname char(50) NOT NULL
Lastname char(50) NOT NULL
Username char(50) NOT NULL
Password char(50) NOT NULL

Figure 5.18

Updated Entity-Relationship Diagram

Although we have an ERD to model how our database should look like, there are many different databases that could have been used to store data for an Android app. Since our application is relatively simple, we chose a database for local storage only. Below are the database options that we found to fit best with our project:

1. SQLite

   SQLite is the most commonly used database for Android applications. It is like the traditional MySQL relational database, but stores local data. This is useful to our team, because a few of us already have experience creating databases with MySQL. However, there are also downsides to this software. For example, SQLite is not scalable for a multi-user system. In order to have multiple users in the same database, a different database software is required. In addition, the database lacks proficient data recovery techniques.

2. MongoDB Realm

   Although SQLite is the most common Android application database, MongoDB Realm is very popular among younger developers and has many advantages. Although our team is more proficient in SQL, we have experience with MongoDB syntax. MongoDB is desirable for its simplified syntax, documentation, and fast memory accesses. However, it lacks some useful SQL functionalities such as the ability to update dynamically, default and required values, and the auto-increment data type.

3. Firebase Cloud Firestore

   Firebase is another NoSQL database that can be used to store data for an Android application. One major benefit to this database system is that the data is very scalable across multiple different types of devices and interfaces. For example, this database can be used to create a mobile and web version of an application for the same software system. This database supports offline data synchronization, so any updates made to an offline mobile app will become synchronized as soon as the device is connected to the internet. In addition, Firestore is JSON based and follows object-oriented principles, so it would fit in well if we decide to use Java or Kotlin to design the application. However, since it is not SQL-based, the syntax would have been a bit of a learning curve due to our team's lack of experience with this DBMS.

4. MySQL

   MySQL is a relational database that has been very popular for many years. There are many ways to connect our Android application to this database, as it has been around for a long time. Some of these methods are listed below in the section on Database connectivity. One advantage to this database is that our team has experience designing an application with SQL. However, designing our database could be more work due to its relational aspect. In addition, SQL restricts creativity in the design of the application due to its age. Aside from its design, a MySQL database must be hosted on a server, which could increase the cost of our application. All of our users' data is stored on the server, which lessens the security of the application's data and increases the steps taken to ensure that the data is secure.

Selection

For our application, we wanted to use software stack components that were easy to learn, flexible, and easy to implement with our other components. Because of this, we ruled out Firebase and MySQL. Although Firebase has many impressive and useful features, our team would have to spend time learning its syntax before constructing the application. In addition, MySQL is an outdated database that would require our team to design the database based on relational concepts, which restricts the flexibility and creativity in our design. When left with SQLite and MongoDB, we ultimately decided on MongoDB due to the flexibility of the database and previous experience. Although SQLite is very popular for Android apps, it lacks flexibility due to its SQL-based syntax and would require extensive research to learn how to use it.

5.2.7 Application/Database Connectivity

Dealing with connecting the database to the application, there are several methods available to use:

1. JDBC

   JDBC stands for Java Database Connectivity and it is the API that is considered the industry standard for connectivity between the Java language and a range of SQL databases. In order to use this API, you need a driver that can mediate between the application and the database. According to the Java JDBC API(55), you can write the driver using a mixture of java and Java Native Interface (JNI) methods. This might prove helpful to us since our team already plans on using JNI methods for achieving communication between the microcontroller and the

application. Using JDBC allows the application the ability to directly communicate with the database. This is applicable in instances such as: adding a new user, adding a new pet, and adding specific data and preferences about each individual pet.
In addition, JDBC is compatible with Android Studio, which is an IDE that we plan on using for developing our application.

2. Android Studio APIs

   For the databases listed above that hold local data, there are APIs to construct and query the database. Since we will be using Android Studio to develop our app, we will use Android compatible libraries to communicate with our local database. For example, MongoDB Realm has an Android Studio API that can be imported using the "io.realm.RealmObject" library. This method is advantageous because there is no need for an API server, and the libraries follow better object-oriented practices.

3. Custom API Server

   For databases that store data on a centralized server, Android does not have a built-in API that can be used to communicate with the database. In these cases, we must create an API server using a common database API stack element such as PHP or Node.js. In addition, all of our users' data will be stored on a modifiable database, so we will have more flexibility with testing and maintainability. However, there are many downsides to this method. This API requires more security maintenance to ensure that our users' information remains private. In addition, our team will have an added cost to our budget, due to requiring an online server to run the API.

Selection

When selecting our database API for our application, we decided that our number one priority was to make sure that our API is flexible. For this reason, we eliminated JDBC first. JDBC is a much outdated database connectivity library for Java, so it would restrict a lot of aspects of our design. In addition, the other components of our software stack restricted the methods that we could choose from. Since we decided to use React Native for the front end of our application, we will be using Javascript to write our GUI, so JDBC cannot be used since there will be no Java coding in Android Studio. In addition, although Android Studio APIs can be easy to learn and flexible, this class of APIs is restricted since we are not using Android Studio directly to build our front end. Therefore, we will be using a custom API server to communicate between the front end and the database. This is advantageous to our project because our team has extensive experience with different types of these APIs, while maintaining a high level of flexibility due to the wide variety of open source frameworks and languages. Specifically, we will be using the Node.js library for Javascript, since it will be very simple to integrate with React Native. In addition, the Node.js library is very flexible due to the flexibility of the

Javascript language and its modern approach to APIs. One downfall to this method is the cost to run an API on a remote server. Although this can be a concern to some projects with a larger user base, our application does not serve a lot of traffic, so we decided to use one of the many free or negligibly low cost servers that are popular among smaller applications. Specifically, we chose Heroku, since there is a free tier able to service enough traffic for our application. Now that the API has been chosen, the software's full stack will be a MERN stack, which stands for MongoDB, Express, React Native, and Node.js. Some team members have developed an Android application with this stack, so it was advantageous as we did not have to incur a large learning curve over the course of software development. In addition, this stack is a very popular modern stack due to its flexibility and intuitive nature, so we were able to come up with a creative solution for our application that was user-friendly and aesthetically pleasing.

5.2.8 UI Design

One of the main goals of our application was to create a user-friendly UI that is also satisfying aesthetically. In order to meet this goal, we decided to create UI mockups of some application pages in Figma so that we can set a basis for what we want our app to look like. Since these figures were just mockups, they did not represent the final look of our application as we made quality and overall design changes before releasing the final application product. In the following sections, we first look at the original app design layout and compare it to the finished application running on an Android phone.

Login Page

The first screen that users see when they open the application is the login page. We used an email address for a user's unique identifier, since we also serviced email verification and password recovery features. As you can see in *Figure 5.19*, we have a link to our registration page in addition to a button that allows the user to send a password reset email with a code. This code is used to verify the user so that they can securely reset their password.

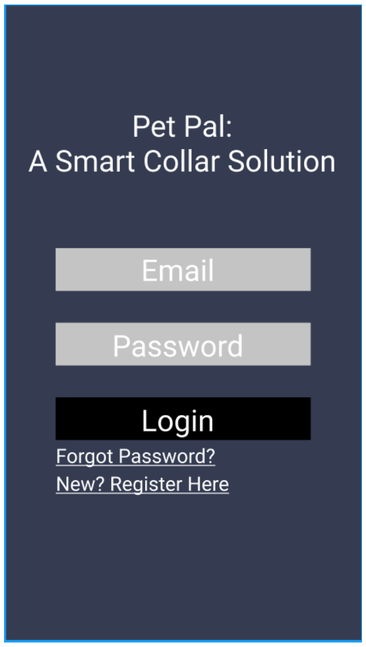


Figure 5.19
Login Page Mockup

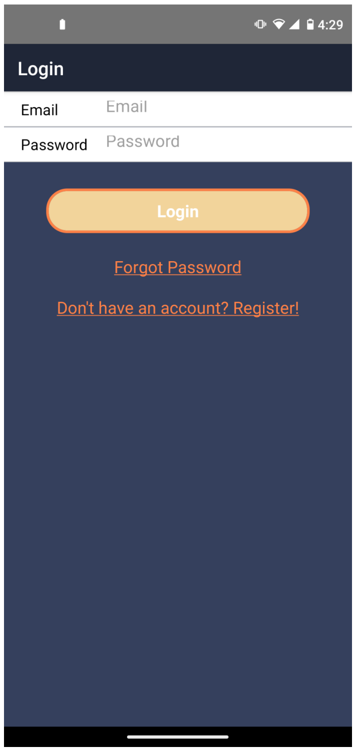


Figure 5.20
Login Page Screenshot

Figure 5.20 highlights the finished login page of our application. As you can see, it is running directly off of an Android phone.

Recovery Page

If a user forgets their password, we have a functionality so that they are able to safely reset their password. To do this, the user must first tap on the "Forgot Password" link, which will send a verification code to the user's email and redirect to the "Recovery" page. On this page, the user is asked to input the code sent to their email along with their new password twice for confirmation. If the user inputs the correct code, their new password is stored in the database and they are able to login. Otherwise, the user will receive an error that the code is incorrect. The initial mockup for this page can be seen in *Figure 5.21*, while the final implementation can be seen in *Figure 5.22* and *Figure 5.23*

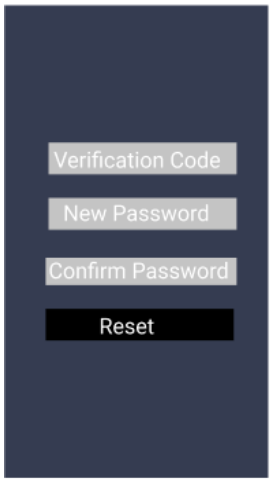


Figure 5.21
Account Recovery Page Mockup

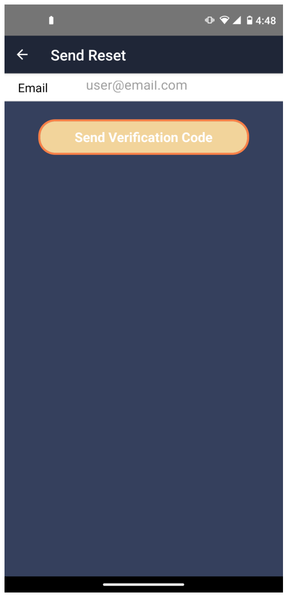


Figure 5.22
Send Reset Email Screenshot

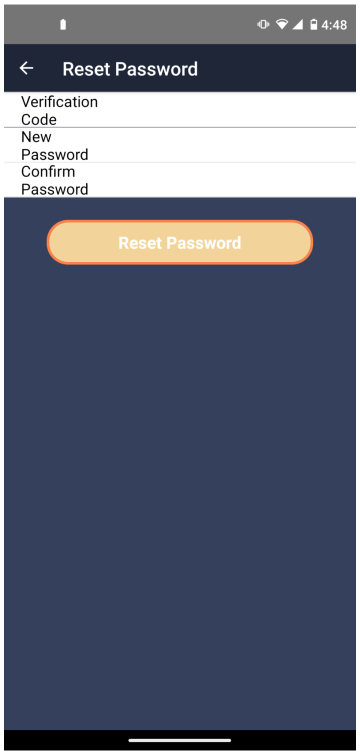


Figure 5.23
Account Recovery Page Screenshot

Registration Page

If the user does not have an account, they are able to click on “Don’t have an account? Register!” to be directed to the registration page for account sign up. As seen in *Figure 5.24*, the registration page asks the user to input their name and a valid email, and their intended password twice for verification. Once the user submits this info, it is placed in the database and the user is sent a verification email. Although their information is already in the database, they are not able to login until their account is verified through email.

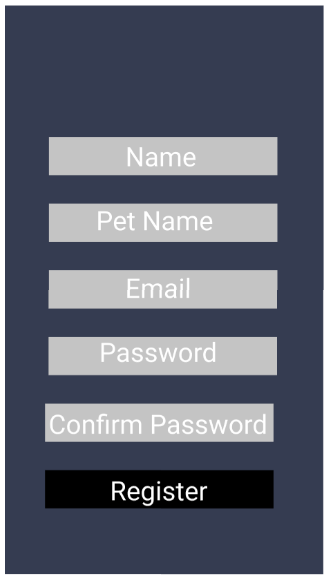


Figure 5.24
Register Page Mockup

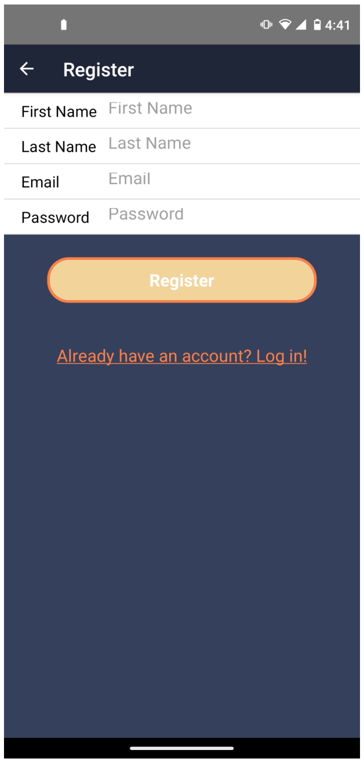
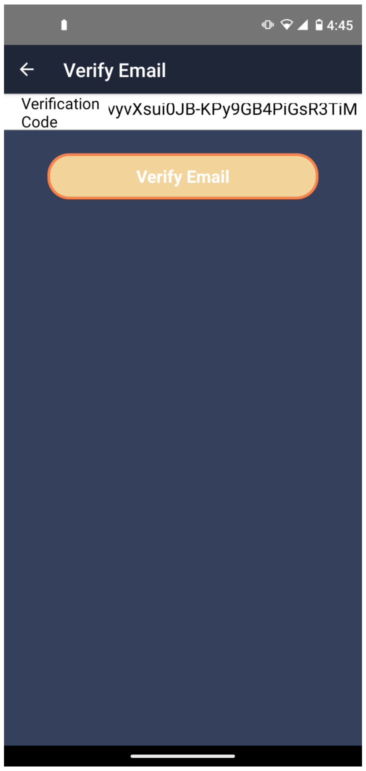


Figure 5.25
Register Page Screenshot



vyvXsui0JB-KPy9GB4PiGsR3TiM

Figure 5.26
Email Verification Screenshot

Home Page

This page acts as a sort of "Landing Page" for when a user successfully logs in to the mobile application. From here, the user has the choice to press a button and travel to their desired page. Our initial mock-up as well as final implementation of the home page is shown in the figure below, and the descriptions for all of the pages found on it are depicted in the coming paragraphs.

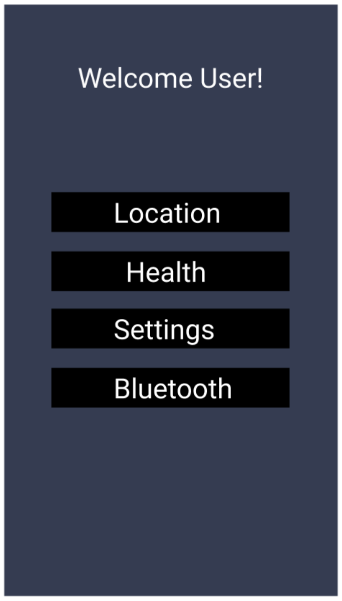


Figure 5.27
Home Page Mockup

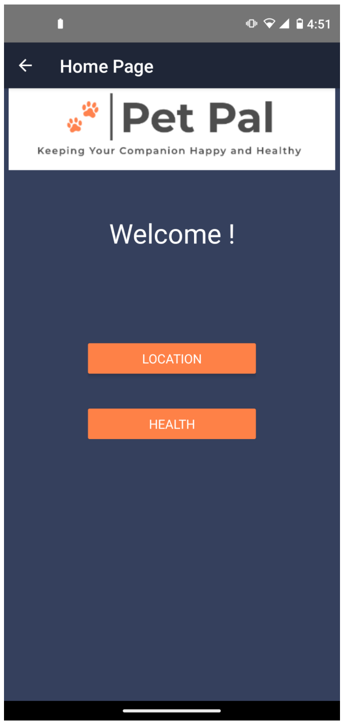


Figure 5.28
Home Page Screenshot

Health Page

The Health Page is the page to go to whenever the user wants to view priority health data and analytics taken from the physical Pet Pal collar. Here, the current pet's heart rate and temperature data is displayed for the user to see. These numbers were initially intended to automatically update and refresh based on the polling rate that we specify in the code. The user was intended to also have access to a button that makes the speakers on the collar system play a calming sound for the Pet. In addition, we intended to have a button labeled "Display Statistics" (as seen below) that opens analytics graphs displaying the history of recorded data. This implementation included one line graph for each health sensor: temperature and heart rate. In addition, the analytics were to be populated from the backend database. These line graphs were also supposed to be able to be viewed in several chronological ranges (i.e. viewing all data from a day, week, month, or even up to a year ago) with the furthest range accessible being a year. We also intended for each graph to have minimum, maximum, and rolling average lines for the user's information. Timestamps of data points will be displayed upon the user's selection. We initially had this goal due to the fact that this information could be easily pulled up and shown to a pet's veterinarian during a visit.

Senior Design 2 Update:

Unfortunately, we were unable to implement pet health statistics and graphs due to time constraints. Our final design works by clicking on the button related to the desired health measurement. Once the button is clicked, the current value of the desired sensor is displayed. For heart rate, this is not instant as the sensor must run for a duration of 10 seconds to get an accurate reading in beats per minute.

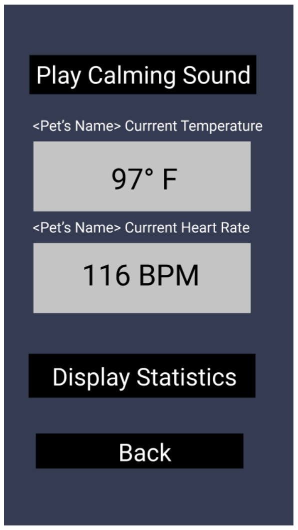

Play Calming Sound
<Pet's Name> Currrent Temperature
97° F
<Pet's Name> Currrent Heart Rate
116 BPM
Display Statistics
Back

Figure 5.29
Health Page Mockup

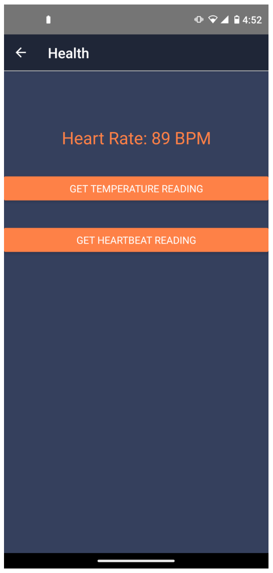


Figure 5.30
Health Page Screenshot

Location Page

Once the user is logged in and connected to Bluetooth, they are able to view their pets location by navigating to the location page using the "Location" button on the home page. Once the user entered this page, we intended to present their location along with the location of the Pet Pal collar on a regional map. The user was then able to scroll around the map, as we planned to use Google Maps API to represent map functionality. Aside from the map, we intended to have a feature to show estimated distance between the user and the collar under the map. Finally, if the user is unable to use these location features to find their pet, they are able to play an alarm from the collar to pinpoint the pet's exact location.

Senior Design 2 Update:

Unfortunately, we were unable to master the Google Maps API that we intended to use for our location page. Therefore, we decided to instead show the collar's current coordinates using longitude and latitude values. Once the user gets these values, they can easily be copied and pasted into their preferred maps application to see the location of the collar. In addition, the user will connect to Bluetooth on this page before having the ability to use the alarm and location functionalities.

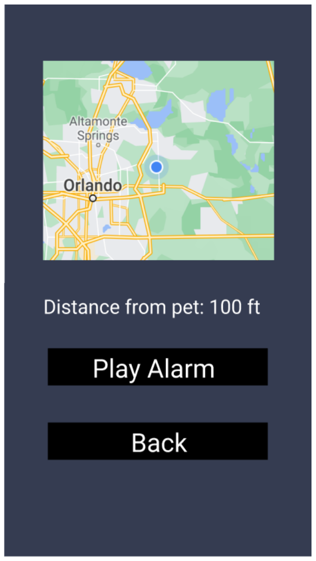


Figure 5.31
Location Page Mockup

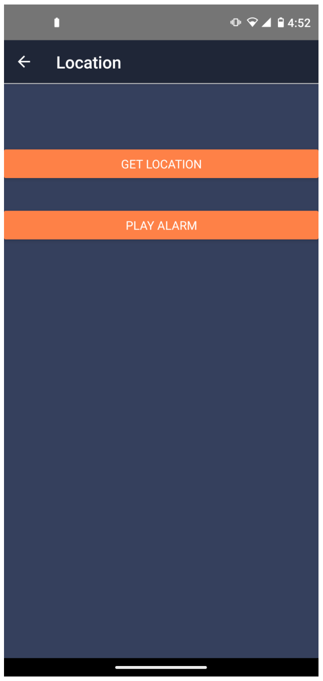


Figure 5.32
Location Page Screenshot

Bluetooth Page

This page was designed to control bluetooth connectivity between the user's phone through the Pet Pal companion application and the Pet Pal collar system. In our initial design, this page was crucial because without the connection to the collar, the health and location services pages would not provide accurate data. Initially, the user would enter the collar's bluetooth MAC address (which is written on the smart collar upon purchase or found in a user manual) in order to form a connection with the collar system. This process was intended to be done once however, as any time the user has bluetooth enabled inside of the application, the phone was supposed to always connect to the dog collar. If a user wished to prevent this feature or to enter another Pet Pal collar system to track instead, they would press the disconnect button and enter another bluetooth MAC address.

Senior Design 2 Update:

While developing the application, we realized that the bluetooth library we used (react-native-ble-plx) would be better implemented by connecting to bluetooth on each individual page. Therefore, this page does not exist in the final implementation.

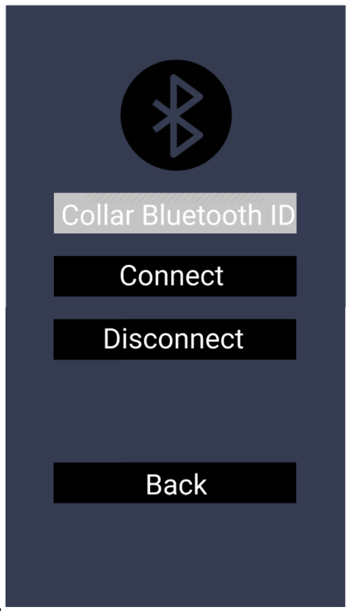


Figure 5.33
Bluetooth Page Mockup

Settings Page

The settings page was designed to be accessible from the home "landing page". Here, the user would have access to several buttons. The "Edit User Info" button would allow the user to edit registration items - such as their email or password. Once these changes were made, the application would go and change the database entry for that user. The "Edit Pet Info" button would have allowed the user to edit their list of pets and specify which pet is using the collar at the current moment. This section would have also let the user change specific parameters about their pet that are taken into account when the application logic decides the tolerable ranges for the temperature and heart rate sensors. The parameters included things such as breed, age, and weight. The "Notifications" button would take a user to a section of the application where they could choose to silence or turn-on push notifications sent to their phone. The "Speaker" button would have acted as a volume control page. Here they would be able to set the default alarm and calming sounds based on premade melodies. Here we had the potential to also implement a way to turn the speaker output volume up or down.

Senior Design 2 Update:

Unfortunately, due to time constraints, we were unable to implement a settings feature in the application. Therefore, this page does not appear in the final implementation.

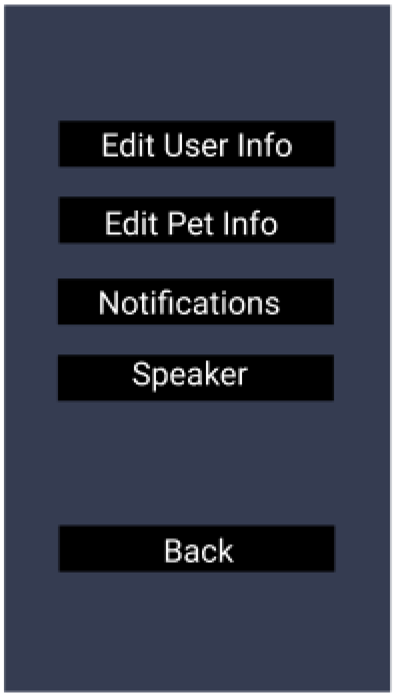


Figure 5.34
Settings Page Mockup

5.2.9 Version Control

In order to keep track of changes made to the application that we are developing, we used a GitHub repository. The GitHub repository was set to private to default, but we allow access to interested parties to view our code base. Each member working on the application side of this project had git downloaded on their local machine and a clone of the github repository on their local repository. This allowed each member to pull code from the GitHub and work on individual changes without interfering with the main code base. Once changes were made on the local machines, they were pushed back to the GitHub repository. GitHub allows for version control; meaning that changes that were made that in the end did not align with the final vision of the project we had the ability to revert to a previous version.

5.2.10 Testing

Every time that major changes were made to the code base, the code was regression tested. In order to do this efficiently, manual testing was implemented.

<u>Manual Testing</u>

We tested the use cases of the application through the perspective of the user. In other words, we went through all of the functionalities of the application as though we were a new user to test the usability of personalized data and to make sure that our sensor data worked correctly.

6. Testing & Overall Integration

In order to have a functional PCB, we must test our components and system extensively to make sure they operate properly and as expected. Individual testing of components will verify that they are operational and also allow us to figure out the proper configuration for that device individually. System level testing will ensure all the components operate together as desired. In addition to this, the configuration of the system will be able to be built off the results of individual testing. This process will take place by adding components to our testing development kit one-by-one. The system schematic will be the reference for testing our components. Our testing can also reveal shortcoming and errors in our schematic as well.

6.1 Bill of Materials

The first step in testing our components is to order the components themselves. To determine which parts we would order, we first looked at multiple products for each module and either decided on one part or ordered multiple parts for the same module. For the modules in which we couldn't decide which part was best, we decided to order multiple different parts and determine which is best after testing. Below is a list of the materials that we have ordered so far for testing and for our overall PCB design. Not included at the moment are any of the power components such as MOSFETs and Battery assembly.

Temperature Sensor			
Component	Amount	Cost	Date of Arrival
DS18B20	10	$10	04/09/21
Pulse Sensor			
Component	Amount	Cost	Date of Arrival
MAX30100	3	$30	04/10/21
SPEAKERS			
Component	Amount	Cost	Date of Arrival
Visaton K15S	3	$17	Summer
Bluetooth Module			
Component	Amount	Cost	Date of Arrival
RioRand Bluetooth Module	1	$15	Summer
Flora BlueFruit LE	1	$10	04/08/21
GPS Module			
Component	Amount	Cost	Date of Arrival
Flora Wearable Ultimate GPS	2	$40	04/09/21
Microcontroller for PCB Design			
Component	Amount	Cost	Date of Arrival
ATSAMG55J19A-AUT	10	$49	4/09-4/14
ATMEGA328PB-ANR	20	$100	Summer
Microcontroller Development Board			
Component	Amount	Cost	Date of Arrival
Arduino Uno Rev3	1	$22	4/7/21
Development Kit			
Component	Amount	Cost	Date of Arrival
Discovery Kit	1	Free	4/8/21
Hantek 2D42	1	Free	4/8/21
Power Components			
MOSFETs	3	$1.50	9/15/21
Batteries	1	$10	9/15/21

Table 6.1
Bill of Materials

6.2 System Testing

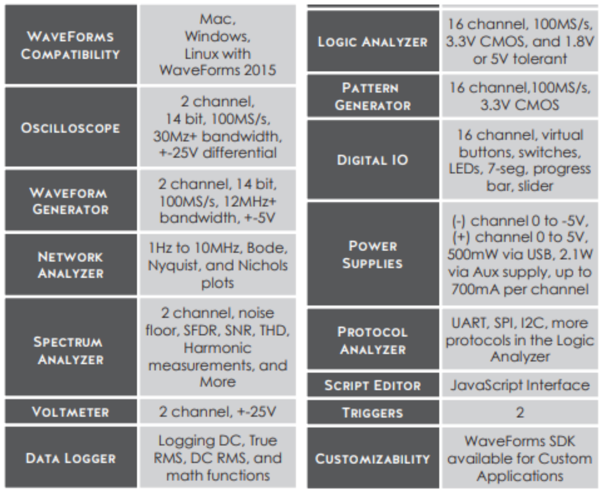
Before integrating our system together as a whole, we have to test all of the individual components using a breadboard to build the circuit, a power supply to power it, and an oscilloscope to analyze the output.

UCF has allowed us to rent a Digilent Ultra Analog Discovery 2 Bundle for the rest of Senior Design. This testing kit comes with the Analog Discovery 2, a breadboard adapter, a BNC adapter, an impedance analyzer, a breadboard breakout, a large breadboard kit, and test leads and probes.

6.3 Analog Discovery 2

The Analog Discovery 2 acts as a two-channel USB digital oscilloscope, a two-channel function generator, a single-channel voltmeter, a spectrum analyzer, digital bus analyzers (for SPI, I^2C, UART, and parallel), and two programmable power supplies among many other things. The breadboard breakout or breadboard adapter can be attached to the pins of the Analog Discovery 2 in order to supply power to a breadboard and give the user the option to construct circuits on it. These circuits can then be further analyzed using the impedance analyzer and the test leads and probes. In order to view output waveform readings from the test probes the Digilent Waveforms 2015 software is required to be downloaded on the computer you wish to receive testing information on.

The core of the Analog Discovery 2 is the Xilinx Spartan-6 FPGA. Upon start-up of the Waveforms software, the application automatically programs the FPGA with a configuration file(42). For a complete list of the tools and features that this kit offers, view the figure below.



WaveForms Compatibility	Mac, Windows, Linux with WaveForms 2015
Oscilloscope	2 channel, 14 bit, 100MS/s, 30Mz+ bandwidth, +-25V differential
Waveform Generator	2 channel, 14 bit, 100MS/s, 12MHz+ bandwidth, +-5V
Network Analyzer	1Hz to 10MHz, Bode, Nyquist, and Nichols plots
Spectrum Analyzer	2 channel, noise floor, SFDR, SNR, THD, Harmonic measurements, and More
Voltmeter	2 channel, +-25V
Data Logger	Logging DC, True RMS, DC RMS, and math functions
Logic Analyzer	16 channel, 100MS/s, 3.3V CMOS, and 1.8V or 5V tolerant
Pattern Generator	16 channel,100MS/s, 3.3V CMOS
Digital IO	16 channel, virtual buttons, switches, LEDs, 7-seg, progress bar, slider
Power Supplies	(-) channel 0 to -5V, (+) channel 0 to 5V, 500mW via USB, 2.1W via Aux supply, up to 700mA per channel
Protocol Analyzer	UART, SPI, I2C, more protocols in the Logic Analyzer
Script Editor	JavaScript Interface
Triggers	2
Customizability	WaveForms SDK available for Custom Applications

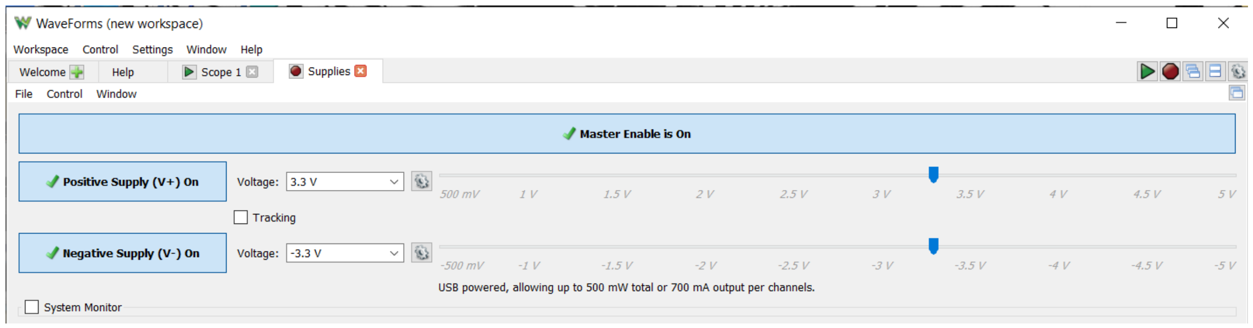
Figure 6.1
List of Analog Discovery 2 Features

6.4 WaveForms 2015

As mentioned above, this software is the virtual testbench environment for the Analog Discovery 2. The WaveForm's main window Welcome tab has buttons for each instrument available through the software: Scope (Oscilloscope), Wavegen (Arbitrary Waveform Generator), Supplies (Supplies and Reference Voltages), Meters (Voltmeters), Analyzer (Logic Analyzer), Patterns (Digital Pattern Generator), Static I/O (Static Digital Input/Output), Bode (Network Analyzer), Spectrum Analyzer and Script instruments [40].

6.5 BreadBoard Testing

We conducted all breadboard testing using the breadboard breakout attachment for the Analog Discovery. We connected it to a full size Digilent Breadboard and supplied power to it through the "Supplies" section of the WaveForms desktop application.


WaveForms (new workspace)
Workspace Control Settings Window Help
Welcome Help Scope 1 Supplies
File Control Window
500 mV 1 V 1.5 V 2 V 2.5 V 3 V 3.5 V 4 V 4.5 V 5 V
-500 mV -1 V -1.5 V -2 V -2.5 V -3 V -3.5 V -4 V -4.5 V -5 V

Master Enable is On

Positive Supply (V+) On Voltage: 3.3 V

Tracking

Negative Supply (V-) On Voltage: -3.3 V

USB powered, allowing up to 500 mW total or 700 mA output per channels.

System Monitor

Figure 6.2
Supplying Power Through Analog Discovery 2 with the Use of WaveForms

After this was completed, we were ready to begin testing the functionality of the sensors for our system individually.

6.5.1 Bluetooth Module

The adafruit Bluefruit LE module comes with a red LED light on board that flashes every second to demonstrate that it is receiving power correctly. Connecting the V+ pin (which we set to supply 3.3 Volts) to the 3.3 V pad and ground to ground we were able to see this flashing red LED. In order to avoid soldering any wires together, we used testing clips to form a temporary connection.

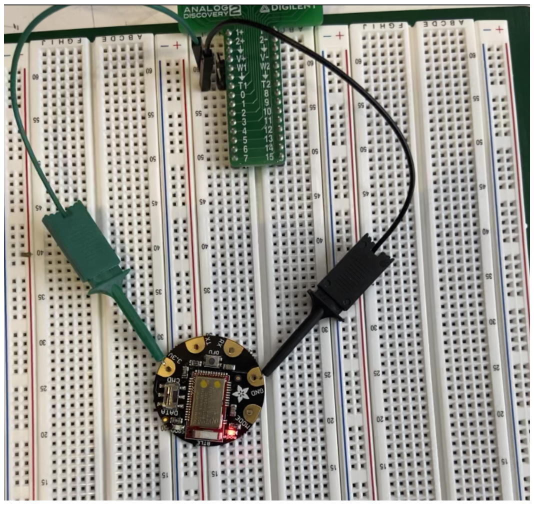


Figure 6.3
BlueFruit LE Showing Red Power LED

From this point, we needed to check if we could connect to it via bluetooth. In order to avoid making any embedded code to program UART communication using the TX and RX pins on the board, we simply downloaded the associated adafruit *Bluefruit Connect* application on our phones. Through this mobile application we were able to connect through bluetooth to the module and we saw the red LED disappear in place of a steady, blue LED. This demonstrated that the module worked as intended when it came to obtaining a bluetooth connection.

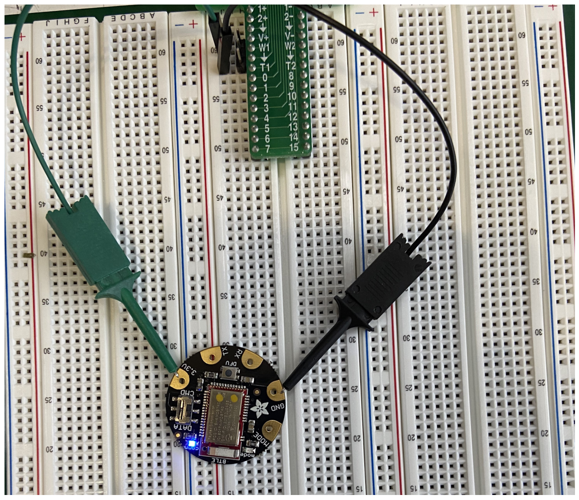


Figure 6.4
Bluefruit Module Showing the Blue Connection LED

It is worth noting that using the Bluefruit Connect application is only a temporary fix used for testing. When it came to the overall implementation of our system, we coded up our own communication method using the UART protocol.

6.5.2 GPS Module

We went about testing the GPS receiver in a similar fashion. We connected V+ to the 3.3 V pad and ground to ground and we were able to confirm that the GPS module on the board was getting power through the red LED. This red LED flashes once every second when the device is powered but has not received a satellite fix yet.

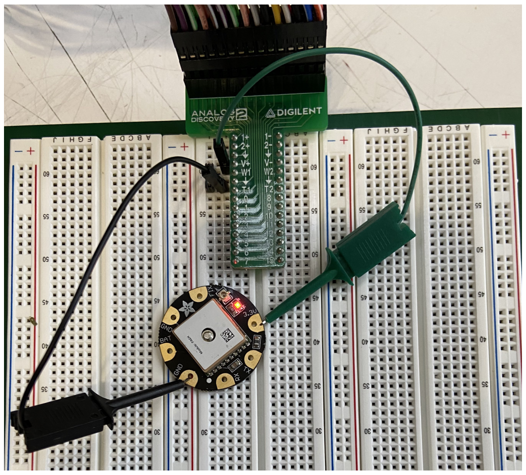


Figure 6.5
GPS Module with the Red Power LED

From here we wanted to see if the GPS could receive GPS data even if it had no significant value. In order to accomplish this we needed to use some embedded arduino code, hence we had to connect the Arduino Uno Rev 3 that we purchased specifically for testing purposes. Due to the need to upload embedded code to the arduino, this board was not able to be powered by the same breadboard. Instead, we connected the board to a computer using the USB-B port.

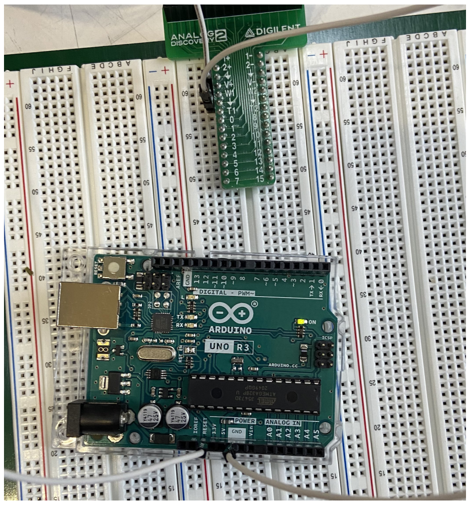


Figure 6.6
The Powered Arduino Uno Rev 3

We opened up the Arduino IDE and downloaded the Adafruit GPS library in order to interact with this GPS module. The library came with example code to perform what is called an “echo test”. Uploading this code to the arduino allows the arduino to make requests to the GPS module using its serial port (COM4) and the UART communication protocol. In order to do this, we connected TX on the module to RX on the board and RX to TX in the same manner. Once we uploaded and ran the program, we were able to see the raw GPS NMEA data (without a proper fix) through the serial monitor section of the Arduino IDE. This GPS data looked similar to the example found above in the GPS section of this document. However, this data that appeared on the Serial monitor page in no way reflected real GPS data. To make sure that the GPS module was able to receive valid GPS data we had to take the arduino and GPS receiver outside where the on-board ceramic antenna could get a clear view of the sky in order to get a fix of a satellite.

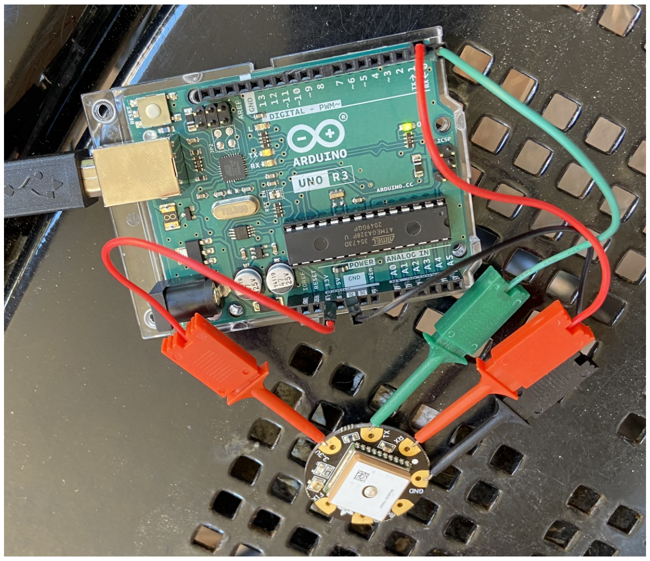


Figure 6.7
GPS Receiver Connected to the Arduino Uno (Fix Received)

Once the GPS module obtains a fix, the red LED ceases to flash once every second and instead reverts to flashing once every five seconds. After the connection with the satellite has been established, going indoors will not break the fix, however, we cannot ensure that the GPS data that is being received indoors is accurate. Once we uploaded and ran the arduino script once again, we were able to see valid GPS data coming in at a much faster rate than when we attempted this echo test without a GPS fix. This proved to us that the GPS module and receiver board were working as intended.

6.5.3 Speakers

In order to test that the 8-ohm speakers were functional, we connected the red cable of the speaker to the output pin of the Arduino Uno we chose to designate - in this case, digital pin 8 - and then connected the black cable to ground. From here we decided to test if the speakers worked by playing a melody whose notes were generated using the tone() command.

The tone() command works by generating a square wave of the specified frequency (and 50% duty cycle) on a pin(41). Important things to note are that the use of this function will interfere with PWM output on pins 3 and 11 of the Arduino Uno and that it is not possible to generate tones lower than 31 Hz. This tone function has two appropriate syntax:

1. <u>tone(pin, frequency)</u>

   This function will play the tone at the designated frequency at the desired arduino output pin until a call to noTone() is sent.

2. <u>tone(pin, frequency, duration)</u>

   This function will perform the same action as the function above, with the difference being that the time is specified in milliseconds in the duration parameter. This eliminates the need to call the noTone() function in the code.

Upon uploading sample code to play a melody, we were able to hear the melody played through the 8-ohm speaker shown in the figure below. It is important to note that if we have the desire to play notes at a louder volume, we must add a power amplifier circuit. In addition, this is not the speaker that we used in the final product design due to its bulky dimensions.

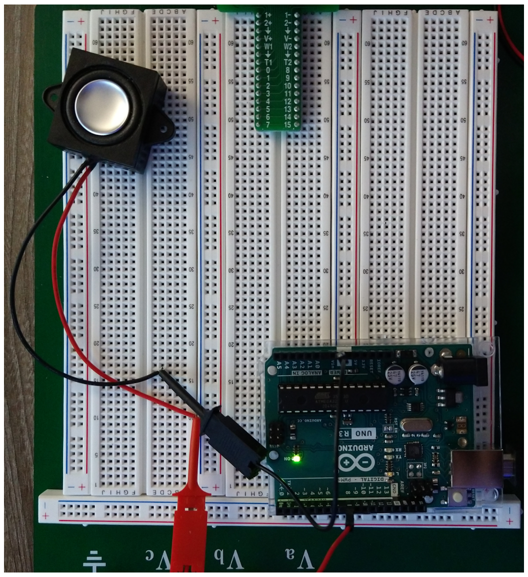


Figure 6.8
8-ohm Speakers Connected to the Arduino Uno Rev 3

6.5.4 Temperature Probe

The connection to the Arduino Uno went as follows: the ground pin of the sensor goes to ground, the data pin and VCC are connected in parallel by a 4.7 kΩ pull-up resistor, with the VCC pin being connected to the 3.3 V pin on the arduino.

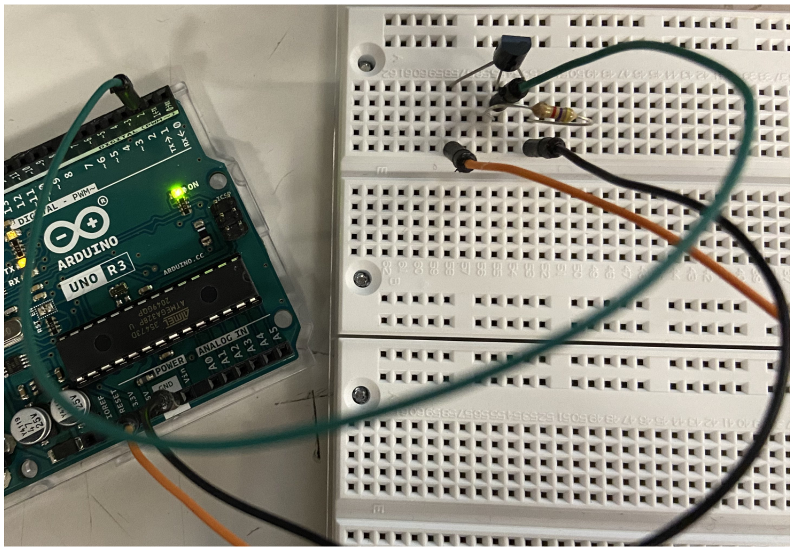
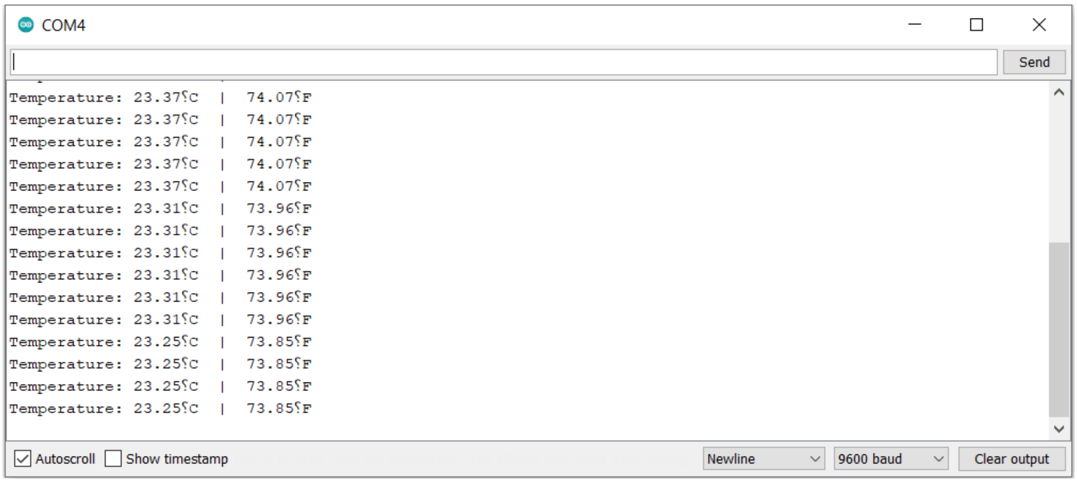


Figure 6.9
Temperature Sensor Connected to the Arduino Uno and Pull-up Resistor

Once this connection was made, we had to install the DS1820 temperature sensor library on the arduino IDE. This came with example code that we were able to run in order to test the functionality of the sensor. After uploading this code to the temperature sensor, we were able to check the serial monitor of the arduino to see if we have received valid data. If invalid or no data was being received, we would see -127° C and -260.6° F. Below is a figure of the data received.



```
Temperature: 23.37°C  |  74.07°F
Temperature: 23.37°C  |  74.07°F
Temperature: 23.37°C  |  74.07°F
Temperature: 23.37°C  |  74.07°F
Temperature: 23.37°C  |  74.07°F
Temperature: 23.31°C  |  73.96°F
Temperature: 23.31°C  |  73.96°F
Temperature: 23.31°C  |  73.96°F
Temperature: 23.31°C  |  73.96°F
Temperature: 23.31°C  |  73.96°F
Temperature: 23.31°C  |  73.96°F
Temperature: 23.25°C  |  73.85°F
Temperature: 23.25°C  |  73.85°F
Temperature: 23.25°C  |  73.85°F
Temperature: 23.25°C  |  73.85°F
```

Figure 6.10
Temperature Probe readings on Arduino IDE Serial Monitor

It was worth noting that the "�" character is not an error but simply the Arduino IDE software failing to recognize the degrees symbol. In addition, this probe was not included in the final design as it was later discovered that the heart rate sensor also reads temperature.

6.5.5 Heart Rate Sensor (Pulse Oximeter)

We hooked the heart rate sensor up to the Arduino Uno in this manner: we connected the Vin pin of the sensor to the 3.3 V pin of the arduino, the SCL pin to A5, the SDA pin to A4, and the ground to ground. However, once we made all the proper connections, a red power LED is supposed to turn on the MAX30100 in order to show us that the sensor is ready to receive data. We attempted this same connection process with our two spare sensors, but they all had the same issue.

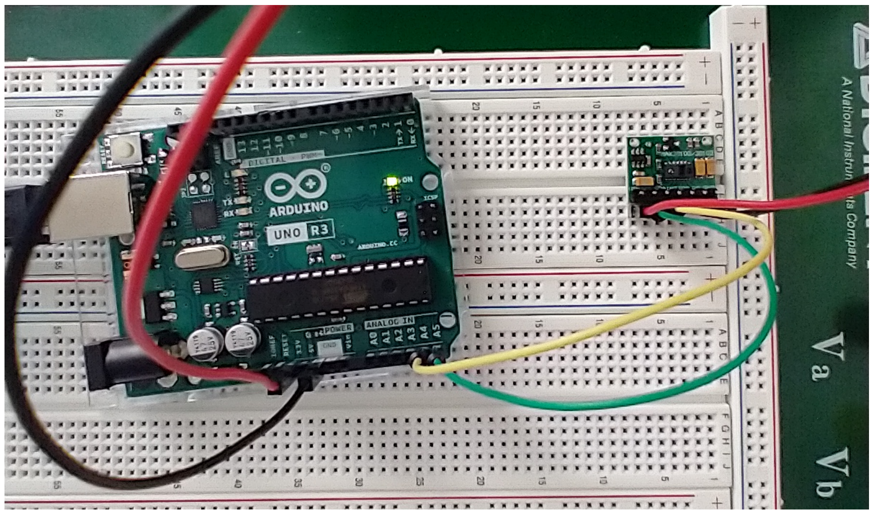
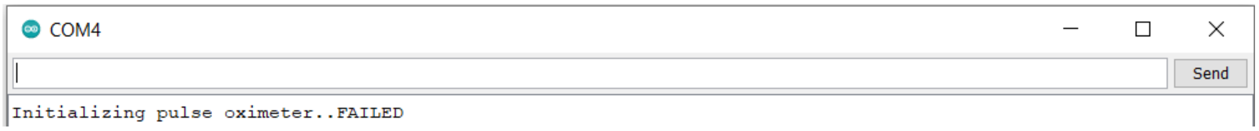


Figure 6.11
MAX30100 Heart and Pulse Sensor Board Connected to the Arduino Uno

Regardless of this, we downloaded a MAX30100 Arduino library that came with example arduino code that would measure heart rate and SpO2 levels. Upon uploading this code, we continued to receive error messages that the MAX30100 failed to initialize.



```
Initializing pulse oximeter..FAILED
```

Figure 6.12
Serial Monitor Failure Message

Upon further research, we found out that the issue lies with how the board is designed. We must connect the three pull-up resistors found on the board to the 3.3 V based network.

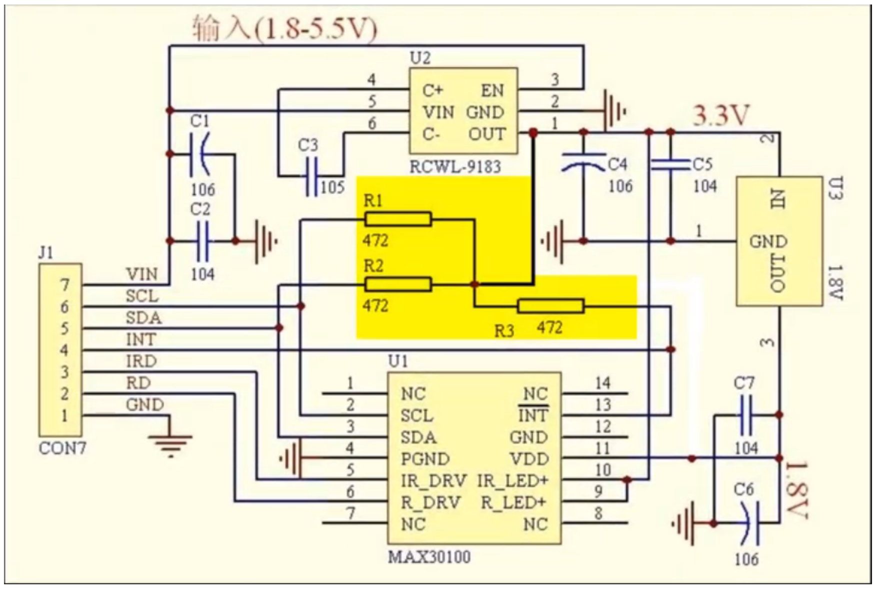


Figure 6.13
The Three Resistors to be Connected to the 3.3 V Network

There are two ways to solve this:

1. <u>Remove the three on-board resistors</u>

   This method involves pulling the three surface mounted resistors. If using the 3.3 Volt connection pin on the arduino, two external 4.7 kΩ resistors must be used to pull-up to the network. If using the 5 Volt pin connection, three resistors must be used instead as shown in the figure below.

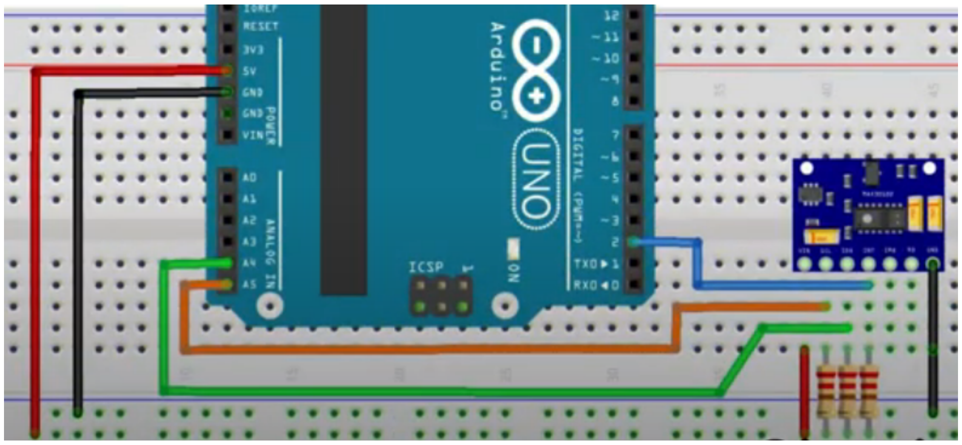


Figure 6.14
New Connection with External 4.7 kΩ Resistors

2. Form a connection between the 3.3 V pin on the regulator and the pull-up resistor

   You can solder a connection to the 3.3 V voltage regulator pin that is initially unoccupied on the board. When you do this, you must cut the other track in order for the circuitry to still work properly. This approach does not require any external resistors to be added.

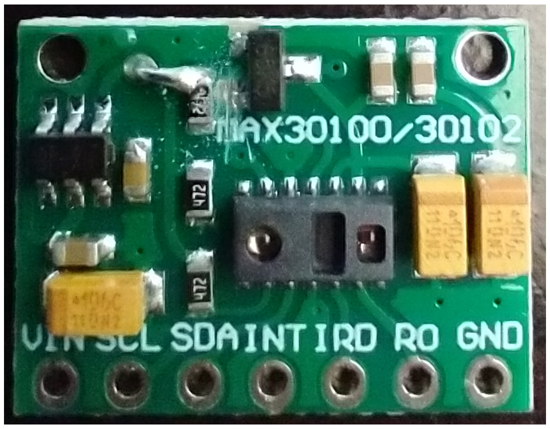


Figure 6.15
Soldered Connection Between Voltage Regulator and On-board Resistors

Unfortunately, neither of these approaches were deemed to be successful, as the red LED never powered on. We attributed this to faulty sensors or incorrect board circuitry, and we have since ordered a different model MAX30100 sensor board to finalize our testing.

In the end we ended up switching to a newer model of MAX30100: the MAX30102. This sensor worked effortlessly once we soldered header pins to the Vin, GND, SDA, and SCL lines. We also discovered during initial testing that this sensor

6.6 System Schematic

This system schematic will act as the foundation for the final PCB design which will be done in the Autocad EAGLE software. It demonstrates the proper connections between the microcontroller and all of the peripherals for our system. In addition to the current design, there will be a second page with all the necessary power circuitry included. Once this design is complete with all the necessary components and power circuitry, a PCB design can be made.

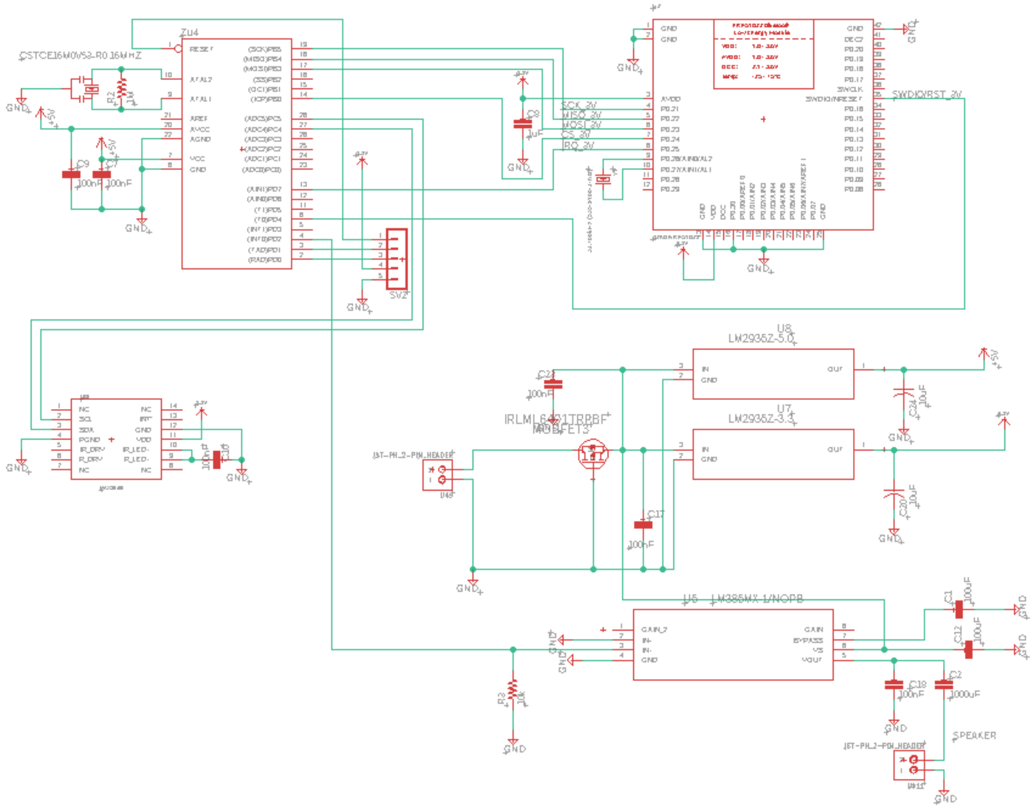


Figure 6.16
System Schematic

The system schematic includes the ATMEGA328P microcontroller, the MAX30100 pulse monitor, the Flora BlueFruit LE bluetooth module. The power system is also present on this sheet of the schematic but is discussed more in depth later in this document. There are header pins included in the design as the Flora Wearable Ultimate GPS module and the battery are off-board. This design is what our system and PCB will be based on. In addition to this, the microcontroller requires 5 volts to operate as opposed to the 3.3 volts which the other components operate off of. This was reflected in our power diagram as both a 5 volt regulator and 3.3 volt regulator are present in the design.

6.6.1 Passive Component Purposes

The passive components in the design have many purposes. The devices in this schematic need certain conditions in order to ensure proper and effective usage. Resistors connected to certain data lines are pull-up resistors. Pull-up resistors, as their nomenclature suggests, pull up the data line to a higher voltage. One purpose of this is to mitigate the effect of noise. By pulling the data line to a higher voltage, it greatly reduces the chance for noise to falsely cross the low voltage threshold. This can cause miscommunication between the master and slave devices. Pull-up resistors are generally only used in the I2C communication protocol. In essence, they can be considered an enable or disable depending on if the data line is high or low. The capacitors in the schematic have two express purposes. The first is to filter out noise, and the second is to keep the input voltage stable. Similar to a low or high pass filter, these capacitors filter out noise by bypassing it to ground. Commonly called, Bypass Capacitors, these reduce noise as well as mitigating spikes in the power supply lines. Concretely it takes the small-signal AC voltage from the DC power and shunts it to ground. The other type of capacitor is the Decoupling Capacitors. These capacitors keep the input voltage to a component stable in two ways. First, if the input voltage drops too much, the stored power in the capacitor will be used to keep the input signal stable. Second, if the input voltage increases too much, the capacitor will absorb the excess power and store it, keeping the input voltage stable.

6.7 Power Schematic

The schematic below indicates the power system which will be used to supply power for the entire system which is a part of the main system schematic. The schematic starts from an input voltage source which are replaceable batteries connected to a polarity protection. The polarity protection will be wired to both 3.3 and 5 volt regulators which produces an output voltage of 3.3 volts and 5 volts respectively. Most of the devices, such as heart rate monitor, GPS module, and Bluetooth module need an input power of 3.3 volts to supply. On the other hand, since our audio circuit needs an input voltage higher than 4 volts, we build a non-inverting amplifier. The input of the non-inverting amplifier is connected directly to the output of the battery after the polarity protection and the output of the amplifier is connected to the input of the audio circuit.

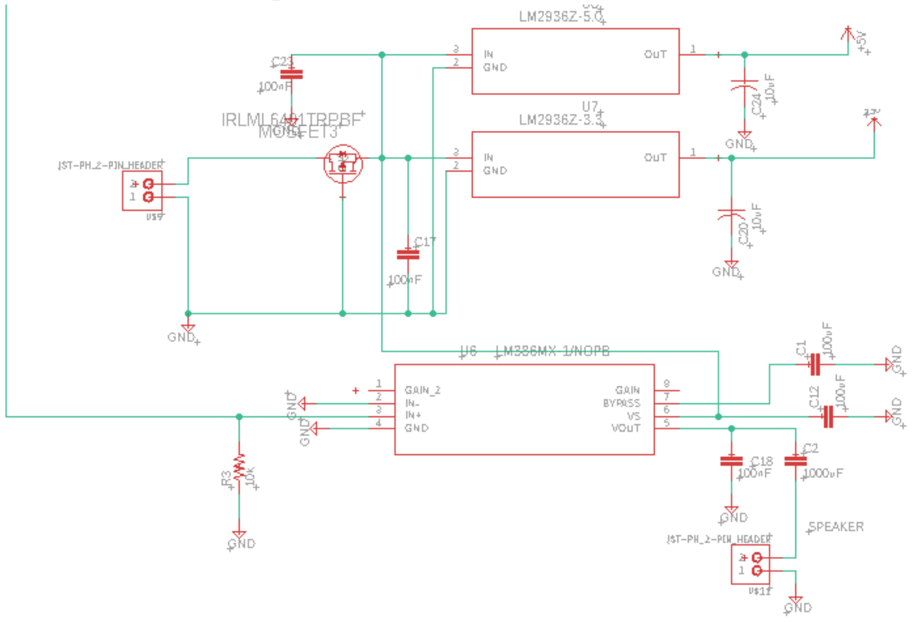


Figure 6.17
Power Schematic

In the schematic, there are seven resistors, and four capacitors are used. In addition, we will use a P channel IRLML6402PBF MOSFET to build the polarity protection. For the voltage regulator, we will use the SPX3819R2-L/TR 8 pins chip, and for the non-inverting amplifier, we will use the TL084ACD14 CHIP. Moreover, in order to build an effective audio circuit, the LM386 operational amplifier which is known as a common op-amp for designing audio circuits is used. These chosen chips have small sizes so that the final PCB can appropriately fit the collar.

6.8 PCB Layout

The PCB layout is essentially the core of our design process. From this design, our board was built and implemented. Thus, it is of paramount importance that our breadboard testing is extensive and accurate. The components on the PCB cannot be changed or moved very easily, so it is important to make sure everything is correct on the first attempt. Likely, there will be more than one iteration of the PCB design before it is finalized in order to allow for improvements.

6.8.1 Considerations

The PCB will need to have enough space to have all our components and still fit in a rather small space on the width of the collar. In order to achieve this and make the process as efficient as possible, there will be a couple necessary features of the board. A multiple layer board was considered to save space. This would have allowed us to use less space laterally. In addition to this, the multiple layer feature would have allowed us to use a technique called Manhattan Routing. Manhattan routing makes use of extra layers when routing. Basically, two layers of the PCB can be used to route either vertically or horizontally, but not both on the same layer. This allows us to save space and avoid common problems with our routing. We opted to not use a multiple layer board for multiple reasons, namely, cost increases with the amount of layers of the board and with the changes made to allow for components to be off-board, it was considered to be unnecessary. The last consideration is the heat generated by the components. As previously mentioned, we did not want the components to interfere with the reading of the temperature sensor. By placing the temperature sensor far from the other components, this problem was mitigated. The heart rate and temperature sensor was also implemented in such a way so as to make contact with the skin in order to give an accurate reading. This means that the heart rate and temperature sensor was placed on the bottom side of the board. The board shall need to be double sided in order to compensate for this.

6.8.2 Board Materials

PCBs are generally made from flat laminated composite material which is non-conductive with layers of copper or, occasionally, other similar conductive materials embedded within it or on the surface. These copper layers can be as simple as one or two layers, to up to over 50 layers in high-end, complex applications. From these copper layers, the traces are etched out and components will be placed on the corresponding pads that are created. The green color that is taken on by most PCBs is a result of Soldermask. Soldermask is the epoxy coating that covers most of the board. It exists to protect the external circuits on the board. These circuits made of copper, if left exposed, will oxidize over time and ruin the board. Lastly, in order to protect any more potentially vulnerable r exposed spots, such as the copper from vias in the board, a metallic coating, or finish, is applied.(48)

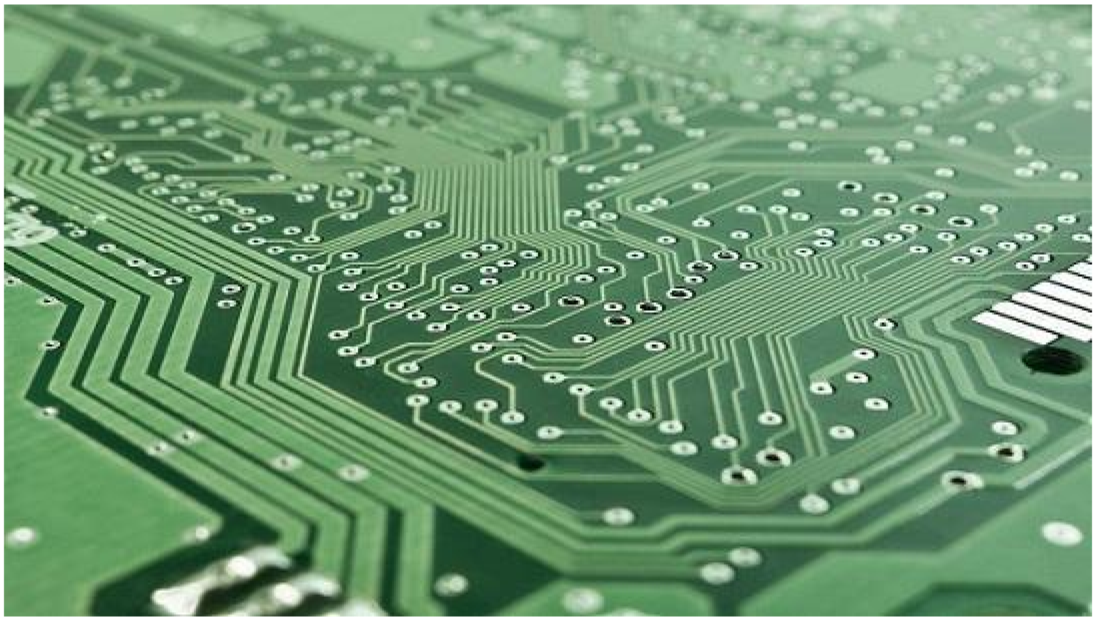


Figure 6.18
Traces and Pads on a Bare PCB

6.8.3 Board Layering

PCB technology has evolved to meet the current and growing needs in the electronic industry. Simple 2-layer boards are now much less common, and it is not uncommon for boards to have anywhere from 4 all the way up to 16 layers for complex designs. This increase in layers has many benefits including reducing cross-talk, eliminating electromagnetic interference, and distributing power more efficiently(47). In our design, a multi-layer board would perform best given our constraints and chosen components. There are many reasons for this. In a design with multiple layers, there can exist a ground plane as well as a power plane. This allows for more space on the top or bottom layer for the placing of components. This power plane can also have more than one voltage level on it to distribute multiple power levels to devices. This is perfect for our design as we need to have both 1.8 volts and 3.3 volts for powering certain components. A multi-layer design will also allow space for routing layers as briefly discussed.

6.8.4 Routing Considerations

Routing is important for more reasons than one might think. Depending on the design as well as the importance of various aspects of the design itself. For instance, in a particular design, the signal integrity or the power integrity might be paramount for that design. As such, the routing plays a more important role than just for saving space. This is not to make the claim that space saving is unimportant. In our design, saving space is perhaps one of the most important considerations. In order to achieve this goal of minimizing the space taken by the PCB, following some of the core rules of routing will be extraordinarily helpful. First and foremost, keeping the traces short and direct will minimize the space taken. In addition, any latency will be much smaller. Latency on a PCB is generally called Signal Propagation Delay. Signal Propagation Delay is generally an issue in extremely high-speed designs, but if not taken seriously in our design could cause issues.The traces connected to the passive components on the board should also be short and direct as well. In some cases, it will be quite important to keep these passive components close to the integrated circuits on the board. This can be best seen in decoupling capacitors. These must be close to the pins of the ICs in order to serve their function of reducing ripple voltage from the supply of the IC. The farther away these are, the parasitic inductance of the lines increases and the ripple voltage will increase.

7. Administration

There are two major elements to keep track of in the administration of this project. The first being the budget and financing, and the second being the timeline for this project. The budget and associated costs of the project must be kept track of in order to ensure proper and fair division of cost among the team members. Inevitably, the contents of these tables will grow as issues and unforeseen circumstances present themselves. The timeline is divided into two sections as is this project. The timeline for Senior Design 1 was determined at the beginning of the spring semester. This allowed ample time for scheduling and allocating tasks in an appropriate manner. The timeline for Senior Design 2 was determined during the beginning of the fall semester as tasks became more clearly defined.

7.1 Budget and Financing

This section below is for the purpose of keeping track of our expenses and maintaining a budget when it comes to designing and manufacturing our smart collar system. The table below is a comprehensive list of all the costs associated with this project. The budget also includes many of the unforeseen expenses such as: receiving faulty parts, changing our decision on a peripheral, and needing expedited shipping. The total budget will be split equally amongst the four members of the Pet Pal development team.

Total Budget	$1000
Number of members	4
Project expenses per member	$250

	Items	**Quantity**	**Price per Unit**	**Total Cost**
1	Battery	12	$1.69	$20.32
2	MOSFET	2	$0.48	$0.96
3	Operational Amplifier	4	$1.55	$6.20
4	Audio Amplifier	4	$1.36	$5.44
5	Voltage Regulator	4	$0.68	$2.72
6	Heart Rate IC	2	$6	$12
7	Pulse Oximeter	2	$7.82	$15.64
8	Bluetooth IC	2	$4.95	$9.90
9	Bluetooth Module	2	$6.10	$12.20
10	GPS Module	2	$24.95	$49.90
11	Arduino Uno	2	$19.55	$39.10
12	PCB	3	$12	$186.00
13	3D Print Case	3	$0.75	$2.25
14	Professional Case	1	$42.08	$42.08
Total Cost				**$404.71**

Table 7.1
Estimated Budget and Financing

7.2 Project Milestones

The table below dictates a prospective timeline for keeping us on pace for completing this project between both Senior Design semesters. This includes the tasks and weeks for both semesters.

Task #	Task for Senior Design 1	Week Timeline (Both Semesters)
1	D&C Version 2	5
2	Order initial components	6
3	Configure sensors (temperature & heart rate) to arduino microcontroller	7-8
4	Configure GPS module with arduino	7-9
5	Configure speaker system and code desired notifications	10-11
6	60 Page Report	12
7	Determine and implement power supply (possible solar cells?)	10-13
8	Connect components to wireless network (for app integration)	13
9	Design and order PCB	13-15
10	100 Page Report	14
11	Code android application along with data backend	12-SD2
12	120 Page Final Document	16

Table 7.2
Project Milestones for Senior Design 1

Task #	**Task for Senior Design 2**	**Week Timeline (Both Semesters)**
13	Redesign project with constructed PCB	17-20
14	Order Components	21-23
14	CDR Presentation	23
15	Test heart rate and temperature sensor	24-25
16	Bluetooth completed with App communication	25-26
17	GPS Completed with App communication	26-27
18	Test the Power System	24-27
17	Mid Demo	27
18	Testing application with collar system	27-28
19	Overall System Testing	29
20	8 Pages Conference Paper and Committee Form	29
21	Build housing for components and attach to collar	30
22	Final Presentation	30

Table 7.3
Project Milestones for Senior Design 2

8. Conclusion

Pet Pal is a smart pet collar that is not only low cost but also brings convenience to the pet owners as well as their pets. Pet Pal combines multiple functions, such as GPS tracker, heart rate and temperature monitor, and sound alerts. Although Pet Pal has a lot of features, our group has meticulously chosen every single part inside Pet Pal just to confidently ensure that Pel Pal is lightweight and has a portable size. In addition, we also write an application and publish it on Android's store so that pet owners can easily gain access to the collar. From an Android device, pet owners can keep track of their pets' heart rate and temperature. With the GPS tracker, the pet owners no longer have to worry of losing their pets because the application will display location data for the pet at any given moment. The sound alerts will allow the pet to be found when the owner cannot locate them.

While working on our product, we found it would be more convenient for the users by using rechargeable and easily accessible batteries for our product. It is also possible for the pet owners to replace the battery without taking off the collar and it does not take time to wait for the collar to be charged. Our product has a polarity protection, for that reason, inserting the wrong battery's polarities will not be able to destroy the components. Most of the parts used inside Pet Pal require a power supply of 3.3 volts which makes it much simpler to design since we just need one supply voltage for a majority of the components and this also reduces the total weight of our product. The sound quality is definitely a feature that we want to improve later. Friendly-user application is a target that we have achieved through this project. The pet owner needs to have an email in order to sign up for an account.

Our initial budget is $1,000, this means each member willingly contributes $250 in order to complete the project. In Senior Design 1, we spent about $200 in total to buy some components, such as heart rate monitor, GPS module, temperature monitor, and microcontroller, in order to be able to test the capability of those components. There is also a discovery kit which is provided by University of Central Florida. In addition, our teammates have some available connection wires as well as speakers. In Senior Design 2, we have spent $204 to buy extra components to improve our final product. We spent $404 in total for our project which is only half of our budget.

Even Though we are suffering from a pandemic which has the ability to shut down businesses and schools in the whole world, we are trying our best to communicate, exchange information, and work together in order to create a complete product. We have strict schedules for every week that we follow. We only meet face to face twice during the semester. The first time we met was to test the components together, however, we did not have enough equipment, so we had to schedule another meeting. Once we got permission to enter the campus lab, we met again and finished our testing product. During our second semester, we met consistently on a week by week basis. Once we had completed the PCB design, we acquired all the components necessary for mounting the board.

However, due to some supply chain issues and shipping delays, components took longer to ship and some components were difficult to find. This required a redesign of the board in order to accommodate for these issues. The surface mount components on the board were mounted by Quality Manufacturing Services with a few through-hole components being mounted ourselves. During this process, the embedded programming was completed using the Arduino Uno Rev3 board and the mobile application was designed and completed. We also created a casing for the board and the components to house the system and once this was done and testing was completed our product was finished.

Pet Pal is not just a product that we create in order to graduate. To some of us, Pet Pal is the very first big project that we have ever worked on. We are a group of four, and this project requires a lot of effort to do research and distribute work evenly every week. However, the final result we get from the Pet Pal is worthy. Beside creating a complete product, we have acquired more skills as well as knowledge about writing software, and designing circuits.

Our priority is first, to make the collar functionable, and second, to make our product practical for daily use. It is important to us that our smart pet collar can surely help to improve the living condition of the pet owners and their pets. We want people to use Pet Pal as a device that can help them to save their time and carefully take care of their pets at the same time. Although our final product is not perfect and there is no doubt that it will have some minor failures, we believe that we can improve our product in the future by spending more time doing research and using advanced technology.

Works Cited

(1) Atkins C, Bonagura J, Ettinger S, et al. Guidelines for the diagnosis and treatment of canine chronic valvular heart disease. *J Vet Intern Med*. 2009; 23(6):1142-1150.

(2) Ineson, Deanna L., Lisa M. Freeman, and John E. Rush. "Clinical and laboratory findings and survival time associated with cardiac cachexia in dogs with congestive heart failure." *Journal of veterinary internal medicine* 33.5 (2019): 1902-1908.

(3) Staff, AKC. "Fever in Dogs: Causes, Symptoms & Treatment." *American Kennel Club*, American Kennel Club, 4 Mar. 2016, www.akc.org/expert-advice/health/dog-fever-and-temperature/.

(4) "HC 06 Bluetooth Module Pinout, Features & Datasheet." *Components101*, components101.com/wireless/hc-06-bluetooth-module-pinout-datasheet. Accessed 2 Mar. 2021.

(5) "RioRand® Bluetooth 4.0 BLE Low Energy/Power RF SOC Transceiver Smart Hardware Serial Module 2.4GHz Bridge/Direct-Driven Mode - CC2540 IBeacon - 24*14mm - For Apple/Android - Circuit Boards - Electronics." *RioRand*, www.riorand.com/electronics/circuit-boards/riorandr-bluetooth-4-0-ble-low-energy-power-rf-soc-transceiver-smart-hardware-serial-module-2-4ghz-bridge-direct-driven-mode-cc2540-ibeacon-24-14mm-for-apple-android.html. Accessed 3 Mar. 2021.

(6) "Lifetime - Classic Collar." Tuff Pupper, tuffpupper.com/products/lifetime-classic-dog-collar. Accessed 17 Mar. 2021.

(7) "BLUEBERRY PET Classic Solid Nylon Dog Collar." *Chewy.Com*, www.chewy.com/blueberry-pet-classic-solid-nylon-dog/dp/135248?utm_source=partnerize&utm_medium=affiliates&utm_campaign=1011l10259&utm_content=0&clickref=1011lf8uSjN7&utm_term=1011lf8uSjN7. Accessed 17 Mar. 2021.

(8) Beck, Alan M., and N. Marshall Meyers. "The pet owner experience." *Allergy and Asthma Proceedings*. Vol. 8. No. 3. OceanSide Publications, 1987.

(9) Sauser, Brian J., Richard R. Reilly, and Aaron J. Shenhar. "Why projects fail? How contingency theory can provide new insights–A comparative analysis of NASA's Mars Climate Orbiter loss." *International Journal of Project Management* 27.7 (2009): 665-679.

(10) "Lithium-Ion Battery." *Clean Energy Institute*, 25 Sept. 2020, www.cei.washington.edu/education/science-of-solar/battery-technology/.

(11) "How Does a Lithium-Ion Battery Work?" *Energy.gov*, www.energy.gov/eere/articles/how-does-lithium-ion-battery-work.

(12) "Smart Dog Collar - Heart Rate Monitor & More: Petpace." *PetPace Smart Collar*, 29 July 2020, petpace.com/.

(13) Dogcollarsbyfi. "GPS Dog Tracker & Activity Monitor." *Fi Smart Dog Collar*, tryfi.com/.

(14) Committee. *ISO/IEC 9899*. 6 May 2005.

(15) "Fast, Accurate, Reliable GPS Dog Collar: LINK AKC." *Link My Pet*, www.linkmypet.com/how-it-works.

(16) *Sewing with Conductive Thread*, learn.sparkfun.com/tutorial/sewing-with-conductive-thread.

(17) “MAX86141 Best-in-Class Optical Pulse Oximeter and Heart-Rate Sensor for Wearable Health - Maxim Integrated.” *In*, www.maximintegrated.com/en/products/interface/sensor-interface/MAX86141.html.
(18) “MAX3102 -- High-Sensitivity Pulse Oximeter and Heart-Rate Sensor for Wearable Health.” www.maximintegrated.com.
(19) Aleator777, and Instructables. “Intro to GPS With Microcontrollers.” *Instructables*, Instructables, 10 Oct. 2017, www.instructables.com/Intro-to-GPS-with-Microcontrollers/.
(20) “NMEA 0183.” *Wikipedia*, Wikimedia Foundation, 11 Mar. 2021, en.wikipedia.org/wiki/NMEA_0183.
(21) “TinyGPS++ : Arduiniana.” *TinyGPS++ | Arduiniana*, arduiniana.org/libraries/tinygpsplus/.
(22) “Location : Android Things : Android Developers.” *Android Developers*, developer.android.com/things/sdk/drivers/location.
(23) “Add Maps : Android Developers.” *Android Developers*, developer.android.com/training/maps.
(24) Ada, Lady. “Adafruit Ultimate GPS Featherwing.” Adafruit Learning System , 29 June 2020.
(25) Stern, Becky. “Flora Wearable GPS.” *Adafruit Learning System*, learn.adafruit.com/flora-wearable-gps/hook-up-gps.
(26) Pelayo, Roland. “Creating an Arduino Bluetooth Serial Interface.” *Microcontroller Tutorials*, Microcontroller Tutorials, 23 Apr. 2019, www.teachmemicro.com/arduino-bluetooth/.
(27) “MongoDB Realm Android SDK¶.” *MongoDB Realm Android SDK - MongoDB Realm*, docs.mongodb.com/realm/sdk/android/.
(28) *Java JDBC API*, docs.oracle.com/javase/8/docs/technotes/guides/jdbc/.
(29) “IEEE 802.15.1-2002 - IEEE Standard for Telecommunications and Information Exchange Between Systems - LAN/MAN - Specific Requirements - Part 15: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Wireless Personal Area Networks (WPANs).” *IEEE SA - The IEEE Standards Association - Home*, standards.ieee.org/standard/802_15_1-2002.html.
(30)*Java SE Technologies - Database*, www.oracle.com/java/technologies/javase/javase-tech-database.html.
(31) Person. “UART: A Hardware Communication Protocol Understanding Universal Asynchronous Receiver/Transmitter.” *UART: A Hardware Communication Protocol Understanding Universal Asynchronous Receiver/Transmitter | Analog Devices*, www.analog.com/en/analog-dialogue/articles/uart-a-hardware-communication-protocol.html.
(32) “Arduino IDE Review: Pricing, Pros, Cons & Features.” *CompareCamp.com*, 15 Mar. 2019, comparecamp.com/arduino-ide-review-pricing-pros-cons-features/.
(33) *The C4 Model for Visualising Software Architecture*, c4model.com/.
(34) “How Much Does IAR Cost?” *EmbeddedRelated.com | Embedded Systems*, www.embeddedrelated.com/showthread/msp430/39036-1.php.
(35) “Bluetooth Overview : Android Developers.” *Android Developers*, developer.android.com/guide/topics/connectivity/bluetooth.

(36) “Download Android Studio and SDK Tools : Android Developers.” *Android Developers*, developer.android.com/studio.

(37) Rozwadowski, Wojciech. “Pros & Cons of Flutter Mobile Development • Future Mind.” • *Future Mind*, www.futuremind.com/blog/pros-cons-flutter-mobile-development.

(38) Dorner, William. “Syllabus Statements.” *University of Central Florida Faculty Center*, 20 Nov. 2020, fctl.ucf.edu/teaching-resources/course-design/syllabus-statements/.

(39) Newton, Alex, et al. “Interfacing MAX30100 Pulse Oximeter Sensor with Arduino.” *How To Electronics*, 10 May 2020, how2electronics.com/interfacing-max30100-pulse-oximeter-sensor-arduino/.

(40) K, Sam. “WaveForms Reference Manual.” *WaveForms Reference Manual - Digilent Reference*, reference.digilentinc.com/reference/software/waveforms/waveforms-3/reference-manual.

(41) “Tone().” *Tone() - Arduino Reference*, www.arduino.cc/reference/en/language/functions/advanced-io/tone/.

(42) K, Sam. “Analog Discovery 2 Reference Manual.” Analog Discovery 2 Reference Manual - Digilent Reference, reference.digilentinc.com/reference/instrumentation/analog-discovery-2/reference-manual?redirect=1#figure_2.

(43) “Arduino Uno Rev3.” Arduino Uno Rev3 | Arduino Official Store, s tore.arduino.cc/usa/arduino-uno-rev3.

(44) “MSP-EXP430G2ET.” *MSP-EXP430G2ET Development Kit | TI.com*, www.ti.com/tool/MSP-EXP430G2ET.

(45) “Animal Welfare Act.” *Animal Welfare Institute*, awionline.org/content/animal-welfare-act.

(46) *U.S.C. Title 7 - AGRICULTURE*, www.govinfo.gov/content/pkg/USCODE-2015-title7/html/USCODE-2015-title7-chap54.htm.

(47) Solutions, Cadence PCB, et al. “PCB Layers Explained: Multilayer Boards and Stackup Rules.” *Cadence*, 29 June 2020, resources.pcb.cadence.com/blog/2019-pcb-layers-explained-multilayer-boards-and-stackup-rules#:~:text=The%20number%20of%20layers%20used,4%E2%80%94as%20the%20signal%20layers.

(48) Sachin. “Printed Circuit Board (PCB) Materials.” *Printed Circuits LLC*, Printed Circuits LLC, 2 Nov. 2020, www.printedcircuits.com/printed-circuits-materials/#:~:text=Printed%20circuit%20boards%20(PCBs)%20are,have%20fifty%20layers%20or%20more.

(49) "memory components data book". *memory components data book*. Intel. p. 2–1. Archived from the original on 4 March 2016. Retrieved 26 April 2021

(50) Apps, BLE Mobile. “What Is Bluetooth Address & Privacy in BLE?: BLE Mobile Apps.” *BLEMobileApps*, 21 Oct. 2020, www.blemobileapps.com/blog/understanding-bluetooth-security-bluetooth-address-p

rivacy/#:~:text=A Bluetooth address is a,it is known as BD_ADDR.&text=A Bluetooth device utilizes at,public and random address types.

(51) Afaneh, Mohammad. "Bluetooth Addresses & Privacy in Bluetooth Low Energy." *Novel Bits*, 14 Apr. 2021, www.novelbits.io/bluetooth-address-privacy-ble/#:~:text=A Bluetooth address sometimes referred,is referred to as BD_ADDR .

(52) Awalt, Ashley. "Series and Parallel Battery Circuits." *DigiKey*, Digi-Key Electronics, 1 Feb. 2019, www.digikey.com/en/blog/series-and-parallel-battery-circuits?utm_adgroup=General&utm_source=google&utm_medium=cpc&utm_campaign=Dynamic Search_EN_RLSA&utm_term=&utm_content=General&gclid=Cj0KCQjwppSEBhCGARIsANIs4p6BHuXCMgRYMWmNpc1hv34pYln3aSab9B03FZZroEZmSOaezEMd9wQaAtsVEALw_wcB.

(53) Lindsay_1029, et al. "Overview of Batteries in Series or Parallel." *Key*, 17 Oct. 2018, forum.digikey.com/t/overview-of-batteries-in-series-or-parallel/1995.

(54) "IEEE Recommended Practice for Software Requirements Specifications," in IEEE Std 830-1998 , vol., no., pp.1-40, 20 Oct. 1998, doi: 10.1109/IEEESTD.1998.88286.

(55) *Java JDBC API*, docs.oracle.com/javase/8/docs/technotes/guides/jdbc/.

Appendix

Pin Configurations

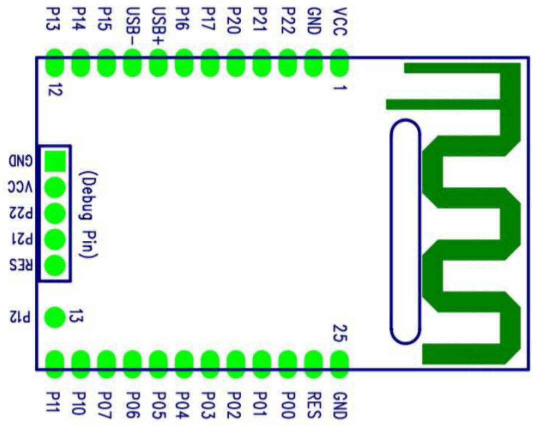
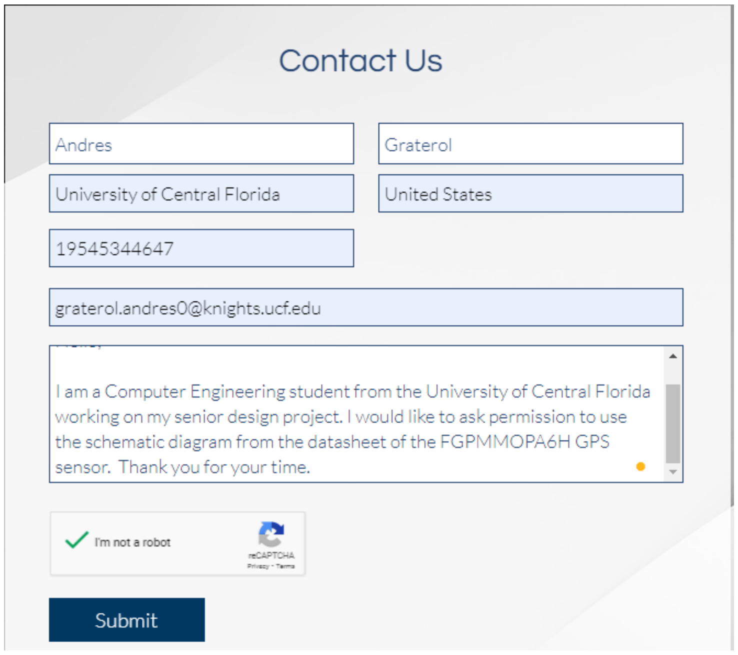


Figure
RioRand Bluetooth Module Pin Configuration

Copyright Requests



Contact Us

Andres	Graterol
University of Central Florida	United States
19545344647	
graterol.andres0@knights.ucf.edu	

I am a Computer Engineering student from the University of Central Florida working on my senior design project. I would like to ask permission to use the schematic diagram from the datasheet of the FGPMMOPA6H GPS sensor. Thank you for your time.

Figure
Copyright requests from GlobalTop Technology Inc.

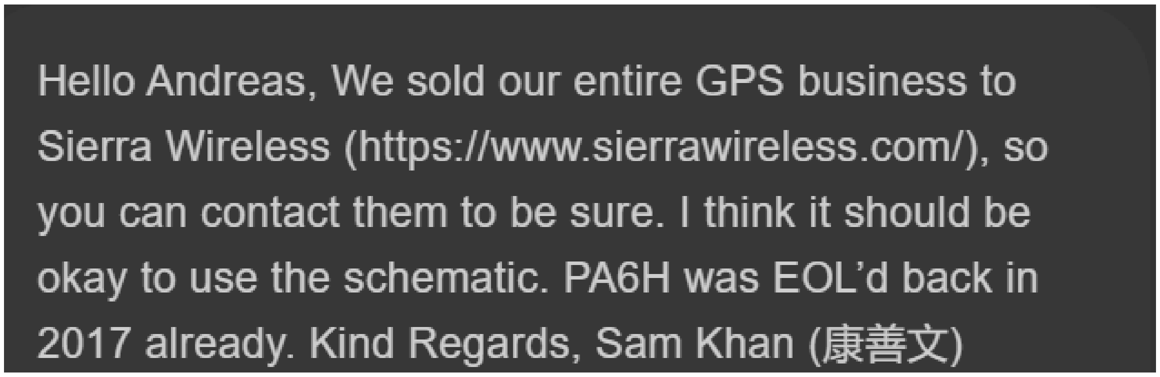
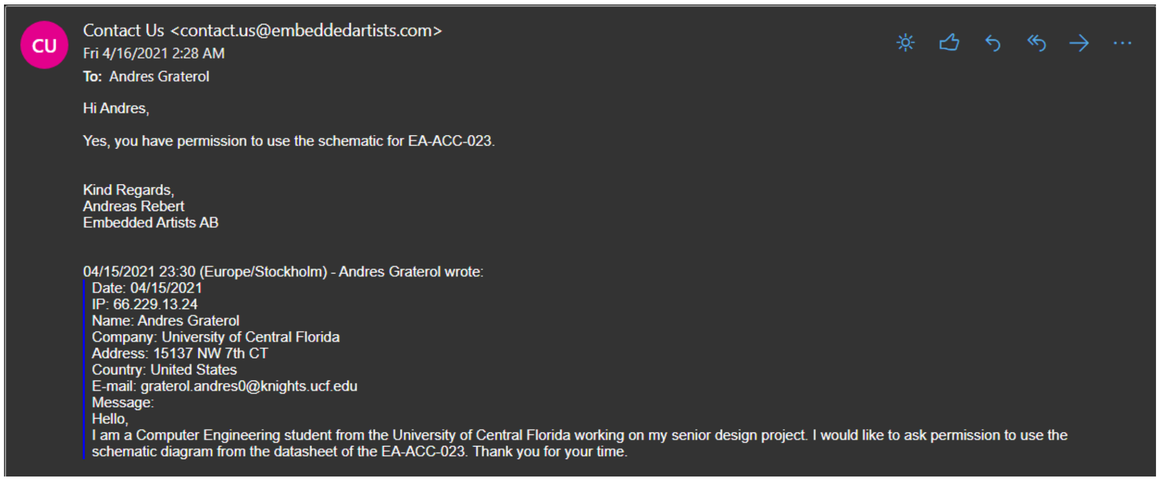

Hello Andreas, We sold our entire GPS business to Sierra Wireless (https://www.sierrawireless.com/), so you can contact them to be sure. I think it should be okay to use the schematic. PA6H was EOL'd back in 2017 already. Kind Regards, Sam Khan (康善文)

Figure
Copyright response from GlobalTop Technology Inc.


Contact Us <contact.us@embeddedartists.com>
Fri 4/16/2021 2:28 AM
To: Andres Graterol

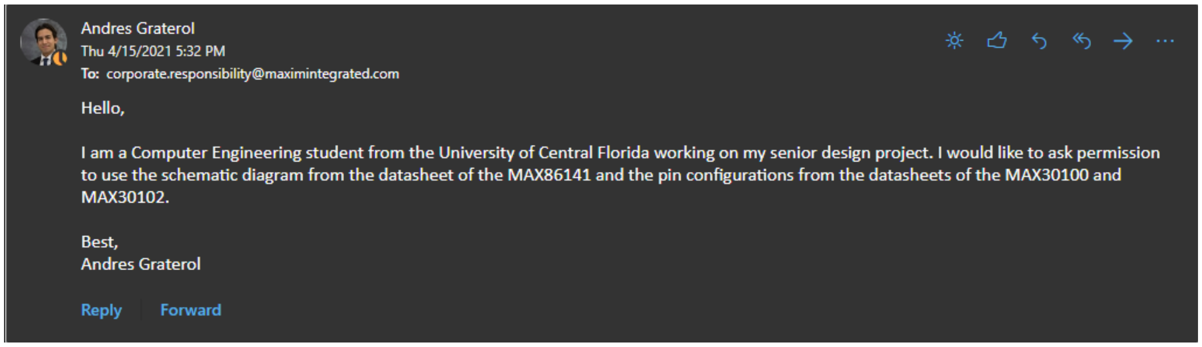
Hi Andres,

Yes, you have permission to use the schematic for EA-ACC-023.

Kind Regards,
Andreas Rebert
Embedded Artists AB

04/15/2021 23:30 (Europe/Stockholm) - Andres Graterol wrote:
Date: 04/15/2021
IP: 66.229.13.24
Name: Andres Graterol
Company: University of Central Florida
Address: 15137 NW 7th CT
Country: United States
E-mail: graterol.andres0@knights.ucf.edu
Message:
Hello,
I am a Computer Engineering student from the University of Central Florida working on my senior design project. I would like to ask permission to use the schematic diagram from the datasheet of the EA-ACC-023. Thank you for your time.

Figure
Copyright permission from Embedded Artists

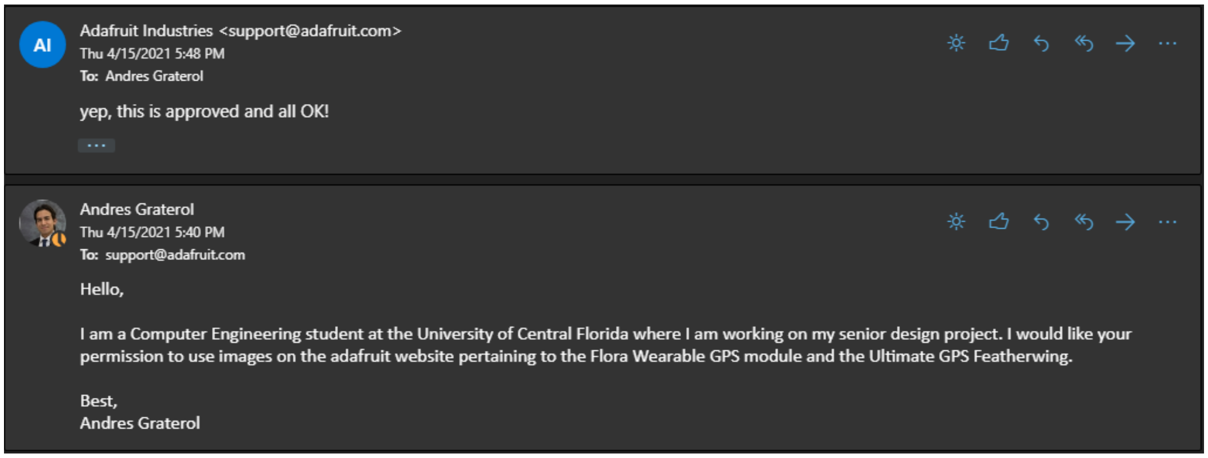

Andres Graterol
Thu 4/15/2021 5:32 PM
To: corporate.responsibility@maximintegrated.com

Hello,

I am a Computer Engineering student from the University of Central Florida working on my senior design project. I would like to ask permission to use the schematic diagram from the datasheet of the MAX86141 and the pin configurations from the datasheets of the MAX30100 and MAX30102.

Best,
Andres Graterol

Figure
Copyright permission from Maxim Integrated


AI
Adafruit Industries <support@adafruit.com>
Thu 4/15/2021 5:48 PM
To: Andres Graterol

yep, this is approved and all OK!

Andres Graterol
Thu 4/15/2021 5:40 PM
To: support@adafruit.com

Hello,

I am a Computer Engineering student at the University of Central Florida where I am working on my senior design project. I would like your permission to use images on the adafruit website pertaining to the Flora Wearable GPS module and the Ultimate GPS Featherwing.

Best,
Andres Graterol

Figure
Copyright permission from Adafruit