Senior Design 1
Initial Project Document
Snoozie Smart Pillow

The University of Central Florida
College of Computer Engineering
Dr. Samuel Richie
Dr. Lei Wei

Group 46:

Jason Davis	CpE	jcdavis2020@knights.ucf.edu
David Kipikash	CpE	kipikash@knights.ucf.edu
Kevin Perez	CpE	kevinperez@knights.ucf.edu
Christian Tomlinson	CpE	Ctomlinson8@knights.ucf.edu

Table of Contents

Section 1: Executive Summary

1.1 Introduction

Technology today is a very important part of our lives during our waking hours and has improved our lives many times over. With that said, our project, the Snoozie, aims to use the same technology that improves our lives when we are awake, to improve the time that we spend asleep. Sleep in humans is a vital part of a person's health and the average person will spend 1/3 of their lives asleep, and we think that our project, the Snoozie, can use its multitude of features to make the time before, during, and after sleep a more enjoyable experience.

With that said, the main goal for this project, the Snoozie, is to make a comfortable, affordable, and durable product that a customer can use to make the time spent in bed more functional, more comfortable, and overall a vastly improved experience that one would have rather than using a typical pillow that is widely available today. In order to achieve this goal, we will utilize many features that will be on the physical pillow, along with an intuitive app that will sync with the pillow to help navigate and employ the various processes and features that the smart pillow can do.

The main features of this project revolve around going to sleep, getting a good sleep, and waking up from the sleep. To do this we have features such as built in speakers, movement sensors, and a microphone on the pillow, and on the connected app we will have a sleep monitor, and wireless control over the pillow. We will go over these features and many more in depth during the coming pages of this paper..

1.2 Motivation

The pillow is an object that we as people get to use for a large portion of our lives, and we want that time spent to be a good experience. The humble pillow is a much used but overlooked part of our lives and can play a big part in how well a person sleeps. Improving this time spent using a pillow where you normally would is the biggest motivator to this project, and the aim is to be able to give others that good night sleep.along with getting a good night sleep, we want to be able to wake up in a healthy way as well because it is important to make it to your scheduled events on time. Missing an exam because your alarm didn't wake you up is a mistake you don't want to make and that is what is motivating us to make this product.

1.3 Project Goals and Objectives

For our team to build the Snoozie, we will need to have some goals and objectives that we will need to set so that we can make a good product that will be able to meet the customers needs. Our main goals and objectives for what we want to do with the Snoozie are to:

1. Provide the user with a comfortable sleeping solution
2. Improve the time spent before and after sleeping
3. Track and improve the sleeping patterns of the user

We want our user to be able to enjoy using the pillow, so we will need to make the pillow comfortable to use which is one of our goals that we have listed. This is probably the biggest concern that we have because at its foundation it is a pillow, and the first thing a person will notice is how comfortable it is to sleep on. If the pillow is not comfortable, it will immediately turn the customer away from using our product.

The next goal for our project is important for the time that is spent before and after you are in a deep sleep. We want to make sure that the pillow has functions that the user can use to put them to sleep, and make sure they wake up in the morning in a healthy manner.

The final goal we have for the project is to be able to use the app to track and improve the sleeping patterns of the user. It is important to get a good rest and many factors while sleeping can affect the quality of sleep you are getting, so we want to track things like light, sound, and movement to help the user achieve the best quality sleep that they can get.

Overall, our goal for the Snoozie is to have a better sleep, and we will use the many features in the software and hardware of our product to provide the user with that amazing sleep that all people need to have a productive day.

Section 2: Product Description

2.1 Project Features

As stated above, one of the main concerns that people might have with a smart pillow is how comfortable it is to sleep on, and that is a totally valid concern with all the parts and electronics that will be inside the pillow. The Snoozie will have all the electronics held sleekly within the pillow and surrounded by a firm memory foam that will stop the electronics from being felt by the user. This will solve the problem of the electronics being felt, but that does not necessarily mean that it will be comfortable to sleep on; to solve this issue, we will have inserts that the user can put in the pillows that will allow the user to change how firm of soft the pillow will be and adjust the thickness of the pillow to the user liking. Many high-end pillows found on the sleep foundation website such as the Brooklinen Down Pillow allow for the buyer to choose what firmness they want in a pillow, and we want the customer to be able to do this, and even change it whenever they want to. This should solve the issue of comfort for the pillow, regardless of how soft/ firm the user wants the pillow, or however the user would like to use the pillow for sleep. This is the foundation of a good pillow, but now we can build off of this to provide the user with an even better experience that will put our product over all other pillows on the market today.

As for the added functions of the Snoozie, we plan to put many features that improve the quality of sleep such as a sleep wellness tracker. A good night’s rest is extremely important for a person to remain awake and alert during the day and the in-app sleep wellness tracker will aid the user in getting those valuable eight hours of rest. The sleep wellness tracker will be a part of the app that will connect to the pillow and will track the sleeping patterns of the user using motion sensors, a light sensor, and a microphone built into the hardware of the pillow. The pillow will track movement, light levels, and sound to see when the user is in REM sleep, how well the user is sleeping, and if the user is snoring or has sleep apnea. Once the pillow has taken all the information it will use an algorithm to tell the user how they can improve their sleeping habits, and the health benefits to changing their sleep habits. Along with that, the pillow will utilize speakers and vibrations in the pillow to allow the user to set alarms, and the sleep tracker will work along with the set alarm to wake you up in a healthy, non-jarring way. The sleep tracking algorithm will also alert the user to when they should go to sleep to have a good amount of sleep through the app notification system. This system will not only provide the user with a comfortable pillow, but also with a way to improve their overall quality of sleep and improve their sleep health.

Another important feature of the Snoozie is the built-in speakers. Many people like to listen to music while they are in bed before they go to sleep, or even while they are sleeping. To meet these needs, we include built-in speakers that can connect to the app or act like a Bluetooth speaker. The speakers will also be able to connect to the app to make different preset sounds such as white noise, comforting rain, crashing waves, and smooth jazz. As said before, the speakers will also be used alongside a vibrating motor for the alarm to alert the user to wake up. To facilitate all of these features, the pillow will also include a side panel that will have auxiliary controls such as the power button, a volume control dial, Bluetooth pair button, and the charging port for power. These are all the features that we plan to add to the project, and we are trying to make the product as affordable as possible to compete with the market prices of pillows on the market.

All in all, the features that we plan to have included inside of the snoozie all point towards the user having a better experience in all things that involve a pillow. Many people might already use similar items that are included in the Snoozie, but we plan to bring all these items together into a smart convenient package that the user can easily use without needing to gather all the different items a person wants.we want it to be as convenient and intuitive as possible so the user can use all the features even when half asleep. With that said, we will have some challenges to overcome to make this product a reality.

2.2 Challenges

For us to make the Snoozie, we will need to overcome some challenges that will come up during the development phase of the project. With the electronics being located inside of the pillow, we will have some issues when it comes to the size of the components. If the components are too big, it will affect the level of comfort that the user will experience. With that said, the electronics will also be heating up, so cooling will be an issue that will also affect comfort. Battery life will need to last for at least the night and batteries can get pretty large when we need to power components for a longer length of time. With all the features that we will be adding to the project, the components will need a large amount of processing power, especially the recording and sound, which is a large part of the Snoozie, so we will need to find an MCU that will be able to handle these very load intensive processes. Along with that, the MCU will need to be able to use bluetooth so that it can connect to the phone and send data to and from the app interface that we will be using. We also want to make sure that the customer will be getting a good deal for the product, and with all the parts and features it can become hard to keep it at a good price. We want to make sure that the Snoozie is as affordable as possible.

2.3 Marketing

2.3.1 Similar Products

When it comes to the market availability, there are some “smart” pillows that do only some of the features that we provide, but there are not many that have all the features that we have planned for our product. For the ones that do share a good portion of the functions of our project are priced very high ranging from $200 - $300, so one aspect that we will be focusing on is the affordability of the Snoozie so that we can give everyone a good quality sleep at a fair price. We also want to make sure that the app is as intuitive as possible, so that it is easy to use even while you are half asleep, which hopefully will be a common occurrence when using the product. Overall, we want the product to serve as a daily tool to assist in all things sleep related.

2.3.2 Target Audience

The target audience that we plan to be able, broadly, is anyone who sleeps generally. Of course, this is most of everybody, so specifically because we are making a product that helps the user sleep, we want to market this to people who are having any issue with sleeping. This can be many different issues such as people who need to record their sleep patterns to diagnose a sleep related problem, people who need white noise to fall asleep, or even people who have trouble waking up from their sleep. We also want people who don’t have issues with sleep to be able to want this as well, and just improve the sleep that they are already getting. All in all, that target audience for this product is very broad, which is a good thing because we will have a large pool of buyers.

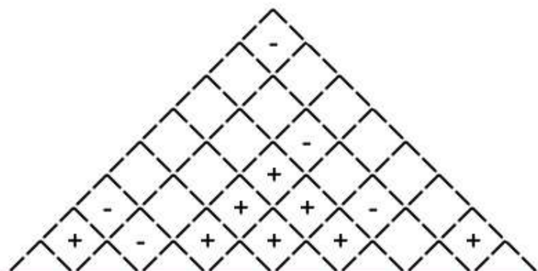
2.4 House of Quality

As you can see in the figure below, our house of quality graph shows what will be important to the process of creating the snoozie such as the importance of the weight and durability of our product. We are going to use this table to see what we can improve and what we can lose to improve the Snoozie.

Correlations	
Positive	+
Negative	–
No Correlation	

Relationships	
Strong	●
Moderate	○
Weak	▽

Direction of Improvement	
Maximize	▲
Target	◇
Minimize	▼



	Column #	1	2	3	4	5	6	7	8
	Direction of Improvement	▼	▼	▲	▲	▲	▲	▼	▲
Direction of Improvement	Customer Requirements (Explicit and Implicit) / Engineering Requirements	Weight	Dimensions	Power Output	Compatability	Signal Strength	Sensor Accuracy	Temperature	Durability
▼	Low Cost	○		●	▽	▽	▽		▽
▲	Easy to Use				●			○	
▲	Easy to Install	●	●		●				○
▼	Small Form Factor	●	●	▽				▽	▽
▲	Long Battery Life	▽		●	○	▽	▽	●	●
▲	Sleep Improvement				●			●	●
	Target	<= 5lbs	< = 20 inches by 30 inches	<= 10 Watts	Bluetooth + Wifi capabilities	5 meters - 10 meters	<= degrees error	23C to 30C	Function 100% all the time

Section 3: Standards and Constraints

3.1 Hardware Specifications

When it comes to the hardware specifications, it is important to focus on the safety concerns. This is because we are working with a pillow containing electronics in it. Therefore, keeping track of the device's temperature and having some sort of passive cooling system is essential to have in this device. Additionally, given that weight and size is another limitation, we must keep the dimensions of the components within the restrictions of the pillow size.

A simple search online will reveal that the standard pillow size is: 20 inches by 26 inches. So, by using these dimensions as our standard, we will be able to prepare the hardware before the installation. With some research online, we also find that the standard pillow weight is 14 ounces. This is light and will be difficult to maintain. Given that we will be including electronics and a sort of memory foam to cover the hardware, the weight will vary significantly.

Standard pillows weight:

Size	Light	Medium	Supportive
Standard 20 x 26	12 oz	14 oz	16 oz
Queen 20 x 30	14 oz	16 oz	20 oz
King 20 x 36	17 oz	20 oz	26 oz

Memory Foam Pillows Weight:

Size	Weiget
Standard 20 x 26	2.75 pounds
Queen 20 x 30	3.6 pounds

King 20 x 36	4.2 pounds

The tables above show different pillow types with their corresponding weights, depending on our project needs we can decide between those on which to use. Though, the best option would be to keep the weight around 5 pounds. Therefore, we must keep track of the hardware weight to not exceed the limit.

In addition, the hardware should withstand normal use. A pillow is moved around a lot. Therefore, the memory foam must be thick enough that the user cannot feel the components at any time by putting pressure in the front of the pillow. This is also a concern for safety as keeping the user and the electronics separated is required.

ID	Priority (1-3)	Requirement Description	Test Description
001	1	The device's passive cooling will maintain the system at room temperature	Pillow's temperature will remain between 23C-30C
002	3	The device (and the electronics housed within it) will maintain a reasonable weight of around 5 pounds	When measured, the pillow should weigh between 5-8 pounds maximum
003	1	The device will contain a compact ambient light sensor for recording ambient light in the room	The ambient light sensor shall not occupy more than the space given within the memory foam, nor shall it weigh more than 10 grams
004	1	The device will contain a compact microphone for recording ambient noise in the room	The microphone shall not occupy more than the space given within the memory foam, nor shall it weigh more than 30 grams
005	1	The device will contain a compact accelerometer for recording motion during the	The accelerometer shall not occupy more than the space given within the memory foam,

		night	nor shall it weigh more than 20 grams
006	1	The device will contain a compact thermal sensor for recording device temperatures and monitor overheating	The thermal sensor shall not occupy more than the space given within the memory foam, nor shall it weigh more than 10 grams
007	1	The bluetooth transceiver shall be able to connect with a mobile application	When connected through Bluetooth, the connection should be maintained until disabled.
008	3	The pillow will contain audio feedback capability. Output from 2 speakers in the pillow.	When device is connected to mobile app, user will be able to hear ambient noise in their environment through the pillow
009	1	The audio received from the external microphone will be transmitted to the app via bluetooth	Bluetooth transfer rate should be between 3 Mbit/sec – 24Mbit/sec
010	1	The data received from the accelerometer will be transmitted to the app via bluetooth	Bluetooth transfer rate should be between 3 Mbit/sec – 24Mbit/sec
011	1	The data received from the ambient light sensor will be transmitted to the app via bluetooth	Bluetooth transfer rate should be between 3 Mbit/sec – 24Mbit/sec
012	2	The user will be able to increase the volume through “volume up” button on the outer control panel	When the button is pressed, volume should increase by 3dB. When the button is held down, the volume will increase until it’s not being pressed.
013	2	The user will be able to decrease the volume through	When the button is pressed, volume should decrease by 3dB.

		“volume down” button on the outer control panel	When the button is held down, the volume will decrease until it’s not being pressed.
014	1	The user will be able to pair their bluetooth device using the “pairing” button on the outer control panel	When the bluetooth button is pressed for 3 seconds. Pairing mode shall activate and be recognized by the mobile device within 5 seconds.
015	1	The user will be able to power the device on and off using the “power” button on the outer control panel	Once the “power button” is installed with a cable (connected to two pins on the motherboard, the circuit being used should be closed on the motherboard. This has the power supply receive the signal that it should supply power
016	3	The user will be able to snooze their alarm by saying a keyword (i.e. “stop”)	The mic will listen for the trigger word while the alarm is going off and if the word is caught, it will shut off the alarm
017	3	If an alarm is going off, the user will be able to shake the device in order to snooze the alarm	The accelerometer will look for sharp movements while the alarm is going off and turn off the alarm when that happens
018	2	The pillow will be able to handle fall damage from a 4-foot drop off a bed	The pillow will have enough padding built in to it to be able to protect the electronics sufficiently
019	2	The pillow will be able to handle light water exposure (i.e. drool or a small spill)	The electronice will be placed inside the pillow enough to shield it from a fall. (note: the pillow will not be completely waterproof and will not be able to be submerged)

020	1	The pillow will be able to handle pressure from a user's normal sleep patterns	The electronics will be encased inside of the foam and protected from normal patterns of sleep
021	2	The pillow will maintain about the same dimensions as a normal pillow (18" x 26")	We will be able to fit all of the components and padding inside of the preferred dimensions by keeping the electronics compact
022	1	The height of the pillow shall remain under 7" to maintain comfort	We will design the components to be wider rather than taller to keep within these parameters
023	1	The device's battery shall be able to last a full 12h before needing to be recharged	The battery will need to be able to last a full night's sleep and some more.
024	2	The device will implement a "power saving mode" that will allow it to save battery overnight, but disable some features	The user can either manually set the low power mode via app by setting a timer or immediately go into LPM. The pillow can also sense when the user is sleeping so it can trigger automatically.
025	1	The device will contain a bottom layer of memory foam to house and protect all the hardware components	The foam will be thick enough to prevent damage to the parts, but thin enough to keep within 7"
026	2	The device's more fragile hardware components will be housed in plastic casings within the memory foam	The case should not exceed 3.5"
027	2	The device's memory foam layer must remain smooth as to not impede the user's comfort	The user should not be able to feel the components and casing while using the pillow as intended

028	1	The device will house a microcontroller responsible for handling all inputs and outputs to the and from the device	The microcontroller should be able to connect to the app via bluetooth and communicate data to and from said app while handling the data from the components
029	1	The device will implement a compact, rechargeable power supply	The rechargeable battery will be able to comfortably fit within the pillow and not affect the user comfort
030	1	The device will be rechargeable through a usb-c port located on the side of the device	The pillow MCU will charge the battery, and when the battery is full be able to run off the wall power

3.2 Software Specifications

For software specifications, we must guarantee that the application can receive data from the device so that it can be organized. This will allow the user to view their data of their sleep. This is what is called their “Sleep Wellness”, the app will gather the data and display it to the user.

Additionally, the app can be provided into three main parts. These parts are actions that the user will have. The first is the ability to organize the data and view it as stated before. Secondly, the ability to edit, add, and remove alarms. These alarms will sound at the time sent through the pillow speakers. The user will also be allowed to upload .mp3 sound files and replace the alarm tones with custom ones that they desire.

Finally, the user will also be able to play custom music by linking a third-party music player application such as Spotify, this will allow them to play anything from those apps through the pillow speakers. In addition, the app will also provide some pre-loaded White Noise that the user can play while trying to sleep.

The pre-loaded white noise is part of a library of preset nighttime sound profiles that the application provides. The app offers a selection of sounds that play through the device's speakers. This will also work with the custom .mp3 files that the user decides to upload to the app to use instead.

Another Specification that can be added is to go into power saver mode. Without a doubt having all these electronics active at the same time and drain the battery of the device. With a power saver mode controlled through the app, the user will be able to enable/disable some components that will not be required/used throughout the night.

Another basic specification is that the app should be available at least for android through the play store. On the other hand, if we decide to make this app compatible with Apple, then we would have to use a cross-platform framework such as Xamarin or React Native to be able to provide the app install through both Android and Apple.

ID	Priority (1-3)	Requirement Description	Test Description
001	1	The application will provide users with a page where they can access and edit their alarms	When the user sets/modifies an alarm. The alarm will go off precisely at the time set.
002	1	The application will provide users with a page where they can track their sleep cycles ("Sleep Wellness")	When the users track their sleep cycle, the app will display the "sleep wellness" according to the data gathered from the sensors.
003	2	The application will provide users with a page where they	The app will provide a selection of sounds/music the user can select. The audio should output through

		can control their nighttime sound profile	the pillow speakers successfully.
004	3	The “Sleep Wellness” page will provide users with a line graph showing audio interruptions over time throughout the night	The app shall gather data using the microphone module and send the data to the app to gather and display successfully.
005	3	The “Sleep Wellness” page will provide users with a line graph showing levels of ambient light recorded throughout the night	The app shall gather data using the transceiver and send the data to the app to gather and display successfully.
006	3	The “Sleep Wellness” page will provide users with a line graph labeling their sleep cycles throughout the night	Using the sensors, the pillow will track the user's sleep cycle through the night every day and display the data accordingly on the graph.
007	3	The “Sleep Wellness” will allow users to customize what information they want shown to them	When the user selects the type of data they want displayed, the app will adjust accordingly and successfully display the information.
008	1	The application will be able to stream all audio of the connected device (similar to a generic bluetooth speaker)	When the smart pillow is connected through Bluetooth to the phone, the speakers should output any type of audio the user decides to play.
009	1	The application will provide users with the ability to control the volume of the device.	Aside from being able to control volume from the panel, the user shall be able to control the volume using the phone’s volume settings.

010	1	The application will provide users with the ability to set an alarm	When an alarm is set through the app, the alarm will sound at the time entered.
011	2	The application will provide users with a library of alarm pre-set alarm tones	When the user sets an alarm, the app will provide a selection of ringtones to use. When one of these ringtones is used, the alarm will play it at the alarm's time.
012	3	The application will allow users to upload custom alarm tones in the form of .mp3 files	If the user decides to have a custom ringtone. The app will allow a custom ringtone file that shall play correctly at the alarm's time.
013	3	The application will be able set alarm tones through the user's music library (i.e. Google Music, or Spotify)	If the user uses a song from a music app, the smart pillow app will adjust accordingly to output the custom sound.
014	2	The application will provide users will a library of preset nighttime sound profiles	The app will offer a selection of background (whitenoise) sounds to select. When one is selected. The app will output the sound through the pillow speakers.
015	3	The application will be able set a nighttime sound profile through the user's music library (i.e. Google Music, or Spotify)	The user can select a background sound from any music app.. When one is selected. The app will output the sound through the pillow speakers.
016	1	The application will use the audio received from the microphone to generate the user's sleep report	Throughout the night the microphone module will track sounds made (snore) and display the data in the app.

017	1	The application will use the data received from the ambient light sensor to generate the user's sleep report	Using the ambient light sensor, the data will be sent to the phone where it will display the user's sleep report successfully.
018	1	The application will use the data received from the accelerometer to generate the user's sleep report	Using the accelerometer sensor, the data will be sent to the phone where it will display the user's sleep report successfully.
019	2	The application will use the data received from the accelerometer to adjust the time to wake the user up	Using the data from the accelerometer, the app will learn the user's sleep cycles and adjust the time the user usually wakes up.
020	3	The user will be able to turn the device on and off using the application	The app will send an on/off signal to the MCU turning the device on or off
021	1	The user will be able to download the application on any Android-enabled device	The app should be made available through the app store
022	1	The application will use the data from the accelerometer and an algorithm to track the user's sleep cycles	The data will be received from the MCU and use it in the learning algorithm along with the other data
023	2	The user will be able to enable/disable "power saver mode" through the application	The app will send data to the MCU to tell it to turn off some of the components to make the battery last longer

3.3 Constraints

In engineering, we encounter multiple limitations and restrictions, these are referred to as constraints. These can involve multiple things varying through, technological, financial, etc. These must be dealt with correctly.

3.3.1 Technological Constraints

Building a smart pillow that contains electronics can bring a lot of issues and constraints that must be addressed. One of the biggest issues is safety constraint, we must solve how to avoid the user from having direct contact with the hardware during sleep. It is important to have these two-thing separated, the components can be damaged if too much pressure is put on them, additionally, having the electronics have a direct contact with the user's head can be dangerous. Therefore, one possible solution for this is to use memory foam. We can also use multiple layers of thermally conductive and inductive foam that protect the components from being in direct contact with the user.

Additionally, compatibility is another issue we must address. We must ensure that the pillow can communicate effectively with a mobile device through the app. For this we need to decide when is the best use case of WIFI and Bluetooth, using these two technologies appropriately can ensure that we build a strong connection with the mobile device. It is important that we offer the users a device that does not encounter any problems when connecting to the app.

Durability is another technological constraint. We must address the issue of pillows being constantly pushed around and being put under pressure on them. Given that our smart pillow will contain electronics, it is important to have these electronics protected from normal pillow use. This can be done similarly to our first technological constraint. We use multiple layers of memory foam, or one thick memory to have the components protected from normal pillow use such as fall and pressure.

3.3.2 Financial Constraint

Cost is a big factor in restricting how much we can add to the device, it can limit how much safety, components, memory foam that we can put. Therefore, the cost constraint has a huge impact on the effectiveness of our device. One solution for this is to already decide all the parts we need to build the pillow, this way we can manage our expectations and adjust accordingly to the parts and what limitations they come with. This allows us to plan ahead for the restrictions that may arise.

3.3.3 Time Constraint

Time is another big constraint that can change the outcome of the project. It is significantly important that we manage our resources and time accordingly to ensure that we make a device and app that work well with each other with the capabilities that we decided that they would offer.

The Following table contains additional constraints and possible solutions:

ID	Difficulty (1-5)	Constraint Description	Solution
001	3	Temperature: It is important that we monitor the temperature as the device should be safe to use. Any heating issues not only can possibly damage the components but also bring harm to the user. Therefore, it is important to have the components protected and have a passive cooling system.	A solution for this would be to minimize the total power needed to run the components, this would reduce the heat output. Additionally, we can also customize the pillow to include small holes on the pillowcase that would increase the airflow between the components.
002	4	Durability: As the device is a pillow with components, keeping it safe from exterior damage such as from fall, liquids, or from simply moving it around.	Adding a thick memory foam that covers the electronics, reducing the damage from fall or water. Additionally, the pillow case can be water-proof to protect the electronics from any liquid damage.
003	3	Safety: It is important to guarantee the safety of the user when the pillow is in use.	Similar to the durability solution, the memory foam will protect the users from being in direct contact with the electronics. Additionally, it is important that the temperature

			is constantly tracked by the app. In case the temperature rises above a threshold, the app will automatically shut down the smart pillow.
004	2	Compatibility: Making sure the device can connect with ease to the phone app and will stay connected is another constraint as we must guarantee the effectiveness of the connection.	A solution to this is to have a strong signal with the transceiver module that can extend the connection's distance and strength.
005	3	Time Management: It is important we plan what we should do each week to maintain our schedule. If deadlines aren't being fulfilled, then our project can suffer significantly. This can impact major features this project has meant to have. Therefore, keeping track of our deadlines and fulfilling them is a high priority.	A solution for this is to constantly update out project milestones so that we know we have deadlines to fulfill. Additionally, we can apply the AGILE method to increase the effectiveness of our project design and implementation.

006	1	Cost: Given our limited budget we will have to think ahead about the components we are getting so that we do not go over the limit and we might need to adjust our requirements to fit the budget. It is important we are realistic with our set budget so that we don’t expect to get better components that require a higher cost. Secondly, the set budget can change depending on the urgency or importance of a certain piece required for the project. This can greatly affect the overall cost.	A solution would be to plan the specific parts needed and set aside a certain amount that can be used if needed. This would allow us to be certain of the total amount of cost to expect.

Section 4: Research and Investigation

4.1 Similar Projects

The projects below are examples we choose to research in as they share similarities with what we intend to make. The following examples may have different purposes, mechanisms, and features, but a lot of ideas can be pulled from them to build our project. These are the 4 projects/examples that we have researched:

4.1.1 Smart Pillow by Silego

The main point of this project is to make a smart pillow that is sensitive to snoring. This device analyzes sounds and activates a vibration motor so that the user wakes up and changes position to one that prevents snoring. This project was done by using a Silego SLG46620V CMIC, a vibration motor, and a sound sensor. Instead of using a microcontroller, they use a GreenPAK CMIC, as they are low cost, and have low power consumption. An issue with this project is that it does not detect the type of sounds the user makes, instead it checks the level of a sound made. Therefore, false triggers occur under different sounds. To avoid this issue, they build a digital filter that can detect a sound segment like that of snoring. Once the user places their head on the pillow, the touch signal sends an activation signal and activates the sound sensor to check for audio. Samples are taken every 30ms within a 5ms time frame. If 15 sound segments are detected, where no silence lasts for more than 400ms between the samples, then it's determined that the sound is persistent and a snoring segment. If this repeats after a silence which lasts more than 400ms and less than 6s, the sound will be considered snoring, and the user will be awoken with the vibration motor. [link]

4.1.2 Sen Smart Pillow by Kelsey Leppek

This project has many similarities and components that we can review that can help us with our Snoozie pillow. This project has a temperature control module that raises and lowers the temperature of the pillow depending on the sensory input. Additionally, it contains a speaker and Bluetooth connection to other smart home products. This project uses Peltier modules. Peltier modules are electronic devices designed to cool objects or maintain an object's temperature by controlled heating and cooling. The Sen smart pillow has an in-built interface for core functionality without the use of a screen. The core functionalities include A climate control tag to increase or decrease the pillow's temperature, and a Sleep preset tag to set the pillow in 3 different modes (Sleep mode, wind down mode, and wake mode). Furthermore, the Sen Smart Pillow, contains multiple layers of thermally conductive foam and thermally insulative foam, this helps protect both the user and electronics from being damaged. The Sen Smart Pillow also includes a companion app concept that has multiple functionalities. These functionalities include the ability to view all connected devices to the pillow, a sleep cycle page where you can see the data in different types of graphs, a page with the pillow modes where you can select different modes and control the temperature modules, speakers, etc. [link]

4.1.3 ZEEQ Smart Pillow by REM fit

A big smart pillow project that already has seen success in the market is the ZEEQ smart pillow by REM fit. Like what we wish to offer, the ZEEQ smart pillow has an app that tracks and analyzes your sleep, stream music, and it also tracks your sleep score, snore monitor and motion monitor. The pillow has a Tencel cover that is 40% lyocell from Tencel and 60* polyester. Inside the pillow is memory foam which can be adjusted by the user. The user can take out memory foam to their liking. Within the foam, the device contains 8 speakers, 2 vibration motors, a microphone, a 3-axis gyroscope, and binaural wave technology. Additionally, the pillow has a remote control directly plugged into the pillow that hangs out. This remote control offers quick and direct control over the pillow. The ZEEQ smart pillow is 21 inches in length, 27 inches wide and 5 inches in depth. This is almost the same measurements of a standard pillow. The microphone in the pillow also allows for voice commands. [Link]

4.1.4 Zola a Smart Pillow Project by Ashlynn Tan

Another smart pillow concept is the Zola smart pillow. This project concept has 3 main features: the ability to play ambient music, the ability to analyze sleep patterns via brain waves, and displaying this data to the user, and finally the ability to have multiple choices of settings to improve the quality of sleep of the user. This pillow contains memory foam, a pressure pad for sensor, an Arduino circuit, speakers, and a USB-C charging port on the side of the pillow. Like what we intend to build into the Snoozie app, this project also can connect to a playlist from outside the pillow's app such as: Spotify, Apple Music, SoundCloud. Though, it will contain its own playlists within the app. Another big feature of this app is the ability to change pillow modes, there are 4 modes: Sleep mode, where ambient music is played and fades away as the user sleeps. Dream mode, it plays ambient music if the pillow senses high brain activity. Alarm mode wakes the user up with a preferred playlist every morning. Sleep goal / Sleep reminder, where the user sets their ideal sleep time, and the app calculates the sleep goal for them. Additionally, the concept app for this project is sophisticated. You can choose any day to view your data, once a day is chosen, the app will show the detailed analyses: the number of hours the user has been awake, in light sleep, and in deep sleep. Additionally, the app also organizes the monthly data into a graph to show the user's sleep progress. This also comes with insight messages where the app mentions if the user had a good sleep the night before. Furthermore, besides the data section, the app also has a page dedicated to music, where the user can decide from which 3rd party app they want to play music from. There's also a settings page where the user can choose what mode they want the pillow to be in. Overall, the app concept for this app is very well made, it has a minimalistic design that keeps track of the user's progress and is easily available for the user to visualize. The colors used for the UI are also very light and easy to the eye.

4.2 Microcontroller

4.2.1 Processor Options

For the Snoozie to operate, we will need to employ the use of a micro control unit for all of the different functions to work correctly. We need to choose a microcontroller that will be able to handle some of the more processing intensive functions of our project such as listening for and outputting sound, being compatible with a Bluetooth controller to communicate with the app on the phone, and have enough memory to hold the larger amount of code that we will be using to operate the project. We want the controller to be located inside the pillow on the main circuit board and connected to all the hardware to be able to perform all the processing. We would like to only need one processor, but if the number of processes becomes too much for one processor, we might need to increase the number to facilitate the large processing power needed. We should not need a large or complete computer such as an ARM processor or any raspberry pi computers because it would be overkill for what we are doing for this project. I will be looking at 3 groups of processors and comparing them to see which we should use. We will compare the things like the cost, power consumption, and the memory to find the one that we will be using.

4.2.2 Texas Instruments MSP430 series

The main reason that we are considering this line of processors is because we all have had experience with these from our various classes such as embedded systems and junior design. This line of 16-bit processors were mainly designed to have a low cost and a low power consumption, so this would be great, but we need to make sure that it will have the memory and processing power needed for our application. the chips we will be looking at are the MSP430FR6989 and the MSP430G2553

4.2.3 Espressif Systems ESP Series

The reason that this series caught our attention is because some of the chips that are included in this line have built-in connectivity through Wi-Fi and Bluetooth, which is something that we need to have to communicate with our app. The manufacturer boasts low price and low power consumption for most of the chips included in the series and with the included connectivity, it makes this line a good consideration.

ESP 8266

This chip has built in TCP/IP communication software that we can use to communicate with the Wi-Fi to be able to connect with the app. We would need to change the way we plan to communicate with the pillow, but we would be able to do it. The main drawback to this chip is the low memory, but it is very low cost so we could use a separate controller to communicate with it.

ESP 32

This processor is the successor of the ESP 8266 and comes with many improvements that make this chip a really good choice. For starters, along with Wi-Fi connectivity, it also includes dual mode Bluetooth connectivity included on the chip, which would mean that it would have all we need to communicate directly with the app. Along with that, it is not much more expensive than the ESP8266 which still makes it pretty low in cost.

4.2.4 Atmel ATmega328p

This processor is widely used in the Arduino series and some of us have experience in this line, so we are considering this processor to use in the project. It is an 8-bit single chip processor that could be used in conjunction with a Bluetooth module to communicate with the app.

4.2.4 Micro Controller Comparison

Micro Controller	Cost	Memory	Communication	Seller
MSP430FR6989	$ 4.99	Non-volatile memory (kB): 128 RAM (KB): 2	GPIO pins (#): 83 UART: 2 Number of I2Cs: 2 SPI: 4	Texas Instruments
MSP430G2553	$ 1.08	Non-volatile memory (kB): 16 RAM (KB): 0.5	GPIO pins (#): 24 UART: 1 Number of I2Cs: 1 SPI: 2	Texas Instruments
ESP 8266	$7.00	32 KiB instruction 80 KiB user data	17 GPIO pins	Espressif
ESP 32	$7.39	Integrated 520 KB SRAM 4 MByte flash	UART: 3 Number of I2Cs: 2 SPI: 4 I2S: 2 ADC: 12	Espressif

4.3 Wireless App Communication

4.3.1 Introduction

The main reason that we need to include wireless communication is because our project will need to send and receive data to an external app. We will need to make sure that the way we choose will be able to facilitate the amount of data that we will be using. We also need to consider the price of the module(s) that we will use to do that will be low cost and efficient.

4.3.2 Wi-Fi

To use Wi-Fi for this project we will need to have a microcontroller with an innate Wi-Fi connection, like the ESP8266, or an external Wi-Fi module that fills the same position. As said before, the ESP8266 is a good candidate for choosing, but because of its low memory we would need to use it to supplement a second controller, making it act like a Wi-Fi module anyways. With that said, the better solution for connection between the pillow and the app would be Bluetooth.

4.3.3 Bluetooth

When it comes to the needs of this project, we think that Bluetooth will fill the role of wireless communication better than Wi-Fi. To start off we can look at the ESP 32 which has an innate dual channel Bluetooth Bluetooth and would be perfect to use to communicate with the app. We can also look at different Bluetooth modules that can also fill this role. We need to make sure that the range of this module can reach a decent range, at least from the pillow to the nightstand where the phone is put while sleeping. We also need to make sure that it can handle the sound portions of our project.

4.4 Serial Communication

4.4.1 - Introduction

Serial Communication is a method that uses one to two lines of transmission that both sends and receives data. The data being involved within the transmission will be continuously sent and received (bit by bit).

4.4.2 - Serial Communication Standards - RS-232,RS-422A, and RS-485

RS-232/RS-422A/RS-485 are all Electronic Industries Association communication standards. The standard that's most common out of the 3 is the RS-232, where it's even standard equipment on computers.

4.4.3 - Half Duplex vs. Full Duplex

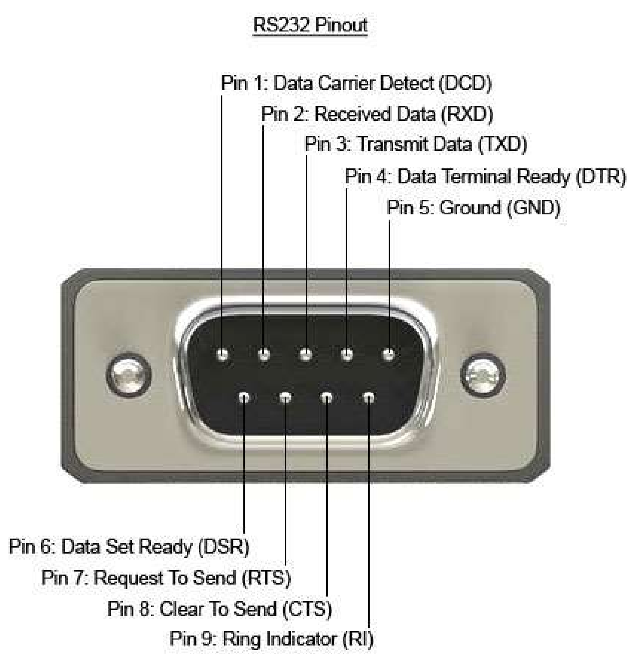
Within this project, it was important to research what Half and Full Duplexes are in order to fully understand what will be the most compatible with our pillow design. This is only the case if our team chose to go with the RS series. Full-duplex is when data transmission and reception is simultaneous over one channel. Half-duplex data will have to move in two directions. As opposed to full-duplex, half-duplex can't occur at the same time. There are a few more differences we would like to mention between these two duplexes. With full-duplex, it saves time within transmissions compared to half-duplex. This is because collisions and frame retransmissions are reduced. Also, full-duplexes have separate functions (between sending and receiving). This creates a system where there's going to be full data capacity in each direction. As opposed to full-duplex, half-duplex can be used to efficiently use bandwidth. In addition to this, half-duplex can immediately display transmitted characters onto a monitor. Full-duplex however, doesn't appear on the monitor until the transmission is both received and returned

4.4.4 - Serial Communication Standards Comparisons

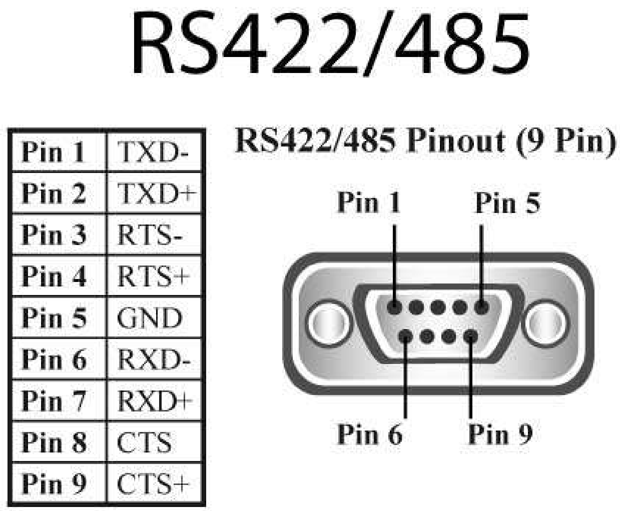
Standard	Description	Examples of Communication Devices
RS-232C	Widely used and usually equipped on computers as standard. Purpose and timing of signal lines and the connectors have been stated (D-SUB 25-pin or D-sub 9-pin). The transmission mode is simplex, with a short distance 1:1 connection. This standard is Full-duplex with a Single-ended, unbalanced type of operation mode.	COM-4CN-USB, CPS-COM-1PC, CPS-COM-2PC, COM-8C-LPE
RS-422A	RS-422A standard solves issues the RS-232C standard contained, such as a short transmission distance and slow transmission speed. Similarly to the RS-232C standard, the purpose and timing of signal lines are stated, however the connectors aren't. Items that are compatible usually use the D-sub 25 pin and the D-sub 9 pin connectors. The transmission mode is Multi-point simplex, with a long distance 1:N connection. This standard can either be a full or half-duplex with a differential, balanced type of operation mode.	CPSN-COM-1PD, COM-2PD-LPE, COM-1PDH-LPE, COM-4PDHN-USB
RS-485	The RS-485 standard solves the issues of a couple of connected devices within the RS-422A. The purpose and timing of the signal lines are stated, just like the other two standards. The connectors aren't connected, however. Just like the RS-422A standard, it uses the D-sub 25 pin and the D-sub 9 pin connectors. The transmission mode is multi-point multiplex, with a long distance N:N	CPSN-COM-1PD, COM-2PD-LPE, COM-2DL-PCI, COM-4DL-PCI

	connection. This standard can either be a full or half-duplex with a differential, balanced type of operation mode.	

4.4.5 - Serial Communication Standards Connection Images



The image above displays the RS-232C pinout outlook

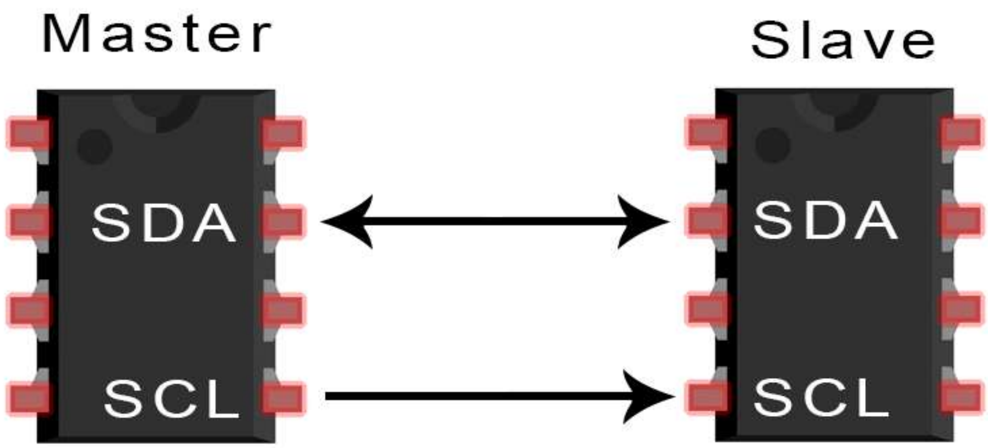


The image above displays the RS-422A and RS-485 pinout outlook

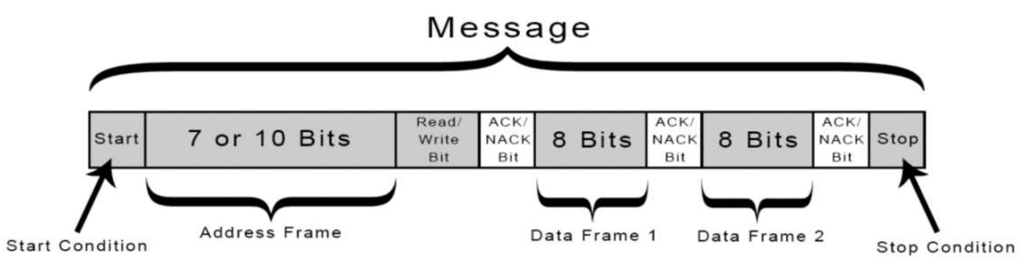
4.4.6 - Serial Communication - I2C

I2C communication combines features both SPI and UART contain. With this type of communication, we are able to connect multiple slaves to a single master, as well as have multiple masters controlling either multiple or single slaves. As a group, we determined this is an amazing feature because more than one microcontroller logging data into a single memory card is possible.

I2C uses two wires to transmit data between devices (as shown below). For reference, SDA stands for “Serial Data”. This is the line for the master and slave to send and receive data. SCL stands for “Serial Clock”. This is the line that carries the clock signal.

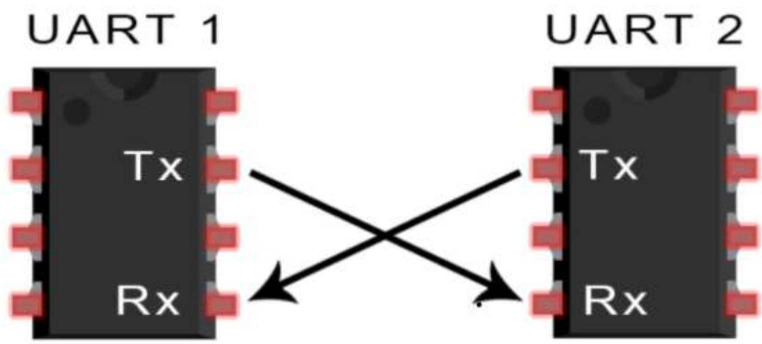


With I2C, data will be transferred in messages. These messages can be broken up into frames of data. Each message has an address frame that will contain the binary address of the slave, and at least one data frame that contains the data being sent over. Messages have ACK/NACK bits, stop and start conditions, and read/write bits for each data frame:



4.4.7 - Serial Communication - UART

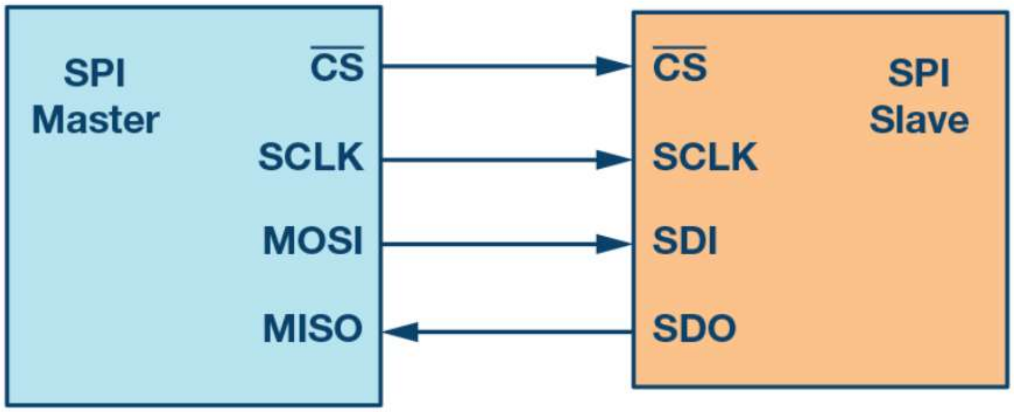
Universal asynchronous receiver-transmitter (UART) is a common device-to-device communication protocol. With UART, two UARTS work together directly, converting parallel data from a controlling device, transmitting it from serial to the receiving UART, which in the end, converts the serial data back into parallel data for the device receiving it. For this type of serial communication, only two wires will be required in order to transmit data between two UART devices.



Data is transmitted asynchronously with UART, meaning there will be no clock signal to synchronize the output of bits from the transmitting UART to the sampling of bits by the receiving UART. Instead, UART adds stop and start bits to the data packet being transferred. With these bits, they determine the end and beginning of a data packet so that the UART receiving the data will know when to start analyzing the bits.

4.4.8 Serial Communication - SPI

Serial Peripheral Interface is an interface that's commonly used between microcontroller and peripherals such as DACs, SRAM, Sensors, etc. It's a full duplex master-slave based interface (also synchronous). Data is synchronized from the master or the slave on the falling/rising clock edge. A feature that's nice with SPI is that the slave and master can transit data simultaneously, where the interface can either be a 4 or 3-wire (4-wire is most popular compared to the 3-wire). The image below displays the SPI configuration with master and a slave:



In this image, we have 4-wire SPI devices that contain 4 signals:

- MISO: Master in, Slave out
- CS: Chip select
- MOSI: Master out, slave in
- Clock: (SPI CLK, SCLK)

For SPI communication, master has to send a clock signal and select the slave by accessing the chip select signal. This signal is an active low signal, meaning that the master has to send a logic 0 on the signal in order to have the slave be selected. With SPI, data communication is transmitted and received at the same time. Serial clock edge will synchronize both the sampling and shifting of data. It would provide our group with flexibility to either pick the falling or rising edge of the clock in order to shift / sample the data.

4.4.9 - Serial Communication - I2S

Inter-IC Sound (I2S) communication is an electrical serial bus interface that's used for connecting digital audio devices together. It's used to communicate PCM audio data that is being implemented between circuits within an electronic device. It's made of 3 wires: Serial Clock (SCK), Frame Select (FS), and Serial Data (SD). The SCK line contains a frequency that will be dependent on the number of bits for channel (sample rate), which is done in the following equation: Frequency = Sample Rate X Bits per channel X number of channels.

4.5 Speakers

For our project, we require speakers that the consumer can use. For the speaker, it requires the following:

Part	Usage
Dust Cap	Protects voice coil from dust
Surround	Lining that will connect the basket to the diaphragm
Pole Piece	Directs voice magnetic field
Cables	Wires that connect to voice coil (within input source)
Magnet	Used in conjunction with other components
Diaphragm	Allows the air in the surroundings to move. This composes sound
Spider	Suspension that keeps tension on the voice coil, allowing it to move. Reduces vibration that’s unnecessary
Former	Material that coil is wrapped around
Top Plate	Material made of iron that's required for speaker to function
Basket	Holds speaker together, held by speaker case
Bottom Plate	Holds magnet and pole piece
Voice Coil	Moves diaphragm via magnetic field (conducted by a current within a wire)

With these parts, electrical signals (from a source that's external) will be brought to the speaker through the speaker wire. This will coil around the voice coil. With the voice coil, it is connected to the diaphragm, which is in turn connected to the metal outer basket frame.

With our suggested stereo speaker, the consumer will be able to create different type of environments that will aid them in their sleep. For example, if the user has a preference of sleeping to rain noises, the speaker will provide the means to produce this noise. Additionally, this speaker will support the alarm feature of the pillow. The controls of the speaker, as well as the charging port, will be visible on the outside of the pillow (most likely on the sides). The pillow will be designed in a way where the controls of the speaker will allow the user to be comfortable and not sleep on the controls. This will prevent unwanted changes to the user's ambience, as well as make the user more comfortable while they sleep. The team requires a speaker that lasts for a long time to be a supporting feature of the ambience the user is looking for. Based on a small survey Jason conducted for this project, users would prefer to wake up to their customizable ambience, than have an ambience that isn't present in the user's awakening. For example, if the user would like to have a “thundering” ambiance, they want to hear this ambience while they sleep, as well as when they wake up. Due to this, a speaker that can last longer than 8-10 hours is most preferred for our pillow design.

Our team requires a stereo speaker that is both compact and effective. The reason it needs to be compact is because it needs to fit the already compact pillow we plan on utilizing. What Jason will do is research the different stereo speakers that will be the most compatible to our design. After researching the different types of stereo speakers, Jason will compare them to understand the advantages and disadvantages of each one. Once the comparison phase is over, Jason will choose the best speaker for this project.

4.5.1 JBL Go 3

This speaker, being covered in fabric that's durable, will work well with our pillow design. If our speaker had a naked design, it would be easier for the other pillow components to intrude with the speaker. In addition to this, the speaker has a IP67 water-resistance rating. This synchronizes with the water-resistant feature our pillow has. A challenge we might run into is the battery life. Research will be conducted to understand how to charge the battery of the speaker using the wiring of our design.

4.5.2 Bose Soundlink Micro

The Tribit StormBox Micro is a great suggestion due to its tiny size and bass effectiveness. Compared to the other speakers, the Bose Soundlink Micro provides more bass capabilities than the other speakers I researched. Additionally, this speaker is fully waterproof, matching the water-resistant feature the pillow design will contain. Similar to the JBL Go 3, research will be conducted to understand how to charge the battery of the speaker using the wiring of our design.

4.5.3 JBL Clip 4 Mini

The fabric cover of the JBL Clip 4 Mini has a 40-millimeter audio driver with a strong frequency response rate. A problem our team can potentially run into are the controls.With these controls, it can be difficult to implement with the pillow's design because if we add the speaker to our pillow, the controls would be hard for the user to use. This is because the controls are directly on the speaker output. This speaker, compared to the others, has one of the longest battery life (10 hours). To charge, it requires a USB-C connector to recharge its battery.

4.5.4 Anker Sound core 3 Mini Bluetooth Speaker

This speaker has the strongest waterproof features compared to the other designs with IPX7 rating. Also, this speaker is the longest lasting out of the other speakers I looked at (15 hours). This works very well for our pillow design because we want the user to have the availability to have their ambience being portrayed for the whole duration of their sleep. A "con" to this speaker is that there is a small amount of bass. If the user prefers to have a strong bass to their sounds or music, it won't be displayed as much as the other speakers the team researched.

4.5.5 Speaker Comparison

Speaker	Battery Life	Waterproof?	Bass Quality	Size

JBL Go 3	Charging time: 2.5 hours Playtime: 5 hours	Yes - IP67	Low-Medium quality	Dimensions: 8.6 X 6.9 X 4.0 in
Bose Soundlink Micro	Charging time: 4 hours Playtime: 6 hours	Yes - IP67	High quality	Dimensions: 3.87 X 3.87 X 1.37 in
JBL Clip 4 Mini	Charging time: 3 hours Playtime: 10 hours	Yes - IP67	Medium quality	Dimensions: 1.8 X 3.4 X 5.3 in
Anker Sound Core 3 Mini Bluetooth Speaker	Charging time: 2-3 hours Playtime: 15 hours	Yes - IPX7	High quality	Dimensions: 2.8 X 2.8 X 3.3 in

4.6 Microphone

When including a microphone to the ESP32, we need to consider first if we're working with Bluetooth or Wireless. For Bluetooth, a profile we could use is the Hands-Free Profile (HFP). This profile is commonly used to communicate devices to mobile phones. This would be useful for this project as we'll need the ESP32 to transmit the data to the app. In our case, the ESP32 will act as the hands free, while the mobile device will act as the audio gateway.

With HFP we may encounter some issues. First, we must deal with the HFP codecs. The HFP codecs are limited to; Continuously Variable Slope Delta Modulation (CVSD) with 8kHz sample rate or modified low-complexity subband codec with 16 kHz sample rate. For low quality audio we would need 4kHz, high quality is 8kHz, while the best quality is 20kHz. Therefore, the codecs we can use are of decent quality but not the best.

An alternative would be to use WIFI if the ESP32 has WIFI compatibility. We can send the data with TCP sockets. Additionally, for this we can also use a virtual audio device such as Black hole that that will read audio from an output device to an input device. This means audio can be played to the audio Blackhole device and other software can record the same audio from the Blackhole input device. The benefit of setting the microphone this way is that the quality is significantly better as we get the full audio bandwidth.

When it comes to the microphone itself, microphone boards such as the MAX9814 and MAX4466 can be used. These integrate both the microphone and audio amplifier. They output an analog signal which is fed to the ESP32's analog to digital converter (ADC). Unfortunately, the MAX9814 and MAX4466 microphones are more susceptible to power supply noise, while the MAX9814 being the best between the 2. Therefore, for high quality and low noise input then the built in ADC in the ESP32 is not the best.

Furthermore, we can use I2S MEMS breakout boards such as INMP441 or the SPH0645. These boards integrate the audio amplifier, ADC, and the I2S interface. Therefore, the output from these boards can be fed directly into the ESP32 without using the integrated ADC from the ESP32. From both microphones, the INMP441 is the better choice as it produces a good signal and doesn't need any power filtering. Compared to the MAX9814, the INMP441 is significantly less noisy.

4.7 Temperature Sensor

Given that this project uses electronics in the pillow, it is important to track the pillow's temperature. This is also a safety concern as the electronics will be in close distance from the user's head. Therefore, we have decided to include 1 or 2 sensors at most to display the temperature data. The two temperature sensors that we are considering are the following: DHT11 sensor, and the DHT22 sensor.

These sensors include a chip that does analog to digital conversion and returns a digital signal with the temperature and humidity. Therefore, they are easy to use with many microcontrollers, and are popular among the Arduino community.

	DHT11	DHT22
Temperature Range (Celsius)	0 – 50 (+-2)	-40 – 80 (+-0.5)
Humidity Range	20% - 90% (+-0.5%)	0%-100% (+-2%)
Operating Voltage (DC)	3V – 5.5V	3V – 6V
Current Supply	0.5mA – 2.5mA	1mA – 1.5mA
Sampling Period (Seconds)	1	2
Data Display	Temperature: 1C Humidity: 1%	Temperature: 0.1C Humidity: 0.1%
Size	15.5mm x 12mm x 5.5mm	15.1mm x 25mm x 7.7mm
Price ($)	$5-$7	$10-$12

From the table above, we can see that the DHT11 sensor has a smaller temperature and humidity range. While the DHT22 is more accurate, for our specific project we don't need the most accurate readings. A +-2C error is good enough for our project. We can also see that with the DHT11 we can request a reading every second while with the DHT22 it's every 2 seconds. There's also a slight price difference between the two, with DHT11 being cheaper. The DHT22 is also bigger than the DHT11, but both are still considered small.

Additionally, even though they are technically different sensors, the same code can be applied for both sensors. The only thing to change would be to define what specific sensor we would be using in the pillow.

When it comes to the safety of both the user and electronics, we can use the DHT sensor to set a limit on the max temperature that we are willing to have the pillow at. If the temperature reading goes above the max limit we set, then a signal should be sent to the microcontroller to turn off the device.

4.8 Ambient Light Sensor:

An ambient light sensor is needed so that the device can know if it's nighttime. This could be helpful in automatically activating the device at nighttime and conserving the battery during the day. There are different sensors we can use for this case:

4.8.1 Adafruit 161 Photoelectric Sensor:

One sensor we could use is the Adafruit 161 Photoelectric Sensor. In this case, this is a photoresistor, also called light-dependent resistor or photocell. Not only is it used to detect light but also to measure the brightness level of the ambient light. This sensor has 2 symmetric pins. This sensor works in a simple way. The more light the face is exposed with, the smaller the resistance is. So, by measuring the photoresistor's resistance we can know the brightness of the ambient light.

Applying this with the ESP32 is also another simple process. The ESP32's analog input pin converts the voltage into integer values (between 0 and 4095. This is called the ADC value, this way we can easily measure the ambient light in the room. For connecting to the ESP32 board, one pin would be directly connected to the 3.3V, and the other pin to an Analog pin on the board such as ADC0/GIOP36.

To use this sensor, connect one side of the photocell (either one, as it's symmetric) to power (for example 5V) and the other side to the microcontroller's analog input pin. Then connect a 10K pull-down resistor from that analog pin to ground. The voltage on the pin will be 2.5V or higher when it's light out and near ground when it's dark.

4.8.2 DFRobot's Gravity: Analog Ambient light sensor:

Another sensor option is to use Gravity: Analog Ambient light sensor for Arduino from DFRobot. This module includes a PT550 light sensor. This is an analog sensor with an illumination range from 1 Lux to 6000 Lux with a response time of 15 us.

When it comes to wiring, it has 3 wire terminals: GND, VCC and Signal. The module has a voltage range from 3.3V to 5V. Therefore, we can power it with 3.3V and the Signal pin will connect to a pin on the ESP32 that has Analog reading support. This sensor ends up working like the Adafruit sensor, and it can output the result of the ADC, which is a value ranging between 0 to 4095. So, if the light source is covered then we should see the output value lower.

4.8.3 TEMT6000 light sensor:

Furthermore, another cheap and simple ambient light sensor is the TEMT6000 light sensor. Like the previous example, this sensor also includes 3 pins: GND, VCC and Signal. The sensor changes its resistance depending on how much light is directed to the sensor. The signal pin would connect to an analog pin on the ESP32. Like DFRobot's light sensor, the TEMT6000 sensor also measures illuminance (lux).

When getting the illuminance in Lux, we will first need to convert the measured voltage to current flowing across the TEMT6000 sensor. This current is also equal to the current flowing across the 10kΩ resistor in the voltage divider circuit, which is I = adc_value/10000kΩ.

According to the TEMT6000 data sheet, there is a proportional correlation between current and illuminance: Every 2 uA of current correlates to 1 Lux in illuminance.

Some important things to note are:

The default voltage range of the ADC for the ESP8266 and ESP32 are from 0 to 1.0V. Therefore, we won't be able to measure above 200 Lux using the default setup.

For the ESP32, a voltage attenuation can be set up. On the other hand, when it comes to the ESP8266, unfortunately the hardware will need to be modified to decrease the voltage. One solution would be to create another voltage divider on the SIG pin to divide the analog voltage by a constant value.

4.8.4 OPT3001 Digital Ambient Light Sensor:

Another ambient light sensor we can consider is the OPT3001 digital ambient light sensor. This sensor measures the intensity of visible light. The OPT3001 sensor is a single-chip lux meter, measuring the intensity of light as visible by the human eye. The precision spectral response and strong IR rejection of the device allow it to accurately measure the intensity of light.

The OPT3001 offers measurements from 0.01 Lux up to 83k lux without having to manually select full-scale ranges by using the built-in setting feature. This is a capability allowing light measurements over a 23-bit effective dynamic range.

The control and interrupt system features autonomous operation, allowing the processor to sleep while the sensor searches for appropriate wake-up events to report via the interrupt pin. The digital output is reported over an I2C- and SMBus-compatible, two-wire serial interface.

The OPT3001 has 3 pins: VCC, GND, SDA, and SCL. This ambient light sensor seems to have a more complicated process than that of the previous examples that use ADC to provide light measurements. On the other hand, this sensor is significantly more accurate.

Overall, these 4 ambient light sensors are cheap and simple options to measure the ambient light directed to the pillow and can be effective to measure when the user has their head on the pillow. For this project, a highly accurate sensor is not needed, but one that can easily detect when light is being blocked. By putting their head on the pillow, light is blocked from the sensor, therefore, we can detect when the user is going to sleep.

4.9 Accelerometer

4.9.1 Preface

The accelerometer will be used to track movement of the user as they sleep. This data will then be fed into a software algorithm which will compile this data and translate it into a history of the user's sleep cycles throughout the night. This will help users track how well they are sleeping on average.

4.9.2 Cost Analysis

The table below provides the cost analysis for variations of the same component from different vendors.

Model	Cost	Axes	Power	TX	Vendor
Memsic 2125	$20.89	2	3V - 5V	I2C	Radio Shack
ADXL-335	$10.95	3	1.8V - 3.6V	I2C	All Electronics
ADXL-335	$18.25	3	3V - 5V	I2C	Amazon
ADXL-345	$9.99	3	2V - 3.6V	I2C, SPI	Radio Shack
ADXL-345	$10.95	3	2V - 3.6V	I2C, SPI	Circuit Specialists
LIS3DH	$7.97	3	3V - 5V	I2C, SPI	Amazon
MPU-6050	$1.69	3	3.3V - 5V	I2C	Amazon
MPU-6050	$3.00	3	3.3V - 5V	I2C	Amazon
MPU-6050	$3.33	3	3.3V - 5V	I2C	Amazon
MPU-6050	$24.95	6	3V - 16V	I2C	Circuit Specialists
MPU-6050	$24.99	6	3V - 16V	I2C	Radio Shack
MPU-9250	$7.50	9	3V - 5V	I2C	All Electronics
MPU-9250	$9.10	9	3V - 5V	I2C	Circuit Specialists

4.10 Power System

4.10.1 introduction

For our project, we want to be able to plug in the Snoozie into the wall and power it and charge a battery at the same time so it will be able to be used without being plugged in. We want the battery to be integrated within the casing of the pillow so that it fits sleekly and naturally within the pillow portion and will not be noticeable when the user is laying on it. We want the battery to last the duration of the user's sleep cycle and then some to allow the user to be able to use the pillow without the need to plug it into the wall for comfort reasons and if the user doesn't sleep near an outlet.

We want the user to not need to worry every night about charging the Snoozie, so we ideally want the batteries to be able to last for a couple of days if used during the nights only. To do this we will need to look at various battery solutions so that we will be able to achieve this. We will also need to look at the way that we will be charging our pillow, and how we can make it as convenient as possible for the user to not need to worry about clunky cords taking up space.

We want to make sure that each component that is contained within the pillow will receive sufficient power to function within each of their specific roles, so we will need to provide different voltages for certain parts. The most common voltages that we will need to have are 5V and 3.3V. With that said, we will also need to make sure that our power source will provide enough amps to the project as well. In the coming sections we will research parts and schematics and see what parts can give us the best possible product

4.10.2 AC to DC Conversion

To start off powering this project, we will have to deal with getting power from the wall. The wall power is going to be 120V so to go from that to DC, we will need to convert it in some way. The most convenient option for this is to use a USB wall adapter that is extremely common for a person to own. Everyone who has a modern phone either has used one and most likely owns one, so this choice will make using the Snoozie tremendously easier for the customer. We can look at the power that these adapters can output and make sure that these will be able to output the correct power that we will need.

Here are some of the different wall adapters that are on the market. Note: we will want to make sure that the project will be able to be powered by an average one to make sure the user can interchange them with one they have if they need to.

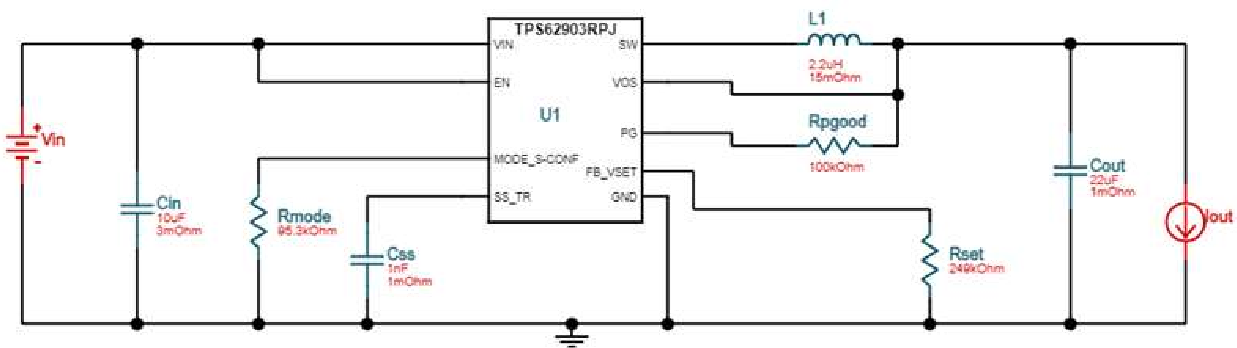
Name	Voltage	Max Amp	Price	Seller
UL Certified USB Wall Charger Power Supply	5V	1 Amp	$8.99	Amazon.com
Quick Charge 3.0, Anker 18W 3Amp USB Wall Charger	5V	3 Amp	$15.99	Amazon.com
Anker Power Port C 2	5V	2 Amp	$24.97	Walmart.com

4.10.3 DC to DC Conversion

Our input from the wall adapter will be 5V, so that will be able to power the bulk of the components, but for most of the sensors that we have, we will need to have an input voltage of 3.3V to be able to run properly. To do that we will use the Texas instruments webench to find a suitable schematic that will accomplish what we need for the sensors.

When going to the webench, we can enter the min and max voltage input that we will be using, which is going to be the 5V from the USB wall adapter, and we can specify the output, which is the 3.3V that the sensors will need to operate. We can also specify if we want our schematic to be balanced, low cost, high efficiency, or have a small footprint. For this project, we chose to go with balanced because we want to keep our options open. The design that we came up with is the TPS62903.

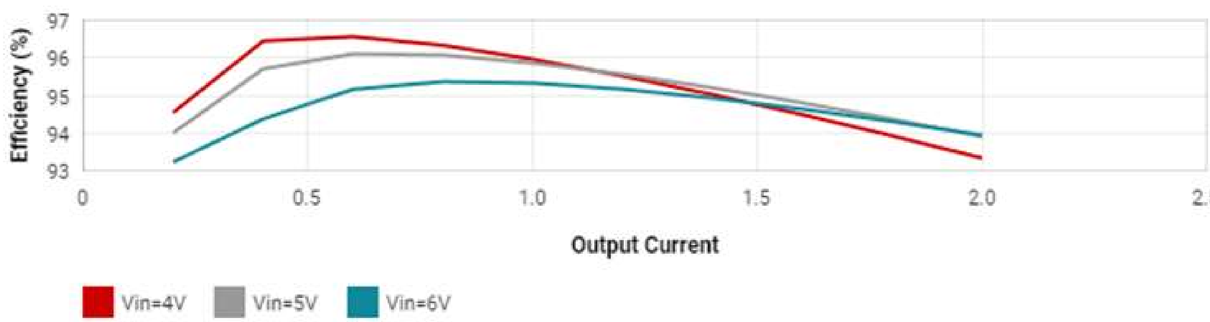
TPS62903



BOM

Part	Manufacturer	Part Number	Quantity	Price ($)	Footprint	Description
Rmode	Vishay-Dale	CRCW040295K3FKED	1	0.01	3	Resistance: 95.3 kΩ Tolerance: 1.0% Power: 63 mW
Rset	Vishay-Dale	CRCW0402249KFKED	1	0.01	3	Resistance: 249 kΩ Tolerance: 1.0% Power: 63 mW
Cin	Kemet	C0805C106K8PACTU	1	0.03	6.75	Cap: 10 µF Total Derated Cap: 4.4 µF VDC: 10 V ESR: 3 mΩ Package: 0805
Rpgood	Vishay-Dale	CRCW0402100KFKED	1	0.01	3	Resistance: 100 kΩ Tolerance: 1.0% Power: 63 mW
Css	MuRata	GRM155R61A102KA01D	1	0.01	3	Cap: 1 nF Total Derated Cap: 1 nF VDC: 10 V ESR: 1 mΩ Package: 0402
L1	TDK	VLP8040T-2R2N	1	0.22	113.42	L: 2.2 µH DCR: 15 mΩ IDC: 6.7 A
U1	Texas Instruments	TPS62903RPJR	1	0.651	7.5	
Cout	MuRata	GRM21BD70J226ME44L	1	0.1	6.75	Cap: 22 µF Total Derated Cap: 15 µF VDC: 6.3 V ESR: 1 mΩ Package: 0805

Efficiency



We believe that this schematic will be sufficient in making sure that we have enough power for the sensors that we will be using. As you can see from the graph, the efficiency will stay above 93% for up to 2 amps. Which should be enough for what we are doing for this project.

4.10.4 Battery Power

As for the battery for the project, we will need to keep this inside the body of the pillow and be able to fit sleekly inside the pillow so the user will not feel it while using the pillow at night. To do this we first need to find the size and profile that the battery will need to be, so when choosing we will need to take that into account. We also need to make sure that the battery will be able to hold a sufficient amount of charge to last a while. We don't want the user to constantly need to charge the pillow every day, so we aim for the pillow to last about 3-5 days if used when sleeping/ off. We plan to reach that with a large battery capacity along with utilizing low power modes in the MCU.

Along with those things, we want to make sure that these batteries are safe to use. These batteries are going to be inches away from the user head, so we want to make sure that the batteries will not explode on the user. Here are some of the batteries that will fit all of the specifications that we put out

Name	Voltage	Capacity	Size	Price	Seller
Lithium (NCM) 14500 Rechargeable Cell	3.7V	800 mAh	13.9±0.2mm x 48.5±0.5mm	$4.05	Batteryspace.com
EMB 3.7V Lithium-ion Polymer Battery	3.7V	2000 mAh	34.5 X 56 X 10.6 mm	$14.35	Amazon.com
LiFePO4 18500/(18490) Rechargeable Cell	3.2V	1000 mAh	18.7mm x 49.5mm	$4.70	Batteryspace.com

4.10.5 Charging System

To charge the batteries, we want to make sure that the 5V input from the wall adapter will have sufficient power to recharge the batteries and power the system. We will have to have a module that will be able to charge the battery with the 5V input. We can find some suitable modules that will fit our needs online, here are some of the ones that we were able to find.

Name	Amp	Price	Seller

Maker Focus 10pcs TP4056 Charging Module	1 Amp	$9.99 for 10	Amazon.com
HiLetgo 3pcs TP4056 Type-c USB	1 Amp	$5.99 for 5	Amazon.com
DZS Elec 4pcs 18650 Lithium Battery Charging Boost Mobile Power Supply	2 Amp	$8.59 for 4	Amazon.com

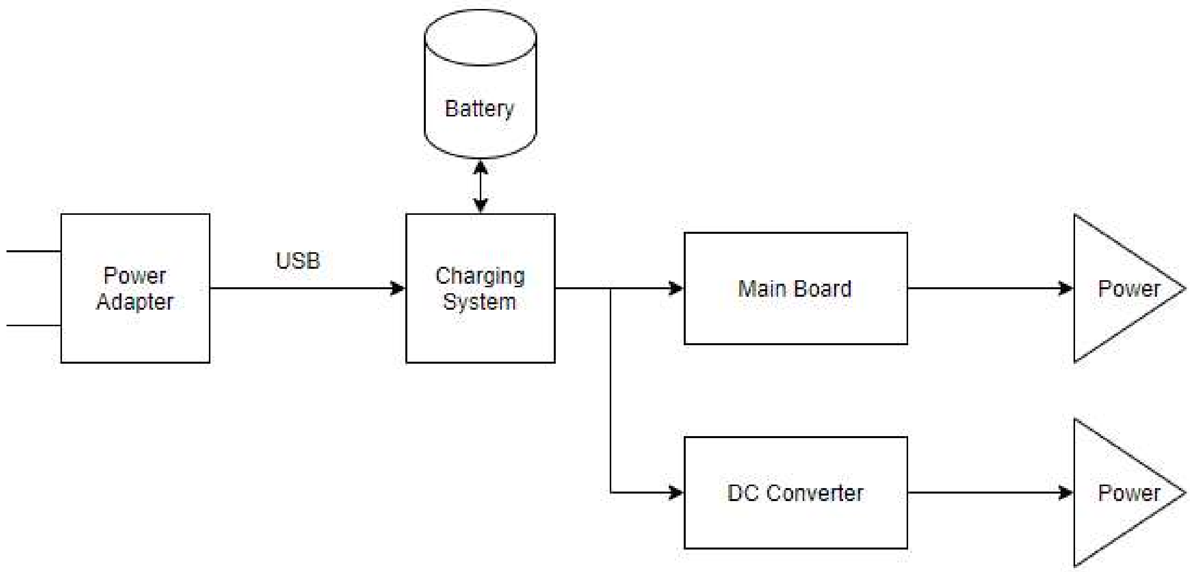
4.10.6 Charging Cable

The cable that we are going to use from the wall adapter to the Snoozie will be a simple USB cable that is widely used by people. We want to make sure that the cable that comes with the pillow will be long enough to make charging the pillow easy, but other than that, there is nothing else that we need to consider for this part of the pillow.

Name	Length	Price	Seller
USB C Cable, 3A Fast Charging	6.6 ft	$5.99 for 2	Amazon.com
Amaitree USB C Cable	6.6 ft	$9.28 for 3	Amazon

Any of these power chords will be perfect to power the Snoozie. With that said, the chord that we are using should be able to be switched out for any usb cable that the user should own. It is important to make sure that the standard of the power transmission for the cable is sufficient enough to transmit power to the project.

4.10.7 Diagram



4.11 Pillows

4.11.1 - Introduction

The foundation of our project is the type of pillow we use to house our electronics. The purpose of a pillow is to keep the upper body in alignment during sleep, making sure pressure is relieved and counterbalancing of the points in the body is present. A couple of the most important things in choosing a pillow is: the comfortability and the size. For comfortability, we need to make sure the user has access to the prevention of common forms of neck and back pains. Additionally, we need to factor in the user's forms of joint pain. For size, we want to make sure the pillow has enough length and width to contain the electronics required for this project. We are looking for larger-sized pillows as opposed to smaller ones, due to the fact that smaller pillows may not be sufficient enough for what we need in regards to space. In this section, the research conducted on specific pillows we want to use will be displayed, as well as a comparison between them all.

4.11.2 Naturepedic Organic Cotton/PLA Pillow

The first pillow we wanted to research was the Naturepedic Organic Cotton/PLA Pillow. This pillow has a 300 thread count with 100% organic cotton outer fabric, with pure PLA batting on the inside. A benefit of this is that it is one of the more comfortable pillows we can provide to our consumers. Also, if the consumer is an advocate towards the betterment of our environment, the pillow is derived from 100% renewable resources. With PLA, our group doesn't need to worry too much about our equipment causing problems with the pillow, where PLA doesn't contain or off-gas harmful chemicals. This gives our group more leeway into the manipulation of the pillow

4.11.3 ComfyComfy

The ComfyComfy pillow is made of soft and durable organic 10 oz cotton twill fabric. Additionally, it is filled with buckwheat hulls that are cleaned without chemicals or fumigants. A potential problem our group may run into when designing our Snoozie with this pillow is that the material may not be compatible with the mechanical equipment we want to use for our design, leading to safety concerns to our consumers. In order to verify this, our group will conduct testing to determine if there are any safety hazards. Something our group needs to consider when choosing our pillow of preference is the price of the ComfyComfy, as it is the most expensive option compared to the other pillows

4.11.4 eLuxurySupply (Down Alternative)

The eLuxurySupply (Down Alternative) is a pillow that has more flexibility for different styles of sleepers. The pillow works well for back, side, and stomach sleepers, where the medium-density pillows provide a soft, gentle yet firm lift for any type of sleeper. The pillow has 100% cotton casing and is pliable. Our group enjoys the flexibility of this pillow, since our main goal of the Snoozie is to make the user be as comfortable as possible by allowing the user to customize their ambience.

4.11.5 Purple Pillow

A feature the Purple Pillow provides is the "Purple Grid" which is a soft and supportive feature of the pillow that cradles the head and neck with No Pressure Support. Additionally, the Purple Pillow is durable, cool, and breathable. Through research, the Purple Pillow may be the most comfortable option compared to the other pillows. The only potential problem that may arise is if the Purple Grid is compatible with the equipment we want to use. We will determine this through the testing phase of our project.

4.11.6 Utopia Bedding Gusseted Pillow

A huge advantage of the Utopia Bedding Gusseted Pillow is the price. This pillow has the cheapest price compared to other pillows, where it can potentially provide the same amount of comfort (will be determined through testing). The pillow contains poly fiber filling, a feature we need to ensure is compatible with our mechanical equipment. Additionally, the pillow provides more customization to our user, allowing the following colors to be implemented onto the pillow: blue, gray, dark gray, and white lining. The customization is important to consider in our design because our main goal is not only comfortability, but personalization. This is whether the personalization is centralized through ambience, or by the look of the Snoozie.

4.11.7 - Pillow Comparisons

Name	Size	Type	Price	Loft
Naturepedic Organic Cotton/PLA Pillow	20" X 26"	Organic Cotton	~$80.00	Adjustable
ComfyComfy	20" X 26"	Cotton	~$105.00	High
eLuxurySupply (Down Alternative)	20" X 26"	Down Alternative	~$60.00	Adjustable
Purple	24" X 16" 3"	Latex-Like Polymer	~$100.00	Low or High
Utopia Bedding Gusseted Pillow	18" X 26"	Polycotton	~$25.00	High

4.12 PCB Design

4.12.1 Board Design

When we are initially designing the PCB board, we are going to have to take something into account. One of the most important things that we need to consider is making sure that the PCB will take a sleek form so that it will be able to fit within the pillow without the user being able to feel it while they are laying on it. The PCB is going to be one of the main parts that will be present in the pillow so size will be an important factor in it.

With all of the components that we are adding to the pillow, we want to make sure that we have a low-cost PCB. To be able to design this, we will be using some PCB design software such as Fusion 360 or KiCad. We will connect all the working components and electronics to this board including the microcontroller and wire all external components to it so that all the parts can communicate quickly and easily.

4.12.2 Board Manufacture

Once we have the design laid out, we can then consider how we will manufacture the PCB. We are looking at different places that will manufacture the PCB and even populate the board to save time and do some of the harder soldering tasks that we will need done.

4.13 Software Development Tools

4.13.1 Preface

To communicate with the smart pillow, we will need an app that can transmit and receive messages from the microcontroller in the pillow. The app should offer the user the ability to view their sleep data and much more. When it comes to building the app itself, we must consider if we want an android app, iOS app, or both. We will investigate these 3 possibilities.

To implement the Snoozie, there are two main software components which need to be addressed. The first of these is programming the microcontroller to handle all inputs and outputs of the device. This includes translating the data received by the microcontroller (either through I2C or SPI) into more easily handled figures. The second software component is the application which user's will leverage in order to interact with the device.

4.13.2 Android Studio

One tool we will be able to leverage is Android Studio. Android Studio is the official IDE of Google's android operating system based on IntelliJ IDEA. In Android Studio, we can use either Java or Kotlin to build the app. With an emulator implemented in the software it is extremely beneficial to test the code in different android devices. Therefore, in

Android Studio, we have a flexible Gradle-based build system, build variants and multiple APK generation.

The purpose of this tool is to improve the software development process for programmers, making it easier to create an android application from instantion to production. This is an IDE specifically designed for development of android applications, and offers a wide array of features such as emulation, testing frameworks, GitHub integration, versioning, and much more. In addition, the editor offers plugins which unlock even more tools that can assist developers.

4.13.3 Xcode

For an Apple only app, we would need to use Xcode which is Apple's office IDE used to develop MacOS, iOS, etc. Making the app in iOS can prove itself to be difficult given that we need a macOS device to use it. If on Windows, the best option would be to use a virtual machine such as Virtual Box and run a MacOS image on it.

4.13.4 Visual Studio

Additionally, Visual Studio is Microsoft's IDE where we can implement Swift to a certain extent. Xamarin can also be used in Visual Studio which is another option if we want the app to work in both Android and in iOS.

4.13.5 Java

For writing our Android application, we will be using Java as the primary programming language. One advantage of this is that Java provides a number of pre-existing libraries that we can leverage to import items such as timers and calendars. In addition, Java is an object-oriented programming language, meaning it allows us to create custom classes which can be used to supply any additional functionality not already provided by Java's pre-existing libraries.

One way we plan to utilize this is by creating objects for each hardware input and output of the device. Within each of these classes we will be able to wrap-up any fields and methods specific to the functionality sought after for that device. This includes things like reading in sensor data, converting data into a parsable format, and enabling or disabling our sensors.

Similarly, custom classes can also be implemented for any software components and functionality native to the application . This includes non-hardware functionality such as sleep-report generation, handling of the audio libraries, and sleep cycle tracking.

Finally, Java allows for easy software testing using the JUnit library. This will allow us to run diagnostics on the pillow during production, and allow us to troubleshoot any hardware issues that may come up during prototyping. JUnit will also handle any testing of the UI or any of non-hardware functionality discussed above.

4.13.6 Arduino

Since we are using an Arduino board, our microcontroller will be programmed using the Arduino programming language. This is a simple, easy-to-use language that provides just enough functionality as this project demands. Using this, we will be able to handle inputs from all of the sensors on the device, and package the data for export via the device's bluetooth transceiver (an output also handled by Arduino). The exported data will then be handled by the application for generating things like sleep-wellness reports. In addition, Arduino will handle all our device's outputs including the device's stereo speakers.

4.14 Software Framework

First, we must understand what a framework is before we use one. A framework is an organized set of tools and libraries that serve as a foundation to build software on. We will investigate Xamarin and React Native, as these are two popular frameworks that have Android and iOS capabilities.

We won't necessarily have to use any of these frameworks, but they are worth considering and looking into incase we do decide to work with them. The following frameworks offer the chance to expand the compatibility of our app

4.14.1 Xamarin

One of the best frameworks we could use to build native applications for Android, iOS, and Windows is Xamarin. Xamarin is a mobile app development framework that is .NET based with native API access. It is intended to be a cross-platform and open-source app building software. We can easily run Xamarin on Visual Studio and write the app in C#. The main benefit of using Xamarin is that it is cross-platform, therefore, we wouldn't have to worry about the issues that come with making the app work on different platforms.

4.14.2 React Native

Additionally, another popular framework is React Native. React Native is an open-source UI software framework made by Facebook. It can be used to make apps in Android, iOS, macOS, etc. React Native mainly uses JavaScript, while Xamarin uses C#. React Native is friendly, as it allows users to choose their preferred IDE/text editor.

Overall, both frameworks are powerful tools that allow for apps to be built in Android and iOS. At the end of the day, picking between the two is just a decision of preference.

4.15 Software Development

The Snoozie app is an important step in the project. The app will transmit data such as: The sleep cycle analysis, the sound/music playback, and the alarm system. It is important that we focus on those 3 components as they are the main objective of the Snoozie app.

4.15.1 Start-up Design Layout

The user will have an easily understandable and simple interface to use to operate the device and to view the data from a mobile device. As stated before, the app will have the ability to receive data from the sensors connected to the board in the pillow and organize the data nicely so that the user can see their sleep cycle.

When the user enters the app, they will be greeted with the project logo. The main page will have a brief and concise explanation on how to use the app, and what is offered to the user. After that, the user can select from three different tabs: Alarm, Data, and Music.

4.15.2 Start-up Design Diagram

4.15.3 Alarm Page Layout

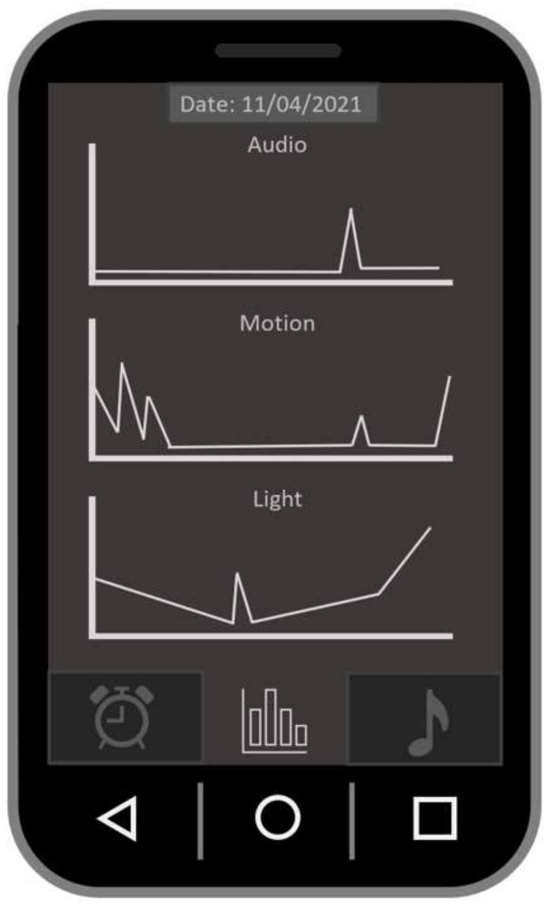
In the alarm section, the user should be able to set a new alarm to any time of the day. Additionally, the alarm library is shown below in the same page where the user should have access to all the previous alarms made and be able to activate, deactivate them. Finally, for this section, the user will be able to edit each alarm, they will have the ability to modify a pre-existing alarm's time, alarm tone, and if they decide they want to delete that specific alarm entirely.

4.15.4 Alarm Page Diagram

4.15.5 Sleep Wellness Page Layout

In the Data section, the user will be provided with an organized graph of the data gathered by the sensors every night the pillow was active. The pillow will keep track of the dates and organize the data for each day. Therefore, the user should be able to select a date from which they want to view the data for. This is extremely useful, as it allows the user to track their process and allows them to see what changes should be made to have a better sleep cycle.

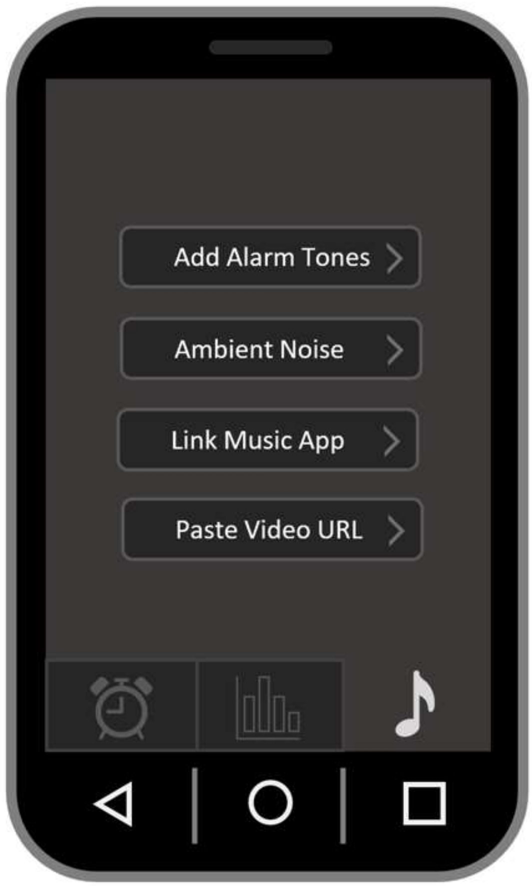
4.15.6 Sleep Wellness Page Diagram



4.15.7 Audio Page Layout

In the Music/Audio playback section, the user will be able to modify the default tone set to every alarm. The user will be provided with some already installed tones that can be chosen. The user should also be able to select a custom tone downloaded on their mobile device. Furthermore, the music section should also offer a selection of White Noise where sounds will be played through the speakers of the pillow. The user will also be able to add a YouTube link if they want a specific video to play. Finally, the user will be able to link their music player app such as Spotify and play music of their choosing through the pillow's speakers.

4.15.8 Audio Page Diagram



4.16 Sleep Recommendations

In trying to better understand the problem our product attempts to solve, we researched the two main sources that affect sleep the most: audio and ambient light. Below describes the problems these sources can pose to a person's sleep, as well as methods users can take to remedy these effects.

4.16.1 Audio

In researching the effect of sound on sleep, it has been shown that noise can have both a positive and negative effect on how well a person sleeps at night. For instance, sudden, abrupt sounds can disturb someone's sleep cycle (most often children), even if the person does not wake up. Examples of these sounds are traffic, house alarms, doors closing, etc. These are most likely to wake someone during stage 2 sleep (non-REM).

There are methods to combat this however. Research shows that having a steady, constant background of noise can alleviate intermittent noise caused by the sources listed above. This can lead to a more peaceful sleep in both children and adults. Worth noting, research has also shown that the severity of effects from both sudden and passive noises vary from person to person, and there isn't a "one size fits all" solution to the problem.

4.16.2 Ambient Light

According to research, light has the biggest impact on a person's sleep. The main thing to understand about the effect light can have is a concept called circadian rhythm. This is essentially your body's internal clock, telling your brain what time of day it is based on the perceived levels of light your eyes take in. This is determined by both natural and artificial sources of light and affects appetite, energy, and most importantly sleep.

To remedy this, researchers recommend reducing artificial light from sources such as smartphones and televisions in the evening. Specifically, blue light has a very strong impact on your circadian rhythm. These light sources decrease the amount of melatonin produced by the brain close to bedtime, causing your body to assume it's still daytime. In addition, throughout the night excessive exposures to light can disrupt transitions between sleep cycles and decrease time spent in REM sleep.

Some other tips to reduce the amount of ambient light taken in during bedtime is using blackout curtains. The main goal here is to make the area you sleep in as dark as possible. In addition it is also recommended to turn off all light sources in your sleeping area and wear an eye mask when you go to bed.

4.17 Sleep Cycle Algorithm

Sleep is a very intricate process. According to research, most sleep tracking apps can only make a rough estimate as to how much a person is actually sleeping. A lot goes into making these predictions such as brain activity, breathing patterns, heart rate, temperature, muscle and eye movements.

The most accurate way to obtain this data would be through a medical sleep study, where scientists can monitor the activity of the person's brain waves in order to analyze the various stages of sleep. To keep the scope of this project within reason however, (and to ensure convenience and comfort when using the product,) we are only able to use a fraction of the elements that go into tracking a person's sleeping patterns.

Despite their gross simplicity of an incredibly complicated process, sleep tracking apps are still very useful in detecting negative sleep patterns based on the data they collect. Below describes the most common methods used by these sleep tracking apps in order to predict a person's sleep cycles.

4.17.1 Sleep Cycles Defined

Depending on the source, the body transitions between 4 or 5 stages during sleep. For the sake of consistency, we'll assume the 4-stage model.

- Stage 1: Falling Asleep (Non-REM); this is where the body is transitioning from wakefulness to sleep. Usually lasts less than 10 minutes.
- Stage 2: Light Sleep (Non-REM); this is where about 50% of time sleeping is spent. Typically lasts 10 to 60 minutes.
- Stage 3: Deep Sleep (Non-REM); this is the deepest sleep state where any noises or activity in the surrounding environment may fail to wake a person up. Usually lasts 20 to 40 minutes.

- Stage 4: REM Sleep (REM); during this stage, the brain's activity most closely resembles its activity when it is awake. This stage is vital for processing emotions and storing memories. Typically lasts 10 to 60 minutes.

4.17.2 Breathing During Sleep Cycles

- Stage 1: The person's breathing rate begins to decrease.
- Stage 2: The person's breathing decreases further, but becomes more regular.
- Stage 3: The person's breathing decreases again, reaching its slowest rate.
- Stage 4: The person's breathing increases and becomes fast and irregular.

4.17.3 Muscle Activity During Sleep Cycles

- Stage 1: Muscle activity begins to decrease (with possibly some minor twitches).
- Stage 2: Muscle activity continues decreasing; body slows down.
- Stage 3: Muscles are completely relaxed; body transitions into motionlessness.
- Stage 4: Muscles are immobilized; body enters temporary paralysis.

4.17.4 Accelerometer Data

We can use the accelerometer data to analyze the user's muscle activity during sleep. Using a 3-axis accelerometer the device should be able to detect both abrupt movements by the user (indicating a full-wake), as well as more subtle movements like body twitches (indicating stage 1 sleep). As the movement in the accelerometer decreases more and more, it will indicate to our software algorithm that the body is transitioning into the later stages of the sleep cycle.

4.17.5 Pressure Sensor Data

We can use the pressure sensor data to analyse both the user's breathing patterns (this combined with the audio data defined below), as well as the user's muscle activity during sleep. By decreasing the threshold on the sensor, the pillow should be able to detect small variations that come when a person is breathing. Additionally, larger more abrupt changes in pressure would indicate movement, implying that the person is transitioning sleep stages or waking up altogether.

4.17.6 Audio Data

The final sensor that will help us determine sleep cycles is the microphone data. This can be used (along with the pressure sensor) to detect the user’s breathing rate. Audio received by the microphone should mirror a sine wave for stages 1 to 3, with the period increasing as the stages transition. In stage 4, there should be a clear distinction, with the wave following erratic periods and amplitude.

4.18 Pressure Sensor

A pressure sensor is an instrument that contains a pressure sensitive element that determines the actual pressure being applied to that sensor. In order for this to be supported, there are working principles, as well as components that convert the information into an output signal. With our design, we aim to use a pressure sensor to measure the consumer's breathing patterns, as well as their muscle activity during their rest.

4.18.1 Force Sensitive Resistor (FSR)

The Force Sensitive Resistor (FSR) is a group of sensors that detect pressure, whether its physical pressure, weight, or squeezing. A nice benefit of using FSRs is that they're simple to use, as well as cheap to purchase. The FSR is made of 2 layers that are separated by a spacer. Once the sensor is pressed, the more it occurs, the more of the active element dots will touch the semiconductor, which will make the resistance go down. This part is a resistor that will change its resistive value depending on the number of times it's pressed. The value of resistance with the FSR will be in ohms.

4.18.2 FSR Advantages

An advantage of the FSR is that they are sold at a low cost, and are easy to use. In regards to cost, the price of the FSR is roughly $7.00. This is valuable to our design because we want to make the best pillow possible, while aiming to pay the lowest prices. For our budget, this is an amazing price we can work with. For the difficulty of use, the FSR is simple to use thanks to several guides being provided to people who would like to know how to measure the pressure being applied to the FSR. In addition to these guides, there are also YouTube videos that can be viewed in order to visually see how the FSR can be used.

4.18.3 FSR Disadvantages

A disadvantage of using the FSR is that they can be inaccurate. Because of this, our group should only expect to get certain ranges of responses, instead of a specific value to the pressure being applied. Though the FSR can detect weight, it's not a great choice to determine exactly how many pounds of weight is on it.

4.18.4 FSR Statistics

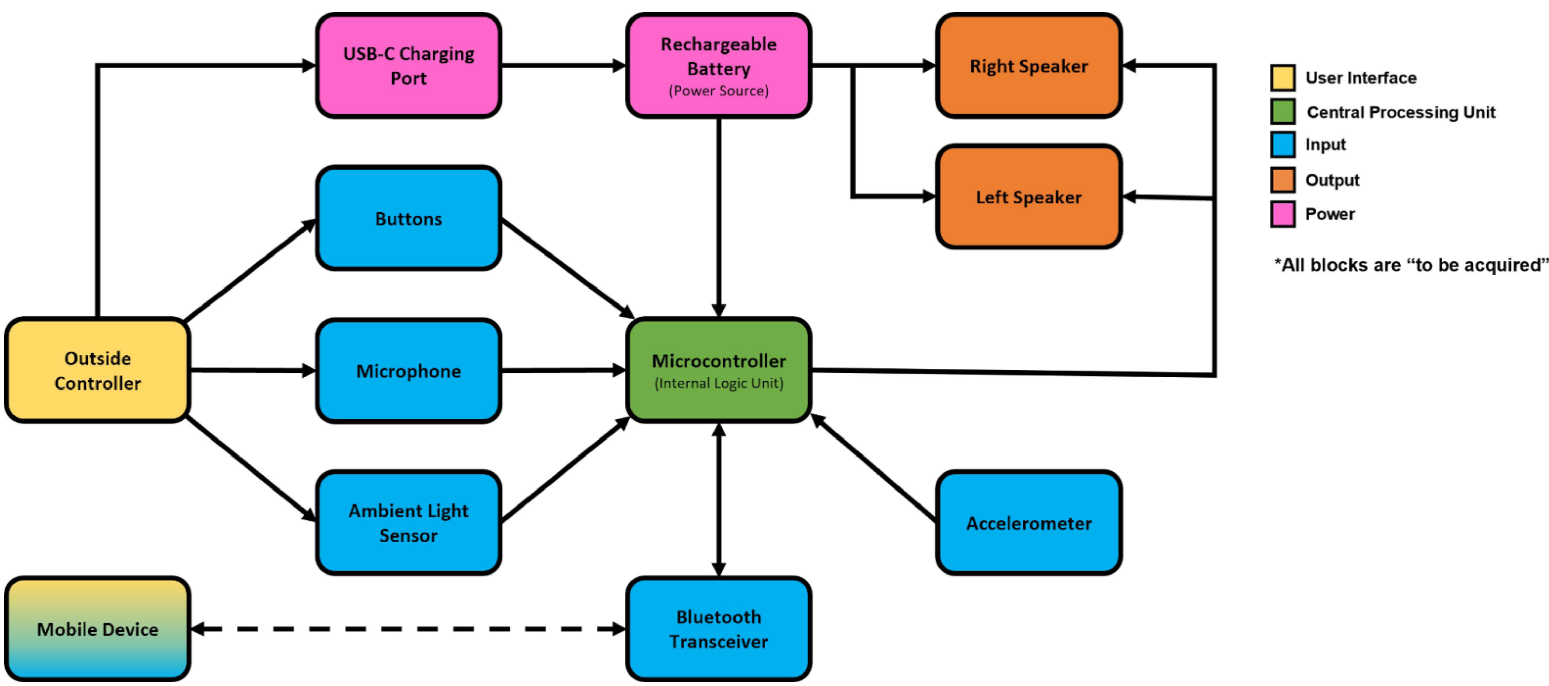
Size	1/2" diameter active area by 0.02" thick
Price	$7.00
Resistance Range	Infinite/open circuit (no pressure), 100K(omega) (light pressure) to 200(omega) (max pressure)
Force Range	0 to 20lb. (0 to 100 Newtons) applied evenly over to the 0.125 sq in surface area
Power Supply	Able to use any power supply. Uses less than 1mA of current (depending on any pullup/down resistors used and supply voltage)

4.18.5 Image of a Force Sensitive Resistor (FSR)

Section 5: Project Hardware and Software Design

5.1 Hardware Design

5.1.1 Hardware Block Diagram



5.1.2 Block Diagram Explanation

For the hardware segment of the project, components can be grouped into five basic categories: user interface (yellow), central processing unit (green), input (blue), output (orange), and power (pink).

Starting with the user interface (yellow), these are components which are exposed to the user and allow them to directly interact with the device. In our design, there are two main ways the user can do this. The first is through the outer control panel (see section 6.1.2). This is a panel that houses buttons, an exposed charging port, microphone, as well as ambient light sensor. This panel and its components directly interfaces with the microcontroller to provide feedback for user input (if button is pressed), collect data on the user's sleeping patterns (passively done by the microphone and ambient light sensor), and supply power to the rechargeable power source.

The second method which users can leverage to interact with the device is through their own hardware, an android smartphone. Pictured above as "Mobile Device", this interfaces wirelessly through a bluetooth transceiver to the device's microcontroller. This point serves as both an input (blue) and output (orange) for the user, allowing them to adjust settings, play music, and access other features through the app (input), or track their sleep cycles, and get a summary of their sleeping data (output).

The final source of input (blue) on the device is the accelerometer. This is housed inside the bottom memory foam layer of the pillow, and is responsible for tracking the user's movement during sleep. (This data is crucial for tracking the user's sleep cycles through the app.)

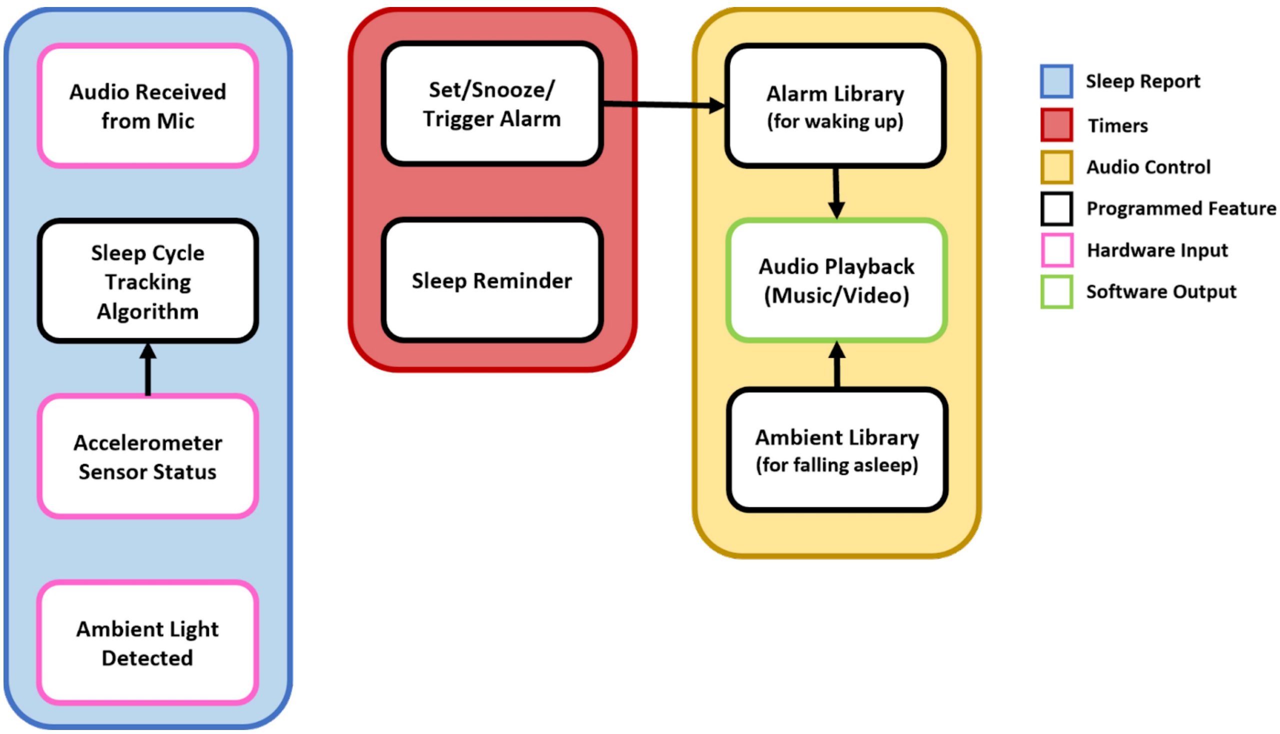
At the center of the device is the microcontroller (green). This is the brain of the device as it handles all the inputs and outputs for The Snoozie. In addition, it also handles some of the business logic used for converting user inputs into outputs (for example, if the user presses the "power button", the device will toggle on or off).

On the right (in orange) we have the two device outputs; these are the left speaker and the right speaker. These interface directly with the microcontroller in order to broadcast audio and control volume, as specified by the user. In addition, these need their own power supply, so they are connected to the rechargeable power source.

Finally, we have the power source (pink). This includes the rechargeable battery (housed inside the device) as well as the exposed charging port on the control panel. Once charged, the battery supplies power to the left and right speaker, as well as the microcontroller and subsequently all connected inputs (excluding the mobile device).

5.2 Software Design

5.2.1 Software Block Diagram



5.2.2 Block Diagram Explanation

For the software segment of the project, there are three basic functionalities the program seeks to achieve: sleep reporting (blue), timing (red), and audio control (yellow). Within each of these groups are individual functions which make up these features. These are broken up into programmed features (black), hardware input (pink), and software output (green).

Programmed features are anything handled specifically inside the app. This includes any logic written and controlled by code. Hardware inputs include any data that is required by the application, and *must* be transmitted from the hardware in order for the feature to work correctly. Conversely, software outputs include any data that is required by the hardware, and *must* be transmitted from the application in order for that hardware feature to work correctly.

The sleep report feature (blue) includes everything related to generating a sleep wellness report for the user. This is meant to show the user how well he or she is sleeping throughout the week. Included in this are three hardware inputs: accelerometer data, microphone data, and ambient light data. While all three of these are used to generate the complete sleep wellness report, the first is fed into an algorithm to make sense of the data before it is presented to the user.

The timing section (red) includes all timer related functionalities. This feature is more passive and includes setting alarms and reminding the user to go to sleep. Notably, this section interfaces with the audio control section (explained below).

Finally, the audio control section (yellow) contains all features related to the transmission of audio to the user, the most important of which is device audio playback. In addition, the application will provide a set of libraries for the user; one set for falling asleep (ambient sounds), and one set for waking up (alarm sounds). This section interfaces with the timers section, calling an alarm sound when the alarm is triggered, and notifying the users when it’s time to go to bed through a tone.

5.3 Hardware Parts Conclusion

The following sections are the specific parts we decided to use for this project:

5.3.1 Microphone

We decided to go with the MAX9814. As mentioned in the microphone section, this board integrates the audio amplifier, ADC. This sensor is better compared to others such as the MAX4466, on the other hand, the INMP441 has better audio signal while also not needing power filtering like the MAX sensors. Though, we decided to go with the MAX9814 sensor given that it is easier to set up with the ESP32 and we do not need a high level of sound detections as we’re just trying to detect snoring.

5.3.2 Temperature Sensor

For the temperature sensor, we decided to go with the DHT11. Both sensors are very similar and effective at doing their job. Though, for our project we just need a basic sensor that can read the temperature in the pillow fairly accurately.

Overall, both sensors are good options. The DHT22 ends up being more accurate with a higher resolution. The biggest downside of the DHT22 is that there's a sampling period of 2 seconds, while with the DHT11 we can request temperature readings every second. This may be an important feature as we must always track the temperature of the pillow. We don't need the bigger temperature and humidity range that the DHT22 offers. Both sensors are of low cost and are well received in the community, therefore, either one is a good option for this project.

5.3.3 Ambient Light Sensor

For the ambient light Sensor, we decided to go with the Adafruit 161 Photoelectric Sensor. This sensor has a simple design which includes 2 symmetrical pins. For this sensor, we would have to also use a 10K pull down resistor. We decided to go with this sensor as it is well documented online and easy to get. Additionally, the TEMT600, Adafruit 161 Photoelectric Sensor, and DFRobot's Gravity: Analog Ambient light sensor all work the same way. Therefore, any of these sensors would work well with our project. Though, as mentioned, we'll be choosing the Adafruit 161 Photoelectric Sensor as it is simple to get and work with on the ESP32.

5.3.4 Microcontroller

For the microcontroller, we chose to go with the ESP 32 because of its innate Bluetooth and sufficient memory. We should be able to connect and control all of the external and internal components that our project will need. In addition to this, it can utilize its Bluetooth so we don't need to get an external Bluetooth module. Using this to control all of the parts that we are using will make creating the project more streamlined and intuitive, and will allow for easy connection with the app. We will test to see if we are able to send and receive all the data that we will need to transmit to facilitate all the functions that we will be using.

5.3.5 Wireless Communication

As stated above, we will use the innate bluetooth of the ESP 32 to communicate with the app. The only foreseeable issue that might come up with only using the esp32 for bluetooth communication is the size of the files for sound in and out. The ESP 32 should be sufficient enough to provide the project with the data transfer rate between the app and the Snoozie that we need to allow for the product to run smoothly, and up to the users standards.

5.3.6 Power Systems

5.3.6.1 AC to DC Conversion

For the AC to DC conversion, we will officially be using the quick charge 3.0 USB wall adapter from Anker. We will be buying it online from amazon and have it prepared to provide power to the project. Note that we should be able to power any of the systems with most usb chargers that the user owns.

5.3.6.2 DC to DC Conversion

For the DC-to-DC conversion we will use the diagram that we made on the Texas Instruments webench. The diagram that we chose was the TPS62903 module and should be sufficient in converting the voltage to what our sensors will require

5.3.6.3 Battery Power

For the battery we are going to go with the LiFePo4 battery cell from batteryspace.com. one of the big reasons that we chose to go with the LiFePo4 battery is because it is a lot safer to use rather than the LiPo battery, and this battery being inches away from the user's head, we aim to make sure the product will not blow up on the user.

5.3.6.4 Battery Charging System

For the battery charging system, we chose to go with the DZS Elec 4pcs 18650 Lithium Battery Charging Boost Mobile Power Supply. We feel like all of the included features of this particular model will aid us in designing the project. With that being said, we might go with the Maker Focus 10 Pcs TP4056 Charging module in case the other does not work with what we are doing.

5.3.6.5 Charging Cable

As for the charging cable, we are going to get the Amaitree one, but this part can be switched out with any usb to usb-c chord that is available on the market. We will follow the standards of the usb cable to make sure that any chord will transfer a sufficient amount of energy to power the Snoozie.

5.3.7 PCB

5.3.7.1 Design Software

When designing the PCB, we will use Fusion 360 from AutoCAD to create and model the custom PCB. This is the most useful option over all the other software's just because it is easy to use and widely available. We will all be able to download and use the software with ease and won't need to buy any expensive product. We will also be able to make the BOM and sch file to send off to the manufacturer to make the custom PCB.

5.3.7.1 Board Manufacture

The board manufacturer that we will be using is OSHPark.com. This company specializes in making boards for hobbyists and is really good for one off PCBs rather than some of the other big companies that will require a large amount to be bought. The PCBs are also made in the USA so the shipping will be cheaper than buying from a Chinese manufacturer. We plan for the manufacturer to populate the board as much as needed. if we can get away with at least the surface mounted part and put the through hole parts on ourselves that would be good as well.

5.3.8 Accelerometer

The part that we chose to fill the role of the accelerometer was the ADXL-345. We believe that this part will be able to allow for the planned functions to be able to work as needed and be able to track the movements of the user and send that data to the microcontroller. Although this is a breakout board, we will be taking components from the schematic and putting them onto a custom PCB that we will be making. This should not affect the functionality of the part too much, and should be sufficient enough to track the user's movement.

5.3.9 Speakers, Pressure sensor, Serial Communications, Pillow Selection

5.3.9.1 Speakers

The speaker we decided to use was the Anker Sound Core Mini 3. We decided to use this speaker because it is the longest lasting out of the other speakers our team researched. This is important because we require a speaker that will run throughout the user’s sleep cycles. Additionally, the quality and sound of the speaker was paramount both when we tested normally, and when it was tested against a pillow. Though the bass on the speaker isn’t as powerful as the other models, the size, quality, and battery life of the speaker outshines the bass.

5.3.9.2 Pressure Sensor

The Pressure sensor our group decided to use is the Force Sensor Resistor. We chose this sensor because of both its low cost and simple usage. A goal we can checkmark with this sensor is paying the lowest amount of money to create the best pillow possible. With the different guides on how to use the FSR, it will give our team more time to conduct research and test the different procedures that will come from other parts of the pillow.

5.3.9.3 Serial Communication

The serial communication protocol our team went with was I2C. We chose this form of communication compared to the others because of its flexibility and simplicity. Because the I2C protocol allows for multi/slave-master support, there is a lot more functionality we can add to our design. Because I2C isn't difficult to understand and utilize, it doesn't complicate the design. This is due to it only requiring two bidirectional signal lines in order to create communication among multiple devices. Another reason why I2C was chosen is because it does a great job in error detection/correction mechanisms. This is from the ACK/NACK feature it contains. Though I2C requires more space and is slower than other forms of serial communication, our team still believes it's the best fit for our design.

5.3.9.4 Pillow

After researching and testing different pillows, our team decided that the NaturePedic Organic Cotton/PLA pillow will best fit our design. We chose this pillow because of a few features: It's size, comfortability, flexibility, and material. The size of the pillow (after conducting measurements) will have more than enough space to fit all of the technology going inside it. In comfortability and flexibility, it provided great results for different types of sleepers; whether you are a stomach, back, or side sleeper. Also, the pillow holds its shape very well, which will be a supporting factor to the comfortability of a user's resting period. For materials, the design is flame retardant, meaning it complements the technology we're going to use. Tests will be conducted in regards to safety hazards to ensure no consumer is harmed through our design.

5.4 BOM

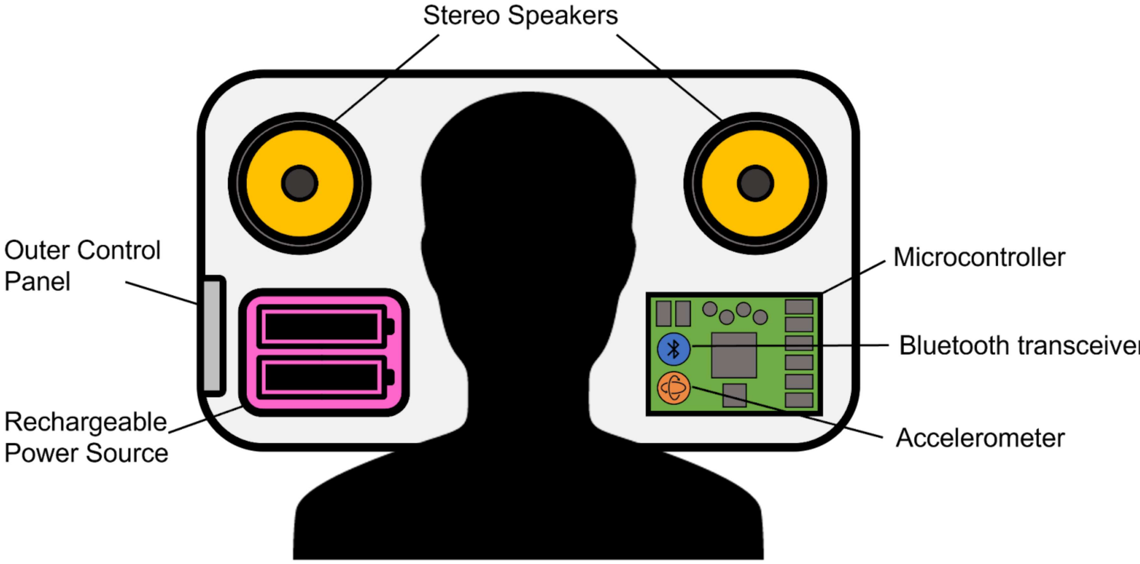
Part Name	Role?	Price	Source
ESP32	Microcontroller/ Wireless Communication	$7.39	Espressif.com

MAX9814	Microphone	$8	Adafruit.com
DHT11	Temperature Sensor	$5	Adafruit.com
Photocell (CDS Photoresistor)	Ambient Light Sensor	$1	Adafruit.com
ADXL-345	Accelerometer	$9.99	Adafruit.com
Quick Charge 3.0, Anker 18W 3Amp USB Wall Charger	Power Source	$15.99	Amazon.com
Amaitree USB C Cable	Power Cable	$9.28 for 3	Amazon
Adafrui lipo 3.7 3 cell battery pack	Battery	$25	Amazon.com

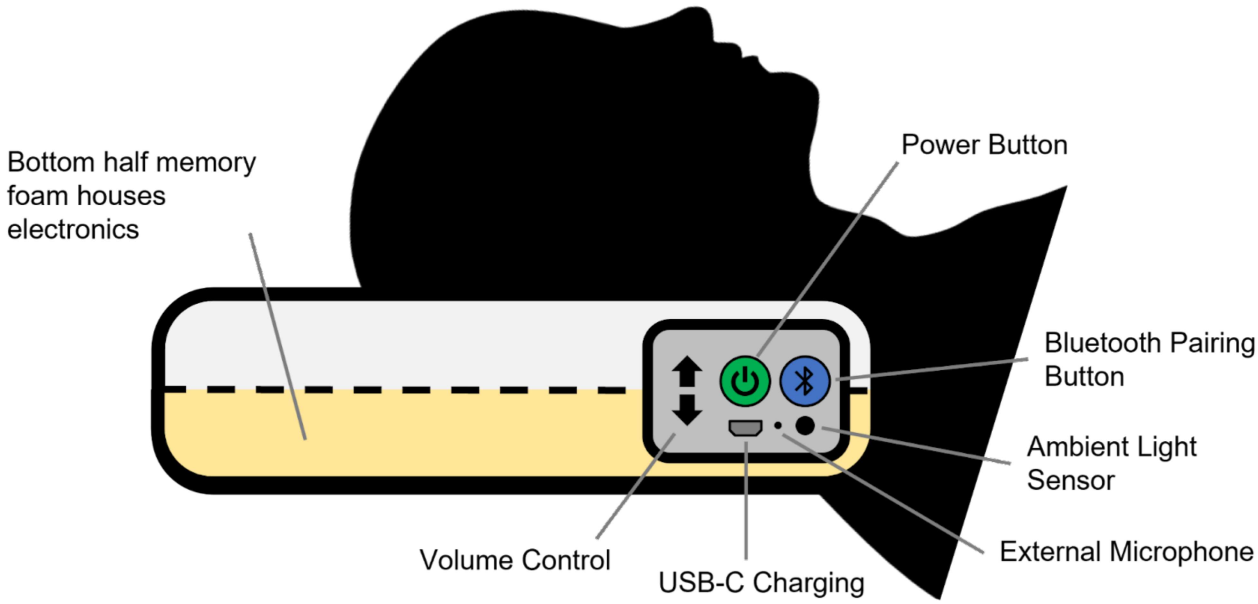
NaturePedic Organic Cotton/PLA pillow	Pillow	$79.00	Amazon
cp2104	Serial Communications	$6.29 (2)	Amazon
Force Sensing Resistor	Pressure Sensor	$11.93 (2)	Amazon
PCB Parts and assembly		$250	

Section 6: Prototyping

6.1 Prototype Design



6.1.1 Prototype Diagram (Top-Down)



6.1.2 Prototype Diagram (Side)

6.2 Prototype Operation

6.2.1 Snoozie Smart Pillow

The user is able to control the snoozie smart pillow via a control panel located on the right side of the pillow. This exposes a number of limited user controls (detailed below).

6.2.1.1 Power Button

One of the large, circular buttons on the side denoted by a power symbol (). This button is used to power the device on and off. When powered on, an LED will be lit up, indicating the device is powered on.

6.2.1.2 Bluetooth Pairing Button

One of the large, circular buttons on the side denoted by a bluetooth symbol (). This button is used to pair the device to the user's android phone or tablet.

6.2.1.3 Volume Controls

Denoted by an up and down arrow on the side of the device. These buttons are used to turn the device's speaker volume up or down. With the pillow laying on a flat surface, the up arrow (pointing at the ceiling) will turn the volume up. The down arrow (pointing at the floor) will turn the volume down.

6.2.1.4 USB-C Charging Port

Located below the power button. This exposed port is used to charge the device. When plugged into a suitable power source, an LED will turn on, indicating the device is successfully charging. Once the LED turns off, the device has been successfully charged

6.2.1.5 Exposed Sensors

Although the user does not directly operate the microphone and ambient light sensor located on the side control panel, it is important to note their location. Both of these sensors should be exposed to the sleeping environment the user is in, and should **not** be covered during operation (if possible). Objects that may obscure data collection by the sensors include other pillows, blankets, the user's arms, etc. Failure to follow these guidelines may result in inaccurate data collection for the user.

6.2.2 Snoozie Companion App

Some features of the Snoozie can only be accessed through the Snoozie Companion Android Application. This is available for use on all Android smartphones and tablets. The descriptions below detail the various sections of the app as well as functions found on each respective page.

6.2.2.1 Alarms Page

Located on the bottom left tab. This page is used to set and adjust alarm settings. Use the clock located at the top of the page to specify what time you would like to wake up. Select the "Alarm Tone" button to set the alarm sound you would like to wake up to.

Underneath the “Tracking Options” header, you can disable and enable sensors on the device by tapping the icon of that sensor. A white icon indicates the sensor is active, while a grey icon indicates that the sensor is disabled. At the bottom of the page is the “Smart Wake” button. Enable this feature if you would like your Snoozie to wake you up using its sleep cycle tracking feature. In this setting you can adjust what time frame you would like to be woken up in.

6.2.2.2 Sleep Wellness Page

Located on the bottom middle tab. This page summarizes your sleeping data that has been collected in the past 24 hours. The top chart labeled “Audio” shows the various audio levels detected by the microphone throughout the night. The center chart labeled “Motion” displays movements detected by the accelerometer throughout the night. Finally, the bottom chart labeled “Light” displays the ambient light levels detected in the room throughout the night. Within each of these charts, the Snoozie Companion App will use this data to predict when you entered and transitions between sleep cycles throughout the night.

6.2.2.3 Audio Page

Located on the bottom right tab. This page is used to manage any and all media playback through the Snoozie. The top button, “Add Alarm Tones”, allows the user to import custom alarm tones from their android device’s library. The second button, “Ambient Noise”, allows the user to select an ambient noise profile to fall asleep to. These are preloaded on the Snoozie Companion App. The next button, “Link Music App”, allows the user to sign in to their Spotify account to enable playback through the Snoozie Smart Pillow. The bottom button, “Paste Video URL”, allows the user to paste a link to a video they would like to play through the Snoozie.

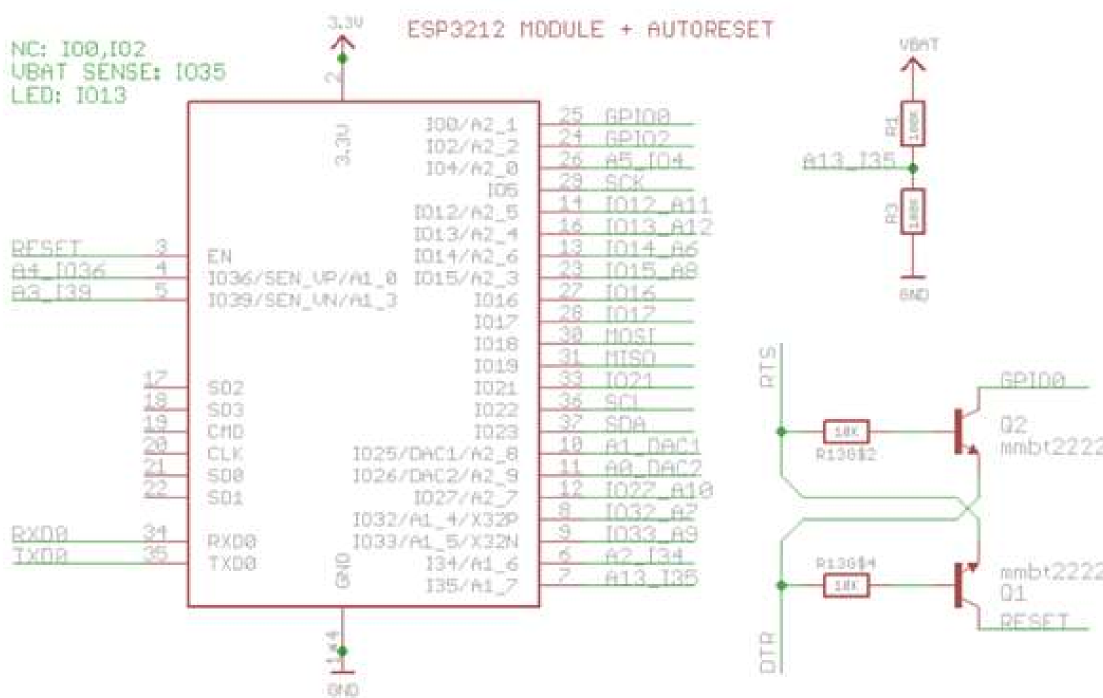
Section 7: Testing

7.1 Hardware Testing

7.1.1 Microcontroller Testing

The Microcontroller will be the heart of our project, so we will need to make sure that this main section of the project is sufficiently tested. Some of the important things that we will need to test when it comes to the microcontroller is the memory, the number of components, and the size of the chip. We need to make sure that it will be able to handle everything that we will require it to do. To start off, we will be taking a look at the pinout of the current dev board that we will be using.

The Dev board that we are using is the Huzzah32 feather board using the ESP32. We can see from this official schematic from the developer, we have many pins that we are able to utilize in this project. We will specifically look at the data transfer pins to test to see if we will have an adequate number of pins for all the components.



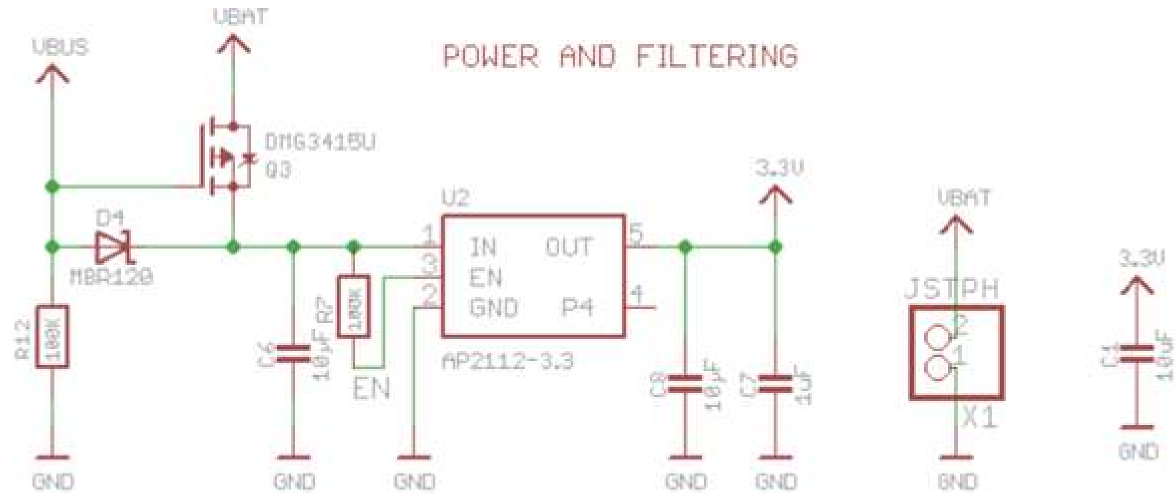
Pin Type	Number of pins
GPIO	**7**

Analog	**7**
I2C	**12 (only 1 needed)**
SPI	**3 (only one needed)**
Serial (RX/TX)	**1 (pair)**
DAC	**2**
I2S Audio	**2**

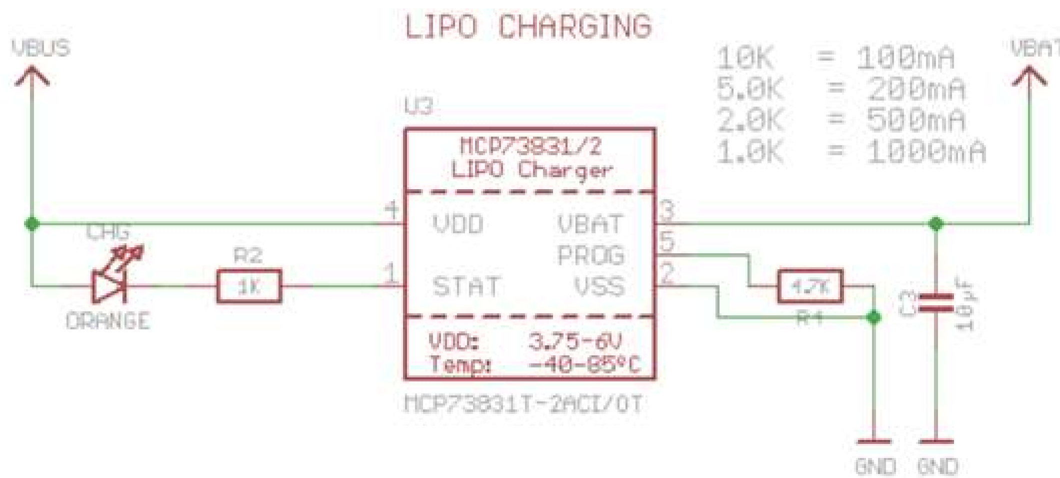
We will have 3 sensors, a microphone, and a pair of speakers, which is well within the number of pins that are available to us. The connections that the ESP32 will be able to handle the ADC signal read of the sensors and be able to read the sound input of the microphone, along with outputting I2S for the sound for the speakers. We will go into further detail about the communication of each part in their respective sections.

The next section of testing will be about powering the microcontroller and its power draw. The connection to the microcontroller will be from the charged batteries that will be connected through the dev boards built in battery unit. The board will handle all the powering from the battery connection to the esp32. It is recommended by the manufacturer that we use this method to power the board, so we will make sure to use the power and filtering module in the schematic sheet when we make our custom PCB, along with the lipo charging circuit.

Huzzah32 Power and Filtering Schematic



Huzzah 32 Lipo Charging schematic



The power draw of the microcontroller can change depending on what power mode that we are in. The esp32 allows us to use 5 different modes that can change the power consumption depending on what we are using. The modes are active mode, modem sleep mode, light sleep mode, deep sleep mode, and hibernation mode. In active mode, the power draw will be between 160 – 260 mA, this is the mode that will require the most power draw so we will take advantage of the sleep modes to keep the power draw low and make sure that we can extend the battery life of the Snoozie as much as possible. Modem sleep mode is when we turn off the wireless capabilities, which reduces the power draw to 3 – 20 mA. We can use this while we are not connected to the app through Bluetooth. The rest of the power modes are going to be similar to a microcontroller's deep sleep mode, so we can use an assortment of those while the pillow is not in use. Here is the table in the esp32 datasheet which will be used during testing.

Power mode	Active	Modem-sleep	Light-sleep	Deep-sleep	Hibernation
Sleep pattern	Association sleep pattern			ULP sensor-monitored pattern	-
CPU	ON	PAUSE	ON	OFF	OFF
Wi-Fi/BT base-band and radio	ON	OFF	OFF	OFF	OFF
RTC	ON	ON	ON	ON	OFF
ULP co-processor	ON	ON	ON	ON/OFF	OFF

Power mode	Description	Power consumption
Active (RF working)	Wi-Fi Tx packet 13 dBm ~ 21 dBm	160 ~ 260 mA
	Wi-Fi / BT Tx packet 0 dBm	120 mA
	Wi-Fi / BT Rx and listening	80 ~ 90 mA
	Association sleep pattern (by Light-sleep)	0.9 mA@DTIM3, 1.2 mA@DTIM1
Modem-sleep	The CPU is powered on.	Max speed: 20 mA
		Normal speed: 5 ~ 10 mA
		Slow speed: 3 mA
Light-sleep	-	0.8 mA
Deep-sleep	The ULP co-processor is powered on.	0.15 mA
	ULP sensor-monitored pattern	25 μA @1% duty
	RTC timer + RTC memory	10 μA
Hibernation	RTC timer only	2.5 μA

Another important thing is that we need to make sure that we have enough memory to facilitate all the code that we will need to use to operate the microcontroller. When it comes to the memory, we need to have the program size to be able to assess if we are able to fit it onto the onboard memory. If we do not have enough flash memory or

SRAM, we can utilize an external memory unit, which the esp32 can support. The esp32 datasheet says that the esp32 supports 4 x 16 Mbytes of external QSPI flash and SRAM, so we have the option to expand if we need to. Here are the memory specifications that will be used when testing the memory of the esp32.

Category	Target	Start Address	End Address	Size
Embedded Memory	Internal ROM 0	0x4000_0000	0x4005_FFFF	384 KB
	Internal ROM 1	0x3FF9_0000	0x3FF9_FFFF	64 KB
	Internal SRAM 0	0x4007_0000	0x4009_FFFF	192 KB
	Internal SRAM 1	0x3FFE_0000	0x3FFF_FFFF	128 KB
		0x400A_0000	0x400B_FFFF	
	Internal SRAM 2	0x3FFA_E000	0x3FFD_FFFF	200 KB
	RTC FAST Memory	0x3FF8_0000	0x3FF8_1FFF	8 KB
		0x400C_0000	0x400C_1FFF	
	RTC SLOW Memory	0x5000_0000	0x5000_1FFF	8 KB
External Memory	External Flash	0x3F40_0000	0x3F7F_FFFF	4 MB
		0x400C_2000	0x40BF_FFFF	11 MB
				248 KB
	External SRAM	0x3F80_0000	0x3FBF_FFFF	4 MB

Microcontroller Testing Table

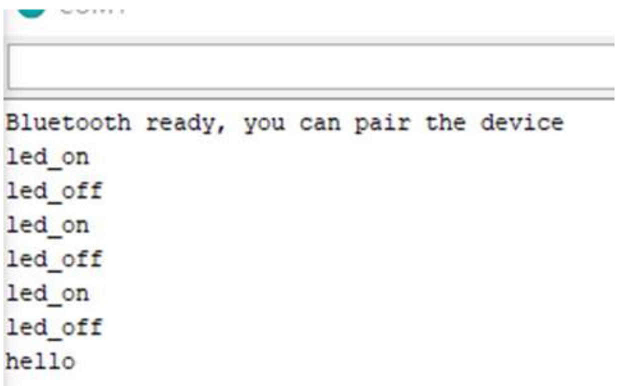
#	Item	Test Method	Criteria
1	Power input	Connect the board to the project power board. The 5V input should be connected to the required input usb port.	The controller is powered and useable

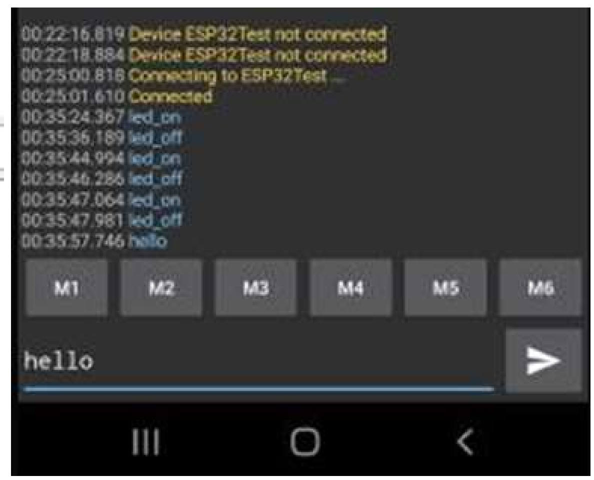
2	Component communication Analog	The component that requires analog communication is connected and powered. The component will send data and the controller will read it	The controller is able to read good analog values from the component
3	Component Communication I2C	The component that requires I2C communication is connected and powered. The component will send data and the controller will read it	The controller is able to read good I2C values from the component
4	Sleep Modes	The power is connected, and the different power modes are turned on while a USB power monitor is connected. the power draw will be measured and testes	The sleep modes work as intended
5	Memory	The controller will receive a signal from a microphone, and the controller will be able to save the sound in memory to be sent	The sound gets transferred

7.1.2 Wireless App Communication Testing

Communication to and from the esp32 is going to be a large part in this project because we are going to need to transmit and receive data from the app. To do this, we will be using the innate Bluetooth capabilities of the esp32. To test the Bluetooth, we will use the Arduino IDE to make some testing code to make sure that the Bluetooth works on the board.

For the testing code, we set up a section of code that will turn the bluetooth of the esp32 on and attempt to communicate via serial bluetooth. To test if the board is successful in connecting to the app, I will make a bluetooth serial communication app send a code to turn on and off the onboard led. Once I downloaded a serial communication app on my phone, I ran my test code and connected my phone's bluetooth to the esp 32 at a baud rate of 115200. Once I do that, I send a text signal from my phone to the esp32 and the esp 32 will be polling for the signal to turn on and off the onboard led.





The commands “led_on” and “led_off” successfully turned the LED on the esp32 on and off. This test shows that we are able to use Bluetooth serial communication to allow the app and the board to communicate.

As for the data rate that we will be able to achieve, we will look at the data transfer rate. This will be tested when we are able to connect the sound portion to the circuit, sound will be the biggest transfer between the app and the project.

Another big part of the wireless transmission will be the range that the esp32 can reach before not being able to transmit data. According to the esp32 datasheet, the esp32 is capable of class 1, 2, and 3 Bluetooth transmission without the need for an external power source. We can look at the specification of Bluetooth to see what ranges that each of these classes can reach.

Class	Maximum Power	Range
1	100 mW (20 dBm)	100 meters
2	2.5 mW (4 dBm)	10 meters
3	1mW (0 dBm)	1 meter

We will test the range by trying to connect to the Bluetooth device at a range and transmit data to make sure that it is working properly.

Along with all of these tests, we will need to test the power draw when we are using the Bluetooth module. We will need the device to be in active mode for the transceiver to get sufficient power, so we will be drawing a decent amount of power. The power draw will change when we are using the Bluetooth, so we can use a USB tester to see just how much the power draw will change.

Wireless Communication Testing Table

#	Item	Test Method	Criteria
1	Bluetooth Range	The controller will be connected to an external Bluetooth enabled device and the device will be moved at intervals until it disconnects from the controller	The range that the device disconnects must be over 3 meters

2	Bluetooth Power Draw	The controller will have its power tested with and without the BLE turned on.	The BLE should operate at a reasonable power draw. (under 260 mA)

7.1.3 Serial Communication Testing

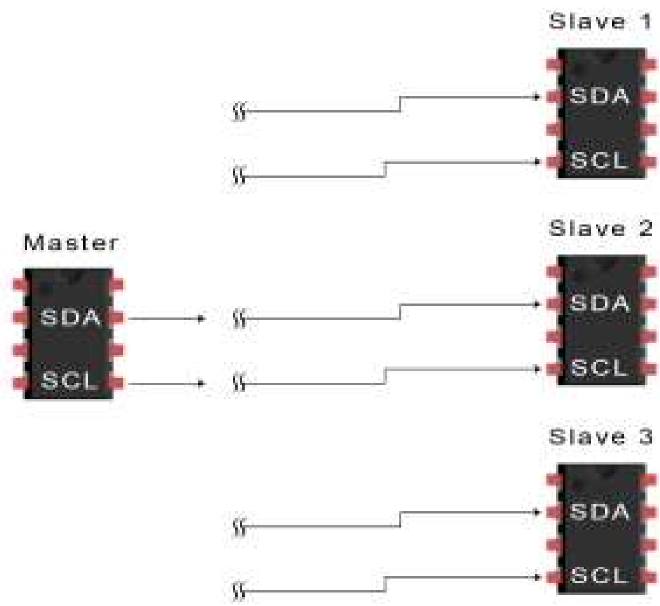
7.1.3.1 - Advantages & Disadvantages

For Serial Communications, we determined that I2C will best suit our design. The advantages of using I2C compared to other means of serial communication is that it only uses two wires, as well as having hardware that's less complicated than UARTs. Also, the ACK/NACK bit will provide confirmation that each frame will be transferred successfully. Lastly, it will be able to work with more than one master and more than one slave. Though these advantages are great, we have to mind the disadvantages I2C can bring to our design. Using I2C as our serial communication will have a slower data transfer rate compared to SPI. Also, the size of the data frame will be limited to 8 bits, which can affect our pillow's performance. Lastly, the hardware needed to implement I2C can be more difficult to understand compared to SPI.

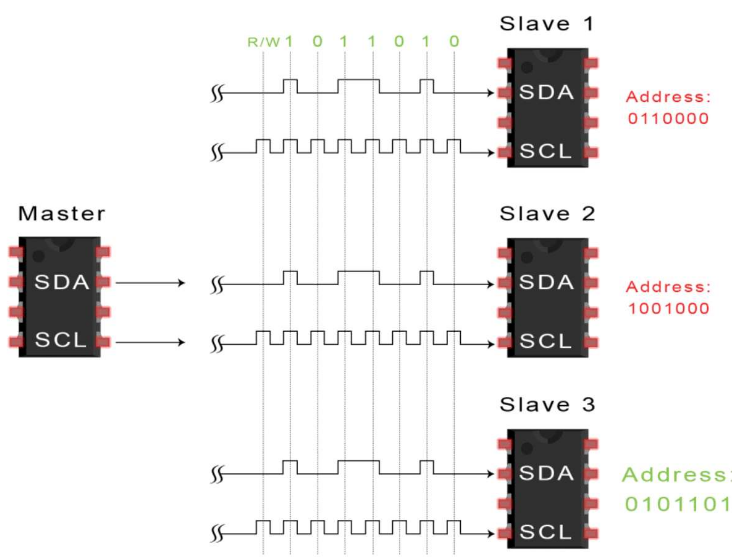
7.1.3.2 - I2C Data Transmission Procedure

As Jason was researching and testing I2C Serial Communication, he was able to analyze the procedures taken in order for the I2C to transfer data to the Snoozie's technology. Below will be said procedures:

1. The master will transfer the start condition to connected slaves. This is done by switching the SDA line from a high-to-low voltage level. This occurs before switching the SCL line from high to low

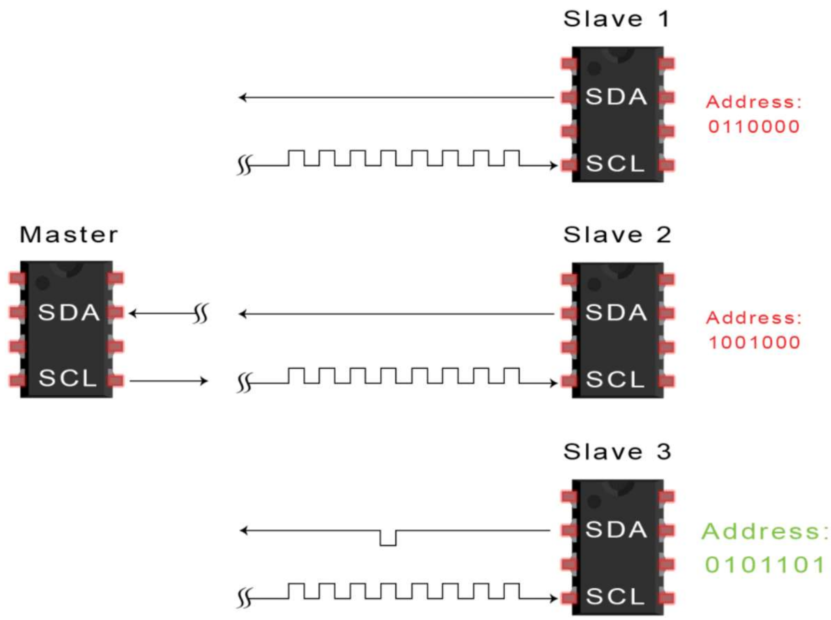


2. Master will then send each connected slave to the 7/10 bit addresses of the slave it would like to communicate with (the read/write bit will be included in this)

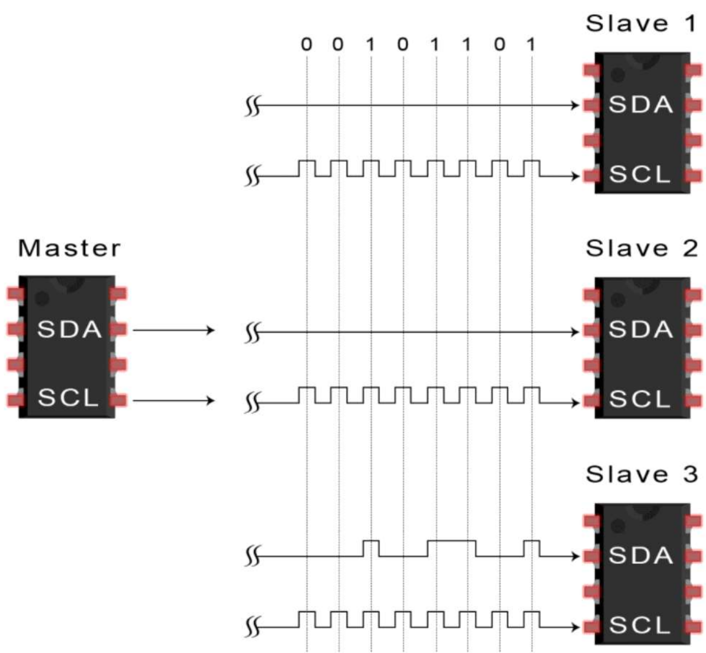


3. Every slave will compare the address it received from the master into its own address. Having an address that matches will allow the slave to return an ACK bit by pulling the SDA line low for one bit. On the contrary, if the address sent

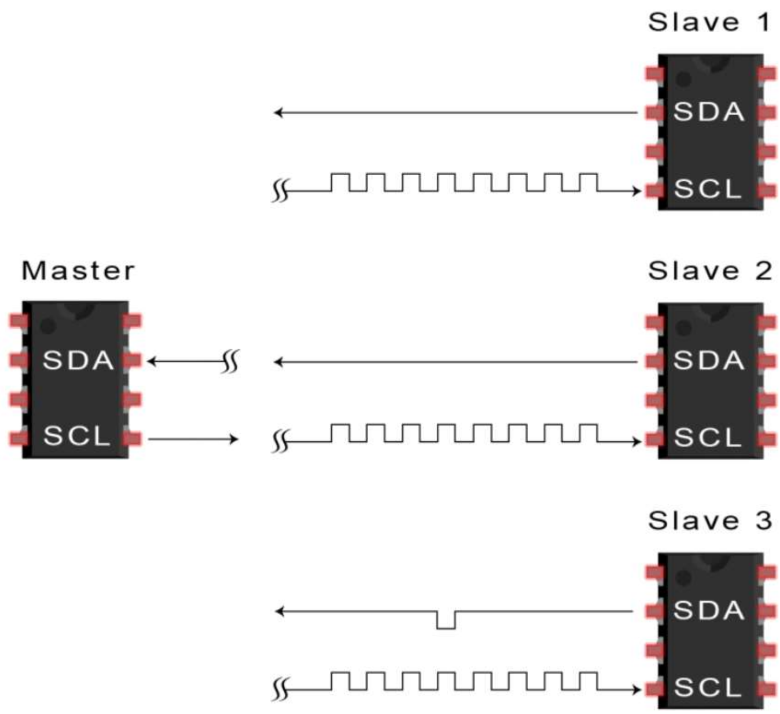
from master doesn't match with the address the slave has, it will leave the SDA line high



4. Master either receives or sends the data frame

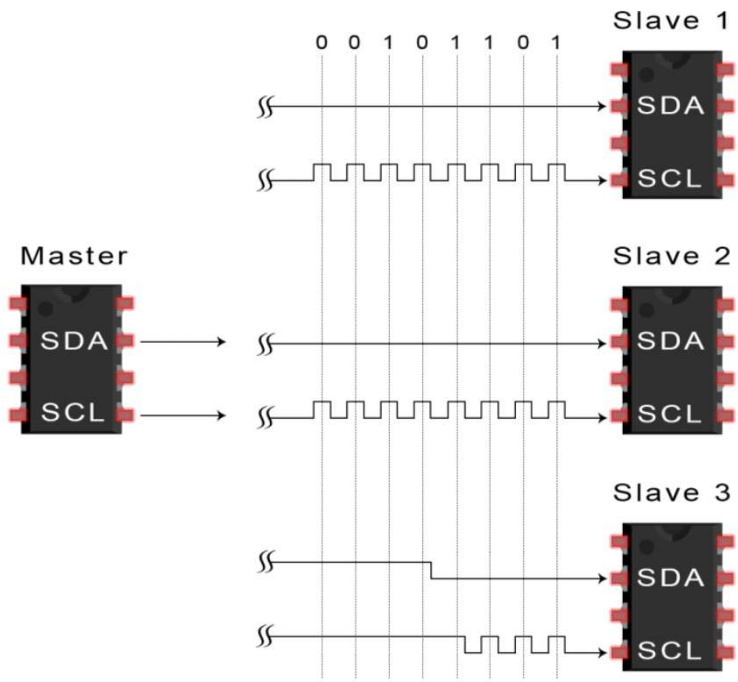


5. When every data frame is transferred, the device receiving it will provide an ACK bit to the sender, which will then acknowledge a successful receipt of the frame



6. Lastly, in order for data transmission to come to a halt, master will provide a stop condition to slave. This is done by switching the SCL high before having SDA

high being switched



With these procedures, we determined that I2C will be the best means of serial communication for our design's sensors. The physical characteristics, as well as the features I2C provides will have our design functioning well.

7.1.4 Speakers Testing

For our design, our team decided to have the Anker Sound Core 3 Mini Bluetooth Speaker be our designated speaker. For serial communications with our speaker, we decided to go with I2S protocols for our design. There were a few features that determined why we wanted to have this speaker be used for our Snoozie. The first thing we took into account was the size of it. The Sound Core 3 Mini is very small compared to the other speaker, but still packs a punch in regards to the sound it produces. Having a small speaker is important because it allows us to have more room to manipulate our pillow's design. Once the speaker was purchased, Jason tested the Sound Core speaker. When using this speaker, Jason wasn't expecting too much on the sound of the speaker due to its size. However, once music was played, it provided a loud and crisp sound. This is great because the team doesn't want to use a speaker that has both bad quality and a low volume. Once testing the speaker normally, Jason put the speaker behind the Naturepedic pillow (pillow being used for the Snoozie design). This was done to test the quality of sound when the speaker would be "inside" the pillow. Once music was played against the pillow, the speaker continued to provide a rich and loud sound.

After testing the sound of the speaker, we tested the battery life on it. Again, having a long battery life (preferably more than 8 hours) is required for our design because we would like to create an audio ambience for our users for the whole duration of their sleep. To test this, Jason played music from a PC and had the speaker output the music. Once the music started playing, and the speaker was fully charged, Jason set a stopwatch to see how long the speaker has until it gives out. After roughly 15 hours, the speaker died. This is very good for our design because this allows the user to have an ambience throughout their sleep cycles.

7.1.5 Microphone Testing

For the microphone we decided to use the MAX9814, an electret amplifier with auto gain control. Because of the AGC in the amplifier, nearby 'loud' sounds will be quieted so they don't overwhelm and 'clip' the amplifier, while quiet, far-away sounds will be amplified. This amplifier is ideal for recording or detecting audio in a changing environment where you don't want to constantly adjust the amplifier gain.

The AGC MAX9814 chip has a few options that can be configured via the breakout. The max gain is set to 60dB by default, but it can be changed to 40dB or 50dB by connecting the Gain pin to VCC or ground.

We can also alter the Attack/Release ratio from 1:4000 to 1:2000 or 1:500. The amp's output is about 2Vpp max on a 1.25V DC bias, so it can be used with any Analog/Digital converter with an input voltage of up to 3.3V. To connect it to a Line Input, we simply connect a 1-100uF blocking capacitor in series.

All surface-mount components are pre-soldered on the board. For ease of use, the included header strip can be soldered on.

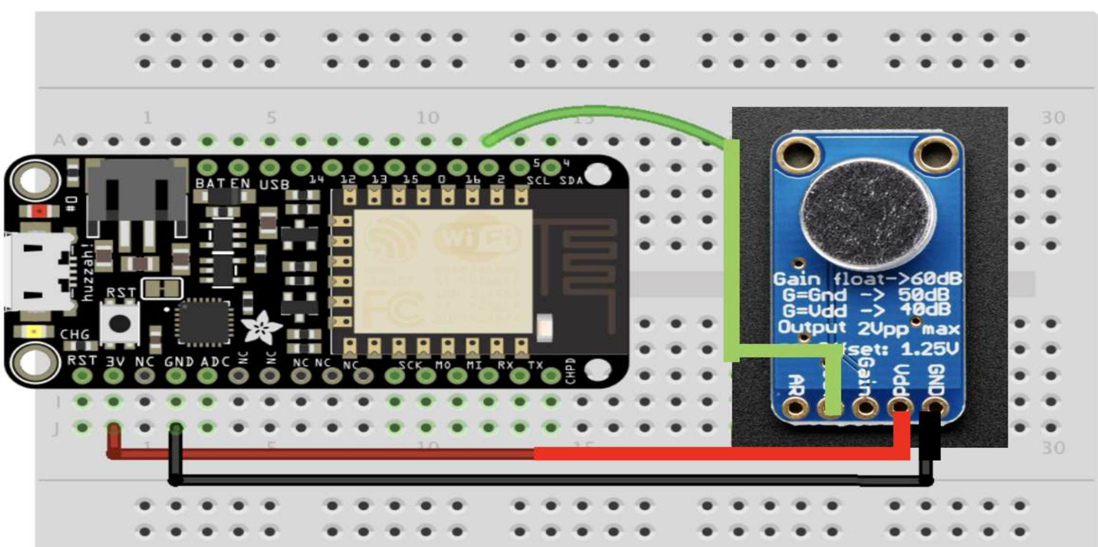
The MAX9814 has 3 main connections that we need to use:

OUT -> Connect to an ADC on the ESP32
GND -> Connect to GND
VCC - > Connect to 3.3V

VCC can range between 2.4-5VDC. We use the 3.3v pin for the best performance because it is the "quietest" supply and we know that the MAX microphones are more susceptible to power supply noise.

The output will have a DC bias of VCC/2, so when everything is perfectly quiet, the voltage will be VCC/2 (1.65v).

The following image has the schematic of what the connection would be like when we connect the MAX9814 to the ESP32 Feather.



For hardware testing of the microphone, we can use a multimeter to test if the audio signal from the output of the amplifier is a varying voltage. By creating different sounds in the room we can make sure that the microphone hardware is working correctly if we see the voltage varies.

To determine the sound level, we must take multiple measurements to determine the signal's minimum and maximum extents, or "peak to peak amplitude."

For the software side, what we can do is choose a sample window of 50 milliseconds, this is enough to measure sound levels as low as 20 Hz, which is the lowest limit of human hearing. We compute the difference between the minimum and maximum samples, convert it to volts. With this we will be able to see the values of the voltage as it varies to the sounds in the environment.

For software testing of the microphone, we can check if the sample window is correctly looping for 50 milliseconds. Additionally, we can check if the values calculated by the voltage function are working correctly by inserting some test values and checking if these values are correct. This way we can make sure that the voltage function works correctly, thus, the values we receive are correct. Furthermore, we can make sure that the code is correctly only saving in the max and min signal values by sending out of range values to see if the values are saved or not.

7.1.6 Temperature Sensor Testing

For the temperature sensor, we decided it was best to go with the DHT11 sensor. To explain the difference between the DHT11 and DHT22. The DHT11 has a smaller range and it's less accurate. However, you can request sensor readings every second. It's also a bit cheaper. Additionally, the code for both of these sensors are the same, we would just need to select the sensor we are using.

When it comes to how we are going to output the data. We can display the values in Celsius and in Fahrenheit. The DHT Sensor Library will save the values in Celsius by default but we can easily have the value in Fahrenheit by using a simple formula:

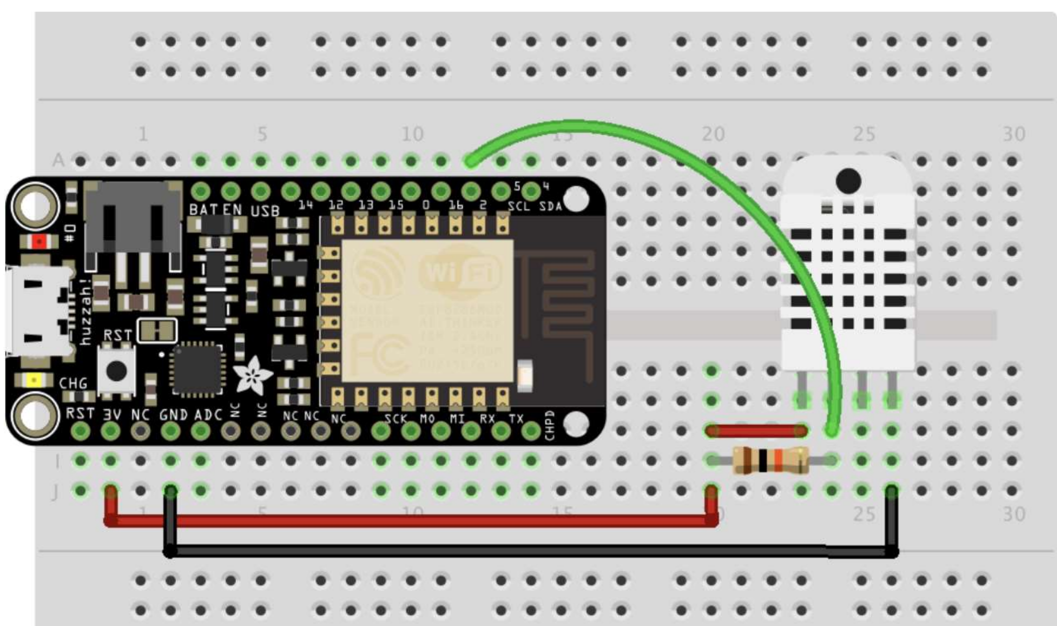
Fahrenheit = (Celsius * 1.8) + 32

Additionally, the humidity is by default a value in percentage, so we don't need any extra calculations in the code to output the correct value. Given that the DHT11 sensor allows us to request sensor readings every second we are able to constantly display the temperature of the pillow. The only disadvantage of the DHT11 is that it has a +-2% error, though this difference isn't a problem for our project.

The DHT11 has 4 pins, of which we'll use 3 to connect to the microcontroller. The Following table shows the pinout of the DHT11:

DHT Pin	Connection
1	3.3V
2	Any digital GPIO. Also connect to a 10K Ohm pull-up resistor
3	To not be connected
4	GND

The following figure is the schematic of how the physical connection to the ESP32 would look like: [Link of image]



Basically, we can see that we need to do the following:

DHT11 Pin 1 to 3V of ESP32
One leg of the 10k Ohm Resistor to 3V of ESP32
Other leg of resistor to Pin 2 of DHT11
DHT11 Pin 2 to ESP32 GPIO (ADC)
DHT11 Pin 4 to ESP32 GND

The DHT uses an NTC sensor. The term “NTC” means “Negative Temperature Coefficient”, which means that the resistance decreases with increase of the temperature.

With this information we can test the hardware, by using a multimeter we can check the value of the resistance and check if it correctly decreases as temperature increases. This way we can confirm that the hardware is working to implement with the microcontroller.

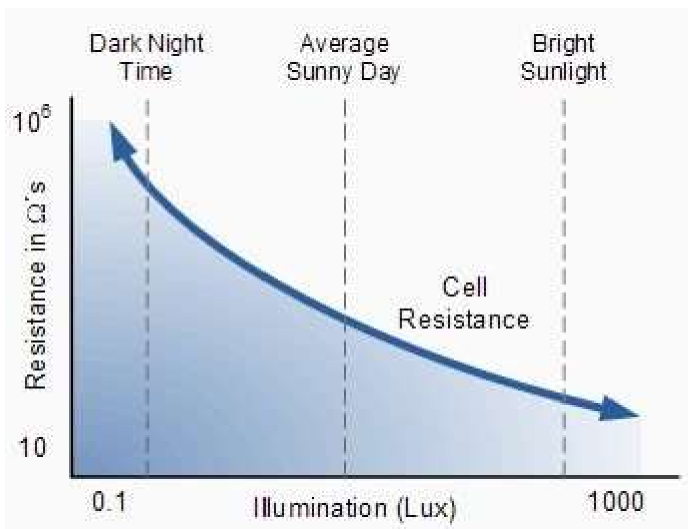
To read from the DHT sensor, we will need to use the DHT library from Adafruit. To use this library we also need to install the Adafruit Unified Sensor library. This can be done from the Arduino IDE.

When it comes to implementing the code, we will need to define the pin we used to read the sensor values. Additionally, we will need to create an instance of the DHT class, this way we can use the temperature and humidity feeds. We will also need a setup function that initializes the DHT11 sensor, and also connects your feather to Adafruit IO. Finally, we will be able to create a loop function to print out the values of the temperature and humidity in either celsius or fahrenheit.

For software testing, we can test some values to make sure that the fahrenheit and celsius conversion is working correctly. This way we can check for either celsius and fahrenheit and make sure that the data being displayed is correct. Additionally, we know that with the DHT11, it can't read negative temperature values, therefore we can check for negative values. In addition, the temperature from the DHT11 can't read over 50 degrees Celsius. So, we can test for values that are outside the range of 0-50 degrees Celsius. This we make sure that the true values tested will fall within the correct range.

7.1.7 Ambient Light Sensor Testing

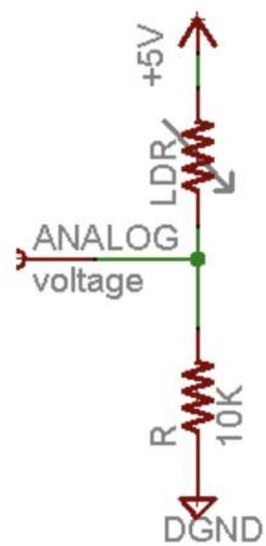
For the ambient light sensor we have decided to go with the Adafruit 161 Photoelectric Sensor. This sensor is essentially a photocell also known as a CdS cell (made of Cadmium-Sulfide). As explained before, the photocell's resistance changes as the face is exposed to more light.

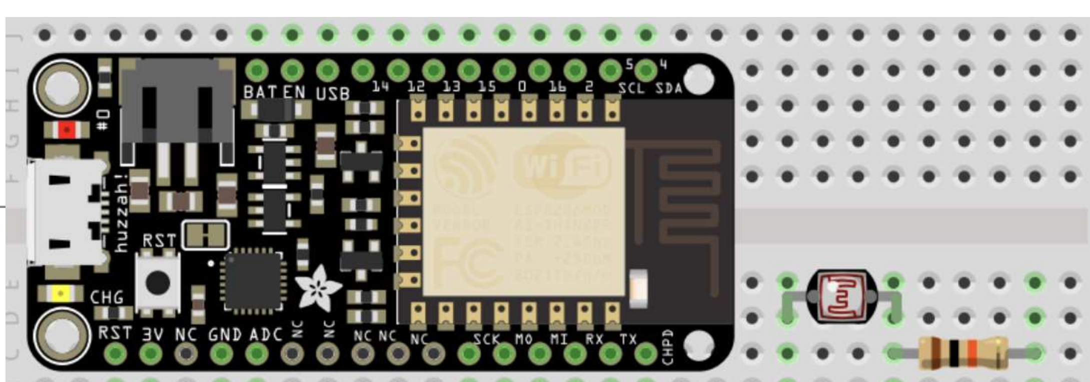


In the figure above, we can see a graph showing the approximate values of the resistance at different light levels. Normally CdS cells are sensitive to light between 500 nm (green) and 700 nm (red) light. In perspective, Typically, the human eye can detect wavelengths from 380 to 700 nanometers. [Link to image]

Now we need to figure on how to display those values and interpret them. For this scenario, it is commonly used to have the values in lux to indicate the resistance at certain light levels. In brief, Illuminance is a measure of how much luminous flux is spread over a given area. One can think of luminous flux (measured in lumens) as a measure of the total "amount" of visible light present, and the illuminance as a measure of the intensity of illumination on a surface. Therefore, the lower the value of lux, the less light is received by the sensor. While, the higher the value of lux, the higher the value of lux due to the resistance increasing.

To test the hardware aspect of this sensor, we can connect a multimeter in resistance-measurement mode to the two leads connected to the sensor. With this, we can view the changes in resistance when different light levels are pointing to the face of the photocell. Effectively, we will be able to see that the equipment is working and ready for use.
Connecting this sensor is straightforward, we connect one lead to the power such as 3.3 V or 5 V, and the other lead to a pull-down resistor to the ground such as a 10K Ohms. Between the fixed pull-down resistor and the photocell we connect to the dedicated ADC pin on the ESP32.





The figure above shows how the connection would be to the ESP32 Feather. We have the photocell connected to the resistor and connected to the ADC pin on the board while the other lead is connected to the 3.3 V on the board. [Link of image]

As the resistance of the photocell decreases, the total resistance of the photocell and the pulldown resistor decreases from over 600KΩ to 10KΩ. That means that the current flowing through both resistors increases which in turn causes the voltage across the fixed 10KΩ resistor to increase.

An issue with using the 10K Ohms pull-down resistor, is that if the sensor is used in a bright area, it can saturate quickly. That means that it will hit the 'ceiling' of 3.3V and not be able to differentiate between kinda bright and really bright. If this happens, we can replace the 10K Ohms with a 1K Ohms pull-down resistor. Though, using 1K Ohms won't allow the sensor to detect dark differences that well, but it will be able to detect higher differences with more light.

On the software side, it is easy to get results from the sensor. The most important part is having the right wiring and knowing what pins are being used on the board as we will need to initialize variables to the appropriate pins that we are using. We don't necessarily need to do any calculations in the code. We can just print out what it interprets as the amount of light in a qualitative manner. The values output should fall within 0-1000, with this we can easily tell the ambient light of the room. The higher the value, means the more light at the face of the photocell, while the lower the value the less light it receives.

If we do intend to convert the light dependant resistor (LDR) value to lumens, we would have to do the following: [Link to image]

```
int sensorRawToPhys(int raw){
  // Conversion rule
  float Vout = float(raw) * (VIN / float(1023));// Conversion analog
to voltage
  float RLDR = (R * (VIN - Vout))/Vout; // Conversion voltage to
resistance
  int phys=500/(RLDR/1000); // Conversion resitance to lumen
  return phys;
}
```

In the above code we can see that we would need to convert the analog output of the sensor to voltage. From there we can convert it to resistance, and finally to lumen.

For testing the software, we can make sure that the values received are not negative. It would be impossible to receive a negative value from the sensor. Additionally, if we use values in lumens, we can simply make sure that the lumen conversion works by inserting some numbers as test values, this way we can make sure the process and output are correct.

7.1.8 Accelerometer Testing

The part for the accelerometer will be the sensor that will send data to the microcontroller about the movement of the user. This data will be important to collect for the algorithm that will track the sleeping habits of the user, so we need to make sure that it's properly tested and works as intended. The ADXL-345 is a 3 axis accelerometer module that we will be using in our project, and is what we will be testing in this section.

The way that this module works is that there is a set of small poly silicon springs that suspend the sensor so that when the part moves, the distance between the sensor and a metal plate changes, which in turn changes the capacitance of a connection and that outputs a voltage that is read and outputted by the module. The module can in theory sense 3 axes, x, y, and z which will be just what we need for the project to function properly.

To test if this part will properly function, we can connect it to our dev board and have the module output numbers that we can translate into useful data. To connect the module, we will connect the Vin and GND pin to the boards power pins respectively, then we will use the i2C connections of the dev board to connect to the SDA and SCL pins of the module and that will complete the hardware assembly of the module. The next part is to make some code that will output the parts data and make sure that the numbers we are getting are correct. We can also change the output resolution to fit with what we need. The table in the datasheet for the part shows us that we have some different resolutions that we can utilize to change how precise the data output that will be sent to the microcontroller can be.

OUTPUT RESOLUTION	Each axis		
All *g* Ranges	10-bit resolution	10	Bits
±2 *g* Range	Full resolution	10	Bits
±4 *g* Range	Full resolution	11	Bits
±8 *g* Range	Full resolution	12	Bits
±16 *g* Range	Full resolution	13	Bits

For this application, we will probably utilize the 8g range of the controller to save on some of the processing power of the microcontroller.

With the main part tested, we can move to testing things like power consumption, basic operation, and various other things that might come up when using this product. The power consumption of the module can be a determining factor in how we use this part. We can see from the data sheet, the operating voltage of the part will be between 2V and 3.6V, which we will be able to supply to the part. As for the amperage that this part will need, the range is 1.7 to 1.8 micro-Amps, which will be fine for the power supply.

POWER SUPPLY					
Operating Voltage Range (V_S)		2.0	2.5	3.6	V
Interface Voltage Range ($V_{DD I/O}$)		1.7	1.8	V_S	V
Supply Current	ODR ≥ 100 Hz		140		µA
	ODR < 10 Hz		30		µA
Standby Mode Leakage Current			0.1		µA
Turn-On and Wake-Up Time[7]	ODR = 3200 Hz		1.4		ms

The module also comes with an auto test mode that will be able to make sure that the part is working correctly. To do that we can start the part and send a bit in the SEL_TEST input bit when we initialize the part. This will use electrostatic force to move the sensor and make sure that the part is functioning the way that is supposed to.

Accelerometer Testing Table

#	Item	Test Method	Criteria
1	Testing the values of the module are correct	Hook up the module to a dev board and run test code	The module will return numbers that reflect the motion of the sensor
2	Test the resolution that we need	Run code on the module and see what each resolution outputs	The sensor can run each resolution and we can notice the differences between each
3	Testing the voltage of the module	Use a multimeter with its pins on the Vin and GND pins	The multimeter should show a voltage between 2 and 3.6 Volts

4	Test the range of motion of the accelerometer	Run the test code and move the part around	The part should be able to detect the movements similar to those of a user moving a pillow
5	Utilize the innate self-testing of the module	Send the module the self-test input bit in the input code	The self-test output should return with a good signal

7.1.9 Power System Testing

The power system will need to be tested when we are able to power our system fully when we have all the parts to test things like the power draw and the voltage. We have access to many things that we can use to test all the power systems that we will be using, equipment like multimeters, oscilloscopes, and USB testing modules. We will use this equipment and more to ensure that our power system is running as intended.

7.1.9.1 AC to DC Testing

For the ac to dc testing, we want to make sure that the output for the wall adapter really outputs 5V. The way that we can set that is to either plug in a USB power tester, or we can use a scrapped USB power cord and attach the multimeter to it. We will also test the amperage the same way. Once we know that our 5V power line is good, we can then plug it into our project. These will be the optimal conditions that the microcontroller will need to function properly.

Recommended Operating Conditions

Parameter	Symbol	Min	Typ	Max	Unit
Battery regulator supply voltage	V_{BAT}	2.8	3.3	3.6	V
I/O supply voltage	V_{IO}	1.8	3.3	3.6	V
Operating temperature range	T_{OPR}	-40	-	125	°C
CMOS low level input voltage	V_{IL}	0	-	0.3 x V_{IO}	V
CMOS high level input voltage	V_{IH}	0.7 x V_{IO}	-	V_{IO}	V
CMOS threshold voltage	V_{TH}	-	0.5 x V_{IO}	-	V

7.1.9.2 DC to DC Testing

For DC-to-DC testing, we will connect the circuit to the 5V input and then use a multimeter to check if the voltage is correct. We should be getting around 3.3V, and the circuit is rated to 2 amps, which should be enough to power the sensors and other components that will require that specific voltage.

7.1.9.3 Battery Testing

There are many things that we need to make sure work when it comes to the battery, so many tests will need to be made to ensure that the batteries are working as we need them to. To start, we need to make sure that the charge will hold for the required amount of time. To test this, we will calculate the power draw in amps and use that to calculate the mAh required to successfully power the device for the needed length of time.

7.1.9.4 Charging Testing

The charging module should be able to charge the included batteries and have them connect to the power output to power the project. this module should be able to withstand charging and discharging the batteries multiple times and will be able to take a USB C connector.

Power System Testing Table

#	Item	Test Method	Criteria
1	AC to DC	The wall adapter will be plugged into the wall and a USB power tester will be plugged into the USB port	The output of the wall adapter will output 5V DC
2	AC to DC	The wall adapter will be plugged into the wall and a ammeter will be connected to the wall outlet	The wall outlet provides enough current
3	DC to DC	The DC converter will be connected to 5V DC and the output will be measured with a multimeter	The DC converter output will be around 3.3V
4	DC to DC	Connect the output of the DC converter to each component power line	The component will be power with sufficient energy
5	Battery	The battery terminals will be connected to a multimeter	The multimeter will read around 3.3V
6	Battery	The battery will be fully drained and then charged to max and then a multimeter will test the voltage	the multimeter should be around 3.3V

7.1.10 - Pillow Testing

Our group has determined that the pillow we believe will best fit our design is the Naturepedic Organic Cotton/PLA pillow. The first thing we looked at for our pillow was the feel of it. Our main objective with our design is the comfortability of it, ensuring that the user will be able to relax and feel great while they're sleeping. With this pillow, it was very easy for Jason to mold the pillow, with him being a stomach sleeper. It was very comfortable to lay down on his stomach with the pillow's stuffing material being a great support. With there being different types of sleepers (stomach, side, back, etc), Jason decided to lay down in different positions to see how comfortable the pillow is. For all positions, the pillow was very easy to sleep on. A great feature about the pillow that Jason noticed was it holds its shape well. For example, if I were to press both of my hands onto the sides of the pillow, they won't be able to touch each other. This shows that the core of filling material is stable and makes sure I receive great support, while also receiving comfort and mold-ability.

Something Jason wanted to test was the cooling feature of the pillow. He noticed that he didn't feel hot at all while using the pillow to lay down. This is because of the fill material providing breathability and cooling through the large volume of air. Not only does the Naturepedic pillow provide a balanced feel for people who sleep in different positions, but the organic and healthy material also provides no flame retardants, which is a significant feature we require for the technology we're adding to the pillow.

The final feature we wanted to take into account in regards to the pillow we want to use is the size of it. The size of the pillow needs to be large enough in order to contain all of the technology our group plans to use. After measuring the length and width of the pillow, it is large enough for us to manipulate it for our design.

7.1.11 PCB Testing

The PCB will need to be able to function as the structure of the main part of the circuitry, so it will be important to make sure that we test it to make sure that everything is correctly made. We will need to check the connections of each component and make sure the parts are placed correctly.

PCB Testing Table

#	Item	Test Method	Criteria
1	PCB	Take a multimeter in continuity mode and check connections	No connections are loose and the multimeter is showing continuity
2	PCB	Look over all the soldering points with a magnifying glass	Make sure that no connections are disconnected
3	PCB	Use multimeter and place nodes around resistors	The resistance matches the resistance in the schematic

7.1.12 Pressure Sensor Testing

7.1.12.1 Pressure Sensor Testing (FSR) - Testing Phase Introduction

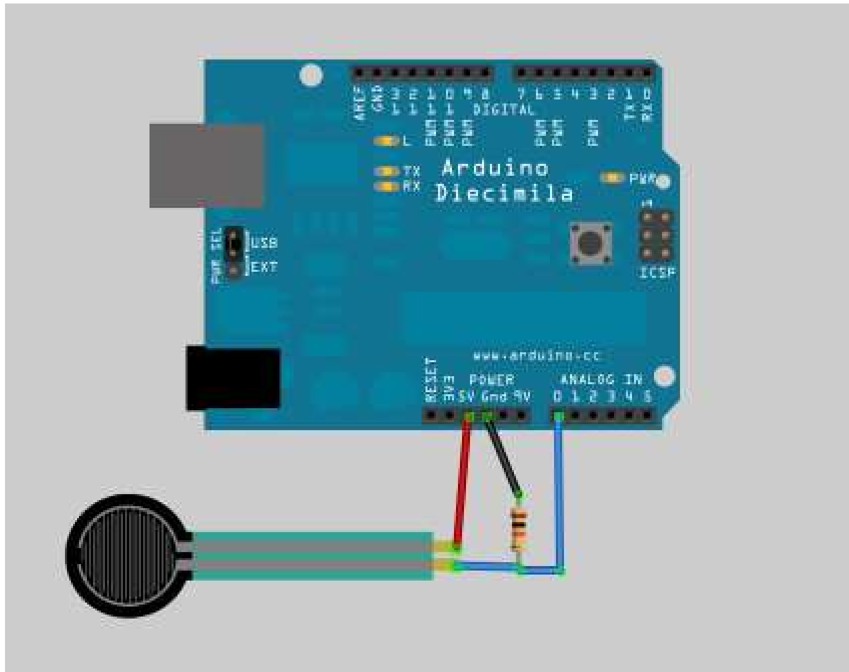
Our team determined that the best way to test the FSR is to connect a multimeter in resistance-measurement mode to the two tabs on our team's FSR and view how resistance changes. Something our team noticed was when the resistance started to change a lot, an auto-ranging meter seemed to help us utilize the FSR and view its measurement.

7.1.12.2 Pressure Sensor Testing (FSR) - Connecting to an FSR

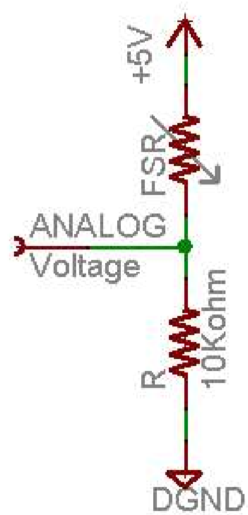
FSRs are resistors, which means they aren't polarized. The good news about this is that our team can connect them in any way we could and it will work. With FSRs being a polymer containing conductive material silk-screened on, the connection tab is compressed on delicate material (with FSRs being plastic). Our team found out that the best way to connect these is to plug them into a breadboard. We were able to use a breadboard that was used in one of our group member's previous courses. Through some research, we found out that it's possible to solder onto the tabs. If we were to do this, we must act quickly because if the iron we're using isn't in good quality, the plastic will melt and can damage the FSR. Because of this, we shouldn't solder directly to our FSR unless we have had enough practice soldering materials.

7.1.12.3 Pressure Sensor Testing (FSR) - Using the FSR

To measure a resistive sensor, we have to connect one end to the power resistor, with the other end connecting to a pulldown resistor to the ground node. Once this is done, the point between the fixed pulldown resistor and the variable FSR resistor is then connected to the analog input of a microcontroller. An example of this is using Arduino.



An example we would like to display is using a 5V supply (we are able to use a 3.3V supply as well, but for this specific example we're using 5V). The analog voltage reading ranges from the ground node (0V) to 5V. This value will mostly depend on the power supply voltage you're utilizing.



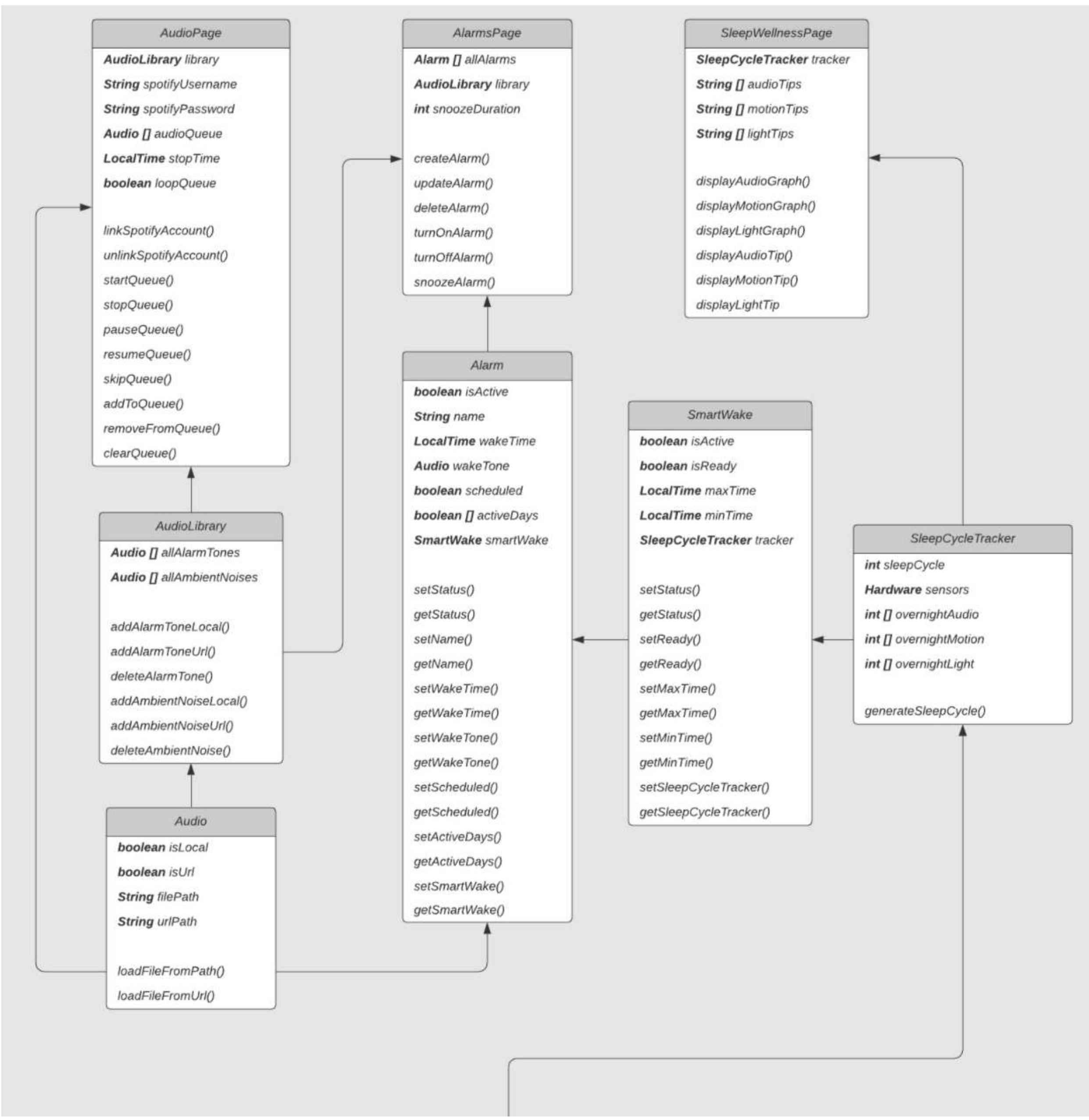
Once the FSR's resistance decreases, the total amount of resistance of the FSR, as well as the pulldown resistor, will decrease from 100Kohm to around 10Kohm

Force (lb)	Force (N)	FSR Resistance	(FSR + R) ohm	Current thru FSR+R	Voltage across R
None	**None**	Infinite	Infinite!	0 mA	0V
0.04 lb	**0.2 N**	30 Kohm	40 Kohm	0.13 mA	1.3 V
0.22 lb	**1 N**	6 Kohm	16 Kohm	0.31 mA	3.1 V
2.2 lb	**10 N**	1 Kohm	11 Kohm	0.45 mA	4.5 V
22 lb	**100 N**	250 ohm	10.25 Kohm	0.49 mA	4.9 V

Within this table, the analog voltage based on the sensor force/resistance with a 5V supply and 10K pulldown resistor

7.2 Software Testing

7.2.1 Application Software



UML Diagram 1

7.2.1.1 AudioPage

- Description
  - UI page for handling all user interactions on the audio tab. This includes adding Audio objects (via the AudioLibrary object), queueing audio, and linking a spotify account.
- Fields
  - **AudioLibrary** library
    - Reference to the audio library
  - **String** spotifyUsername
    - Username used for logging the user into spotify
  - **String** spotifyPassword
    - Password used for logging the user into spotify
  - **Audio []** audioQueue
    - The queue of audio files to play
  - **LocalTime** stopTime
    - Time set by the user to stop playing audio
  - **boolean** loopQueue
    - Specifies if the queue will restart once finished
- Methods
  - linkSpotifyAccount()
    - Allows the user to connect their spotify account to stream music
  - unlinkSpotifyAccount()
    - Allows the user to sign out of their spotify account if connected
  - startQueue()
    - Begin playing Audio items in the queue
  - stopQueue()
    - Stop Audio item currently playing; clear the queue
  - pauseQueue()
    - Pause Audio item; save current position in the queue to be resumed
  - resumeQueue()

- Resume Audio item from last saved position
  - skipQueue()
    - Skip to the next Audio object in the queue; start playing Audio item
  - addToQueue()
    - Insert Audio object at the end of the queue
  - removeFromQueue()
    - Remove audio item from the queue
  - clearQueue()
    - Remove ALL audio items from the queue

7.2.1.2 AudioLibrary

- Description
  - Used to store all alarm tones and ambient noises, as well as make them available to other classes as needed.
- Fields
  - **Audio []** allAlarmTones
    - Holds all Audio objects meant to be used as alarms. This includes native libraries as well as custom Audio objects added by the user
  - **Audio []** allAmbientNoises
    - Holds all Audio objects meant to be used as ambient noise. This includes native libraries as well as custom Audio objects added by the user
- Methods
  - addAlarmToneLocal()
    - Create a new Audio object with the file path provided to the function, then update the allAlarmTones array with the new Audio object.
  - addAlarmToneUrl()
    - Create a new Audio object with the url provided to the function, then update the allAlarmTones array with the new Audio object.
  - deleteAlarmTone()
    - Remove an existing Audio object from the allAlarmTones array, then delete the Audio object.
  - addAmbientNoiseLocal()

- Create a new Audio object with the file path provided to the function, then update the allAmbientNoisesarray with the new Audio object.
- addAmbientNoiseUrl()
  - Create a new Audio object with the url provided to the function, then update the allAmbientNoisesarray with the new Audio object.
- deleteAmbientNoise()
  - Remove an existing Audio object from the allAmbientNoises array, then delete the Audio object.

7.2.1.3 Audio

- Description
  - Used to store information for accessing a single audio file as well as methods to load said file.
- Fields
  - **boolean** isLocal
    - Specifies if the audio file is located in local storage (in the android file system)
  - **boolean** isUrl
    - Specifies if the audio file is accessed from the internet via URL
  - **String** filePath
    - Local path to the audio file
  - **String** urlPath
    - URL to access the audio file
- Methods
  - loadFileFromPath()
    - Loads the audio file from local storage using the path stored in filePath
  - loadFileFromUrl()
    - Loads the audio file from the internet using the URL stored in urlPath

7.2.1.4 AlarmsPage

- Description

- UI page for handling all user interactions on the alarms tab. This includes creating, updating, and deleting alarms.
- Fields
  - **Alarm []** allAlarms
    - Holds all custom alarms created and saved by the user.
  - **AudioLibrary** library
    - Reference to the audio library.
  - **int** snoozeDuration
    - The duration (in minutes) an alarm will snooze for.
- Methods
  - createAlarm()
    - Create a new Alarm object with settings provided by the user, then add it to the allAlarms array.
  - updateAlarm()
    - Update an existing Alarm object in the allAlarms array with new settings provided by the user.
  - deleteAlarm()
    - Remove an existing Alarm object from the allAlarms array, then delete the Alarm object.
  - turnOnAlarm()
    - Set an alarm from the allAlarms array to active.
  - turnOffAlarm()
    - Set an alarm from the allAlarms array to inactive. Disable snooze if set.
  - snoozeAlarm()
    - Snooze the current active alarm. Alarm will activate again x minutes from now (specified by snoozeDuration)

7.2.1.5 Alarm

- Description
  - Used to store settings for an alarm as well as getter and setter methods for those settings.
- Fields
  - **boolean** isActive
    - Specifies if the alarm is active.

- **String** name
  - User-given label to the alarm.
- **LocalTime** wakeTime
  - Time at which the alarm will trigger.
- **Audio** wakeTone
  - Audio file that will play once the alarm is triggered.
- **boolean** scheduled
  - Specifies if the alarm should repeat for certain days of the week.
- **boolean []** activeDays
  - Array of size 7 representing days of the week the alarm is active for (index 0 = Monday).
- **SmartWake** smartWake
  - Reference to the smartWake object

- Methods
  - setStatus()
    - Set the value of the isActive field of the alarm object.
  - getStatus()
    - Get the value of the isActive field of the alarm object.
  - setName()
    - Set the value of the name field of the alarm object.
  - getName()
    - Get the value of the name field of the alarm object.
  - setWakeTime()
    - Set the value of the wakeTime field of the alarm object.
  - getWakeTime()
    - Get the value of the wakeTime field of the alarm object.
  - setWakeTone()
    - Set the value of the wakeTone field of the alarm object.
  - getWakeTone()
    - Get the value of the wakeTone field of the alarm object.
  - setScheduled()
    - Set the value of the scheduled field of the alarm object.
  - getScheduled()
    - Get the value of the scheduled field of the alarm object.
  - setActiveDays()

- Set the value of the activeDays field of the alarm object.
  - getActiveDays()
- Get the value of the activeDays field of the alarm object.
  - setSmartWake()
- Set the value of the smartWake field of the alarm object.
  - getSmartWake()
- Get the value of the smartWake field of the alarm object.

7.2.1.6 SmartWake

- Description
  - Used to store settings for the SmartWake feature as well as getter and setter methods for those settings.
- Fields
  - **boolean** isActive
    - Specifies if SmartWake is active.
  - **boolean** isReady
    - Specifies if the user is ready to be woken up.
  - **LocalTime** maxTime
    - The upper bound for the latest time the user would like to wake up from their alarm.
  - **LocalTime** minTime
    - The lower bound for the earliest time the user would like to wake up from their alarm.
  - **SleepCycleTracker** tracker
    - Reference to the sleep tracker
- Methods
  - setStatus()
    - Set the value of the isActive field of the SmartWake object.
  - getStatus()
    - Get the value of the isActive field of the SmartWake object.
  - setReady()
    - Set the value of the isReady field of the SmartWake object.
  - getReady()

- Get the value of the isReady field of the SmartWake object.
  - setMaxTime()
    - Set the value of the maxTime field of the SmartWake object.
  - getMaxTime()
    - Get the value of the maxTime field of the SmartWake object.
  - setMinTime()
    - Set the value of the minTime field of the SmartWake object.
  - getMinTime()
    - Get the value of the minTime field of the SmartWake object.
  - setSleepCycleTracker()
    - Set the value of the tracker field of the SmartWake object.
  - getSleepCycleTracker()
    - Get the value of the tracker field of the SmartWake object.

7.2.1.7 SleepWellnessPage

- Description
  - UI page for displaying sleep wellness data to the user. This includes creating and displaying charts of overnight data collection, as well as providing the user tips to achieve a better overall sleep experience.
- Fields
  - **SleepCycleTracker** tracker
    - Reference to the sleep tracker
  - **String []** audioTips
    - Collection of tips to improve the audio conditions for the user.
  - **String []** motionTips
    - Collection of tips to improve the user's sleeping position.
  - **String []** lightTips
    - Collection of tips to improve the lighting conditions for the user.
- Methods
  - displayAudioGraph()
    - Display a graph of the audio data collected throughout the night. Data accessed through the SleepCycleTracker object.
  - displayMotionGraph()

- Display a graph of the motion data collected throughout the night. Data accessed through the SleepCycleTracker object.
- displayLightGraph()
  - Display a graph of the ambient light data collected throughout the night. Data accessed through the SleepCycleTracker object.
- displayAudioTip()
  - Display a random audio tip to the user from the audioTips array.
- displayMotionTip()
  - Display a random motion tip to the user from the motionTips array.
- displayLightTip()
  - Display a random light tip to the user from the lightTips array.

7.2.1.8 SleepCycleTracker

- Description
  - Used to stored sensor data overnight and used said data to generate a prediction as to what sleep cycle the user is currently in.
- Fields
  - **int** sleepCycle
    - The estimated current sleep cycle of the user based on the data provided.
  - **Hardware** sensors
    - Reference to the hardware components
  - **int []** overnightAudio
    - Holds the audio data collected overnight.
  - **int []** overnightMotion
    - Holds the motion data collected overnight.
  - **int []** overnightLight
    - Holds the ambient light data collected overnight.
- Methods
  - generateSleepCycle()
    - Uses the data polled from the active sensors to generate an estimate as to what sleep cycle the user is currently in, then updates the sleepCycle field to reflect this.

7.2.2 Application Software Testing

7.2.2.1 AudioLibrary testing

- Description
  - Check to make sure audio files can be successfully added to and removed from the user's library
- Methods
  - testAddToneLocal()
    - Add a sample audio file to the user's profile through a local file path on the Android system
  - testAddToneUrl()
    - Add a sample audio file to the user's profile through a URL provided by the user
  - testDeleteTone()
    - Remove a saved audio file from the user's profile

7.2.2.2 Audio testing

- Description
  - Check to make sure audio files can be loaded in and played correctly
- Methods
  - testFromPath()
    - Load a sample audio file through a local file path on the Android system
  - testFrom URL()
    - Load a sample audio file via a URL

7.2.2.3 Alarm testing

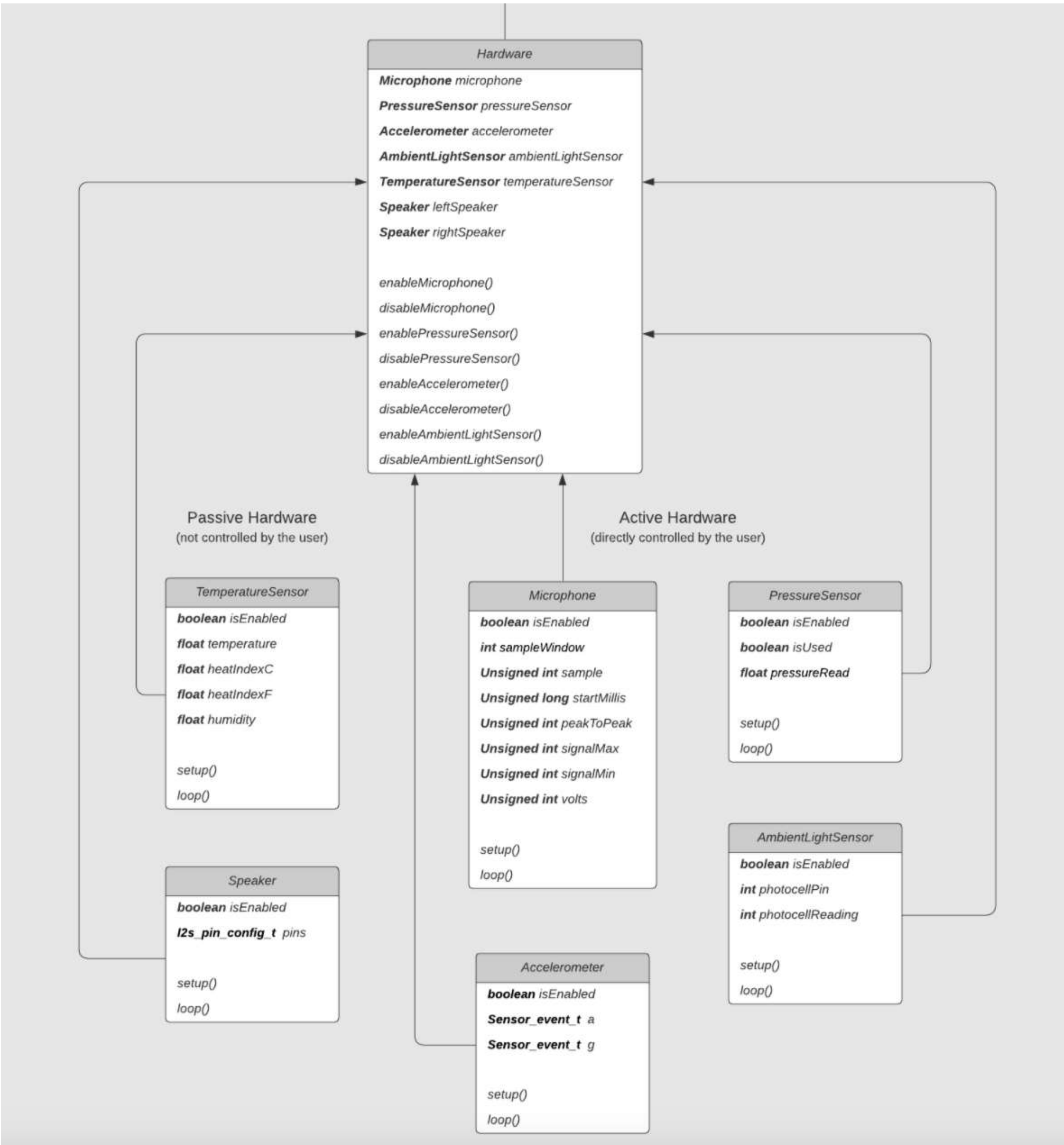
- Description
  - Check to make sure the app is creating new alarms, updating, and deleting existing alarms
- Methods
  - testCreateAlarm()

- Create a new alarm with settings passed from the test. New alarm should be displayed in the user's profile
  - testUpdateAlarm()
    - Update an existing alarm created by the user. The alarm should update successfully with the new settings provided by the test
  - testDeleteAlarm()
    - Delete an existing alarm created by the user. Alarm should no longer appear in the user's profile

7.2.2.4 SleepCycleTracker testing

- Description
  - Check to make sure the pillow is accurately making the correct predictions as to what sleep cycle the user is in
- Methods
  - testAll()
    - Generate sleep cycle predictions based on all sensor data
  - testNoAudio()
    - Generate sleep cycle predictions based on all sensor data excluding the microphone
  - testNoMotion()
    - Generate sleep cycle predictions based on all sensor data excluding the accelerometer
  - testNoLight()
    - Generate sleep cycle predictions based on all sensor data excluding the ambient light sensor

7.2.3 Hardware Software



UML Diagram 2

7.2.3.1 Hardware

- Description
  - Collection of all the active sensors (directly controlled by the user) and passive sensors (not controlled by the user), as well as methods to enable or disable any active sensors.
- Fields
  - **Microphone** microphone
    - Reference to the microphone.
  - **PressureSensor** pressureSensor
    - Reference to the pressure sensor.
  - **Accelerometer** accelerometer
    - Reference to the accelerometer.
  - **AmbientLightSensor** ambientLightSensor
    - Reference to the ambient light sensor.
  - **TemperatureSensor** temperatureSensor
    - Reference to the temperature sensor.
  - **Speaker** leftSpeaker
    - Reference to the left speaker of the device.
  - **Speaker** rightSpeaker
    - Reference to the right speaker of the device.
- Methods
  - enableMicrophone()
    - Sets the microphone to active. The sensor will continue recording data in this state.
  - disableMicrophone()
    - Sets the microphone to inactive. Data from the sensor will not be recorded in this state.
  - enablePressureSensor()
    - Sets the pressure sensor to active. The sensor will continue recording data in this state.
  - disablePressureSensor()
    - Sets the pressure sensor to inactive. Data from the sensor will not be recorded in this state.
  - enableAccelerometer()

- Sets the accelerometer to active. The sensor will continue recording data in this state.
- disableAccelerometer()
  - Sets the accelerometer to inactive. Data from the sensor will not be recorded in this state.
- enableAmbientLightSensor()
  - Sets the ambient light sensor to active. The sensor will continue recording data in this state.
- disableAmbientLightSensor()
  - Sets the ambient light sensor to inactive. Data from the sensor will not be recorded in this state.

7.2.3.2 Microphone

- Fields
  - **boolean** isEnabled
    - Set to true if the sensor is active.
  - **int** sampleWindow
    - Sample window width in mS (50mS = 20Hz)
  - **Unsigned int** sample
    - Store sample values of microphone
  - **Unsigned long** startMillis
    - Start of sample window
  - **Unsigned int** peakToPeak
    - Peak-to-peak level
  - **Unsigned int** signalMax
    - Max signal value
  - **Unsigned int** signalMin
    - Min signal value
  - **double** volts
    - Store volts value
- Methods
  - setup()
    - Initialize the serial debugging at a baud rate of 9600
  - loop()
    - Loop for 50mS and collect analog readings

- Store only the max and min levels
- Calculate peak to peak (signalMax - signalMin)
- Calculate Volts

7.2.3.3 PressureSensor

- Fields
  - **boolean** isEnabled
    - Set to true if the sensor is active.
  - **boolean** isUsed
    - Set to true if the pressure sensor detects weight on the pillow.
  - **float** pressureRead
    - Store analog date received from sensor.
- Methods
  - setup()
    - Initialize the serial debugging at a baud rate of 9600
  - loop()
    - Set delay time
    - Read pressure data using the readPressure method and store into PressureRead
    - Transmit data to the app.

7.2.3.4 Accelerometer

- Fields
  - **boolean** isEnabled
    - Set to true if the sensor is active.
  - **Sensor_event_t** a
    - Store event readings of the accelerometer
  - **Sensor_event_t** g
    - Store event readings of the gyroscope
  -
- Methods
  - setup()
    - Initialize the serial debugging at a baud rate.
    - Initialize the sensor
    - Set accelerometer measurement range

- Set gyroscope measurement range
- Set filter bandwidth
- loop()
- Get sensor events
- Get readings, measurement default to meter per second square
- Get gyroscope readings
- Set delay time
- Libraries
- Adafruit Bus IO
- Library used to abstract away UART, I2C and SPI interfacing.

7.2.3.5 AmbientLightSensor

- Fields
    - **boolean** isEnabled
        - Set to true if the sensor is active.
    - **int** photocellPin
        - What pin the cell and pull-down resistor are connected to
    - **int** photocellReading
        - Analog reading from the analog resistor divider
- Methods
    - setup()
        - Initialize the serial debugging at a baud rate of 9600
    - loop()
        - Using AnalogRead method, save values in PhotocellReading
        - Set thresholds to determine the current light level
        - Set delay to 1000 ms (1 second)
        - Transmit data to the app.

7.2.3.6 TemperatureSensor

- Fields
    - **boolean** isEnabled
        - Set to true if the sensor is active.
    - **float** temperature

- Get humidity using the readTemperature() method on the DHT object.
- Read in Celsius by default, if we want in Fahrenheit we would pass the value 'True' in the readTemperature(True) method.
  - **float** heatIndexC
- Using the computeHeadIndex method from the library, get the heat index in Celsius.
  - **float** heatIndexF
- Using the computeHeadIndex method from the library, get the heat index in Fahrenheit.
  - **float** humidity
- Get humidity using the readHumidity() method on the DHT object.
- Methods
  - setup()
    - Initialize the serial debugging at a baud rate of 9600
    - Initialize DHT sensor
  - loop()
    - Set delay 1000 ms (1 second) as the DHT11 sampling period is 1 second, so we can get readings every second.
    - Check if the sensor returns valid humidity and temperature values.
    - Transmit data to the app.

- Libraries
  - Adafruit Unified Sensor Library
  - DHT Library (Adafruit)

7.2.3.7 Speaker

- Fields
  - **boolean** isEnabled
    - Set to true if the sensor is active.
  - **I2s_pin_config_t** pins
    - Used to define pins

- Methods

- setup()
  - Initiate bluetooth connection with speaker
  - Define pins by calling set_pin_config method
  - Output to internal DAC of the ESP32
  - Access sink data stream with callbacks
  - Support for metadata
  - Support for AVRC commands
- loop()
- Libraries
  - Arduino A2DP Library
    - Used to receive music with an ESP32 board via bluetooth

7.2.4 ESP32 Microcontroller Testing

- Description
  - The first important part we need to test is the communication to and from the ESP32 microcontroller. For this, we must test the Bluetooth capabilities that the ESP32 has. For this project we will be implementing the Arduino IDE. Before testing the Bluetooth, we must add some code to test that the microcontroller is initiating and currently active. We can do this by turning on and off a LED light on the microcontroller. As described in the previous testing section, we will apply the Bluetooth serial library with a Baud rate of 115200. Using a Bluetooth serial reader app, we can easily test the connection. With this, we can make sure that the connection to the ESP32 is successful and stable.

- Fields
  - **BluetoothSerial** SerialBT
    - To use Bluetooth Serial and test it.
  - **char** incommingChar
    - To read SerialBT.

- Methods
  - testDevice()

- Test if ESP32 can turn on and off by turning LED on and off, and if the ESP32 is active.
  - readSerial()
    - Code to read incoming Bluetooth serial messages.

7.2.4.1 Microphone Testing

- Description
  - For the microphone, we have determined that a sample window of 50 milliseconds is enough. We compute the difference between the minimum and maximum samples, convert it to volts. With this we will be able to see the values of the voltage as it varies to the sounds in the environment.
  - Given that we calculate the voltage between the minimum and maximum samples, we'll need to verify that the computation is correct, by this we use some test values to verify that the computation for voltage is working.
  - Additionally, we'll need to check if the microphone is correctly saving only the max and min signals, by inserting values and check if it is saving the max and min.

- Fields
  - **boolean** isEnabled
    - Set to true if the sensor is active. We use this to test if the sensor works.
  - **int** sampleWindow
    - Sample window width in mS (50mS = 20Hz)
  - **Double** testVal
    - Test values to check voltage computation.
  - **Double** expResult
    - Expected voltage computation value that we expect to compare to.
  - **double** volts
    - Store volts value
- Methods
  - testDevice()
    - Test if the sensor can turn on and off.
  - testVoltage()
    - Test that it loops for 50 ms

- Test if code only stores only the max and min levels using test values
- Test that voltage computation is working correctly using sample values.

7.2.4.2 Pressure Sensor Testing

- Description
  - For the pressure sensor testing, we must make sure that the sensor correctly reads when there is pressure put on the pillow.
  - We can test the pressure by measuring in hPa (hectopascal), we can read the values and check If they correspond to the correct pressure depending on the items placed on the pillow.

- Fields
  - **boolean** isEnabled
    - Set to true if the sensor is active. Allows to test if the sensor can be turned on and off.
  - **boolean** isUsed
    - Set to true if the pressure sensor is currently reading values due to pressure being on the pillow.
  - **float** pressureRead
    - Store analog date received from sensor.
- Methods
  - testDevice()
    - Test if the sensor can turn on and off.
  - testPressure()
    - Test if the values given by the pressureRead are appropriate to the actual pressure on the pillow.

7.2.4.3 Accelerometer Sensor Testing

- Description
  - For the accelerometer, we must test that it can correctly test x, y, and z axis directions. This would help us detect motion when the pillow is used.

We must make sure that no data is received when there is no movement, and that data is received when movement is detected.

- We must also make sure that the gyroscope can measure the orientation and angular velocity correctly.

- Fields
  - **boolean** isEnabled
    - Set to true if the sensor is active.
  - Sensor_event_t a
    - Store event readings of the accelerometer
  - Sensor_event_t g
    - Store event readings of the gyroscope
  -
- Methods
  - testDevice()
    - Test that sensor can turn on and off
  - testReadings()
    - make sure values are correctly outputting depending on the movement on the pillow.

7.2.4.4 AmbientLight Sensor Testing

- Description
  - For the ambient light sensor, we must test that the device can turn on and off. Additionally, we must test that the values must fall within the range of 0-1000. If using lumens, we must make sure that the calculations are correct using test values.

- Fields
  - **boolean** isEnabled
    - Set to true if the sensor is active.
  - **int** max
    - Max range value allowed (1000).
  - **int** min
    - Min range value allowed (0)
  - **int** photocellReading

- Analog reading from the analog resistor divider
  - **Int** testVal
    - Value to test lumens calculation.
- Methods
  - testDevice()
    - Test device on and off.
  - testReading()
    - Test that the range falls within 0-1000 and that the lumen function works correctly.

7.2.4.5 TemperatureSensor Testing

- Description
  - For the temperature sensor, we must test that the conversion of Celsius and Fahrenheit works correctly. Also make sure that the sensor returns valid results for humidity and temperature.

- Fields
  - **boolean** isEnabled
    - Set to true if the sensor is active.
  - **float** temperature
    - Get temperature using the readTemperature() method on the DHT object.
  - **float** humidity
    - Get humidity using the readHumidity() method on the DHT object.
  - **float** testVal
    - Test value to check if conversions are correct.
- Methods
  - testDevice()
    - Check if the sensor can turn on and off.
  - testValues()
    - Test that conversion works and that values for humidity and temperature are valid.

7.2.4.6 Speaker testing

- Description
  - For the speaker we must test that it can correctly output audio and support Bluetooth connection

- Fields
  - **boolean** isEnabled
    - Set to true if the sensor is active.
- Methods
  - testDevice()
    - Test Bluetooth compatibility.
    - Test that audio outputs correctly.

7.3 Facilities and Equipment

When it comes to facilities. We are allowed to use the Senior Design Laboratory that acts as a workshop for students. There is instrumentation, equipment, and software for the students to design and build their project. The following contains the equipment that the senior design laboratory contains:

- Tektronix Oscilloscopes
- Tektronix Dual Arbitrary Function Generators
- Tektronix DMM 4050 Digital Multimeters
- Keithley 2230-30-1 Triple-Channel Power Supplies
- Dell Precision 3420 Computers
- SMD Rework Station
- Soldering and Desoldering Stations
- Digital Microscope Inspection Station

For this project, we'll need to design and order a PCB. For this, the university provides software that assists us with the design of the schematic of a PCB. We can use the following software:

- Autodesk Eagle
  - Free resource, eagle is a scriptable electronic design automation application with schematic capture, printed circuit board layout, auto-router, and computer-aided manufacturing features.

- Autodesk Fusion 360
  - Priced software that is a cloud-based 3D modeling, CAD, CAM, CAE, and PCB software platform for product design and manufacturing.

For ordering a custom PCB design we have many options from online sources to order from. We will consider the following providers when considering ordering PCBs from: pcbway, and jlcpcb.
The main benefit with pcbway is that it provides instant quoting. We can just upload our design and get a quick pricing for it. Additionally, every Gerbers, BOMs, Parts uploaded to the site get expert design for manufacturability feedback. This includes instant, automated warnings, and compare demands/specifications against capabilities to ensure compliance and determine the process steps and associated checks.

When it comes to jlcpcb, it offers cheap PCB assembly ($7). Jlcpcb also has instant quoting, and free design for manufacturability feedback. Additionally, the build time is quick, that of 1 day. With the estimated delivery time to be 2-4 days.

Both pcbway and jlcpcb prove themselves to be great resources for ordering PCBs and both are reliable. Either one would work for this project.

Section 8: Administrative Content

8.1 Budgeting

Item	Price Estimate
Temperature Sensor	\$5-\$10
Accelerometer Sensor	\$5-\$20
Pressure Sensor	\$5-\$20
Wireless Transceiver	\$5-\$20
Memory foam	\$20-\$40
2AA Battery Holder	\$2-\$5
Pillow	\$10-\$40
PCB	\$80-\$120
Stereo Speakers	\$20-\$50
Ambient light sensor	\$5-\$10
Microphone Module	\$5-\$10
USB-C Charging Module	\$5-\$10
Volume Control Module	\$5-\$20
Total =	\$172-\$375

This project has a budget limit of \$500, it will be self-funded by the group members. The prices are estimates based on online research for the items listed. The price can vary depending on the number of parts we need but generally the prices are not expensive for each individual item. The price estimate will likely change as we continue to develop the project.

8.2 Project Milestones

In this section, we are going to run through our thought process on the milestones we designed to complete this project efficiently. To do this, we laid out everything our group needed to accomplish, and strategically determined when each task needed to be completed.

4.18.1 Senior Design: SD Project Ideas and Project Finalization

For these milestones, we needed to determine what project our group wanted to work on for the duration of our time in Senior Design. In order to determine a project we wanted to work on, our group communicated through the Discord app (application individuals can use to communicate). After a discussion, we determined that the Snooze Pillow was our most preferred project.

4.18.2 Main Project Document: Divide and Conquer

For this milestone, we had to complete a document with a total page count limited to 10 pages that contains content that would jumpstart our 120-page document. One of the purposes of the Divide and Conquer document is to help our group understand what it means to be a team. Some ways this document helped us do that was: appointing different sections to individuals, setting deadlines to ensure our group members are being punctual, and brainstorming as a collective on how to tackle each section required from the Divide and Conquer document. This document helped each group member understand how to collaborate with each other, as well as develop the time management skills that are usually required for large projects like our Snoozie Pillow.

4.18.3 60 Page Draft of Main Project Document

Within this milestone, we had 5 weeks to write 60 pages worth of content for our project. For this deadline, we met as a group on Discord and discussed how we're dividing each section to 4 group members. To do this, we laid out everything we needed to write about (similar to the Divide and Conquer document, but a little more extensive), and made a to-do list of tasks for each group member. For example, once the group had a conversation of everything that needed to be done for the project, Jason was tasked to complete the following: Milestone expansion, Serial Communication, Speakers, Pillow, Housing, Pressure Sensor, etc. The group believes that assigning tasks to each individual was more efficient than completing tasks in a "free-for-all" type of format, where more than 1 person would work on the speaker's portion of the project for example. Assigning tasks to each individual allowed our group members to "shine" in their roles where they best seem fit, allowing people with knowledge about certain aspects of the "Snoozie" to be able to conduct research and carry out testing. For our team, we were able to successfully exceed our own expectations of the 60 page draft, with our group writing about 67 pages before the deadline was due. A method towards our group's success in meeting the deadline was through our time management and collaborative skills. For example, a few of our group members used "Google Calendar" in order to create time slots throughout their day to commit time towards the 60 page draft. Through collaboration, our group made sure to communicate with each other if they were struggling with research or meeting a certain page count by the end of a self-assigned deadline. Through this communication, we were able to work together and help build each other as a team to complete the 60 page draft deadline

4.18.4 Individual Snoozie Part Milestones

In this portion of milestone expansion, we would like to discuss “part” milestones as a whole. For the Snoozie, there are several parts our group has done research for, as well as tested in order to fulfill the Snoozie's design requirements. Some of these parts include: Thermal sensors, pressure sensors, sleep tracker, motion sensors, etc. Each group member is assigned a part they need to research and test, ensuring that no individual is given an unfair amount of work to do. Once a group member is assigned a part, they will have a set amount of time (determined as a group via Discord) to do research and test those parts. For example, Jason was provided the "Speakers" portion of the Snoozie. What Jason had to do from there is conduct research on different speakers that would be compatible with the Snoozie. After that, he would compare each speaker he conducted research for, then test the speaker that he believes is the most compatible with the Snoozie. It is important to emphasize that a group member is required to research more than one product that will potentially be implemented into the Snoozie. Researching and comparing multiple products allows the team to view a multitude of options, determining the one product that most aligns to Snoozie's purpose. If a group member only decided to research and test 1 product, the product can potentially be incompatible to the Snoozie. Researching multiple products allows the group to see which product is the most effective as well as being the most compatible with the project.

Milestone #	Deadline Name	Start Date	Deadline	Status of Deadline	Individual Responsible
	Senior Design 1				
1	SD Project Ideas	8/23/2021	8/27/2021	Completed	Group 46
2	Project Finalization	9/3/2021	9/9/2021	Completed	Group 46
	Main Project Document				
3	Initial Document - Divide and Conquer	9/7/2021	9/17/2021	In progress	Group 46
4	Updated Divide & Conquer Document	9/25/2021	10/1/2021		Group 46
5	Table of Contents	9/21/2021	10/19/2021	Unassigned	Group 46
6	60 Page Draft of Main Project Document	9/27/2021	11/5/2021	Unassigned	Group 46
7	100 Page Report Submission (updated) of Main Project Document	11/8/2021	11/19/2021	Unassigned	Group 46
8	Final product of Main Project Document	11/22/2021	12/7/2021	Unassigned	Group 46
	Research, Documentation & Design				
9	Thermal Sensors	9/27/2021	10/11/2021	In development	Christian
10	Microphone Installation	9/27/2021	10/11/2021	In development	Jason
11	Machine Learning Algorithm	9/27/2021	10/11/2021	In development	David
12	Sleep Tracker	9/27/2021	10/11/2021	In development	Kevin
13	Auxiliary Controls	9/27/2021	10/11/2021	In development	Jason
14	App creation	10/18/2021	11/6/2021	In development	Group 46
15	Passive Cooling System	10/18/2021	11/6/2021	In development	David
16	Motion sensor	10/18/2021	11/6/2021	In development	Kevin
17	Pressure Sensor	10/18/2021	11/6/2021	In development	Christian
18	Power supply	10/18/2021	11/17/2021	In development	Jason
19	Purchasing & Testing of Supplies	11/17/2021	11/26/2021	In development	Group 46
	Senior Design II				
20	Prototype Development	11/30/2021	12/3/2021	In development	Group 46
21	Testing Phase	TBD	TBD		Group 46
22	Finalize Prototype	TBD	TBD		Group 46
23	Project Presentation (Peer)	TBD	TBD		Group 46
24	Report Deadline	TBD	TBD		Group 46
25	Final Presentation	TBD	TBD		Group 46

Section 9: Project Summary

9.1 Project Summary

Our project, the Snoozie, will be a functioning smart pillow that will help its users with all things pillow related in a comfortable, affordable and sleek way that our users will hopefully enjoy. As stated above, we will include many features such as a sleep wellness tracker, movement sensor to know when the user is moving, a temperature sensor, a light level sensor, microphone to record audio, and speakers to play music. We hope that with these features added to a normal pillow, the user will have a greatly improved experience when using the Snoozie over other traditional pillows. The design process has gone smoothly as we set goals and constraints for ourselves to make sure the Snoozie is functional, useful and can be assembled by use with minimal effort. We want the pillow to reflect the work and experience that we all as a team put into making it.

We hope to be able to work as a team to be able to make this product a reality. Each of our team members share the common goal of making the Snoozie as good of a product as it can be, and with that said, our team will be working fervently to accomplish this goal. The work has been separated out and planned to accommodate each person’s strengths and weaknesses to make sure that each group member has a good experience working on the project. We hope to gain as much experience as possible when making this product, both in design and assembly, and to overcome the many unforeseen challenges that will show up when bringing this product into reality.

9.2 Project Conclusion

As stated in the document above, we hope to be able to give the user a comfortable sleeping solution that they can use to improve their sleep, improve the time the user spends before and after sleeping on the pillow, and track and improve the sleeping patterns of the user to better improve their sleep. We think that with the pillow along with the accompanied companion app we can achieve the goals that we set out to achieve. All in all, we want to make sure that we as a team can learn and improve ourselves by working on this project, so that we can bring the skills we learn here into the world.

Section 10: Appendices

10.1 References

Smart Pillow technology:

- https://remfit.com/pages/rem-fit-technology

Lecture 5 - The Requirement for Specification:

- https://webcourses.ucf.edu/courses/1384424/pages/lecture-5-abet1-the-requirement-for-specification

Google Shopping - Price estimate for electronics:

- https://www.google.com/search?q=microcontroller+board&rlz=1C5CHFA_enUS872US872&biw=1440&bih=708&tbm=shop&sxsrf=AOaemvJhBbLFSXVGfdAH2t9KOPeUdwN1Jw%3A1631822306587&ei=4qFDYd-Il5nA0PEPidaa0A0&oq=microcontroller+board&gs_lcp=Cgtwcm9kdWN0cy1jYxADMgUIABCABDIFCAAQgAQyBQgAEIAEMgQIABAYMgQIABAYMgQIABAYMgQIABAYMgQIABAYMgQIABAYMgQIABAYOgQIIxAnOgQIABBDOgoIABCxAxCDARBDOgsIABCABBCxAxCDAToECAAQAzoNCAAQgAQQsQMQgwEQCjoHCAAQgAQQCIDCJFjsdWC8dmgCcAB4AIABsgGIAZkSkgEEMTcuN5gBAKABAcABAQ&sclient=products-cc&ved=0ahUKEwjfnq7no4TzAhUZIDQIHQmrBtoQ4dUDCAs&uact=5

Sleep Facts:

- https://www.ninds.nih.gov/Disorders/Patient-Caregiver-Education/Understanding-Sleep

Sleep Foundation best pillows:

- https://www.sleepfoundation.org/best-pillows

Pillow size data:

- https://allaboutdown.com/pages/products-down-pillows-fill-weights-work-on

Memory Foam Pillows:
- https://mindinsiders.com/how-much-does-memory-foam-pillow-weigh/

Effects of Audio on Sleep:
- https://amerisleep.com/blog/sound-impacts-sleep-cycle/

Effects of Light on Sleep:
- https://www.sleep.org/how-lights-affect-sleep/

How Sleep Trackers Work:
- https://www.hopkinsmedicine.org/health/wellness-and-prevention/do-sleep-trackers-really-work
- https://smartifylife.com/how-do-sleep-trackers-work/

Sleep Cycles:
- https://www.verywellhealth.com/the-four-stages-of-sleep-2795920
- https://www.healthline.com/health/healthy-sleep/stages-of-sleep
- https://www.sleepfoundation.org/how-sleep-works/stages-of-sleep
- https://www.sleepfoundation.org/sleep-apnea/sleep-respiratory-rate

Photoresistor links:
- https://learn.adafruit.com/adafruit-io-basics-analog-input/arduino-setup
- https://learn.adafruit.com/photocells/using-a-photocell
- https://invootech.blogspot.com/2017/06/how-to-convert-ldr-dependent-resistor.html

Temperature sensor links:
- https://randomnerdtutorials.com/esp32-dht11-dht22-temperature-humidity-sensor-arduino-ide/

- https://learn.adafruit.com/adafruit-io-basics-temperature-and-humidity/arduino-setup
- https://howtomechatronics.com/tutorials/arduino/dht11-dht22-sensors-temperature-and-humidity-tutorial-using-arduino/

Microphone:
- https://www.adafruit.com/product/1713
- https://learn.adafruit.com/adafruit-microphone-amplifier-breakout/measuring-sound-levels

Accelerometer:
- https://randomnerdtutorials.com/esp32-mpu-6050-accelerometer-gyroscope-arduino/
- https://learn.adafruit.com/adxl345-digital-accelerometer/programming

Speakers:
- https://github.com/pschatzmann/ESP32-A2DP
- https://www.bestbuy.com/site/anker-soundcore-mini-3-pro-portable-bluetooth-speaker-black/6457700.p?skuId=6457700
- https://www.cnet.com/tech/home-entertainment/best-portable-mini-bluetooth-speaker/

Pillow
- https://www.sleepadvisor.org/best-pillows/
- https://www.amazon.com/Naturepedic-Organic-Cotton-PLA-Pillow-Standard/dp/B00AXFYAH6

Serial Communications
- https://www.contec.com/support/basic-knowledge/daq-control/serial-communicatin/#anc-01
- https://learn.sparkfun.com/tutorials/i2c/all

- https://learn.sparkfun.com/tutorials/serial-communication/uarts
- https://www.sparkfun.com/datasheets/BreakoutBoards/I2SBUS.pdf
- https://learn.sparkfun.com/tutorials/serial-peripheral-interface-spi/all

Pressure Sensor

- https://www.servoflo.com/applications/sensors-for-medical-use/respirator-and-breath-detection
- https://www.interlinkelectronics.com/force-sensing-resistor

Arduino

- https://www.arduino.cc/en/guide/introduction

Microcontroller

- https://www.espressif.com/en/products/socs/esp32
- https://www.adafruit.com/product/3405
- https://www.espressif.com/en/support/documents/technical-documents