SLEEP TRACKING ALARM

By

Elliot Salvaggio

Kishan Surti

Rutu Patel

Final Report for ECE 445, Senior Design, Spring 2020

TA: Shuai Tang

8 May 2020

Project No. 43

Abstract

This report provides a detailed look into the following objective: create a better solution to the problem that is waking up in the morning. The previous team's solution was to create a quiz alarm that wakes the user by asking a set of pre-inserted questions in the morning, aiming to spark brain activity. Our solution is still an alarm, but with a completely different design, consisting of a wrist wearable and camera system. The wearable includes sensors to track user's sleep patterns to determine the optimal time to wake them, which is in their lightest sleep. This helps the user wake up feeling more awake and less groggy. The paired camera system analyzes the user's posture to determine if the user has successfully moved out of their bed, at which point the alarm is finally disabled.

Contents

1. Second Project Motivation 1
1.1 Updated Problem Statement 1
1.2 Updated Solution 1
1.3 Updated High Level Requirements 2
1.4 Updated Visual Aid 2
1.5 Updated Physical Design 3
1.6 Updated Block Diagram 4
2 Second Project Implementation 5
2.1 Implementation Details and Analysis 5
2.2 Implementation of Software 5
2.3 Circuit Analysis and Schematic 9
2.4 Mathematical Analysis 10
2.5 Mathematical Modeling (Ground truth values for Wearable Device) 13
2.6 Bill of Materials 13
3. Second Project Conclusions 15
3.1 Implementation Summary 15
3.2 Unknowns, uncertainties, testing needed 15
3.3 Ethics and Safety 15
3.4 Project Improvements 16
References 17
Appendix A Requirement and Verification Table 18

1. Second Project Motivation

1.1 Updated Problem Statement

According to a study done in the United States, 58% of people reported they spend more than five minutes in bed after turning off their alarm in the morning. Additionally, 57% say they still feel tired after waking up and only 33% describe their experience waking up as good [1]. Clearly, waking up is difficult for many people, and many wake up feeling tired or groggy. Turning off the alarm on a phone or hitting snooze is so easy. This does not force a person to get up and out of bed, it merely wakes them up for a small amount of time.

Oftentimes, people wake up feeling tired due to being interrupted in the middle of deep sleep. Humans go through different sleep cycles which range from very light sleep to deep sleep. These cycles last around 90 minutes, as we drift from light sleep, into deep sleep, and back to light sleep. During lighter sleep people are relaxed but still restless, and as they fall deeper into sleep, the body moves very little and heart rate slows down [2]. Waking up during deep sleep is difficult and can be disorienting [3]. Our goal is to wake a person by finding an optimal time in their sleep trends, which is some period of time in light sleep. If our solution can successfully do this towards the end of a person's sleep cycle, rather than at a set time, we believe this can help people wake up feeling more motivated to start their day.

1.2 Updated Solution

Our solution is a new type of alarm system that can get the user out of bed as well as make them wake up feeling energized. The design is a wearable that goes on a person's wrist, paired with a camera system. The wearable system includes sensors to track sleep patterns which are used to determine the best time to trigger an alarm. It has an LED screen with buttons used to set a preferred wake up time interval. This is paired with the computer vision camera system. Once the alarm goes off, the camera system waits until it can confirm the person is standing in an upright position before suppressing the alarm. This forces the user to get out of bed without turning off the alarm and falling back asleep. The various sensors on the wearable detects the user's sleep cycles to determine whether they are in deep or light sleep. We use this information to wake them up while they are in their lightest sleep during the preset wake up time interval, which is anywhere from 15 minutes to 60 minutes, in 15-minute intervals. Let's say a person wants to wake up by 8 am and be at work by 9. If they set their alarm interval to 30 minutes, the system will wake them up at sometime within 30 minutes prior to 8 am if it finds they are in very light sleep. Although they wake up earlier than the time selected, they will actually feel more refreshed by waking up at the end of a sleep cycle. This further motivates them to get up out of bed and stand up straight when the alarm starts ringing.

1.3 Updated High Level Requirements

1. Able to detect a heart rate between 50 - 120 bpm (0.83 - 2 Hz) +/- 5 bpm with Pulse Oximeter sensor, +/- 3g of force in the x, y, and z dimensions through the accelerometer, as well as recognize sounds within the range of 50-65 dB with the microphone to study user's sleep trend.
2. Able to take the user's set alarm time and wake up interval from the wearable device and find an optimal time to wake up the user based on sleep trends, with exact precision of hour and minute.
3. Able to function for at least 8 hours after the battery is charged fully.

1.4 Updated Visual Aid

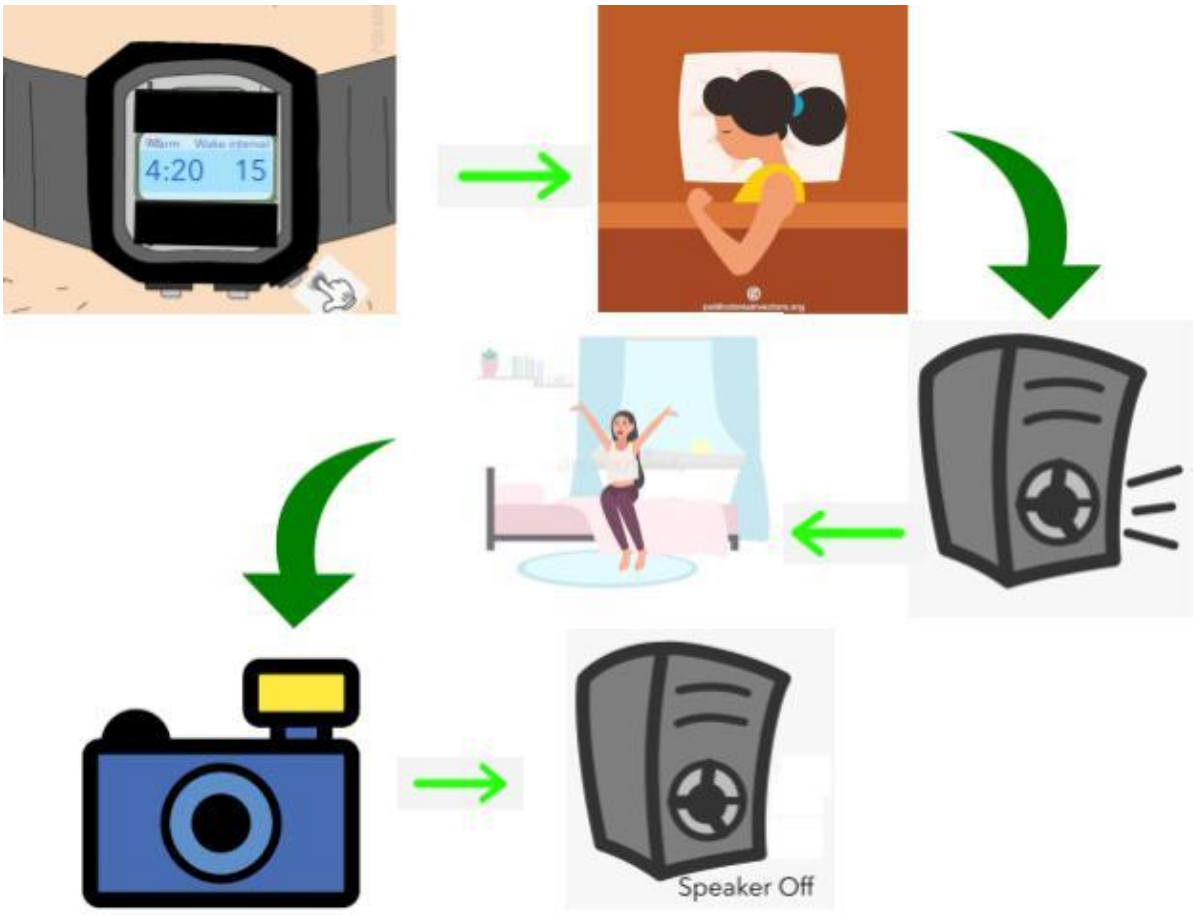

Figure 1: Visual Aid of a typical use case

1.5 Updated Physical Design

Figure 2 shows the physical diagram of the sleep tracking alarm, consisting of the wrist-wearable and computer vision module. It shows an ideal setup of how the system is intended to be used. For example, the computer vision module is about 5 feet away from the bed, so the camera can study the user's body posture. The wearable device should be worn by the user while sleeping so the sensors can analyze movements and track the user's sleep.

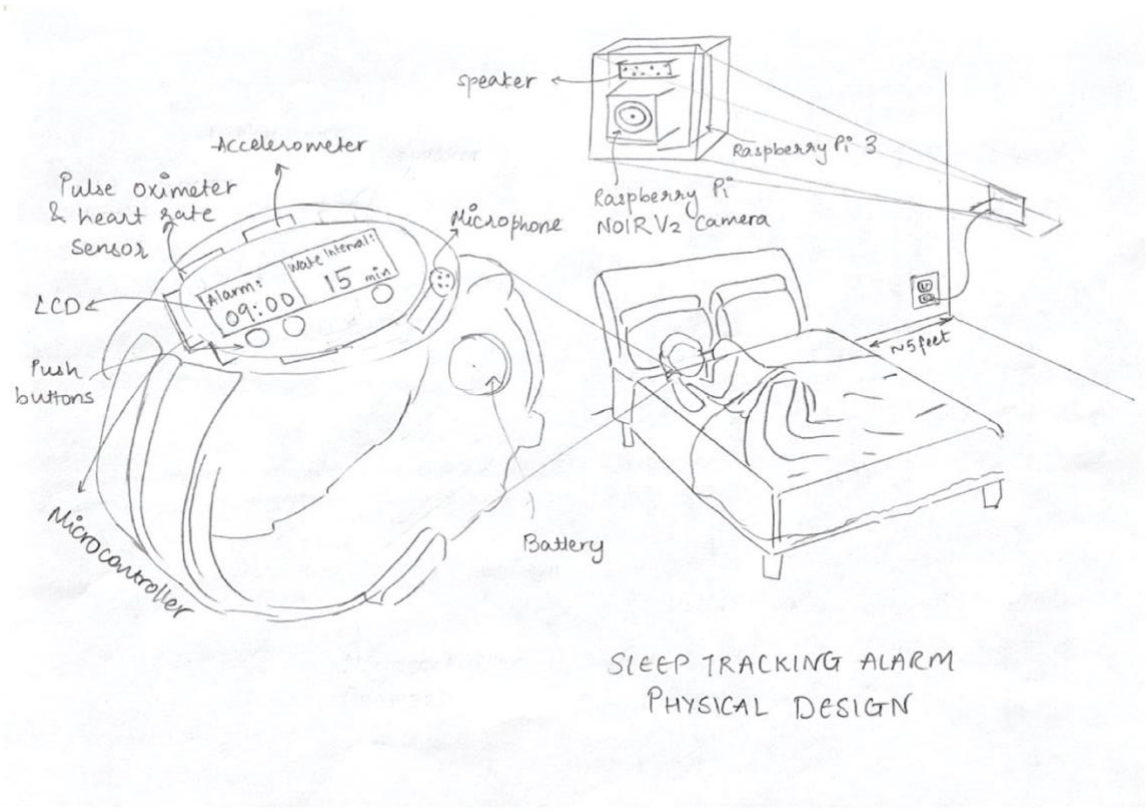

Figure 2: Physical Diagram of the wrist-wearable and camera system set up.

1.6 Updated Block Diagram

Figure 3 shows the block diagram of the entire project. There are two main parts to the design of our system. The most important being the wrist-wearable aspect, consisting of various sensors such as accelerometer, pulse oximeter and heart rate sensor, and microphone, an LCD screen, push buttons, and the microcontroller. Secondly, the computer vision module studies the user's posture to confirm if the user is out of bed and disable the alarm.

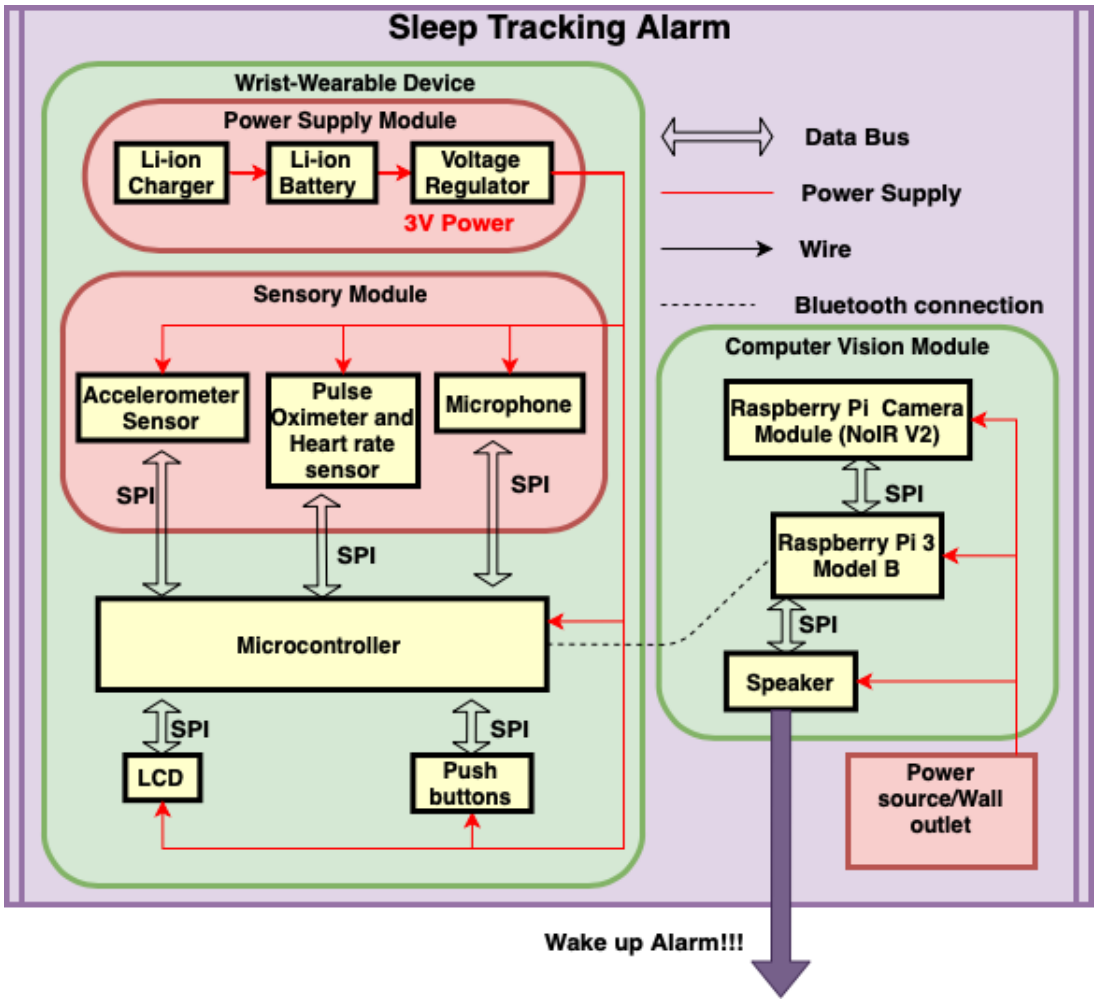

Figure 3: Block Diagram of Sleep Tracking Alarm.

2 Second Project Implementation

2.1 Implementation Details and Analysis

Although we are unable to implement our full project, we have focused on implementing or analyzing multiple parts of our design to have a taste of what our final product would look like. We've tested the computer vision software, made detailed circuit schematics, and analyzed what input values are needed for successful sleep monitoring data.

A large part of our design depends on the functionality of our computer vision module. The computer vision module is used to verify the user is awake and out of bed. As a partial implementation of our project, we looked to test the functionality of the computer vision software we would use to determine if the user is still in bed or out of the bed. We are using OpenPose, a real-time multi-person keypoint detection library for body, face, hands, and foot estimation developed by students at Carnegie Mellon's Robotics Institute [4]. The software allows input from live video from a webcam as well as from images and video files. Our camera would be pointed at the bed and be able to determine the user's posture in real time from captured video clips or images. OpenPose captures information about body part locations and detection confidence and stores this information in an array called pose_keypoints_2d, formatted as x1,y1,c1,x2,y2,c2,.... We would then use this information to determine whether the user is lying down or standing up straight, facing the camera. This information is relayed to the alarm that is going off, which then determines whether to disable it. When we were first deciding the orientation of the camera system, we considered pointing the camera at the wall by the bed and having the user step in front of the system, which is capable of identifying a person in the frame. In the end we went with the camera facing the bed setup, as we can differentiate between when the user is lying in bed and standing up, which provides more accuracy than simply checking if the user is in the frame. As our intended setup discussed in the physical diagram, the laptop camera was facing the user and bed from about 5 feet for this implementation.

2.2 Implementation of Software

For purposes of testing the software, due to the fact we do not have the necessary parts, we have run pose estimation algorithm on a laptop with the webcam pointed towards the bed. We are able to see the software identify a person both laying down and standing up. The location of the different key points can be used to determine the person's posture, which for our usage, is standing up straight versus laying down. The JSON output further below displays the contents of each pose keypoint array, as well as the corresponding part candidates, that break down the array into the specific body parts, which we can use to identify the location of each body part in relation to each other. This is how we would determine if a person is lying down or standing up. If we find that the location of the user's legs or feet are at a similar y-location to their head or chest, we know they are lying down. If their legs and chest have similar x-locations, but the y-location of their head is less (closer to the top) than their legs y-location, then we know that they are standing straight. We can see in figure 8, the standing position, that the location of the user's nose (part candidate 0) is at y-coordinate 57.3, which is close to the top of the screen, which has index (0,0) in the top left corner. We can also see that the user's ankles, part candidates 11 and 14, are at y-coordinate 715.6. The height of the window is 720 pixels, so we know the

ankles are near the floor. It becomes clear the person is standing. This is just a basic example of how we can use the location of each body part to determine whether the user is standing or laying down.

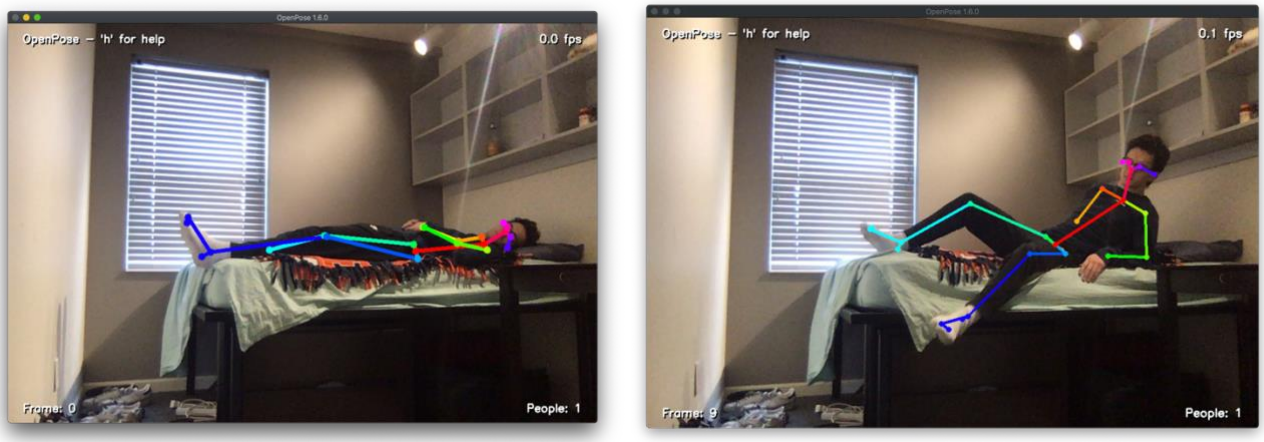

Figure 4: OpenPose pose detection: laying down, getting up

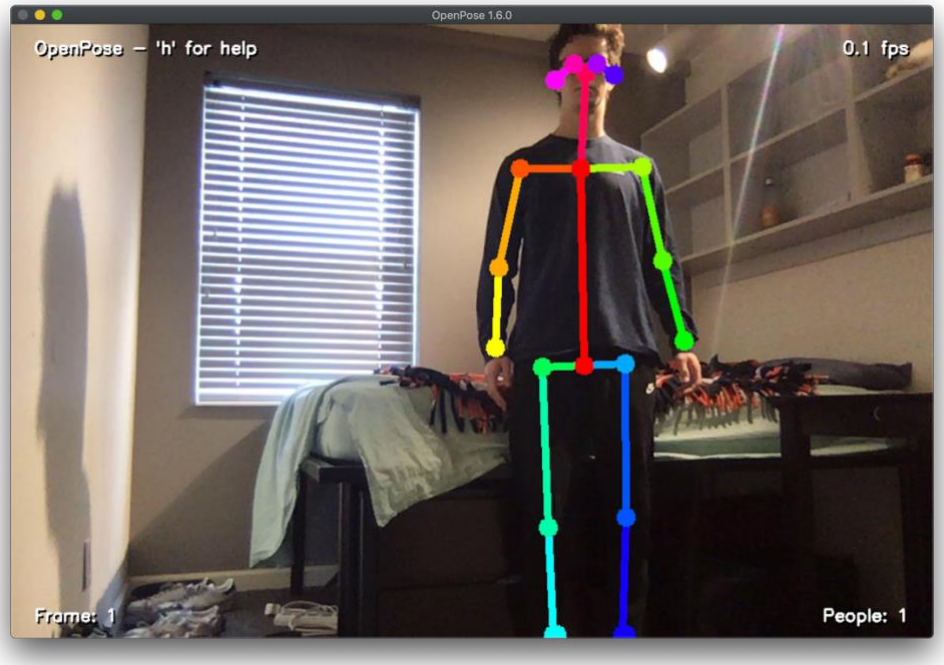

Figure 5: OpenPose pose detection: standing

```
{   //laying position keypoint data output
    "version":1.3,
    "people":[
        {
        "person_id":[-1],
        "pose_keypoints_2d":[854.646,266.972,0.945816,840.942,319.878,0.874294,803.788,300.193,0.827261,758.712,357.102,
        0.125677,0,0,0,880.195,343.321,0.808922,882.122,419.717,0.832936,813.558,417.754,0.857301,721.465,398.258,
        0.66608,703.817,384.498,0.603747,570.661,325.706,0.880336,449.121,394.262,0.739182,739.11,413.872,0.624507,
        676.398,413.829,0.85128,566.704,523.523,0.767522,850.8,253.169,0.927359,870.428,261.065,0.94181,839.031,253.189,
        0.162285,895.823,274.826,0.922173,519.691,537.302,0.600286,533.418,547.086,0.61447,556.877,529.45,0.415452,
        394.297,370.752,0.652306,398.212,368.848,0.580185,443.228,406.02,0.521208],
        "face_keypoints_2d":[],"hand_left_keypoints_2d":[],"hand_right_keypoints_2d":[],
        "pose_keypoints_3d":[],"face_keypoints_3d":[],"hand_left_keypoints_3d":[],"hand_right_keypoints_3d":[]},
        "part_candidates":[
            {
            "0":[854.646,266.972,0.945816],     /* Nose */      "1":[840.942,319.878,0.874294],     /* Neck */
            "2":[803.788,300.193,0.827261],     /* RShoulder */ "3":[758.712,357.102,0.125677],     /* RElbow */
            "4":[],                             /* RElbow */    "5":[880.195,343.321,0.808922],     /* LShoulder */
            "6":[882.122,419.717,0.832936],     /* LElbow */    "7":[813.558,417.754,0.857301],     /* LWrist */
            "8":[721.465,398.258,0.66608],      /* MidHip */    "9":[703.817,384.498,0.603747],     /* RHip */
            "10":[570.661,325.706,0.880336],    /* RKnee */     "11":[449.121,394.262,0.739182],    /* RAnkle */
            "12":[739.11,413.872,0.624507],     /* LHip */      "13":[676.398,413.829,0.85128],     /* LKnee */
            "14":[566.704,523.523,0.767522],    /* LAnkle */    "15":[850.8,253.169,0.927359],      /* REye */
            "16":[870.428,261.065,0.94181],     /* LEye */      "17":[839.031,253.189,0.162285],    /* REar */
            "18":[895.823,274.826,0.922173],    /* LEar */      "19":[382.544,366.889,0.0655173,519.691,537.302,0.600286],
            "20":[533.418,547.086,0.61447], /* LBigToe */       "21":[556.877,529.45,0.415452],     /* LHeel */
            "22":[394.297,370.752,0.652306,507.801,531.468,0.0652216],"23":[398.212,368.848,0.580185],/* RSmallToe */
            "24":[443.228,406.02,0.521208]// RHeel
            }
        ]
}
```


Figure 6: JSON keypoint data output for laying position


```
{   //getting-up position keypoint data output
    "version":1.3,
    "people":[
        {
        "person_id":[-1],
        "pose_keypoints_2d":[911.432,374.72,0.88718,872.304,398.151,0.661723,868.371,384.533,0.507837,825.271,392.321,
        0.115459,760.638,366.825,0.0697596,878.136,407.975,0.5756,823.293,396.268,0.517525,762.653,366.86,0.836497,
        746.927,409.93,0.433076,742.993,398.188,0.324877,580.386,382.496,0.503583,478.527,407.966,0.138737,750.837,
        423.64,0.384474,584.33,384.451,0.127535,370.756,411.876,0.287625,915.438,366.808,0.937302,921.342,384.429,
        0.869487,907.551,359.012,0.199561,915.454,404.055,0.603611,329.604,349.264,0.358757,327.675,358.984,0.376546,
        355.052,419.78,0.273078,0,0,0,0,0,0,0,0,0],
        "face_keypoints_2d":[],"hand_left_keypoints_2d":[],"hand_right_keypoints_2d":[],
        "pose_keypoints_3d":[],"face_keypoints_3d":[],"hand_left_keypoints_3d":[],"hand_right_keypoints_3d":[]}],
        "part_candidates":[
            {
            "0":[911.432,374.72,0.88718],   /* Nose */      "1":[872.304,398.151,0.661723],     /* Neck */
            "2":[868.371,384.533,0.507837], /* RShoulder */ "3":[825.271,392.321,0.115459],     /* RElbow */
            "4":[760.638,366.825,0.0697596],/* RElbow */    "5":[878.136,407.975,0.5756],       /* LShoulder */
            "6":[823.293,396.268,0.517525], /* LElbow */    "7":[762.653,366.86,0.836497],      /* LWrist */
            "8":[746.927,409.93,0.433076],  /* MidHip */    "9":[742.993,398.188,0.324877],     /* RHip */
            "10":[580.386,382.496,0.503583,335.449,458.894,0.0648453],                          /* RKnee */
            "11":[478.527,407.966,0.138737,378.586,409.917,0.407392],                           /* RAnkle */
            "12":[750.837,423.64,0.384474], /* LHip */      "13":[584.33,384.451,0.127535,678.368,445.257,0.141823],
            "14":[370.756,411.876,0.287625,621.547,453.093,0.281605],                           /* LAnkle */
            "15":[915.438,366.808,0.937302],/* REye */
            "16":[921.342,384.429,0.869487],/* LEye */      "17":[907.551,359.012,0.199561],    /* REar */
            "18":[915.454,404.055,0.603611],/* LEar */      "19":[329.604,349.264,0.358757,543.204,439.358,0.114804],
            "20":[327.675,358.984,0.376546,545.167,451.13,0.101544,580.375,453.08,0.0803782],   /* LBigToe */
            "21":[355.052,419.78,0.273078,611.72,451.075,0.136941],                             /* LHeel */
            "22":[327.647,349.251,0.311695],/* RBigToe */   "23":[327.722,353.112,0.256521],    /* RSmallToe */
            "24":[360.953,419.727,0.308045]// RHeel
        }
    ]
}
```


Figure 7: JSON keypoint data output for getting up position

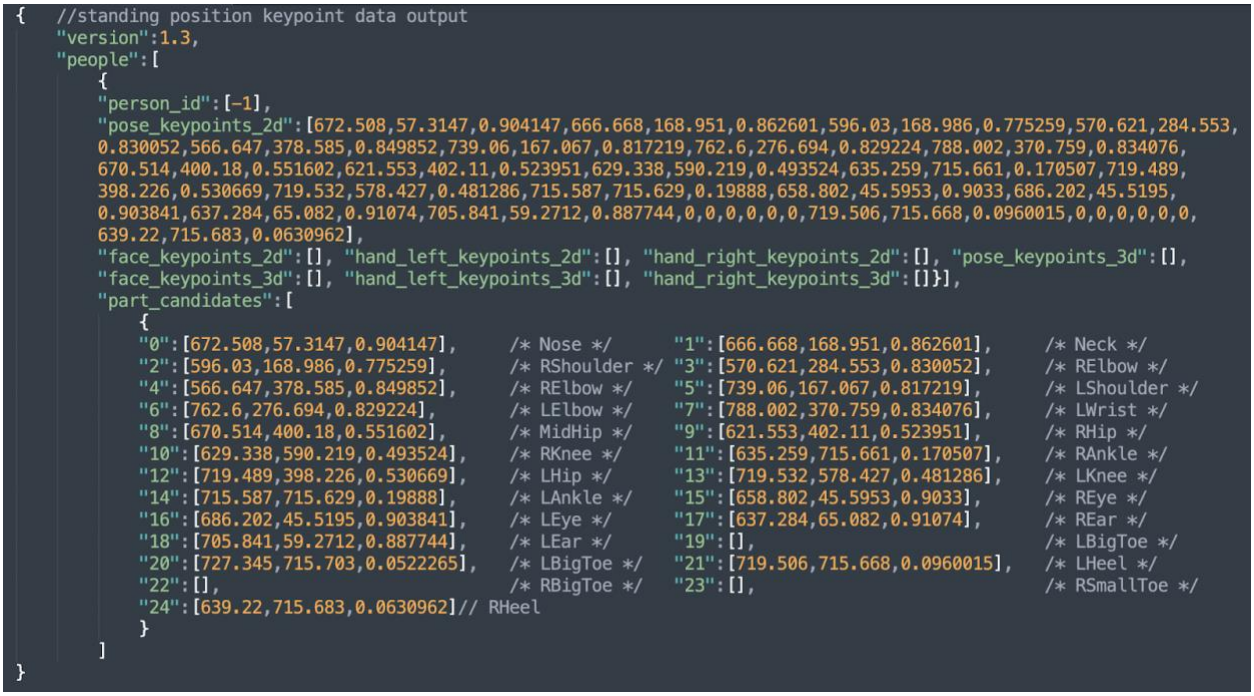

Figure 8: JSON keypoint data output for standing position

When testing the software, we rarely ran into edge cases that would create problems for our pose estimation. There were a few cases in which the algorithm incorrectly identified certain body parts or was not able to recognize parts of the body when the user was moving quickly and a clear picture was unable to be taken, or the person was in a confusing position. This could cause problems if the algorithm incorrectly identifies a body part, making it seem that you are in a different position than you really are. Luckily, along with the x-coordinate and y-coordinate data, OpenPose tracks its confidence of its estimation in the third variable, c, which contains a value from 0 (no confidence) to 1 (100% confident). To work around the edge cases, we can use the information recorded by the confidence variable to make sure you're where you seem to be.

2.3 Circuit Analysis and Schematic

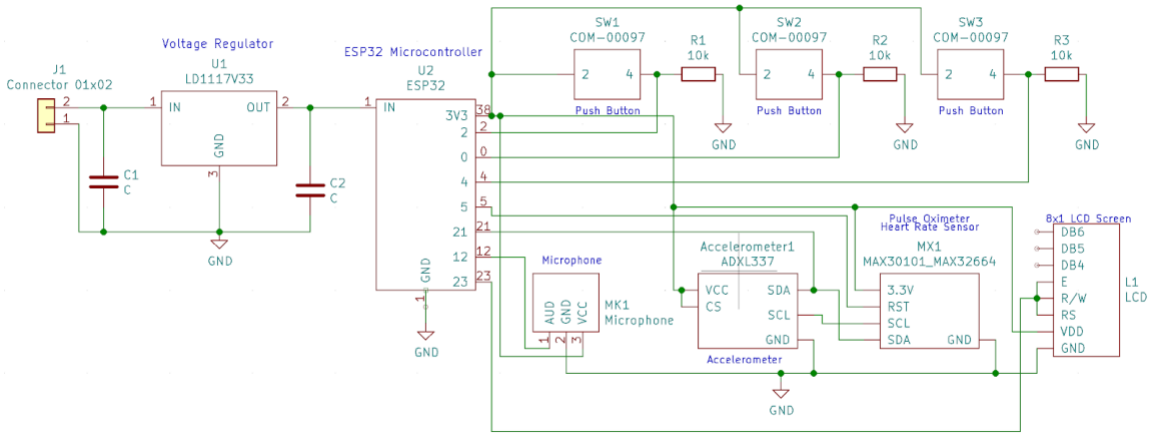

Figure 9: Circuit Diagram of the wearable

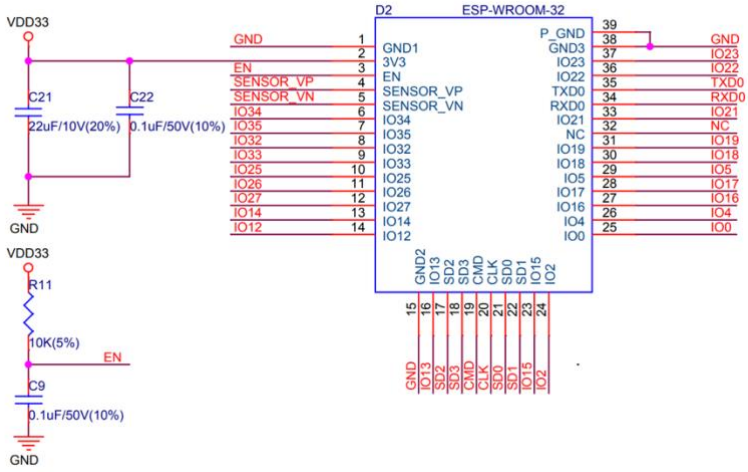

Figure 10: Microcontroller Espressif ESP32 schematic

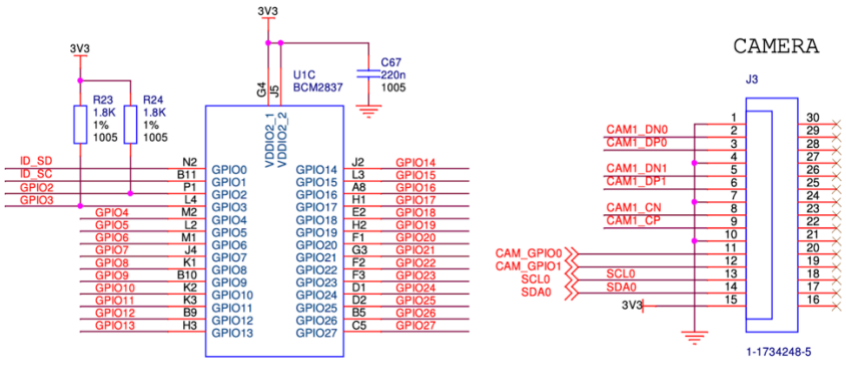

Figure 11: Detailed Raspberry Pi 3 B and Raspberry Pi Camera Module NoIR V2 schematic

2.4 Mathematical Analysis

The portion of our project that poses the biggest risk to completion would be our Wearable Device module. This is the most important piece of the project because this is where all the sensory inputs are at, and it will be the one to determine when to wake the user up from their sleep. It also is the most complex, since we will need to take in data from these various inputs and be able to recognize what stage of sleep the user is in. This will provide a higher level of accuracy than normal sleeping tracking devices, since we have additional inputs than the existing devices. There will be several requirements from this module that must be addressed for it to function properly in the way we need it to: 1) The Pulse Oximeter and Heart Rate Monitor must be able to detect a heart rate between 50 - 120 bpm (0.83 - 2 Hz) +/- 5 bpm, 2) the Accelerometer needs to be able to measure +/- 3g of force in the x, y, and z dimensions, 3) the Microphone needs to pick up sounds within a range of 50-65 dB, 4) the battery must be able supply power to all components for at least 8 hours.

Our first requirement will be taken care of by two chips that are integrated with the Pulse Oximeter: the MAX30101 and MAX32664 chips. The former does all the sensing by using its internal LEDs to bounce light off the skin's arteries and measuring how much light is absorbed from the use of photodetectors. This process is called *photoplethysmography*, which will be

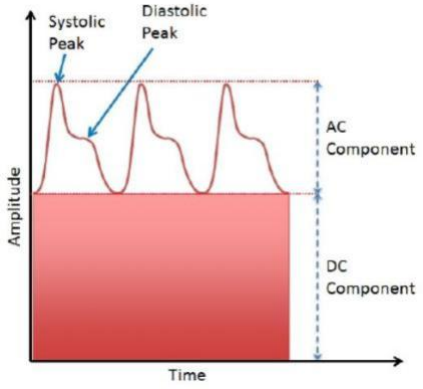

Figure 12. Example photoplethysmograph

explained more down below. The latter of the two chips receives data from the other and applies algorithms to determine both the heart rate and blood oxygen saturation.

As mentioned before, the way in which this particular sensing will work is known as *photoplethysmography*. It creates a photoplethysmograph (PPG) that helps to detect blood volume changes inthe microvascular bed of tissue [6]. The figure above shows what a simple PPG signal would look like and consists of both an AC and DC component. The AC signal is superimposed on top of the DC, and this specifically shows the changes in arterial blood volume, which can then be utilized to determine heart rate [6]. As shown in Figure 8, we want to measure the peak-to-peak interval between two Systolic peaks. Once this value is obtained, we can calculate two different forms of a heart rate: the instantaneous heart rate (HR_{inst}) and the mean heart rate (HR_{med}). For the first equation, $\boldsymbol{t_1}$ represents that peak-to-peak interval time

from before. For our second equation, Q_{nn} corresponds to the number of normal intervals (***NN***) in the time frame of [T_i, T_f] [7]. Here, we would most likely choose to go with determining a mean heart rate, since this can provide us with a more accurate reading and give us a tolerance of +/- 5 bpm.

$$HR_{inst} = \frac{60}{t_1}$$

$$HR_{med} = \frac{1}{Q_{nn}} \sum_{k \in [T_i, T_f]} NN\,[k]$$

Figure 13: Heart Rate Value (HRV) Equations

Our second requirement is that the accelerometer needs to measure +/- 3g of force in x, y, and z dimensions. This is important for when we want to recognize any small vibrations or movements when a person is sleeping. For this reason, we have decided to go with the ADXL337 accelerometer, which is excellent for detecting small movements. It is ideally

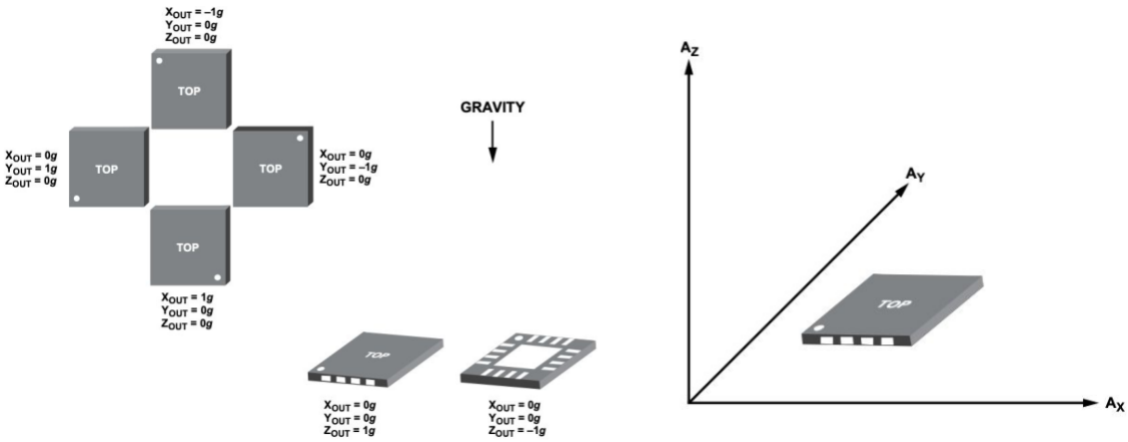

Figure 14: Accelerometer layouts

powered with a voltage supply of 3 V, or it can be powered with a minimum of 1.8 V or a maximum of 3.6 V. There are outputs for each of the axes, and these are measured with an analog-to-digital converter, which correlates to the acceleration in that given axis. After reviewing its specifications, it has been noted that the output is also ratiometric, meaning that its sensitivity varies directly to the supply voltage itself. At a V_S= 3 V, the output sensitivity is approximately 305 mV/g.

In addition to this, capacitors must be placed at the output supply pins in order to implement low-pass filtering for antialiasing and noise reduction. This will help with the accuracy of measuring those small movements. Down below are the potential capacitor choices and the corresponding bandwidth. The selected bandwidth will decide the measurable

Bandwidth (Hz)	Capacitor (µF)
1	4.7
10	0.47
50	0.10
100	0.05
200	0.027
500	0.01

Figure 15. Capacitor selections for corresponding bandwidth

resolutions, or the smallest detectable acceleration, but the output usually has a typical bandwidth of greater than 500 Hz. For this project, a 0.1 µF capacitor should suffice.

The third requirement is that the microphone needs to be able to pick up sound within a range of 50-65 dB, which is typically what snoring falls under. The reason why this is important is that we differentiate our product through the use of a microphone and determine whether a person is snoring or not actually helps to determine their sleep stage. Nonetheless, the one we have chosen to implement is the SparkFun Electret Microphone Breakout. This operates within a frequency range of 100 - 10kHz and has a sensitivity of about -46 +/- 2 dB. This is pretty good for an average speaker, and since it is integrated into the Wearable Module, it should have no issue picking up normal sounds in our desired range.

Lastly, the battery must be able to supply enough power to all the components in the wearable for at least 8 hours after a full charge. This is a big challenge because the battery must fit compactly inside the wearable, which means it needs to be smaller and will result in less capacity. For our design, we were able to go with a 1000 mAh rechargeable battery to power the whole wearable. The component that arguably draws the most power would be the ESP32. When the total functionality of the microcontroller is active, including WiFi, Bluetooth, and radio, it can draw current anywhere from 200-750 mA depending on how much processing it is doing. If we kept it running like this, there would be no way that it can last 8 hours. However, in order to save battery life, we will put the ESP32 into "Modem Sleep" overnight, which will turn off the extra functionality like WiFi and Bluetooth and turn these back on in the morning in order to transmit a signal to the Computer Vision module. This would result in the ESP32 only drawing a current of 3-20 mA. Taking this into account, along with all the other components in the wearable, we can now calculate the expected battery life:

$$\underline{\text{Battery Life} = \text{Battery Capacity (mAh)} / \text{Load Current (mA)}}$$

$$\text{Battery Capacity} = 1000 \text{ mAh}$$

$$\text{Load Current} = 5 \text{ mA} + 12 \text{ mA} + 300 \text{ µA} + 0.5 \text{ mA} + 12 \text{ mA} * 3 + 23 \text{ mA} + 40 \text{ mA} + 600 \text{ µA}$$

$$\text{Battery Life} = 1000 \text{ mAh} / 117.4 \text{ mA} \approx \mathbf{8.52 \text{ hours}}$$

After this calculation, we can see that the battery is expected to last at least 8 hours, which satisfies our third high level requirement.

2.5 Mathematical Modeling (Ground truth values for Wearable Device)

Heart rate values during sleep vary from person to person and can range from normally from 40-100 bpm. Many other factors can affect a person's sleeping habits as well. For the sake of this project, we will use normal and healthy baseline values to look for in determining sleep cycles. Based on an existing sleep tracking application, Oura [5], the most common nighttime heart rate is around 55 bpm. The lowest values during the night can be seen within the range of 35-84 bpm as well [5]. The table below describes the trends we are typically looking for in the different sleep stages.

Sensors Vs Sleep stages	**Stage 1: Light Sleep**	**Stage 2: Normal sleep (similar to light)**	**Stage 3: Deep Sleep**	**Stage 4: REM Sleep (Dream state)**
Accelerometer	Between +/- 3g of force (some movement, maybe turning sides or adjustment)	Between +/- 2.5g of force (some movement, maybe turning sides or adjustment)	Between +/- 0.5 g of force (little to no movement)	Between +/- 2 g of force (mostly no movement, can move due to dreams, however)
Pulse Oximeter	50-90 bpm (mostly in the middle, less than a resting heartbeat)	50-70 bpm (heart rate begins to lower a bit)	40-70 bpm (mostly on lower end)	50-100 bpm (heart rate varies most here because of dreaming)
Microphone	Little to no sound	Little to no sound	Potential sound (snoring)	Little to no sound

Table 1: Sensory readings vs sleep stages

2.6 Bill of Materials

Table 2 shows the estimated cost of components for the project. The project uses 11 components, with a total cost of $141.64.

Part Name	Qty.	Part Description	Cost
Raspberry Pi 3 Model B	1	Computer Vision processing device	$35.00
Raspberry Pi Camera Module NoIR V2	1	Take picture of user's posture	$24.99
Rechargeable Lithium-Ion Battery	1	Power supply for wearable	$9.95
Espressif ESP32	1	Microcontroller	$4.50

SparkFun Triple Axis Accelerometer Breakout - ADXL337	1	Used to sense user's slight body movements	$9.95
SparkFun Electret Microphone Breakout	1	Used to sense user's slight body movements	$6.95
Mini Pushbutton Switches COM-00097	3	Push button used to set the time and interval	$0.35
Newhaven Display NHD-0108HZ-FSW-GBW	1	8x1 LCD display on wearable to display set time and interval	$8.50
SparkFun Pulse Oximeter and Heart Rate Sensor - MAX30101 & MAX32664	1	Pulse Oximeter and Heart Rate sensor used to determine depth of user's sleep	$39.95
Sparkfun Thin Speaker	1	To wake user from sleep	$0.95
Digikey 3.3 V 800 mA Regulator - LD1117-3.3	1	Brings voltage from 5V power source down to 3.3V for ESP32	$0.55
Total:			$141.64

Table 2: Parts Cost

3. Second Project Conclusions

3.1 Implementation Summary

As mentioned in Chapter 2, we were able to test some functionality of our Computer Vision module. Given the current circumstances and no access to the lab and required components, we felt that partial implementation on the software end would still be crucial to this project. This brings heavy significance because the functionality of the project depends on this module to work correctly, represented by the separate Raspberry Pi and a NoIR V2 Camera. This portion determines whether a user has successfully risen and left their bed, which then signals to the alarm that it should be disabled.

Elliot and Rutu both focused on the implementation of the OpenPose library. They came to the decision that with limited time it was best to get this functionality to work on their own devices before trying to test it with a Raspberry Pi. Kishan was able to perform a tolerance analysis of the several input sensors needed in the wearable component. He analyzed the different range of values that needed to be measured by input sensors in order to have accurate sleep monitoring data.

3.2 Unknowns, uncertainties, testing needed

The main parts of the project were not able to be completed due to the lack of access to the design lab and the components required to build the entire project. This includes the hardware for the wearable device such as the pulse oximeter and heart rate sensor, accelerometer, microphone, LCD screen, push buttons, as well as the components from the computer vision module such as Raspberry Pi 3 and camera module. Unfortunately, none of the group members had access to these parts at home, nor the proper equipment, and could not put them together without the tools from the lab. Our plan was to first put all the components together as described in the schematic on the breadboard and test out. Later, was to obtain the PCB and begin to solder the several different sensors and inputs onto the circuit board. This would have made up most of the hardware in that single wearable module. In addition, we really missed out on the machine shop, where they would have been able to help us create a proper casing for the wearable that could house the hardware. The part we worked on during these uncertain times was the computer vision algorithm to work with the Raspberry Pi, in detecting human pose. We used our laptop camera to record and take pictures and ran the OpenPose algorithm for human pose estimation for results. Although we were able to partially implement the Computer Vision portion of the project, it is a different process when it comes to adding this to a physical Raspberry Pi board and the camera. This was going to be the extra feature of the project that really makes "Sleep Tracking Alarm" stand out from other existing products out there.

3.3 Ethics and Safety

Since we are not able to build and create this design, we have listed the ethical and safety measures as a hypothetical situation where the design is being produced.

One ethical concern during the potential development of this project will be to make the wearable both compact and comfortable for the user. To address this, we will need to find the smallest hardware components that we can and try to arrange them in a manner that will keep it from being bulky. There is also a safety concern that comes with a wearable, such as any shocks or electrocutions. To prevent this, we cannot have any possible openings in the hardware that could harm a person while wearing our device. In

addition, there may be a possibility of the wearable heating up while it is touching the skin. For this concern, we will have to make sure that we are supplying the proper power to all components, and that certain components will be limited in power when they are not in high use. These safety measures all fall under the IEEE Code of Ethics #1.

Another potential concern that could arise is maintaining a user's privacy. We record a user's sleep cycle and have a computer vision component that looks to see if the user has moved out of their bed. Both of these will accumulate data in some manner, but we do not retain any information on our end. We will not keep the user's sleeping data over time, as it is only used on a nightly basis and will be overwritten with new data each night. The camera from our Computer Vision module will not be active overnight or during the daytime, and it will include a shutter if the user would like to cover the camera at any point they like. It will only begin detection once it receives a signal from the wearable device that the alarm has been triggered. The user will be informed of these privacy measures, and any images/videos that are taken to detect a person's posture will not be retained either. Both of these cases align with the IEEE Code of Ethics #1 and #9 about protecting the safety and health of users, as well as avoiding malicious practices.

In regard to any potential breaches, we will need to encrypt data stored in both the Wearable Device and Computer Vision modules. The security of our Computer Vision module will be very crucial since we are examining a person's posture by utilizing a camera. We will not have to worry about breaches from the internet, as the Raspberry Pi will communicate to the wearable via bluetooth and its WiFi chip will not be powered. As mentioned before, any images that are taken with the camera will be protected and will not be retained in any other way except to detect if a person is up from their bed. We will dispose of this data each time the alarm has been suppressed, since it will no longer be needed.

3.4 Project Improvements

Considering the time constraints of ECE 445 as well as the scope of the project in this class, there is definitely room for improvement in our design and assembly. Sleep tracking is a gray area right now with fewer sensors to detect sleep trends. Once we experiment and test the consistency and accuracy of the sleep tracking device, there might be a case that we need to add additional sensors to improve accuracy or remove redundant ones to reduce the weight of the wearable device. Initially, we designed the wearable itself to be able to note the user's desired alarm time for the user's simplicity. However, there could be major design changes, such as having a mobile application to note the user's alarm time. Having a mobile application would require a wireless connection such as Bluetooth from the wearable to itself. The advantage of this new design change is that it would eliminate the use of some components from the wearable such as LCD screens and push buttons, making the wearable lighter in weight and more likable to wear. Except for the slightly larger microcontroller added to the wearable, all the sensors are chosen to be small and almost weightless for any wearable device. After trials and experiments wearing the wearable device, we might find it's heavier to practically sleep wearing it. A smaller wearable microcontroller can be chosen to solve that. By these modifications, the wearable would definitely be more comfortable to wear and reliable in terms of accuracy!

References

[1] S. Roy, “57% of Americans Hit the Snooze Button,” Sleep Review, 28-Aug-2014. [Online]. Available: https://www.sleepreviewmag.com/sleep-health/sleep-whole-body/heart/americans-snooze-button-withings/. [Accessed: 04-Apr-2020].

[2] “3 Things You Should Know About REM Sleep,” Sleep Study, Sleep Clinic | Valley Sleep Center | Arizona, 18-Feb-2020. [Online]. Available: https://valleysleepcenter.com/3-things-you-should-know-about-rem-sleep/. [Accessed: 04-Apr-2020].

[3] S. Davis, “How to Wake Up More Easily,” WebMD, 27-Oct-2009. [Online]. Available: https://www.webmd.com/sleep-disorders/features/trouble-waking-up#1. [Accessed: 04-Apr-2020].

[4] Z. Cao, G. Hidalgo Martinez, T. Simon, S. Wei, and Y. A. Sheikh, “OpenPose: Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.

[5] “the most accurate sleep and activity tracker,” Oura Ring. [Online]. Available: https://ouraring.com/heart-rate-while-sleeping. [Accessed: 04-Apr-2020].

[6] J. L. Moraes, M. X. Rocha, G. G. Vasconcelos, J. E. Vasconcelos Filho, V. H. C. de Albuquerque, and A. R. Alexandria, “Advances in Photopletysmography Signal Analysis for Biomedical Applications,” Sensors (Basel, Switzerland), 09-Jun-2018. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6022166/. [Accessed: 18-Apr-2020].

[7] K. Shelley and S. Shelley, Pulse Oximeter Waveform: Photoelectric Plethysmography, in Clinical Monitoring, Carol Lake, R. Hines, and C. Blitt, Eds.: W.B. Saunders Company, 2001, pp. 420-428. [Accessed: 18-Apr-2020].

Appendix A Requirement and Verification Table

Pulse Oximeter/HRM Requirement	Pulse Oximeter/HRM Verification
Be able to detect heart rate between 50 - 120 bpm (0.83 - 2 Hz) +/- 5 bpm	1. Measure resting heart rate manually through pulse and keep this data. 2. Write code in Arduino to obtain output data and print out heart rate and confidence percentage values. 3. Hook up the chip to an Arduino board and place finger on the sensor firmly (can use a rubber band to hold in place) 4. Compare output values to the heart rate measured manually and determine if it successfully read within 50-120 bpm.

Accelerometer Requirement	Accelerometer Verification
Be able to detect changes in forces from +/- 3g in the x, y and z-dimensions.	1. Hook the chip up to a breadboard, supply it with 3.3 V and connect each of the output pins to a DMM one at a time. 2. Begin to move the chip around the air so it can detect a change in gravity. 3. Check the DMM to ensure that it has output values of [1.6 V, 3.3 V] or [-3.3V, -1.6V]

Microphone Requirements	Microphone Verifications
Be able to detect sounds at a range of at least 50-65 dB when supplied with specified voltage of 2.7V-5.5V.	1. Connect microphone to an Arduino board with a supplied 3.3V, and its output to an LED. 2. Produce normal sounds within a 50-65 dB range (such as clapping, knocking, snoring, or normal talking level) and observe if the LED lights up

Microcontroller Requirements	Microcontroller Verifications
Microcontroller must be able to take sensory inputs and trigger alarm at an optimal time with time precision of hour and minute.	1. Connect the microcontroller to all the sensors; accelerometer, pulse oximeter, and microphone. 2. Program the microcontroller to take an average of readings from the ground truth shown in *Table 1 for Sensory inputs vs Sleep stage*. 3. Program the microcontroller to find the next sleep cycle of 90-minute close to wake up time and in wake up interval 4. Ensure manually if the microcontroller logic found the correct optimal time as desired.
Microcontroller must be able to communicate with the Computer Vision Module (precisely Raspberry Pi) using wireless Bluetooth communication protocol in less than 20 seconds.	1. Send any dummy data to the Pi, from microcontroller via Bluetooth 2. Check the data received on the Pi to see if the data matched with sent information. 3. Ensure that the dummy data is verified as expected in the time range.

Push Button Requirement	Push Button Verification
Can be momentarily clicked, incrementing the interval time by 15 minutes when rated up to 50mA current.	1. On a breadboard, connect the push button to an LED circuit, with one pin to specified power supply, second to ground and third to a digital I/O pin of Raspberry Pi (pin 7). 2. Examine from the third pin, while button open (unpressed), the pin is connected to power, so we read High. 3. Similarly, when the button is closed (pressed), the pin is connected to ground, so we read Low. 4. Program microcontroller for a logic where each button press/unpress increments the minute timer by 15 minutes, and resets to 0, reaching 90 minutes.

LCD Display Requirement	LCD Display Verification
Must be able to display numbers precisely when a push button is pressed.	1. Connect your laptop to ESP32 with LCD directly to its GPIO pins. 2. The I2C address should be displayed on the serial monitor. Find the LCD I2C address. 3. With the LCD properly wired to the ESP32, upload the I2C Scanner sketch. 4. Select where to display characters on screen, and simply send a "Hello" for testing.

Computer Vision Requirements	Computer Vision Verifications
Be able to trigger the Raspberry Pi Camera NoIR V2.	1. Connect the Raspberry Pi 3 along with the Camera Module NoIR V2 to any laptop. 2. Confirm that Raspberry Pi 3 triggers the Camera Module V2, after keypress from laptop 3. Confirm the camera takes a picture in any picture formats stored on the laptop.
Must be able to perform 2D Pose Estimation accurately stating the person is sitting, sleeping or standing.	1. Visually confirm if the output from the open source 2D Pose Estimation algorithm accurately judges the person's posture.
Be able to record surroundings when notified by the Raspberry Pi 3 that is good enough quality of an image that it is properly analyzed by the 2D Pose Estimation software.	1. Connect the Raspberry Pi 3 along with the Camera Module NoIR V2 to any laptop using a USB cable. 2. Run the Raspberry Pi script for taking a picture via camera module. 3. Visually confirm the camera takes a picture of good quality stored on the laptop.

Speaker Requirement	Speaker Verification
Be able to output alarm sound when supplied with input voltage of ~1.5V.	1. Hook up the speaker to the Arduino board and add it to a 1.5 V power supply. 2. Add Arduino code to take in an input on the board and have the speaker output a loud sound each time the input is triggered