SleepRight

Team 7

Baran Cinbis, Evan Hashemi, Pilar Luiz

EE459 Spring 2023

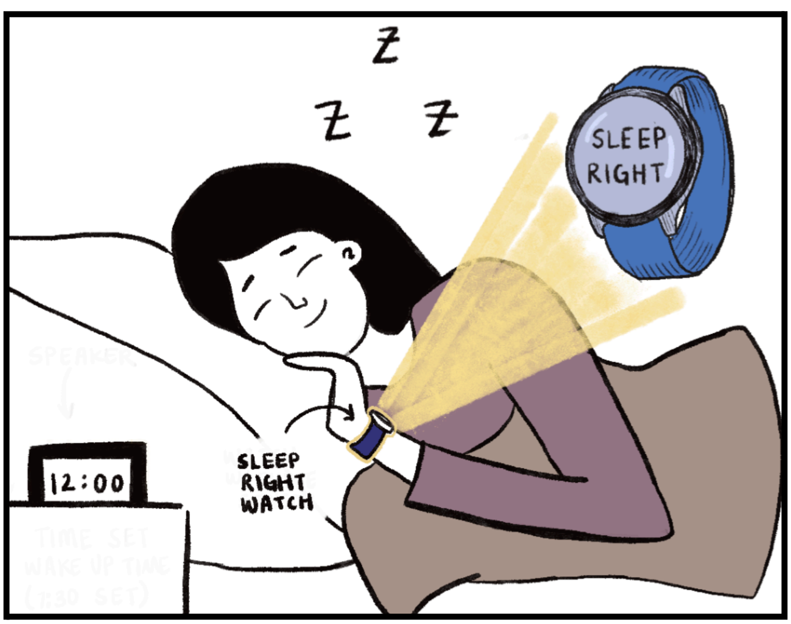
Table of Contents

Problem Statement

Improvements in sleep research have shown us that we sleep in multiple cycles throughout the night. In each sleep cycle, we start in a state of light sleep known as the N1 stage. This is the shortest stage of sleep where the brain slowly begins to decrease neural activity. Next we have the N2 stage which lasts about 4x longer than the N1 stage. In the N2 stage we see a drop in heart rate and periodic spikes in brain activity. Next is the N3 stage of sleep, otherwise known as deep sleep. Here there is very little brain activity, the heart rate drops to its lowest point, and there is little to no movement. Then we move back up through the stages until we reach the final stage, REM sleep. In REM our brain activity is the closest to our waking brain and our heart rate/respiration are at the peak of the cycle. Each of these cycles lasts around 1.5 hours and we go through, on average, 4-6 cycles every night.

So why is this important? Further research has shown that what stage of sleep you wake up in is incredibly important to the quality of your sleep and how much energy you have throughout the day. If you wake up in the N3 stage of sleep, you are more likely to have less energy throughout the day regardless of how long you were asleep. But if you wake up during the REM or N1 stage of sleep you are more likely to feel well rested and have more energy throughout the day. That is why Sleep Right will choose to wake a user up 30 minutes earlier than their alarm. If they are in REM or N1 sleep at that time, it is more beneficial to the user to wake up then, than risk going into deeper sleep and sleeping the extra 30 minutes.

We have also seen that the method in which you are woken up impacts your energy levels. If you are woken up by a loud alarm, it can cause your blood pressure and cortisol levels to spike. This leads to feelings of stress early in the morning which consumes large amounts of energy. Conversely, waking up from a more gentle source will allow the user to wake up in a better state.



Project Overview

Sleep Right is an embedded system designed to optimize the user's sleep by tracking what stage of sleep they are in, dynamically deciding the optimal time to wake them, and using haptics to smoothly wake them up. This will help them maximize the energy they have for the rest of the day. First, Sleep Right has the user input their desired wake time on an LCD and tracks it with a real time clock. Then it uses a pulse rate sensor to identify what stage of sleep and which cycle they are in throughout the night. Then Sleep Right decides when to wake up the user by predicting the closest time that the user will be both in the lightest state of sleep and closest to the desired wake up time. This actual wake up time changes depending on the user's sleep cycle patterns throughout the night and ensures the user is awake by their set time.

For example, let's say a user decides they want to be awake by 8 am. Sleep Right will begin tracking their biometrics and start tracking their sleep data throughout the night. As this happens Sleep Right could identify, based on the user's previous cycles and current biometric data, that the user will be at their lightest state of sleep at 7:30. Sleep right will then trigger haptics at 7:15 to slowly wake the user up by 7:30. This ensures that they wake up in the most wakeful state of their sleep cycle to minimize the grogginess the user feels in the morning and maximize the energy they feel throughout the day. Haptics were chosen instead of a light or speaker to ensure the user feels like they are naturally waking up.

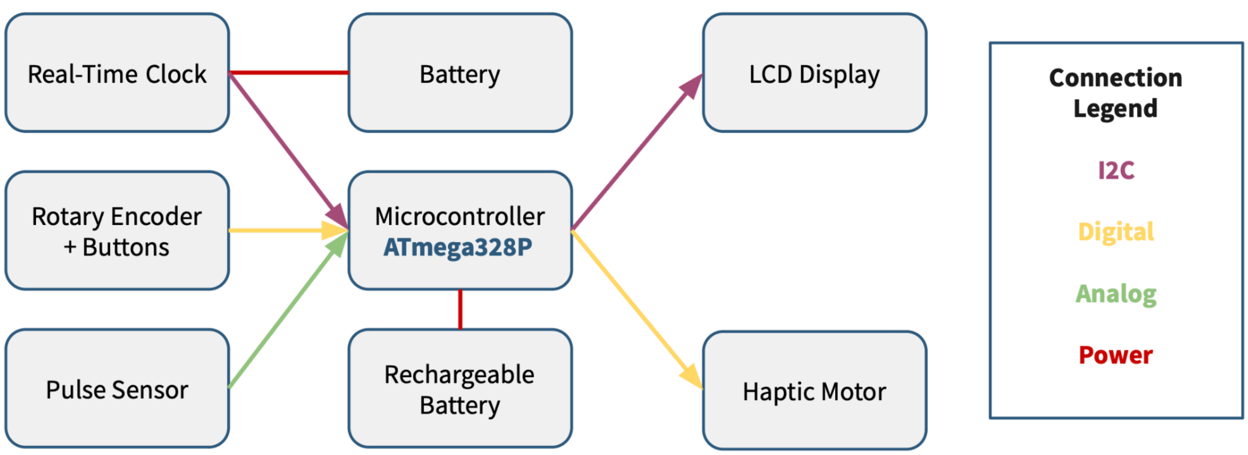


Figure 1: Block Diagram of System Design. Input/output indicated by direction of arrows.

Sleep Algorithm

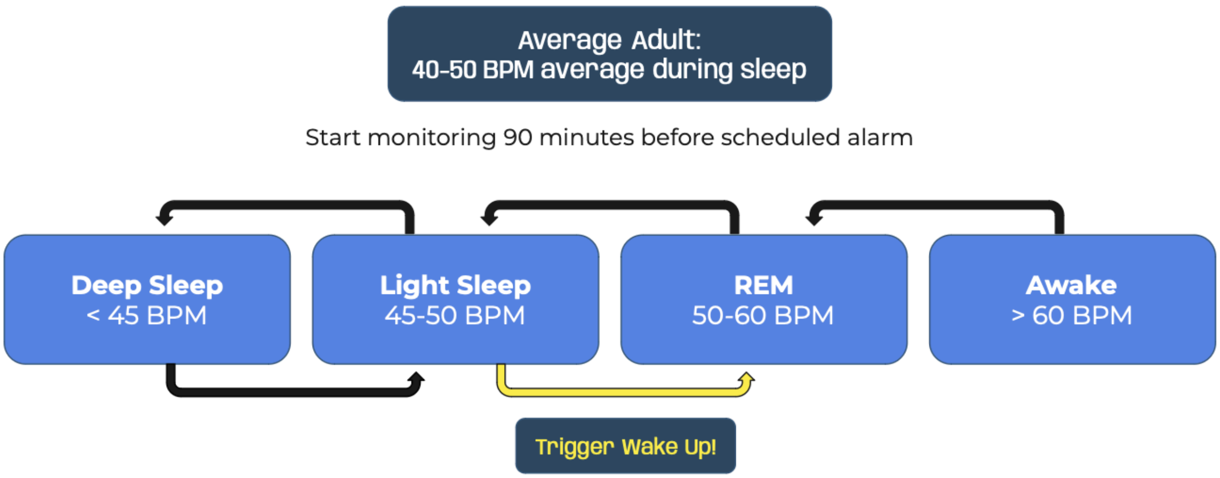


Figure 2: Sleep algorithm flow chart

In order to predict the optimal time for the user to wake up, we use a pulse sensor to measure the heart rate of the user. This is only done when the user is within one sleep cycle, or 1.5 hours, of their wakeup time, in order to save power and avoid taking unnecessary measurements. To extract the heart rate, the pulse sensor has a threshold analog value that it uses to assess whether it has actually detected a heart beat, or just picked up some noisy data. It then averages heartbeat data over a period of time to calculate a heart rate. Since timing is crucial to calculate this, interrupts are used to periodically take measurements.

Our algorithm then averages data to predict the user's stage of sleep. As mentioned in earlier sections, research has determined that the optimal time to wake up is when transitioning from light sleep to REM sleep. Therefore, when the user is in light sleep and then goes to REM sleep, we wait and take additional measurements to ensure that this transition has taken place. If the data is valid, the user is prompted to an "early wakeup" before their scheduled wakeup time, which should improve their quality of sleep.

Proposed Design

Our product's goal is to be a wearable device, thus we aimed for a sleek and comfortable design for users' wrists. In designing the physical prototype, our team researched data relating to comfortable materials that are flexible, water resistant, and pose low risk for allergic reactions. Our team also researched wrist sizes for all age groups and genders to ensure a universally comfortable design. Our team decided on a fabric strap with velcro would be the best design for a future prototype.

Our final design would move all components from a protoboard to a small PCB that is about 1.5 inches across. We believe that it would be realistic to accomplish a few minor changes to the system design. This would include replacing the 5V AA batteries with a smaller coin cell battery. This could be done with a 3V 2000 mAh non-rechargeable battery, although lower capacity rechargeable batteries. In the case of the non-rechargeable battery, there would be 77 hours of use before needing to replace the battery. A boost converter can be used to bring its voltage up to 5V. The serial interface (DB-9 and MAX232 IC) could also be removed since it's only use during the product's development was to debug our software. Finally, the large LCD used for prototyping would be replaced with a smaller, more power efficient LCD (see: appendix). With these simple modifications, all large footprint parts would be removed, and our design could be encased in a compact watch form factor.

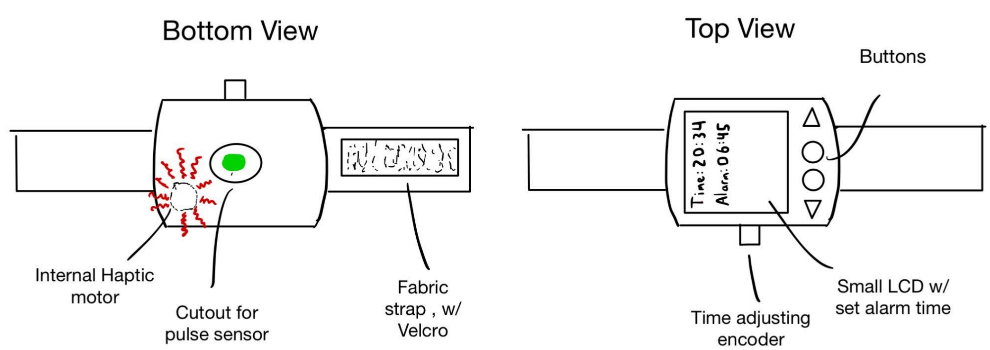


Figure 3: Proposed Product Design

Project Breakdown

Microcontroller

We used the Microchip ATmega328P Microcontroller for our project. It has a serial communications interface, I^{2}C interface, and SPI interface. It has an internal 10-bit analog-to-digital converter for converting analog voltages to binary values. The ATmega328P has 28 pins. 13 are digital pins and 5 are analog pins. It utilizes a 5 volt power supply.

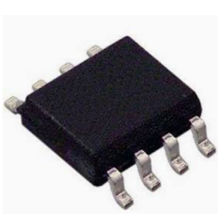
Our project utilizes I^{2}C protocol to communicate with the LCD, serial communication for debugging, and analog-to-digital conversion for the pulse sensor. It provided enough pins to connect all our components. For these reasons, the ATMega238P was the ideal microcontroller for our project.

Inputs

Pulse Sensor (Analog input)

Our project utilizes an analog Adafruit 1093 Pulse Sensor, to acquire our BPM data from the user. It works by constantly sending out green light with a flashing diode and measuring how much light is returned with a light sensor. The sensor then converts the amount of light received into an analog value representing how much blood is present. When this value reaches a certain threshold, set by the initialization code for the sensor, it triggers a hardware interrupt letting us know a heartbeat was detected. From there we can measure the time in between heartbeats and get an instantaneous bpm for our user.

Real Time Clock (I^{2}C)



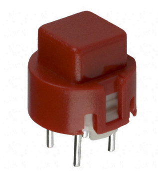
We used the DS1307 RTC Chip for our product to keep track of time. Standard I^{2}C protocols were used to communicate with the clock. This initially consists of first sending the clock the user's desired start time. From here the clock uses its own internal interrupt created by suppling current to a crystal to keep track of the passing of each second. In other words, each interrupt generated by the crystal represents a second of real time passing. From this, the clock is able to continue counting the passage of time on its own without the assistance of the IC chip. This makes time keeping consistent and not memory exhaustive.

In order to read back the time, we only have to send a read request over I^{2}C. Then the clock would send the corresponding seconds, tens of seconds, minutes, tens of minutes, hours, and tens of hours for its current time. The time can then be computed by combining all of these values.

Rotary Encoder (Digital input)

The rotary encoder is pretty standard issue. One leg is connected to ground, and the other two are connected to pins of the IC with pullup resistors. Then as it is turned the pins values are slowly changed (one at a time) from 1 to 0 and 0 to 1 etc. Each time this occurs, an interrupt is triggered which we interpret as a change to the value of either the minutes or hours of the clock or alarm time. This change is then updated on the LCD.

Buttons (Digital input)



The buttons are also very standard issue. The work by having one end attached to a pin on the IC with a pullup resistor attached. The other end of the button is connected to ground. This means that when the button is pushed the high signal we currently see dissipates and we see a low one signifying the button was pressed.

In our program we used the buttons to select what value we were changing; either the clock's hours or minutes, or the alarm's hours or minutes. We thought having two buttons to select between hours and minutes and then one to select between the alarm and clock time made editing efficient and easy. We also included one more button that functioned as an enter key to be pressed by the user when they are done selecting their times.

Outputs

Haptic Motor (Digital Output)

Our project uses a haptic motor to wake the user up. Once (1) the current time equals the alarm wakeup time or (2) the user transitions from light sleep to REM sleep within 90 minutes of their wakeup time, the haptic motor is activated and remains on until the user presses a button to deactivate it or until 10 seconds have passed.

We chose to use a haptic motor as opposed to a buzzer for an alarm sound because the haptic motor would provide a more natural and less jarring wakeup. Research has shown that using an alarm clock to wake up causes a spike in heart rate which can be detrimental in the long run. This problem is worsened when people snooze their alarms and awake to alarm sounds repeatedly. Therefore, we avoided this problem and opted for a gentler alarm through the haptic motor.

We activate the haptic motor through a digital signal. We considered using pulse width modulation to control the strength of the vibration with the idea of starting with a gentle vibration and increasing its strength as time goes on. However, due to a shortage of time we decided to omit implementing this.

LCD (I^2C)

The LCD is the main component of the user interface. Throughout the states of our project, the LCD displays the current time, alarm time, current heart rate in BPMs, current sleep stage, wakeup message, general user instructions, and optional debug messages.

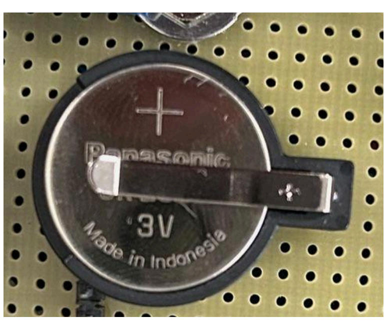
We chose this LCD because it is I^2C capable, which means we save a large number of pins by using it. The downside is that it consumes a lot of power which is a challenge when running using batteries. In the future, we would change to a smaller, less power hungry display.

Power

5V – 4 AA Batteries

We used 4 AA Batteries to create a 5V power supply for our design. This was to show that we could power the Sleepright with batteries and that we were not tied to the digital power supply. This addition helped make the proof of concept for our product much more believable as a wearable device. It made the entire product portable and not stuck to any one place

3V – Lithium Coin Cell



We also used a 3v lithium coin battery as a backup power source for our real time clock. This was to ensure that our product maintained an accurate time even if there was a problem with our main power supply. This battery would continue to power the real time clock so that when the issue with the main supply was fixed, the system could resume its function with the proper time still in place.

Communications

I^2C

The LCD communicates with the microcontroller through the I^2C protocol. We considered using an LCD that communicated with the microcontroller digitally but decided against it because of a digital pin shortage. Thus, we chose this LCD because it communicated to the ATmega328P via I^2C protocol. The real time clock also communicates with the microcontroller via I^2C protocol.

The greatest advantage of I^2C is that the same data line is used to communicate with multiple devices. Our project only uses two pins to communicate with the LCD and real time clock. This helped leave pins open for other components which was an important consideration for our project.

Serial

We used serial communication protocol for debug messages and for testing before the LCD screen was implemented. It was optimal to use serial messages for longer debug statements that wouldn't have fit on the LCD screen. We did this using the MAX232 IC connected to a DB9 connector.

User Operation

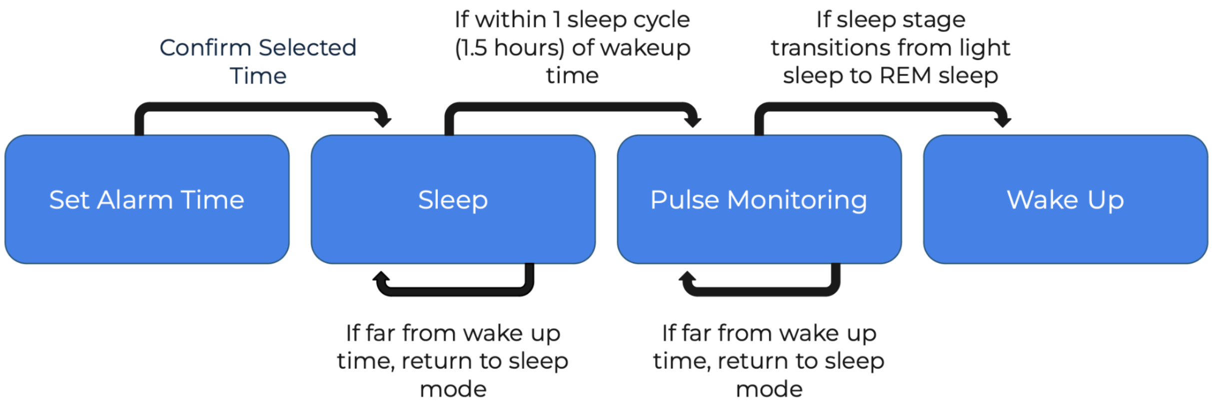
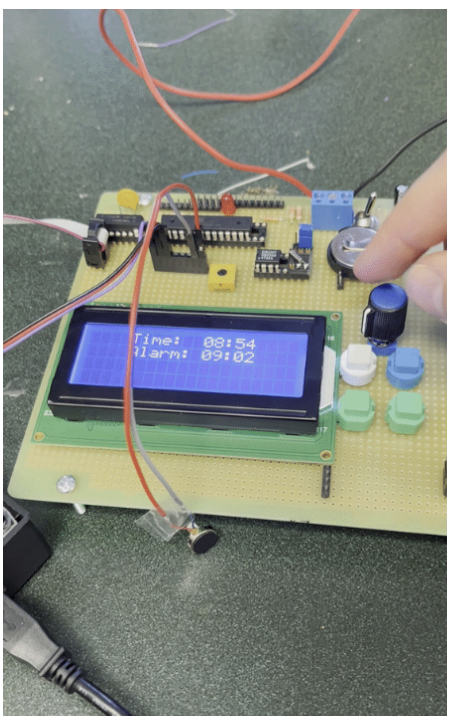


Figure 4: Typical Operation Flow Chart

State 1: Set Desired Alarm Time

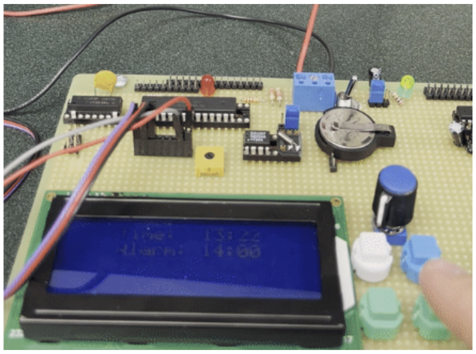


First, the user must set their desired alarm time. The white and blue buttons allow them to toggle between hours and minutes, respectively. Rotating the knob allows the user to pick a time in the selected category (hours or minutes). The last set alarm time is automatically stored in the project's memory, so this is only necessary if the user needs to change their wakeup time from the previous time.

The user can toggle between setting the alarm time and the current time by pressing the bottom left green button. This allows the user to set the current time if it is off for whatever reason. This should not happen since the real time clock operates continuously but is an added layer of redundancy.

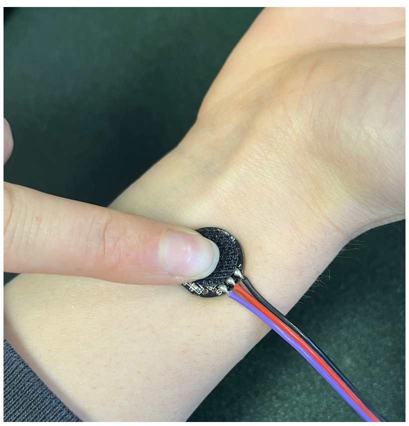
Once the user has set their times, they can press the bottom right green button to confirm their entries. This puts the project into sleep mode.

State 2: Sleep Mode



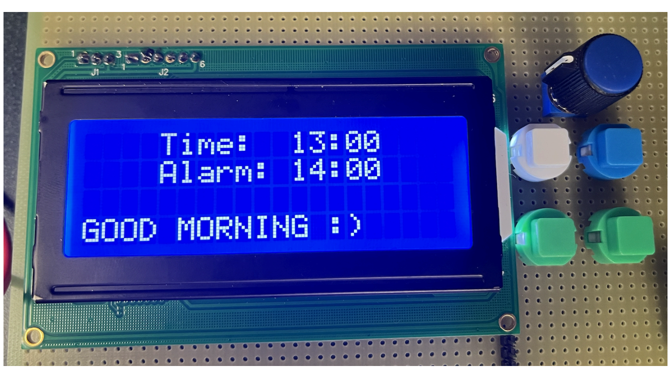
Once the user sets the current time and desired wakeup time, SleepRight goes into low power mode. The LCD screen is turned off and only one interrupt is left on. The single interrupt periodically checks if the user is within 90 minutes of their desired wakeup time. If so, SleepRight exits the low power stage by turning on the LCD, enabling interrupts, and transitioning into the pulse monitoring stage.

State 3: Pulse Monitoring



While the user is sleeping, they should strap the pulse sensor onto the bottom of their wrist. In the finished design with the electronics encased in a watch form factor, this will just require them to wear the watch on their wrist. However, for prototyping purposes the sensor should be placed on the wrist as shown to the left. The haptic motor is also small and can be strapped anywhere along the user's arm.

State 4: Wake Up



Once the wakeup message is displayed and the haptic motor begins vibrating, the user can press any of the four buttons to disable the alarm. This returns the product to its initial state and allows them to set their alarm time for the next day. When the haptic motor begins vibrating and the display shows "Good Morning" the user can press any of the four buttons to snooze the alarm. After this, the product returns to its initial time setting state.

Engineering Standards

To ensure the greatest reliability and safety when using the SleepRight device, we adhered to professional engineering standards implemented for user health data and battery usage. Our project records and analyzes sensitive user data (heart rates) which classifies as protected medical information. The Health Insurance and Accountability Act (HIPAA) sets standards for the collection, privacy, and security of healthcare companies and workers that handle protected health information (PHI). This federal law also requires written authorization from patients before using or disclosing their PHI. To comply with these standards, our data is collected anonymously with no identifiable information tied to the user, such as name or age. The information we do collect is completely transparent and deleted after the program is run. In other words, user information is not currently stored in a database that could be breached, thus there's little risk for information leaking. If we chose to store health data long term for a user, we would ensure user consent before collecting data.

Along these lines, we considered the engineering standards set by children using our product. Per the Children's Online Privacy Protection Act (COPPA), companies are required to obtain viable parental consent before collecting, using, or disclosing personal information, which includes health data, from children under the age of 13. SleepRight fully complies with engineering standards set for minors to ensure the utmost health and safety of children.

When not connected to a direct power source, Sleep Right is powered by AA batteries and the RTC is powered by a coin battery. To ensure the greatest safety and limit the risk of fire or other hazards, we ensured that our batteries complied with the International Electrotechnical Commission's standards specifying the dimensions, discharge characteristics, and marking requirements for AA batteries. We also ensured that our batteries were new and not leaking or unsafe for our product. In the future, we would create a design that fully encases the batteries to limit their exposure to moisture, extreme temperatures, or other hazardous environments.

Finally, we used standard IEEE conventions for I^2C communications and applied them to our LCD panel and RTC. We verified proper communications were taking place using an oscilloscope configured to capture and translate I^2C data. We also ensured that there were no errors in sending I^2C data by making use of error codes that indicate if an unsuccessful transmission is sent or received.

Signature Sheet

	Baran Cinbis	Evan Hashemi	Pilar Luiz
System Design	33%	33%	33%
Component Selection	33%	33%	33%
Software Design	40%	20%	40%
Hardware design	40%	40%	20%
Proposal/DDR/Final Report	40%	30%	30%

May 8, 2023

Baran Cinbis **Date**

May 8, 2023

Evan Hashemi **Date**

May 8, 2023

Pilar Luiz **Date**

Appendix

Parts List

Part	Quantity	Unit Cost
ATMega 328P IC	1	$2.86
Adafruit 1093 Pulse Sensor	1	$25.00
Adafruit 1201 Haptic Motor	1	$1.95
63P202 Potentiometer	1	$1.00
D6C40 F1 LFS Push Button	4	$1.08
Bourns 3315Y-001-016L Rotary Encoder	1	$4.01
Jameco 290125 Rotary Encoder Shaft	1	$3.29
NHD-0420D3Z-NSW-BBW-V3 LCD Screen	1	$27.58
Panasonic CR 2032 3V Battery	1	$1.15
Digikey 103K Coin Cell Battery Holder	1	$1.50
Amazon NiMH 5V Rechargeable AA Battery	4	$1.88
Digikey 36-2477 AA Battery Holder	1	$1.87
DS1307 RTC Chip	1	$5.51

Total Cost Per Unit: $87.56

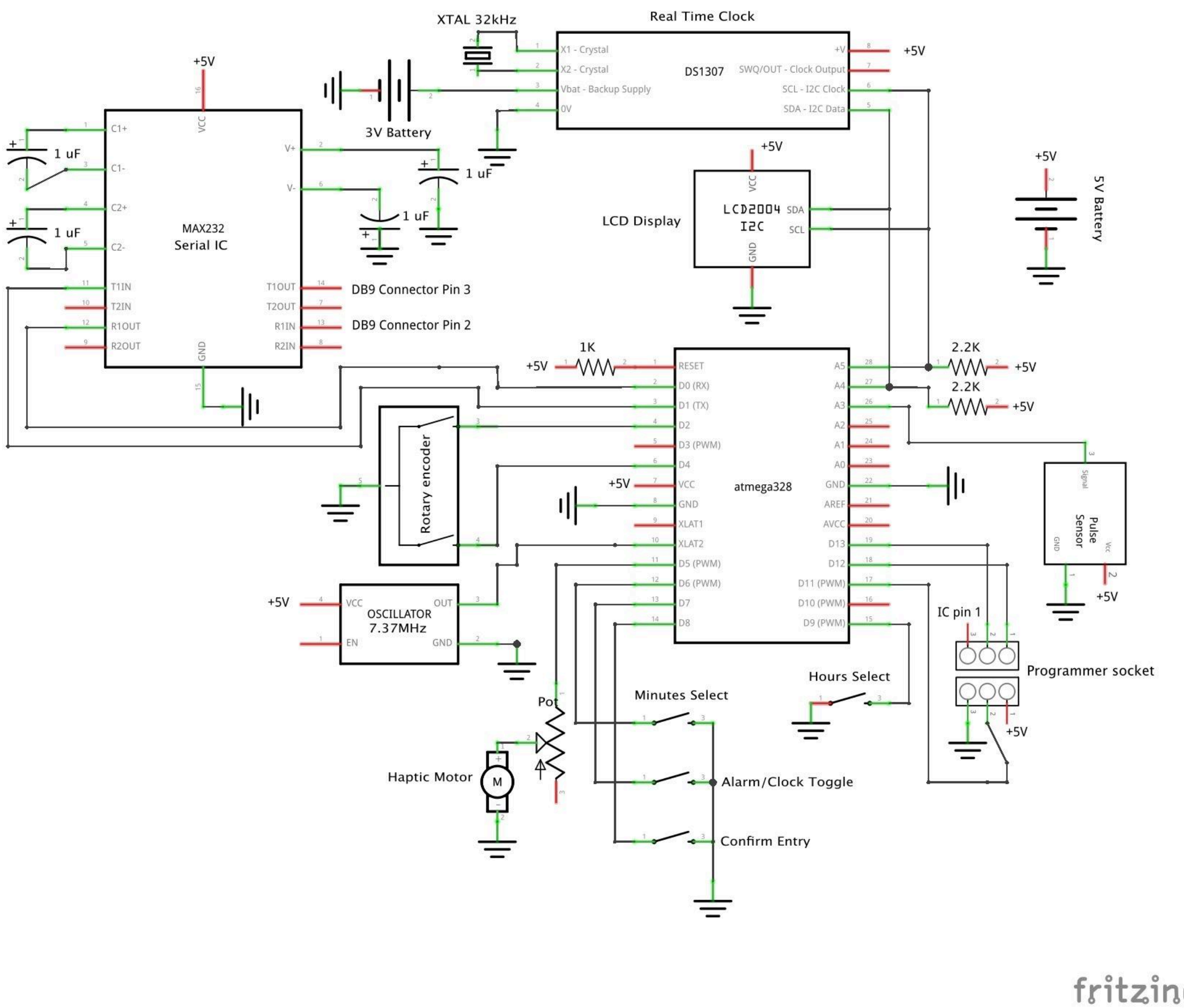
Reduced Form Factor Part Modifications

To fit our design into a more compact form factor, the AA Battery Holder, 5V AA Batteries, and 3V coin cell battery would be replaced with a single 3V coin cell battery and a boost converter. Additionally, our 4x20 LCD would be replaced with a smaller panel. These additional parts are listed below:

Part	**Quantity**	**Unit Cost**
Add CR 3677X 3V Coin Cell battery	1	+ $6.57
Add LP3985IM5X Boost Converter	1	+ $0.62
Add GC9A01 LCD Screen	1	+ $8.90
Remove NHD-0420D3Z-NSW-BBW-V3 LCD Screen	1	- $27.58
Remove Digikey 36-2477 AA Battery Holder	1	- $1.87
Remove Amazon NiMH 5V Rechargeable AA Battery	4	- $1.88
Remove Panasonic CR 2032 3V Battery	1	- $1.15

Cost Change : Decrease of $22.03
New Total Cost: $65.53

Schematic



Source Code:

To view our source code, check out https://github.com/pilarluiz/sleep-right