A CHEAPER ALTERNATIVE FOR TEMPERATURE CONTROLLED SLEEP

By

Alex Dicheva

Patrick Wang

Wyatt Sass

Final Paper for ECE 445, Senior Design, Spring 2024

TA: Luoyan Li

1 May 2024

Project No. 12

Abstract

Sleep quality significantly impacts overall health, yet many struggle with achieving optimal sleeping conditions, particularly with respect to body temperature. Existing temperature-controlled sleep solutions typically come with high costs or energy demands, making them unattainable for the average consumer. This paper details the development of a smart heated blanket meant to act as a cost-effective alternative to these existing solutions. Our project integrates temperature and motion sensors to dynamically adjust heating levels in response to real-time data. The blanket's design aims to produce an ideal sleep environment tailored to each individual user, thereby improving their sleep quality.

Contents

1. Introduction

1.1 Problem and Solution

Extensive research has been done on the impact temperature has on your sleep. Due to a decrease in the body's ability to thermoregulate, body temperatures decline throughout the sleep cycle. To allow for a faster decline in body temperature and to ensure relaxation, bodies prefer warmth before trying to fall asleep. Around 10 min before falling asleep, the body starts to decline in temperature; therefore, cooler temperatures allow the body to stay in sleeping mode, which drives bedroom temperature recommendations to be around 66-70 degrees [1]. Various efforts have been made toward temperature regulation while we sleep to improve the quality and length of sleep. One solution is a Smart Thermostat; however, this requires heating the entire room or the entire home. There are also temperature-controlled bed sheets or duvets, such as the BedJet ($1329), Smartduvet ($1555), and EightSleep ($2195) [2, 3, 4], but all of these are priced at over $1000, making them unaffordable for the average consumer to even consider.

To achieve a cheaper alternative to luxury temperature-controlled sleep, we created a smart and interactive heated blanket. The blanket incorporates temperature sensors to track the blanket temperature within its environment. It also uses accelerometers to detect user movement, giving a reasonable guess as to whether the user is asleep or awake. The user can use a simple yet powerful iOS application to control the temperature of the blanket. They can schedule temperatures and address each of the blanket's four heating zones independently; this is a feature ideal for couples or those who have different preferences for their upper and lower body temperatures. They can also see all the data collected on the blanket to learn about how their temperature affects their sleep throughout the night. Ultimately, this product serves as an attainable yet high-quality option for consumers who want to optimize their sleep and improve their everyday lives with a single purchase.

1.2 Block Diagram and Descriptions

All figures for this chapter will be in section 1.4. As shown in Figure 1.2, our product is split into three distinct subsystems: the Control System, Power System, and Heating System. Each subsystem has its own subcomponents that contribute to its success. Starting with the Control System, there is a microcontroller (MCU) board with a Bluetooth transceiver on board, temperature and motion sensors (IMUs) on separate boards, and a user input device in the form of an iOS application. Overall, the Control System is responsible for gathering data and using it to determine at what level the Heating System should be heating the blanket itself. Next, the Power System uses a typical wall outlet supplying 120 V AC to the power distribution board, which powers the Control System at 3.3 V DC and Heating System at 120 V AC. Finally, the Heating System's temperature controller takes in a PWM input from the Control System and supplies the heated blanket's heating wires with the appropriate level of power to achieve the desired temperature. These three distinct subsystems work together to achieve the high-level functionality of the project as a whole.

1.3 High-Level Requirements

1. The user will be able to set the temperature remotely through an app, and the blanket will be able to control the physical temperature in response. They will be able to schedule a temperature, and the blanket will reach that temperature within 45 minutes. While in use, the heating coils will reach the preset temperature (+/- 3 degrees Celsius), and that temperature will be presented on the app. The blanket will be able to maintain that temperature for over 10 minutes and adjust to the next target temperature under the same requirements.
2. The blanket can detect the user's movements with acceleration greater than 10 m/s^2 (+/- 1 m/s^2). This data will be relayed to our app, and the controller can use this data to detect a "sleeping" state after a period of non-movement.
3. The temperature reading of the blanket should be broken down into 4 distinct zones. The blanket should be able to determine the absolute temperature within each zone (+/- 1.5 degrees Celsius) and should be accurate within a 2-inch radius. This temperature will be displayed on the app.

1.4 Figures

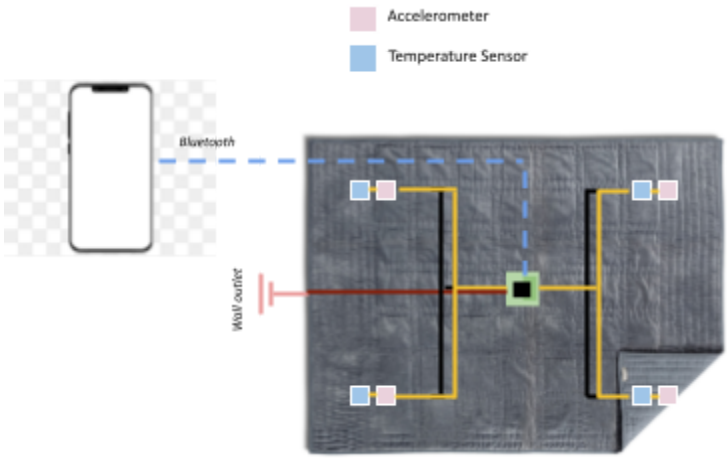


Fig. 1.1. Diagram components adapted from [5, 6]

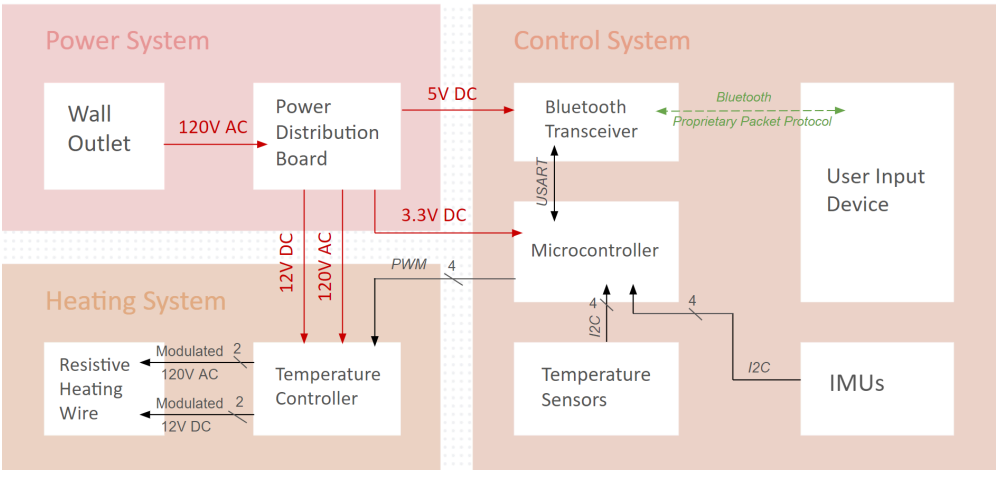


Fig. 1.2. Block diagram of overall blanket design.

2 Design

2.1 Design Procedure

2.1.1 Power System

The main objective was to have a voltage supplied to each of our components that would allow them to operate safely for extended periods of time. Given that the ideal heated blanket is being used by a sleeping user, likely for an entire night, having the reliability and stability of the individual components is a really important consideration for our particular case. The considerations varied depending on the particular components.

The least sensitive component was the heating wire itself. We wanted to run this element at 120VAC, as is the standard for commercially available heated blankets [7]. For all intents and purposes, the heating wire is well modeled as a simple resistive element. As such, all we needed to do in our design was to have direct connections from the wall outlet (120VAC) to each of the heating wires in all four zones of our blanket. We believed this was the most straightforward design, and didn't feel the need to overcomplicate it.

We also needed to provide +12VDC and -12VDC rails to an op-amp circuit on each of our heating PCBs. The exact circuit and reasoning will be discussed in further detail later on, but the general idea was that we needed to use an op-amp to scale up a 3.3V signal to a 10V signal. As such, we knew that we needed to provide the +12VDC and -12VDC rails to four separate PCBs, but we weren't concerned about the stability of these supplied voltages. So long as our positive rail was above +10VDC, we felt that it would be fine for our purposes. As such, we simply used two buck converters to translate a 120VAC signal into a +/- 12VDC signal. We took a similar approach with the voltage supplied to our bluetooth module. Since we weren't too concerned about a perfectly stable supplied voltage, we used a buck converter to step down 120VAC to 5VDC, which was fed directly to our bluetooth module. We were satisfied with the simplicity of this approach, and felt it was appropriate for our purposes.

While we weren't concerned about the stability of the supplied voltages to our op-amps, the same cannot be said for our more sensitive circuit components: our sensors and microcontroller. In this case, we used the 5VDC output from our buck converter, but used a linear regulator to step the 5V down to 3.3V. We had the option of using a buck converter that directly outputted 3.3VDC, but we felt that this approach would lead to a noisy supply voltage. This raised concerns for the longevity of our microcontroller and sensors, so we felt that using a linear regulator would smooth out the signal received by these more sensitive components, ultimately improving their longevity.

2.1.2 Heating System

There were two main considerations when designing the heating system. The first consideration was figuring out how we wanted to control the heating coils. From the beginning of our design phase, we were confident that we wanted to use pulse-width modulation (PWM) to control the current/voltage across our heating wire at any given moment. We felt that this approach would give us finer control over the speed of heating of our heating wires, as compared to using a global switch that would only allow us to turn the coil either completely off or running at full capacity.

The second consideration was the total drawn power of our blanket. Our cursory research suggested that our blanket should be using around 100-150 Watts on average [7]. We also knew that the heating wire's resistance is directly proportional to its length. As such, we could use the average power to determine a suitable length for our heating wire using the following equations.

$$\overline{P} = I_{RMS} V_{RMS} \qquad V = IR \qquad R = \rho_{Resistance/Meter\ of\ wire} * L_{Length\ of\ wire}$$

We also briefly considered the difference between running our heating wire segments in parallel or in series with each other. We believed that running the coils in series would allow us to better control the total power consumed by our entire blanket at once, but we knew that in order to control the wire segments independently from each other, we were forced to run them in parallel. Thus, we chose the latter option for our final design.

2.1.3 Control System

All figures for this chapter will be in section 2.3. As shown in Figure 2.1, our control system consisted of the microcontroller, peripheral sensors, and our phone app which connected through bluetooth. Given the importance of our sensors to our control system, we needed to check if they would be sensitive enough for our purposes. We intended to use the MCP9808, which is small enough to fit multiple sensors on our blanket without being too noticeable, and we consulted its datasheet to perform the tolerance analysis [8]. Realistically, we do not expect to heat our blanket outside of the 15°C - 50°C range. The operation range for our sensors is -55°C to 125°C, which means our expected temperatures should not pose a problem for our temperature sensors. Let us assume that due to random error, or due to uneven heating, our coils end up being 5% hotter than we intend them to be. This means that our coils could reach up to 52.5°C, which is still well within the standard operation range. As such, operating temperature will likely not be an issue for our temperature sensors. For the sensitivity of the needed, it should give us measurements within +/-0.4°C of the real temperature. This will be more than sensitive enough for us to achieve an ideal output tolerance of +/- 1.5°C.

Secondly, we needed to check if our accelerometers (LIS3DHTR) would work for our use case, and we used its datasheet for this [9]. Firstly, checking the operating temperature range, -40°C - 85°C, we see

that these sensors will have no problems at the temperatures we intend to work at. Let's say we set our accelerometers at their most sensitive setting. In this case, they can measure up to +/- 2g of acceleration, or around +/- 20 m/s^2. This will cover our high level requirement of movements with acceleration exceeding 10 m/s^2 in magnitude. Our ideal output tolerance for our accelerometers will be +/- 1 m/s^2, or around +/- 0.1g. At 25°C, the datasheet tells us to expect +/- 1% tolerance in our measurements, and that this percentage will scale with our operating temperature by a rate of 0.01%/°C. At our highest operating temperature of 50°C, this should mean a tolerance of around +/- 1.25% (1 + 25*0.01), which comes out to about a +/-0.025g tolerance for our maximum linear acceleration measurements of +/- 2g. As such, these sensors will be more than sensitive enough for us to achieve an output tolerance of +/- 1 m/s^2 for our acceleration measurements at all times.

We also knew that we needed a PWM output from our microcontroller for each of our heating wire circuits. The high pin requirement of multiple PWM signals and sensors led us to choose to hook up our sensors on I^2C lines allowing us to limit the number of wires running along our blanket. We ultimately chose the STM32L412KBT6, for its low energy, high frequency, and its availability in LQFP packages. This was not the only option we had at the time, but we felt it was sufficient for our purposes. However, after having finished the project, we believe that alternative processors like the ESP32 may have been a better choice for future versions of the blanket, especially considering the built-in bluetooth capabilities.

2.2 Design Details

2.2.1 Power System

In our final design, we used five PCBs concurrently. Four out of the five PCBs were the heating coil PCBs, which were identical, and the design for which is found in Appendix A. The fifth PCB was our microcontroller PCB, and the design for this is found in Appendix B. We ultimately did not choose to make a separate PCB to house the components for our power subsystem. Instead, the individual components were split amongst the heating PCBs and the microcontroller PCB. The exact implementation can be seen in Figures 2.3. and 2.4. The full circuit diagrams, from which Figures 2.3 and 2.4 were generated, can be found in figures A.1 and B.1 in Appendix A and B respectively.

As seen in Figure 2.3, we had an AC/DC buck converter as well as a linear regulator on our microcontroller PCB. We used the buck converter to first translate the 120 VAC signal from the wall outlet into a 5VDC signal which could be used by our bluetooth module. However, as mentioned in section 2.1, we were concerned about the noisiness in the signal when feeding it into our microcontroller and sensors. As such, we used a linear regulator to step down the 5V into the 3.3V necessary for these aforementioned circuit components, and we were confident that the output of a linear regulator would be stable enough for our purposes. Capacitors were added to the input and output sides of our regulator in the configuration shown in Figure 2.3 to stabilize the circuit operation.

We also had buck converters on the heating PCB, as shown in Figure 2.4. These generated the +/- 12VDC from the 120VAC wall outlet voltage, which was then used by the op-amps in each heating wire circuit. In

order to reduce the number of overall components, we had external connectors on each heating PCB for the +/- 12VDC, so we would only need to use one set of buck converters, and the output of the buck converters could be relayed to each consecutive heating PCB. We also had external connectors for the PWM and sensor voltage signals, as shown on the left hand side of Figure 2.4. These would be provided by the microcontroller PCB, and long, physical wires would be attached between the microcontroller PCB and each individual heating PCB. Finally, we had explicit connections for the two nodes of our 120VAC wall outlet voltage. The +VAC and -VAC external connectors shown in Figure 2.4. would be connected directly to the hot and neutral nodes of the standard U.S. wall receptacle, which can be seen in Figure 2.2. Eventually, all of the boards would be connected in parallel to each other with respect to these hot and neutral nodes.

2.2.2 Heating System

The heating system can be broken down into two main subcircuits. First, a modulation circuit that uses a PWM input from our microcontroller to modulate the AC voltage across our heating coil. The specific implementation can be seen in Figure 2.5. The second subcircuit is the op-amp signal amplification circuit, which is used to boost the signal coming from our microcontroller to a higher voltage. The implementation of this circuit can be seen in Figure 2.6. The full circuit diagram, from which Figures 2.5 and 2.6 were generated, can be found in Figure A.1 in Appendix A.

Starting with the modulation circuit, the initial design took inspiration from a post on *labprojectsbd.com* by MKDas [10]. While our final circuits look relatively different, we took the core idea of using a bridge rectifier in combination with an N-type MOSFET to achieve our desired modulation effect. The reason a bridge rectifier is necessary in this case is because of the structure of a MOSFET, which does not allow for reverse current flow from the source to drain nodes. By using a rectifier, we can ensure that the voltage at the drain node is always greater than or equal to the voltage at the source node, regardless of the phase of our AC voltage source. We then use the NMOS as an electrically controlled switch by hooking up our microcontroller generated PWM wave into the gate node of our NMOS. By doing so, we should be able to finely control the voltage, and therefore current, running across our heating wire at any point. We ran a simulation using LTSpice in order to see the feasibility of this circuit. Figure 2.7 shows the circuit that we recreated in LTSpice, and the voltage source V1 represents the PWM wave we expect from our microcontroller. The results of the simulation can be found in Table C.1 in Appendix C, and very clearly we can see that this is exactly the kind of behavior we want from a modulated AC signal. As such, this gave us the confidence to move forward with the exact design in Figure 2.5.

This brings us to the utility of the op-amp signal amplification circuit in Figure 2.6. We know that in certain circumstances, a MOSFET can be seen as a variable resistor, depending on the gate to source voltage supplied. Typically, for higher V_{GS}, the higher the drain current and the lower the apparent Rds(On) (drain to source resistance while MOSFET is on). For the kind of power MOSFET we were using in our design, the IRF840, we really want a V_{GS} of around 10V, in order to maximize the efficiency of our MOSFET [11]. This poses a problem, since our microcontroller can only output signals of up to 3.3V, so

our PWM waves can only range from 0-3.3VDC. Thus, we decided to use a standard op-amp amplifier circuit as shown in Figure 2.6. We chose a non-inverting orientation, so the gain would be: $A = 1 + \frac{R1}{R2}$, where R1 and R2 correspond to the same labeled resistors in Figure 2.6. We want a gain of 3, since we want to scale a 3.3V signal to $10V \approx 9.9V$, which leads us to wanting $\frac{R1}{R2} = 2$. Thus, as shown in Figure 2.6, we chose R1=100k and R2=50k, which gives us our desired voltage gain. The amplified signal is then fed into the gate node of the MOSFET in our Figure 2.5. circuit.

The last part was choosing the length of our heating wire. We can do this using the equations referenced in section 2.1.2. We also wanted each wire to use approximately $I_{RMS} = 1A$. While this was a somewhat arbitrary choice, we believed it was reasonable considering the upper current limit of a standard U.S. wall receptacle is 15A, and we wanted our heating wires to heat up relatively quickly if desired. Given this, our average consumed power should be around 120W. As such, this gives us all we need to determine the length of each heating wire.

$$\overline{P} = I_{RMS} V_{RMS} = I^2_{RMS} * R = 1^2 * R$$
$$= \rho_{Resistance/Meter\ of\ wire} * L_{Length\ of\ wire} = 33/Meter * L_{Meters} = 120$$

$$L = 120/33 = 3.6364\ meters.$$

As such, if we want to play it relatively safe and run on the lower end of our satisfactory power consumption, we want our heating wires to be around 3.75-4m minimum, and that is the design that we ultimately chose.

We also chose to add fuses in series with all of our heating wire connections, particularly in the area between the bullet connection from the wire to the heating PCB. We used the AD55ABB. We did not add the resistance of the fuse into any of our calculations because the fuses have resistance on the order of 4 milliohms, which is too small to have any significant impact for our purposes [12]. The current limit for this version was 35A, which is a value that we would never reach under normal conditions. However, the AD55ABB *does* trip at 55°C, just above our maximum desired operating temperature. This means that if at any point, any of our heating coils reaches above our max operating temperature, the fuse would trip and break open the circuit that is the sole source of heat, cutting off any current that is flowing through a problematic heating wire. This is to ensure that our blanket is never at a dangerous temperature for the user.

Final note for Figure 2.5: the circuit elements R3 and R4 were used to minimize long term damage to our MOSFET. R3 helps to reduce unwanted spikes in voltage across our MOSFET, which could potentially cause issues like avalanche voltage, and R4 helps the MOSFET turn off quickly when our PWM goes low, reducing the chance for an unwanted latching effect [13]. Circuit elements C3 and R9 are included in the diagram, but were ultimately not used, and therefore do not have explicit values.

2.2.3 Control System

The control subsystem consists of two sub-subsystems, the electrical part that lies on the blanket and the user interface which is designed as an iOS app.

2.2.3.1 Control System - MCU and Sensors

The electrical component of the control system is spread across five PCBs and includes the MCU, sensors, and Bluetooth transceiver module. The MCU is held in a central PCB alongside the Bluetooth module, and the sensors lie on the heating PCBs that are in each zone of the blanket allowing use to receive sensory information from all over the blanket. Per the specification in the STM32L4+ Application Note [14], we carefully chose sizes for our decoupling capacitors that lie next to our STM32L412 chip. These lie next to the MCU in order to keep noise off the voltage line, one for each VDD input. Further, we placed a Ferrite Bead on the VDDA line to ensure that we filter out all noise, considering the delicacy of the analog line. We used a Mux to be able to select boot modes, and ultimately chose to boot our microcontroller on startup from Flash.

For safety, we included a power signal LED, and attached a debugging LED to pin PA4 on our MCU. In order to program our MCU, we went with a connector for an ST-Link in order to use the Serial Wire Debug (SWD) interface. This allowed us to acquire data from all variables within our code and store it for statistical analysis and debugging. Finally, we added a connector for our selected HM11 Bluetooth module to one set of the USART lines available on the MCU and used the pull-up resistors built into the pins.

To interface our board with the heating PCBs, we included four PWM lines coming out of the microcontroller for the heating PCBs, divided the eight sensors among the two I^2C lines on the MCU, and added 3.3V, +12V, -12V and GND connectors to feed the sensors and opamps of the heating boards. To minimize the amount of wires traveling across our blanket we chose to use I^2C as our communication with our sensors and placed two IMUs and two heat sensors on each I^2C line, resulting in a total of 8 wires going to each heating PCB.

After hashing out our schematic for the microcontroller board, we placed the sensors on the heating PCBs as seen in Fig. B.1., one IMU and one heating sensor per board. We added 10uF and 100nF decoupling capacitors across the power of the IMUs because they are sensitive to noise and added 10k Ohm pull-up resistors on our I^2C lines. Each of the sensors has the same base I^2C address, but with the help of MUX's we set the address pins on the sensors. Upon receiving our IMUs we realized that there was only one address pin, which allowed us to have only two IMUs per I^2C line and ultimately limited us to four zones.

The last and most critical part of the electrical control system is the correct PWM output for the heating of each zone of the blanket. This is necessary both for meeting the High-Level requirements of our project and in general for the safety of our users. In order to compute the correct duty cycle to output on each PWM line, we design an algorithm that takes as input the current temperature of the blanket and the target temperature of the blanket.

For proof of concept, we used threshold values for the difference between the current temperature and the target temperature in the first iteration of our algorithm. The STM32L412 has a PWM duty cycle scale of 0 to 1000. We manually set our PWM to 0, 300, 800, and 1000 based on the thresholds of temperature differences of < 3 F, < 8 F, < 15 F, and >= 15 F. We did not know how long it would take to heat the blanket, so these numbers were an educated guess.

Upon successfully heating the blanket, we finalized our equation and modeled it in Python. We decided to use a logarithmic model to allow us to heat at a smooth rate. It takes as input the current temperature of the blanket and the target temperature, allowing the zone of the blanket to immediately start heating to a new value if the target temperature is changed:

$$\mathrm{PWM}(T_{\text{target}}, T_{\text{curr}}) = \begin{cases} \text{scale}\left(\ln(\mathrm{dt}+1)\right) & \text{if } \mathrm{dt} > 0 \\ 0 & \text{else} \end{cases} \qquad \mathrm{dt} = T_{\text{target}} - T_{\text{curr}}$$

$$\text{scale}(x) = \frac{x}{\ln(T_{\text{target}} - T_{\text{init}} + 1)} \times 1000$$

The model of PWM output as a function of the current temperature for a target temperature of 89 degrees F and starting blanket temperature of 72 degrees F is found in Appendix D, Fig D. 1. Additionally, we created a model of the desired temperature change of the blanket by integrating our log function, and this can be found in Fig D. 2. To confirm our results, we stored the temperatures of zone one into an array, over every few iterations of our main code loop. This data is for setting a target heating temperature of 89 degrees F. There was a delay between the heating of the heating wire and the heating of the blanket zone, so we had to actually adjust the equation to target a temperature that was lower than the true target. After messing with the parameters, we were able to closely reproduce our desired temperature change in zone one, and we created a graph of this data in Fig D. 3.

Overall this was a success; however, the speed of heating of the blanket is quite high and with more time we would like to adjust the formula to slow it down for the sake of user safety considering that we blew a fuse during demo. In the future we want to further improve our heating curve by implementing PID control, as well as a time scaling factor that will allow the user to set how long they would like it to take to heat their blanket.

2.2.3.2 User Interface

The user interface for this product is in the form of an iOS application, coded in Swift using the Xcode editor. The app provides the users with the ability to set the blanket to a specific temperature, schedule a temperature for a time, and see all of the data collected by the on-board sensors. The app communicates with the HM-11 Bluetooth module on the MCU board via BLE protocol, which is designed specifically for short bursts of data transmission. The Bluetooth functionality is implemented using Swift's CoreBluetooth library. For our purposes, we designed our own protocol for communications between the MCU and app: packages of four bytes in the format of zone identifier, data byte one, data byte two, package end signal. With this, we can use a single package to transmit and receive data as large as 127.99 (due to the 127 max size of a signed integer) pertaining to one of the four zones on the blanket.

The app itself is split into pages to make navigation easier for the user. The Home page (Fig 2.8) lets the user know the status of the connection with the blanket; the page either says Bluetooth not enabled, blanket not connected, or blanket connected. Once connected, the page allows the user to set a single global temperature across all four zones, power off the blanket's heating, see the current average temperature across the four zones, or navigate to the other pages. The first of these pages is the Zones page (Fig 2.9), which allows users to set a temperature for each zone individually and view the sensor data for each zone. The Graphs page (Fig 2.10) uses charts created with Swift's Charts library in order to show a line graph of the average temperature across the zones over time as well as a plot of each accelerometer reading over time. The final page is the Schedule page (Fig 2.11), which allows users to set a single global temperature at a single time of their choosing -- when this time is reached, each zone's goal temperature is set to the chosen temperature.

In terms of the implementation of the app, it is split into three main files: ContentView, CoreBluetooth, and DataStorage. ContentView contains all of the user interface code, including layouts, buttons, and charts. CoreBluetooth contains code for sending, receiving, and dissecting signals to make their data usable. DataStorage contains arrays of data points, each with a unique identifier number, a value, and a timestamp.

Whenever a signal is received and dissected, the data is given an ID, time-stamped, and logged to a DataStorage function that determines by value whether the data point is a temperature or accelerometer reading. The data point is then added into its corresponding array. These arrays are non-persistent, meaning that they only contain data from the current session on the app; once the app is closed, that data is gone forever. ContentView functions use the data points to create the charts on the Graph page and they use CoreBluetooth functions to send signals and use data like the last received temperature to display to the user.

2.3 Figures

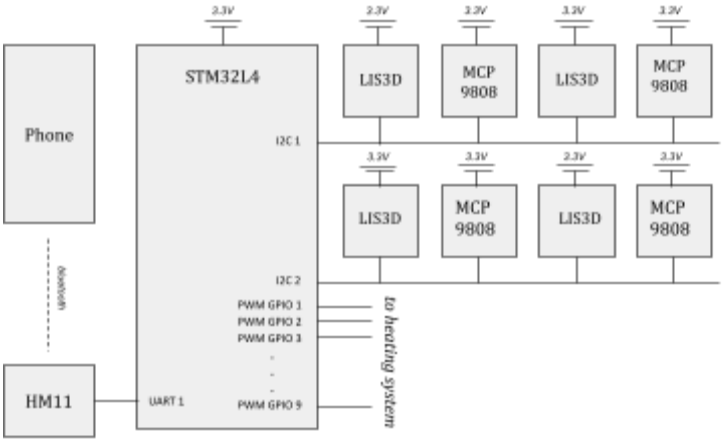


Fig. 2.1. Block diagram of control system.

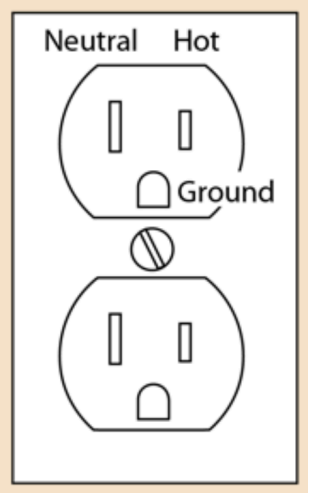


Fig. 2.2. U.S. wall receptacle [15].

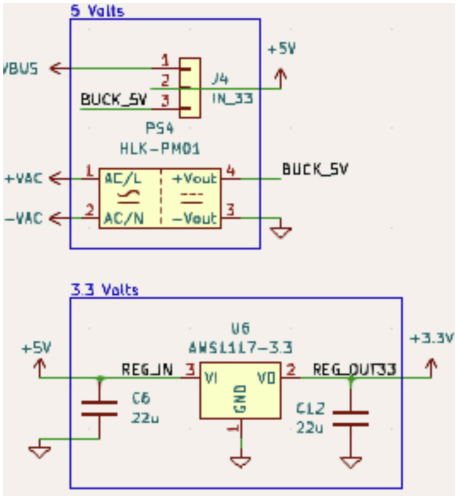


Fig. 2.3. Power subsystem on microcontroller PCB

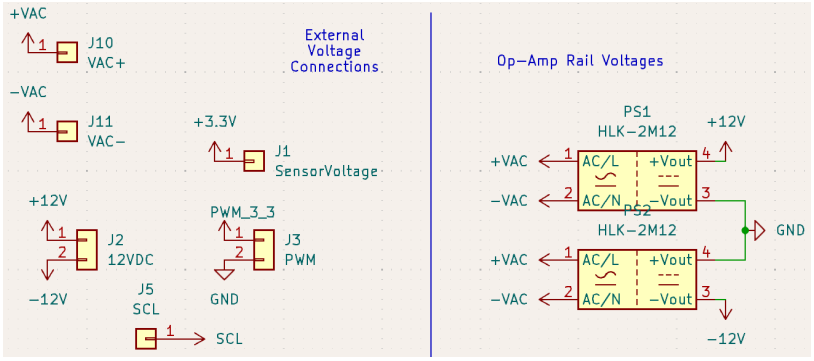


Fig. 2.4. Power subsystem on heating PCB

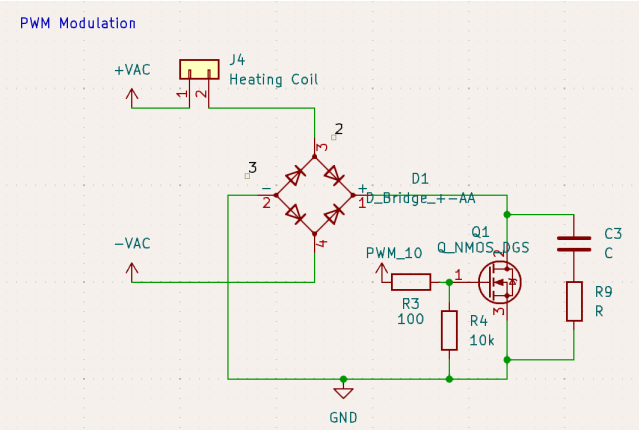


Fig. 2.5. Heating coil modulation circuit.

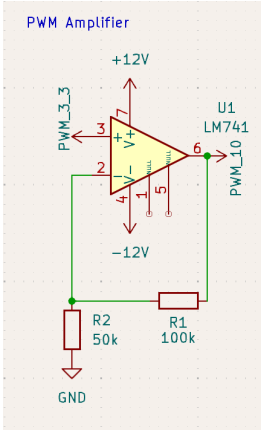


Fig. 2.6. Op-amp signal amplifier circuit.

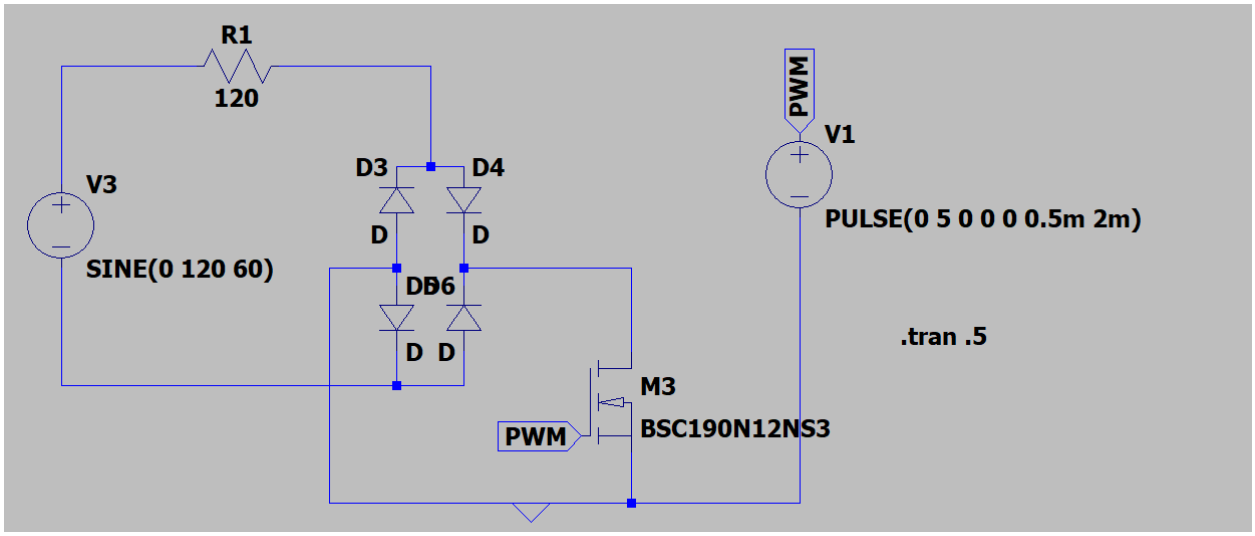


Fig. 2.7. AC Pulse Width Modulation circuit for heating coil.

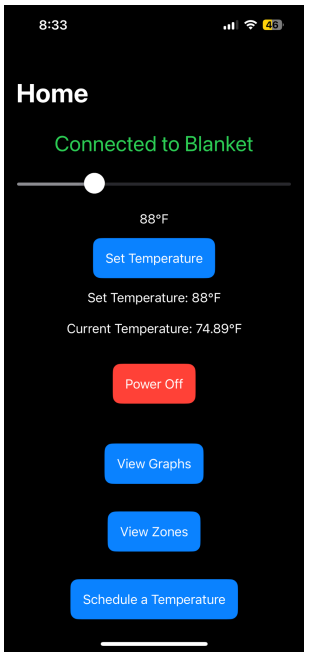


Fig. 2.8. Home page.

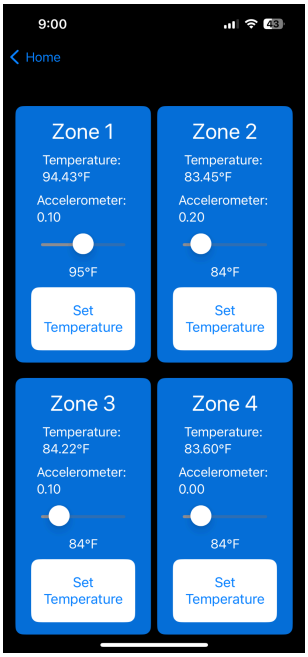


Fig. 2.9. Zones page.

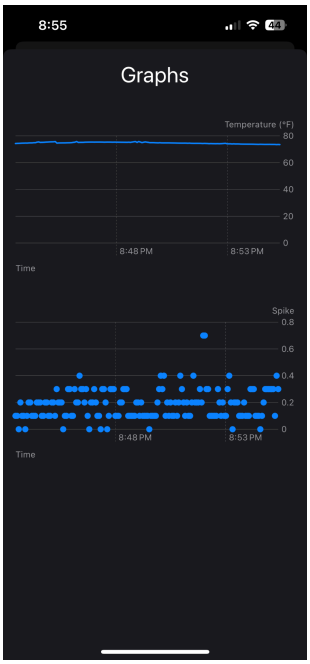


Fig. 2.10. Graphs page.

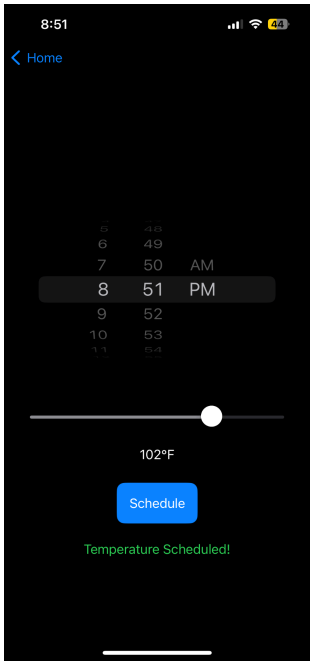


Fig. 2.11. Schedule page.

3 Verification

3.1 Requirements and Verification

3.1.1 Power System

The requirements and verification for this subsystem can be found in Appendix E, specifically in Table E.1. All of the requirements are very low level, essentially checking that the voltages at certain points were within acceptable ranges. We were able to verify these simply by using a multimeter at the desired points, and the values we saw were satisfactory for our purposes. All of these low-level requirements were successfully verified, so nothing further will be discussed on this.

3.1.2 Heating System

The requirements and verification for this subsystem can be found in Appendix E, specifically in Table E.2. In large part, the verification for this subsystem is checking whether or not an input PWM signal is being properly translated into the AC voltage across our heating wire, both individually and concurrently with multiple wires at once. In section 2.2.2 we discussed how the simulated circuit was working correctly, so in order to conduct the verification for the physical circuit, we had to check whether or not the physical circuit was replicating the behavior of our simulations.

We constructed the circuit in accordance with Figure 2.5, which was the heating coil modulation circuit that we built onto our PCB. We anticipated that the minor differences between the PCB version and the simulated circuit would cause negligible changes in our empirical results. After constructing the circuit, we tested the circuit using the waveform generators and power supplies in the lab, and used the ADALM2000's oscilloscope function to check our voltages. We tested the circuit at lower voltages first and foremost due to safety considerations, but secondly because the instruments that we were using were unlikely to be able to handle 120VAC over an extended period of time. As such, we decided that testing at lower voltages would be sufficient to verify the viability of our design.

The empirical data we collected can be found in Tables C.2 and C.3 in Appendix C for single wire performance and multiple wire performance, respectively. For the multiple wire case, we simply ran multiple wires at the same time and tracked the voltage across one of the wires, chosen at random. This was partially due to the fact that we did not have the hardware to simultaneously track four oscilloscope readings, but also because we felt that there should not be a noticeable difference between the individual coils in that case. Upon first glance, the behavior is very similar to the simulated data in Table C.1, but we wanted to perform some more in-depth analysis of the results to be absolutely certain.

Zooming into our oscilloscope readings from both Tables C.2 and C.3, we can use identical segments to determine the duty cycle across the wire, as shown in Figures 3.1 and 3.2. All unique figures for this chapter will be in Section 3.2. We see that in both cases (individual heating wires and multiple heating wires), at least visually, the duty cycle of the voltage across our heating wire aligns well with the input duty cycle. For the 25% duty cycle inputs, the time that the wave is non-zero is approximately one out of four equal segments between two rising edges. For 50% duty cycle inputs, the time that the wave is

non-zero is approximately one out of two equal segments between two rising edges. For 75% duty cycle inputs, the time that the wave is non-zero is approximately three out of four equal segments between two rising edges. We felt that this was well within the +/-5% tolerance that we allotted ourselves in our requirements and verifications for this subsystem, which again can be found in Table D.2.

3.1.3 Control System

The low level requirements and verification for this subsystem can be found in Appendix E, specifically in Table E.3. Each of our tests for our MCU power, sensors, and PWM signals meet the verification criteria. To keep well within the switching time specs of the heating subsystem's power mosfets, we scaled the frequency of the PWM outputted by the MCU from the base 80 MHz to the 1000 Hz.

Concerning our app-MCU communication requirement, our Bluetooth module uses Bluetooth Low Energy for its low price and power consumption; however, it comes at the cost of sending single byte packets at a time. After meeting our low-level verification criteria for sending integer based sensor readings, we wanted to allow for decimal numbers to meet our higher sensitivity high-level requirements. As we discussed in Section 2.2.3.2, this is why we developed our proprietary four-byte packet protocol.

3.2 Figures

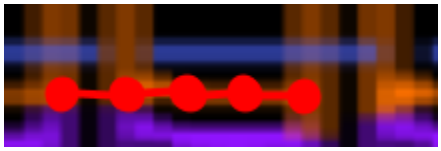
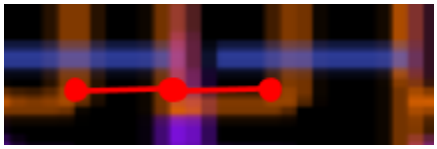
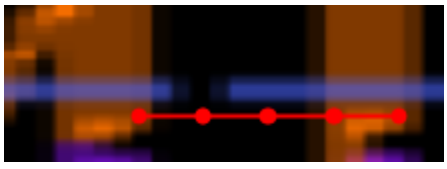
Duty Cycle	Voltage Across Heating Wire
25%	
50%	
75%	

Fig 3.1. Duty cycle measurements (individual wire)

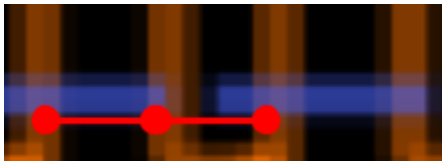
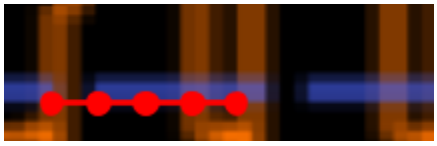
Duty Cycle	Voltage Across Heating Wire
25%	
50%	
75%	

Fig 3.2. Duty cycle measurements (multiple wires)

4 Costs and Schedule

4.1 Costs

When considering the success of this project, the cost is one of the most important factors to consider. Because we aim to create a cheaper alternative to existing solutions, the cost needs to be low enough to justify a low price. Starting off with labor prices, we have three members; each member put in an estimated 100 hours of work. Assuming each member makes $40/hour, which would be reasonable for this type of project work, the total labor cost would be: 3 (members) * 120 (hours of work each) * 40 ($/hour) * 2.5 (overhead) = $36,000. In addition to labor costs, we have to buy materials for the blanket. Looking at Table F.1 in Appendix F, we can add up the materials to get $289.14 in total material for the project. This gives us a total cost of $36,000 + $289.14 = $36,289.14. This could possibly be slightly off due to estimating costs from the ECE shop, but it is a very realistic estimate.

Although the final cost is significantly higher than even the top existing options on the market, we anticipate that this price would reduce greatly once the product begins mass production. For one, labor costs will naturally decrease given that the workers would be less "skilled" in nature given they would be working on assembly and manufacturing rather than research and development. The number of hours would also decrease once the design is finalized, as the vast majority of the time spent on this project was research and prototyping. Finally, the actual cost of materials for the product would decrease due to buying in bulk directly from manufacturers and buying only the materials we need. As a simple estimate for future material costs, this could get as low as $160 factoring in items we purchased that we would not need and saved costs due to bulk purchasing. Given all of this, it is feasible to anticipate that once the product begins mass production, it will in fact be a cheaper alternative compared to many existing solutions to temperature-controlled sleep.

4.2 Schedule

Week	**Milestones completed**
02/18-02/24	Design Document and Project Proposal regrade are finished (all members work together synchronously).
02/25-03/02	We started the first draft of our PCB, headed by Patrick, and brought it to the PCB review, and began to order parts, headed by Alex. Wyatt did brief brainstorming on desired functionality for the blanket.
03/03-03/09	Revised the PCB and submitted an order in the first round. Ordered all other parts we needed (ECE shop, Digikey, Amazon, etc.) (all members work together synchronously).
03/10-03/16	Because this was spring break, all members worked on the project asynchronously on their own time. Patrick made modifications to the circuit as necessary, Wyatt and Alex began drawing out the skeleton for the application code.
03/17-03/23	Wyatt worked on the code for the IOS app. Alex worked on the code for the STM32

	Microprocessor, starting with the dev board. Patrick did unit testing on all of the sensors.
03/24-03/30	Each member completed their individual progress report. Patrick did most of the circuit soldering and testing, but everyone was involved. Second iteration of the PCB was submitted for 3rd round ordering.
03/31-04/06	Alex finished up the microcontroller PCB and submitted that for 4th round ordering. Delays in the 3rd round caused concerns, because without a stencil the IMUs were not physically testable. Patrick tested everything else on the heating PCB. Wyatt ordered breakout boards for the IMUs just in case.
04/07-04/13	Received our 3rd round PCBs. Patrick finished up testing the PCBs, along with Alex. Wyatt began flushing out the STM32 dev board code to test I^2C communication with IMUs. Due to previous PCB order delays, Patrick submitted the final versions of all PCBs and ordered stencils.
04/14-04/20	Mock demo week. Patrick wired up an example heating coil circuit with the available resources. Wyatt finished up the bluetooth module code to connect with the STM32 dev board. Alex helped debug the STM32 UART communication code.
04/21-04/27	Alex focused on transferring STM32 dev board code onto our actual STM32, debugging all issues that came up. Wyatt added all desired functionality to the IOS app and worked with Alex to get two-way communication working over bluetooth. Patrick finished building and wiring up the physical blanket, testing all physical connections.
04/28-05/04	Gave final presentation as a team. Worked together on finishing up the final report, and we each worked individually to finish up our lab notebooks.

5 Conclusions

5.1 Project Outcome

Overall, this project is a success. While the cost of development was significantly higher than planned, much of the cost came from labor and prototyping. Now that the initial product has been completed, further versions will take less time and fewer purchases as the design is refined. Once the design is refined, the time and therefore the cost of labor will significantly decrease. As mentioned in Section 4, the cost of parts will also significantly decrease due to bulk buying. All of these factors will lead to a price that we believe will be cheaper than other temperature-controlled sleep products.

Apart from cost, the product itself worked very well. There is a simple and easy-to-use app that allows users to easily interact with the blanket's features. The blanket itself is comfortable on the user side and the heat it provides is pleasant. The sensors provide accurate data and the software can use the data to allow for accurate heating, as well as giving that data to the user for them to use for pattern detection between their movement and the blanket's temperature. The only issues the product currently has are with the heating system: the two types of heating wires result in different heating patterns that need to be further refined to work in tandem, and we burned a fuse during our demonstration due to this unrefined heating. However, these are issues that can be fixed with further testing and future versions of the product.

5.2 Ethical Considerations

As outlined in the IEEE Code of Ethics [16], this project was conducted with responsibility, ethical conduct, and safety as the top priorities. We strove to create a product that protected the safety and welfare of those involved in the project, avoided all unlawful or unprofessional activities throughout the production, and ensured the team had the proper qualifications to undertake any task before starting it. Each team member was treated with fairness and respect, and any risky or potentially hazardous elements of the project were communicated as such, resulting in no damage to people or property. We did collect data from the blanket's sensors, but this data was not associated with any individual; in the future, any identifiable data will be held to the highest standards of privacy and care. Overall, we used the IEEE code of ethics throughout the project to ensure each member had its contents in mind, and we ensured our actions were in accordance with the code.

5.3 Impacts of Project

While this project was overall a success, it will need to face much more intense testing and refinement before being available for any consumer. As it stands, the blanket's heating is not very fine-tuned, resulting in a fuse burning during the official demonstration. While nobody was in physical contact with the blanket during this event, it is a clear indication that the blanket is not ready to undergo any testing with a user. However, once the blanket is more rigorously tested and refined and it is clear that the product is safe, we do believe that it can make a positive impact on society. Sleep is undeniably important for everyday human processes, and quality sleep can be the difference between success and

failure in any number of human undertakings. It is unfair to keep premier sleep technology behind an exorbitant payment of thousands of dollars, and this product addresses this issue and will hopefully allow many more people to experience the benefits of temperature-controlled sleep and improve their quality of life.

References

[1] McCoy Abby, “Thermoregulation During Sleep: How Room and Body Temperature Affects Your Rest Quality,” *Sleepopolis*, para. 3, “Long Story Short”, January 25, 2024. [Online], Available: https://sleepopolis.com/education/thermoregulation-sleep/#:~:text=Once%20we%27ve%20fallen%20asleep,nearly%20stop%20regulating%20temperature%20altogether. [Accessed Feb. 7, 2024].

[2] BedJet, “BedJet 3 Dual Zone Climate Comfort Sleep System for Couples”, *BedJet.com*. [Online], Available: https://bedjet.com/products/bedjet-3-d ual-zone-climate-comfort-system-for-couples. [Accessed Feb. 7, 2024].

[3] Smartduvet, “Smartduvet”, *Smartduvet.com*. [Online], Available: https://www.smartduvet.com/products/smartduvet. [Accessed Feb. 7, 2024].

[4] EightSleep, “Pod Cover”, *EightSleep.com*. [Online], Available: https://www.eightsleep.com/product/pod-cover/. [Accessed Feb. 7, 2024].

[5] Pngtree, “Mobile Phone Png Smartphone Camera Mockup”, *pngtree.com*. [Online], Available: https://pngtree.com/freepng/mobile-phone-png-smartphone-camera-mockup_6067590.html. [Accessed Feb. 7, 2024].

[6] Demdaco, “Velvet Throw Blanket - Gray”, *demdaco.com*. [Online], Available: https://www.demdaco.com/velvet-throw-blanket-gray/. [Accessed Feb. 7, 2024].

[7] Jackery, “How Many Watts Does an Electric Blanket Use?”, *jackery.com*, August 30, 2023. [Online], Available: https://www.jackery.com/blogs/knowledge/how-many-watts-does-an-electric-blanket-use. [Accessed May 1, 2024].

[8] Microchip Technology, “±0.5°C Maximum Accuracy Digital Temperature Sensor,” MCP9808 datasheet, Nov. 7, 2016.

[9] STMicroelectronics, “MEMS digital output motion sensor: ultra-low-power high-performance 3-axis ‘nano’ accelerometer,” LIS3DH datasheet, Dec. 2016.

[10] MKDas, “Learn to regulate an AC load using PWM Signal”, *labprojectsbd.com*, July 7, 2020. [Online], Available: https://labprojectsbd.com/2020/07/21/ac-load-control-using-pwm-technique/. [Accessed Feb. 22, 2024].

[11] Vishay Intertechnology, “Power MOSFET,” IRF840 datasheet, Aug. 30, 2021.

[12] Bourns Inc., “AD Series Breaker (Thermal Cutoff Device),” AD55ABB datasheet, March 31, 2015.

[13] WellPCB, “MOSFET Protection: How to Properly Protect It,” *wellpcb.com*. [Online], Available: https://www.wellpcb.com/mosfet-protection.html. [Accessed May 1, 2024].

[14] STMicroelectronics, “AN4555 Application note,” *st.com.* [Online], Available: *]*https://www.st.com/resource/en/application_note/an4555-getting-started-with-stm32l4-series-and-stm32l4-series-hardware-development-stmicroelectronics.pdf. [Accessed May 1, 2024].

[15] R. Nave, “Polarized Receptacles,” *http://hyperphysics.phy-astr.gsu.edu*. [Online], Available: http://hyperphysics.phy-astr.gsu.edu/hbase/electric/hsehld.html. [Accessed May 1, 2024].

[16] IEEE, “IEEE Code of Ethics,” *ieee.org*, Jun. 2020 [Online], Available: https://www.ieee.org/about/corporate/governance/p7-8.html. [Accessed May 1, 2024].

Appendix A: Heating Coil

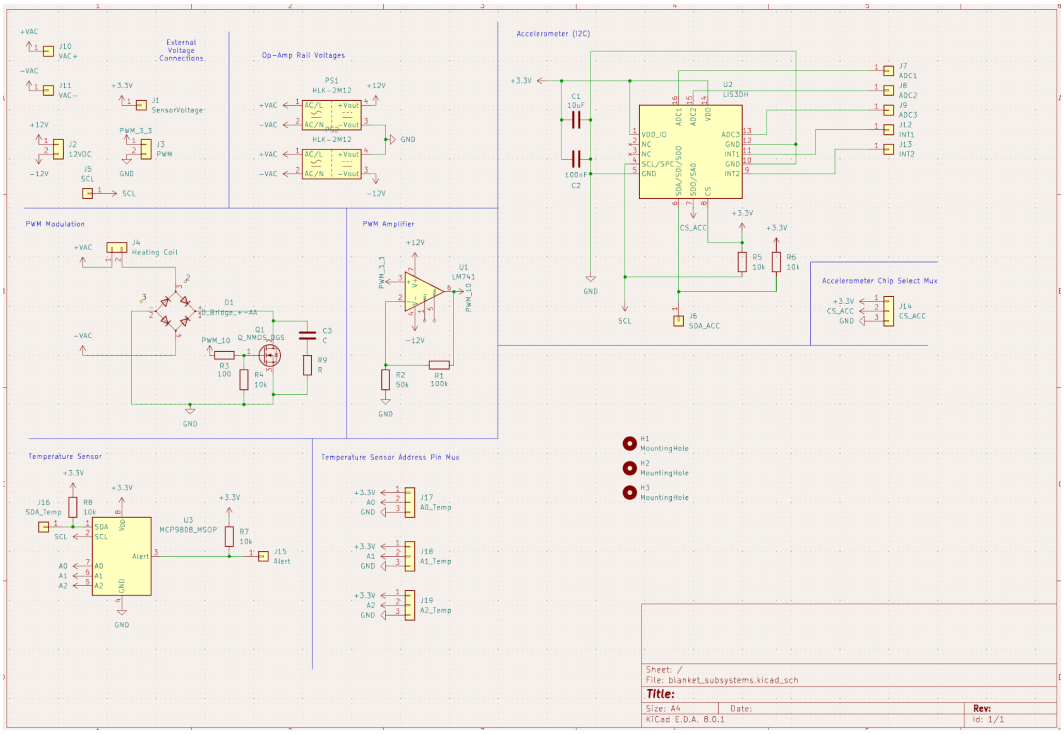


Fig. A.1. Heating coil PCB schematic.

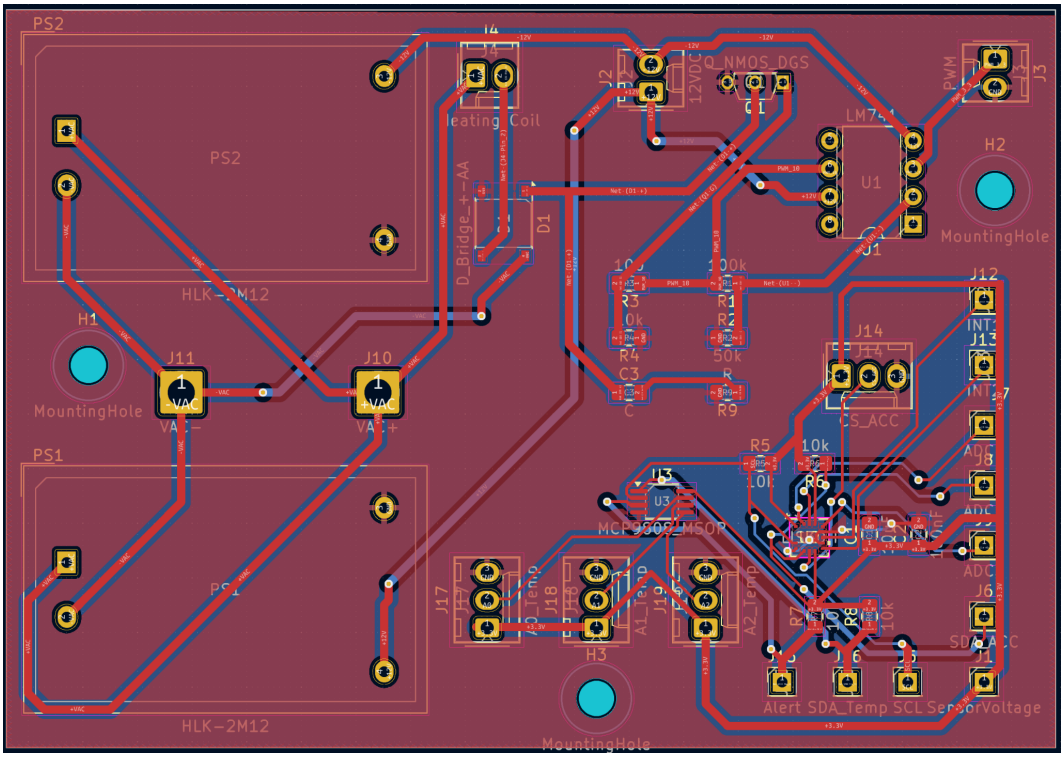


Fig. A.2. Heating coil PCB (KiCAD).

Appendix B: Microcontroller

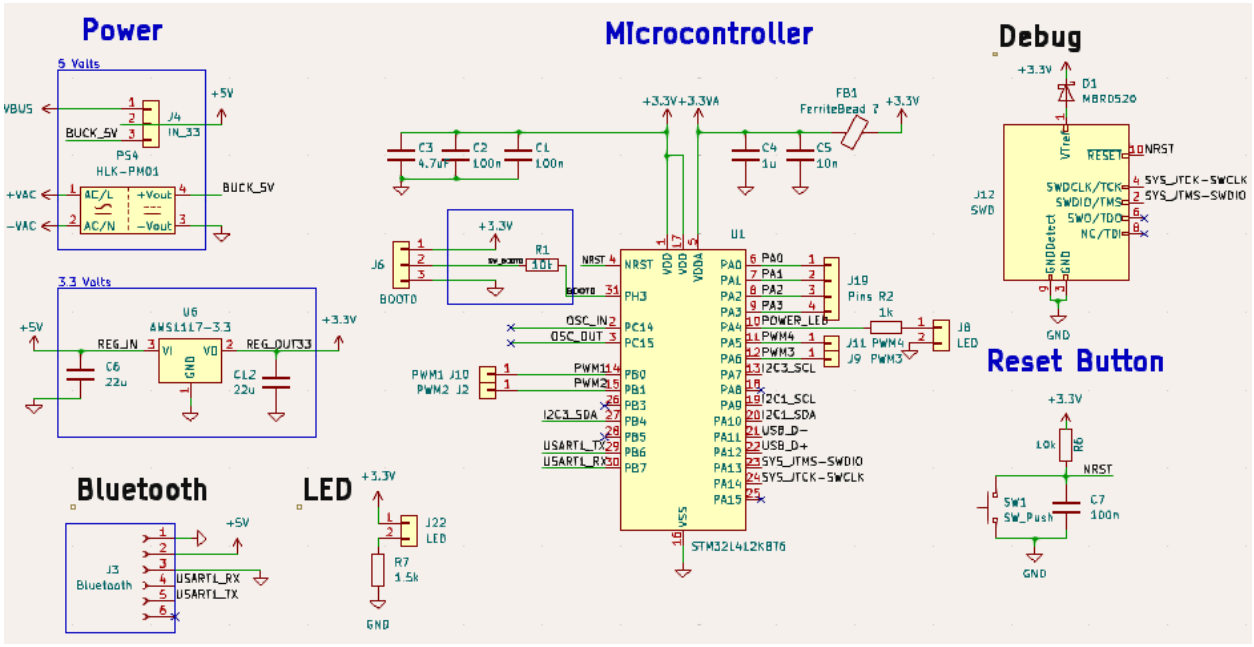


Fig. B.1. Microcontroller PCB schematic.

Appendix C: Heating Coil Simulation and Empirical Data

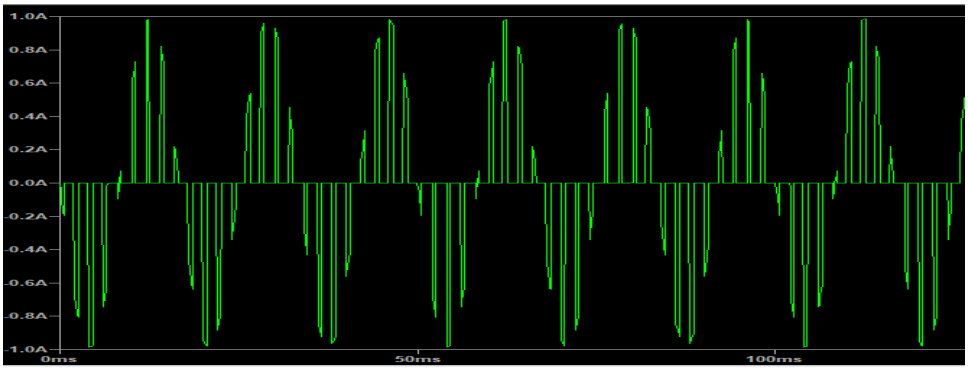
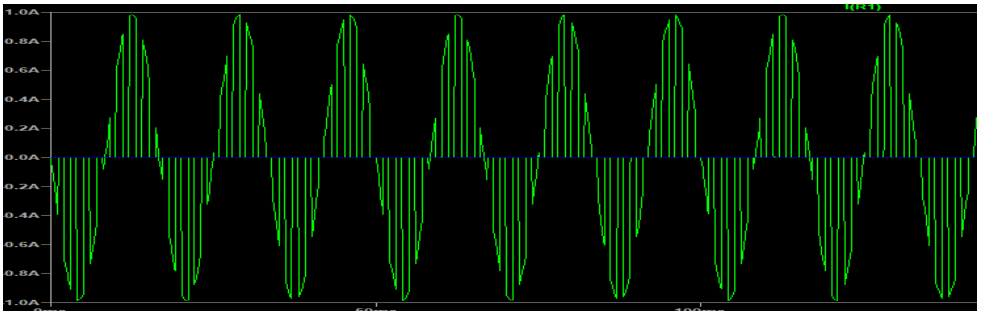
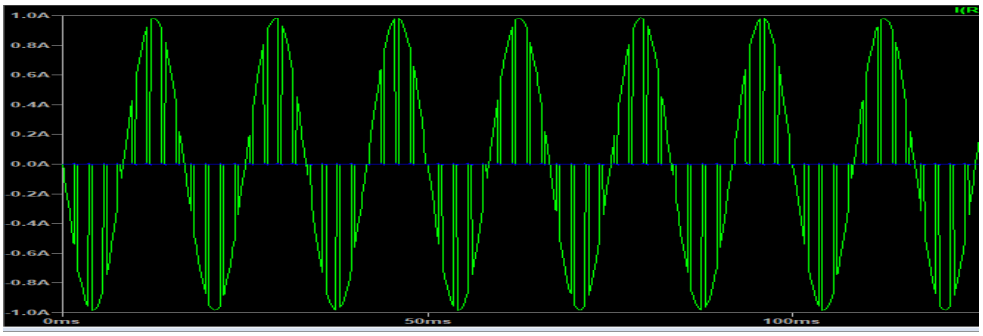
Duty Cycle (%)	Current Across Heating Coil
25%	
50%	
75%	

Table C.1. Simulated AC Pulse Width Modulation for heating wire.

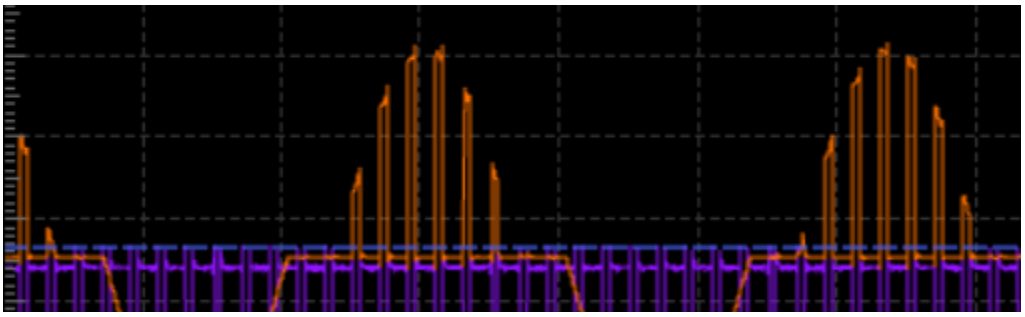
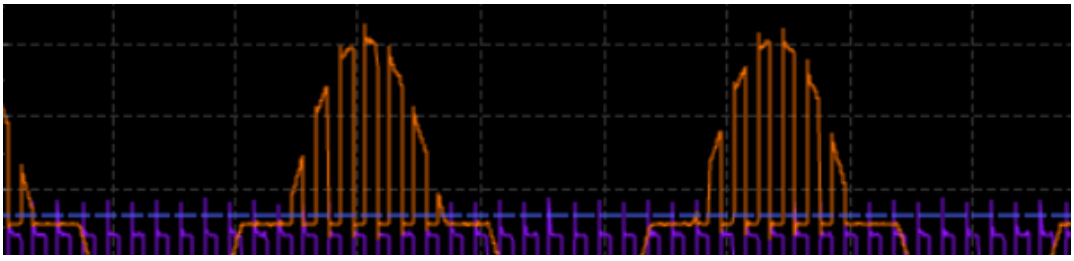
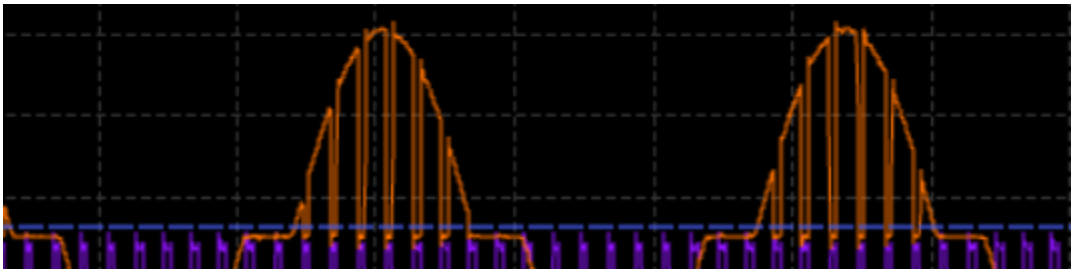
Duty Cycle (%)	Voltage Across Heating Wire
25%	
50%	
75%	

Table C.2. Recorded AC Pulse Width Modulation for individual heating wire (measured with oscilloscope).

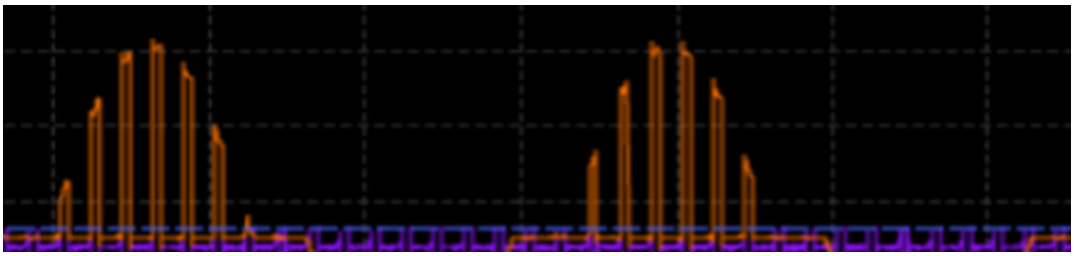
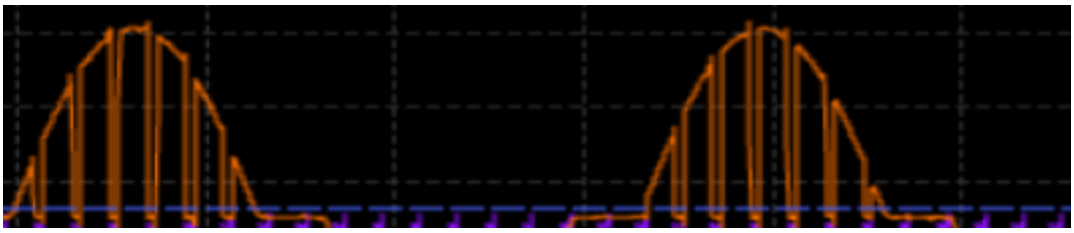
Duty Cycle (%)	Voltage Across Heating Wire
25%	
50%	
75%	

Table C.3. Recorded AC Pulse Width Modulation for concurrent heating wires (measured with oscilloscope).

Appendix D: PWM Models and Data

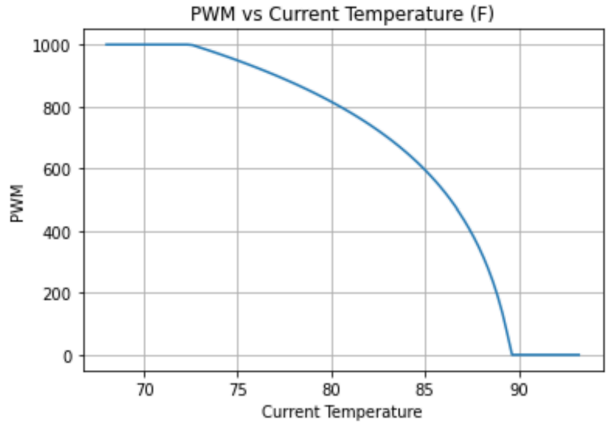


Fig D.1. PWM vs Temperature in Fahrenheit; T_{target} = 89 F.

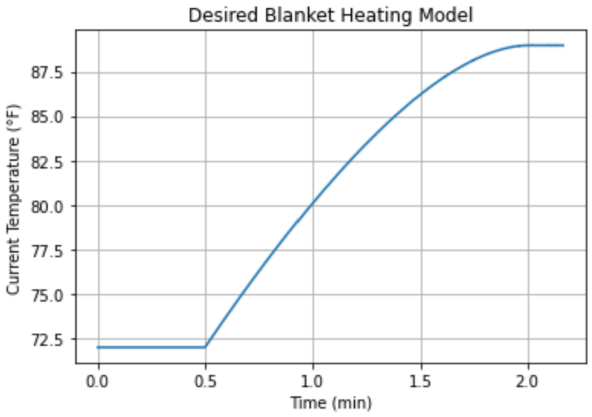


Fig D.2. Ideal heating curve over time; T_{target} is 89 F.

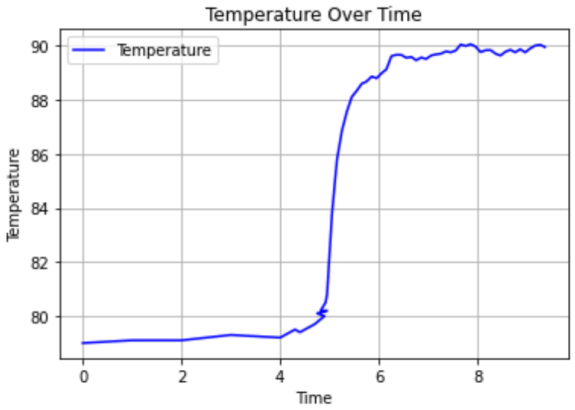


Fig D.3. Empirical heating curve over time; T_{target} = 89 F.

Appendix E: Requirements and Verification Tables

Requirement	Verification
Cord connecting a wall outlet, supplying 115-125V AC to a power distribution circuit for an expected current load of no more than 1.5A.	Measure and track the voltage w/ oscilloscope (30-60s), and see that it is in the desired range (115-125V AC). Determines magnitude validity and consistency of sine wave signal.
Utilize HLK-PM05 buck converter and 5272-AMS1117-3.3CT-ND voltage regulator to produce 3.0-3.6V DC voltage for control subcircuit	Use a voltmeter to track voltage values of 3.0-3.6V (30-60s). Use an oscilloscope to check that the signal is DC.
The power distribution circuit will be used to run the other two subsystems in parallel at the aforementioned desired voltages.	Use configuration of resistors and capacitors to model the expected resistance and capacitance values of our subcircuit designs. Use oscilloscopes/voltmeters to determine each component is running at desired voltage (30-60s).

Table E.1. Requirement and verification table for Power Subsystem.

Requirement	Verification
We want our heating subsystem to be able to interpret the incoming PWM waves and to change its behavior depending on the signal. More specifically, we want the duty cycle of the AC current powering each of our coils to be within 5% of the duty cycle of the incoming wave. If our incoming duty cycle is 50%, we want our AC current to be modified to 45-55% duty cycle as well.	1. For just a single coil, input a 50% duty cycle PWM wave into the gate of our MOSFET in the power circuit as described in Figure 4 above. At this point, we want to use an oscilloscope to measure and track the voltage that is output to our load: the heating coils. The goal is to see our modified AC wave to have between a 45-55% duty cycle as well. 2. Change the input duty cycle to 25% and then 75% in turn. Repeat the same test as in part 1, and check that our duty cycles are in the correct tolerance range: 20-30% and 70-80% for 25% and 75% respectively.

Our blanket will ideally be able to utilize all 4 PWM waves - for each of our 4 heating coils - all simultaneously. We recognize that the signals in the coils might interfere with each other, but ideally our end result will be within +/-10% of our ideal duty cycle for all coils at the same time.	1. For all the coils at the same time, input a 50% duty cycle PWM wave into the gates of all the MOSFETs in the power circuits for each heating coil. Then, use the oscilloscope to measure and track the voltage across all of our heating coils. The goal is to have each AC wave have between 40-60% duty cycle. 2. Repeat step 1 for input duty cycles at 25%, looking for our AC voltage to be within 15-35% across our heating coils. Do the same thing with our input at 75%, looking for 65-85% duty cycle across our heating coils.

Table E.2. Requirement and verification table for Heating Subsystem.

Requirement	Verification
Powered by a 3.0-3.6V DC parallel network.	Use a voltmeter to track voltage values of 3.0-3.6V (30-60s), oscilloscope to check that signal is DC, and an ammeter to check that the provided current is >= 0.5A.
Output 4 distinct PWM signals, one for each block of the blanket, that feed into the heating system. Set each signal to a distinct duty cycle.	Check that our microcontroller can output 4 distinct PWM signals. Try to set each signal to a specific duty cycle at intervals of 10% (10%, 20%, 30%, … 90%). Use an oscilloscope to estimate the duty cycle of each wave. We want each wave to be within +/- 1% of the desired duty cycle (i.e. 19-21% for a 20% duty signal).
Each accelerometer will contribute to checking whether the user is asleep. Our goal is to report the user's movements spanning greater than 3 inches (+/- 1.5 inches). This should be visible in the app.	Test the instantaneous magnitude (and direction) of the accelerometer. Integrate to determine the distance traveled within a set time. Output the distance serially to a computer to check validity. Test accelerometer: flip and check magnitude; physically move it by set intervals (e.g. 1 in, 2 in, 3 in, 4 in). Check that measurements fall within the tolerance intervals specified by the requirement. Repeat for 4 accelerometers simultaneously.
Temperature sensors will feed input to the microcontroller to ensure that the temperature that is being achieved by the blanket is consistent with the desired temperature. Our goal is to determine the absolute temperature	With just one temperature sensor, test that we can output a physical temperature using our sensors and microcontroller. Output serially to the computer. Next, use an external thermometer to determine the temperature (goal: sensor data within 1.5 degrees Celsius). Repeat for 4 temperature sensors at the same time.

within each zone (+/- 1.5 degrees Celsius).	
The phone/web app communicates with the microcontroller through Bluetooth. The app sends temperature settings to the blanket that adjust PWM waves. The app receives temperature and movement readings from sensors.	Establish Bluetooth communication. Display the message to a computer upon successful connection. Test output: check that movement/temp. readings sent to the app are equal to computer readings during sensor testing. Test input from app to microcontroller: 1. Specify a heating temperature in our app 2. Track the temperature of the blanket with a thermometer. 3. Once the temperature stops changing significantly (+/-0.5 C/min), check the mean temperature for 2 min (goal: within 3 C of our desired temperature) 4. Increase the temperature of the blanket in the app by 5 degrees Celsius and repeat step 3.

Table E.3. Requirement and verification table for control subsystem.

Appendix F: Costs

Item	Part number	Source	Unit Price	Quantity	Total Price	Estimated
Connector	ED1543-ND	Digikey	$0.33	3	$0.99	N
USB-C Board	1528-2873-ND	Digikey	$2.95	1	$2.95	N
Voltage regulator	5272-AMS1117-3.3CT-ND	Digikey	$0.43	1	$0.43	N
Button	10-EVP-BT1A4A000CT-ND	Digikey	$0.29	3	$0.87	N
LED	1080-1063-ND	Digikey	$0.30	2	$0.60	N
LED	1080-1128-ND	Digikey	$0.29	2	$0.58	N
LED	365-1182-ND	Digikey	$0.26	4	$1.04	N
Capacitor	1276-1244-1-ND	Digikey	$0.069	10	$0.69	N
Capacitor	1276-1066-1-ND	Digikey	$0.062	10	$0.62	N
Capacitor	1276-1015-1-ND	Digikey	$0.040	10	$0.40	N
Capacitor	1276-1003-1- D	Digikey	$0.042	10	$0.42	N
Diode	MBR0520L-TPMSCT-ND	Digikey	$0.33	4	$1.32	N
Capacitor	399-C0603S103K3RAc7867CT-ND	Digikey	$0.31	2	$0.62	N
Capacitor	1276-CL21A226MAYNNNECT-ND	Digikey	$0.162	10	$1.62	N
Diode	264-CUZ5V6H3FCT-ND	Digikey	$0.237	10	$2.37	N
Filter	240-2389-1-ND	Digikey	$0.21	2	$0.42	N
Resistor	RNCP0805FTD1K00CT-ND	Digikey	$0.027	10	$0.27	N
Resistor	RNCP0805FTD10K0CT-ND	Digikey	$0.027	10	$0.27	N
Resistor	RNCP0805FTD1K50C	Digikey	$0.027	10	$0.27	N

	T-ND					
Resistor	738-RNCF0603DTE22K0CT-ND	Digikey	$0.049	10	$0.49	N
STM32 Cable	HiLetgo ST-Link V2	Amazon	$10.34	1	$10.34	N
Bluetooth transceiver	DSD Tech HM-11 BLE	Amazon	$11.99	1	$11.99	N
Header Pins	Male Lystaii Straight 40 Pin 0.1 Inch	Amazon	$9.80	1	$9.80	N
Bullet connectors	PONFY Heat Shrink Bullet	Amazon	$11.99	1	$11.99	N
Jumper wires	EDGELEC 50cm Breadboard	Amazon	$13.99	1	$13.99	N
AC/DC Step Down Converter (220VAC-5VDC)	EC Buying Mini Step-Down	Amazon	$10.89	1	$10.89	N
Jumper Caps	California JOS 100 PCS	Amazon	$5.22	1	$5.22	N
Heating Wire	LKXHarleya 12K Carbon Fiber Heating Wire	Amazon	$8.99	1	$8.99	N
Blanket	Bertte Plush Fleece	Amazon	$8.99	1	$8.99	N
Power N-Type MOSFET	IRF840N	Bojack	$0.799	10	$7.99	N
AC/DC Step Down Converter (220VAC-12VDC)	EC Buying Mini Step-Down	Amazon	$9.99	1	$9.99	N
Heated Blanket	JKMAX Heated Blanket Electric Throw	Amazon	$26.99	1	$26.99	N
Accelerometer	LIS3DHTR	Digikey	$1.59	12	$19.06	N
Temperature Sensor (Thermometer)	MCP9808T-E/MS	Digikey	$1.27	12	$15.24	N

STM32 Microprocessor	STM32L412KBT6	Digikey	$3.95	4	$15.80	N
Thermal Fuses	AD55ABB	Digikey	$1.98	10	$19.82	N
Bridge (Diode) Rectifier	ABS10A-13	Digikey	$0.26	15	$3.92	N
Resistor	RMCF0805ZT0R00	ECE Shop	$0.10	5	$0.50	Y
Resistor	RK73H2ATTD3300F	ECE Shop	$0.10	5	$0.50	Y
Resistor	RMCF0805JT2K20	ECE Shop	$0.10	5	$0.50	Y
Resistor	RMCF0805JT2K20	ECE Shop	$0.10	10	$1.00	Y
Resistor	RMCF0805JG10K0	ECE Shop	$0.10	30	$3.00	Y
Resistor	RMCF0805JT100K	ECE Shop	$0.10	10	$1.00	Y
Capacitor	CL21F104ZAANNNC	ECE Shop	$0.10	8	$0.80	Y
Capacitor	GRM21BR61H106ME 43L	ECE Shop	$0.34	8	$2.72	Y
MOSFET	IRFR18N15DPBF	ECE Shop	$1.07	5	$5.35	Y
Connector	PPTC041LFBN-RC	ECE Shop	$0.46	12	$5.52	Y
PCBs & Stencils		PCBWay	$20	2	$40	N

Table F.1 All utilized parts and their prices.