Bluetooth Burner

ECE 445 Final Report - Spring 2024

Shaunak Fadnis, Varun Kowdle, Navin Ranganathan
Project #63

Professor Viktor Gruev
Teaching Assistant Zicheng Ma

Abstract

This report goes through the product that we built for our ECE 445 project. Our project is the Bluetooth Burner, and it is used to increase or maintain the temperature of the users' beverage of choice at multiple different settings. It also incorporates a mobile application to allow user-friendly control and monitoring of the heat levels of the pad. This report addresses our team's design choices, verifications, cost analysis and references. We were overall able to accomplish two of our three high-level requirements on the breadboard.

Contents

1 Introduction

1.1 Problem

Each day, millions of people drink warm coffee, tea, or soup. However, one common challenge faced is maintaining the ideal temperature over time, especially in large sit-down environments or during extended periods of consumption. Moreover, traditional methods like reheating in microwaves can degrade the quality of the drink or food, while passive insulating containers often fail to maintain the desired temperature for long. The repeated process of reheating can be time-consuming and energy-inefficient, making it a less than ideal solution for both home and office settings. This results in a compromised experience, as the taste derived from hot beverages and soups is significantly tied to their warmth. Currently, the beverage warmer market only focuses on two main targets, high-end and low-end, leaving no options for a quality, yet affordable warmer.

1.2 Solution

To address this issue, we propose to make a heating pad with Bluetooth capabilities so that users can adjust temperature to three settings. This allows users to change the heating pad to their ideal temperature to the requirements of the beverage or soup. Integration of Bluetooth allows for a convenient and remote control, enabling users to adjust settings directly from their smartphones. The pad will be energy-efficient and durable, suitable for both home and office use. It will also feature a smart timer function for automatic temperature adjustments or shutdown, promoting convenience and energy savings. Our design prioritizes eco-friendliness by using sustainable materials, aligning with consumer demand for environmentally responsible products. The surface is designed to accommodate a variety of cup and bowl sizes, making it versatile for different beverages and soups. A companion app will offer control over the pad and provide hydration reminders and optimal consumption prompts. Our solution aims to bridge the market gap with a quality, affordable product, enhancing the hot beverage and soup experience.

1.3 High Level Requirements

1. The heating pad should have temperature capabilities of $30 - 60^o$ C for heating and reach at least -10^oC for cooling.
2. The infrared sensor observing the heating pad should be able to identify pad temperature within 1^oC
3. The device should communicate and receive information such as change in temperature via an iPhone app in a range of at least 10 - 20 meters.

1.4 Visual Aid

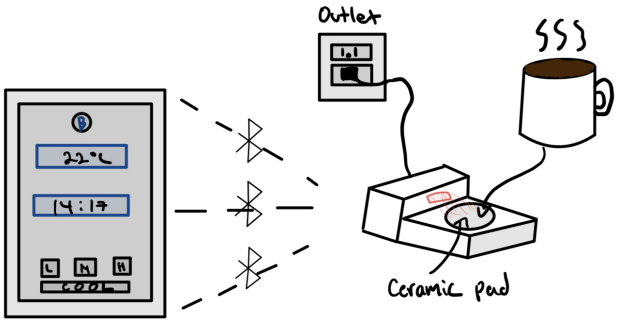

Figure 1: Visual Aid

1.5 Block Diagram

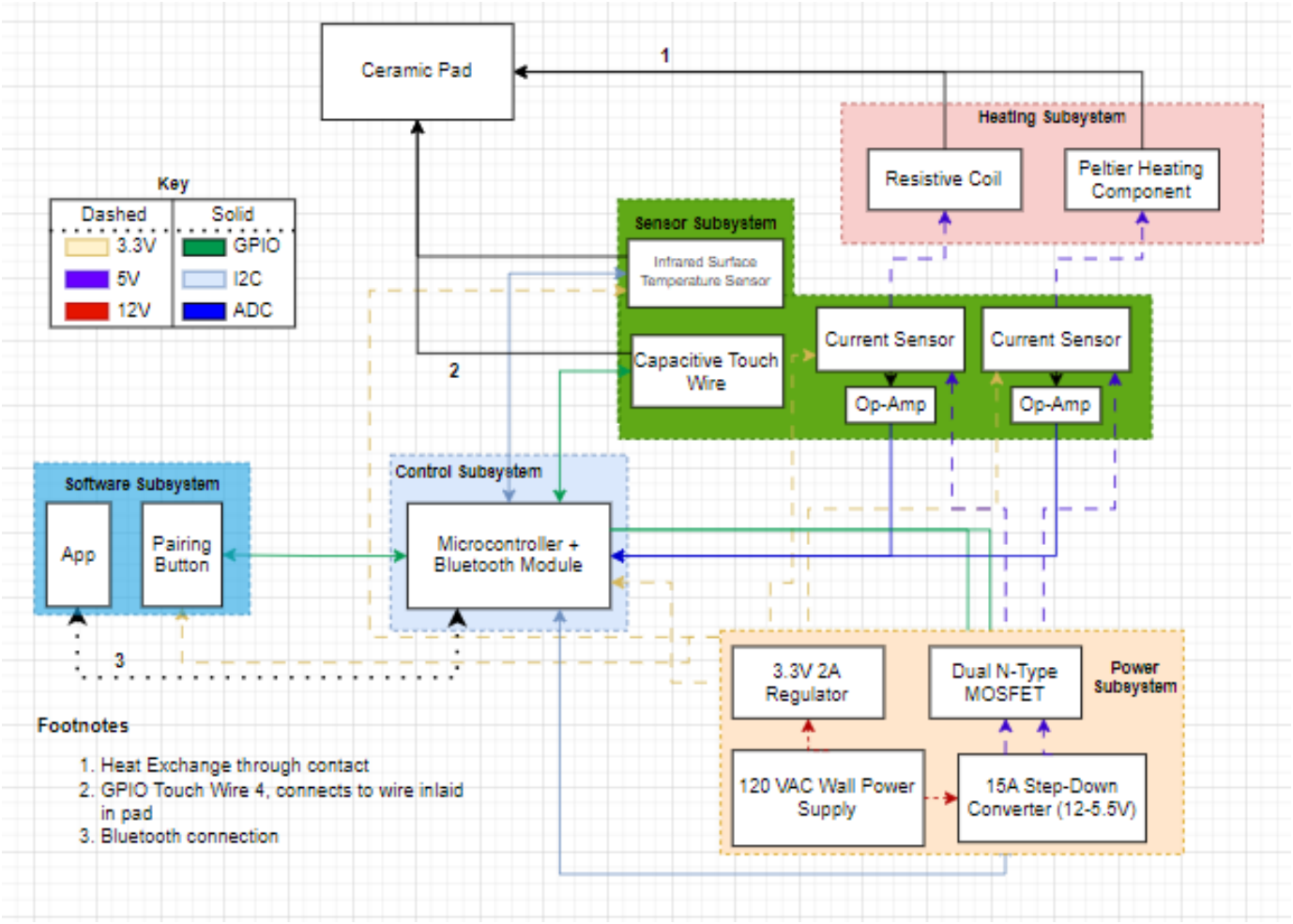

Figure 2: Block Diagram

*Some of the schematics may not have resistor or capacitor values, these can be found in the datasheets for the ACS711EX Current Sensor [1] , MCP6001U Op-Amp [2], TPS540A50 Buck Converter [3], AP62200WU-7 3.3V Converter [4], and ESP32-WROOM-32E [5]

1.6 Subsystem Overviews

1.6.1 Control

The control subsystem serves as the central processing unit of the Bluetooth Burner, powered by an ESP32-WROOM-32E-H4 equipped with a Bluetooth module. Its primary functions include controlling the heating/cooling devices, monitoring the burner's status, and facilitating communication with the user's device via Bluetooth. It receives input from the sensor subsystem, which includes an infrared temperature sensor and a current sensor, the former using I2C connection while the latter uses ADC pins. Additionally, the control subsystem links with the software subsystem, which allows users to interact with the burner through a dedicated application. The ESP32 also connects to a button that initiates the Bluetooth pairing process. This subsystem is also responsible for operating the power relay, which switches the MOSFETS responsible for the Peltier module and the Resistive Coil.

1.6.2 Heating

The heating subsystem is responsible for heating and cooling the pad upon which our user can place items. The heating device we are using is a Resistive Coil using Nichrome wire. Our cooling device is the CP40236 Peltier Module. The two heating elements will be powered using 5V from our 12V-5V Step-Down Buck Converter. The ESP32 is in control of turning the heating elements on/off. The pad we are using will be made of ceramic and go on top of the heating subsystem. The IR Temperature sensor will relay the surface temperature of the pad back to the ESP32 so it can better control the heating elements.

1.6.3 Sensor

The Sensor Subsystem serves to provide real-time monitoring and data collection regarding the operational status of the burner. It includes two current sensors, a capacitive touch sensor, and an infrared surface temperature sensor. These sensors that comprise the subsystem are essential in ensuring the safety, efficiency, and effectiveness of the heating process, each serving a unique purpose. The two current sensors are connected to the power lines for the heating subsystem to monitor the power output, while the GPIO capacitive touch pin helps to detect human contact through the inlaid wire on the ceramic pad, and the infrared surface temperature sensor provides temperature control to prevent overheating. The sensor subsystem communicates relevant information to the controller and power subsystems using I2C for the IR sensors, and GPIO-ADC pins for the MOSFETS and the current sensors respectively. In the case of the ADC pins connected to the current sensors, the output of our current sensors goes through an amplifier to take it from a range of 0-110mV to a readable range on the ADC pin.

1.6.4 Power

This subsystem is responsible for powering the control subsystem, the heating subsystem and the control subsystem. The Power subsystem takes in 12V from an AC/DC wall-adapter. This 12V is stepped down by the TPS540A50 Buck Converter to 5V to be used by the heating subsystem. The 12V is also Vin for the 3.3V AP62200WU Converter that powers the Control and Sensor subsystems. The 5V for the heating subsystem is also controlled by a Dual P-type MOSFET with the gate's connected to GPIO pins on our ESP32. These MOSFETS are able to provide our desired power to the heating subsystem and has a P_{max} of $48W$.

1.6.5 Software

The software subsystem of the Bluetooth Burner serves as the graphical user interface for interacting with the device, providing a seamless and intuitive experience. Moreover, the primary function of the software subsystem is to enable users to initiate the Bluetooth pairing process, adjust temperature settings, and monitor the status of their Bluetooth Burner in real time. At the heart of the subsystem lies the bluetooth communication facilitated by the ESP32's onboard module, which bridges the gap between the user's commands and the device's operational logic in the control subsystem. By using ESP32 communication protocols, the software subsystem is able to send and receive data, which will utilized for visual indicators of the device's current state and providing feedback on system statuses as detected by various subsystem.

2 Design

2.1 Physical Design

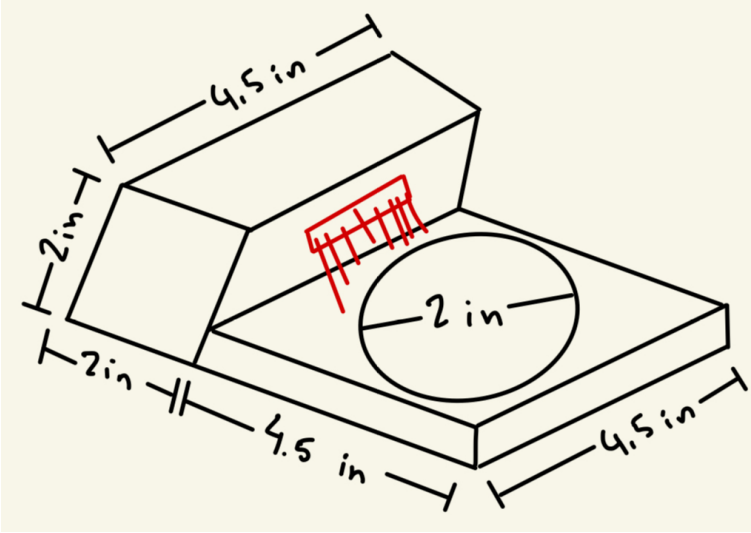

Figure 3: Physical Design - Side View Sketch

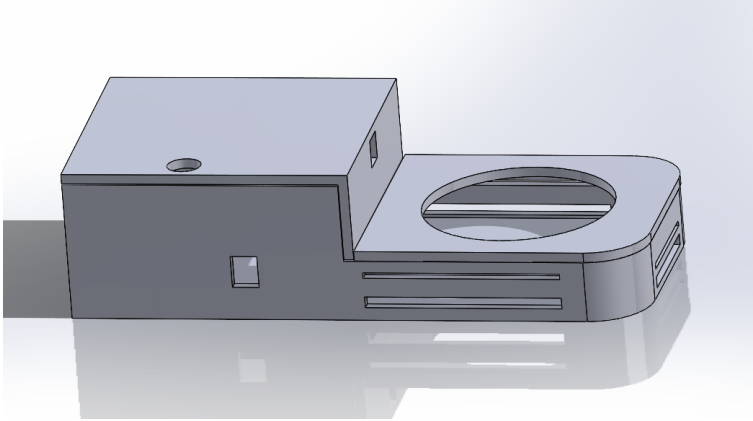

Figure 4: Physical Enclosure Solidworks

The physical design of our device is an enclosure built around our heating subsystem and ceramic pad. The ceramic pad has a diameter of 76cm while the opening on our enclosure has a diameter of 68cm so that the pad is slightly raised. The opening is for our heating element to be housed while the PCB is in the back of the device. The latter half of the enclosure is also taller compared to the front half and this is due to our IR Temp Sensor. As we need an angle where the IR's Field-of-View is on the pad, we needed it higher than the pad. We also wanted to avoid the ambient temperature nearer to the heating subsystem as this could have altered some of our readings. The enclosure was 3D-printed using PLA and is very durable. Also as the temperature of our pad doesn't go too high, the material will withstand the heat.

2.2 Control Subsystem

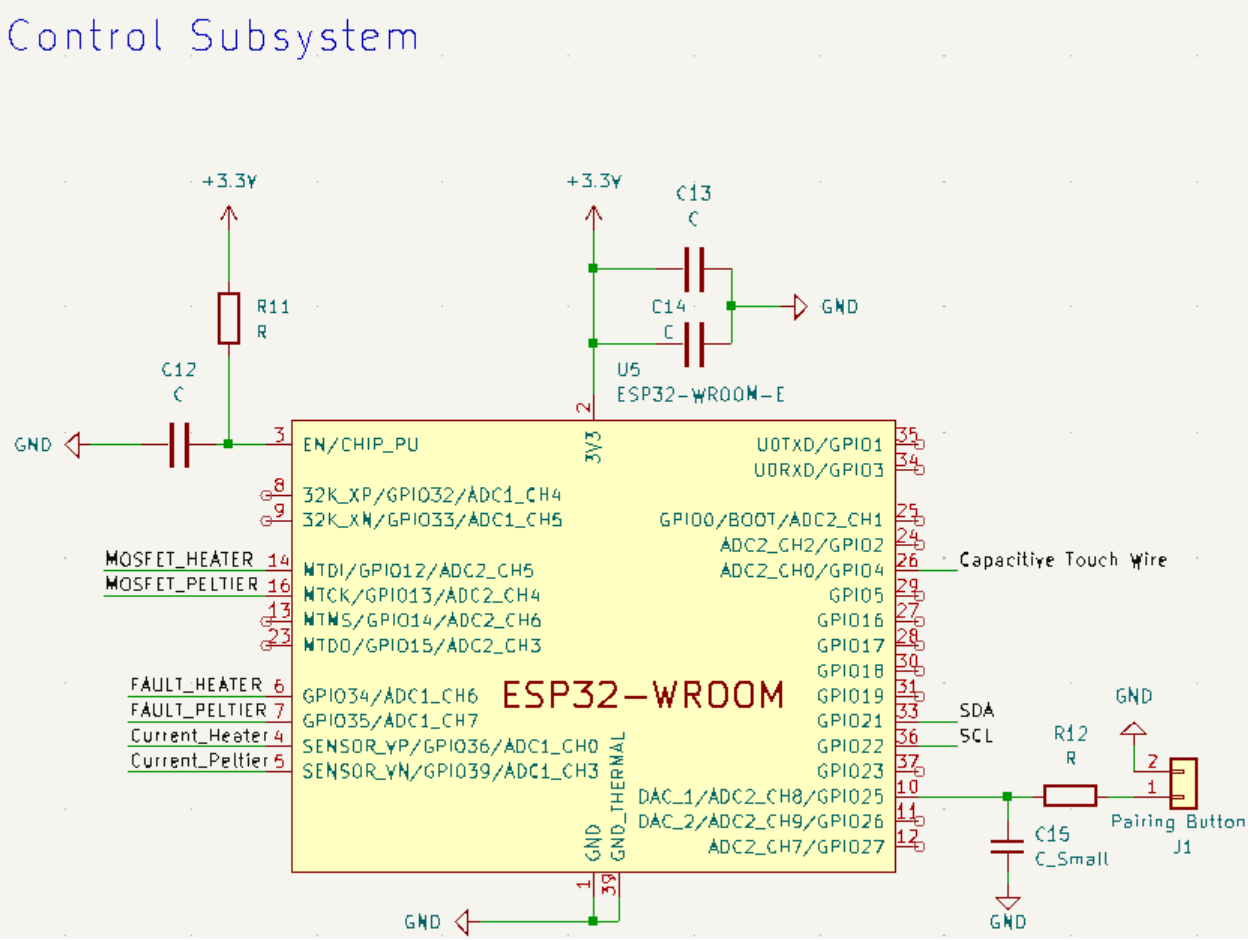

Figure 5: Control Schematic

2.2.1 Design Decisions

For the control subsystem, there were a number of design decisions made in order to reach our requirements. For starters, the ESP32 module was chosen as the main control unit as it provided both Wifi and Bluetooth connectivity. With that being said, we also had to choose between two types of bluetooth control: Bluetooth Low Energy (BLE) or Bluetooth Classic. Despite BLE having lower power consumption and longer range, we opted to go with Bluetooth Classic as it provided higher data rates, typically up to 3 Mbps and a much more stable Bluetooth connection.

Another design decision that we had to make was the implementation of power width modulation (PWM). Initially, we had designed our control subsystem to control our heating subsystem by gating it with a P-Type mosfet. However, when testing power delivery, we noticed that too much power was being released into our heating subsystem, and simply having a P-Type mosfet gate was not enough. As a result, we decided to implement PWM duty cycle calculations in our control subsystem, which would then be directed to our P-Type mosfet. This allowed us to control and provide a range of voltage from 4.3 to 5V to our heating subsystem.

2.2.2 Verifications

To Test and verify the control subsystems requirements, we followed the steps outlined in Appendix B, Table 4: Control Subsystem Requirements & Verifications. For starters, to test Bluetooth connectivity, we paired the Bluetooth Burner with a mobile device and checked the signal strength. This process was repeated several times at different locations to ensure that Bluetooth connection was still valid up to 20 meters.

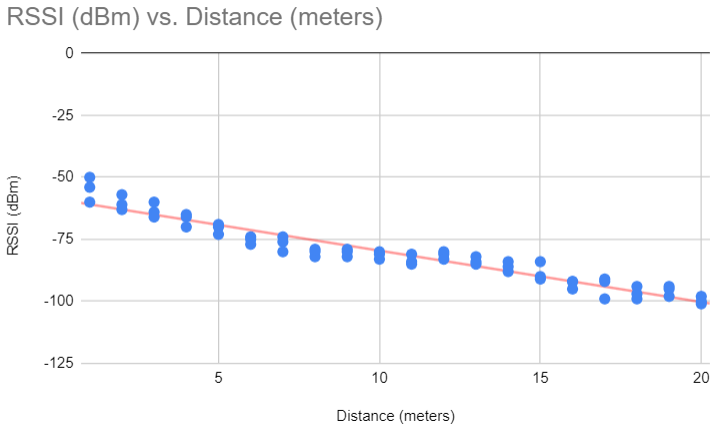

Figure 6: Received Signal Strength Indicator vs Distance

Figure 6 above describes the received signal strength indicator (RSSI) from the ESP32 to a mobile device. Over 20 meters, the RSSI value falls in between -50 and -100 db, indicating good to fair signal strength. This falls in line with our high level and subsystem requirement of having Bluetooth connection up to 20 meters.

Baud Rate	Transmission Succesful
300	
1200	
9600	
19200	
31520	
115200	

Figure 7: Baud Rate Testing

To test and verify data transmission, we ran a our program over various baud rates indicated in the Aruduino IDE. From the findings, we noticed that our program worked at various baud rates ranging from 300 to 115200, with 115200 being the fastest data transmission rate. This verified another subsystem requirement for data transmission from control subsystem to the software subsystem.

Finally, the last requirement included controlling power relay between the control and power subsystem. To implement this feature we used PWM, and verified our implementation by testing the voltage of the power subsystem at three different temperature settings. From the findings, we noted that low setting received 3.68V, medium setting received 4.21V, and high/cool settings received 5V. These numbers fall in line with our duty cycle expectations.

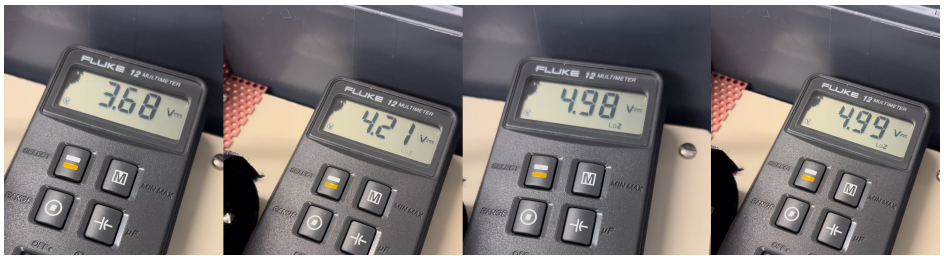

Figure 8: Voltage readings from Low, Medium, High, Cool duty cycles

2.3 Heating Subsystem

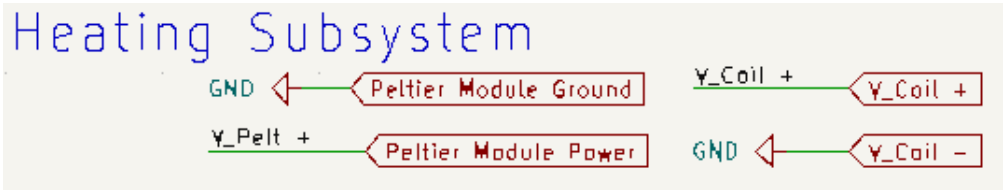

Figure 9: Heating Schematic

2.3.1 Design Decisions

For our heating subsystem, we had to make some changes in our design from our initial proposal. For one, we had initially planned on using the CP402036 Peltier module to heat our pad but upon further research and meetings with TAs we decided to make it a cooling element and add a resistive coil. Due to how peltier modules work, this was an easy change as we only had to change the side touching our ceramic pad to be the cold side. This also meant adding a heatsink so that the hot side was not just in open air, but due to the small size of the peltier module sourcing a heatsink wasn't a problem. The peltier module also seemed to fit our requirements for cooling, given it has a $\Delta T = 66$ for $T_h = 27^oC$ [6] it would be able to reach -10^oC with the proper electrical connections.

For the coil, we decided to use Nichrome wire as it has high thermal conductivity and a low enough resistance per meter so that we didn't need an exorbitant amount of current. These calculations are further explained in 2.5.1. The Nichrome wire we used was also a flat wire since we wanted as much surface area in contact with our ceramic pad. It also made mounting our coil much easier as well, with more direct areas to apply a clamping mechanism.

2.3.2 Verifications

Our requirements followed those in Appendix B, Table 3. To test the heating coil, we initially connected the coil to a lab power supply. As shown in Figure 10, the coil's voltage draw bottlenecks at around 3.3V, a large disparity from our 5V converter output.

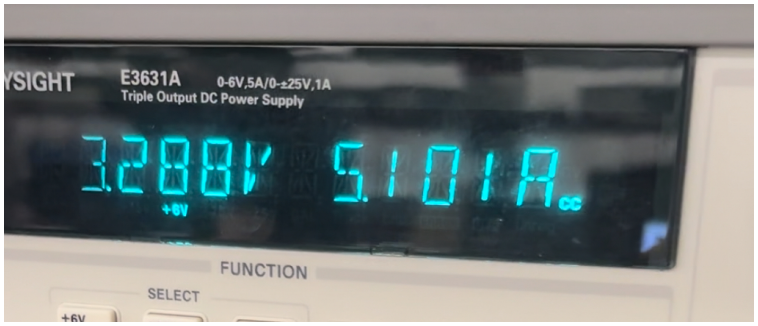

Figure 10: Power Supply Readings Connected to Heating Coil

Furthermore, when looking at the coil's temperature directly we found that it was only around 120^0 F or 49^o C as shown in Figure 11. However, we had multiple problems with out 5V SMD converter which meant we were unable to test our coil with its maximum power output. This will be further discussed in section 2.5.2. As such, our requirement regarding the heat output of the coil was based on the sub-optimal readings from the power supply. Using the temperature of the coil at 120^o F and the room temperature of 71^o F, we

Figure 11: Coil Temperature Reading

were able to use the specific heat capacity of Nichrome[7] in order to determine the energy output.

$$c \text{ (Specific Heat Capacity of Nichrome)} = 0.480 J/g -^o C \quad 10.4g \text{ Flat Nichrome Wire}$$

$$\Delta T = T_i - T_f = 88\,^\circ\text{C} - 40\,^\circ\text{C} = 48\,^\circ\text{C} \tag{1}$$

$$Q = mc\Delta T = (10.4\,\text{g}) \times (0.48\,\text{J/g}^\circ\text{C}) \times (48\,^\circ\text{C}) \tag{2}$$

$$= 239.616\,\text{J}$$

Using Equations (1) & (2) we can see that the coil is still putting off almost 240J, however, this didn't seem to affect the pad, meaning some of our early calculations may have overlooked other methods of heat dissipation.

Similar to our testing of the coil, we had to use the power supply for our peltier module. This yielded some contradictory results to our expectations. For one, the peltier module at 5V from the power supply got hotter than the coil, on both the hot and cold sides. Our heatsink was unable to properly dissipate the heat being sent to the hot side of the peltier module, making the entire device heat up. This actually meant we could physically feel the heat from the pad with our touch. This meant our third requirement of reaching -10^o C would not be feasible, and we weren't able to source a better heatsink/cooling system on time.

2.4 Sensor Subsystem

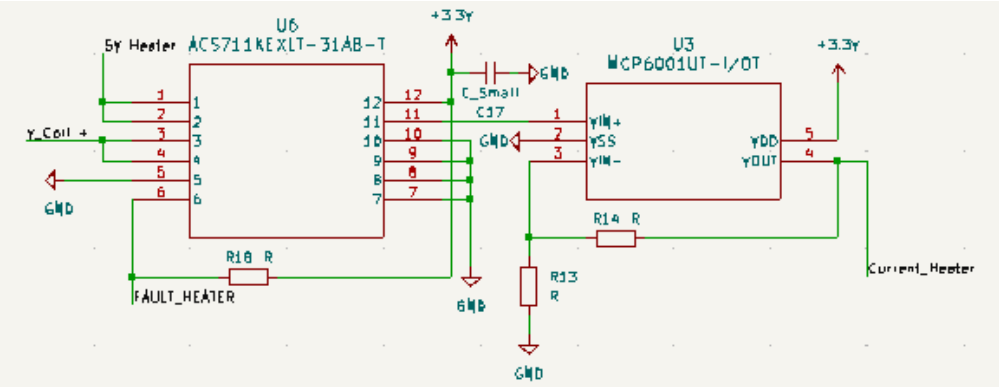

Figure 12: Coil Current Sensor Schematic (Replicated for Peltier Module)

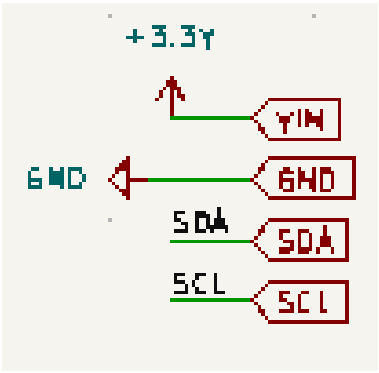

Figure 13: IR Temp. Sensor Schematic

2.4.1 Design Decisions

For our sensor subsystem, we chose an infrared sensor to accurately measure the temperature that the ceramic pad was relaying and the live temperature of our beverage on the ceramic pad. It was also an efficient way to monitor via Bluetooth, and was important in emphasizing the safety of our end users. We also chose to use a capacitive touch wire connected to our ESP32 GPIO pin because we wanted to have a kill-switch that cuts the power to our heating subsystem in the event that a user makes contact with the ceramic pad. We found this to be a viable option due to the inherent capacitive touch pin on the ESP32, allowing for a seamless connection to monitor contact past a certain threshold. We also wanted to implement current sensors that directly communicate with our control subsystem as well, but were unable to incorporate them with our breadboard design.

2.4.2 Verifications

For the sensor subsystem, some of the components were not fully able to be tested because they were small QFN that needed to be tested on PCB. Nevertheless, some of the requirements for the requirements for the sensor subsystem were able to be tested such as IR sensor and capacitive touch. To do this, steps from Appendix B: Table 6 were followed.

Figure 14 shows the sensor's console output. Temperatures were recorded over a series of time, and then compared with a thermometer, like in Figure 11. The temperatures noted were within 0.1 degree Celsius.

```
Sent Value: Ambient: 20.47 C, Object: 20.39 C
Sent Value: Ambient: 20.51 C, Object: 20.61 C
Sent Value: Ambient: 20.79 C, Object: 29.45 C
Sent Value: Ambient: 21.25 C, Object: 28.77 C
Sent Value: Ambient: 21.35 C, Object: 22.19 C
Sent Value: Ambient: 21.55 C, Object: 31.49 C
Sent Value: Ambient: 21.89 C, Object: 29.77 C
Sent Value: Ambient: 22.57 C, Object: 27.91 C
Sent Value: Ambient: 22.47 C, Object: 21.63 C
```

Figure 14: Arduino IDE Temperature Readings

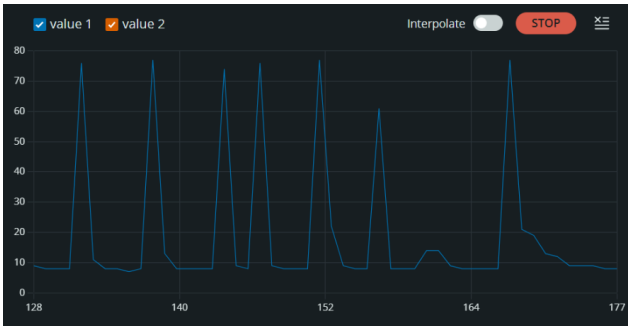

Figure 15: Serial Plotter with Capacitive Touch Readings

In addition to temperature, another part of the of the sensor subsystem that was able to be tested and verified is capacitive touch sensor. Figure 15 is the serial plot from the capacitive touch sensor, where the peaks indicate that excessive human contact has been reached above a certain threshold. This falls inline with our requirement for safety.

2.5 Power Subsystem

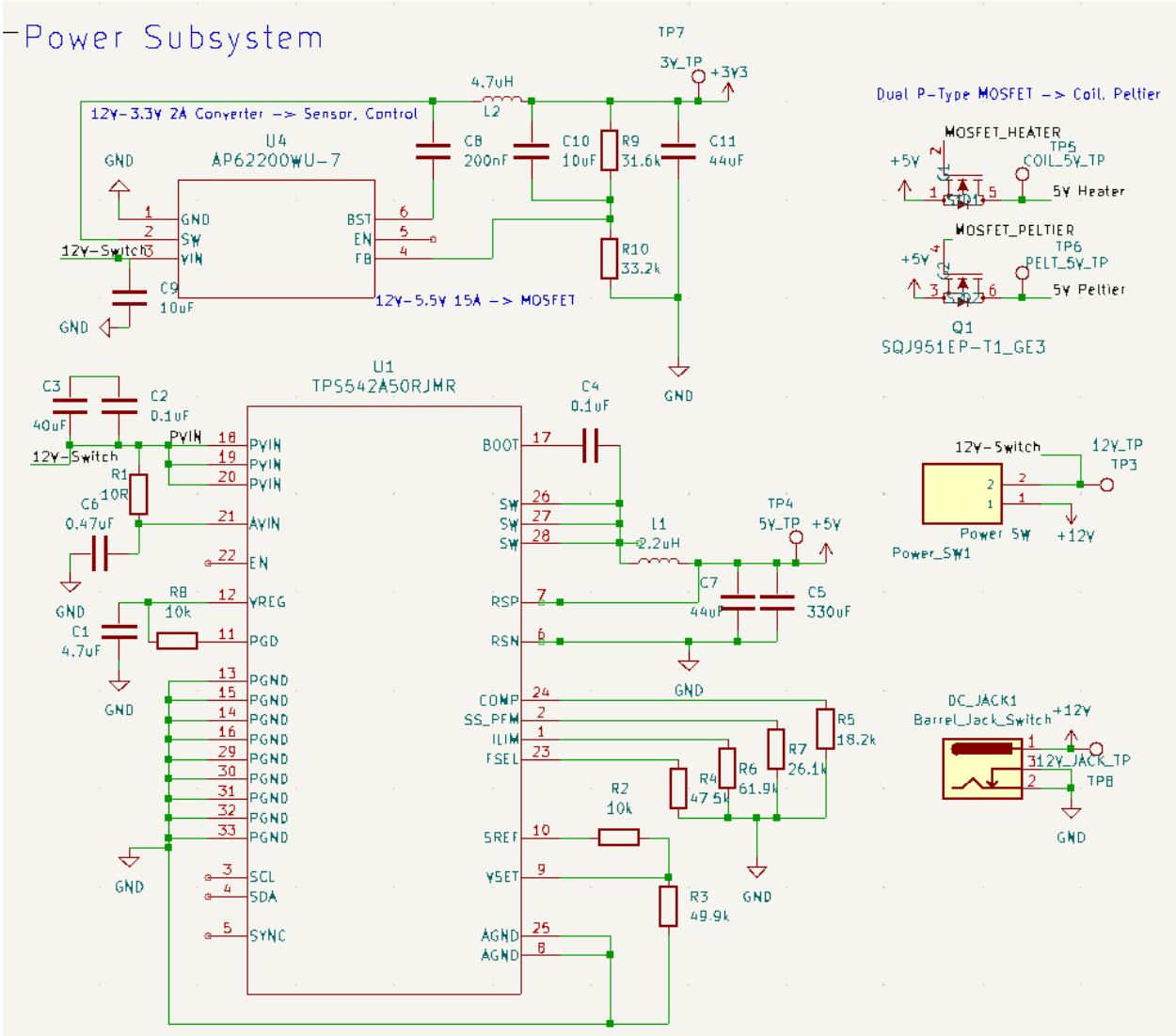

Figure 16: Power Schematic

2.5.1 Design Decisions

The power subsystem was our most important subsystem as it was responsible for powering our heating elements and our feedback control system. For powering the heating subsystem, we required a high amount of current which led us to using the TPS542A50RJMR 5V buck converter. This converter allowed for up to 15A and with our 12V 10A Wall Adapter, we followed the following calculations to select this IC:

$$P = 50W/m_{coil} \quad R_{coil} = 5\Omega/m_{coil} \quad V_{tp} = 5V$$
$$I = \sqrt{\frac{P}{R}} = \sqrt{\frac{50W/m}{5\Omega/m}} = 5A \tag{3}$$

Equation (3) shows our calculation for the minimum current we needed for the heating subsystem. We found that 50W/m was a reasonable estimate for the heat we needed, especially as our converter could handle double the current present.

Given how much power we were passing into the heating subsystem, we needed a robust method of turning it on/off. To do this we used the SQJ951EP Dual P-type MOSFET. This MOSFET can handle up to 50W and had a V_{gs} range of -1.5V to -2.5V[8] which works with our ESP32 given $V_{Source} = 5V$ and $V_{Gate} = 3.3V_{ESP32}$ so our V_{gs} would be around -1.7V, within the threshold.

The other main component was our 3.3V converter which was used in order to power our ESP32 and our Sensor subsystem. The biggest power draw was going to be from our ESP32's Bluetooth capabilities, given it could get up to as high as 600mA at any point [9]. However, as the other sensors had significantly lower current draws we opted for a 2A 3.3V converter, the AP62200WU from Diodes Inc.

2.5.2 Verifications

For verifying the requirements of the power subsystem we followed Table 5. However, due to issues with our PCB implementation mentioned in section 2.3.2, many of our verification tests resulted in failure. To begin with the 3.3V converter we used was damaged and shorted during our testing. As such rather than outputting 3.3V, it first outputted nothing in the damaged circumstance, and then output 1.6V in the shorted version. In the case of the 5V output, we were simply unable to test it on the PCB as our 5V converter was a QFN package with very small pads, meaning they bridged every time we used the reflow oven. However, we were able to get the 5V converter working on a breadboard, with it powering multiple mosfets. This system is referenced in our verifications for the control system, specifically Figure 8. The 3.3V converter we used on the breadboard, the LM3117T had mixed results. This converter read 3.3V on the voltimeter in our lab kit but when connected as the sole power supply for the ESP32 and IR Sensor, it was unable to power the IR sensor. This leads us to the current output verification. As we were unable to use the high current TPS542A50RJMR chip, the L7805CV 5V Regulator specified a maximum output of 1.5A[], which did not meet our requirements. We did test and monitor the current using the lab bench equipment but found that similar to the power supply heating tests, the current limited the voltage output for the coil.

For verifying the MOSFETS, we used the multimeter to check the voltage when the MOSFET's gate is driven high, or closed, compared to driven low, or on. This worked correctly with our multimeter going from 0V to 4.99V in a moment, with their outputs being taken from a video and shown in Figure 17. From this we were also able to test the PWM capabilities for the Control Subsystem as shown in Figure 8.

Figure 17: Left: Gate Closed Output
Right: Gate Opened Output

2.6 Software Subsystem

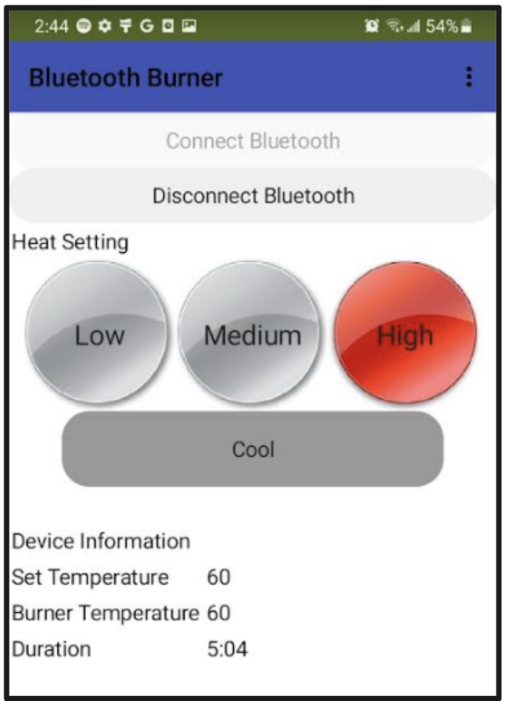

Figure 18: Software Visual

2.6.1 Design Decisions

The software subsystem is broken down into two components: the client and the server. In this case, the ESP32 will serve as a server and a mobile device will serve as the client. Each component will have specific responsibilities. For the server side, we will be using Arduino BLE modules to communicate with our client side code code. Data will be sent through the bluetooth channel configured on both server and client. On the client side, users will have to connect to bluetooth via the iOS interface; however, once a device is connected, the app will determine if its the ESP32 by a set UUID and then be able to transmit data. The client application is developed in MIT App Inventor, using the CoreBluetooth module's CBCentralManager to communicate with the ESP32. Additionally, the app's UI will consist of various event listeners to be able to remotely control the ESP32.

2.6.2 Verifications

To verify that our software application was able to communicate with our ESP32 and accurately change the temperature output, we ran 10+ trials and made sure that it was within 1% accuracy of the actual temperature desired, and we cross-checked with our personal thermometer, the thermometer we used can be referenced in Figure 11. To test that we could connect to Bluetooth, we initialized device scans from various distances from the ESP32, to make sure we could accurately connect and disconnect from our desired range of 10-20 meters. This was also included as a verification in the control subsystem, Figure 6. To make verify our pairing button, we made sure to activate pairing mode on our Android test device and see if the advertised ESP32 would show in the device display within our desired time frame. This ended up being verified using our ESP32 devkit as well, with there being a blue LED that would light up upon pairing and connection.

3 Cost & Schedule

3.1 Cost Analysis

3.1.1 Estimated Cost of Development

After doing some research into what the average UIUC graduate from Computer Engineering would make annually, we established that the wage of each team member would be about $52 per hour. We estimated that each team member did about 15 hours of work per week for the 8 weeks of project development, coming out to a cost of $6,240 per teammate. After multiplying this by 2.5, we have a cost of $15,600 per teammate, or $46,800 overall for labor costs.

3.1.2 Components Breakdown

Component (Link)	Vendor	Quantity	Total Cost
ESP32-WROOM-32E-H4	DigiKey	1	$2.68
Peltier Thermoelectric Module	DigiKey	1	$14.57
Digital IR Temperature Sensor	Amazon	1	$15.99
5.5V I2C Buck Converter	DigiKey	1	$5.46
Hall Effect Linear Current Sensor	DigiKey	2	$3.02
12V AC/DC Wall Converter	Amazon	1	$11.99
Dual P-Type MOSFET	DigiKey	1	$1.55
12V-3.3V Buck Converter	DigiKey	1	$0.41
Op-Amp	DigiKey	2	$0.60
MCP4018 Digital Potentiometer	DigiKey	1	$0.68

Table 1: Bill of Materials

3.1.3 Total Cost

When adding together our total labor cost ($46,800) with the total cost of our components broken down in the table (total components cost of $59.60), we compute a total cost of **$46,859.60** for our Bluetooth Burner project.

3.2 Schedule

Week	Major Deadlines	Shaunak	Varun	Navin
2/26	Design Review	Finalize PCB	Finalize PCB	Finalize PCB
3/4	First Round PCB Orders	Order Parts	Start front-end of application	start front-end of application
3/11	Spring Break	N/A	N/A	N/A
3/18	Second Round PCB Orders	Order PCB/start backend	Start backend for bluetooth interface	Start backend for bluetooth interface
3/25		Soldering	Soldering	Soldering
4/1		Test Heating System	Test Control System	Test Sensor System
4/8		Design Enclosure/Test Feedback loop with heating and control systems	Test bluetooth interface with control system	Test capacitive touch sensor
4/15	Mock Demo	Finish Unit testing/Begin full device testing	Finish Unit testing/Begin full device testing	Finish Unit testing/Begin full device testing
4/22	Final Demo	Last minute changes	Last minute changes	Last minute changes
4/29	Final Paper	Finalize Paper	Finalize Paper	Finalize Paper

Table 2: Weekly Schedule

4 Conclusion

4.1 Accomplishments

Our team found several successes in this project throughout the course of the semester. We were able to fully design and develop our mobile application with a user-friendly interface and read and displayed the pad and object temperatures accurately from our sensor subsystem. We were also able to create a ceramic pad for efficient and safe beverage heating and an enclosure design to house our subsystem components. Our power subsystem also had successes in driving regulated voltage to our MOSFETS using PWM switching for temperature control with our low, medium, high, and cooling settings. Within our control subsystem, we were also able to fully instantiate the Bluetooth advertising and connection protocols, to connect devices from a range of 20 meters and transmit data to and from our IR sensor. With our heating subsystem, we were able to successfully integrate our nichrome wire for heating and peltier module for cooling using thermal paste and electrical tape. All in all, we were able to complete all of our proposed subsystems using a breadboard.

4.2 Uncertainties

Despite the accomplishments we were able to reach throughout the semester, we did have a few uncertainties with regards to our project. Our biggest issues arose with the PCB, as we had difficulty getting it to work with the minute size of our QFN packages and driving enough voltage to flash our ESP32 device. Some things we could have considered were to test and break our PCBs earlier rather than focusing on perfecting the PCB through schematics. Something else we could have done was incorporate current amplifiers to get more voltage into our heating subsystem, as this was a failing point of our project. Another issue we later had was cooling using the peltier module, as we found that it was actually more resourceful as a heating mechanism rather than the nichrome wire. Perhaps had we purchased and tested these components earlier on, we could have discovered this and abandoned the difficult coil implementation.

4.3 Future Work

Our team has considered some ways to expand upon our current project proposal for the future. We wanted to potentially incorporate a broader range of temperature heating, as our current product as it stands has a range of $30 - 60^o$ C for heating. We also wanted to consider increasing the surface area of our ceramic pad so that we can provide extended heating for larger beverage holders and potentially any other food or drink items. Our current Bluetooth Burner does not have a pairing button on the actual product, but rather in our mobile application, and this is also something we considered adding to our product so that Bluetooth would not constantly be advertised and potentially subsidize power consumption of our whole system. One last thing that we discussed was smart data logging and analytics, so that our end users may keep track of their temperature preferences.

4.4 Ethics & Safety

As our product was something that will be heated to substantial temperatures, we ensure to have ethical and safety precautions that could help avoid burning the users or other objects within a close vicinity. As the team developing the project, we made sure to also be aware of the precautionary measures required as we worked in the lab, and we made sure to be as careful and responsible as possible to avoid any misuses of any potentially harmful materials. Our product also contains a Bluetooth aspect, so we made sure to respect the user's privacy when connecting their device. As per the context of IEEE Code of Ethics I-2 [10], we made sure that users of our product understand the correct methods of usage of our product, specifically that it is not meant in any way, shape, or form to harm others or burn other items. Our team also understands that the heating pad may still be warm after it is turned off for a bit of time, and there will be a temperature limitation to mitigate risk of overheating.

Appendix

A - PCB Schematics & Layouts

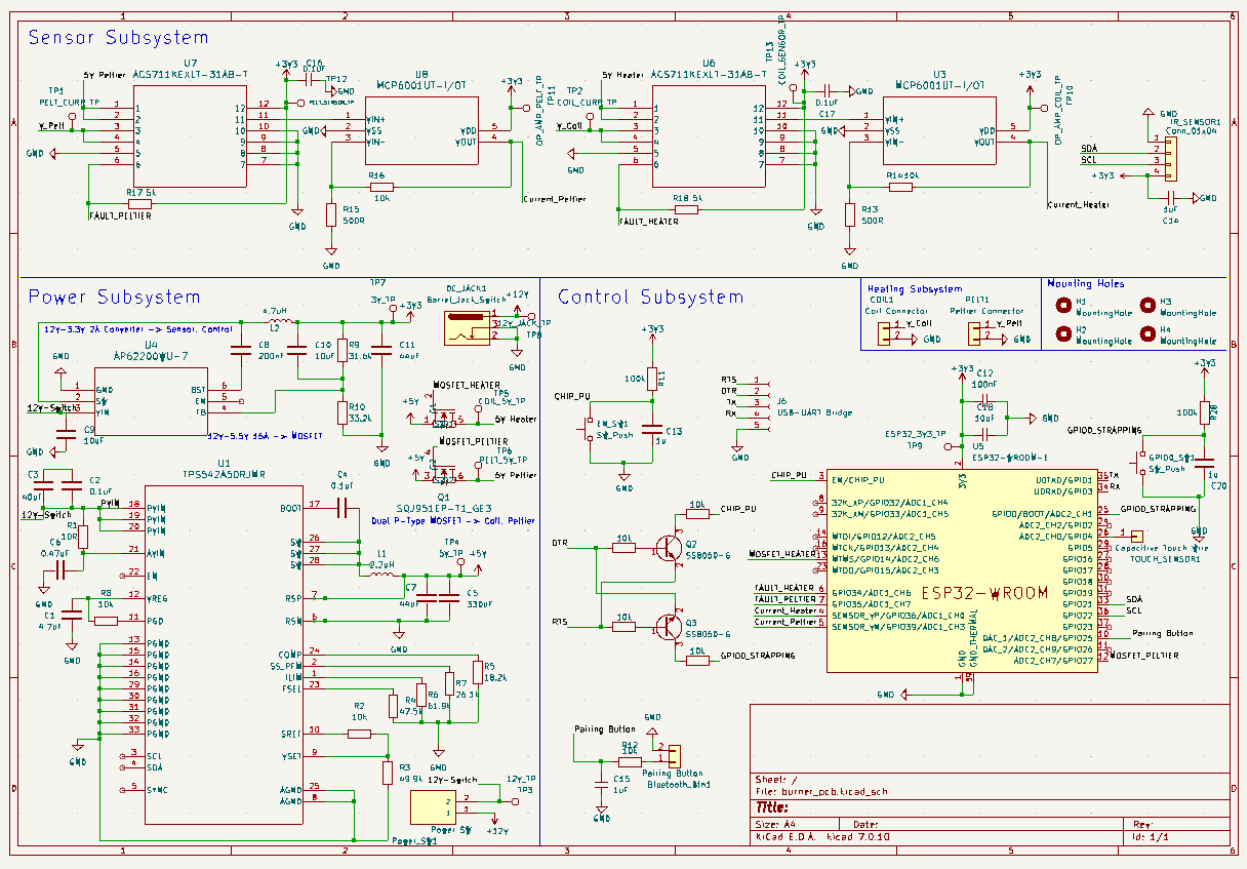

Figure 19: Full PCB Schematic

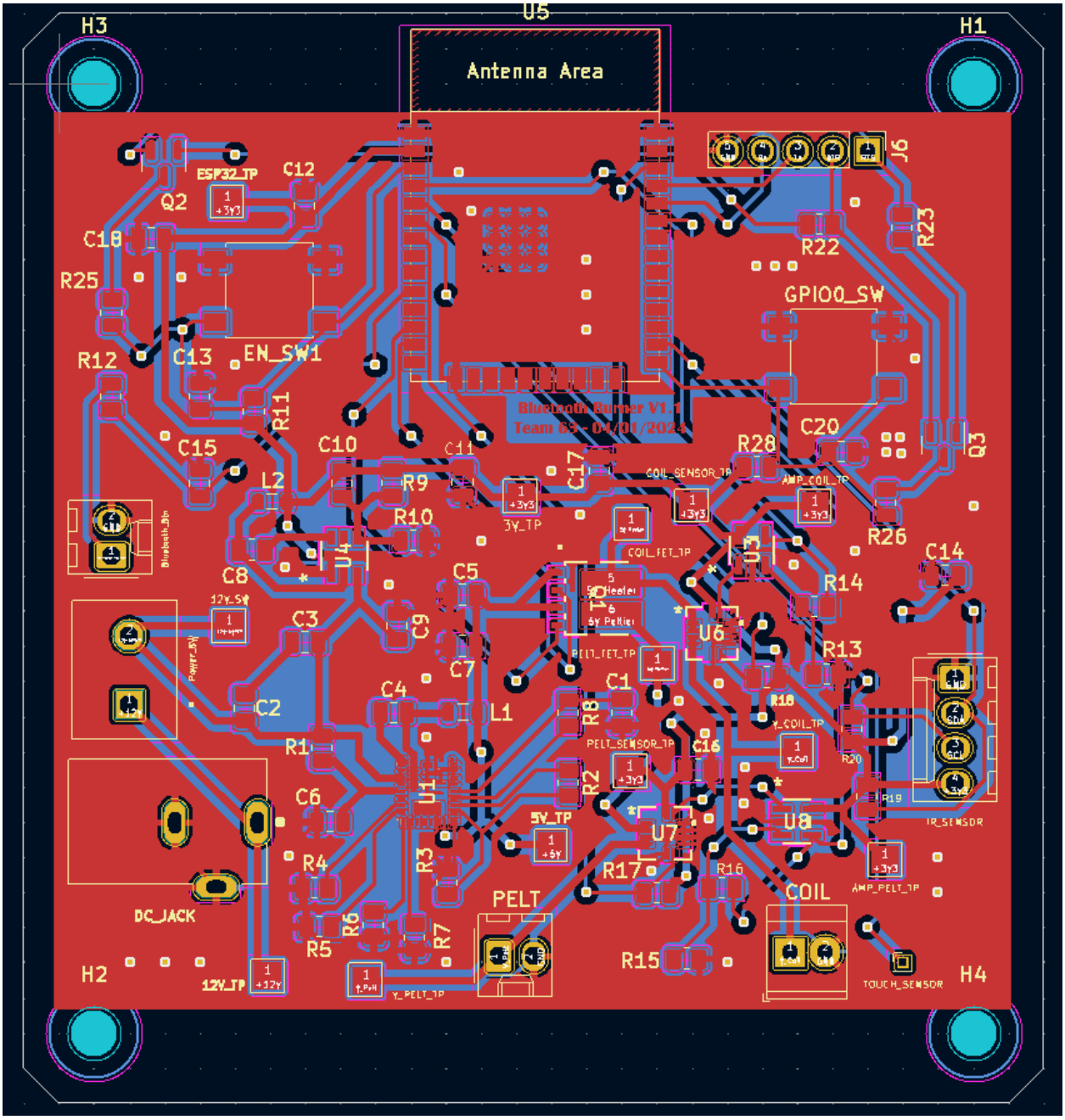

Figure 20: KICAD PCB Layout

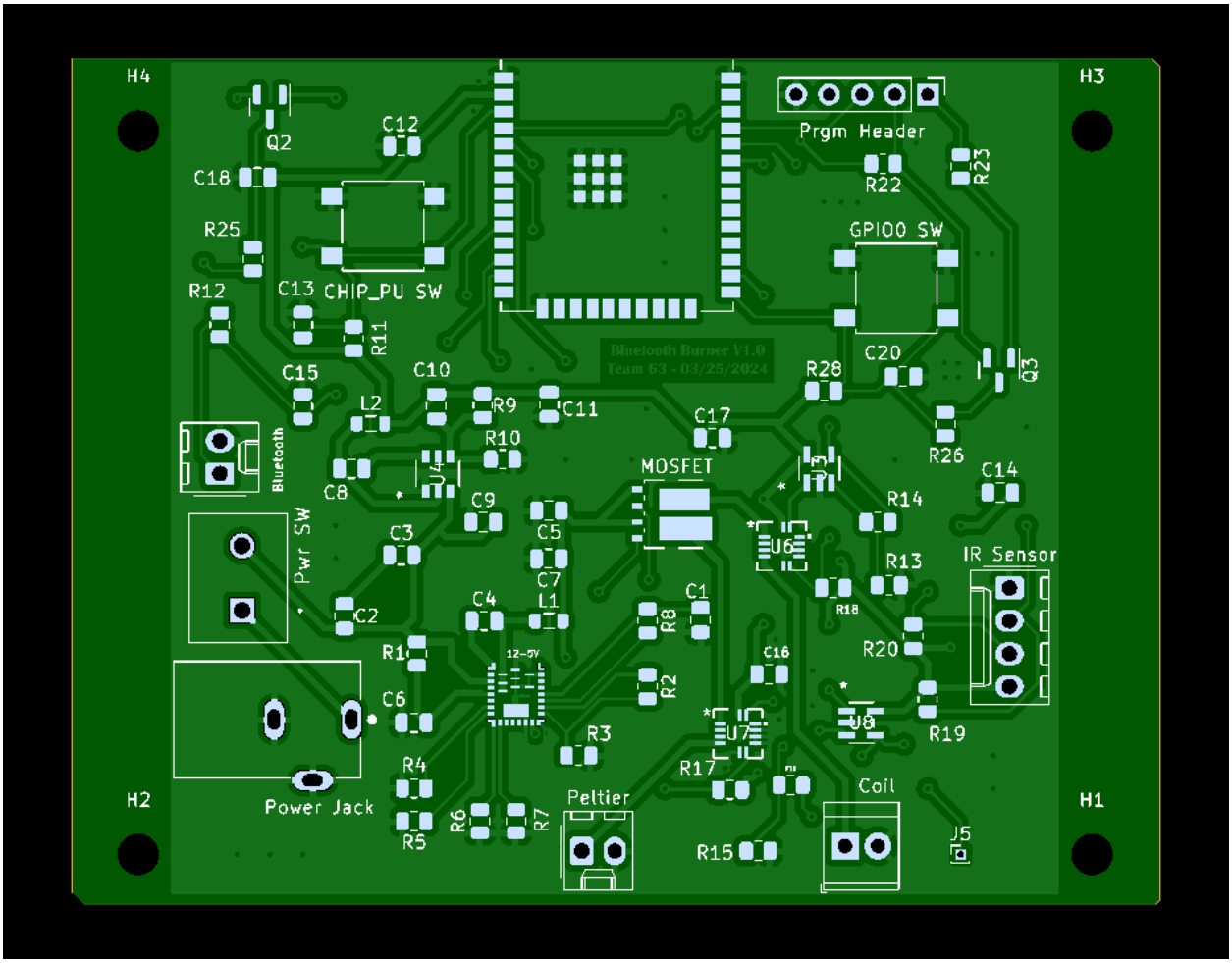

Figure 21: PCBWay Preview

B - Requirements & Verification Tables

Table 3: Heating Subsystem Requirements & Verifications

Requirements	**Verifications**
Resistive Coil must be able to go up to 65^oC.	1. Connect coil to power supply and begin heating 2. Measure temperature increaase until 65^oC.
Resistive Coil must be able to impart (heat energy transfer) at least 11J $\pm$ 2J per minute.	1. Fill a cup of 100 grams of room temperature water 2. Set the peltier module to the highest setting, and let the cup sit for a few minutes 3. Measure temperature and calculate the heat energy imparted $Q = m \times c \times \Delta T$ to verify that it falls within the required range of 11J $\pm$ 2J.[11]
Peltier module must have a lower-bound temperature for T_C (cold-side temperature) of -10^oC.	1. 2. Fill a cup of 100 grams of room temperature water 3. Set the peltier module to the highest setting, and let the cup sit for a few minutes 4. Measure temperature to ensure that the lower bound is -10^oC utilizing the ceramic heat transfer [11]

Table 4: Control Subsystem Requirements & Verifications

Requirements	**Verifications**
The ESP32 must initiate the Bluetooth pairing process within 30 seconds upon user command through the pairing button, ensuring a quick and seamless connection with the user's device.	1. Load up iOS bluetooth connection 2. Attempt to pair with esp32 3. Repeat and record 20 tests and record average connection time
The ESP32 must be able to read data from the infrared sensor consistently via the I2C communication protocol. The ESP32 should also be able to send this data at various baud rates.	1. Connect to Bluetooth Burner 2. Run sensor to record temperature on surface 3. Send sensor data to mobile device 4. Repeat steps over various baud rates
The control subsystem should control power relay, enabling the device to toggle the heating in response to the app's commands and the sensor's feedback.	1. Connect to Bluetooth Burner 2. Set temperature to Low 3. Ensure that power read from the mosfet falls in line with power requirements 4. Repeat steps for Medium, High, Low, and Cool.

Table 5: Power Subsystem Requirements & Verifications

Requirements	**Verifications**
The power subsystem must provide 3.3V ± 0.1V to the control subsystem and sensor subsytem.	Conduct a precision voltage measurement using a calibrated digital multimeter. Verify the output is within 3.3V ± 0.1V under no-load and full-load conditions to test the regulator's stability and load regulation.
The power subsystem must provide 5V as output from the Step-Down Buck Converter.	Conduct a precision voltage measurement using a multimeter.
The power subsystem must provide up to 15A as output current from Buck Converter.	Check both circuit using multimeter and output from I2C data to determine output.
MOSFETS must be capable of switching to allow 5V to heating subsystem.	Test MOSFETS at low voltage, but above V_{gs} threshold voltage [3] to test ESP32 control over switch. Use multimeter to test connection from Buck Converter and then output after switching.

Table 6: Sensor Subsystem Requirements & Verifications

Requirements	**Verifications**
Infrared sensor must accurately measure the surface temperature of the heating path with a precision of ±1°C	1. Gather set of objects at different temperatures 2. Scan temperature via IR sensor to gather object temperature reading 3. Compare IR sensor reading with reading from a thermometer 4. Repeat steps over a series of different items and different temperature points
The current sensors should have an accuracy of ±0.5% of the measured value to ensure precise monitoring of electrical consumption and to detect any unusual current draw.	1. Connect current sensors to voltmeter 2. Supply various loads of current to current sensor 3. Read current and print to terminal 4. Compare terminal reading to voltmeter to check for accuracy
The GPIO capacitive touch wire should make sure to respond within 0.1-0.5 seconds of a human touch to protect users from any heat damage	1. Connect capacitance touch wire to ESP32 2. Open serial plotter to see touch wire data being read by the ESP32 3. Make sure copper wire is not hot, and touch the wire with finger 4. Watch serial plotter spikes indicating capacitance touch

Table 7: Software Subsystem Requirements & Verifications

Requirement	**Verifications**
Software subsystem must be able to communicate with the control subsystem to change the temperature to three pre-set settings, and a single cooling, with a precision of 1^oC for each setting.	1. Initialize communication protocol with the control subsystem. 2. Send command to retrieve current temperature setting. 3. Display the retrieved temperature to the user. 4. Provide interface for user to select one of three pre-set temperatures or a cooling option. 5. On user selection, send command to update the temperature setting with a precision of $\pm 1°C$. 6. Confirm the change and display updated setting to the user.
Phone app must be able to connect to device within 10 - 20 meters of it	1. Ensure the phone app is within the required range (10 - 20 meters) of the device. 2. Scan for devices within the connectivity range. 3. Display list of available devices to the user. 4. Allow the user to select the device to connect. 5. Establish connection with the Bluetooth burner and verify validity by prompting temperature changes
Pairing button must initiate pairing search for Bluetooth devices and successfully connect within atmost one minute	1. Activate the pairing mode on the phone app. 2. Scan for Bluetooth devices available for pairing. 3. Provide a list of devices to the user for selection. 4. Initiate the pairing process with the chosen device. 5. Test connectivity time and record average time over twenty attempts

References

[1] Allegro Microsystems. Current sensor datasheet. hhttps://www.digikey.com/en/products/detail/allegro-microsystems/ACS711KEXLT-31AB-T/3868195.

[2] Microchip. Op-amp datasheet. https://ww1.microchip.com/downloads/en/DeviceDoc/MCP6001-1R-1U-2-4-1-MHz-Low-Power-Op-Amp-DS20001733L.pdf.

[3] Texas Instruments. Tps540a50 datasheet. https://www.ti.com/lit/ds/symlink/tps542a50.pdf.

[4] Diodes Inc. 3.3v converter datasheet. https://www.diodes.com/assets/Datasheets/AP62200_AP62201_AP62200T.pdf.

[5] Espressif Systems. Esp32-wroom-32e datasheet. https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32e_esp32-wroom-32ue_datasheet_en.pdf, .

[6] CUI Devices. Cp40 series peltier module datasheet. https://www.cuidevices.com/product/resource/cp40.pdf.

[7] Nichrome 60-15 medium temperature resistor material. URL https://www.matweb.com/search/datasheet_print.aspx?matguid=014093642976472984e91c7392e67b55. Retrieved from https://www.matweb.com.

[8] Vishay Siliconix. Sqj951ep datasheet. https://www.vishay.com/docs/63658/sqj951ep.pdf.

[9] Espressif Systems. Esp32 bluetooth le & bluetooth faqs. https://docs.espressif.com/projects/esp-faq/en/latest/software-framework/ble-bt.html, .

[10] Ieee code of ethics. https://www.ieee.org/about/corporate/governance/p7-8.html.

[11] Thermal conductivity of ceramics. https://global.kyocera.com/fcworld/charact/heat/thermalcond.html.