Healthy Chair

ECE 445 Final Report

Team 5
Wang Qiuyu (qiuyuw2)
Ryan Chen (ryanc6)
Alan Torkarsky (alanmt2)

TA: Shivang Charan
Spring 2022

Abstract

The Healthy Chair project aims at lowering the risks of health issues stemming from prolonged sitting periods as well as unhealthy sitting posture. Our chair is able to gather a user's sitting posture by collecting data from pressure sensors located on the seat and back of the chair. The data is displayed on a user interface (UI) system with visual and audio components. The user also enjoys the freedom of setting their desired sitting time for the UI system to remind them to leave the chair.

Table of contents

1. Introduction **3**

1.1 Problem 3

1.2 Solution 3

1.3 High Level Requirements 3

2. Design **4**

2.1 Block Diagram 4

2.2 On-Chair System 4

2.2.1 Data Collection 4

2.2.2 On-Chair Power 5

2.2.3 Fan 6

2.3 UI System 6

2.3.1 OLED Display 6

2.3.2 User Power 6

2.3.3 User interface 7

2.3.4 Speech 7

2.4 Microcontroller System 7

3. Design Verification **8**

3.1 On-Chair system 8

3.1.1 Data Collection 8

3.1.2 On-Chair Power 9

3.1.3 Fan 11

3.2 UI System 11

3.2.1 OLED Display 11

3.2.2 User Power 12

3.2.3 User Interface 12

3.2.4 Speech 13

3.3 Microcontroller System 14

4. Costs and Schedule **16**

4.1 Labor 16

4.2 Parts 16

4.3 Total Cost 17

4.4 Schedule 17

5. Conclusion **19**

5.1 Accomplishments 19

5.2 Uncertainties 19

5.3 Ethics 19

5.4 Future work 20

References **21**

Appendix **22**

1. Introduction

1.1 Problem

The majority of the population are seated for most of the day, whether they are office employees, working at their desks, or students doing homework. In particular, during the COVID-19 pandemic, many people are either working remotely at home or quarantining for several weeks. They tend to work out less and sit longer and they are more likely to suffer from health problems like hemorrhoids, obesity, and even heart diseases [1]. Research has shown that sitting for too long is detrimental to one's bottom and urinary tract that can result in urinary urgency. In addition, poor sitting posture can lead to reduced blood circulation, joint and muscle pain, and other health-related issues [2].

1.2 Solution

To solve the problem, we propose the Healthy Chair that alerts the user of his or her sitting time based on customized input. To simulate a realistic scenario, we use a mesh office chair for our project. A speech module is used to pre-record our voices and send messages to remind the user to get up from the chair and stretch for a while. Two pressure sensor matrices are installed on the chair's seat and back, and users are able to see which pressure sensors in the matrices are activated. The matrices are displayed to the user via the OLED screen on our UI system. The UI system is separate from the on-chair system and is designed to be placed wherever the user would like. In this way, users can get a good understanding of their sitting posture. A non-contact IR temperature sensor and a fan are also installed under the seat of the mesh office chair. When one sits for too long, the IR sensor will detect an increase in the seat's temperature and the fan will be turned on to cool off the chair. For the transmitting and receiving we use two HC05 Bluetooth Modules hooked up to the serial ports on our microcontrollers. The UI microcontroller collects data from the Input microcontroller and does all the processing for both systems. It can then send necessary data and signals to the chair in order to turn on and off the fan. On the other hand, all the data collected from the chair gets displayed on the OLED after it is processed. All components of our chair can be powered either through a rechargeable 3.7 V Lithium-ion battery or from a direct connection to 120 V ac. However, the UI system is directly powered from a wall adapter.

1.3 High Level Requirements

The high level requirements for our Healthy Chair project is to first alert the user to stand up using the speech module after the exact time duration inputted by the user through the UI system. The OLED screen on the user interface must display the chair seat's temperature and the triggered pressure sensors in real time. Finally, our Healthy Chair must be able to switch between a rechargeable 3.7 V Lithium-ion battery and a 120 V ac wall power source.

2. Design

2.1 Block Diagram

Figure 1 is our block diagram that includes the data collection, input power, user power, speech, user interface, microcontrollers, and the OLED display. The user power system is responsible for regulating two different voltages to power the entire user interaction system. The input power subsystem is responsible for charging the battery and regulating the voltage supply. The speech subsystem is responsible for recording and playing audio to alert the user when to take a break. The user interface subsystem allows the user to set parameters and it interacts with the entire on-chair system. The OLED display is used to show graphical and text to the user. The microcontrollers form the core subsystem since they control the program logic. They communicate via the HC-05 Bluetooth Module, so additional wires for the communication between the on-chair system and the UI system are removed.

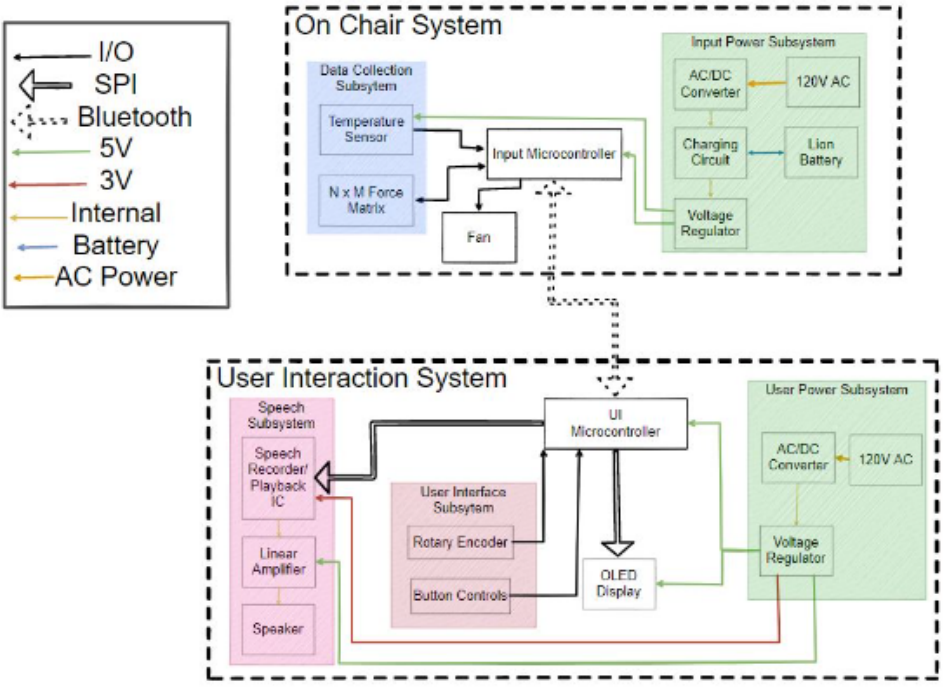


Figure 1: Healthy Chair Block Diagram

2.2 On-Chair System

2.2.1 Data Collection

The data collection subsystem is responsible for collecting data on the user's sitting position, and it consists of twenty-four pressure sensors and one IR temperature sensor. The pressure sensors are arranged into two matrices: a 4x4 matrix on the seat cushion and a 4x2 matrix on the back

support of the chair. By utilizing 8 ADC pins for 24 sensors, we are able to use a matrix scanning algorithm for reading the sensor values to reduce the number of pin usage of the input microcontroller. Figure 2 shows how the matrices are arranged. For the matrix scanning algorithm, one row of resistors is enabled while all the columns are read simultaneously. This process will continue until each row has been read. We record the voltage level between the pull-up resistor and each corresponding pressure sensor and compare it to 5 V. The internal ADC has 10 bits of resolution, indicating that each pressure sensor's output falls between 0 and 1023. The larger the output value, the less pressure is exerted on the sensor, and vice versa (a value of 1023 means that no one is sitting on the sensor).

The second part of the data collection subsystem is the IR temperature sensor. It is connected to the input microcontroller via I2C. The temperature sensor can output the object temperature as well as its ambient temperature. Since we want to know the temperature of the seat and its surrounding area, we only care about the IR sensor's ambient temperature readings. If the reading exceeds a threshold value set by the user, the fan at the bottom of the chair will be turned on to cool off the chair.

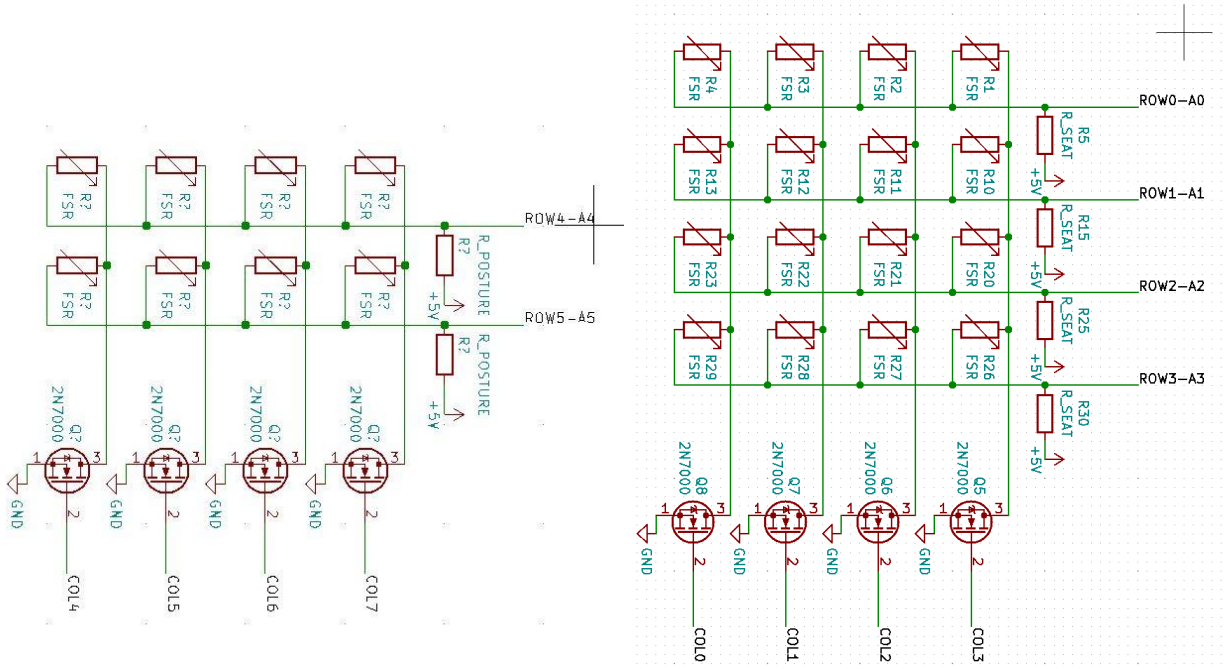


Figure 2: Schematic of the 4*2 and 4*4 Pressure Sensor Matrices

2.2.2 On-Chair Power

The on-chair power subsystem is responsible for providing power to each one of the chair's components. The on-chair power schematic and PCB layout can be found in Figures 14 and 16 in the Appendix respectively. Its main power source is a 3.7 V Lithium-ion battery that is connected to a boost converter so that 5 V can be provided to the sensors and input

microcontroller when the chair is using battery or wall power. The output voltage is determined by the following:

$$V_{out} = 1.25 * (1 + \frac{R_2}{R_1}) \tag{1}$$

Choosing the values of R2 and R1 to be 6 kΩ and 2 kΩ, we are able to obtain a 5 V output as long as the input voltage to the boost converter is greater than 3 V (its turn-on voltage). The Lithium-ion battery has a capacity of 1250 mAh, and since the total current draw of the on-chair components is less than 100 mA, there should be no issues of battery draining too quickly. In order to keep the battery charged, we use a battery charging IC to ensure that the voltage of the battery does not drop below its minimum discharge voltage of 3 V. The charging circuit receives power from an ac/dc wall adapter and USB power.

2.2.3 Fan

As mentioned earlier, the fan is used to cool down the chair when its surface temperature surpasses the user's set threshold value. We debated on using the fan at first since it draws more power than the rest of the systems combined. We decided to use it as the fan is not constantly on. We also put constraints to turn off the fan as much as possible to save power. The fan is wired up using general I/O and runs through a MOSFET for power to reduce current flow through the input microcontroller.

2.3 UI System

2.3.1 OLED Display

The OLED display is the main output of the UI system. It allows users to navigate through menus, save settings, and view their sitting positions. The OLED display uses SPI protocols to communicate with the UI microcontroller. We only use the data_command pin on the OLED as it is necessary for differentiating between control signals and data values.

2.3.2 User Power

The goal of the user power subsystem is to provide constant power to all UI components. The user power consists of a wall adapter that takes an input of 120 Vac and outputs a steady 9 Vdc. We use two LM317 linear voltage regulators to provide output voltages of 3 V and 5 V. The equation for the output voltage is the same that of the boost converter:

$$V_{out} = 1.25 * (1 + \frac{R_2}{R_1}) \tag{1}$$

In our design, we choose 336 Ω and 240 Ω for R2 and R1 to achieve an output voltage of 3 V, and we set R2 and R1 to be 720 Ω and 240 Ω for achieving 5 V. The OLED display and the speech module are the two main devices that need to be powered. The OLED display runs at 3 V with an average current draw of 20 mA. The speech module requires 3 V/ 3 mA for the speaker, 3 V/ 1 mA for the voice recording IC, and 5 V/ 7 mA for the linear amplifier.

2.3.3 User interface

The user interface is used for inputting commands into the UI System. It consists of 3 buttons and a rotary encoder. Originally, we had all our inputs connecting directly to the output microcontroller, but we realized that we also needed to include the debouncing circuits. We thought about using a software debounce function, but since we were collecting inputs through interrupts, we didn't want to hold up the system with delays inside the interrupts. We ended up using the LS19 debouncer for all the switch inputs and a simple RC debouncing circuit for the encoder output pictured in Figure 3**.**

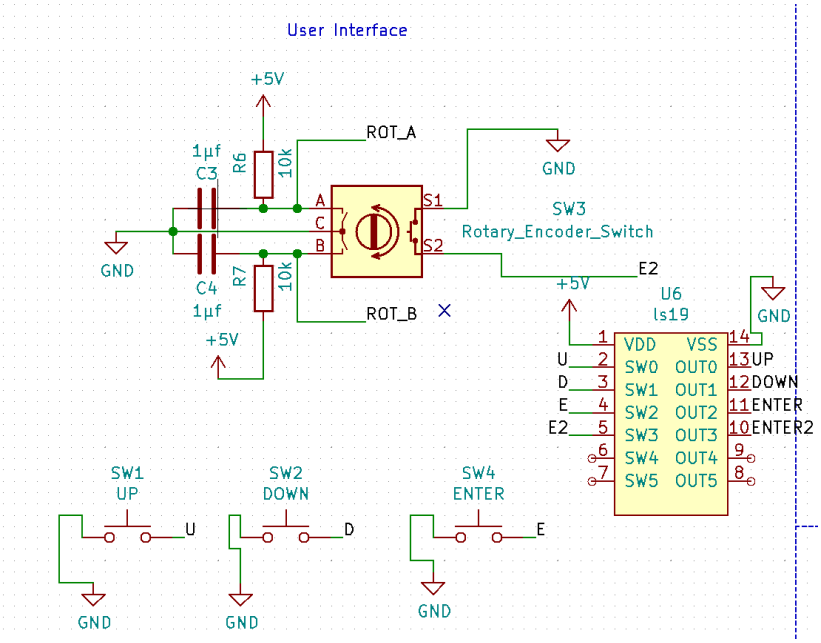


Figure 3: User Interface Schematic

2.3.4 Speech

Controlled by the microcontroller and powered by the power system, the speech module consists of an ISD4002 voice recording chip, an LM4860M linear amplifier, a microphone, and a speaker. The voice chip operates at 3 Vdc, while the linear amplifier requires 5 Vdc supplied by the user power system through a boost converter.

When we record our voices using the microphone, signals are recorded and played back through the voice chip, amplified by the linear amplifier, and finally outputted by the speaker. The voice chip interacts with the output microcontroller through SPI communication.

2.4 Microcontroller System

The microcontroller system is the logic unit for our Healthy Chair. We have two ATMEGA1284-PU microcontrollers for both of our systems. The input microcontroller on the chair system is responsible for collecting data and sending it over to the user microcontroller. The user microcontroller is responsible for driving all the peripherals including the speech

module, the OLED display, and the user interface. The two microcontrollers interact via HC05 Bluetooth modules connected to the serial ports on their respective microcontrollers . We have the microcontrollers working in a master slave configuration, meaning that the on-chair microcontroller relies on the user microconroller to tell it what to do. The on-chair microcontroller will continuously poll until it receives a command from the UI microcontroller.

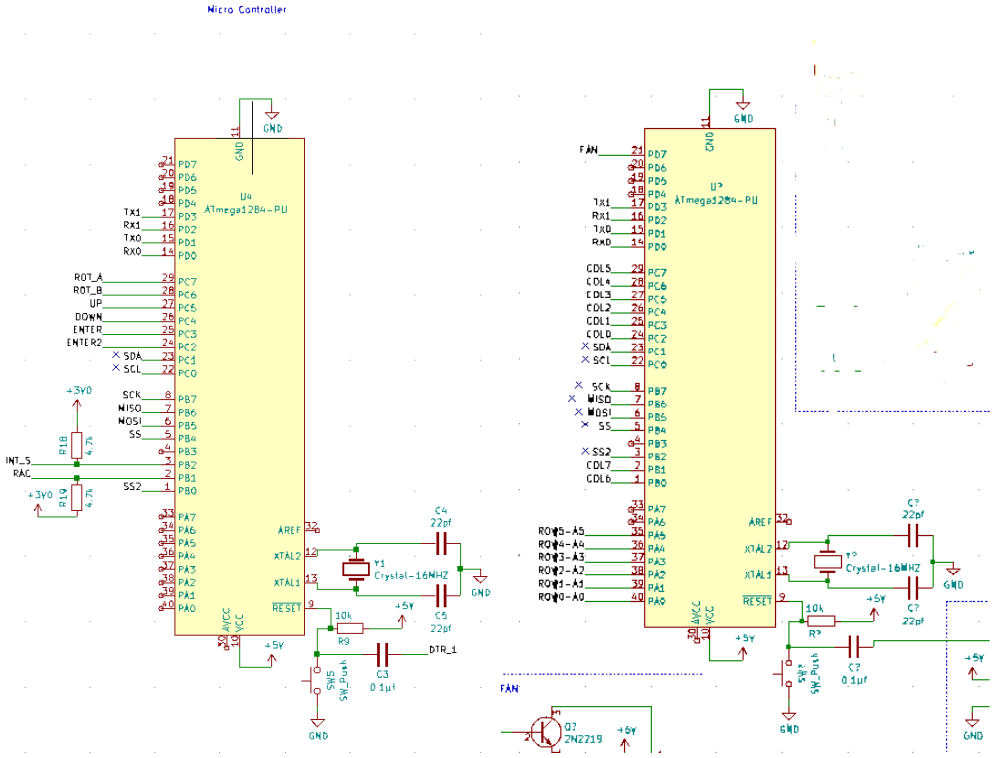


Figure 4: UI(Left) and Input (Right) Microcontroller Schematic

3. Design Verification

3.1 On-Chair system

3.1.1 Data Collection

To verify the functionality of our data collection subsystem, we need to test each sensor first to make sure they all read the values independently. Since we are using a matrix approach, we need to verify that sensors do not bleed their values into other sensors' data. From Table 3 in the Appendix**,** we need to make sure that we can have five diffrentable zones and read them all within 10 ms. The five colored zones appeared clearly on the display as we sat on the chair. On the other hand, we needed to add timers to our code in order to test the pressure sensors' readings. From Figure 5, we can see that the results from the readings are between 960 and 970 ms. It may seem that we failed our test at first glance, but in fact, we had a hard-coded constant of 40 ms per pressure sensor reading. This is equivalent to having a hard-coded delay of 960 ms. Therefore, all 24 pressure sensors read in under 7 ms, which is below our intended 10 ms per sensor.

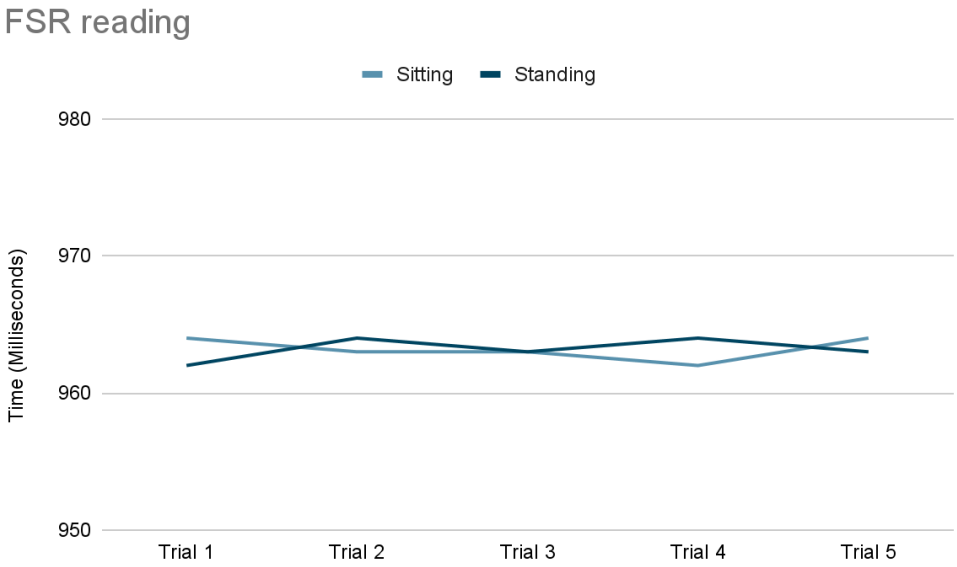


Figure 5: Pressure Sensor Read-In Time

The IR temperature sensor is the other part of the data collection system that we need to verify. To that end, we sat on the chair for a couple of minutes and watched the OLED screen displaying data from the sensor. We then tested the temperature displayed against an external temperature gun we obtained in the lab. It turned out that the temperature sensor's reading was within the constraint we put forward in Table 3 (±0.5 °C of accuracy).

3.1.2 On-Chair Power

To verify our on-chair power system, we first wanted to ensure that our chair could run on wall power. A multimeter was used to measure the voltage across the USB (5 V). After testing our power system with wall power, we connected our battery to the on-chair system and removed the charger from wall power. The output remained at 5 V, which means the boost converter worked as the battery voltage was around 4.1 V. After connecting the wall power charger back, we saw that our red LED lit, indicating the battery was indeed charging. We placed a multimeter across the battery terminals while the system was plugged into the wall and saw that the voltage was increasing at a rate of 1 mV / 30 seconds. The second and third verification are related to the 3.7 V Lithium-ion battery itself. The battery we have includes a protection circuit to ensure that the voltage does not drop below its minimum discharging voltage of 3 V. When the battery was charging and discharging, we measured the temperature of the battery using an infrared thermometer and verified that it stayed around 24 ℃, which is around room temperature. This is well below the rated temperature of 45 ℃ from the datasheet. We also tested the battery discharging results to determine how long the chair could be powered with the fan being turned on and off.

Below is the battery power capacity:

$$3.09\ Wh = 1250\ mAh * \frac{3.75\ V}{1000} * 0.66 \tag{2}$$

The expected run time with fan turned on satisfies the following formula:

$$4.29\ hours = \frac{3.09\ Wh}{0.55\ W(Fan) + 0.15\ W(Bluetooth) + 0.002\ W(Arduino) + 0.02\ W(Temperature\ Sensor)} \tag{3}$$

The expected run time with fan turned off satisfies the following formula:

$$18\ hours = \frac{3.09\ Wh}{0.15\ W(Bluetooth) + 0.002\ W(Arduino) + 0.02\ W(Temperature\ Sensor)} \tag{4}$$

We tested these results when the battery was between 20%-80% of its power capacity where the discharging rate is relatively linear. We calculated that the battery discharged at an average rate of 0.5 V/hour with the fan on and 0.09 V/hour with the fan off. Figure 6 depicts our calculated battery discharge time.

We tested the battery's actual discharging rate and found that it lasted around half of the calculated time. In addition, it is worth noting that we extrapolated the data because the discharging rate is linear around the middle 60% of the battery power capacity. Our findings of the expected and actual run time with the fan being turned on and off is shown in Figure 7.

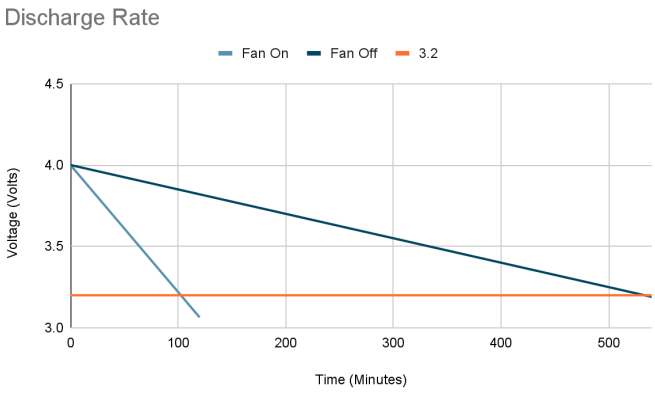


Figure 6: Battery Discharge Rate w/ & w/o Fan On

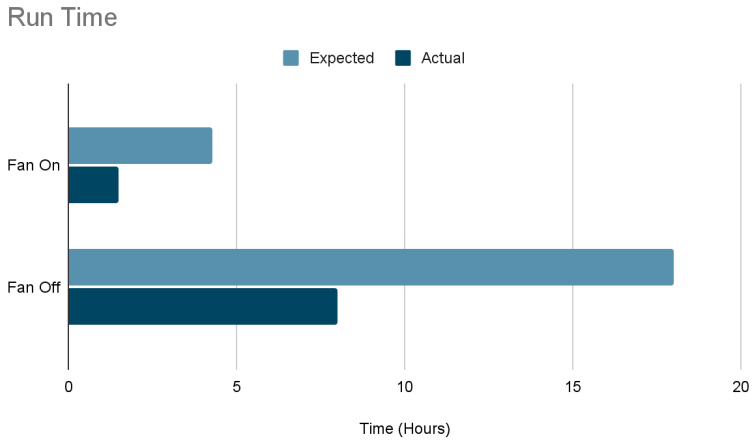


Figure 7: Expected and Actual Time for Healthy Chair to Operate

3.1.3 Fan

To verify the fan's functionality, we first had to heat up the chair by sitting on it. After several minutes, we manually turned on the fan and started a timer to see how long it would take for the temperature to drop 1 ℉. From our testing, we concluded that the fan lowers the chair temperature by 1 ℉ in about 4 minutes and 17 seconds. This fell within our constraint of 10 minutes. To test the fan's noise when it was on, we used a decibel app on our phone and measured the noise to be 20.3 dB that also satisfied the requirement listed in Table 5.

3.2 UI System

3.2.1 OLED Display

To verify the OLED display, we needed to make sure that the OLED screen could display all the information as well as draw the screen within one second of the input as per Table 7 in the Appendix. First, to test display, we had to create the menus and verify that the screen showed our anticipated graphics. As soon as we turned on the system, we could see the menus pop up. By using the user interface, we can scroll through the menus and toggle settings. This proved that the OLED was displaying the correct data. The second test for speed required us to program in

timers before and after the OLED draw function. We used the most graphically intensive screen, the chair sensor map, for the OLED timing test. As shown in Figure 8, the OLED performed very well. This is because we were able to have the OLED draw in under 285 ms, which beat our constraint by almost four times.

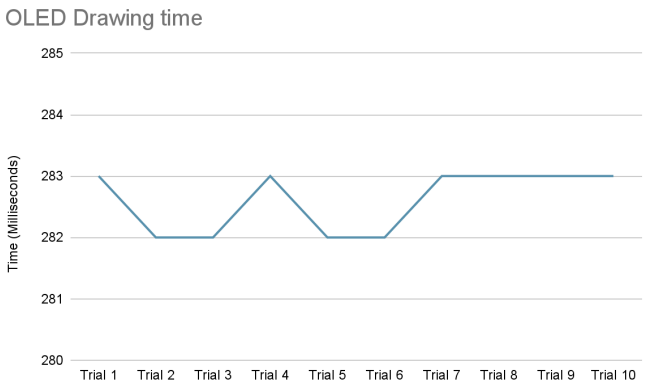


Figure 8: OLED Display Time

3.2.2 User Power

Before we received the PCB, we tested the LM317 voltage regulators on a breadboard first to ensure that the resistor values we chose provided the correct output voltage. A multimeter was placed at the outputs of the 3 V and 5 V linear voltage regulators. The output voltages were within 5% of the desired voltages. The speech module and linear amplifier were connected and both of the components were able to run properly.

3.2.3 User Interface

To verify our user system, we just had to do one simple test. We needed to watch the OLED screen and make sure that the buttons do what they are labeled. To test the rotary encoder, we needed to make sure that the settings could change with one spin of the encoder. Likewise, spinning the encoder the other way should decrease the value. One boundary condition that we tested was decreasing the settings value past zero. The OLED screen showed us that all the buttons worked. We also tested button speeds as shown in Figure 9. The buttons responded within 1 ms, while the encoder usually responded within 1 ms. Sometimes the encoder got caught up and took up to 30 ms, and we think this comes from the fact that we were using a different debouncing method for the encoder outputs.

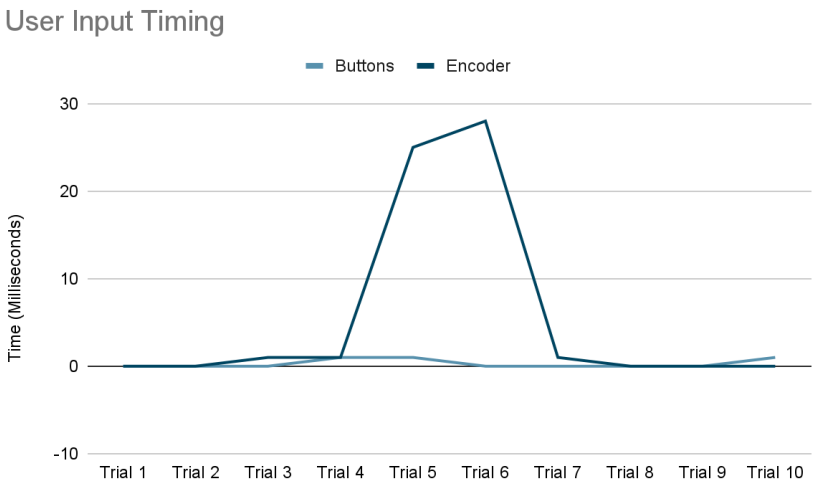


Figure 9: User Input Time

3.2.4 Speech

The verification for the speech module is quite straightforward: Although the voice chip can record and playback voices of up to 4 minutes [3], during our actual demonstration, we just recorded a 10-second speech and let it playback in a cycle as long as the user did not leave the chair.

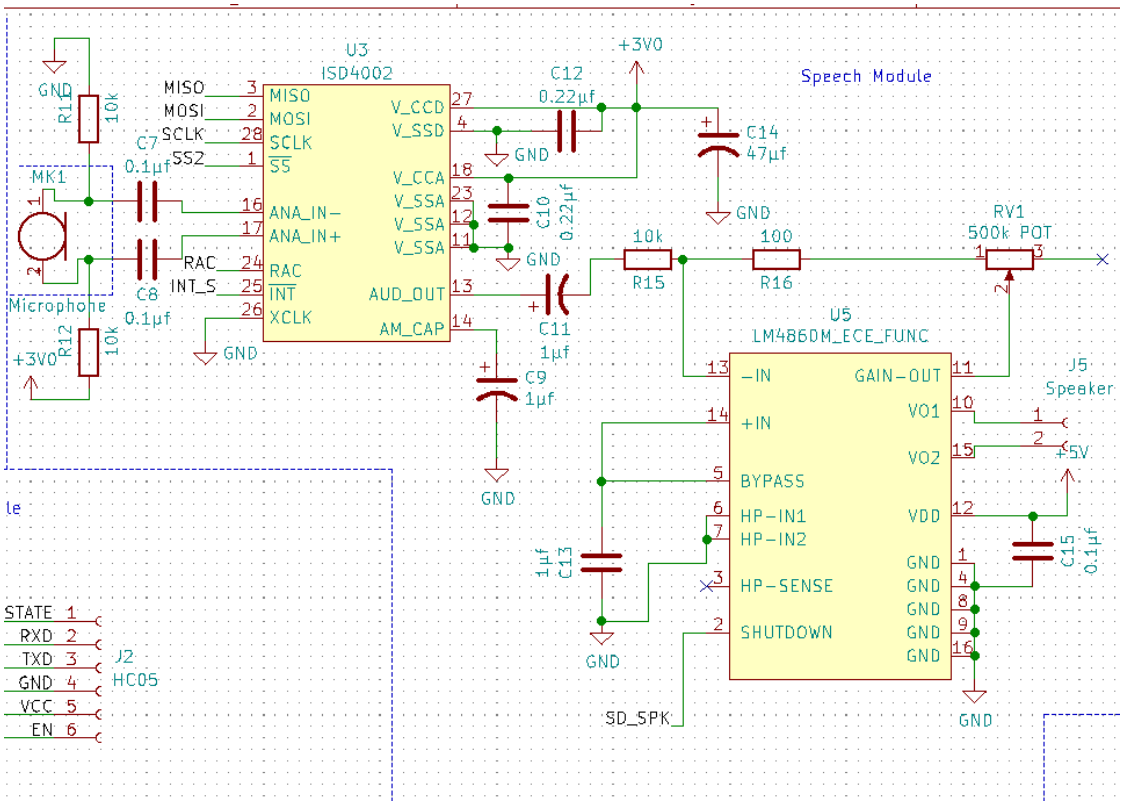


Figure 10: Speech Module Schematic

One important design decision we made during our testing of the speech module is settling on an adequate gain of the linear amplifier. As is shown on the speech module schematic above, the gain is given by

$$Gain = 2 * \frac{R_{V1}}{R_{15}} \quad (5),$$

where RV1 denotes the potentiometer's value and R15 is a fixed resistance of 10 kΩ. In our testing, we made RV1 to be as large as 2 MΩ, increasing the gain to be as large as 400. However, it turned out that we were not able to hear any playback messages clearly given the fact that surrounding noises were also significantly amplified. We tried to make our own low pass RC filter by soldering a 150 pF capacitor between pin 11 and pin 13 of the linear amplifier, but the noise decoupling capacitor did not reduce the noise. Consequently, we decided to reduce RV1 to be 500 kΩ, and we were able to hear clear playback messages with the linear amplifier's gain being 100.

3.3 Microcontroller System

The microcontroller system included tests that required other systems being functional, so we had to verify it at the very end. The first test we did was to test the Bluetooth response time as shown in Figure 11. Our constraint was to be within 2 seconds, and we managed to stay under 1.4 seconds. This response time included the send and receive function calls and the data processing. We included the data processing timing in Figure 12. Without our debug statements, the data processing would drop to 1 milliseconds. Although we dropped 80 ms, the Bluetooth response time stayed constant. Hence, it led us to conclude that the serial communication between the microcontroller and the Bluetooth module was our limiting factor. Our next test included the user interface and its interaction with the microcontroller. We needed to make sure that the user interface changed the settings that were stored in the microcontroller. This test is described in detail in Table 10 in the Appendix. The last test included our parameters and their ability to be stored between power cycles. To test this out, we changed the parameters and then proceeded to unplug our product and plug it back in. The settings were able to be stored and saved regardless of the amount of power cycles we had.

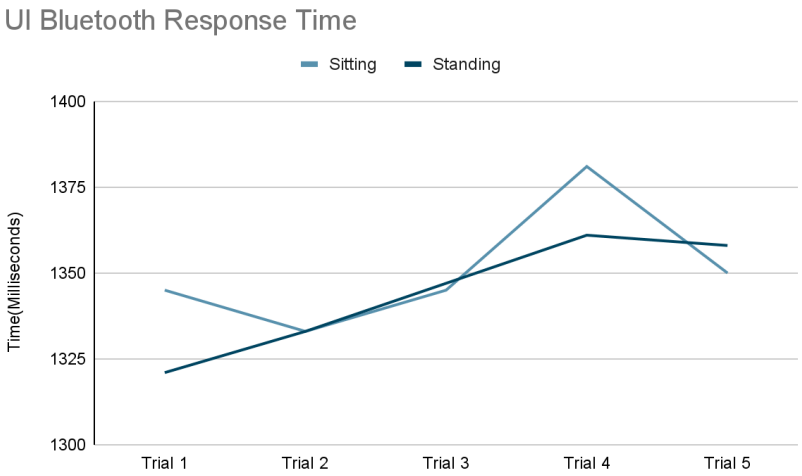


Figure 11: UI Bluetooth Response Time

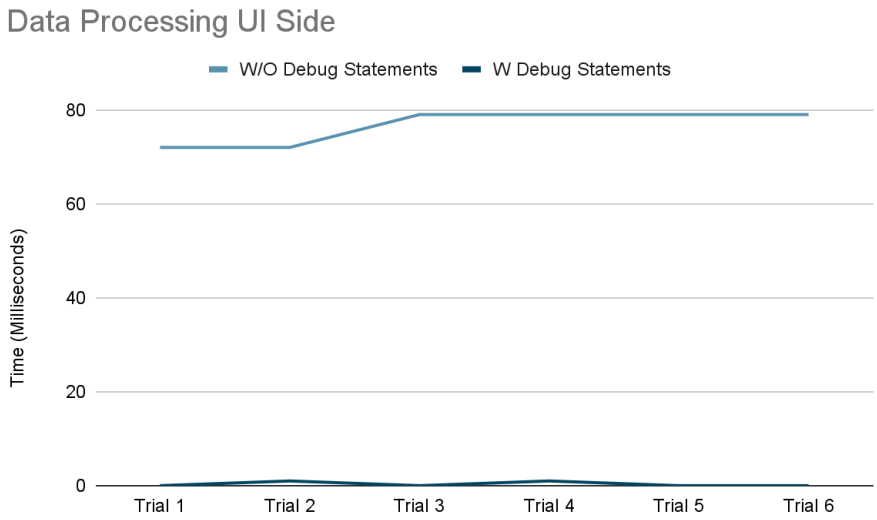


Figure 12: Microcontroller Data Processing Time w/ & w/o Debug Statements

4. Costs and Schedule

4.1 Labor

Assuming the salary of labor is $30/hour for each individual, each team member works for an average of 10 hours per week, and there are 16 weeks throughout the spring semester (including spring break since we will also work on the project). Hence, the estimated total labor cost per team member is

$$\$36,000 = \frac{\$30}{hour} * 2.5 * \frac{10\ hours}{week} * 16\ weeks * 3\ members$$

4.2 Parts

Description	Manufacturer	Part #	Quantity	Individual Cost ($)	Bulk Cost ($)
OLED Display	Adafruit	ST7789	1	24.99	24.99
Wire	Adafruit	1528-4730-ND	1	6.95	6.95
Thin Film Pressure Sensor	DFRobot	RP-C18.3-ST	24	5.00	120.00
Bluetooth Module	DEVMO	HC-05	1	19.99	19.99
IR Temperature Sensor	DFRobot	MLX90614	1	16.00	16.00
Microcontroller	Microchip Technology	ATMega1284	2	6.77	13.54
Office Chair	Furmax	/	1	154.98	154.98
Linear Amplifier	Texas Instruments	LM4860M	1	3.08	3.08
Speaker	CUI Devices	CLS0231-L152	1	4.73	4.73
Voice Record IC	Nuvoton	ISD4002-240PY	1	8.59	8.59
Voltage Regulator	Torex Semiconductor	XC6206P302MR-G	1	0.65	0.65

Power Barrel Connector	CUI Devices	PJ-102AH	1	0.81	0.81
Switch Push Button	E-Switch	TL2230OAF 140	1	0.75	0.75
USB Connector	GCT	USB1130-15	1	0.79	0.79
Boost Converter	Texas Instruments	MC33063A	1	0.40	0.40
Lithium-ion Battery	PKCell	1781	1	9.95	9.95
JST Connector	TE Connectivity	440055-2	1	0.26	0.26
Charging IC	Texas Instruments	BQ24210DQCR	1	2.85	2.85
Fan	Sunon Fans	MF40100V2	2	6.01	12.02
Total					**401.33**

Table 1: Parts Cost

4.3 Total Cost

The total cost is equal to the sum of the labor and parts costs. It is given by the following expression:

$$\$36,401.33 = \$36,000 + \$401.33$$

4.4 Schedule

Week	Tod	Alan	Ryan
2/14/2022	Research Part Number for Speech Module	Research Part Number for Microcontroller	Research Part Number for Power
2/21/2022	Design Speech PCB circuits/schematics + Work on Design Documentation	Design Microcontroller PCB circuits/schematics + Work on Design Documentation	Design Power PCB circuits/schematics + Work on Design Documentation

2/28/2022	Order Parts and Finalize PCB Design	Order Parts and Finalize PCB Design	Order Parts and Finalize PCB Design
3/7/2022	Research Microcontroller Coding and Test Arrived Components	Research Microcontroller Coding and Test Arrived Components	Research Microcontroller Coding and Test Arrived Components
3/14/2022	Spring Break	Spring Break	Spring Break
3/21/2022	Test Parts and Integrate Into Project	Write Test Codes for Microcontroller	Test Parts and Integrate Into Project
3/28/2022	Test Parts and Write Individual Progress Report	Test Parts and Write Individual Progress Report	Test Parts and Write Individual Progress Report
4/4/2022	Test Parts and Prepare for Mock Demo	Test Parts and Prepare for Mock Demo	Test Parts and Prepare for Mock Demo
4/11/2022	Finalize Physical Project for Mock Demo	Finalize Physical Project for Mock Demo	Finalize Physical Project for Mock Demo
4/18/2022	Finalize Any Errors from Mock Demo and Prepare for Final Demo	Finalize Any Errors from Mock Demo and Prepare for Final Demo	Finalize Any Errors from Mock Demo and Prepare for Final Demo
4/25/2022	Demo, Give Presentation and Write Final Report	Demo, Give Presentation and Write Final Report	Demo, Give Presentation and Write Final Report

Table 2: Schedule After Design Review

5. Conclusion

5.1 Accomplishments

We are glad that all three high level requirements have been met and all R&V tables have been verified. The sensors on the chair are able to communicate with the input microcontroller, and the UI system is responsive to the output microcontroller as well as the user's customized inputs. In addition, the UI system can use wall power and the on-chair system is able to run on its rechargeable battery.

5.2 Uncertainties

Although all of our parts are functioning, there is still much room for improvement in our project. When we tested the pressure sensors on the back, they did not activate because there was not enough force to trigger them.

In a mechanical engineering perspective, our pressure sensors are not installed nicely as they are simply exposed on chair surfaces. Also, our wires (24 pressure sensors on the chair, 48 long wires in total that all go to the bottom of our chair) are somewhat messy under the chair.

5.3 Ethics

Throughout the entire course of our project, we will keep ethics as one of our top priorities. Section I.1 of IEEE code states, “to hold paramount the safety, health, and welfare to the public, to strive to comply with ethical design and sustainable development practices, to protect the privacy of others”. Our main goal of this project is to ensure the safety and health of people that use our chair. To that end, our chair aims at users who sit for long periods of time, especially with improper posture. We are working to make sure that our chair correctly notifies the user when it is time to take a break from sitting for too long and to let them know if they are sitting unevenly or without proper back support.

A safety concern in our project is the power management system. Our project will involve lithium-ion batteries and utilization of high voltage. Higher voltages are more likely to have greater currents, which proposes an increased risk of injury and death [4]. We will use the one hand method and also make sure to check if anything is connected to power before touching.

Lithium-ion batteries are normally safe when they are used in the right operating conditions, but there have been over 25,000 overheating or fire incidents reported from consumer products in a five year period [5]. Overcharging occurs when charging still continues when the battery is at full charge. This causes the battery to prematurely age and for current to continue flowing into the battery which can cause overheating, explosion, and fire. We will implement the proper overcurrent protection to ensure that the battery stops charging when it is full and to make sure that there is no way for the user or anyone around to get shocked.

Since our product is a chair, we consider our product as something that people would want to sit on. Our design is to add pressure sensors in a way that accurately captures the sitting data of the user, but also does not interfere with the user's comfort. Ideally, we want our project's users to not feel the pressure sensors, but depending on trial and error, it may be necessary for them to feel them but they will not cause any discomfort.

Section I.5 of IEEE describes working with honest criticism of technical work, to fix any potential errors, make accurate claims, and to properly give credit for people's contributions. We are always looking for areas to improve and are eager to receive feedback, whether it is positive or negative.

5.4 Future work

To improve our design, we can use pressure sensors with lower activation ranges on the back of the chair so that smaller forces can be detected by the pressure sensor. We can also add something to cover the chair's seat in order to hide the sensors from the user's view. There would need to be some modifications made in order to still collect accurate readings from the pressure sensors. To make our wiring more efficient, we can run our wires through the sides of the chair instead of soldering all of them straight to the on-chair PCB on the bottom of the chair. To enhance our UI system's portability, we can also improve our UI power system so that they can be charged with a battery instead of a fixed wall power source. Finally, the HC05 Bluetooth Module is the component that consumes the second most power. Finding a more efficient Bluetooth module would be one way to increase the Healthy Chair's battery life.

References

[1] Harvard Health Publishing, “The dangers of sitting” [Online]. Available at: https://www.health.harvard.edu/pain/the-dangers-of-sitting

[2] Global News, “Sitting all day at work? Get up every 30 minutes to cut your risk of death”[Online]. Available at: https://globalnews.ca/news/3740438/sitting-all-day-at-work-get-up-every-30-minutes-to-cutyour-risk-of-death/

[3] Nuvoton, “ISD4002 Series Datasheet” [Online]. Available at: https://www.nuvoton.com/export/resource-files/DS_ISD4002_DataSheet_EN_Rev1.6.pdf

[4] Division of Research Safety, “Electrical Safety In the Research Laboratory” Division of Research Safety Library [Online]. Available at: https://drs.illinois.edu/Page/SafetyLibrary/ElectricalSafetyInTheResearchLaboratory

[5] Occupational Safety and Health Administration, “Preventing Fire and/or Explosion Injury from Small and Wearable Lithium Battery Powered Devices” [Online]. Available at: https://www.osha.gov/sites/default/files/publications/shib011819.pdf

Appendix

Requirements	Verifications	Verified?
1. There must be at least 5 different pressure sensing zones	1. We will press on the pressure sensors with five increasing values of force and record the voltage across them. We will test for a range from 0-5 V with a 1 V step size.	Yes
2. The pressure sensors must respond within 10 ms	2. We need to turn on power to the pressure sensors we are trying to test. We start the timer when we provide power. We stop the timer after the ADC has returned its value. We then check to see if the value is returned in under 10 ms.	Yes
3. The IR temperature sensor must measure the chair seat's temperature with ±0.5°C of accuracy	3. Use a mercury thermometer and place it on the chair seat's surface. Ensure the temperature sensor's output displayed on the OLED screen matches the reading of the mercury thermometer and satisfies Requirement 3.	Yes

Table 3: RV Table for Data Collection

Requirements	Verifications	Verified?
1. When the battery is being charged, power must be drawn from the wall instead of the battery	1. Ensure the battery is placed in parallel with the charging circuit output. Use a current probe to measure current leaving the battery when it is being charged and ensure the current is less than 10 mA.	Yes
2. When the battery is supplying power, the battery does not drop below its discharge voltage of 2.75 V.	2. Place voltage probes across the positive and negative terminals of the battery. When the system is running, record the voltage of the battery and monitor that it does not fall below 2.75 V.	Yes
3. The battery is stored in a place where the temperature does not	3. Once the battery is placed on the chair, measure the temperature with an IR thermometer while the system is running.	Yes

exceed its maximum operating temperature	Monitor that it doesn't not reach 45℃ while charging and 60℃ when discharging.	

Table 4: RV Table for Input Power

Requirements	Verifications	Verified?
1. The fan must have a noise output of less than or equal to 25dB. 2. The fan must cool off the chair by 1± 0.5 ℉ within 10 minutes	1. Measure the noise output of the fan by using a decibel meter and ensure it maintains below our set threshold 2. When the fan is working, use an IR thermometer to measure the chair seat's temperature for 10 minutes and ensure it satisfies Requirement 2	Yes Yes

Table 5: RV Table for Fan

Requirements	Verifications	Verified?
1. The ISD4002 voice chip must be provided with a voltage of 3 V ± 10% from the voltage regulator, and the LM4860 linear amplifier must be provided with a voltage of 5 V ± 10% from the power supply	1. Measure output voltages from the ISD4002 voice chip and from the LM4860 linear amplifier using an voltmeter or oscilloscope, and ensure they maintain a steady voltage of 3 V allowing a 10% deviation and 5 V allowing a 10% deviation, respectively	Yes
2. The ISD4002 voice chip must supply an output current of 25 mA ± 10% when recording, and the LM4860 linear amplifier must supply an output current of 15 mA ± 10% when playing back the recording	2. Measure output currents from the ISD4002 voice chip and from the LM4860 linear amplifier using an ammeter, and ensure they maintain a steady current of 25 mA allowing a 10% deviation and 15 mA allowing a 10% deviation, respectively	Yes
3. The speech module must record and replay our voices for 20-30 seconds clearly and	3. Record a 25-second speech of warning into the speech module and make sure it can record and accurately play back the recordings	Yes

comprehensively 4. The speech module must maintain its temperature below 70° C.	4. When the speech module is working, use an IR thermometer to measure its ambient temperature for 5 minutes	Yes

Table 6: RV Table for Speech Module

Requirements	Verifications	Verified?
1. OLED display should display all pertinent information	1. To verify functionality, we will have to watch the display as we use the chair system	Yes
2. OLED response time should be within one second from input	2. To verify we will time when we input something until the screen responds.	Yes

Table 7: RV Table for OLED Display

Requirements	Verifications	Verified?
1. Both linear voltage regulators must provide 5 ± 0.5 V and 3 ± 0.3 V	1. With the load(s) connected to either 3 V or 5 V regulator, the output voltage is measured within 10% of 3 V and 5 V	Yes
2. Each output of the linear voltage regulator must be at least 100 mA	2. The output current is connected to a multimeter. The current draw is at least 100 mA for each of the output devices.	Yes

Table 8: RV Table for User Power

Requirements	Verifications	Verified?
1. All buttons and rotary encoder react to user input	1. Clicking the buttons and twisting the encoder should change menus and settings on the OLED display. The OLED display should react accordingly when a button is clicked.	Yes

Table 9: RV Table for User Interface

Requirements	Verifications	Verified?
1. The microcontrollers can communicate via bluetooth	1. Data will be sent over to the microcontroller every 2 seconds. The oled display will display how long between every bluetooth send.	Yes
2. User parameters can be changed from the user interface	2. After a user parameter is changed for example duration of sitting. The microcontroller will act accordingly. If time is changed to two minutes then after two minutes the chair will alert the user that they have been sitting too long. Changing this value again to one minute will verify that we can change and store parameters.	Yes
3. Parameters are remembered between power cycles	3. Cycling power to the microcontrollers and then checking the parameter list should display what was set before the power cycle.	Yes

Table 10: RV Table for Microcontrollers

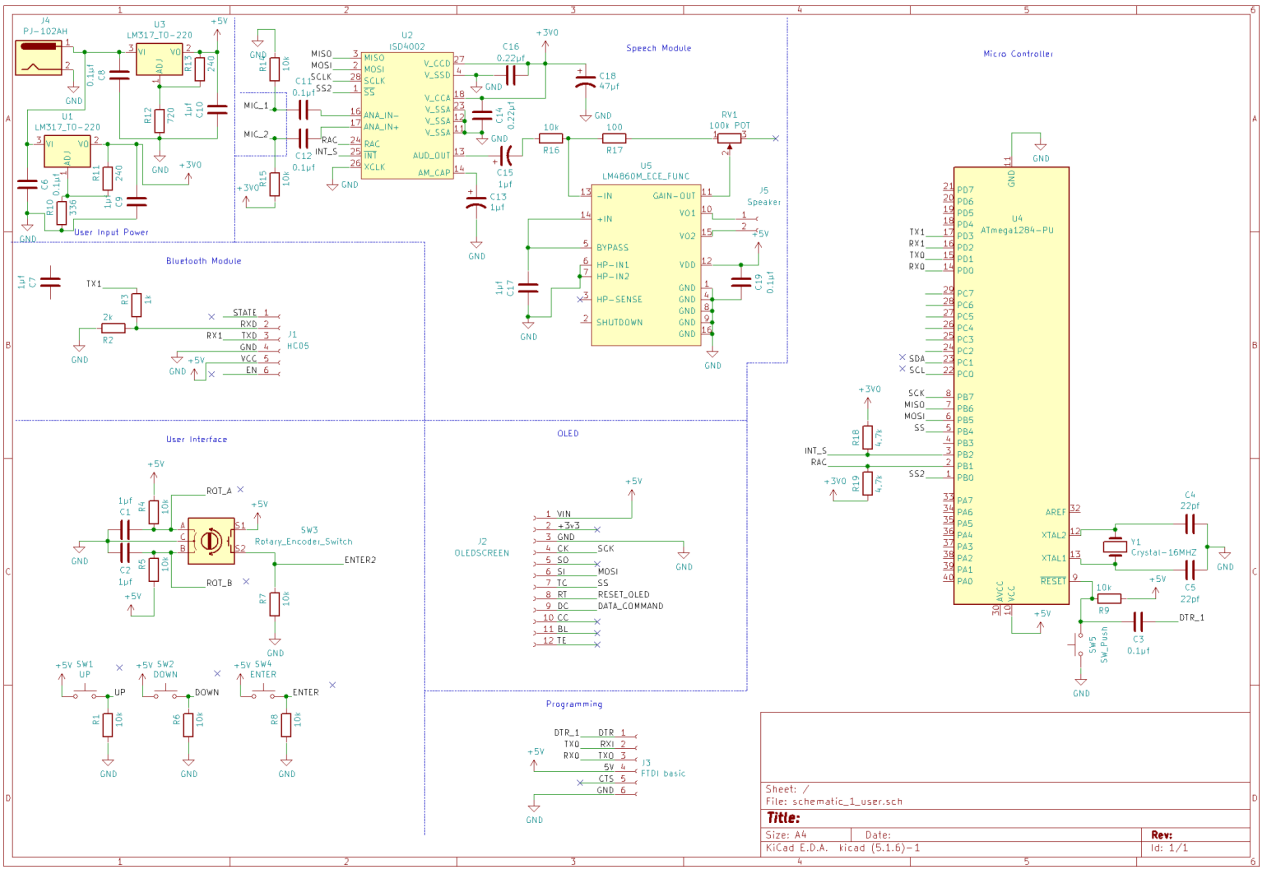


Figure 13: User Interface Schematic

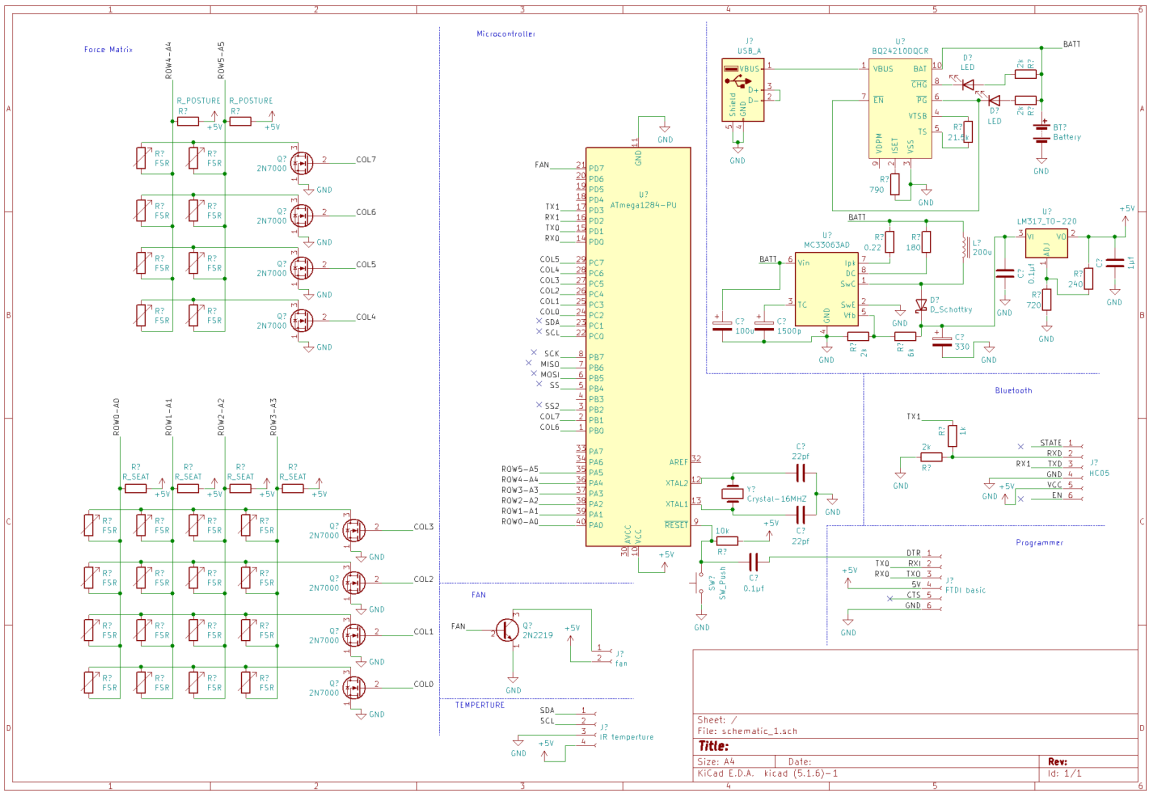


Figure 14: On-Chair Subsystem Schematic

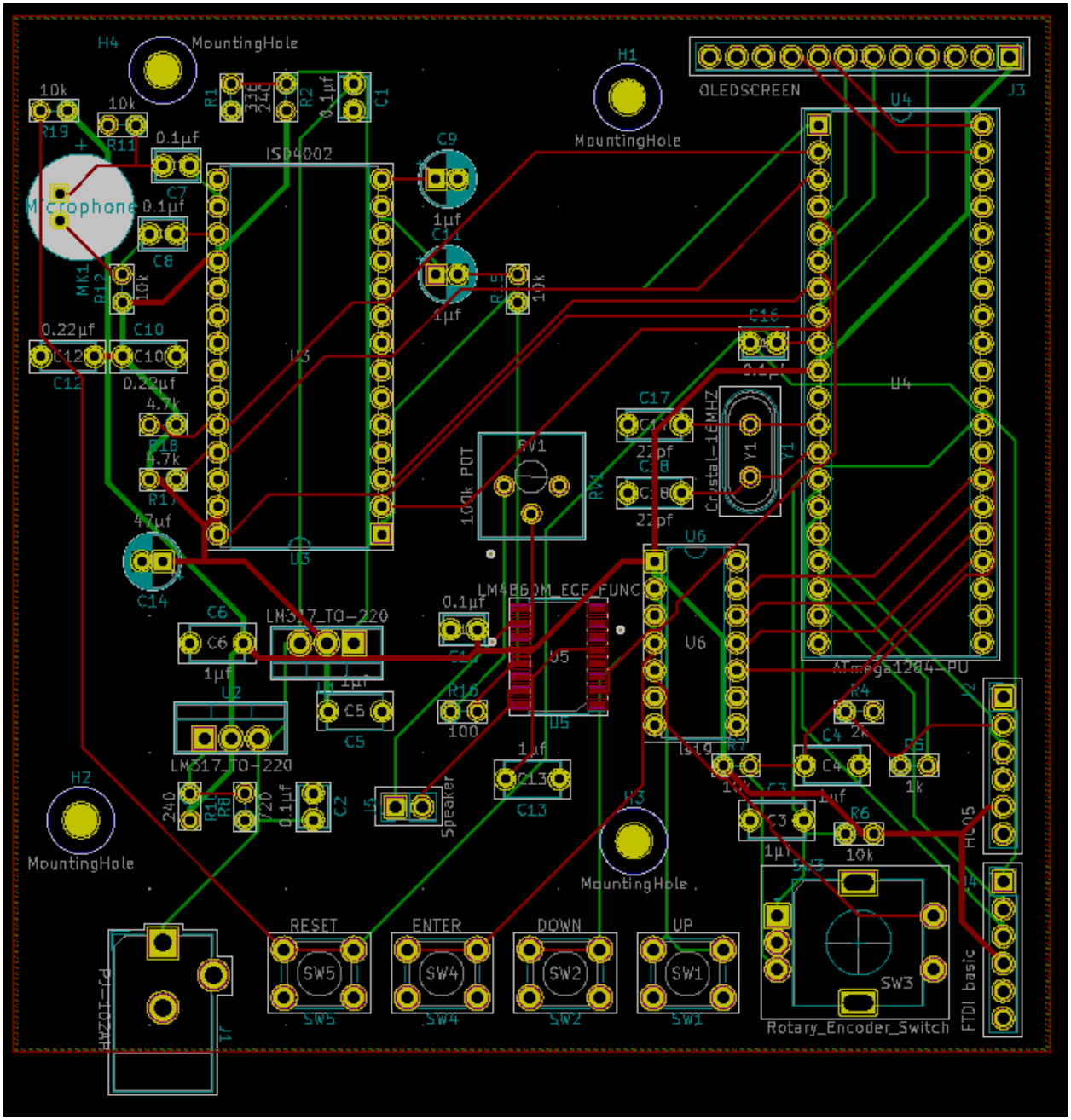


Figure 15: User Interface PCB Layout

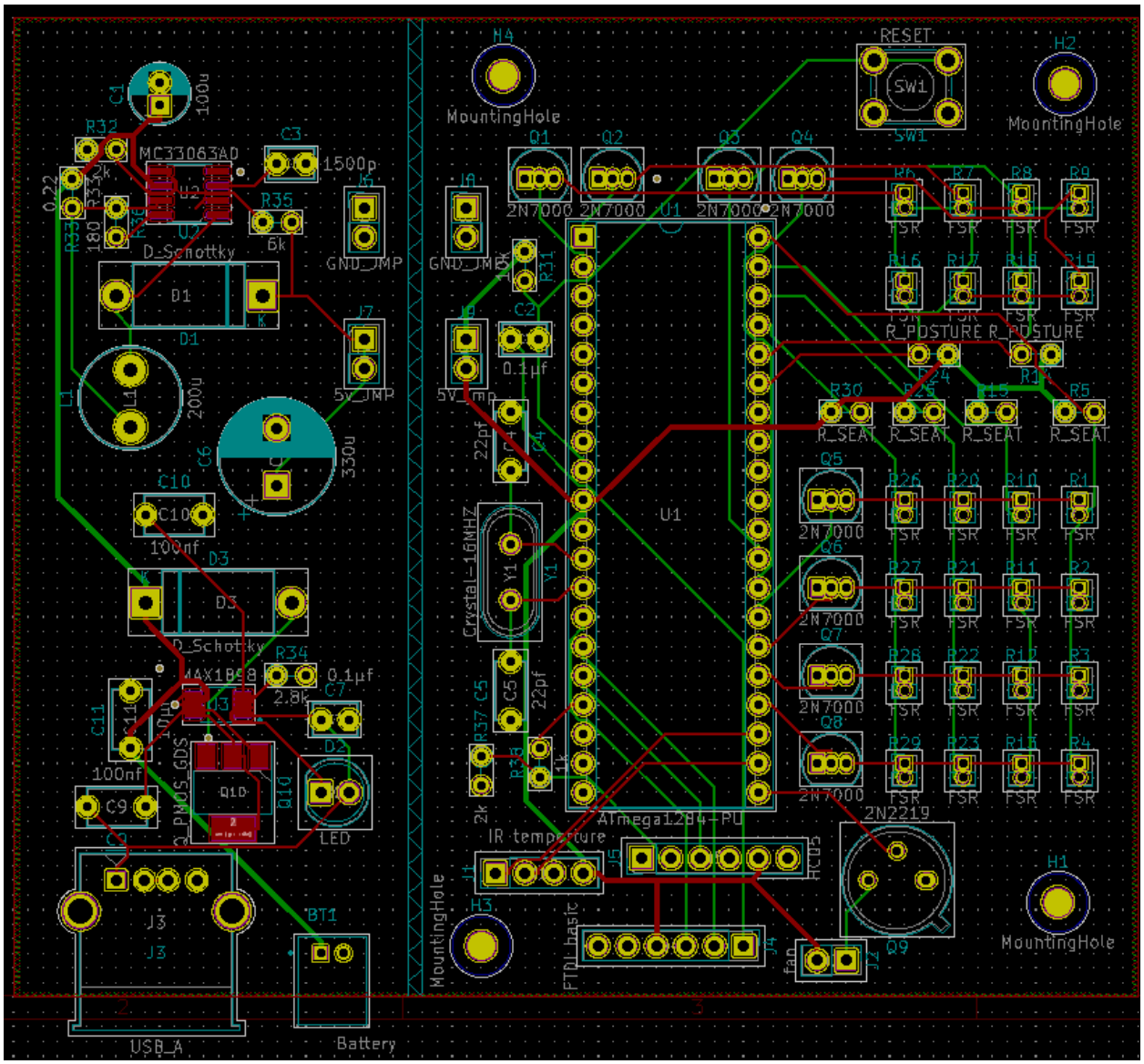


Figure 16: On-Chair PCB Layout

```
#include <Adafruit_GFX.h>    // Core graphics library
#include <Adafruit_ST7735.h> // Hardware-specific library for ST7735
#include <Adafruit_ST7789.h> // Hardware-specific library for ST7789
#include <SPI.h>
#include<EEPROM.h>


//need to write timer code and settings
#define ISD_OPCODE_POWERUP   B00100
#define ISD_OPCODE_SETPLAY   B00111
#define ISD_OPCODE_PLAY      B01111
#define ISD_OPCODE_SETREC    B00101
#define ISD_OPCODE_REC       B01101
#define ISD_OPCODE_STOP      B01100
#define ISD_OPCODE_POWERDOWN B00000
```

```
#define ISD_POWERUP_DELAY    25

//PINDEFS
#define CLOCK 23 //pin 29
#define DATA 22 //pin 28

#define UP 21 //pin 27
#define DOWN 20 //pin 26
#define ENTER 19 //pin 25
#define ENTER2 18 //pin 24
#define ROTATION 99

#define DATAOUT     5//11//5  // MOSI
#define DATAIN      6//12//6  // MISO
#define SPICLOCK    7//13//7  // SCK
#define SLAVESELECT 0//10//0 // SS
#define SLAVE 4

#define TEMP_SAVE 0
#define TIME_HR 4
#define TIME_MIN 8

#if defined(ARDUINO_FEATHER_ESP32) // Feather Huzzah32
  #define TFT_CS         14
  #define TFT_RST        15
  #define TFT_DC         32

#elif defined(ESP8266)
  #define TFT_CS         4
  #define TFT_RST        16
  #define TFT_DC         5

#else
  // For the breakout board, you can use any 2 or 3 pins.
  // These pins will also work for the 1.8" TFT shield.
  #define TFT_CS        4//4
  #define TFT_RST       24  // Or set to -1 and connect to Arduino RESET pin
  #define TFT_DC        25
#endif

Adafruit_ST7789 tft = Adafruit_ST7789(TFT_CS, TFT_DC, TFT_RST);
//Rot A - 29 - pcint23
//Rot b - 28 - pcint22
//enter2 - 24 - pcint18

//Up - PC5 - 27 - pcint 21
//down - pc4 - 26 - pcint 20
//enter - pc3 - 25 -pcint 19
volatile int buttonPressed = 0;
int lastButton = -1;
int startMenuState = 0;
int allState = 0;
int settingsMenuState = 0;
//int timer_count = 0;
//bluetooth coms
char dataArray[32] =
{'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0'};
int index = 0;
```

```
char c = ' ';

int timer_on = 0;
int timer_count = 0;
int break_timer_on = 0;
int break_timer_count = 0;
int prev_is_sit = 0;
int time_limit_reached = 0;
int resetState = 0;


int box_x;
int box_y;

float p = 3.1415926;
int counter = 0;
int p_counter = -1;
int current_clock;
int lastClock;
int data;

int analogEnabled = 1;
int seat_map[4][2] = {{0,1},{2,0},{0,4},{1,0}};
int chair_map[4][4] = {{0,0,0,0},{0,1,1,0},{2,0,4,0},{4,4,0,0}};
int is_sit = 0;
int temp = 0;

unsigned long pmill = 0;
unsigned long interval = 15000;
void setup(void) {

//  EEPROM.write(TEMP_SAVE,90);
//  EEPROM.write(TIME_HR,2);
//  EEPROM.write(TIME_MIN,5);


Serial1.begin(38400);
  spi_init();
  Serial.begin(9600);
  isd_powerup();

  pinMode(CLOCK,INPUT);
  pinMode(DATA,INPUT);
  // put your setup code here, to run once:
  cli();
  PCICR |= 0b00000100;    // turn on port c
  //PCMSK2 |= 0b00111000; //No analog
  PCMSK2 |= 0b11111100; //yes analog
  sei();

  lastClock = digitalRead(CLOCK);
  Serial.begin(9600);
  Serial.print(F("Hello! ST77xx TFT Test"));

  tft.init(240, 240);           // Init ST7789 240x240
  tft.setRotation(2);           //flip 180
  Serial.println(F("Initialized"));

  uint16_t time = millis();
```

```
  tft.fillScreen(ST77XX_BLACK);
  time = millis() - time;

  Serial.println(time, DEC);
  delay(500);
  StartMenu(0);
}

void loop() {
  unsigned long cmill = millis();
  if(cmill - pmill > interval){
    unsigned long time_start_bt = millis();
    getBT();

    Serial.println( millis()- time_start_bt);
    Serial.println("bluetooth Response time");
    pmill = cmill;
  }
  if(time_limit_reached){
    //print screen and play audio
    tft.fillRect(0,0,240,240,ST77XX_BLACK);
    alertScreen();
    if(is_sit){
      isd_play(1);
      delay(10000);
      isd_stop();
      Serial.println("stop play");
    }
      //draw warning screen

      if(resetState){
        tft.fillRect(0,0,240,240,ST77XX_BLACK);
        StartMenu(0);
        startMenuState = 0;
        allState = 0;
        settingsMenuState = 0;
        resetState = 0;
        time_limit_reached = 0;
      }
  }//check here
else if( buttonPressed!= 0){
  if(allState == 0){ //start menu
    if(buttonPressed == UP){
      startMenuState +=2 ;
      StartMenu(startMenuState%3);
    }else if(buttonPressed == DOWN){
      startMenuState += 1;
      StartMenu(startMenuState%3);
    }
    else if(buttonPressed == ENTER){ //select state from start menu
      switch(startMenuState%3){
        case 0:
          allState = 1;
          tft.fillRect(0,0,240,240,ST77XX_BLACK);
          counter = SettingsMenu(settingsMenuState%3) * 2;
          SettingsMenu(0);
          break;
        case 1:
          allState = 2;
```

```
        tft.fillRect(0,0,240,240,ST77XX_BLACK); // use when changing to and from allstate menu
        DrawChair(seat_map,chair_map);
        break;
      case 2:
        isd_record(1);
        delay(10000);
        isd_stop();
        Serial.println("Stop Record");
        break;
    }
  }
  else if(buttonPressed == ENTER2){
    isd_play(1);
    delay(10000);
    isd_stop();
    Serial.println("stop play");
  }
}
else if(allState == 1){
   if(buttonPressed == UP){
    settingsMenuState +=2 ;
    counter = SettingsMenu(settingsMenuState%3) * 2;
  }else if(buttonPressed == DOWN){
    settingsMenuState += 1;
    counter = SettingsMenu(settingsMenuState%3) * 2;
  }else if(buttonPressed == ENTER){
    allState = 0;
    tft.fillRect(0,0,240,240,ST77XX_BLACK);
    settingsMenuState = 0;
    StartMenu(0);
  }else if(buttonPressed == ENTER2){
    //store in eeprom
    switch(settingsMenuState%3){
      case 0:
        EEPROM.put(TEMP_SAVE,counter/2);
        break;
      case 1:
        EEPROM.put(TIME_HR,counter/2);
        break;
      case 2:
        EEPROM.put(TIME_MIN, counter/2);
        break;
    }
  }
  else if(buttonPressed ==ROTATION){
    switch(settingsMenuState%3){
      case 0:
        box_x = 199;
        box_y = 24;
        if(counter != p_counter && counter%2 == 0){
          if(counter < 0){
            counter = 0;
          }
          updateCounter(counter,box_x,box_y);
          p_counter = counter;
        }
        break;
      case 1:
        box_x = 199;
```

```
          box_y = 72;
          if(counter != p_counter && counter%2 == 0){
           if(counter < 0){
              counter = 0;
            }
            updateCounter(counter,box_x,box_y);
            p_counter = counter;
          }
          break;
        case 2:
          box_x = 199;
          box_y = 119;
          if(counter != p_counter && counter%2 == 0){
           if(counter < 0){
              counter = 0;
            }
            updateCounter(counter,box_x,box_y);
            p_counter = counter;
          }
          break;
      }
    }
  }
  else if(allState == 2){ //view chair menu,
    if(buttonPressed == ENTER2){
      getBT();
    }
    else if(buttonPressed == ENTER){//press to return
      tft.fillRect(0,0,240,240,ST77XX_BLACK); // use when changing to and from allstate menuq
      StartMenu(startMenuState % 3);
      allState = 0;
    }
  }
  buttonPressed = 0;
}

//if(run == 1){
// run = 0;
// DrawChair(seat_map,chair_map);
//}

}
ISR(PCINT2_vect){
  unsigned long interrupt_timer = millis();
  cli();
  if(analogEnabled){
    buttonPressed = ROTATION;
    current_clock = digitalRead(CLOCK);
    data = digitalRead(DATA);
    if(current_clock != lastClock){
      if(data != lastClock){
        counter --;
        lastClock = current_clock;
        sei();
      }
      else{
        counter ++;
        lastClock = current_clock;
        sei();
```

```
      }
    }
  }
  if(!digitalRead(18)){
    Serial.println("ENTER2 PRESSED");
  buttonPressed = ENTER2;
  sei();
 }
  if(!digitalRead(19)){
    Serial.println("ENTER PRESSED");
    buttonPressed = ENTER;
    sei();
  }
  if(!digitalRead(20)){
    Serial.println("DOWN PRESSED");
    buttonPressed = DOWN;
    sei();
  }
  if(!digitalRead(21)){
    Serial.println("UP PRESSED");
    buttonPressed = UP;
    sei();
  }
  Serial.println("interrupt response time");
  Serial.println(millis() - interrupt_timer);
}

void DrawChair(const int s_map[][2], const int c_map[][4]){
  tft.fillRect(70,0,100,120,ST77XX_WHITE);
  tft.fillRect(20,125,200,115,ST77XX_WHITE);
  int color_array[5] = {ST77XX_RED,ST77XX_YELLOW,ST77XX_GREEN,ST77XX_CYAN,ST77XX_BLUE};
  //fill grid
  int x_constant_seat = 46;
  int y_constant_seat = 28;

  int x_constant_chair = 46;
  int y_constant_chair = 28;

  for(int i = 0; i < 4; i++){
    for(int j = 0; j < 2; j++){
    //tft.fillRect(72,1,45,28,ST77XX_GREEN);
    tft.fillRect(73 + (x_constant_seat+2)*j,2 +
(y_constant_seat+1)*i,x_constant_seat,y_constant_seat,color_array[s_map[i][j]]);
    }
  }

  for(int i = 0; i < 4; i++){
    for(int j = 0; j < 4;j++){
      tft.fillRect(25 + (x_constant_seat+2)*j,127 +
(y_constant_chair+1)*i,x_constant_chair,y_constant_chair,color_array[c_map[i][j]]);

    }
  }
  tft.fillRect(0,0,60,100,ST77XX_BLACK);
  tft.setCursor(0,0);
  tft.setTextColor(ST77XX_WHITE);
  tft.print(temp);

}
```

```
void updateCounter(int counter, int x, int y){
  tft.setCursor(x,y);
  tft.fillRect(x-2,y, 60, 40, ST77XX_BLACK);
  tft.setTextColor(ST77XX_WHITE);
  tft.print(counter/2);
}

void StartMenu(int state){
  tft.setCursor(0,0);
  tft.setTextColor(ST77XX_BLACK);
  tft.setTextSize(3);
  //tft.setTextWrap(true);
  //void drawRect(uint16_t x0, uint16_t y0, uint16_t w, uint16_t h, uint16_t color);
  Serial.print("State");
  Serial.println(state);

  if(state == 0){
  tft.fillRect(0,0, 240, 180, ST77XX_BLACK);
  tft.fillRect(0,15,240,40,ST77XX_WHITE);
  tft.println("\n1. Settings");
  tft.setTextColor(ST77XX_WHITE);
  tft.println("\n2. View Chair");
  tft.println("\n3. Audio");
  }
  if(state == 1){
  tft.fillRect(0,0, 240, 180, ST77XX_BLACK);
  tft.setTextColor(ST77XX_WHITE);
  tft.println("\n1. Settings");
  tft.fillRect(0,65,240,40,ST77XX_WHITE);
  tft.setTextColor(ST77XX_BLACK);
  tft.println("\n2. View Chair");
  tft.setTextColor(ST77XX_WHITE);
  tft.println("\n3. Audio");
  }
  if(state == 2){
  tft.fillRect(0,0, 240, 180, ST77XX_BLACK);
  tft.setTextColor(ST77XX_WHITE);
  tft.println("\n1. Settings");
  tft.println("\n2. View Chair");
  tft.fillRect(0,115,240,40,ST77XX_WHITE);
  tft.setTextColor(ST77XX_BLACK);
  tft.println("\n3. Audio");
  tft.setTextColor(ST77XX_WHITE);

  }

}

void alertScreen(){
  tft.setCursor(0,0);
  tft.setTextColor(ST77XX_WHITE);
  tft.setTextSize(4);
  tft.print("TAKE A BREAK");
}

int SettingsMenu(int state){
  tft.setCursor(0,0);
  tft.setTextColor(ST77XX_WHITE);
```

```
  tft.setTextSize(3);
  //tft.setTextWrap(true);
  //void drawRect(uint16_t x0, uint16_t y0, uint16_t w, uint16_t h, uint16_t color);
  Serial.print("Settings");
  Serial.println(state);
  tft.fillRect(0,0, 240, 180, ST77XX_BLACK);
  //tft.fillRect(0,15,240,40,ST77XX_WHITE);
  int temp_save = EEPROM.read(TEMP_SAVE);
  int time_hr = EEPROM.read(TIME_HR);
  int time_min = EEPROM.read(TIME_MIN);
  tft.print("\n  Temp     ");
  tft.println(temp_save);
  tft.print("\n  Time hr  ");
  tft.println(time_hr);
  tft.print("\n  Time min ");
  tft.println(time_min);
  if(state == 0){
    tft.fillTriangle(5,25,5,45,15,35,ST77XX_WHITE);
    return temp_save;
  }
  if(state == 1){
    tft.fillTriangle(5,73,5,93,15,83,ST77XX_WHITE);
    return time_hr;
  }
  if(state == 2){
    tft.fillTriangle(5,120,5,140,15,130,ST77XX_WHITE);
    return time_min;
  }
}

//////////////////////////////////////SPI for sound
char spi_transfer(volatile char data) {
  SPDR = data;
  while (!(SPSR & (1<<SPIF))) { }
  return SPDR;
}

void spi_init() {
  byte clr;
  pinMode(DATAOUT, OUTPUT);
  pinMode(DATAIN, INPUT);
  pinMode(SPICLOCK, OUTPUT);
  pinMode(SLAVESELECT, OUTPUT);

  digitalWrite(SLAVE,HIGH);
  digitalWrite(SLAVESELECT, HIGH); // Disable device

  SPCR = (1<<SPE)|(1<<MSTR);
  clr=SPSR;
  clr=SPDR;
  delay(10);
}

void isd_send_command(unsigned char command, unsigned int address) {
  digitalWrite(SLAVESELECT, LOW);
  spi_transfer(address & 0xff);
  //Serial.println("address Sent");
  spi_transfer(((address & 0xff00) >> 3) | command);
  digitalWrite(SLAVESELECT, HIGH);
```

```
}

void isd_powerdown() {
  Serial.println("Powering down...");
  isd_send_command(ISD_OPCODE_POWERDOWN, 0);
}

void isd_powerup() {
  Serial.println("Powering up...");
  isd_send_command(ISD_OPCODE_POWERUP, 0);
  Serial.println("leaving");
  delay(ISD_POWERUP_DELAY);
  Serial.println("leaving");
}

void isd_play(unsigned int address) {
  Serial.println("Playing...");
  isd_send_command(ISD_OPCODE_SETPLAY, address);
  isd_send_command(ISD_OPCODE_PLAY, 0);
}

void isd_record(unsigned int address) {
  Serial.println("Recording...");
  isd_send_command(ISD_OPCODE_SETREC, address);
  isd_send_command(ISD_OPCODE_REC, 0);
}

void isd_stop() {
  Serial.println("Stopping...");
  isd_send_command(ISD_OPCODE_STOP, 0);
}

void getBT(){

  int run_bt = 1;
  //This is just for simulating the chair. Get rid for final program
  //get data
        //Serial.write('#');
       Serial1.write('#');
//Serial.println("in get_bt");
//end

//% fan on
//$ fan off
  unsigned long cur_millis_sub = millis();
  int timeout_bt = 0;
  while(run_bt){
    //timeout variable
    if(millis() - cur_millis_sub > 3000){
      Serial.println("timeout");
      timeout_bt = 1;
      break;
    }
    //random
if (Serial1.available())
    {
        c = Serial1.read();
        //Serial.print(c);
        if(c > 31){
```

```
        //strncat(dataArray[index],&c,1);
        dataArray[index] = c;
        index++;
        if(index == 30){
          //Serial.println("waiting for #");
          //Serial.println(c);
        }
        if(c == '#'){
          run_bt = 0;
//          for(int i = 0; i < index; i++){
//            //Serial.print(dataArray[i]);
//          }
          index = 0;
        }
    }
    }
  }
//update map
//seat_map[4][2] = {{0,1},{2,0},{0,4},{1,0}};
//int chair_map[4][4] = {{0,0,0,0},{0,1,1,0},{2,0,4,0},{4,4,0,0}};
unsigned long processStart = millis();
if(!timeout_bt){
int count_bt = 0;
for(int i = 0; i < 4; i++){
  for(int j = 0; j < 4; j++){
    chair_map[3-i][j] = dataArray[count_bt]- '0';
    count_bt++;
  }
}

for(int i = 4; i < 8; i++){
  for(int j = 4; j < 6;j++){
    seat_map[i-4][j-4] = dataArray[count_bt] - '0';
    count_bt++;
  }
}

int temp_100 = dataArray[count_bt] - '0';
count_bt++;
int temp_10 = dataArray[count_bt] - '0';
count_bt++;
int temp_1 = dataArray[count_bt] - '0';
count_bt++;
temp = temp_100*100 + temp_10*10 + temp_1;

int saved_temp = EEPROM.read(TEMP_SAVE);


is_sit = dataArray[count_bt] - '0';
//Serial1.write('%');
if(saved_temp > temp){
       //Serial.write('$');
       Serial1.write('$');
       //Serial.println("fan off");
}
else if( saved_temp < temp && is_sit){
       //Serial.write('%');
       Serial1.write('%');
       //Serial.println("fan on");
```

```
}else if(!is_sit){
       //Serial.write('$');
       Serial1.write('$');
       //Serial.println("fan off");
}
//finished receing

if(allState == 2 && !time_limit_reached){
  unsigned long oledTime = millis();
DrawChair(seat_map,chair_map);
Serial.println(millis() - oledTime);
Serial.println("redraw time");
}

  manageTimer();
}

}




//int timer_on = 0;
//int timer_count = 0;
//int break_timer_on = 0;
//int break_timer_count = 0;
void manageTimer(){
//    int time_hr = EEPROM.read(TIME_HR);
//  int time_min = EEPROM.read(TIME_MIN);
//interval = ms
  int timer_limit_seconds = EEPROM.read(TIME_HR)*60*60 + EEPROM.read(TIME_MIN)*60 - 1;
  Serial.println("serial timer limit");
  Serial.println(timer_limit_seconds);

  //done condition

  if(is_sit){
    timer_on = 1;
  }
  if(is_sit && timer_limit_seconds<(interval*timer_count/1000)){
    //timer limit reached
    time_limit_reached = 1;
  }else if(timer_limit_seconds<(interval*timer_count/1000)){
    //on break but time limit reached
    if(is_sit){
      break_timer_count = 0;
    }
    else if(!is_sit){
      break_timer_count++;
    }
  }else if(timer_on){
    if(is_sit){
      timer_count++;
      break_timer_on = 0;
      break_timer_count = 0;
    }else{
      if(prev_is_sit){
        break_timer_on = 1;
      }else{
```

```
        break_timer_count++;
      }
    }
  }else if(!timer_on){
    if(is_sit){
      timer_on = 1;
    }
  }
  prev_is_sit = is_sit;
  Serial.println("break timer");
  Serial.println(break_timer_count);
  Serial.println(break_timer_count*interval/1000);
  Serial.println("timer count");
  Serial.println(timer_count);
  if(break_timer_count*interval/1000 > 1*60-1){
    timer_on = 0;
    timer_count = 0;
    if(time_limit_reached){
      resetState = 1;
    }
    //time_limit_reached = 0;
  }
}
```

Figure 17 : UI Code

```
#include <Adafruit_MLX90614.h>

#define START_1 0
#define START_2 300
#define START_3 500
#define START_4 800
#define START_5 1000

#define ROW0 A0
#define ROW1 A1
#define ROW2 A2
#define ROW3 A3
#define ROW4 A4
#define ROW5 A5

#define COL0 18
#define COL1 19
#define COL2 20
#define COL3 21
#define COL4 22
#define COL5 23
#define COL6 0
#define COL7 1

#define FAN 15

//define TEMP SENSOR STUFF

Adafruit_MLX90614 mlx = Adafruit_MLX90614();

int fsrVal;
```

```
int seat_grid[4][4] = {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
int back_grid[4][2] = {0,0,0,0,0,0,0,0};
int temp = 0;
int is_sit = 0;

//col is gate

int gate[8] = {COL0,COL1,COL2,COL3,COL4,COL5,COL6,COL7};
int inputRead[6] = {ROW0,ROW1,ROW2,ROW3,ROW4,ROW5};


const long baudRate = 38400;
char c=' ';
char dataArray[32] = "00000000000000000000000000000000";
int index = 0;

void setup()
{
    Serial.begin(9600);
    Serial1.begin(baudRate);
    pinMode(COL0, OUTPUT);
    pinMode(COL1, OUTPUT);
    pinMode(COL2, OUTPUT);
    pinMode(COL3, OUTPUT);
    pinMode(COL4, OUTPUT);
    pinMode(COL5, OUTPUT);
    pinMode(COL6, OUTPUT);
    pinMode(COL7, OUTPUT);

    pinMode(ROW0, INPUT);
    pinMode(ROW1, INPUT);
    pinMode(ROW2, INPUT);
    pinMode(ROW3, INPUT);
    pinMode(ROW4, INPUT);
    pinMode(ROW5, INPUT);

    digitalWrite(COL0,LOW);
    digitalWrite(COL1,LOW);
    digitalWrite(COL2,LOW);
    digitalWrite(COL3,LOW);
    digitalWrite(COL4,LOW);
    digitalWrite(COL5,LOW);
    digitalWrite(COL6,LOW);
    digitalWrite(COL7,LOW);

    pinMode(FAN,OUTPUT);
    digitalWrite(FAN,LOW);
if(!mlx.begin()){
  Serial.println("Error connecting to MLX sensor");
  //while(1);
}

}

void loop()
{

    // Read from the Bluetooth module and send to the Arduino Serial Monitor
    if (Serial1.available())
```

```
    {
        c = Serial1.read();
        if(c > 0){
          if(c == '#'){
          //Send Data
            unsigned long allData = millis();
            setFSRData();
            readIRTemp();
            sendData();
            Serial.println(millis() - allData);
            Serial.println("all data read");
            Serial1.write('#');
          }
          if(c == '%'){
            //Turn on fan
            toggleFan(1);
          }
          if(c == '$'){
            toggleFan(0);
            //Turn off fan
          }
    }
    }
}

void setFSRData(){
  //chair seat
  int is_sit_count = 0;
  for(int i = 0; i < 4; i++){
    digitalWrite(gate[i], HIGH);
    for(int j = 0; j < 4; j++){
      delay(20);
      fsrVal = analogRead(inputRead[j]);
      //Serial.println(fsrVal);
      delay(20);
      int mapped_val = returnMapped(fsrVal);
      seat_grid[i][j] = mapped_val;
      if(mapped_val == 4){
        is_sit_count ++;
        //Serial.println("sit detected");
      }
    }
    digitalWrite(gate[i],LOW);
  }
//set is sit
  if(is_sit_count < 7){
    is_sit = 1;
  }else{
    is_sit = 0;
  }
  is_sit_count = 0;
//Serial.print("is_sit ");
//Serial.println(is_sit);
  //chair back
  for(int i = 4; i < 8; i++){
    digitalWrite(gate[i],HIGH);
    for(int j = 4; j < 6; j++){
      delay(20);
      fsrVal = analogRead(inputRead[j]);
```

```
      //Serial.println(fsrVal);
      delay(20);
      back_grid[i-4][j-4] = returnMapped(fsrVal);
    }
    digitalWrite(gate[i],LOW);
  }

}

int returnMapped(int val){
  if(val < START_2){
    return 0;
  }else if( val < START_3){
    return 1;
  }else if( val < START_4){
    return 2;
  }else if( val < START_5){
    return 3;
  }else{
    return 4;
  }
}

void readIRTemp(){
  Serial.print("Ambient = "); Serial.print(mlx.readAmbientTempC());
  Serial.print("*C\tObject = "); Serial.print(mlx.readObjectTempC()); Serial.println("*C");
  Serial.print("Ambient = "); Serial.print(mlx.readAmbientTempF());
  Serial.print("*F\tObject = "); Serial.print(mlx.readObjectTempF()); Serial.println("*F");

  temp = mlx.readAmbientTempF();
}

void sendData(){
  //seat - 16 digits
  for(int i = 0; i < 4; i++){
    for(int j = 0; j < 4; j++){
      Serial.print(seat_grid[i][j]);
      Serial1.write(seat_grid[i][j] + '0');
    }
  }
  //chair - 8 digits
  for(int i = 0; i < 4; i++){
    for(int j = 0; j < 2; j++){
      Serial1.write(back_grid[i][j] + '0');
      Serial.print(back_grid[i][j]);
    }
  }
    //temp - 3 digits - tested
    Serial1.write(temp/100 + '0');
    Serial1.write((temp%100)/10 + '0');
    Serial1.write((temp%10) + '0');

  //sitting on chair - 1 digit
    Serial1.write(is_sit  + '0');

    Serial.println(temp/100);
    Serial.println((temp%100)/10);
    Serial.println((temp%10));
}
```

```
void toggleFan(int toggle){
  if(toggle){
    digitalWrite(FAN,HIGH);
  }
  else{
    digitalWrite(FAN,LOW);
  }
}
```

Figure 18: On Chair Code