King of Kings TechBed

Senior Design

Miami University 2020-2021

ENT 497 - 498

Academic Advisor: Reza AbrishamBaf, PhD

Team Members: Jacob Klopfenstein, Chris Waidelich

May 10, 2021

Executive Summary:

The purpose of this project is to create a new king size bed that will stand apart from everything else on the market by incorporating a sound system, minifridge, and other Raspberry Pi driven technology into a bedframe that has a natural wood appearance, functional design, and built-in drawers beneath the mattress for efficient floor space usage. The idea was chosen because it included aspects in both fields that Electro-Mechanical Engineering Technologies covers: mechanical engineering for the design of the bed structure and electrical engineering for the design and integration of technology and Raspberry Pi circuitry. Designing and building this bed will gave both valuable experience and knowledge to the members of this team and made them more competent in their relative fields of study.

Table of Contents

Table of Figures

Design Research

Research began by searching online for other technology integrated bedframes that were on the market, and it quickly became clear that there were not many options to choose from. A bedframe, at its core, is a very simple structure when its only task is to support a mattress, so little creativity has been placed into it for the mass market. The closest thing to the design of the *King of Kings TechBed* was the *Ultimate Bed* [1], which sported a speaker system and some storage under the bed, as well as other technologies and features the *King of Kings TechBed* lacked. However, the style was very modern, and according to the *King of Kings* group, quite ugly. It lacked the natural wooden appearance the *King of Kings* group was aiming to achieve.

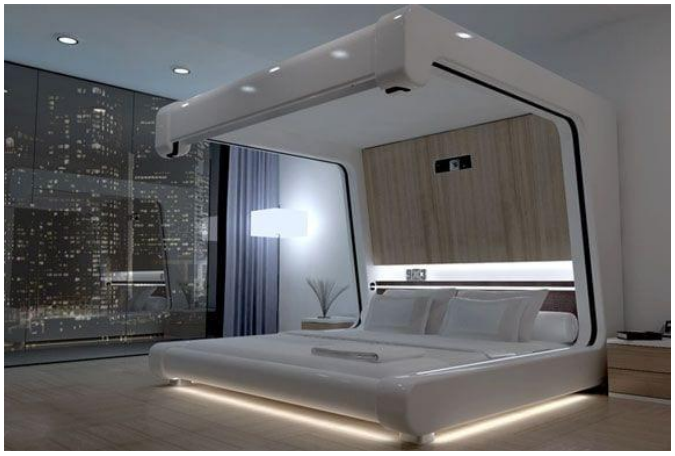


Figure 1: The Somnus-Neu Bedframe [2]

Another website [2] discussed various technology bedframes, but all appeared impractical and too modern for the style to last, such as was the case for the Somnus-Neu Bedframe (see figure 1). It was also unclear how many of the bedframes presented in the article were actually available for consumers verses being designs that didn't make it to production due to low demand. Finally, no bedframes contained the concept of a minifridge built into the bed, and very few had the RGBW lighting system similar to what the *Kings of Kings TechBed* would display.

After the initial research on bedframe concepts already available, the group began to explore the technology wanted in the bed and identifying which product should be designed around. The first technology piece explored was the minifridge. While a freezer or icemaker was originally desired, the size of minifridges available that sported such features, such as the Frigidaire 3.1 cubic foot mini fridge [3], were far too large to be properly integrated into the bed. There was also too much space available in the fridge, leading to inefficient use of furniture space. Ultimately, the group settled on using an Insignia 1.7 cubic foot mini fridge [4]. While this meant the bed would not be able to have a freezer, and therefore ice, it also meant the bed could be designed to incorporate the minifridge underneath the bed as opposed to a nightstand attachment. This particular minifridge also was made by a reputable brand and had a lot of good reviews.

The second piece of technology chosen was the sound system. A few of the features desired were HDMI input and output for integration with a projector, a surround sound layout, and a subwoofer. While initial plans were to buy the mixer and speakers separately, the decision was made to choose a soundbar for the main speakers and place it in the footboard. With the new parameter of a soundbar, the Vizio 5.1.2 Home Theatre Soundbar System [5] became the obvious choice for the design. Another advantage of the system was a relatively low cost, smaller dimensioned speakers for easier integration, and multiple input options for both a projector and Raspberry Pi.

The third main piece of technology researched was the lights. Originally, the group was researching six-pin LED strips, such as the Giderwel RGBWW LED Strip Lights [6], but the problems with that option quickly became clear. First of all, the light sections were overly spread out, leading to inconsistent lighting and obvious gaps. Second, the entire strip would be the same, as there was no option to individually access the LEDs. Third, the strips were too wide to easily install around the bedframe. In need of knowledge on lighting options, the *King of Kings* group met with [7] Mike Kwaitkowski, an electrical engineering technologies instructor at Northwest State Community College. He directed us to Adafruit NeoPixels [8] as our choice lighting option. These lights would solve all the problems brought up by the Giderwel option, along with providing more resources for assistance when programming the software.

The final main piece of technology that needed chosen was the different fans that would be implemented on the bed. However, two different parameters existed for these fans. The ones on the headboard needed to be powerful, capable of moving massive amounts of air in order to create a stead breeze over the sleepers. It also didn't matter if they were louder, as that would provide a background noise for sleepers that would be off in non-sleeping hours. However, the fans to cool the minifridge compartments and electronics bay needed to be slow, steady, and quiet, as they would be on consistently. Using these parameters, the group choose to use Highfine 120mm 4000RPM CPU Cooling Fans [9] for the headboard and Antec 120mm case fans [10] for the constant cooling

Figure 2: Highfine 120mm CPU Fan [9]

operations. This provided an added benefit of economical purchasing, as the Antec fans came in a pack of five, the number needed.

The decision for which fans to use in headboard later changed to Wathai 12038 fans [15] instead of the original Highfine 120mm fans. There were several factors in making the change, with the largest being the desire for the Raspberry Pi to be capable of controlling the speed of the fans. The newly selected fans have a four-pin connection, while the Highfine fans were a three-pin connection. The added pin is a Pulse Width Modulation pin, or PWM. This is what will allow the Raspberry Pi to enact control over the fan's RPM. The specifications for the fan remained at 12V, but with an increased RPM from 4000RPM to 5300RPM. The chosen 12V power supply unit was also able to power these fans, so the fans ended up being the only parts that were changed on the project after purchasing.

A Raspberry Pi 4 Model B [11] will control the various technologies through an interface with the Raspberry Pi official 7in touch screen [12]. This version of Raspberry Pi was chosen to make sure it had as much power as possible for future-proof purposes, as well as enough RAM for a complex program while still being able to operate a touch screen. The official 7in touch screen was a perfect combination, as it will allow for smooth interfacing with the control systems and will eliminate needs to labels buttons, switches, and potentiometers had they been physical.

Finally, the *King of Kings* group had to learn about woodworking and wood design of large furniture. Obviously, this level of information is difficult to learn in an online format, so various experts in the field were consulted on the matter. The first consultant was Randy Wise [13], an engineering consultant and construction manager. His information consisted of various wood types and their advantages and disadvantages. Ultimately,

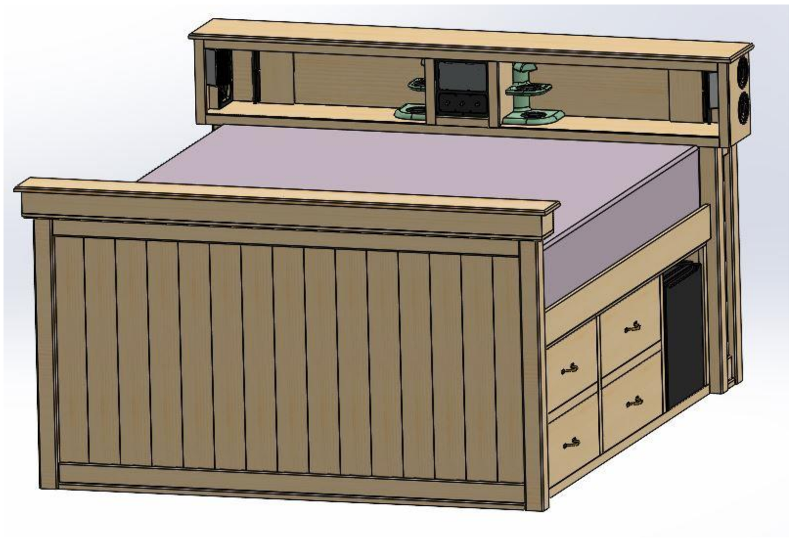


Figure 3: Final Model of Bed

it was his advice that led the *King of Kings* group to use pine in the design as opposed to a finer wood type that would be stronger but cost much more.

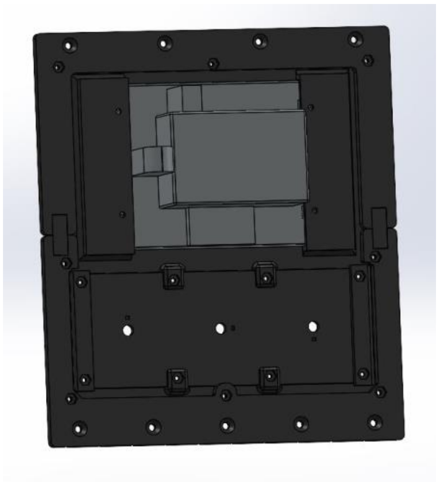


Figure 5: Design for Touch Screen Mount

The second main resource on wood design was Andrew Shultz [14], a cabinet builder at Lanz Construction in Mediapolis, Iowa. His instruction centered around designing for various wood fasting techniques on furniture, such as pocket screws and where to use wood glue. He also trained the group on the importance of "facing" a piece of furniture by placing a wooden frame on the front of a structure to hide any wood ends for appearance purposes.

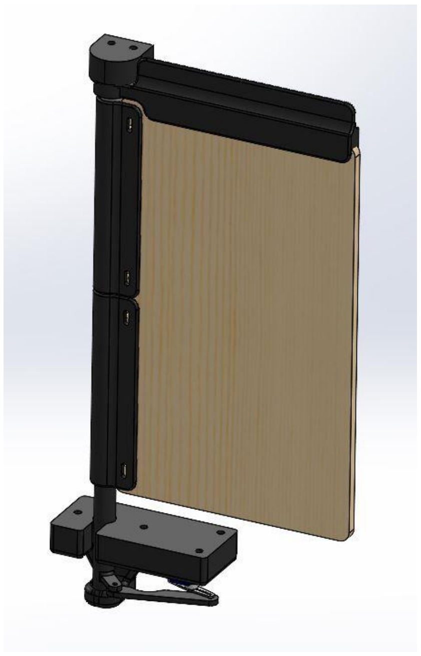


Figure 4: Design for Air Deflector

After the large technology pieces were chosen and the knowledge of wood design had been gained, the *King of Kings* group began to develop the design of the bedframe. Several of the main design points included designing the bed for disassembly into components that could be easily moved, for the wiring of various components, and for proper housing and cooling of electronics. For designing purposes, a model was put together in SOLIDWORKS 2020, which supplied the group with a complete rendering of what the bed would look like and what sizes all of the parts would be. It also created a virtual environment that could be used when doing the detailed designing of several 3D printed components.

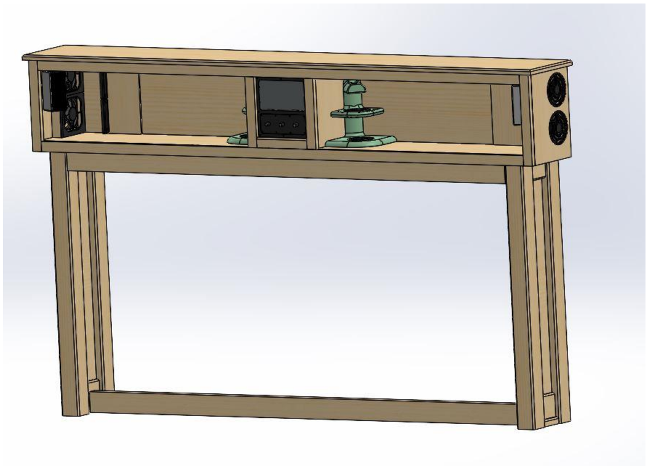


Figure 6: Design of Headboard

For the structure of the bed itself, little attention was paid to the specific weight distribution and strength of the frame. The reason for this was simple. A bedframe is meant to hold a mattress and occupants. At most, it will only experience a few hundred pounds of weight. Thus, in our design, the goal was to simply overbuild it strength-wise, as we needed the space and style of strength anyway. Plywood framing around the drawers and minifridge provided a column grid for weight distribution to the floor of the room, and a plywood top for the mattress to sit on, as opposed to boards spaced evenly across, provided a cheaper and more consistent weight distribution solution. The resulting design didn’t have a weight limit, but it was very clear that the bed is built strong enough to hold way more weight than it will ever experience. Thus, with all considerations taken into account during the design phase, no specific calculations needed to be done.

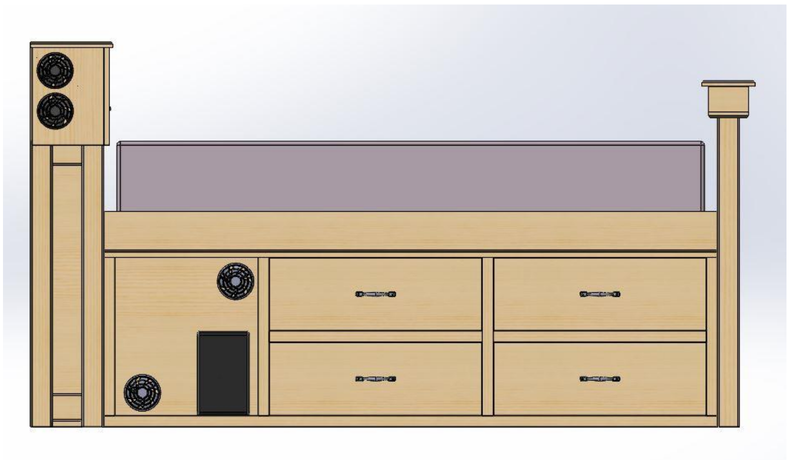


Figure 7: Final Model of Bed Side View

Due to a minifridge not being designed to be in an enclosed space, fans were built into the back of the minifridge boxes for airflow and cooling purposes. As we wanted the bed to be capable of having the minifridge being on either side of the bed, we built the main frame modules as mirror images of each other. These fans are controlled by their own hard-wired switch to prevent a case of improper ventilation due to a computer glitch. The design was made so the subwoofer for the sound system could be stored in the minifridge box that wasn’t being used. This also allowed the heavy vibrations to be transferred into the bedframe itself, thus giving the experience of listening to audio through the sound system more exhilarating.

The resulting design fit all of our specifications. The bedframe breaks down into smaller modules for moving purposes: footboard, headboard, two main frame halves, and the mattress and top plywood boards. Each piece, while large in size, can fit through modern bedroom doors with careful guidance. The frame also neatly integrates the technology pieces as desired.

Step by Step Plan:

Developing the Design

A. Research Existing Frame Designs
B. Choose Technology to Integrate
    i. Sound system
        1. Surround sound
        2. Theatre capable
        3. Easily integrated into bedframe
    ii. Minifridge
        1. Compact
        2. Large enough to hold a few drinks and snack food
        3. Door that can hinge from either side
    iii. Strip lights
        1. RGBW capable
        2. 100% dimmable
        3. Can be cut into sections and connected together
        4. Can be interfaced with Raspberry Pi
            a. Can smoothly change colors
            b. Possible unique light patterns
            c. All lighting options controlled by main control panel
    iv. Fans for cooling sleepers and minifridge
        1. Small and compact
        2. Easily interfaced on control panel
        3. Controllable fan speed
        4. Run quietly
    v. Raspberry Pi
        1. Interfaced into control panel on headboard
            a. Touchscreen for main interface
            b. Possible potentiometer for brightness
            c. Buttons for color mode selection
            d. Possible light automation

            i. Lights automatically turn on in the morning
            ii. Lights automatically turn on in the evening
        2. Control fans, lights, and more
            a. User can control fan speed
            b. User can control light color and brightness
            c. User can turn on and off outlets on bed

C. Research Designing Products out of Wood
    i. What sizes of wood are available
    ii. What type of wood should be used
        1. Pine is the most economical option
        2. Wood can either be clear (no knots) or knotty
    iii. Fastening
        1. Components can be fastened with pockets screws
        2. Wood glue works in some areas, but wood expansion must be considered
        3. Avoid showing grain-ends when possible
        4. Large surfaces can be made with Cabinet Quality pine plywood
    iv. Most abnormal sizes can be obtained with a planer and a table saw

D. Design Modular Bedframe
    i. Headboard
        1. Can detach from bed
        2. Have space for lighting
        3. Have electronics bay for control panel and any wiring
    ii. Footboard
        1. Can detach from bed
        2. Designed for strip lights along the back
        3. Space for built in sound bar
        4. Designed for easy access to wiring
    iii. Two Mainframe Modules
        1. Minifridge must be incorporated
        2. Minifridge must be able to go on either side of bed

3. Drawers should be built in to make use of excess space under bed

E. Design Electrical Controls
    i. One main plug to wall outlet
    ii. Two uncontrolled outlet boxes (for fridge, sound system, and any other always-on electronics)
    iii. Two controlled outlet boxes (controlled by relays connected to Raspberry Pi)
    iv. Power supply for fans
        1. Fans consist of four sets of two
            a. One set of two on either side of the headboard
            b. One set of two for each minifridge compartment
    v. Power supply for NeoPixels

Construction Logistics Plan

F. Transportation
    i. Large, enclosed trailer
        1. Borrow from Phillip Jackson Family
    ii. Truck to pull trailer with
        1. Chris Waidelich or Jerry Klopfenstein

G. Construction Location
    i. Rough board cutting
        1. Waidelich Shed
        2. Rocke Wood Shop (IL)
        3. Lanz Construction (IA)
    ii. Board finishing and fastening
        1. Lanz Construction (IA)

H. Electrical Assembly and Project Staging
    i. Northwest State Community College

Construction of Bedframe

I. Footboard
    i. Prepare main posts
        1. Add grooves and chamfers

        2. Cut posts down to size
    ii. Tongue & Groove wall
        1. Add grooves and chamfers to boards and cut to length
        2. Glue top channel together and the bottom channel together
        3. Build speaker box base and attach to top channel
        4. Attach posts to footboard base
        5. Lay rubber/foam compression strip
        6. Assemble T&G wall and secure with speaker box base/top channel
    iii. Build speaker box
        1. Cut side boards to shape and mount
        2. Add top mounting blocks
        3. Build footboard top
            a. Add chamfers and finish wood
            b. Resin in screws to align with mounting holes
    iv. Final Assembly
        1. Make sure all parts fit together
        2. Sand any rough areas smooth
        3. Add finish

J. Headboard
    i. Main Frame
        1. Prepare Main Posts
        2. Use extra T&G channels for sides
        3. Pocket screw cross members to posts
    ii. Headboard Housing
        1. Cut base to size and secure to frame
        2. Add dividers
        3. Cut and mount front face frame
        4. Cut, glue, and mount top frame
        5. Screw on headboard top
        6. Secure backing plywood
    iii. Final Assembly

1. Make sure all parts are fitting correctly and the frame is within dimension
2. Cut holes for electrical wires and NeoPixels
3. Sand any rough areas smooth
4. Add finish

K. Right and Left-hand Frame Modules
   - i. Cut plywood sheets to size
   - ii. Construct main beams
   - iii. Cut top frame pieces and secure together
   - iv. Cut front frame pieces and secure together
   - v. Main assembly
     1. Construct frame from prepared plywood pieces
     2. Build minifridge box
        - a. Cut holes for cooling fans
        - b. Finish interior
     3. Add front and top faces to plywood frame
     4. Add drawers
        - a. Mount drawer rails
        - b. Mount drawer faces to drawer boxes
        - c. Add drawer handles
     5. Check fit dimensions
     6. Ensure modules fit together well
     7. Sand any rough areas
     8. Add finish

L. Final Assembly
   - i. Build fan grids
     1. 3D print fan grids
     2. Install in proper holes on frame
   - ii. Assemble all modules
     1. Check module fit with each other
     2. Make final holes and pathways for electrical wiring

Installation of Electrical Components

M. Main Electrical Outlets
   i. Main cord to junction box
      1. Power leading to footboard for speaker bar
   ii. Mount electrical outlet boxes
      1. Two directly from junction box
      2. Two through relays controlled by Raspberry Pi

N. Installation of Sound System and Minifridge
   i. Sound system
      1. Test sound system outside of bedframe to ensure function
      2. Mount speaker bar in footboard
      3. Mount subwoofer under center of bed
      4. Mount surround speakers in headboard
      5. Run HDMI cables and speaker wires
   ii. Minifridge
      1. Test minifridge to ensure function
      2. Mount minifridge in compartment
      3. Run minifridge and monitor temperature with and without fans

O. Programming of Raspberry Pi (outside of bedframe)
   i. Touchscreen
      1. Develop program for interfacing with touchscreen
      2. Test program and work out bugs
   ii. Relay-controlled outlets
      1. Add program to activate and deactivate relays from touch screen
      2. Test and debug program
   iii. NeoPixel Strips
      1. Develop program for controlling NeoPixels from touchscreen
      2. Test and debug program
      3. Cut NeoPixels to needed lengths and connect into replica of what it will be on bed

4. Develop program and interface to control the different sectors separately
    a. Headboard left
    b. Headboard Right
    c. Around the Frame
5. Test and debug control program
6. Develop program for passive color patterns
7. Test and debug program

iv. Fan controllers

1. Minifridge fans
    a. Develop program to passively control cooling fans for minifridge
    b. Test and debug program
2. Sleeper cooling fans
    a. Develop program to control fans for headboard
    b. Develop interface for head fan controls
    c. Test and debug program

P. Installation of Raspberry Pi

i. Mount the Raspberry Pi in the control box

ii. Make control panel

1. 3D Print control panel
2. Secure touch screen to control panel
3. Mount buttons/potentiometers to panel
4. Install panel on headboard

iii. Connect control panel connections to Raspberry Pi

iv. Secure any circuit boards into control box

Q. Installation of NeoPixel Strips

i. Install the NeoPixel strips around the edge of the bedframe

ii. Install the NeoPixel strips in left and right compartments

iii. Connect strips back to Raspberry Pi

R. Installation of Cooling fans

        i. Screw all eight fans into location
        ii. Connect wires back to control box

Final Testing and Staging

S. Test Sound System and Minifridge
    i. Ensure sound system works through various input sources
    ii. Ensure minifridge is cooling properly
    iii. Ensure minifridge isn't overheating and cooling fans are working properly

T. Test Raspberry Pi Interface
    i. Test relay interface
    ii. Test fan interface
        1. Left and right fan sets should work separately
        2. Speed control is working properly
        3. Fans can be turned off
        4. Fans with fridge should run automatically
        5. Fans with empty compartment should be left off
    iii. Test NeoPixels
        1. Test various passive color patters
        2. Make sure the different sectors can be turned off independently
        3. Test brightness adjustment on all sectors

U. Assemble Bed in Staging Area
    i. Bed frame fully assembled in staging area
    ii. Make bed
        1. Place mattress on bed
        2. Add sheets, pillows, and bedspread
        3. Add accessories to headboard, snacks to fridge
        4. Display lighting

<u>Project Timeline:</u>

This project was the responsibility of both members of the group, and each one participated in some way on each task. Each phase of project development was assigned to one person to take the lead on that specific phase, but the other team member was still actively involved with that task as it progressed, allowing for the gain in knowledge and experience to be shared by all team members.

Member Responsibilities:

- Jacob Klopfenstein:
  - Mechanical Structure Design
  - Wood Construction
  - Transportation/Travel
  - Electrical Wiring and Troubleshooting
  - Papers/Progress Reports
- Chris Waidelich:
  - Electrical Design and Schematic
  - Raspberry Pi Programming
  - GUI Development
  - Program Debugging
  - Slideshows/Presentations

Throughout the project, there were various deviances from the originally crafted timeline. All of these instances related back to lack of experience and knowledge of exactly how the project would be executed and what components would take more time. As seen in the Gantt chart for the project, the construction of the bed and the programming was supposed to be run congruently, with equal amounts of time divided for each piece of technology. In reality, the construction of the bedframe started a little late (January 4th, as opposed to the projected January 1st), yet still finished months early (February 17th, as opposed to April 1st).

Meanwhile, the electrical install and programming progress didn’t go according to the schedule at all. Instead, the GUI was developed first, with simple tasks like the fans and relay controls being completed in a few days. The lights, however, took a majority of the programming time, due to major complications with Raspberry Pi control integration. While a solution was found, the total programming time was longer in that area than expected. Wiring was installed for one component at a time instead of all at once as originally projected. This was done in order to test the programming for each section, which was very beneficial to the project. Obviously, all of these factors were not reflected on the original timeline.

Finally, a few additional software projects were added. An alarm clock feature with custom alarms, ambient sound menu, and more advanced light menus to allow for a color wheel and custom animations all extended the programming time even more than originally planned. In the end, even with all the delays and changes, the project finished on April 14th instead of the projected April 22.

Please Refer to Gantt Chart in Appendix C

Cost Projection:

The projected cost for the technology that will integrated into the bed was approximately $2,183. This included the major prefabricated components such as the minifridge, sound system, lights, mattress, and other components.

The projected cost for the hardware needed to construct the bed was approximately $415. This include all major fasteners, including screws, bolts, drawer slides (side mount), and corner brackets for attaching the headboard and footboard to the main frames.

The projected cost for the electrical components needed to wire and power the bed was approximately $250. This included various wires, connectors and plugs, receptacles, and power supply units to power each of the bed’s components.

Finally, the projected cost for the stock wood supplies needed to construct this bed was approximately $2,005. This included the assembled drawer boxes, plywood sheets, and various stock pine boards that would be cut to shape during construction.

In total, the projected cost of the completed bed was approximately $4,850. The main source of funding was the *Armin Fleck Senior Design Scholarship*, which provided *King of Kings TechBed* with $4,000 toward the project. All additional funding was supplied by the team members. Below is the BOM for the project and all the parts it was projected to require and the respective prices of the components:

Technology Products:	Quantity:	Cost:	Total Cost:
VIZIO 36" 5.1.2 Home Theater Sound System with Dolby Atmos® | SB36512-F6	1	$ 499.99	$ 499.99
Insignia™ - 1.7 Cu. Ft. Mini Fridge - Black Model:NS-CF17BK9	1	$ 99.99	$ 99.99
Adafruit NeoPixel Digital RGBW LED Strip - White PCB 60 LED/m PRODUCT ID: 2842 (4 meters)	2	$ 107.80	$ 215.60
Adafruit NeoPixel Digital RGBW LED Strip - White PCB 60 LED/m PRODUCT ID: 2842 (2 meters)	1	$ 53.90	$ 53.90
Raspberry Pi 4 Model B - 8GB RAM	1	$ 75.00	$ 75.00
Raspberry Pi 7" Touch Screen Display PRODUCT ID: 2718	1	$ 79.95	$ 79.95
Pair of Bedside Reading Lamps Deeplite	1	$ 29.99	$ 29.99
Wireless Charger Set	2	$ 24.99	$ 49.98
Noctua NF-P12 redux-1700 PWM, High Performance Cooling Fan, 4-Pin, 1700 RPM (120mm, Grey)	4	$ 13.90	$ 55.60
Antec 120mm Case Fan, PC Case Fan High Performance, 3-pin Connector, PF12 Series 5 Packs	1	$ 24.99	$ 24.99
Nectar King Size Mattress	1	$ 999.00	$ 999.00
Total Technology Cost:	**$**		**2,183.99**
Hardware/Fasteners:	**Quantity:**	**Cost:**	**Total Cost:**
1"x1" Key-hole Slide Corner Bracket	8	$ 4.00	$ 32.00
5/16" x 1.5"Lag Bolts (4pk)	4	$ 1.31	$ 5.24
Kreg Pocket Screws - 1-1/4", #8 Coarse, Washer-Head, 500ct	1	$ 14.99	$ 14.99
32" Drawer Sliders USF-NJ81A02-31	8	$ 41.98	$ 335.84
eBoot 100 Pieces Adhesive Cable Clips	1	$ 7.25	$ 7.25
Various other washers, nuts, and bolts	1	$ 20.00	$ 20.00
Total Hardware Cost:	**$**		**415.32**
Electrical Supplies:	**Quantity:**	**Cost:**	**Total Cost:**
1875W Flat Plug Extension Cord 20 Feet 14 AWG 15A Black 20FT	1	$ 17.99	$ 17.99
Junction Box	1	$ 1.99	$ 1.99
4 in. x 2 in. Electrical Box	5	$ 0.99	$ 4.95

Wire Duplex Outlet	4	$ 4.49	$ 17.96
14 AWG Electrical Wire (25')	1	$ 14.99	$ 14.99
20 AWG Electrical Wire (100' spool)	1	$ 13.95	$ 13.95
5 Way Spring Terminal Block (20pk)	1	$ 11.89	$ 11.89
3 Way Spring Terminal Block (20pk)	1	$ 9.59	$ 9.59
2 Way Spring Terminal Block (20pk)	1	$ 10.79	$ 10.79
Controlled Relay (3 pack)	1	$ 7.99	$ 7.99
Solder Wire	1	$ 8.00	$ 8.00
Wire Shrink	2	$ 1.99	$ 3.98
3-pin JST SM Plug + Receptacle Cable Set	8	$ 1.50	$ 12.00
Various Other Electrical Components	1	$ 30.00	$ 30.00
12V 5A Power Supply	1	$ 11.19	$ 11.19
Aclorol 5V 40A 200W Power Supply	1	$ 22.49	$ 22.49
Raspberry Pi USB-C 5V Power Supply	1	$ 10.99	$ 10.99
High-Speed Male to Female HDMI Extension Cable - 15 Feet	1	$ 14.99	$ 14.99
High-Speed Male to Female HDMI Extension Cable - 6 Feet	1	$ 7.50	$ 7.50
High-Speed HDMI Cable, 15 Feet	1	$ 11.99	$ 11.99
High-Speed HDMI Cable, 6 Feet	1	$ 7.00	$ 7.00
Total Electrical Supply Cost:	**$ 252.22**		
Wood/Boards:	**Quantity:**	**Cost:**	**Total Cost:**
.75" x 11.25" x 10' Pine Board	4	$ 79.17	$ 316.68
.75" x 7.25" x 8' Pine Board	1	$ 34.36	$ 34.36
.75" x 5.5" x 8' Pine Board	3	$ 24.65	$ 73.95
.75" x 3.5" x 8' Pine Board	8	$ 15.13	$ 121.04
.75" x 1.5" x 10' Pine Board	16	$ 8.68	$ 138.88
3" x 3" x 8' Pine Board	3	$ 58.22	$ 174.66
1.5" x 3.5" x 8' Pine Board	4	$ 25.26	$ 101.04
Tongue & Groove 7\16" x 5" x 10' Pine Boards	7	$ 19.17	$ 134.19
Drawer Box 32" x 9.675" x 27.8125" Pine with Clear Finish	8	$ 66.26	$ 530.08
1/4" x 4' x 8' Appearance Grade Pine Plywood	1	$ 26.99	$ 26.99

15/32" x 4' x 8' Pine Plywood Grade A	1	$ 48.93	$ 48.93
15/32" x 4' x 8' Pine Plywood Grade BC	8	$ 38.00	$ 304.00
Total Wood Cost:	**$ 2,004.80**		

Total Overall Cost:	**$ 4,856.33**

The actual costs ended up being higher than originally projected. While some of the technology, including the sound system and minifridge, were bought at discounted prices due to a sale at the time of purchase, the higher cost of headboard fans, added Apple Watch chargers, more LED lights for the charger mounts, and other costs pushed the actual technology cost to around $2,375. Underestimations in electrical supplies and hardware costs were also higher, coming in at around 365 for electrical supplies and $425 for hardware. The wood budget was approximately $1975, coming in slightly underbudget.

The main source of overbudgeting the project again stems from lack of knowledge and experience, along with unforeseen events that left added cost to the project. First, a trailer was promised to us by friends to use when transporting the bed to Ohio from Iowa where it was assembled, but last minute the trailer was deemed busy and we couldn't use it. This led to a $250 transportation charge to rent a trailer instead. Second, lack of experience in wood working led us to neglect to include the cost of finishing supplies. This includes the wood stain, sealer, and finish product, which totaled to an additional $100. Finally, the biggest cost that we neglected to factor in was tool and shop space rental for the construction process. Originally, we believed we could build the project with tools we had, but it became clear that wasn't possible. We ended up renting some space and shop tools in Iowa, which added an additional $500 to the project. The total added costs to the project were approximately $1000 higher than expected.

Building of Project

The King of Kings TechBed was finished on April 17th, 2021, when the final installments were completed and the last major software bugs were patched. Seeing the bed go from a concept, then to a design, and finally into a finished product that is both "cool" and usable was an amazing experience for both team members.

Construction of Bed Frame

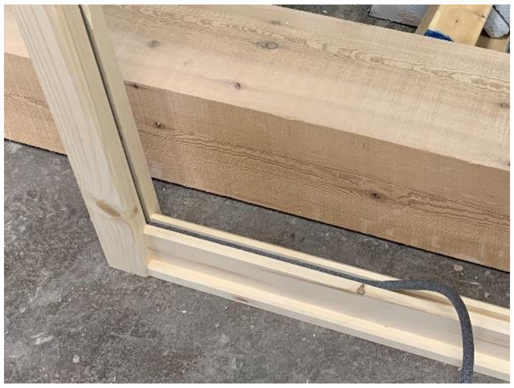


Figure 8: Foam Strip

The process of building the bed began with the framing of the footboard. The design consisted of a frame build around tongue and groove paneling. This frame had a groove all the way around it for the T&G to sit in, with a foam weatherstrip placed between the T&G paneling and frame to keep pressure on the paneling while allowing for the wood to expand and contract due to temperature and humidity without damaging the frame.

Next, holes were drilled in the posts that began from the top of the posts and exited below where the mattress would sit, allowing for wires to be passed through. The top beam of the frame also held a groove for the lights that would be placed in the footboard, along with hidden wire channels for powering and controlling the lights. These grooves were cut with a combination of dado blade and band saw. Once the bottom portion of the footboard was finished, the speaker box was constructed and secured on top. The top of the footboard is removable to allow for access in during the wiring process. The way this works is bolts protruding downward from the footboard top that fit through holes in framing on the speaker box, after which nuts can be threaded on to secure the top in place. Finally, bolts were placed in the side of the posts where the main frame will attach via the custom brackets.

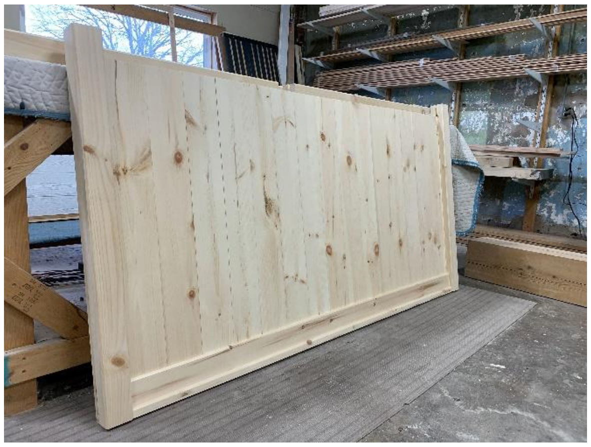


Figure 10: Footboard Framing and T & G

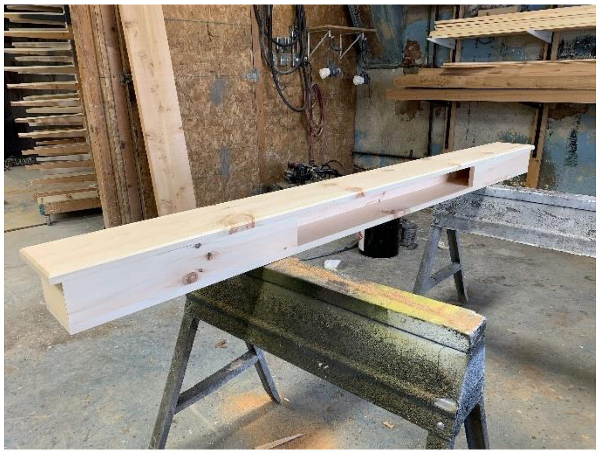


Figure 9: Speaker Box

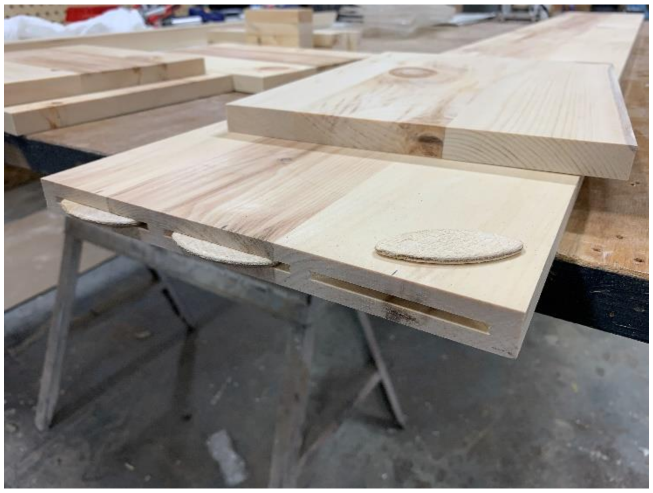


Figure 11: Biscuit Connections

The headboard again began with a base frame, with the side panels using excess lumber from the footboard to made an exact copy of the footboard framing style for the lights there. In this case, holes had to be carefully drilled in the back for the wiring channels, as the access for the wires to enter and exit were in back of the pieces instead of the top like the footboard. A post in the center of the frame was discussed, but it was decided as unnecessary due to the headboard box itself acting as a beam. Any bowing and warping that would take place would be fairly minimal.

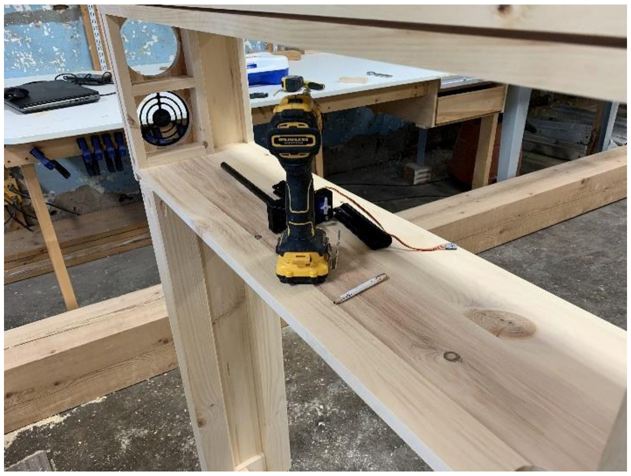


Figure 13: Drilling Holes for Fans

The headboard box was then constructed. Because the joints were not exact corners, and we didn’t want any screw heads exposed for aesthetic purposes, we opted to use biscuit connectors for these points.

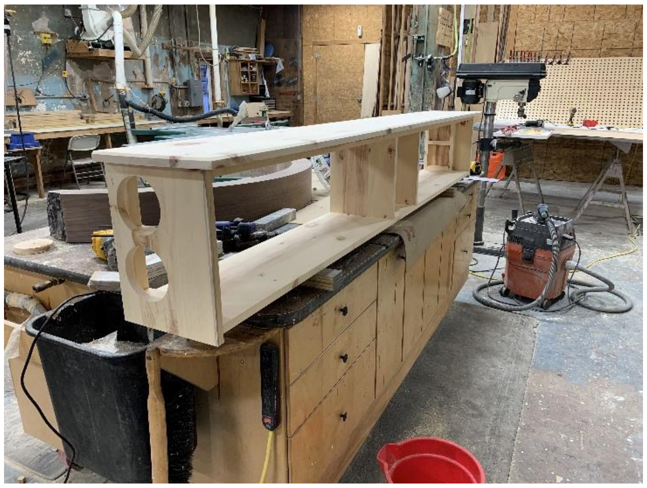


Figure 12: Headboard Box

Grooves were cut into both pieces where the connection would take place, and wooden “biscuits” inserted. With large amounts of wood glue to ensure a solid connection, everything was clamped tightly. While the joints worked very well, the clamps were not placed properly, and without adequate support in the center for the way the box was sitting while still drying, a slight warp was developed. Measured to be around a quarter inch, the warpage wasn’t enough to redo the build, but enough that the trained eye could spot. The warpage was reduced as the rest of the construction took place and the proper support was added to the headboard box design, but it was never fully removed.

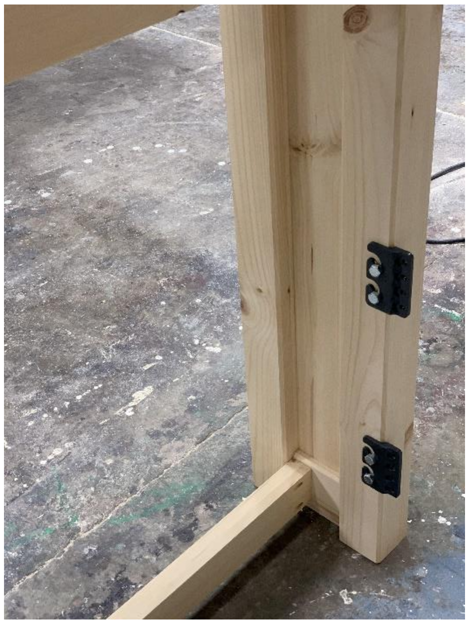


Figure 15: Mounting Brackets

Adding the face frame helped reduce the warpage even more, as the frame both aids appearances, hides any end grain, and acts as support beams for the structural integrity of the headboard box. The top frame was also placed on, completing the large construction phases for the headboard box. It was decided that it would be easier to mount the top of the headboard box after the wood was finished. Holes were then drilled in the sides of the headboard box and the framing glued in place for the headboard fans. Finally, the sanding process took place to prepare the headboard for finishing. It should be noted the headboard box backing was left off for the same reason as the top, as mounting after finishing the wood would be a much better process.

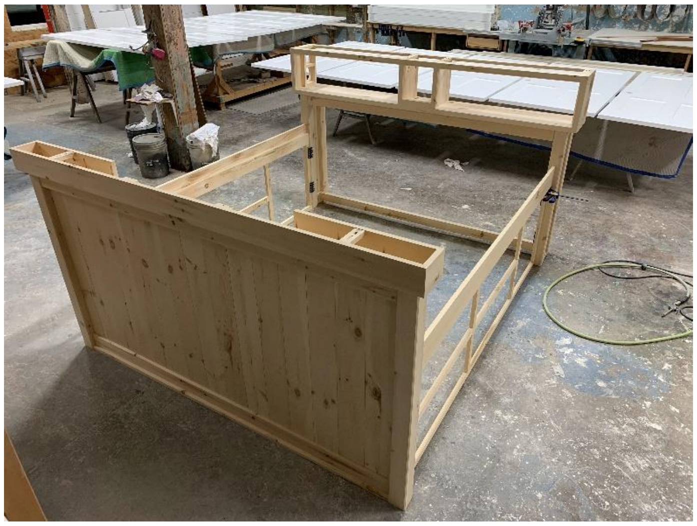


Figure 14: Mounting Side Frames

For the main frames, the process was a little more complicated. The side

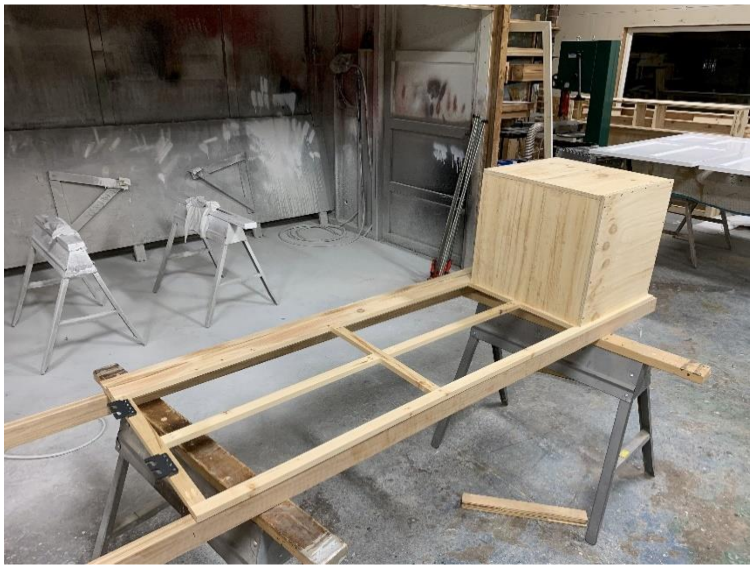


Figure 16: Mounting Minifridge Box

frames were constructed, which was then used to mark where the brackets would mount on the footboard and headboard. This process was done by mounting the brackets to the footboard and headboard, then mounting the brackets to the main frame front face with screws. This allowed for a perfect fit on mounting brackets (although that later proved to be a slight problem when one of the brackets had to be remounted and didn't fit quite as smoothly the second time).

The top beam of the main frame front face side frames also contained a groove for the NeoPixel lights, cut with the same blade that cut the groove in the footboard frame piece. Wiring notches were also placed in the beam for hidden wiring purposes.

The minifridge boxes were constructed next. A higher-grade plywood was used for aesthetic purposes. It was also thicker, which makes the boxes capable of bearing a lot of weight. This was done on purpose for possible future expansion projects, as discussed in the conclusions and recommendations section of this report. The main plywood sections were made after the minifridge boxes. The sections of plywood had to be carefully assembled in order to keep everything square, as the drawer sliders would eventually be mounted to them. The minifridge boxes were then mounted to the side frames, with the final step being to assemble the drawer framing with

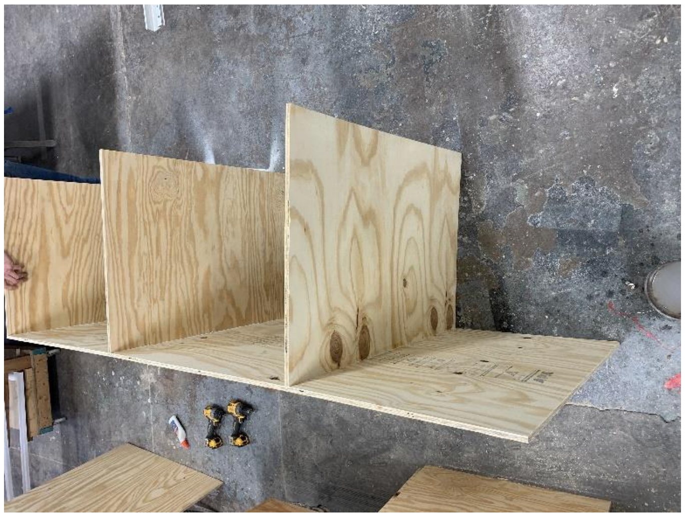


Figure 17: Drawer Space Framing

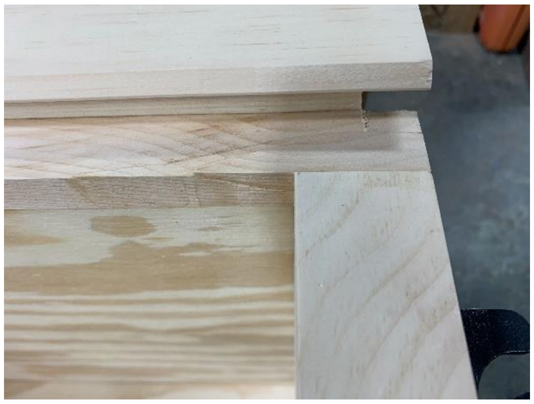


Figure 18: Fixed Face Frame After Crack

the side framing. The entire process was one of the most complicated parts of the project, as need for the drawers to fit and function within the constraints was crucial.

During the mounting of the plywood drawer space framing to the side face frames, we had to constantly ensure our tolerances were being held in check. During the clamping process, multiple cracks were heard as the wood was manipulated to fit the correct shapes and have tight mounts. It was discovered that the face frame had cracked above one of the minifridge boxes somewhere along the process. Glue was used to fix the damage, and after drying, the wood was sanded smooth. While still slightly visible in the above image, the damage was nearly impossible to find after the wood was finished. This was the biggest case of damage during construction.

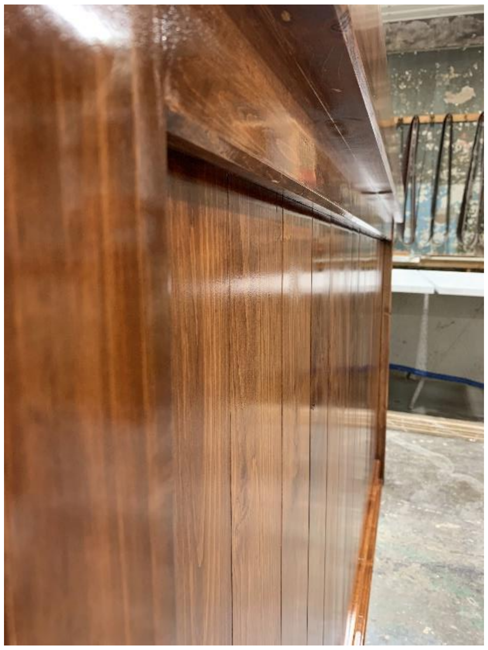


Figure 19: Wood Finish on Footboard

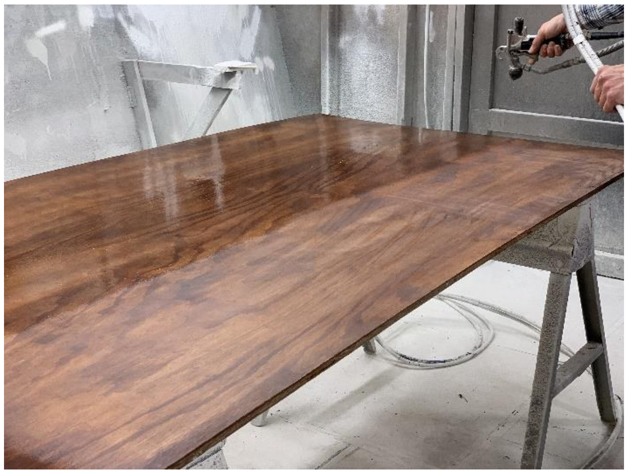


Figure 20: Finishing the Plywood Top

The next step was finishing the wood. All parts of each module had to be carefully sanded and prepared. A wood stain was applied first. We choose Chestnut Base as the color of the stain, as it matches most other furniture colors. Next, a sealer was applied to the wood to protect it. This layer was again given a light sanding before a finishing coat was applied. This coat was designed to give the wood a glossy appearance and act as a final protection coating for the wood. The process took two days, as each coat had to dry before the next could be applied. During this process, it must be noted the topping plywood and drawer faces were also cut, sanded, and finished along with the rest of the modules and pieces.

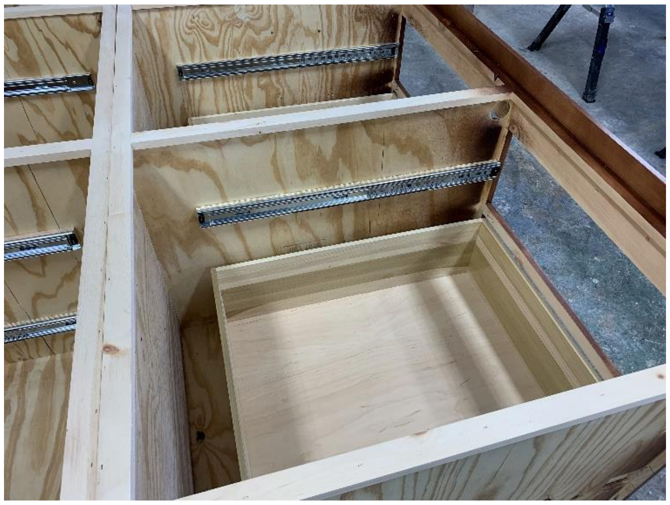


Figure 21: Mounting of Drawer Sliders

Mounting the drawers was the final stage of the construction process. This was done in three phases: the mounting of the slider on the boxes, the mounting of the slider on the main frame, and the mounting of the drawer face and handles.

The first phase of mounting the sliders on the drawer boxes consisted of mounting the internal part of the slider to the drawer boxes, using careful measurements to ensure they will line up in the main frame. The second phase consisted of mounting rail mounting pieces to the sides of the drawer box space, as the plywood itself was too thin to mount the drawers directly to in the first place. This was the design plan all along. Then, the outer portion of the drawer slider was mounted to the rail mounts, again using careful measurements to ensure the boxes would be positioned correctly.

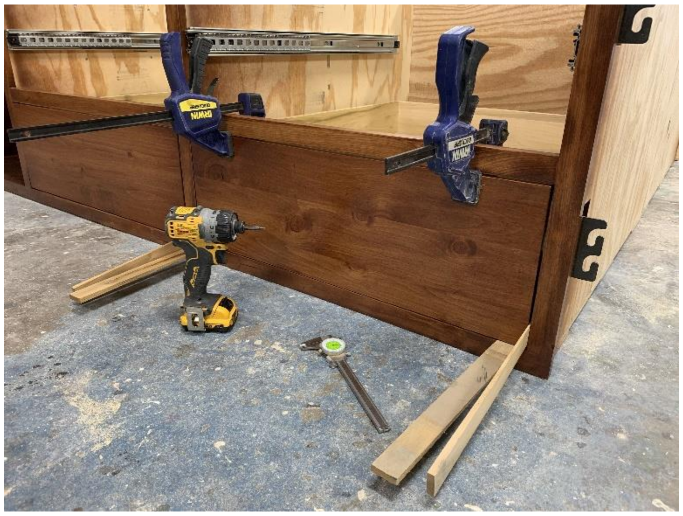


Figure 22: Mounting Drawer Faces

Finally, the drawer faces were mounted to the drawer boxes. By mounting the drawers in this order, we didn’t have to worry about mounting the drawers absolutely perfectly in order for the face to look nice. The drawer could be slightly crooked in one way or another, but as long as the face frame was mounted with a centering to the opening in the

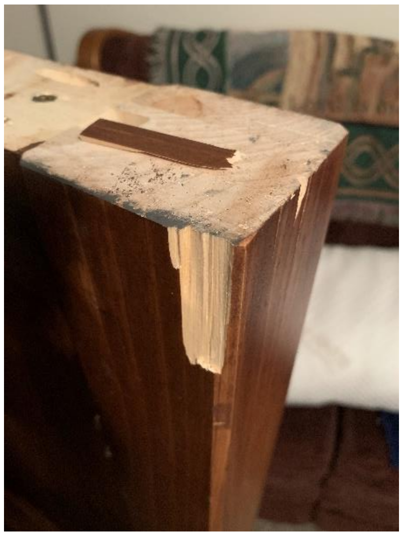


Figure 23: Damaged Footboard Leg

frame instead of to the drawer box itself, any misalignment would go unnoticed. Mounting went well, with only one drawer proving to be mounted at a slight angle, causing one of the drawer face corners to protrude from an otherwise flush face. However, the deformation was less than a sixteenth of an inch, so it was ignored. After the handles were added to the drawers, the plywood tops were placed on.

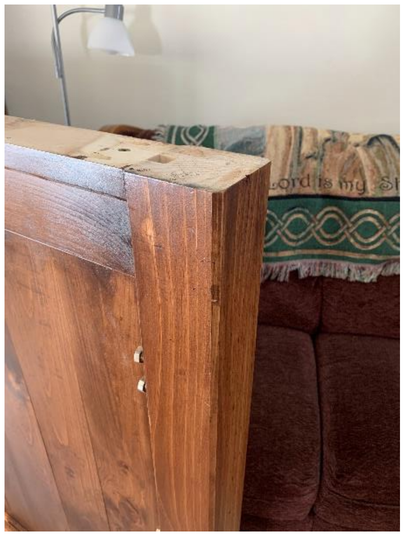


Figure 24: Repaired Footboard Leg

This completed the construction process of the physical bedframe. The last major milestone was transportation back to Ohio from Iowa. As previously mentioned, the trailer that was going to be borrowed for the journey was unavailable at the time of transportation. A U-Haul was rented from a nearby town, and the bed frame modules were placed in the trailer, carefully wrapped in protective blankets, and driven back to Ohio. However, there wasn’t enough packing underneath the legs of the footboard, leading to damage from parts of the wood chipping off due to rough roads. Thankfully, the damage was mostly hidden after being glued back on and coloring put on any white pine showing through the damaged stain.

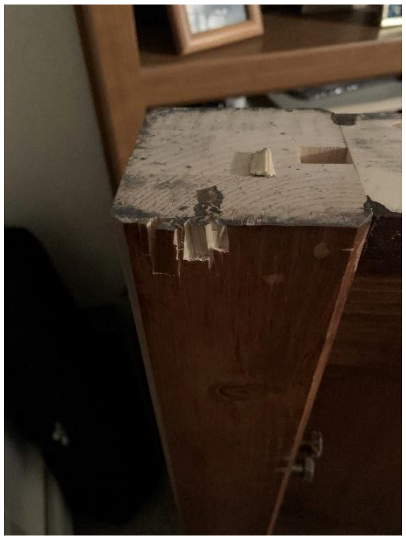


Figure 26: Second Damaged Footboard Leg

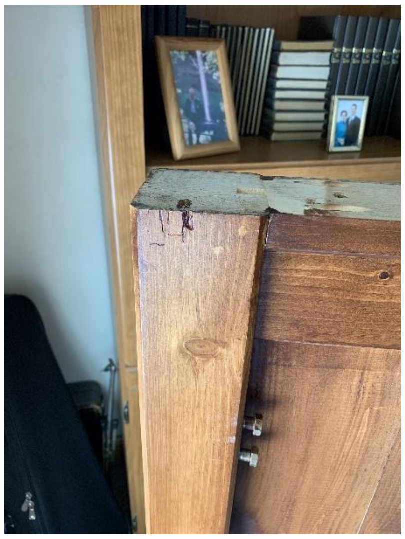


Figure 25: Second Repaired Footboard Leg

The only other assembly that took place was the printing of 3D components for the fans and air deflector assembly. The assembly took place as the pieces were printed and were mounted alongside the process of wiring. In some case, parts were modified and reprinted based on slight design changes or incorrect hole dimensions. The main 3D printed parts were for 120mm fan covers, air deflector mechanism, touch screen mount, and wireless chargers. For the fan covers, the design was made to be a tight slip-fit into the holes in the wood. Behind the visible cover is a small structure designed to give the guard strength should someone push the covers with their hand, such as a child. For the air deflectors, the mechanism was sanded in places for a smooth fit. The deflector blade attaches to the post via 3D printed mounting grips. These grips were designed to be held on with friction by tightening the screws. Most parts of the air deflectors worked well, but small extensions were added on top, along with airfoils directly to the headboard box top, because too much air was escaping around the deflector.

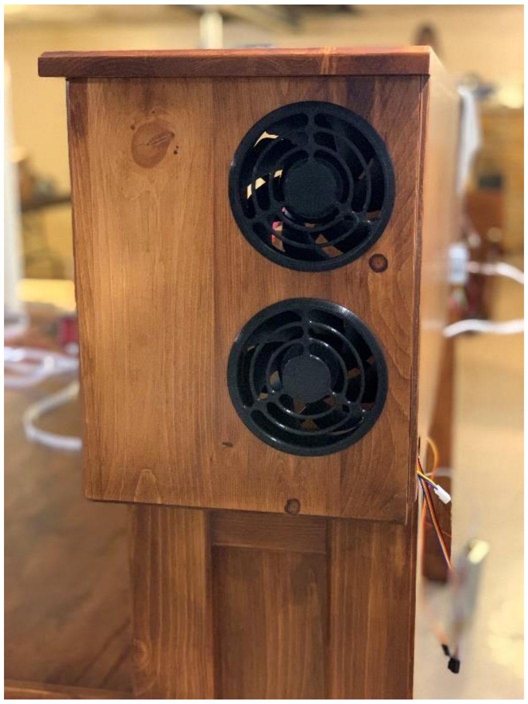


Figure 27: 3D Printed Fan Covers

The touch screen mount was going to be too large to print in a single piece, so it was printed in two separate pieces with a smaller piece that could be friction fit and glued in slots to give the frame strength. The second aspect was that the frame had to be assembled and mounted in a way that access could be gained to the electronics bay through the front panel. Thus, an outer structure, the one printed in two pieces, screws directly to the front face of the headboard box while the main panel screws into that. Finally, it was decided to print the physical control knobs on their own sub-panel for future-

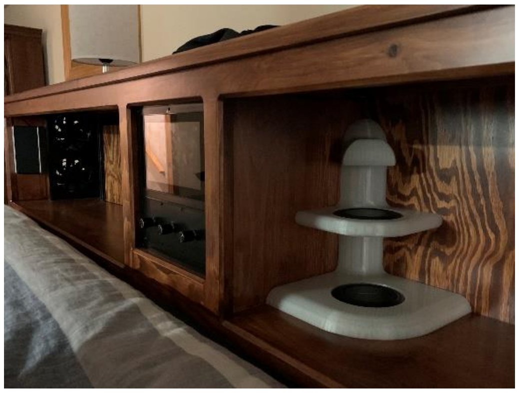


Figure 29: Wireless Charger Mount (Lights Off)

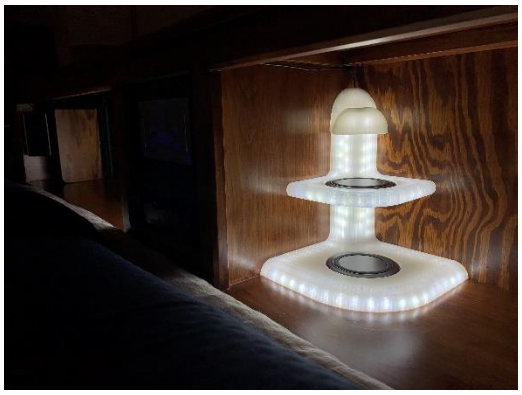


Figure 28: Wireless Charger Mount (Lights On)

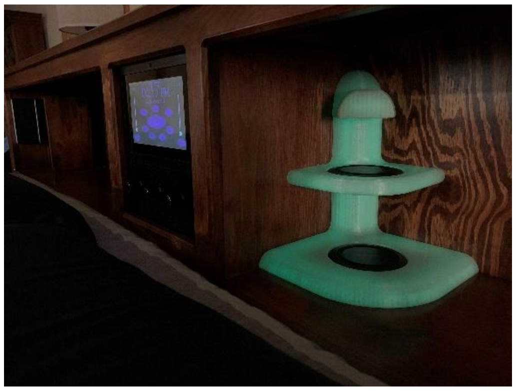


Figure 30: Wireless Charger Mount (Glowing)

proofing purposes. If the user would ever want to change what the physical controls were, they wouldn't have to reprint the entire panel.

The final 3D printed parts to be created and mounted were the wireless charger mounts. The three-tier design was built for the modern trio of wireless charging accessories: smartphone, wireless earbuds, and smart watch. In this case, the smart watch designed for was specifically and Apple Watch, while the wireless earbuds and phone chargers can fit any brand. These chargers are of course plugged into a relay controlled plug for the Raspberry Pi to control. A second feature that was added was internal paths throughout the structure. There were select paths for the wiring that would power the charging pads themselves, and other paths made for stringing LED lights through. Since the plastic used to make the wireless charging mounts, as opposed to black PLA for the other 3D printed parts, the internal lights allowed the wireless charger mounts to glow a brilliant white when toggled on via physical button (explained more in the programming section of this report), and would glow a neon green when toggled off after a minute or two of charging.

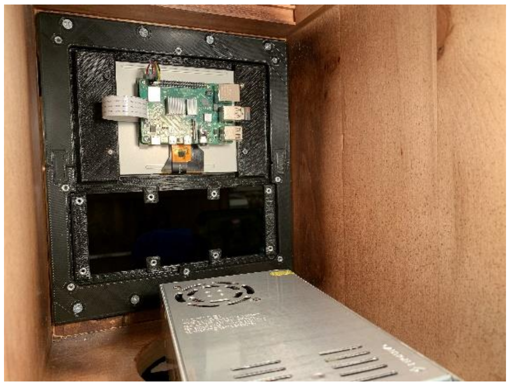


Figure 31: Touch Screen Mount

As the physical construction ended, the focus shifted to running all the wires needed to power and control the bed, along with the testing of various programming aspects and refinement of the graphic user interface, as put together by Chris Waidelich.

Programming

The King of King's Techbed program was mainly written in Python and Kivy. Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Kivy is an open-source Python library for rapid development of applications that make use of innovative user interfaces, such as multi-touch apps. Python was chosen because of its user friendly, high leveled characteristics. It made for an easier experience when writing code for the

King of King’s Techbed program. Kivy was chosen for the graphical user interface because out of all the graphical user interfaces available on the Raspberry Pi, Kivy provided the most modern and up-to-date look.

Referencing the King of King’s program code there are two program files. The first program file is the Python file and the second file is the Kivy file. The Python file is for importing libraries, defining variables and GPIO pins, defining classes and functions, and more. The Kivy file is for coding and organizing Kivy widgets. There are two files when writing code for Kivy because if only one file were used, the main file could become very long. This approach keeps the program organized and more efficient to manage. By coding conventions, libraries are imported at the beginning of the programming file so they can be used throughout program. Likewise, the King of King’s Techbed program imported 38 libraries at the beginning of the Python file. Below is a list of some of the important libraries that were imported into the program.

- Screen Manager
- Parser
- String Property
- List Property
- Label Base
- Backlight
- Popup
- Clock
- Builder
- App
- GPIO
- LED Animations
- Threading

- Board
- Neopixel
- Time
- Pygame
- Datetime
- Encoder
- Re
- Decimal

These libraries and their functionality will be discussed throughout the rest of the report.

The second section of code is where the Neopixel LED light strips are initialized, variables are defined, GPIO and PWM pins are defined, the Pygame audio channel is set up, the backlight object is defined, and more. Specifically looking at lines 34 and 35 the Label Base library is used to import a new font. Kivy only has so many fonts available and there were other fonts desired. Centaur was the desired font. The Label Base library allowed for the import of this font. Looking at lines 37 through 42 the Neopixel LED light strips are initialized. Neopixel lights use a different method of interacting with the GPIO pins compared to other general-purpose outputs. The Neopixel lights use the board library to do this. The King of King's Techbed program has the light strips communicate with pin 10 by setting the variable pixel_pin equal to board.D10. The other lines of code set up the number of LED lights that will light up, the brightness, pixel order, etc. Looking at lines 43 through 50, variables are set up for the GPIO devices. These variables help identify a given GPIO and can be referenced by the variable name to be used.

The next section of code, lines 52 through 61, determines the identification system for the GPIO pins and sets up the numbered pins as either an input or an output. The identification system can either be the physical pin number or the GPIO number. The King of King's Techbed uses the GPIO identification system. The next section of code, lines 70 through 75, creates the Pygame audio channel, the encoder object, and the screen brightness object. When running the program, the audio channel does not have anything loaded into it. By default, the King of King's Techbed program loads the birdsong alarm, so no errors occur. The last section of code is where the encoder function is defined along with global variables that contribute to this function. The encoder function adjusts the brightness of the screen. It does this with an event detection on the 25th pin. Whenever the Raspberry Pi senses a voltage change on pin 25 it fires off the change brightness function. The changes brightness function works by using three global variables x, y and Count. X is set equal to the value of the encoder and y is set to the previous value of the encoder. If x is greater than y when the function is executed, then it adds 10 to the variable Count and the screen brightness value. If x is less than y, then 10 is subtracted from Count and the screen brightness value. There are also other statements to ensure that the value never goes under 0 or above 100. The last section of code is where the encoder button press function is defined. It also uses an event detect to execute itself. If the voltage changes on GPIO pin 1, then the toggle white noise function is fired off. The toggle white noise simply pauses and plays the audio file loaded into the pygame channel.

The next section of code is where the Main Screen is initialized, and its functions are defined. The main screen is one of the most complex screens in terms of the number of functions that are defined. The first function is the Relay on function titled "ROn". This function takes one additional argument which identifies which relay to turn on. This function simply turns on a relay if it is not already on. The next function on the main screen is the Relay Off function, titled "ROff'. This function works in a similar fashion as the Relay On function except it turns off a relay. The third function defined for the Main Screen is the Left Fan on function. This function does more than just turning a relay on or off. It also sets the speed of the left fan, changes the left slider position, and changes the left fan label text accordingly. Getting into more detail, if the state of the left fan relay is on or equal to 1, executing this function sets the left fan speed to 0, changes the slider position to zero, and changes the fan label text to 0. If the left fan relay state is off or equal to 0, then executing this function sets the left fan speed to 25, changes the slider position to 25, and sets the left fan label text to 25. The right fan Off function works in a very similar fashion as the left fan on function except it adjusts the right fan speed, slider position, and label text. The fifth and sixth function on the main screen are the change left fan speed and change right fan speed titled LFanCS and RFanCS. These functions take an additional argument, value1 and value2, and changes the value of these PWM devices in real time. The seventh function on the Main Screen is the power off function. This function sets all the output devices to OFF and then exits the program. The eigth function is the updateTime function. The updateTime function works a little different than the other functions. This is because it is performed on a specified time interval. It runs every five seconds. This function has many sub functions inside of it which update the time and date labels, adjusts the am/pm label position, and checks to see if an alarm needs to go off. The sub function that updates the time and date labels works by utilizing the datetime module. It does this by taking the hour, minute, and am/pm parameters of the datetime module and setting them equal to the time and am/pm label text. The date works a little differently because when using the day and month parameter of the datetime module a number is returned instead of the word. For example, 1 is returned for Monday and January. To get around this, a case switch function was utilized. The case switch functions takes the given number and then gives the equivalent word value to that number. To get the date and year, the program simply takes the date and year parameter of the datetime module and sets it equal to the date label text. The next lines of code, lines 246 through 250, assure that the time and am/pm label are aligned properly. Since times very from 3 digits to 4 digits the program needs to align these labels accordingly. The next sub function in the update time function is the open alarm function. This function does what it implies. It opens the alarm popup when the function is executed. The last portion of the code, lines 260 through 280, checks to see if an alarm needs to go off. It does this by simply comparing the current time string to the alarm string. This code also turns on the Neopixel LED lights slowly from five minutes before the alarm is supposed to turn on until the alarm needs to turn on. It does this by first taking the current alarm string and subtracting five minutes from it. Then it compares that five minutes before alarm string to the current time string and if they are equal, it sets a Boolean variable lightBool equal to True. Then one last if statement checks to see if lightBool is true and the brightness of the lights is less than 255. If this if statement holds true than the alarm lights begin to turn on. These lights continue to get brighter as the update time function continues to execute. The last function on the main screen is the on enter function and on kv post functions. The on enter function checks to see if an alarm is set so it can activate the alarm icon.The on_kv_post function executes the update time function as soon as the main screen is opened and then every five seconds after that.

The next section of code is where the Advanced Color Screen is initialized, and its functions are defined. This screen is a little simpler than the main screen in terms of the number of functions that are defined. At the beginning of this screen there are four global variables labeled r,g,b,w. These variables are used by multiple screens to change the rgbw values of the Neopixel Led lights. The first function on the Advanced Color Screen is the addWhite function. This function takes an extra argument titled value which gives the white slider the ability to change the amount of white that is showing on the Neopixel LED light strips. When the user adjusts the slider on the Advanced Color Screen, they adjust the value of the white light in real time. The second function on the Advanced Color screen is the tuner function. This function is like the add white function except it alters the green and red values on the Neopixel LED lights. This allows the user to create a warmer or colder color by adding or subtracting the value of red and green that is showing on the Neopixel LED light strips. The last function is the clear light's function. This function sets the r,g,b,w values to zero which turns off the lights all together.

The third screen in the King of Kings Techbed Program is the LED Animations 1 screen. An animation is a moving graphic that displays on the Neopixel LED light strips. This screen plays and customizes the first six LED animations available. At the beginning of the LED Animations 1 screen there are four different variables titled speed1, cString, size1, and colorTouple that aid in adjusting the properties of the Neopixel lights. The first function on the LED Animations screen is the set size function. This function does as the name implies and sets the size of the animation where it is applicable. It does this by taking an extra argument titled val and then sets that equal to the global variable size1. The next function is the set speed function. This function works a lot like the set size function except it sets the speed where applicable. The third function on the LED Animations screen is the set color function. This function is very complex because it needs to set the color of the animation, display the text in a label, and set the color of the label text to the color that is displayed.

The first part of the set color function is the case switch function which selects the appropriate color text to display based off of the position of the color slider. Then once the color is selected on the color slider the appropriate color needs to be selected for the text and for the Neopixel lights. A series of if statements run through the possible position selections on the slider and sets the appropriate color value. The third function on the LED Animations screen is the set Animation State function. This function seven global bool values that represent each of the animations available on the first Animation screen. When this function is executed, it sets all of these variables to False. Then depending on which animation was selected, that animation

Boolean value is set to true. It does this by first addressing which animation the user selected. When the user selects an animation, a number 0 – 6, is returned. The if statements determine which number was returned and sets the appropriate animation Boolean variable to True. The fourth function on the LED Animations 1 screen is the start thread function. The program needs to utilize multithreading because the animations need to be run inside of an infinite loop. If multi-threading was not implemented, then the main program would get stuck inside of this loop. So, the first section of the start thread function is where the seven different animations are initialized. These different initializations take many parameters such as speed, size, and color. This is where the global variables speed1, cString, and colorTouple are used to adjust the animation properties.

The fifth function in the LED Animations screen is the startThread function. Since Python has no way of terminating threads inside of the main thread there needed to be a work around in order stop and terminate a thread. The way this was accomplished was by declaring a global variable inside of the start thread function. Depending on what this variable is set to it will determine whether a thread will be created later in this function. This function starts the appropriate animation infinite loop inside a thread based of which animation Boolean variable is true and if the stop_thread variable is false. The sixth function on the LED Animations 1 screen is the Stop Thread function. This function sets global variable stop_thread to false. This breaks the infinite loop that the animation is running in and then terminates the thread. The last function on the LED Animations screen is the UpdateTime2 function. This function is identical to the updateTime function on the main screen. The difference is that it displays the time in the labels on the LED Animations 1 Screen.

The fourth screen part of the King of King’s Techbed program is the LEDAnimations 2 screen. This screen is almost identical to the LED Animations 1 screen except that it has different variable and function names, and it runs rainbow animations.

The fifth screen in the King of King’s Techbed program is the LED color screen. This screen sets the solid color of the Neopixel LED lights. The first function of the LED color screen is the set color function. Inside of this function there are four global variables titled r,g,b,w that are used throughout the program. These variables are then set to zero and sent out to the Neopixel lights to ensure an accurate color selection every time the function is run. Then, in real time, values are then retrieved from what the user selected on the color picker wheel. These value are converted to an integer and then set equal to the global r,g,b,w values that were declared earlier. Then once these values are set, they are sent out to the Neopixel LED light strips and displayed. The second function on the LED color screen is the clear light's function. This sends out a zero value for each of the r,g,b,w parameter on the Neopixel LED light strip and turns the lights off.

The sixth screen that is part of the King of King's Techbed program is the alarm screen. At the beginning of this screen there are global variables atime, which is short for alarm time, and am/pm that are declared to be accessed throughout the program. They are set to zero at first to assure that there are no alarms set. The first function on the alarm screen is the set alarm function. The King of King's Techbed program utilized the Regular Expression library to ensure that no invalid times could be set for the alarm time. This function does what the title implies. It sets the user input equal to the global variable atime and then sets the alarm. There are two subfunctions that are defined under the set time function which are the open ValidPopup and open InvalidPopup functions. A series of if statements check to see if a valid or invalid time was submitted. If the alarm is a valid time, a Popup appears titled valid alarm. This Popup reminds the user of what time they set their alarm and to press continue to return to the main screen. If the user enters an invalid time, a Popup appears showing what the user submitted. The Popup tells the user they need to enter a valid time and they press continue to return back to the alarm screen.

The second function on the alarm screen is the setAMPM. This function sets the global am/pm variable to am or pm based off what the user picked. The third function on the Alarm Screen is the time Input function. This function takes what the user is writing and puts it in a text box. It also assures that the user cannot write over 5 characters because this would result in an invalid time. The fourth function on the Alarm screen is the backspace function. This function does what its name implies. It deletes a character from the farthest position to the right. The fifth function on the Alarm screen is the turnAlarmOff function. This function sets the global atime variable equal to zero, shuts off any audio that is playing, and turns the Neopixel lights off. Essentially turning the alarm off. The last function on the Alarm Screen is the updateTime3 function. This works identically to the other updateTime functions, but it displays the time on the Alarm Screen.

The seventh screen in the King Of King’s Techbed program is the Alarm Sounds screen. This screen works as a place where the user can test out and select an alarm they like. This screen has a global variable declared at the top named current alarm. This variable holds the name of the current selected alarm and assures that this alarm sound is played when the alarm goes off. The first function is the load Alarm function. This function takes an extra argument, which is the path of the alarm sound and loads it into the audio channel. It then sets itself equal to the currentAlarm variable. The second and third function on the Alarm Sounds screen are the play and stop sound functions. They simply do what the name imply. They play or stop whatever audio file is loaded in the audio channel. The last function on the Alarm Sounds Screen is the updateTime4. This works just like the other updateTime functions and displays the current time on this screen.

The eighth and last screen of the King of King’s Tehcbed program is the White noise screen. This screen works just like the Alarm Sounds Screen, except it provides users a place to select white noises and it provides the extra functionality of pausing a sound. The pause function checks to see if there are any sounds playing, and if there is, then the audio file is paused. If there are no sounds playing, and there is an audio file loaded into the channel, then the file resumes playing where it was last paused.

The next portion of code is where the Popup menus are defined, and their functionality is also defined. The first popup that is defined is the Valid Popup. This popup is simple in terms of functionality. When the popup appears on the screen it executes a function that shows what the user entered for an alarm.

The next popup that is defined is the Invalid Popup. This popup is basically the same as the Valid Popup except it also tells the user that they need to re-enter a valid time.

The third popup that is defined is the alarm popup. The first function that is in the Alarm Popup is the turn alarm off function. This function sets the global variable atime to zero, which turns the alarm off. It also stops the current audio file and changes the activated icon on the home page to its off position. The next function is the turn alarm and lights off function. This function is the same as the turn alarm off function, but it also turns off the lights that were turned on from the alarm. Adding both functions allowed for the user to decide whether they want to leave the lights on so they can see when the alarm goes off.

The fourth Popup that is defined is the LWCPopup. This stands for Left Wireless Charger Popup. This popup appears when the user turns on the wireless chargers. Since everyone knows that it is not good to leave your phone plugged into a charger when it is completely charged, the King of King's Techbed gives users various options for how long to turn on the wireless chargers. When the popup appears, the user has four different options to choose from: 1.5 hours, 1 hour, 30 mins, and On. The wireless chargers then stay on for the amount of time that was pressed. The icons also stay lit up to show that they are on. The first function inside of the left wireless charger is the timed relay function. This function has two subfunctions that are titled relayOn and relayOff. When a time is pressed in the popup, the relay On function is executed. Then the clock interval function waits for the specified time to pass and then executes the relay off function. The second function in the left wireless charger popup is the relay On function. This function just turns the relay on if the user decides to turn the wireless chargers on indefinitely. The last popup named RWCPopup works in a similar way as the LWCPopup except it controls the left wireless charger.

Line 1033 is another important line of code because it uses the builder library to load the Kivy file. This line of code is important because without it the program file would not be able to build the GUI and function correctly.

The last line of code is where the app class is defined. The app class is the base for creating Kivy applications. It is the main entry point into the Kivy run loop. Here the Screen Manager and all the other screens are defined as well. The screen manager does about what the name implies. It allows for other screens to be defined and it cycles through the screens when the user or code tells it to do so.

That is a detailed overview of how the Python file works in the King of King's Techbed program. The Kivy file is the second file, and this is where the widgets are organized and where the GUI is built and functions are binded to the GUI events. At the top of the file there is a library called Factory that will be later described in the report.

The first section of code in the Kivy file is where the Main Screen GUI is defined. Here the name of the screen is set to main, and the declaration of a float layout is stated. The name of the screen is important because it is how the screen manager switches between different screens and it is also how different classes access properties from that screen. The float layout is how the widgets can be laid out in the screen. Kivy offers many different layouts but the float layout was desired because it allows for the programmer to choose exactly where they want the widgets to go. Looking inside of the float layout there are five labels, nine buttons, and two sliders. When looking at each of the widgets there are many properties that can be set to change the behavior and appearance of the widget. For example, the first label has 6 different properties that are defined. It has an id, color, font, font size, size, and position property. These properties allows it to have the widget appear and behave how it was intended. Every widget has a set of properties that can be set to get a desired appearance and behavior. The six labels were used to display the date, time, and cooling fan percentage speed. The sixth widget was the LED light button.

Looking at the properties for this button there is a property that states on press. This means that on the press of this button it performs the function that is set equal to it. This is also known as binding a function to the button. The function that is bond to this button is app.root.current = “advancedcolor”. This function takes the user to the advanced color screen from the main screen. Lastly it can be seen that the last property is an image. A Led light image was used as an icon for this button instead of text. This was to give it a more modern look. The seventh and eighth widgets on the main screen are the left and right lamp buttons. Rounded Toggle buttons were chosen because they have two states. They have a down and a normal state. The desired outcome was to have the button icon light up to a pink color when the button was in the down state. Then when the icon was in the normal state it would stop executing any functions that were running and return to a black color.

The other advantage of having two states is that each state can have a function bond to them. In this case the down state had the ROn function bond to it and the normal state had the ROff function bond to it. These functions will turn the left lamp relay on for the left button and the right relay for the right button. This was the case for most of the toggle buttons in the King of King’s Techbed program. The ninth and tenth widgets on the main screen are the left wireless charger and right wireless charger toggle buttons. These toggle buttons have the ROn function bond to the down state and the ROff function bond to the normal state. The eleventh widget on the main screen was the white noise button. This button has a function that takes the user from the main screen to the white noise screen bond to it.

The twelfth widget on the main screen is the power button. This button has the power off function bound to it. This function will turn all of the output devices off and then exit the program. The thirteenth widget is the alarm button. This button has a function bond to it which will take the user from the main screen to the Alarm screen. The fourteenth and fifteenth widget is the left and right fan buttons. These are toggle buttons, and they have the LFanOn and RFanOn functions bound to their normal and down states. If the fan was previously off, these functions will turn the assigned fan on, set the slider position to 25 percent, and set the label text to 25 %. If the fan was on these functions will turn the assigned fan off, set the slider position to zero percent, and set the label text to 0%. The last two widgets on the main screen are the left and right fan sliders. These sliders are bound to the LFanCS and RFanCS functions and will change the speed of the cooling fans to the desired speed in real time. They are also bound to a function that shows the position of the slider in a percentage format in the assigned labels.

The next portion of code on the KV file is where the Advanced Color GUI is defined. Here the name is set to advanced color and a float layout is utilized. This screen has a total of eight widgets defined. It has four buttons, two labels, and two sliders. The first widget is the custom color rounded button. This has a function bound to it which will take it to the custom colors screen from the advanced color screen. The second widget is the animations rounded button. This button has a function bound to it that will take it from the Animations screen from the advanced color screen. The third widget on the advanced color screen is the clear lights rounded button. This button has the clear lights function bound to it and will clear the value of the r,g,b,w variables and turn the lights off. The fourth widget that is on the Animations screen is the return button. This has a function bound to it which will take the user back to the main screen from the Advanced Color Screen. The fifth and sixth widgets are labels. These widgets simply show what which slider stands for in the following widgets. The last two widgets are the tunable slider and the add white slider. The tunable slider is bound to a function that adjusts the value of the green and red parameters on the Neopixel lights in real time. This allows the user to create a warmer or colder color. The white slider is bound to a function that adjusts the white value of the Neopixel lights. This allows the user to create a lighter or darker color.

The next portion of the code in the KV file is where the LED Animations 1 screen is defined. Here the name is set to animations and the float layout is utilized. This screen has twenty widgets. These include ten buttons, eight labels, and two sliders. The first two widgets are the time and date labels. These labels display the current time and am/pm information. These labels do not have functions bound because they have functions that access the label properties to update the time on a five second interval in the Python file. The next six widgets that are on the LED animation screen are six rounded toggle buttons. In Kivy, toggle buttons can be set to a group to assure that only one toggle button is pressed down, and the rest are up. In this case, these six buttons are set to one group called animations. Each of these six toggle buttons have a function bound to them that sends value signifying which button was pressed. This sets a Boolean value true in the Python file which lets the program know which animation to have set up for use. The next two widgets on the LED Animations Screen are the pause and play buttons. These buttons are bound to the start Thread and stop Thread function. When the user presses the play button the play icon lights up and the animation starts. When the user presses the stop button the play icon goes back to black and the animation stops. The next two widget on this screen are the return button and rainbow animations button. The return button is bound to a function that returns it to the main screen. The rainbow animations button has a function bound to it that will take the user to the LED Animations 2 screen from the current screen. The next four widgets on the LED Animations Screen are the labels. The color and speed label simply have color and speed set for there text to show what the two sliders stand for. The other two labels show what speed and color the animation is set to based off the slider positions. The last two widgets on the LED Aniamtions Screen are the speed and color sliders. These sliders have functions bound to them which allow the user to adjust the color and speed of the animation. The LED Animations 2 screen is almost identical to this screen except it plays rainbow animations and it has speed and size as the customizable properties instead of color and speed.

The fifth screen that is defined in the KV file is the LED color screen. Here the name was set to ccolor which is short for custom color. The floatlayout was utilized for this screen as well. The custom color screen has five widgets. It has a color picker and three buttons. The first widget defined on the custom color screen is the color picker. The color picker is a color wheel that can be used to generate a custom color. Depending on where the user touches on the color wheel the color wheel will generate rgbw, hsv, and hexadecimal values that match the color the user created. These values are set equal to the rgbw values in the Python file. The user can display this color on the Neopixel lights if they decide to. The next widget on the custom color screen is the set button. Since the color wheel does not work in real time to display colors on the

Neopixel light strips a set button needed to be made. The set button is bound to a function that sends the necessary information out to the LED light strips to display the color. The next widget in the KV file is the clear button. This button does what it implies and is bound to a color that clears the rgbw values and turned the lights off. The last widget on the custom color screen is the return button. This button is bound to a function that takes the user back to the advanced color menu.

The sixth screen defined in the KV file is the Alarm screen. The Alarm Screen name was set to alarmscreen and utilized a float layout. The alarm screen has 23 widgets. It has five labels, fifteen buttons, two checkboxes and a text input box. The first two labels are the date and time labels. These label's properties are accessed by the Python file and display and update the time every five seconds. The third widget is the alarm label. This label simply displays where the text input box is where the user is going to input an alarm time. The fourth widget is the time input text box. This is where the alarm shows up as the user inputs the information with the numbered buttons which will be further explained in the paper. The next four widgets are two labels and two checkboxes. Checkboxes work in groups and only allow for one of them to be checked. This is what was desired because the user needed to choose between am and pm. The two labels differentiate the two checkboxes from am to pm. The text in these labels are "am" and "pm" respectively. The two checkboxes are then bound to functions that set the alarm time to am or pm. The next widget is alarm button. This button is bound to a function that takes the user to the alarm sounds menu from the alarm menu. The next two widgets are buttons as well. These buttons are the set and cancel button. The set button is bound to a function that will take the text and am/pm parameters and will attempt to set the alarm. The Python file will check to see if this time is valid or not and if it is the alarm will be set. The cancel button is bound to a function that will turn the alarm off and erase the any previous alarms that were set. This button will also take the user back to the main screen. The next 12 widgets are buttons that make up a custom keyboard that the user can use to set alarm times. The buttons have text from 0-9, a colon, and a backspace. These buttons have functions bound to them to allow typing alarm times and backspacing if needed to.

The seventh screen that is defined on the KV file is the Alarm Sounds screen. This screen's name is set to alarmSound and utilizes the float layout. This screen has 17 widgets. These are two labels and fifteen buttons. Like a lot of the screens the first two labels display the time and date. The next twelve widgets are toggle buttons. Each of these toggle buttons are apart of a group and represent the alarms that are preloaded into the King of King’s Techbed program. Since these buttons are in a group only one of the alarms can be selected at a time. Once a user selects an alarm the text turns pink. These buttons are bounded to functions that will load the designated alarm file into the audio channel and also set them equal to a variable that will ensure that this file will be played if an alarm goes off. The next two buttons are the play and stop buttons. These buttons are bound to functions that will allow the user to listen to an alarm to make sure that is what they want set. Then the stop button is bound to a stop function which stops the audio file. The last widget on the alarm sounds screen is the return button. This button is bound to a function that returns the user to the alarm screen.

The last screen that is defined in the Kv file is the white sounds screen. This screen is very similar to the alarm screen except the user chooses from ambient sounds to fall asleep to instead of alarms. Like the alarm screen the time and date are displayed at the top of the screen. There are also twelve toggle buttons that the user can choose from to select an ambient sound. At the bottom is where the white noise screen differs a little bit from the alarm screen. There is an extra button that allows the user to pause the ambient sound. It was chosen to do this because the white noise files are so long that the user might want to pause the sound instead of stopping it.

The next section of the KV file is where the Popups were defined. There was a total of five Popups in the Kivy file. Each of these popups have parameters at the beginning of their decleration that define the title, size, position on the screen, and background. Every popup in the Kivy file was .6 by .6 in size, was positioned at the x and y coordinates .2, and .9, and had a black background with a blue border.

Digging in deeper the first popup that was defined was the valid alarm popup. This popup’s title was set to “Alarm Notice!” and also utilized the float layout. The valid popup had only two widgets which were a label and a button. The label displayed the alarm time that the user entered and let the user know that it was a valid time. The button was bound to a function that simply closed the popup and sent the user back to the main screen.

The next popup that was defined was the Invalid popup. This popup’s title was set to “Alarm notice!” and used the float layout. The invalid poup had two widgets. It had a label and a button. The label let the user know what they set as an alarm and let them know that it was an invalid input. The button closed out of the popup and returned the user back to the alarm screen to enter a valid time.

The next two popups that were defined were the left and right wireless charger popups. These popups were titled “Left Wireless Charging Configuration” and “Right Wireless Charging Configuration” respectively. They both used a float layout and had four widgets which were buttons. The first three buttons had 30 minutes, 1 hour, and 1.5 hours set to their texts. This text represented the amount of time the wireless charger would stay on. These buttons also had functions bound to them that would set a timer for how long the wireless charger would stay on. They also had a function bound to them that would close out of the popup and take the user back to the main screen. The last button had the text “On” inside of the button. This button had a function bound to it that would turn the button on indefinitely and then close out of the popup and take the user back to the main screen.

The last popup that was defined was the Alarm Popup. The alarm popup was responsible for going off when the alarm would go off. This popup’s title was set to “Alarm Notice!” and utilized the float layout. This popup had only two widgets which were two buttons. These two buttons were the turn alarm off button and turn everything off button. The first button had a function bound to it that would turn the audio off and reset the alarm value when the alarm went off. The second button had a function bound to it that would also turn the alarm audio off and reset the alarm value, but it would also turn the lights off.

The last section of the code in the Kivy file is where custom widgets were made. By default, Kivy uses square buttons and toggle buttons. Round buttons were desired to get a more of a modern look. So, the last several lines of codes is where custom widgets were defined so the King of King’s Techbed program could utilize round buttons.

As stated above, most of the program was written in Python and Kivy on the Raspberry Pi. A small portion of code was written on an Arduino in C++ though. This was because a servo motor needed to be controlled by the Raspberry Pi using an encoder. This requires analog inputs which the Raspberry Pi does not have. An ADC could have been used to get around to solve this, but this would have taken up pins the Neopixel lights needed, and it also made the code much more complex. So instead, an Arduino was used.

The first section of code in the C++ script is where the necessary libraries were imported. In the case of the C++ Arduino script only the servo motor library needed to be imported. This library was used to control the servo motors properly.

The next section of code is where variables are defined, and servo motors are initialized. Since there was only two servo motors there was only a left and right servo motor initialized. The next two lines of code is where variables are created to reference the servo motor pins and set them up as an output. The next six variables were created to aid in the process of creating the functions. The next eight variables are defined to help with the functionality of the rotary encoders. The last eight variables are defined to help with the functionality of the button press of the rotary encoders.

The next section of code is where the input and output pins are setup. There are multiple input and output pins setup for the servo motors and multiple pins for the rotary encoders. There are also variables set up for determining the position of the rotary encoders and Boolean variables to help with debouncing the button presses of the encoders.

The last section of the code is where the functionality is defined. The first block of code is where the debouncing method is defined for the left encoder. It is also where the encoders can turn on the left wireless charging lights. The next block of code is where the debouncing method is defined for the right rotary encoder. It is also where the encoders can turn on the right wireless charging lights.

The last section of code is where the functionality for the turning of the rotary encoders are. In these last two functions when the rotary encoders are turned left the servo motors rotate left. When the rotary encoders are turned right then the servo motors rotate right.

Throughout the programming progress, several bugs were uncovered as the programming for each system was designed. As mentioned above, it became necessary to branch off some of the processing to a designated microprocessor in the form of an Arduino Uno. Thankfully, the project ended with a fully functional code, which can be found in Appendix D and E of this report.

Results

The resulting project was something the King of Kings TechBed group was proud to have created. The resulting wooden structure and function of the bed electronics turned out very well. While there are a few areas where things could have been done differently to make an even better finished product, the project is declared a success by Jacob Klopfenstein and Chris Waidelich.

Finished Bed Photos

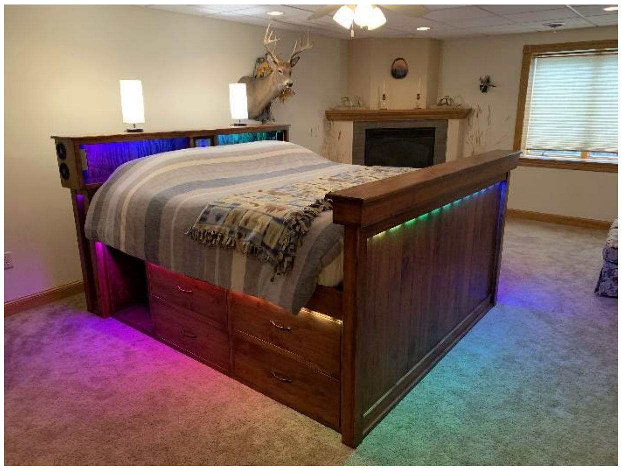


Figure 32: Left View with Color Animation

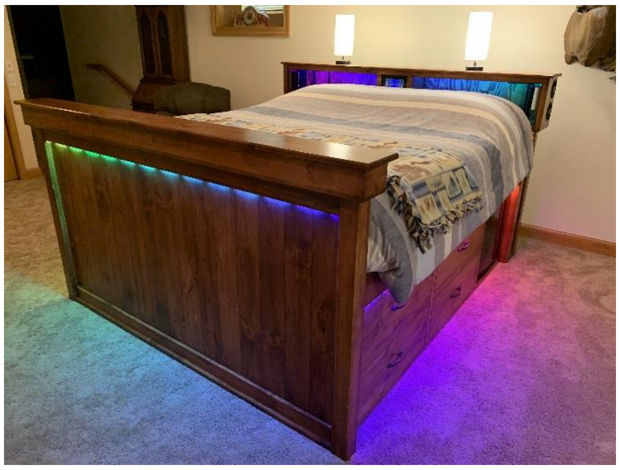


Figure 33: Right View with Color Animation

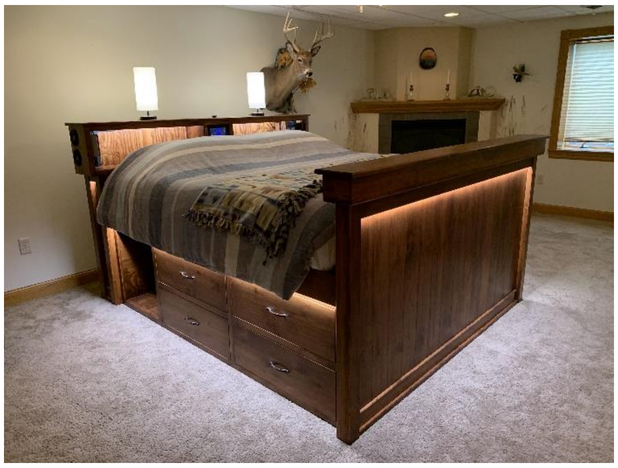


Figure 35: Left View with White Light

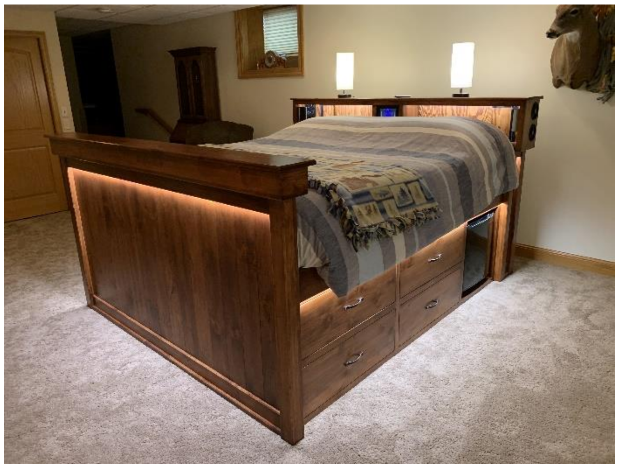


Figure 34: Right View with White Light

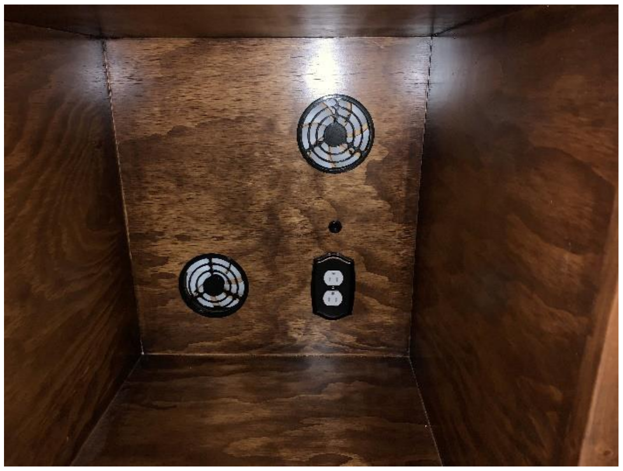


Figure 41: Inside Minifridge Box

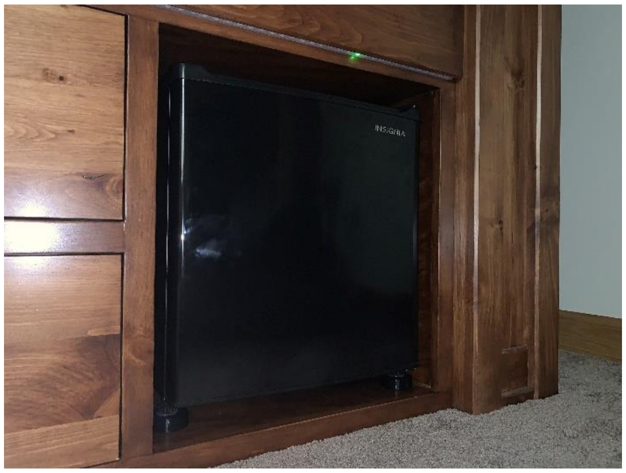


Figure 40: Mounted Minifridge

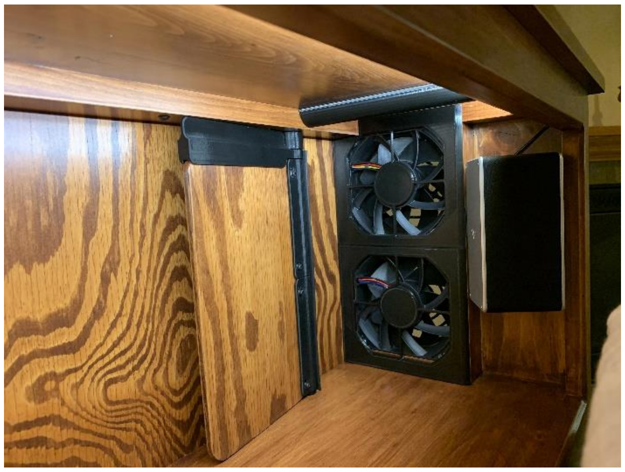


Figure 37: Inside Headboard

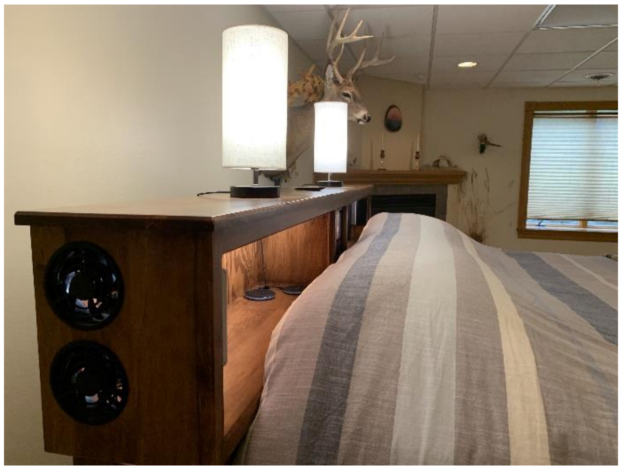


Figure 36: Across Headboard

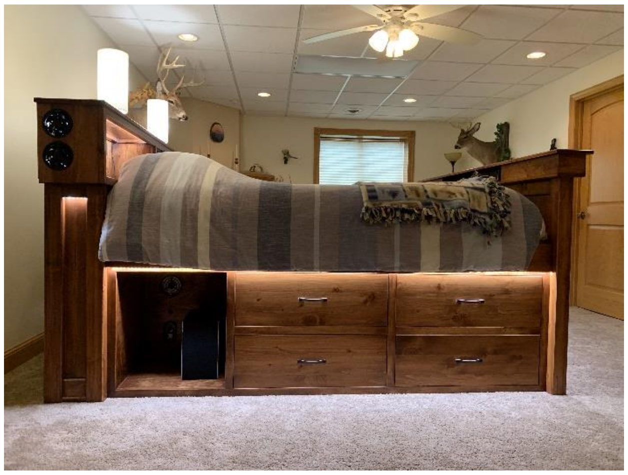


Figure 38: Side View with White Light

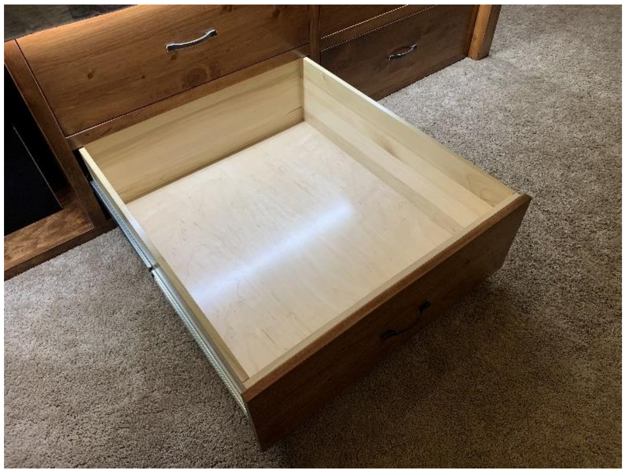


Figure 39: Extended Drawer

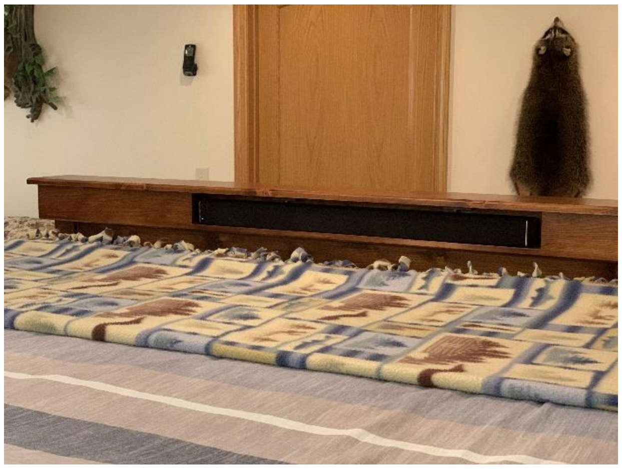


Figure 45: Sound Bar in Footboard

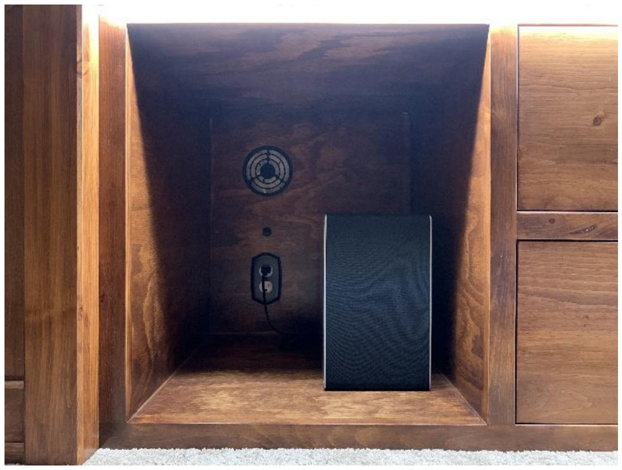


Figure 44: Subwoofer in Other Minifridge Box

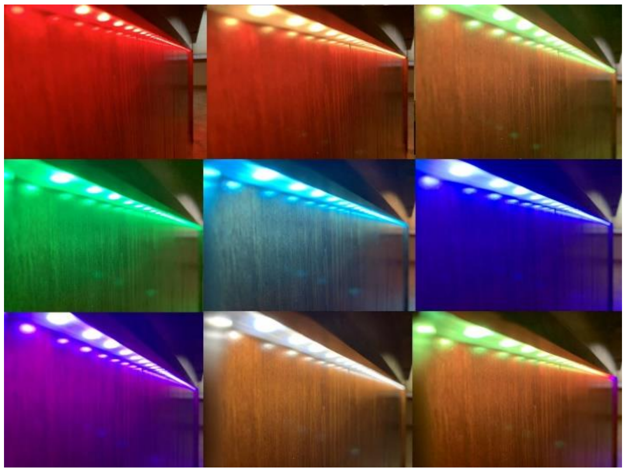


Figure 42: Sample of Light Strip Colors

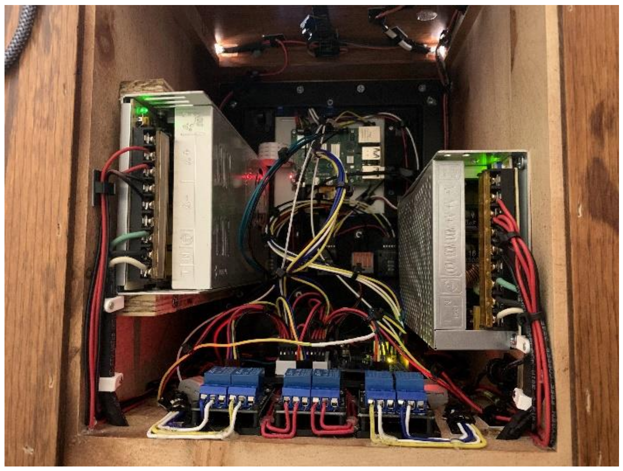


Figure 43: Wiring in Electronics Bay

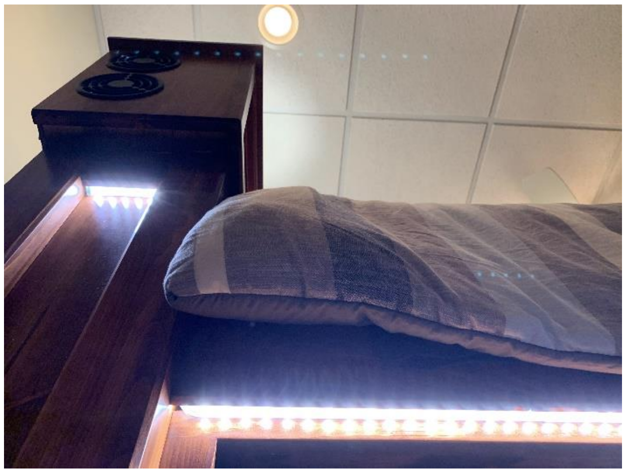


Figure 47: View up at Headboard Lights

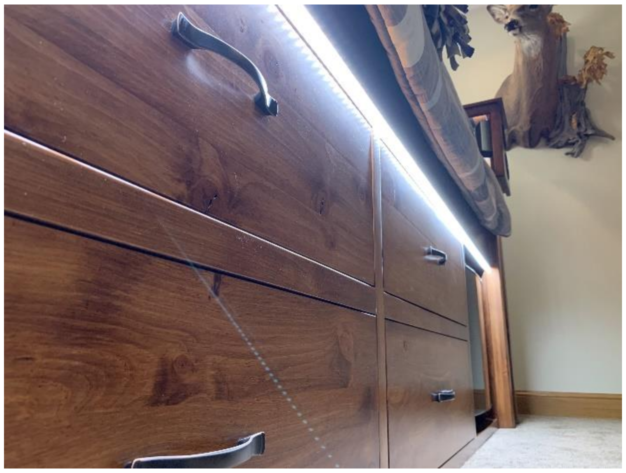


Figure 46: View Along Main Frame Lights

Graphical User Interface

The GUI was made completely from scratch for the interface for this bed. While there will always be minor bugs, as there are for any software, we have amazingly been able to work out most of them. The touch screen is responsible, the program is stable and operable, and the physical controls integrate well with the touch screen controls. This next sections talks through what you see on each menu and how it functions.

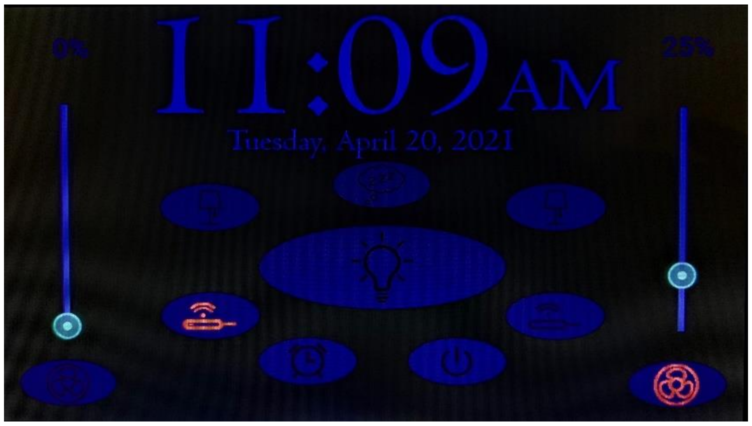


Figure 49: Main GUI Menu

Starting with the main menu, when you press top button, which is the ambient sounds button, this takes you to another screen that you can select to play relaxing music/sounds in the background. When you press the lamp buttons, they turn on the left and right lamps through relay control. Also, it can be noticed that the lamp icons light up showing the user that the lamps are on. When lamp button is pressed this turns off the lamp and turns the icon off showing that the lamps are off. The wireless charger buttons work in a similar fashion except when you press these buttons a pop up appears. This pop up asks the user if they want to turn on the wireless chargers for a pre-determined set of time or if they want to turn it on indefinitely. Once the user selects one of the given options the popup disappears, the icon lights up, and the wireless charger turns on through relay control. If the user pressed a preset time, the wireless charger stays for the given amount of time and then turns off. When the wireless charger turns off the icon also deactivates. If the user chose to turn the wireless charger on indefinitely then the user presses the button to turn it off.

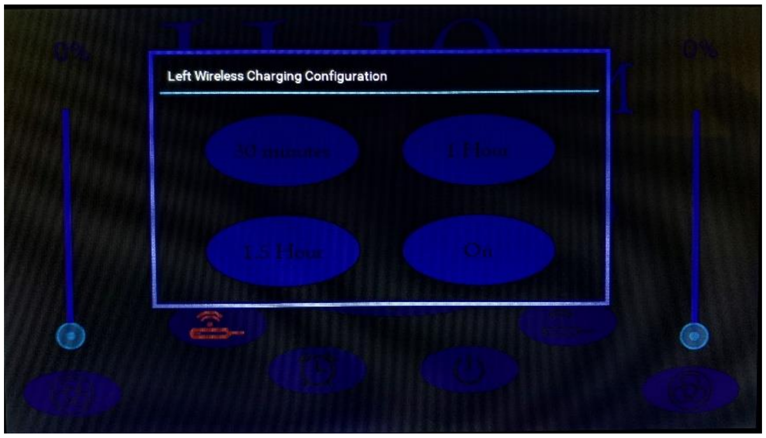


Figure 48: Wireless Charger Menu Pop-Up

Next, the cooling fans will be covered. When the cooling fan button is pressed it turns the cooling fan on with a relay and sets the speed to 25 percent. It sets the speed by changing the duty cycle of the PWM signal that is sent to the fan. This also lights up the icon showing that the fan is turned on. The user will also see that the fan speed percentage is shown on the top percentage label and the slider is set to a 25 percent position. This is where the sliders come into play. The user can decide which speed they want the fans to be set to by sliding the slider up and down to a desired position. The label percentage changes as the slider moves showing the user what speed they are at. Lastly, if the user wants to turn the fan off they can toggle the cooling fan button and the fans relay will turn off. Doing this will also set the fan speed percentage to zero and putting the slider position to zero. It should also be noted that due to the nature of the fans selected for this project, most of the slider (1% - 99%) is actually around 35% - 75% of the fan speed. Once the slider hits 100%, only then will the fan jump up to full power, while the fan cannot operate under 35%.

The next button is the LED lights button. This button will take the user to the advanced light menu where they will have several options for altering the light settings. The last two buttons at the bottom are the power off and alarm buttons. The alarm button will take the user to the alarm screen where they will be able to set an alarm and modify other alarm settings. And the power off button. This button will power off all relays and electronics and will shutdown the King of King’s software.

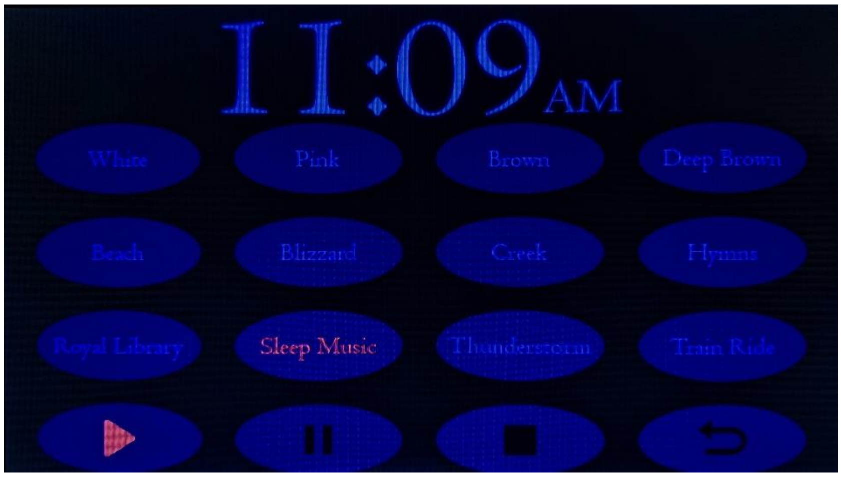


Figure 50: Ambient Sound Menu

The next screen that will be referenced is the ambient sound screen. It can be seen there are 12 different ambient sounds the user can choose from to play in the background. When the user selects a sound, such as white, the text will light up to show that that sound is the current selected loaded sound. The user can select the play button at the bottom left of the screen to play the sound indefinitely. When the user selects the play button the icon will light up signifying that the ambient sound is currently playing. They can press the return button to go back to the main menu while the sound continues to play on. The user can also go back to the ambient sound screen and press the pause button to pause the sound. This will pause the sound instead of restarting the sound like the stop button will do. The user can also click the middle rotary encoder to toggle between pause and play. Lastly, the user can select the stop button which will stop the sound and restart the audio file so it can be played from the beginning again. If the user wants to change the current loaded sound, is all they need to do is just press another provided sound and that button will light up and the other sound text will turn off. They can press the return button which will take them back to the main menu.

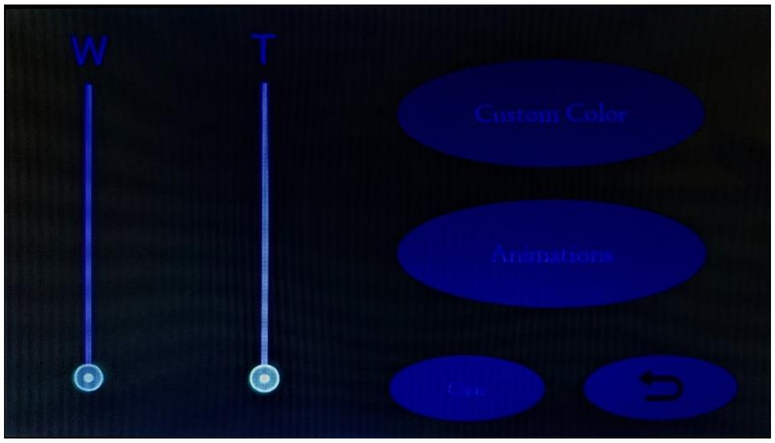


Figure 52: Light Main Menu

The next screen that will be referenced is the custom solid colors screen. The user gets to this screen by pressing the LED light button on the main menu and then this will take them to the advanced light menu. The user can then press the Custom Color button and then will arrive to the custom color screen. On the custom color screen, you will see a color wheel, 3 buttons, and 8 different sliders. The color wheel works in two different ways to turn the LED lighting on. The first way the user can use this is to pinch in and out to alter different color settings and brightness. Then the user can choose different colors and then select their desired color. Once the color is selected it will show up on the slider background and these values will be loaded into the color variables. The user can press the set color button, and this will light the LED lights up to the chosen color. The second way the user can use the color wheel is adjusting the color sliders. On the left, you can see 7 different sliders labeled R for red, G for green, b for

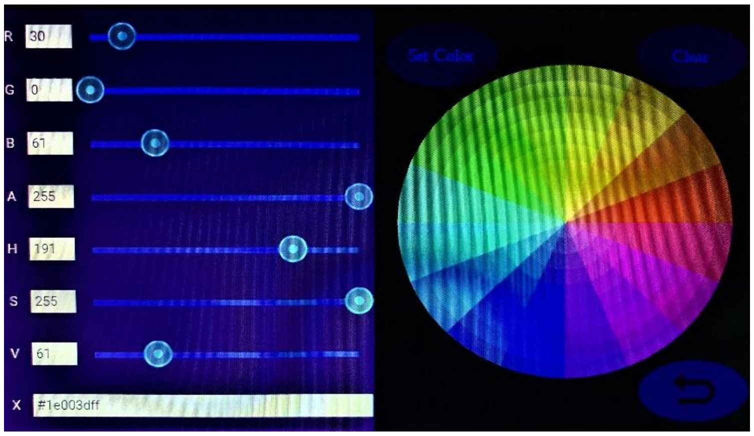

R 30
G 0
B 61
A 255
H 191
S 255
V 61
X #1e003dff

Figure 51: Custom Color Menu

blue, a for alpha, h for hue, s for saturation, and v for color value. The user can also adjust these sliders to change the desired color. The last two buttons are the clear and return button. The clear button sets the color variables to zero and turns the lights off. The return button sends the user back to the advanced light menu.

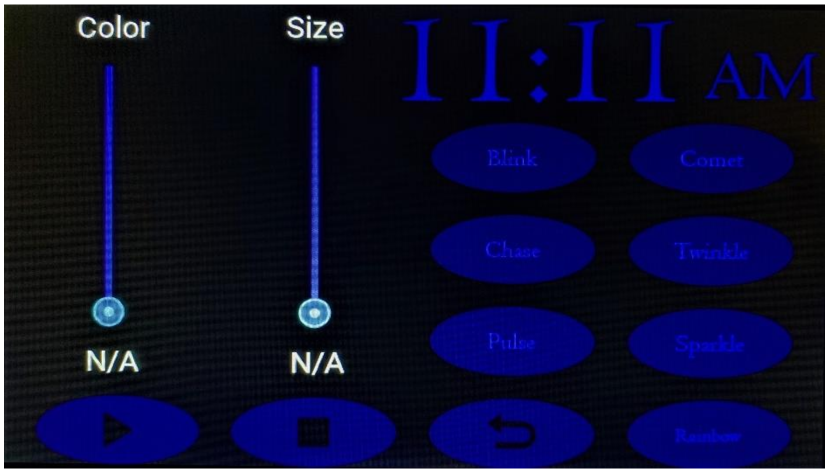


Figure 53: Solid Color Animation Light Menu

The next screen that will be referenced is the advanced light menu. Here the user is provided with several options to adjust light settings. The screen has two sliders and 4 buttons. The two sliders labeled W and T are for tuning the lights to a desired effect. The slider labeled w adjust the amount of white that is added to the light. The slider labeled T adjusts the red and green values of the light. This allows the user to create a warmer or colder color. The two buttons labeled custom color and animations will take the user to the custom color screen and animations screens. The clear buttons clear the light values and turns the lights off. The return button sends the user back to the main menu.

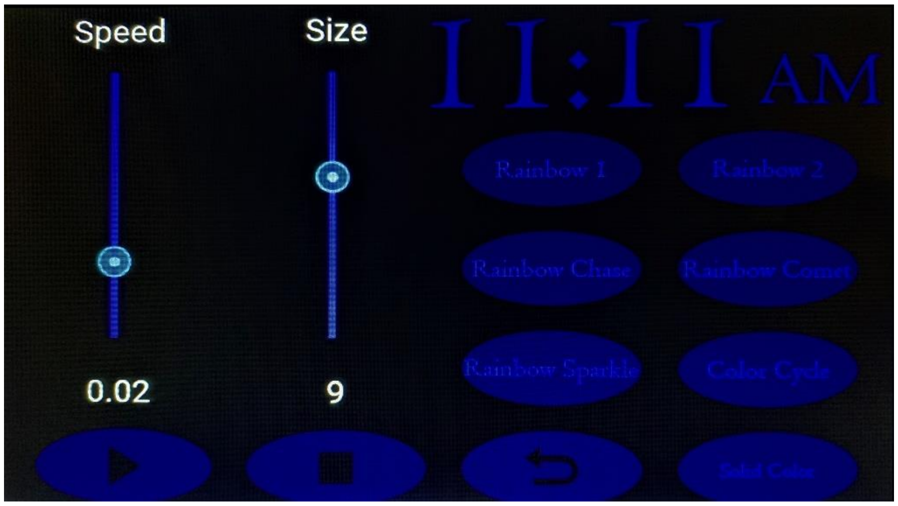


Figure 54: Rainbow Animation Light Menu

The next screen I will be referenced is the basic animations screen. Here the user can decide from 6 different animations. When the user selects a certain animation the text lights up signifying that animation is loaded and selected. They can also select between the color and size slider to adjust the animation settings. They can then press the play and pause buttons to start and stop the animations. The page 2 button will take the user to another page of animations to select from. The return button will take the user back to the advanced light menu.

The next screen that will be referenced is the rainbow animations screen. This screen is a lot like the basic animations screen page, but it has rainbow animations instead. Like the basic animation screen the user selects the desired animation, changes the settings, and then starts and stops the animations with the play and pause button. The user can select the return button to take them back to the advanced light menu or press the page one and take them back to the advanced light menu.

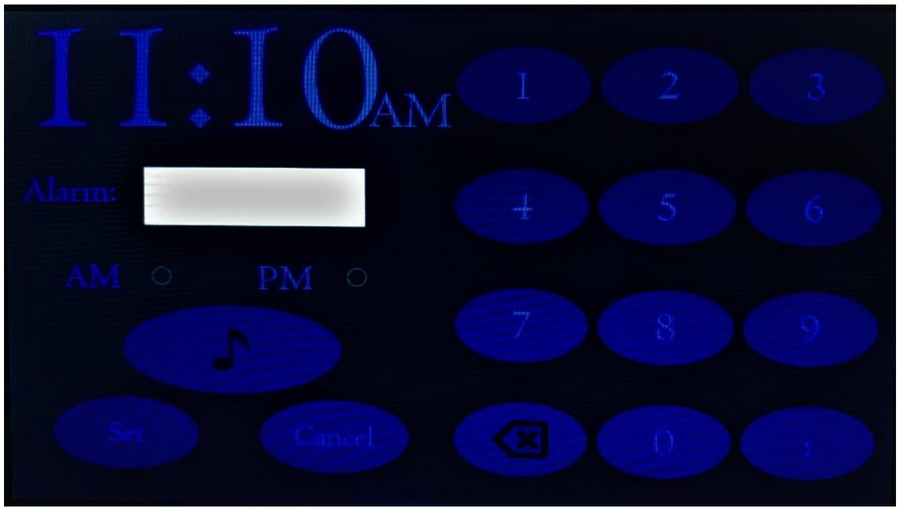


Figure 55: Alarm Clock Screen

The next screen that will be referenced is the Alarm Screen. Here the user has the options to set an alarm time and set the alarm sound. The user sets the alarm time by pressing the numbered buttons on the right. The selected numbers appear in the alarm text box showing the user what they've written. Then the user selects am or pm and then determining am or pm. They then select an alarm sound. To set the alarm the user needs to select "Set". If the user submits a valid time a pop up shows up telling them the alarm time the have selected. The user can press the close button on the popup, and this will take them back to the main menu. On the main menu they will see the alarm button lit up signifying that there is an alarm set. If the user selects an invalid time, a pop up appears telling the user that they have entered an invalid time. The user then presses the close button which returns them back to the alarm screen to set a valid alarm and the incorrect alarm is not set. The last two buttons that were not mentioned were the alarm sound button and the cancel button. When the alarm sound is pressed the user is taken to the alarm sound screen where they can select an alarm sound. When the user presses the cancel button it erases any set alarms and takes them back to the main menu. Lastly when an alarm is set, the LED lights will slowly start to light up to a warm value 5 minutes before the alarm is set to go off. When the alarm goes off a pop up comes up where there are two different options to choose from. These two options are turn alarm off and turn all off. If the user selects turn alarm off the alarm is turned off and is erased from the memory but the light stays on. If turned all off is selected the alarm and lights turn off and the alarm is erased from the memory. When either option is selected the user returns to the main menu.

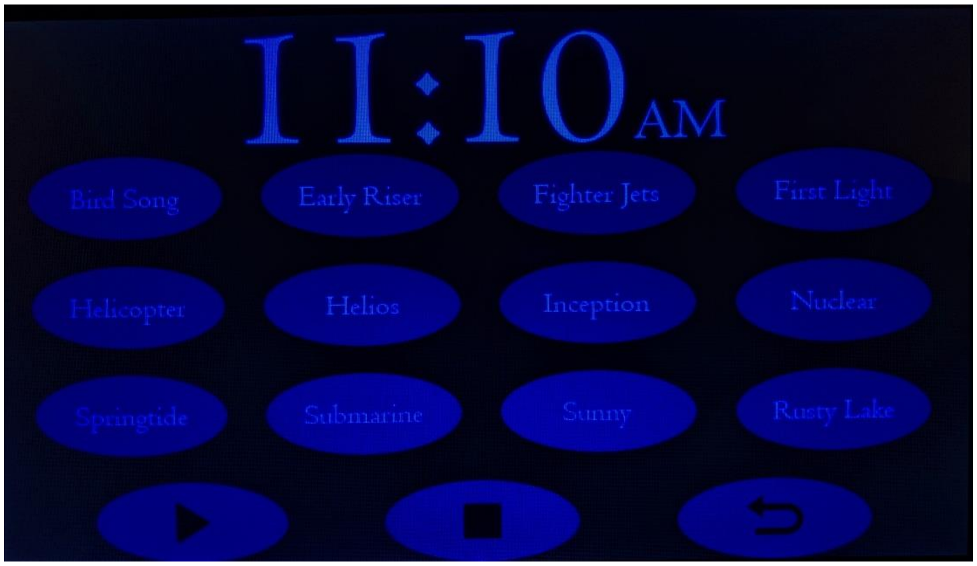


Figure 56: Alarm Clock Sound Menu

The last screen that will be referenced is the Alarm Sounds Screen. This screen works a lot like the ambient sound screen except the sound doesn’t play indefinitely and there is no pause button. An alarm can be selected by simply pressing one of the alarm buttons. When the button is pressed the alarm is loaded and the text lights up signifying that that alarm is selected. The user can play the alarm too to hear what the alarm sounds like. The user then can stop the alarm if they’ve heard enough by pressing the stop button. Once the alarm is selected the user can press the return button which will take them back to the alarm screen.

Results Summary

In all the success of the project, we can easily consider this project a complete success. The resulting bedframe is one that one of us will use for years to come, while at the same time giving us a cool story to tell about our senior design project at Miami University. However, in the midst of this success, there are still issues with the bed we noticed in final testing. The main issue is with the right servo motor for the right air deflector. Due to the amount of power the minifridge takes, when the compressor turns on and off, a power lapse in the system causes the right servo, and only the right servo, to drift to the zero position, no matter where it is set to be. It is theorized that adding a large capacitor to give more consistent power levels might work, but that alteration has yet to be implemented. The second issue is with the GUI alarm lights integration. While initially working, the lights haven't been growing brighter properly before the alarm goes off. This is presumably a simple software fix and is being researched. Finally, the chosen 5V power supply unit has a built in fan that is extremely noisy. The plan for the future is to swap out the power supply unity for a different one with a quiet fan. This change doesn't effect the bed's performance in any way. Other than those three issues, the bed functions as expected.

Conclusion and Recommendation for Further Study:

The *King of Kings TechBed* was outlined in the objectives of the project proposal. It is by the work and knowledge of the group, Jacob Klopfenstein and Chris Waidelich, that the various goals and parameters laid out at the beginning of this project were achieved. There are no initial requirements for the product that are not accomplished in the resulting product, plus, a few bonus features were added to the design that were not initially planned on.

This project isn't perfect, by any means. Because the group members lacked knowledge of woodworking, there are minor defects in the construction of the frame and imperfect tolerances in places. Thankfully, there were no major flaws that would render the project unusable or too ugly to use. For the most part, the technology was able to cover up a majority of any inconsistencies from construction.

One thing that was noticed throughout the project was the fact that the bed was so outlandish to begin with, once a person heard that a minifridge was installed on the bed, any category of recommendation could be given for future expansions, including the addition of a roll up TV from the footboard, remote controlled lift to automatically sit the bed up into an incline for viewing media, the mounting of a microwave or toaster oven on the headboard, or even a launch mechanism to throw the occupants out of bed if they didn't get up within five minutes of the alarm going off. Of course, these are just the comments heard the most often, as there were many others not recounted here. Of course, nearly all of these expansions could be completed at some point. The minifridge boxes were designed to be strong enough that a remote "sit-up" system could be installed.

Also, because there are still a wide variety of ports and pins available on both the Raspberry Pi and Arduino, there is lots of room to expand programming with, with a large variety of controlled systems and sensors that could be added. One of the most prominent example of future expansion in this area is the addition of temperature sensors in the minifridge box and electronics bay to regulate the speed of the cooling fans in those boxes instead of the fans being either “on” or “off”. Finally, with the bed being so tall, it might be necessary to swap out of the drawers for a pull-out step stool that can be used to get in and out of the bed once the user reaches an older age.

The King of Kings TechBed is finished and will go to the possession of Jacob Klopfenstein at the completion of this project. The total estimate of time input on the project between all team members is between 500-600 hours from design to final bug fixes. This project represents the culmination of our collective knowledge from the courses we have taken from Miami University and displays our abilities and experience in both building something with mechanical, electrical, and programming aspects, along with the management of a complex project.

References

[1] "The Ultimate Bed With Integrated Massage Chair, Speakers and Desk." Website. ultimatesmartbed.com/product/ultimate-bed/ (September 10, 2020).

[2] "Coolest High Tech Smart Beds." Website. furniturefashion.com/10-of-the-coolest-high-tech-beds/ (September 10, 2020).

[3] "Frigidaire – 3.1 Cu. Ft. Mini Fridge." Website. bestbuy.com/site/frigidaire-3-1-cu-ft-mini-fridge-silver/6257387.p?skuId=6257387 (September 10, 2020).

[4] "Insignia – 1.7 Cu. Ft. Mini Fridge." Website. bestbuy.com/site/insignia-1-7-cu-ft-mini-fridge-black/6145100.p?skuId=6145100 (September 10, 2020).

[5] "Vizio – 5.1.2-Channel Soundbar System." Website. bestbuy.com/site/vizio-5-1-2-channel-soundbar-system-with-6-wireless-subwoofer-and-dolby-atmos-black/6288824.p?skuId=6288824 (September 13, 2020).

[6] "Giderwel RGBWW LED Strip Lights." Website. amazon.com/GIDERWEL-Flexible-300LEDsChangingKitchen/dp/B07JKNFMC3/ (September 15, 2020).

[7] Mike Kwaitkowski, private communication, September, 2020.

[8] "Adafruit NeoPixel Digital RGBW LED Strip – Black PCB 60 LED/m." Website. adafruit.com/product/2837?length=1 (September 14, 2020).

[9] "Highfine 12cm 12mm 200CFM 4000RPM CPU Cooling Fan." Website. amazon.com/HIGHFINE4000RPMCoolingFFC1212DEComputer/dp/B01LLYQ2VE/ (October 1, 2020).

[10] "Antec 120mm Case Fan." Website. amazon.com/Antec-F12-Performance-NoiseValue/dp/B07PFBPHL6/ (October 1, 2020).

[11] "Raspberry Pi4 Model B Quad Core 64 Bit." Website. amazon.com/Raspberry-Model-QuadCoreBluetooth/dp/B08C4SK5C3/ (September 14, 2020).

[12] "Raspberry Pi 7" Touch Screen Display." Website. amazon.com/Raspberry-Pi-7-TouchscreenDisplay/dp/B0153R2A9I/ (September 14, 2020).

[13] Randy Wise, private communication, September, 2020.

[14] Andrew Shultz, private communication, October, 2020.

[15] “Waithai 12038 120mmx38mm 5300rpm High Airflow 12V 4pin PWM FG DC Brushless Cooling Fan.” Website. amazon.com/Wathai-5300rpm-Airflow-Brushless-Cooling/dp/B07SGWNV5J/ (January 28, 2021).

Appendix A: Electrical Design

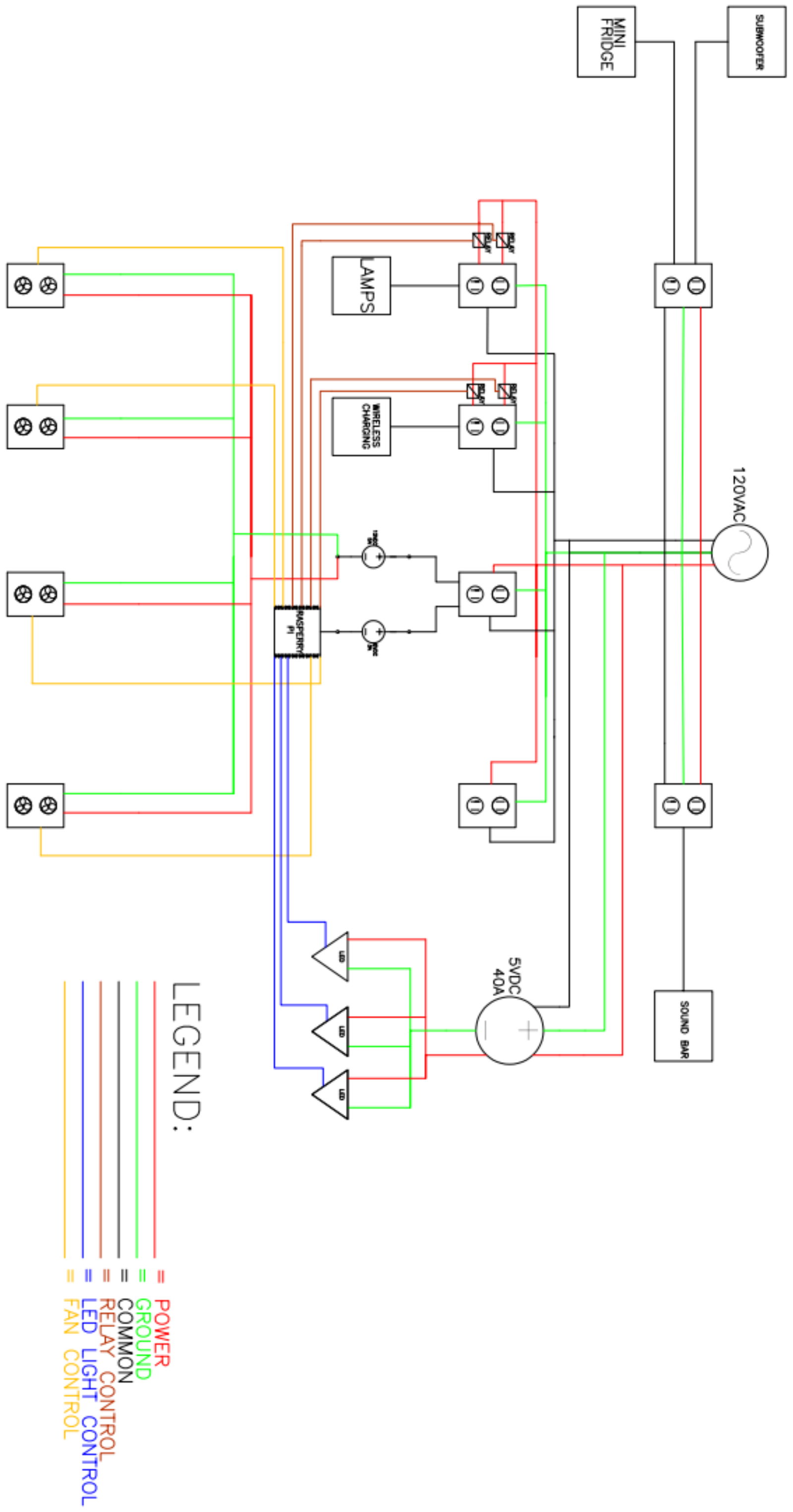


Figure 57: Origional Electrical Design

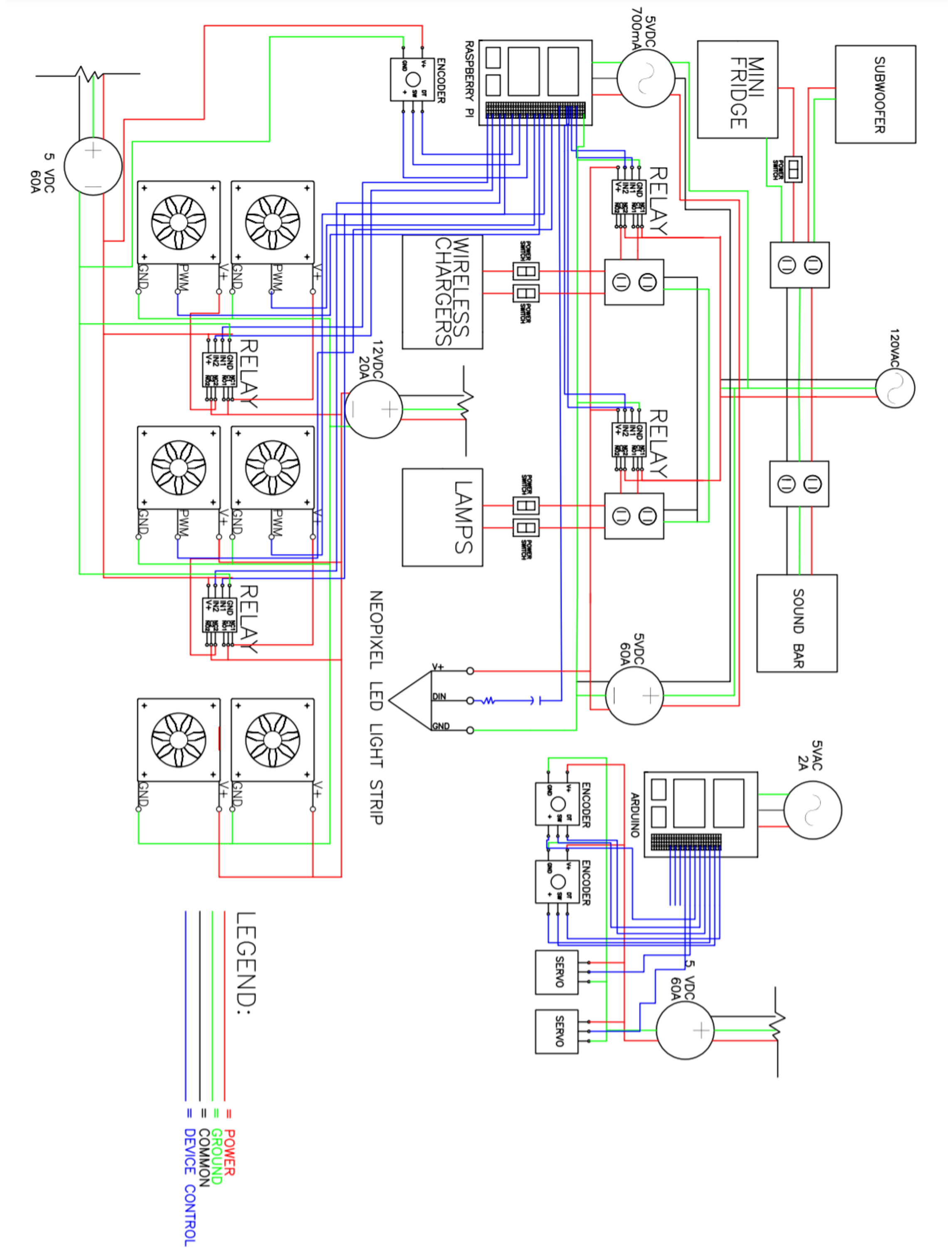


Figure 58: Final Electrical Design

Appendix B: Mechanical Design

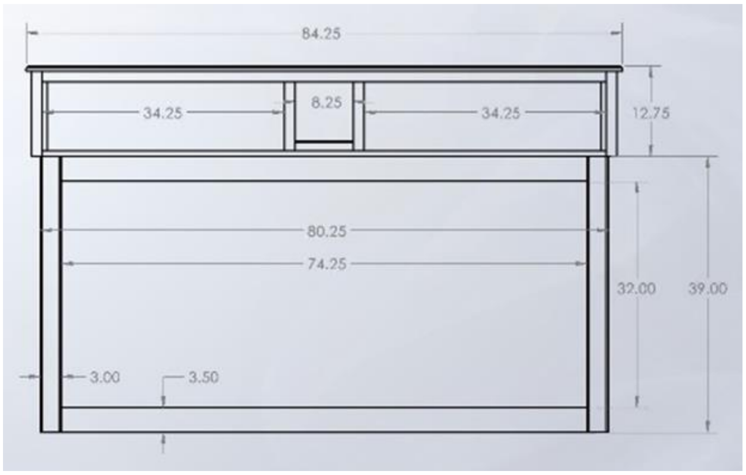


Figure 60: Headboard Front View

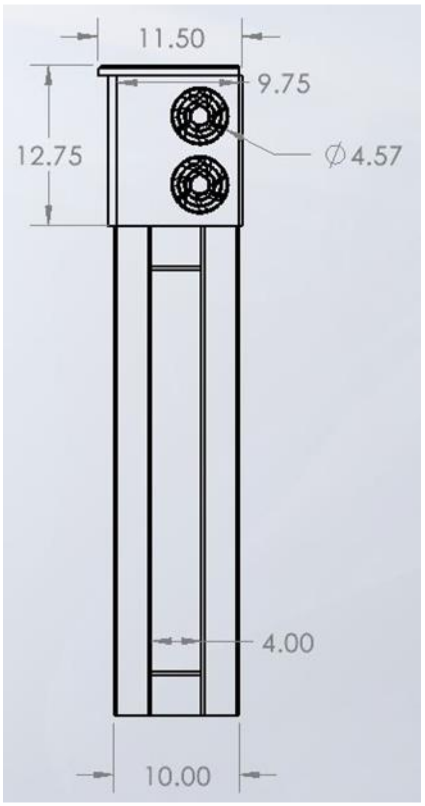


Figure 59: Headboard Side View

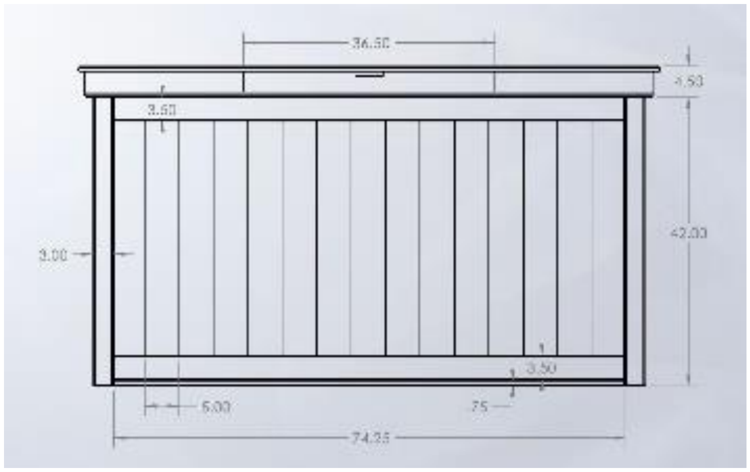


Figure 61: Footboard Front View

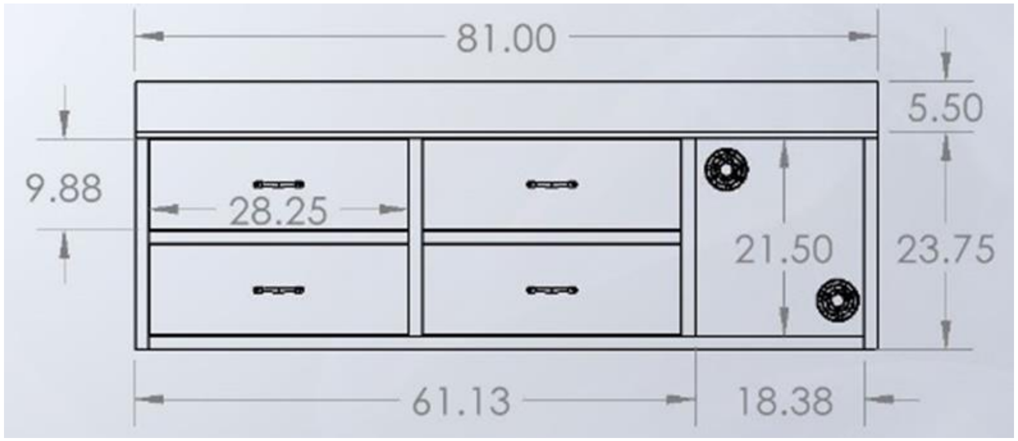


Figure 62: Main Frame Side View

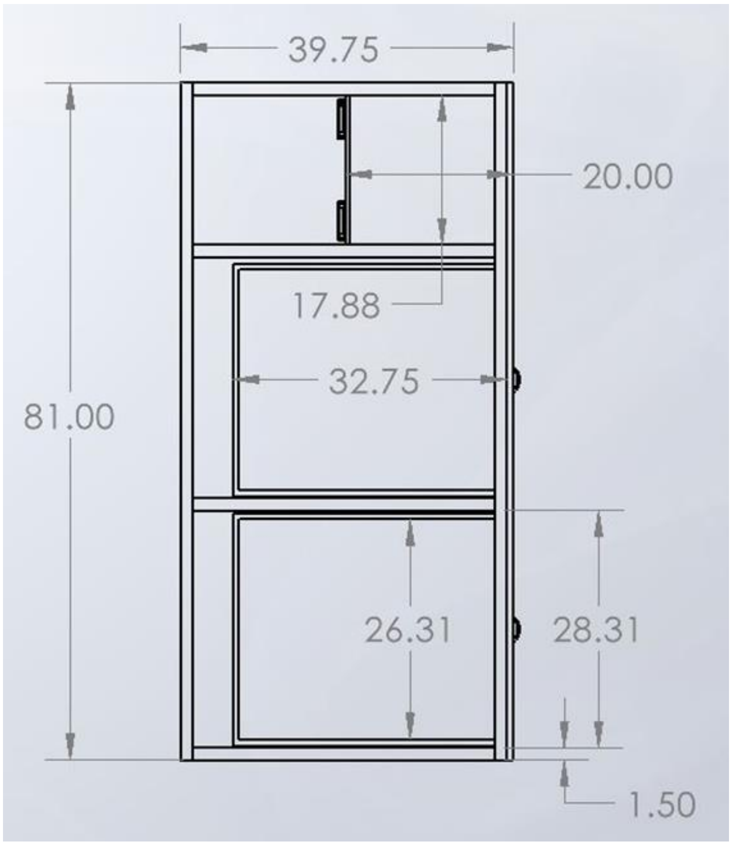


Figure 64: Main Frame Top View

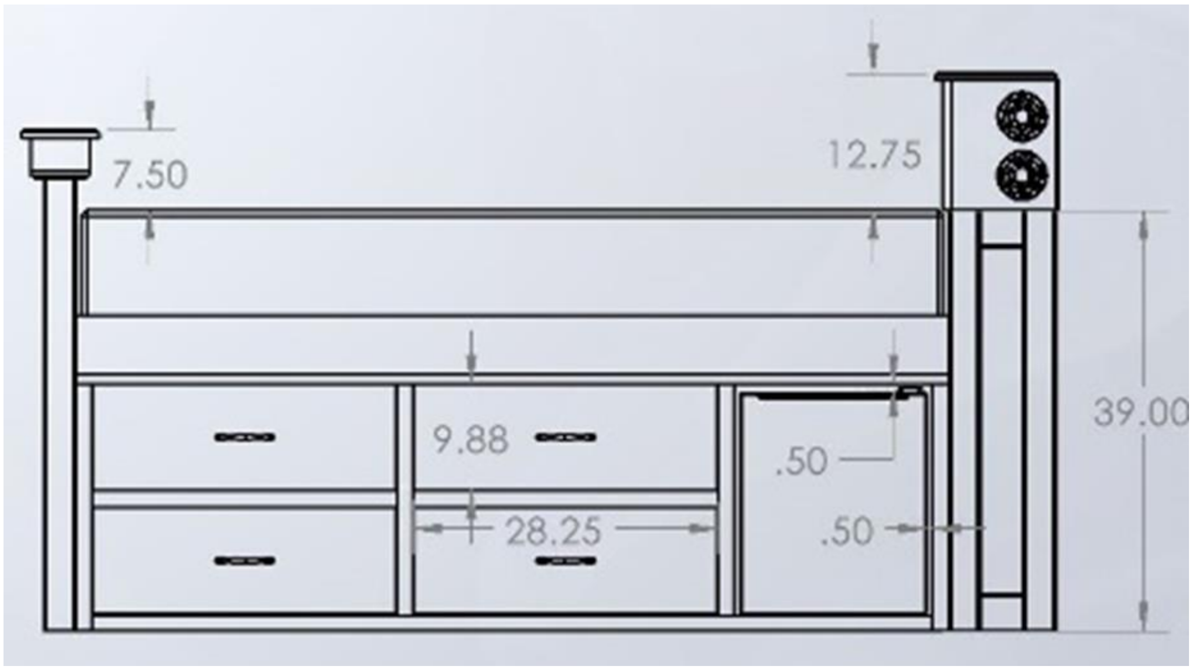


Figure 63: Full Bed Side View

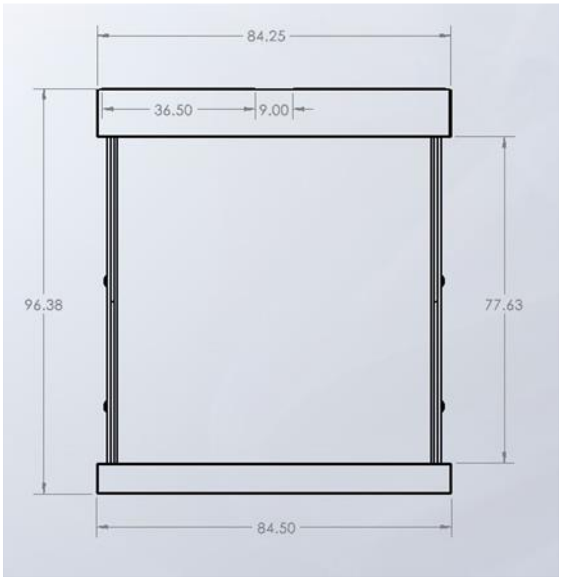


Figure 66: Full Bed Top View

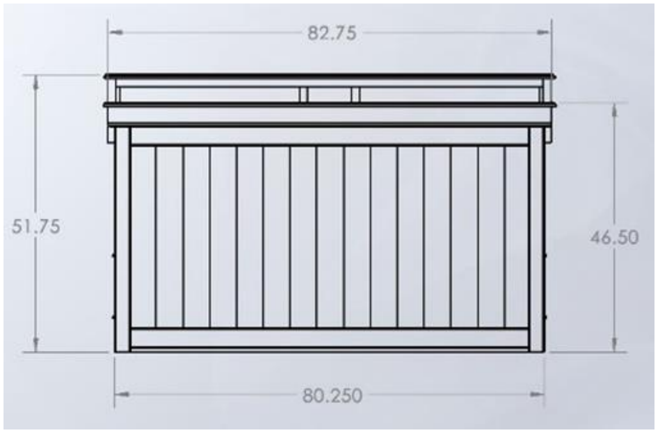


Figure 67: Full Bed Foot View

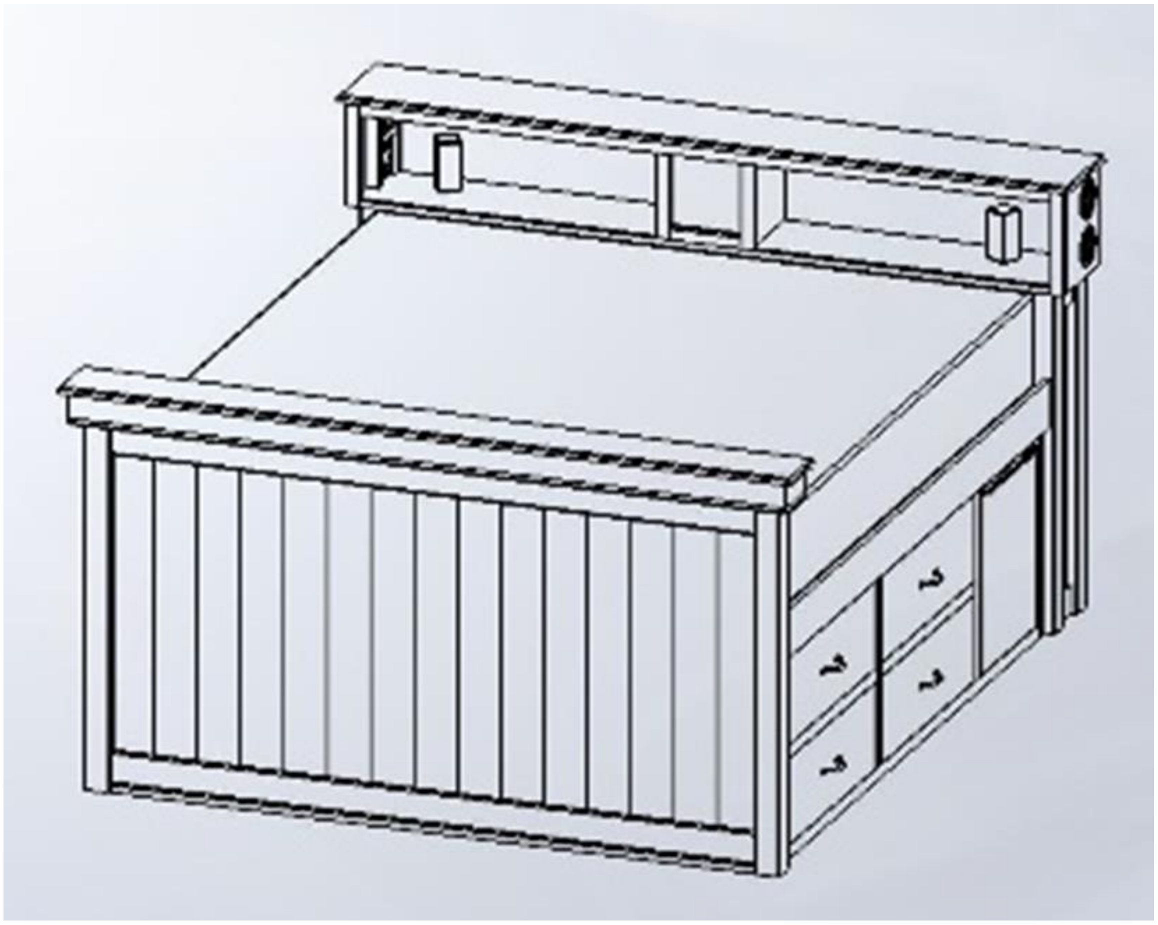
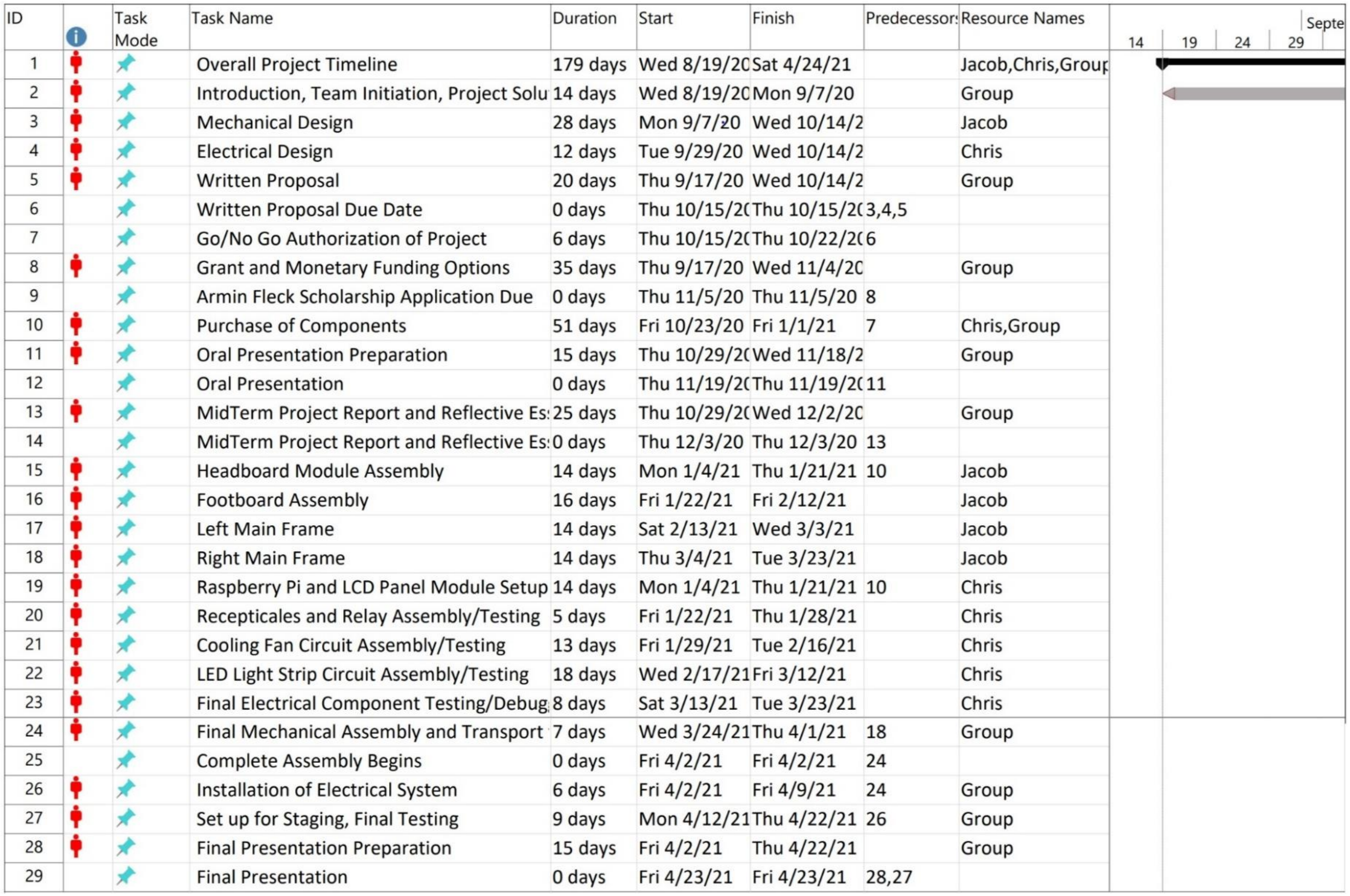
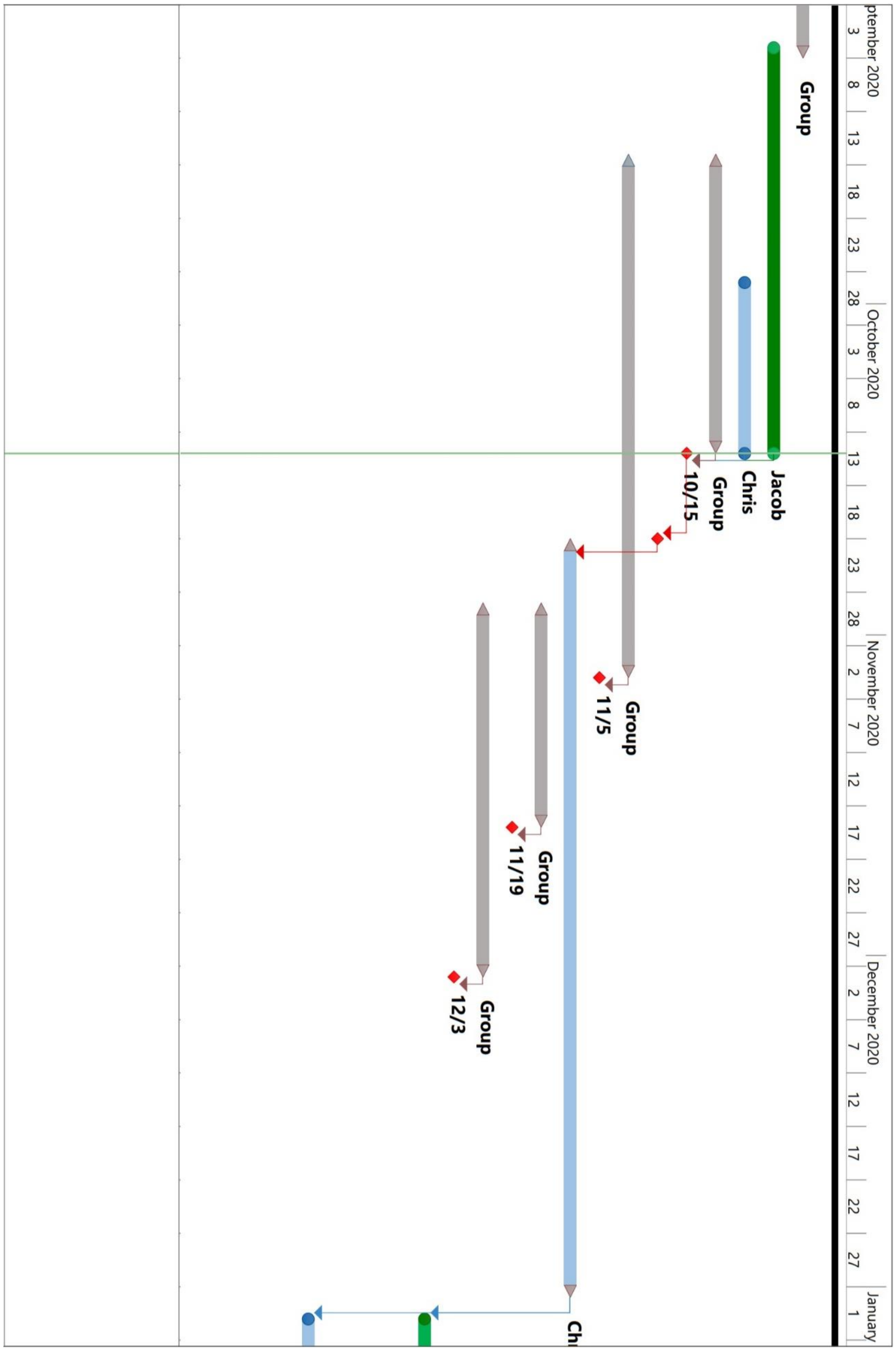


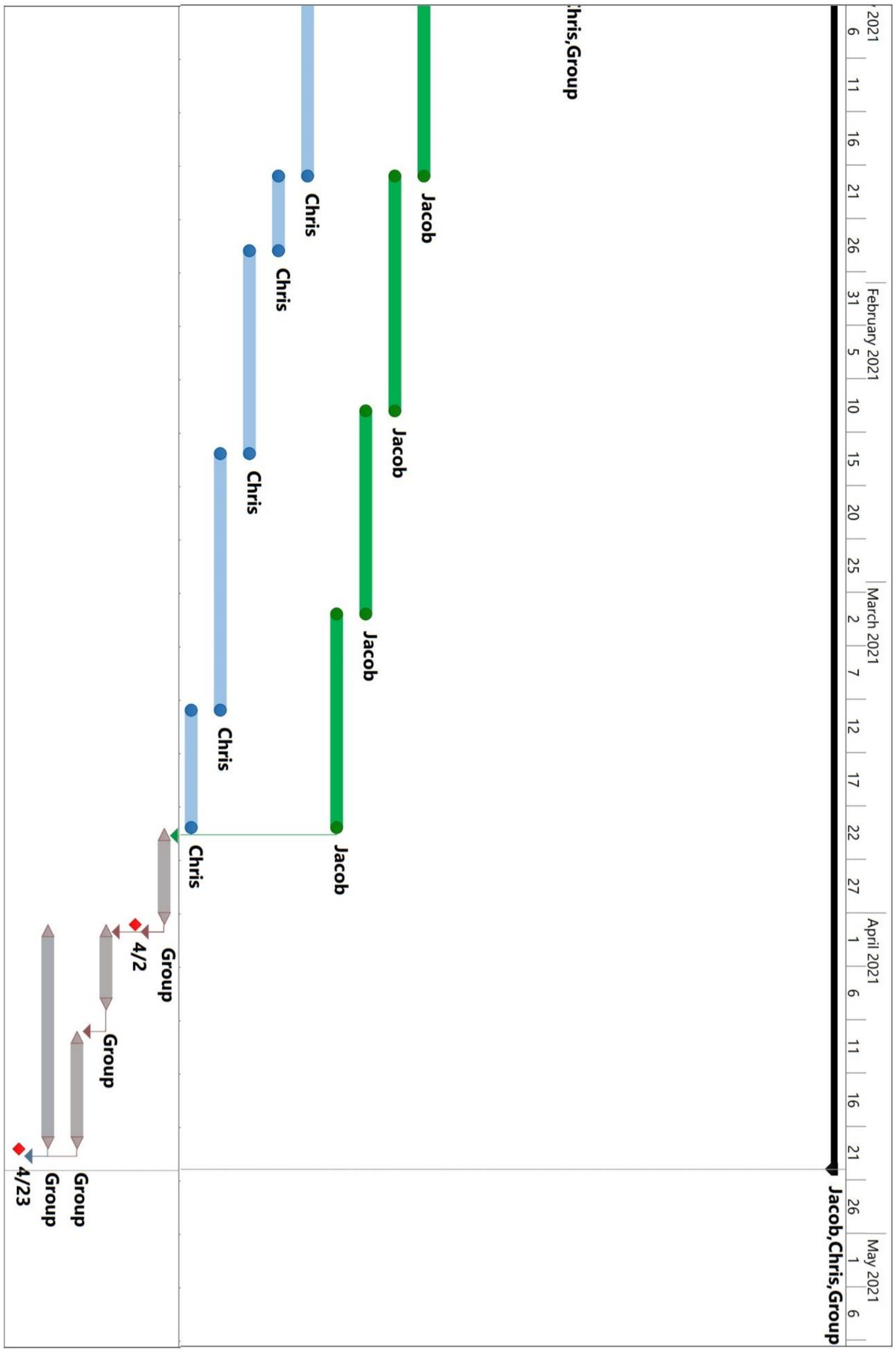
Figure 65: Dimetric View of Complete Bed with Technology Installed

Appendix C: Gantt Chart


Septe
14
19
24
29

ID	ℹ	Task Mode	Task Name	Duration	Start	Finish	Predecessors	Resource Names
1	👤	📌	Overall Project Timeline	179 days	Wed 8/19/20	Sat 4/24/21		Jacob,Chris,Group
2	👤	📌	Introduction, Team Initiation, Project Solu	14 days	Wed 8/19/20	Mon 9/7/20		Group
3	👤	📌	Mechanical Design	28 days	Mon 9/7/20	Wed 10/14/2		Jacob
4	👤	📌	Electrical Design	12 days	Tue 9/29/20	Wed 10/14/2		Chris
5	👤	📌	Written Proposal	20 days	Thu 9/17/20	Wed 10/14/2		Group
6		📌	Written Proposal Due Date	0 days	Thu 10/15/2(	Thu 10/15/2(	3,4,5	
7		📌	Go/No Go Authorization of Project	6 days	Thu 10/15/2(	Thu 10/22/2(	6	
8	👤	📌	Grant and Monetary Funding Options	35 days	Thu 9/17/20	Wed 11/4/20		Group
9		📌	Armin Fleck Scholarship Application Due	0 days	Thu 11/5/20	Thu 11/5/20	8	
10	👤	📌	Purchase of Components	51 days	Fri 10/23/20	Fri 1/1/21	7	Chris,Group
11	👤	📌	Oral Presentation Preparation	15 days	Thu 10/29/2(	Wed 11/18/2		Group
12		📌	Oral Presentation	0 days	Thu 11/19/2(	Thu 11/19/2(	11	
13	👤	📌	MidTerm Project Report and Reflective Es:	25 days	Thu 10/29/2(	Wed 12/2/20		Group
14		📌	MidTerm Project Report and Reflective Es:	0 days	Thu 12/3/20	Thu 12/3/20	13	
15	👤	📌	Headboard Module Assembly	14 days	Mon 1/4/21	Thu 1/21/21	10	Jacob
16	👤	📌	Footboard Assembly	16 days	Fri 1/22/21	Fri 2/12/21		Jacob
17	👤	📌	Left Main Frame	14 days	Sat 2/13/21	Wed 3/3/21		Jacob
18	👤	📌	Right Main Frame	14 days	Thu 3/4/21	Tue 3/23/21		Jacob
19	👤	📌	Raspberry Pi and LCD Panel Module Setup	14 days	Mon 1/4/21	Thu 1/21/21	10	Chris
20	👤	📌	Recepticales and Relay Assembly/Testing	5 days	Fri 1/22/21	Thu 1/28/21		Chris
21	👤	📌	Cooling Fan Circuit Assembly/Testing	13 days	Fri 1/29/21	Tue 2/16/21		Chris
22	👤	📌	LED Light Strip Circuit Assembly/Testing	18 days	Wed 2/17/21	Fri 3/12/21		Chris
23	👤	📌	Final Electrical Component Testing/Debug	8 days	Sat 3/13/21	Tue 3/23/21		Chris
24	👤	📌	Final Mechanical Assembly and Transport	7 days	Wed 3/24/21	Thu 4/1/21	18	Group
25		📌	Complete Assembly Begins	0 days	Fri 4/2/21	Fri 4/2/21	24	
26	👤	📌	Installation of Electrical System	6 days	Fri 4/2/21	Fri 4/9/21	24	Group
27	👤	📌	Set up for Staging, Final Testing	9 days	Mon 4/12/21	Thu 4/22/21	26	Group
28	👤	📌	Final Presentation Preparation	15 days	Fri 4/2/21	Thu 4/22/21		Group
29		📌	Final Presentation	0 days	Fri 4/23/21	Fri 4/23/21	28,27	


ptember 2020
3
8
13
18
23
28
October 2020
3
8
13
18
23
28
November 2020
2
7
12
17
22
27
December 2020
2
7
12
17
22
27
January
1
Group
Jacob
Chris
Group
10/15
Group
11/5
Group
11/19
Group
12/3
Ch


2021
February 2021
March 2021
April 2021
May 2021
6
11
16
21
26
31
5
10
15
20
25
2
7
12
17
22
27
1
6
11
16
21
26
1
6
hris,Group
Jacob
Chris
Chris
Jacob
Chris
Jacob
Chris
Jacob
Chris
Group
4/2
Group
Group
Group
4/23
Jacob,Chris,Group

Appendix D: Main Python Program

```
#Importing Needed Modules and Libraries
from kivy.uix.screenmanager import ScreenManager, Screen
from dateutil import parser
from kivy.properties import StringProperty
from kivy.properties import ListProperty
from kivy.core.text import LabelBase
from rpi_backlight import Backlight
from kivy.uix.popup import Popup
from kivy.lang import Builder
from kivy.clock import Clock
from kivy.app import App
import RPi.GPIO as GPIO
from adafruit_led_animation.animation.blink import Blink
from adafruit_led_animation.animation.colorcycle import ColorCycle
from adafruit_led_animation.animation.chase import Chase
from adafruit_led_animation.animation.comet import Comet
from adafruit_led_animation.animation.pulse import Pulse
from adafruit_led_animation.animation.rainbow import Rainbow
from adafruit_led_animation.animation.rainbowchase import RainbowChase
from adafruit_led_animation.animation.rainbowcomet import RainbowComet
from adafruit_led_animation.animation.rainbowsparkle import RainbowSparkle
from adafruit_led_animation.animation.sparkle import Sparkle
from adafruit_led_animation.animation.sparklepulse import SparklePulse
from adafruit_led_animation.color import (
    PURPLE,
    YELLOW,
    WHITE,
    AMBER,
    JADE,
    TEAL,
    PINK,
    MAGENTA,
    ORANGE,
    BLUE,
    RED,
    GREEN,
    GOLD,
    AQUA,
    CYAN,
)
import threading
import board
import neopixel
import time
import busio
import digitalio
import pygame
import datetime
import Encoder
from multiprocessing import Process
import re
import sys
from decimal import *
import sched, time
from os import system
getcontext().prec = 2
# importing Centaur font
LabelBase.register(name = "Centaur",
        fn_regular = "centaur.ttf")

pixel_pin = board.D10
num_pixels = 491
ORDER = neopixel.GRBW
pixels = neopixel.NeoPixel(pixel_pin, num_pixels, brightness=1,
                auto_write=False, pixel_order=ORDER)
#Initilizing Variables for GPIOLWC = 17
LWC = 17
RWC = 4
LL =  22
RL = 27
RFan = 13
LFan = 19
LFanR = 7
RFanR = 9

GPIO.setmode(GPIO.BCM)
#Setting up desired GPIO pins as outputs
GPIO.setup(LWC, GPIO.OUT)
GPIO.setup(RWC, GPIO.OUT)
GPIO.setup(LL, GPIO.OUT)
GPIO.setup(RL, GPIO.OUT)
GPIO.setup(LFan, GPIO.OUT)
GPIO.setup(RFan, GPIO.OUT)
GPIO.setup(LFanR, GPIO.OUT)
GPIO.setup(RFanR, GPIO.OUT)

#Initializing pwm pins
pwm1 = GPIO.PWM(LFan, 25000)
pwm1.start(0)
pwm2 = GPIO.PWM(RFan, 25000)
pwm2.start(0)


pygame.mixer.init()
pygame.mixer.music.load("Birdsong.mp3")


Brightness = Encoder.Encoder(23,24)
backlight = Backlight()

GPIO.setup(25, GPIO.IN, GPIO.PUD_DOWN)
GPIO.setup(1, GPIO.IN, GPIO.PUD_DOWN)


global Count
Count = 0
global y
y = 0

def bEncode(channel):
    global y
    global Count
    x = Brightness.read()

    if x > y and Count < 76:
        Count = Count + 10


    if x < y and Count > 20:
        Count = Count - 10
    if x < y and Count <= 20:
        Count = Count - 5
        if Count < 1:
            Count = 3

    backlight.brightness = Count

    y = x


GPIO.add_event_detect(25, GPIO.BOTH, callback = bEncode, bouncetime=100)

currentAlarm = ("Birdsong.mp3")

def toggleWhiteNoise(self):
    pygame.mixer.init()

    if(pygame.mixer.music.get_busy() == True):
        pygame.mixer.music.pause()
        app = App.get_running_app()
        app.root.get_screen('whitenoise').ids.pause.state = "down"
        app.root.get_screen('whitenoise').ids.pause.source = '/home/pi/Desktop/Final
GUI/activated icons/pause.png'

    else:
        pygame.mixer.music.unpause()
        app = App.get_running_app()
        app.root.get_screen('whitenoise').ids.pause.state = 'normal'

GPIO.add_event_detect(1, GPIO.BOTH, callback = toggleWhiteNoise, bouncetime=500)

global r,g,b,w
r = 0
g = 0
b = 0
w = 0

global brightVal
brightVal = 0

class MainScreen(Screen):
```

```
def ROn(self, Pin):
    GPIO.output(Pin, GPIO.HIGH)

def ROff(self, Pin):
    GPIO.output(Pin, GPIO.LOW)

def LFanON(self, *args):
    state1 = GPIO.input(LFanR)
    if state1 == 1:
        self.ids.leftFan.state == 'normal'
        GPIO.output(LFanR, GPIO.LOW)
        pwm1.ChangeDutyCycle(25)
        self.ids.leftSlider.value = 25
    if state1 == 0:
        self.ids.leftFan.state == 'down'
        GPIO.output(LFanR, GPIO.HIGH)
        pwm1.ChangeDutyCycle(0)
        self.ids.leftSlider.value = 0

def RFanON(self, *args):
    state1 = GPIO.input(RFanR)
    if state1 == 1:
        self.ids.rightFan.state == 'normal'
        GPIO.output(RFanR, GPIO.LOW)
        pwm2.ChangeDutyCycle(25)
        self.ids.rightSlider.value = 25
    if state1 == 0:
        self.ids.rightFan.state == 'down'
        GPIO.output(RFanR, GPIO.HIGH)
        pwm2.ChangeDutyCycle(0)
        self.ids.rightSlider.value = 0
# Left Fan Speed function.
def LFanCS(self, instance, value1):
    pwm1.ChangeDutyCycle(int(value1))
# Right Fan speed function.
def RFanCS(self, instance, value2):
    pwm2.ChangeDutyCycle(int(value2))

def powerOff(self):
    GPIO.output(RFanR, GPIO.HIGH)
```

```
        GPIO.output(LFanR, GPIO.HIGH)
        GPIO.output(RWC, GPIO.HIGH)
        GPIO.output(LWC, GPIO.HIGH)
        GPIO.output(LL, GPIO.HIGH)
        GPIO.output(RL, GPIO.HIGH)
        pixels.fill((0,0,0,0))
        pixels.show()
        sys.exit()

#     GPIO.add_event_detect(1, GPIO.BOTH, callback = LFanON, bouncetime=500)

    global aBool
    aBool = False
    global lightBool
    lightBool = False
    def updateTime(self, *args):

        pattern1 = re.compile("^[1][0-2][:][0-5]\d")
        pattern2 = re.compile("^[1-9][:][0-5]\d")
        #datetibject created from datetime module
        now = datetime.datetime.now()
        #Setting Day of week value to variable DOW
        DOW = datetime.datetime.today().weekday()
        DOW = DOW
        #Setting Month of Year to variable MOY
        MOY = int(now.strftime('%-m'))
        #Setting Year value to variable YEAR
        YEAR = now.strftime('%Y')
        #Setting Day value to variable DAY
        DAY = now.strftime('%-d')
        #Converting  Day of Week Value to appropriate
        def Dint_to_DString(day):
            switcher = {
                0: "Monday",
                1: "Tuesday",
                2: "Wednesday",
                3: "Thursday",
                4: "Friday",
                5: "Saturday",
                6: "Sunday",
                }
            return switcher.get(day)

        def Mint_to_MString(month):
            switcher = {
                1: "January",
                2: "February",
                3: "March",
                4: "April",
                5: "May",
                6: "June",
                7: "July",
                8: "August",
                9: "September",
                10: "October",
                11: "November",
                12: "December",
                }
            return switcher.get(month)

        #Settng date and time label text to actual date and time
        self.ids.timeLabel.text = now.strftime('%-I:%M')
        self.ids.ampmLabel.text = now.strftime('%p')
        self.ids.dateLabel.text  = (Dint_to_DString(DOW) + ", " + Mint_to_MString(MOY) + " "
+ DAY + ", " + YEAR)
#         if(re.search(pattern1, now.strftime('%-I:%M'))):
        # Alarm function checking to see if Alarm time matches

        if((re.search(pattern1, now.strftime('%-I:%M')))):
            self.ids.ampmLabel.pos_hint = {'x' : .65, 'top' : .88}

        if((re.search(pattern2, now.strftime('%-I:%M')))):
            self.ids.ampmLabel.pos_hint = {'x' : .6, 'top' : .88}

        def open_alarm_popup(self):

            alarmPopup1 = AlarmPopUp()
            alarmPopup1.open()

        global currentAlarm
        global aBool
        if str(atime) == now.strftime('%-I:%M') and str(ampm) == now.strftime('%p') and aBool
== False:
            pygame.mixer.music.load(currentAlarm)
            pygame.mixer.music.play(loops=-1)
            open_alarm_popup(self)
            aBool = True


        if atime != "0":
            now = datetime.datetime.now()
            fivmin = "0:05"
            fivMinParse = datetime.datetime.strptime(fivmin, '%H:%M')
            alarmParse = datetime.datetime.strptime(atime, '%H:%M')
            fivMinBef = alarmParse - fivMinParse
            string1 = str(fivMinBef)
            stringMod = string1[0:5]

            if str(stringMod) == now.strftime("%-I:%M"):
                global lightBool
                lightBool = True

        global brightVal
        if lightBool == True and brightVal < 255:
            brightVal = int(brightVal + 4.25)
            pixels.fill((0,0,0,brightVal))
            pixels.show()


    def on_enter(self):
        global atime
        if atime == "0":
            self.ids.alarm1.source = '/home/pi/Desktop/Final GUI/icons/alarm.png'
        if atime != "0":
            self.ids.alarm1.source =  '/home/pi/Desktop/Final GUI/activated icons/alarm1.png'

    def on_kv_post(self, basewidget):
        Clock.schedule_once(self.updateTime)
        Clock.schedule_interval(self.updateTime, 5)

class AdvancedColor(Screen):
    global r,g,b,w


    def addWhite(self, instance, val):
        global w
        w = int(val)
        pixels.fill((r,g,b,w))
        pixels.show()

    def tuner(self, instance, val):
        global r, g
        r = int(val)
        g = int(val*.38)
        pixels.fill((r,g,b,w))
        pixels.show()

    def clearLights(self):
        self.ids.whiteSlider.value = 0
        self.ids.tunableSlider.value = 0
        global r
        global g
        global b
        global w

        r = 0
        g = 0
        b = 0
        w = 0

        pixels.fill((r,g,b,w))
        pixels.show()
        pixels.fill((0,0,0,0))
        pixels.show()

class LEDAnimations1(Screen):
    global speed1
    speed1 = 1
    global cString
    cString = "N/A"
    global size1
    size1 = 1
```

```
global colorTuple
colorTuple = (0,0,0,0)

def setSize(self, instance, val):
    global size1
    size1 = int(val)
    self.ids.sizevallabel.text = str(size1)

def setSpeed(self, instance, val):
    global speed1
    speedInt = int(val)
    speed1 = (speedInt/100)

    self.ids.speedvallabel.text = str(speed1)



def setColor(self, instance, val):
    global cString
    global colorTuple

    def num_to_color(color):
        switcher = {
            1: "RUBY RED",
            2: "CURSED RED",
            3: "AMBER",
            4: "YELLOW",
            5: "LIME",
            6: "EMERALD",
            7: "FORREST",
            8: "MINT",
            9: "AQUA",
            10: "BABY BLUE",
            11: "LIGHT BLUE",
            12: "SAPPHIRE",
            13: "DARK BLUE",
            14: "PALE BLUE",
            15: "VIOLET",
            16: "DARK PINK",
            17: "LIGHT PINK",
            18: "HOT PINK",
            19: "WHITE",
            20: "SOFT WHITE",
            }
        return switcher.get(color)

    cint = int(val)
    cString = num_to_color(cint)

    if cint == 1:
        self.ids.colorValLabel.color = (1,0,0,1)
        colorTuple = (255,0,0,0)
    if cint == 2:
        self.ids.colorValLabel.color = (100/255,0,0,1)
        colorTuple = (100,0,0,0)
    if cint == 3:
        self.ids.colorValLabel.color = (1,95/255,0,1)
        colorTuple = (255,95,0,0)
    if cint == 4:
        self.ids.colorValLabel.color = (1,1,0,1)
        colorTuple = (255,255,0,0)
    if cint == 5:
        self.ids.colorValLabel.color = (100/255,1,0,1)
        colorTuple = (100,255,0,0)
    if cint == 6:
        self.ids.colorValLabel.color = (0,1,0,1)
        colorTuple = (0,255,0,0)
    if cint == 7:
        self.ids.colorValLabel.color = (0,50/255,0,1)
        colorTuple = (0,50,0,0)
    if cint == 8:
        self.ids.colorValLabel.color = (0,1,50/255,1)
        colorTuple = (0,255,50,0)
    if cint == 9:
        self.ids.colorValLabel.color = (0,1,150/255,1)
        colorTuple = (0,255,150,0)
    if cint == 10:
        self.ids.colorValLabel.color = (0,1,1,1)
        colorTuple = (0,255,255,0)
    if cint == 11:
        self.ids.colorValLabel.color = (0,85/255,1,1)
        colorTuple = (0,85,255,0)
    if cint == 12:
        self.ids.colorValLabel.color = (0,0,1,1)
        colorTuple = (0,0,255,0)
    if cint == 13:
        self.ids.colorValLabel.color = (0,0,100/255,1)
        colorTuple = (0,0,100,0)
    if cint == 14:
        self.ids.colorValLabel.color = (50/255,0,1,1)
        colorTuple = (51,0,255,0)
    if cint == 15:
        self.ids.colorValLabel.color = (125/255,0,1,1)
        colorTuple = (125,0,255,0)
    if cint == 16:
        self.ids.colorValLabel.color = (1,0,1,1)
        colorTuple = (255,0,255,0)
    if cint == 17:
        self.ids.colorValLabel.color = (1,60/255,150/255,1)
        colorTuple = (255,5,100,100)
    if cint == 18:
        self.ids.colorValLabel.color = (1,0,105/255,1)
        colorTuple = (255,0,75,25)
    if cint == 19:
        self.ids.colorValLabel.color = (1,1,1,1)
        colorTuple = (0,0,0,255)
    if cint == 20:
        self.ids.colorValLabel.color = (50/255,50/255,50/255,1)
        colorTuple = (0,0,0,50)


    self.ids.colorValLabel.text = str(cString)

def setAnimationState(self, animationId):

    global blinkBool
    global cometBool
    global chaseBool
    global colorcycleBool
    global pulseBool
    global sparkleBool
    global sparklepulseBool

    blinkBool = False
    cometBool = False
    chaseBool = False
    pulseBool = False
    sparkleBool = False
    sparklepulseBool = False

    if animationId == 1:
        blinkBool = True
    if animationId == 2:
        cometBool = True
    if animationId == 3:
        chaseBool = True
    if animationId == 4:
        pulseBool = True
    if animationId == 5:
        sparkleBool = True
    if animationId == 6:
        sparklepulseBool = True

global stop_thread
stop_thread = True

def startThread(self):

    global speed
    global cString
    global stop_thread
    global size1
    stop_thread = False
    blink = Blink(pixels, speed = speed1, color = colorTuple)
    comet = Comet(pixels, speed = speed1, color = colorTuple, tail_length = size1, bounce = 
True)
    chase = Chase(pixels, speed = speed1, size = size1, spacing = 6, color = BLUE)
    pulse = Pulse(pixels, speed = speed1, color = colorTuple, period = 3)

    def startAnimation():

        global stop_thread
        global blinkBool
        global cometBool
        global chaseBool
        global pulseBool
        global sparkleBool
        global sparklepulseBool
        blink = Blink(pixels, speed = .75, color = colorTuple)
        chase = Chase(pixels, speed = .001, size = size1, spacing = 6, color = colorTuple)

        comet = Comet(pixels, speed = .001, color = colorTuple, tail_length = size1,
                bounce = True)
        pulse = Pulse(pixels, speed = (size1/500), color = colorTuple, period = size1)
        sparkle = Sparkle(pixels, speed = (size1/100), color = colorTuple, num_sparkles =
size1)
        sparkle_pulse = SparklePulse(pixels, speed = (size1/500), color = colorTuple)

        while blinkBool == True and stop_thread == False:
            blink.animate()
            if stop_thread == True:
                break
        while cometBool == True and stop_thread == False:
            comet.animate()
            if stop_thread == True:
                break
        while chaseBool == True and stop_thread == False:
            chase.animate()
            if stop_thread == True:
                break
        while pulseBool == True and stop_thread == False:
            pulse.animate()
            time.sleep(.001)
            if stop_thread == True:
                break
        while sparkleBool == True and stop_thread == False:
            sparkle.animate()
            if stop_thread == True:
                break
        while sparklepulseBool == True and stop_thread == False:
            sparkle_pulse.animate()
            if stop_thread == True:
                break

    t1 = threading.Thread(target = startAnimation)
    t1.start()
def stopThread(self):
    global stop_thread
    stop_thread = True
    pixels.fill((0,0,0,0))
    pixels.show()
    time.sleep(.5)
    pixels.fill((0,0,0,0))
    pixels.show()


def updateTime2(self, *args):
    pattern1 = re.compile("^[1][0-2][:][0-5]\d")
    pattern2 = re.compile("^[1-9][:][0-5]\d")

    now = datetime.datetime.now()
    self.ids.timeLabel.text = now.strftime('%-I:%M')
    self.ids.ampmLabel.text = now.strftime('%p')

    if(re.search(pattern1, now.strftime('%-I:%M'))):
        self.ids.ampmLabel.pos_hint = {'x' : .87, 'top' : .95}

    if(re.search(pattern2, now.strftime('%-I:%M'))):
        self.ids.ampmLabel.pos_hint = {'x' : .82, 'top' : .95}

def on_kv_post(self, basewidget):
    Clock.schedule_once(self.updateTime2)
    Clock.schedule_interval(self.updateTime2, 3)

class LEDAnimations2(Screen):
global speed2
speed2 = 1
global cString1
cString1 = "N/A"
global size2
size2 = 1
global colorTuple2
colorTuple2 = (0,0,0,0)
global rainbow1Bool
rainbow1Bool = False
def setSize(self, instance, val):
    global size2
    size2 = int(val)
    self.ids.sizevallabel.text = str(size2)

def setSpeed(self, instance, val):
    global speed2
    speedInt = int(val)
    speed2 = (speedInt/100)

    self.ids.speedvallabel.text = str(speed2)



def setColor(self, instance, val):
    global cString1
    global colorTuple2

    def num_to_color(color):
        switcher = {
            1: "RUBY RED",
            2: "CURSED RED",
            3: "AMBER",
            4: "YELLOW",
            5: "LIME",
            6: "EMERALD",
            7: "FORREST",
            8: "MINT",
            9: "AQUA",
            10: "BABY BLUE",
            11: "LIGHT BLUE",
            12: "SAPPHIRE",
            13: "DARK BLUE",
            14: "PALE BLUE",
            15: "VIOLET",
            16: "DARK PINK",
            17: "LIGHT PINK",
            18: "HOT PINK",
            19: "WHITE",
            20: "SOFT WHITE",
            }
        return switcher.get(color)
```

```
cint = int(val)
cString = num_to_color(cint)

if cint == 1:
    self.ids.colorValLabel.color = (1,0,0,1)
    colorTuple = (255,0,0,0)
if cint == 2:
    self.ids.colorValLabel.color = (100/255,0,0,1)
    colorTuple = (100,0,0,0)
if cint == 3:
    self.ids.colorValLabel.color = (1,95/255,0,1)
    colorTuple = (255,95,0,0)
if cint == 4:
    self.ids.colorValLabel.color = (1,1,0,1)
    colorTuple = (255,255,0,0)
if cint == 5:
    self.ids.colorValLabel.color = (100/255,1,0,1)
    colorTuple = (100,255,0,0)
if cint == 6:
    self.ids.colorValLabel.color = (0,1,0,1)
    colorTuple = (0,255,0,0)
if cint == 7:
    self.ids.colorValLabel.color = (0,50/255,0,1)
    colorTuple = (0,50,0,0)
if cint == 8:
    self.ids.colorValLabel.color = (0,1,50/255,1)
    colorTuple = (0,255,50,0)
if cint == 9:
    self.ids.colorValLabel.color = (0,1,150/255,1)
    colorTuple = (0,255,150,0)
if cint == 10:
    self.ids.colorValLabel.color = (0,1,1,1)
    colorTuple = (0,255,255,0)
if cint == 11:
    self.ids.colorValLabel.color = (0,85/255,1,1)
    colorTuple = (0,85,255,0)
if cint == 12:
    self.ids.colorValLabel.color = (0,0,1,1)
    colorTuple = (0,0,255,0)
if cint == 13:
```

```
            self.ids.colorValLabel.color = (0,0,100/255,1)
            colorTuple = (0,0,100,0)
        if cint == 14:
            self.ids.colorValLabel.color = (50/255,0,1,1)
            colorTuple = (51,0,255,0)
        if cint == 15:
            self.ids.colorValLabel.color = (125/255,0,1,1)
            colorTuple = (125,0,255,0)
        if cint == 16:
            self.ids.colorValLabel.color = (1,0,1,1)
            colorTuple = (255,0,255,0)
        if cint == 17:
            self.ids.colorValLabel.color = (1,60/255,150/255,1)
            colorTuple = (255,5,100,100)
        if cint == 18:
            self.ids.colorValLabel.color = (1,0,105/255,1)
            colorTuple = (255,0,75,25)
        if cint == 19:
            self.ids.colorValLabel.color = (1,1,1,1)
            colorTuple = (0,0,0,255)
        if cint == 20:
            self.ids.colorValLabel.color = (50/255,50/255,50/255,1)
            colorTuple = (0,0,0,50)


        self.ids.colorValLabel.text = str(cString1)

    def setAnimationState(self, animationId):

        global rainbow1Bool
        global rainbow2Bool
        global rainbowChaseBool
        global rainbowCometBool
        global rainbowSparkleBool
        global colorCycleBool

        rainbow1Bool = False
        rainbow2Bool = False
        rainbowChaseBool = False
        rainbowCometBool = False
        rainbowSparkleBool = False
        colorCycleBool = False

        if animationId == 1:
            rainbow1Bool = True
        if animationId == 2:
            rainbow2Bool = True
        if animationId == 3:
            rainbowChaseBool = True
        if animationId == 4:
            rainbowCometBool = True
        if animationId == 5:
            rainbowSparkleBool = True
        if animationId == 6:
            colorCycleBool = True

    global stop_thread
    stop_thread = True


    def startThread(self):

        global speed
        global cString
        global stop_thread
        global size2
        global threadActive2
        stop_thread = False
        rainbow = Rainbow(pixels, speed = speed2, period = size2)
        rainbow_chase = RainbowChase(pixels, speed = speed2, size = size2, spacing = 5)
        rainbow_comet = RainbowComet(pixels, speed = .001, tail_length = size2, bounce =
True)
        colorcycle = ColorCycle(pixels, speed = 2, colors = [RED, YELLOW, GREEN, GOLD,
BLUE, PURPLE, TEAL, PINK, AQUA, WHITE])
        rainbow_sparkle = RainbowSparkle(pixels, speed = speed2, num_sparkles = size2)

        def startAnimation():

            global size1
            global rainbow1Bool
            global rainbow2Bool
            global rainbowChaseBool
            global rainbowCometBool
            global rainbowSparkleBool
            global colorCycleBool

            rainbow = Rainbow(pixels, speed = speed2, period = size2)
            rainbow_chase = RainbowChase(pixels, speed = speed2, size = size2, spacing = 3)
            rainbow_comet = RainbowComet(pixels, speed = .001, tail_length = size2, bounce = 
True)
            colorcycle = ColorCycle(pixels, speed = 2, colors = [RED, YELLOW, GREEN, 
GOLD, BLUE, PURPLE, TEAL, PINK, AQUA, WHITE])

            def wheel(pos):

                if pos < 0 or pos > 255:
                    r = g = b = 0
                elif pos < 85:
                    r = int(pos * 3)
                    g = int(255 - pos * 3)
                    b = 0
                elif pos < 170:
                    pos -= 85
                    r = int(255 - pos * 3)
                    g = 0
                    b = int(pos * 3)
                else:
                    pos -= 170
                    r = 0
                    g = int(pos * 3)
                    b = int(255 - pos * 3)
                return (r, g, b, 0)


            def rainbow_cycle(wait):
                global size2
                for j in range(255):
                    for i in range(num_pixels):
                        pixel_index = (i * 256 // (num_pixels // size2)) + j
                        pixels[i] = wheel(pixel_index & 255)
                    pixels.show()
                    time.sleep(wait)
            while rainbow1Bool == True and stop_thread == False:
                rainbow_cycle(.001)
                if stop_thread == True:
                    break
            while rainbow2Bool == True and stop_thread == False:
                rainbow.animate()
                if stop_thread == True:
                    break
            while rainbowChaseBool == True and stop_thread == False:
                rainbow_chase.animate()
                if stop_thread == True:
                    break
            while rainbowCometBool == True and stop_thread == False:
                rainbow_comet.animate()
                time.sleep(.001)
                if stop_thread == True:
                    break
            while rainbowSparkleBool == True and stop_thread == False:
                rainbow_sparkle.animate()
                if stop_thread == True:
                    break
            while colorCycleBool == True and stop_thread == False:
                colorcycle.animate()
                if stop_thread == True:
                    break

        t1 = threading.Thread(target = startAnimation)
        t1.start()

    def stopThread(self):
        global rainbow1Bool
        if rainbow1Bool == True:
            raPU = rainbowAnimationPU()
            raPU.open()
        global stop_thread
        stop_thread = True
        pixels.fill((0,0,0,0))
        pixels.show()
        time.sleep(.5)
```

```
        pixels.fill((0,0,0,0))
        pixels.show()

    def updateTime2(self, *args):

        pattern1 = re.compile("^[1][0-2][:][0-5]\d")
        pattern2 = re.compile("^[1-9][:][0-5]\d")

        now = datetime.datetime.now()
        self.ids.timeLabel.text = now.strftime('%-I:%M')
        self.ids.ampmLabel.text = now.strftime('%p')

        if(re.search(pattern1, now.strftime('%-I:%M'))):
            self.ids.ampmLabel.pos_hint = {'x' : .87, 'top' : .95}

        if(re.search(pattern2, now.strftime('%-I:%M'))):
            self.ids.ampmLabel.pos_hint = {'x' : .82, 'top' : .95}

    def on_kv_post(self, basewidget):
        Clock.schedule_once(self.updateTime2)
        Clock.schedule_interval(self.updateTime2, 3)


class LEDColor(Screen):

    def setColor(self):
        global r
        global g
        global b

        r = 0
        g = 0
        b = 0

        pixels.fill((0,0,0,0))
        pixels.show()

        red = Decimal(self.ids.colorpicker.color[0])
        red = float(red) * 255
        r = int(red)
        green = Decimal(self.ids.colorpicker.color[1])
        green = float(green) * 255
        g = int(green)
        blue = Decimal(self.ids.colorpicker.color[2])
        float(blue)
        blue = float(blue) * 255
        b = int(blue)
        alpha = Decimal(self.ids.colorpicker.color[3])
        a = int(alpha)

        pixels.fill((r,g,b,a))
        pixels.show()

    def clearLights(self):
        global r
        global g
        global b
        global a

        r = 0
        g = 0
        b = 0
        a = 0

        pixels.fill((r,g,b,a))
        pixels.show()

class AlarmScreen(Screen):

    global atime
    atime = "0"
    global ampm
    ampm = ""
    pygame.mixer.music.load("Birdsong.mp3")

    def setAlarm(self):
        global atime
        global aBool
        aBool = False
        atime = self.inp1.text
        pattern1 = re.compile("^[1][0-2][:][0-5]\d")
        pattern2 = re.compile("^[1-9][:][0-5]\d")
        def open_Vpopup(self):
            validPopup1 = ValidPopUp()
            validPopup1.open()
            validPopup1.addAlarmText()

        def open_InVpopup(self):
            invalidPopup1 = InValidPopUp()
            invalidPopup1.open()
            invalidPopup1.addAlarmText()

        if((re.search(pattern1, atime)) or (re.search(pattern2, atime)) and (ampm ==
                                "AM" or ampm == "PM")):
            open_Vpopup(self)

        else:
            open_InVpopup(self)

    def setAMPM(self, AMorPM):
        global ampm
        ampm = str(AMorPM)

    def timeInput(self, timeCharacter):

        timestr = self.ids.timeInput.text
        timelength = len(timestr)
        if(timelength <= 4):
            previous = self.ids.timeInput.text
            self.ids.timeInput.text = f'{previous}{timeCharacter}'

    def backspace(self):

        def convert(s):
            str1 = ""

            return(str1.join(s))
        ti = self.ids.timeInput.text
        charint =  len(ti)
        if(charint >= 1):
            new = list(ti)
            new[-1] = ''
            self.ids.timeInput.text = convert(new)
        else:
            pass

    def turnAlarmOff(self):

        global atime
        global lightBool
        lightBool = False
        pygame.mixer.music.stop()
        atime = "0"
        pixels.fill((0,0,0,0))
        pixels.show()

    def updateTime2(self, *args):
        pattern1 = re.compile("^[1][0-2][:][0-5]\d")
        pattern2 = re.compile("^[1-9][:][0-5]\d")

        now = datetime.datetime.now()
        self.ids.timeLabel.text = now.strftime('%-I:%M')
        self.ids.ampmLabel.text = now.strftime('%p')

        if(re.search(pattern1, now.strftime('%-I:%M'))):
            self.ids.ampmLabel.pos_hint = {'x' : .435, 'top' : .93}

        if(re.search(pattern2, now.strftime('%-I:%M'))):
            self.ids.ampmLabel.pos_hint = {'x' : .4, 'top' : .93}



    def on_kv_post(self, basewidget):
        Clock.schedule_once(self.updateTime2)
        Clock.schedule_interval(self.updateTime2, 29)


class AlarmSounds(Screen):

    global currentAlarm

    def loadAlarm(self, Sound):
        global currentAlarm
        pygame.mixer.music.load(Sound)
        currentAlarm = Sound

    def playAlarm(self):
        if(pygame.mixer.music.get_busy() == True):
            self.ids.play.state = 'down'
            pygame.mixer.music.play(loops = -1)
        else:
            pygame.mixer.music.play(loops = -1)
            self.ids.play.state = 'down'

    def stopAlarm(self):
        pygame.mixer.music.stop()

    def updateTime3(self, *args):

        pattern1 = re.compile("^[1][0-2][:][0-5]\d")
        pattern2 = re.compile("^[1-9][:][0-5]\d")

        now = datetime.datetime.now()
        self.ids.timeLabel.text = now.strftime('%-I:%M')
        self.ids.ampmLabel.text = now.strftime('%p')

        if(re.search(pattern1, now.strftime('%-I:%M'))):
            self.ids.ampmLabel.pos_hint = {'x' : .68, 'top' : .93}

        if(re.search(pattern2, now.strftime('%-I:%M'))):
            self.ids.ampmLabel.pos_hint = {'x' : .62, 'top' : .93}

    def on_kv_post(self, basewidget):
        Clock.schedule_once(self.updateTime3)
        Clock.schedule_interval(self.updateTime3, 29)
#
class WhiteNoise(Screen):



    def LoadSound(self, Sound):
        global currentAlarm
        pygame.mixer.music.load(Sound)
        currentAlarm = Sound
```

```
def PlaySound(self):
    if(pygame.mixer.music.get_busy() == True):
        self.ids.play2.state = 'down'
        pygame.mixer.music.play(loops = -1)
    else:
        pygame.mixer.music.play(loops = -1)
        self.ids.play2.state = 'down'

def stop(self):
    pygame.mixer.music.pause()

def pause(self):
    pygame.mixer.init()

    if(pygame.mixer.music.get_busy() == True):
        pygame.mixer.music.pause()
        self.ids.pause.state = 'down'

    else:
        pygame.mixer.music.unpause()


def updateTime4(self, *args):

    pattern1 = re.compile("^[1][0-2][:][0-5]\d")
    pattern2 = re.compile("^[1-9][:][0-5]\d")

    now = datetime.datetime.now()
    self.ids.timeLabel.text = now.strftime('%-I:%M')
    self.ids.ampmLabel.text = now.strftime('%p')

    if(re.search(pattern1, now.strftime('%-I:%M'))):
        self.ids.ampmLabel.pos_hint = {'x' : .67, 'top' : .93}

    if(re.search(pattern2, now.strftime('%-I:%M'))):
        self.ids.ampmLabel.pos_hint = {'x' : .62, 'top' : .93}

def on_kv_post(self, basewidget):
    Clock.schedule_once(self.updateTime4)
    Clock.schedule_interval(self.updateTime4, 29)
```

```
class ValidPopUp(Popup):

    def addAlarmText(self):
        self.ids.alarmlabel.text = "Alarm set for " + atime + " " + ampm

class InValidPopUp(Popup):

    def addAlarmText(self):
        self.ids.alarmlabel.text = "Invalid Alarm: Enter a Valid Time..."

class AlarmPopUp(Popup):

    global aBool
    aBool = False
    global lightBool
    lightBool = False

    def turnAlarmOff(self):

        global atime
        global aBool
        aBool = False
        pygame.mixer.music.stop()
        atime = "0"
        global lightBool
        lightBool = False
        app = App.get_running_app()
        app.root.get_screen('main').ids.alarm1.source = '/home/pi/Desktop/Final
GUI/icons/alarm.png'

    def turnAlarmandLightsOff(self):
        global atime
        pygame.mixer.music.stop()
        global lightBool
        lightBool = False
        global aBool
        aBool = False
        atime = "0"
        pixels.fill((0,0,0,0))
        pixels.show()
        app = App.get_running_app()
        app.root.get_screen('main').ids.alarm1.source = '/home/pi/Desktop/Final
GUI/icons/alarm.png'

class LWCPopUp(Popup):

    def timedRelay(self, time):

        def relayOn(self):
            GPIO.output(LWC, GPIO.LOW)

        def relayOff(self):
            GPIO.output(LWC, GPIO.HIGH)
            app = App.get_running_app()
            app.root.get_screen('main').ids.leftWC.state = 'normal'

        Clock.schedule_once(relayOn)
        Clock.schedule_once(relayOff, time)

    def relayOn(self):
        GPIO.output(LWC, GPIO.LOW)

class RWCPopUp(Popup):

    def timedRelay(self, time):

        def relayOn(self):
            GPIO.output(RWC, GPIO.LOW)

        def relayOff(self):
            GPIO.output(RWC, GPIO.HIGH)
            app = App.get_running_app()
            app.root.get_screen('main').ids.rightWC.state = 'normal'

        Clock.schedule_once(relayOn)
        Clock.schedule_once(relayOff, time)

    def relayOn(self):
        GPIO.output(RWC, GPIO.LOW)

class rainbowAnimationPU(Popup):
    def turnLightsOFF(self, x):
        time.sleep(12)
        pixels.fill((0,0,0,0))
        pixels.show()
        self.ids.cd.text = "Ready to Close"

    def on_kv_post(self, basewidget):
        Clock.schedule_once(self.turnLightsOFF)

# Initilizaiton of Builder to read KV file
presentation = Builder.load_file("final_master.kv")

#Initilizing App for foundation of program
class finalMasterApp(App):
    def build(self):
        ScreenManagement = ScreenManager()
        ScreenManagement.add_widget(MainScreen(name = 'main'))
        ScreenManagement.add_widget(AdvancedColor(name = 'advancedcolor'))
        ScreenManagement.add_widget(LEDColor(name = 'customcolor'))
        ScreenManagement.add_widget(LEDAnimations1(name = 'animations'))
        ScreenManagement.add_widget(LEDAnimations2(name = 'animations2'))
        ScreenManagement.add_widget(AlarmScreen(name = 'alarmScreen'))
        ScreenManagement.add_widget(AlarmSounds(name = 'alarmSound'))
        ScreenManagement.add_widget(WhiteNoise(name = 'whitenoise'))
        return ScreenManagement

if __name__== "__main__":
    finalMasterApp().run()
```

Appendix E: Main Kivy Program

```
#:import Factory kivy.factory.Factory
#Initilizing Main Screen
<MainScreen>:
    name: "main"
    #Utilizing Float Layout format
    FloatLayout:

        Label:
            id:timeLabel
            color: (10/255.0,0,150/255.0,1)
            font_name: "Centaur"
            font_size: 175
            size_hint: (.6,.3)
            pos_hint: {"x": .13, "top": 1}

        Label:
            id:ampmLabel
            color: (10/255.0,0,150/255.0,1)
            font_name: "Centaur"
            font_size: 88
            size_hint: (.2,.15)
            pos_hint: {"x": .65, "top": .88}

        Label:
            id: dateLabel
            font_size: 35
            font_name: "Centaur"
            color: (10/255.0,0,150/255.0,1)
            size_hint: (.2,.5)
            pos_hint: {"x": .4, "top": .93}

        Label:
            id: LFan
            text: "0%"
            color: (10/255.0,0,90/255.0,1)
            font_size: 30
            size_hint: (.2,.2)
            pos_hint: {"x": .0, "top": 1}
```

```
Label:
    id: RFan
    text: "0 %"
    halign: 'right'
    font_size: 30
    color: (10/255.0,0,90/255.0,1)
    size_hint: (.2,.2)
    pos_hint: {"x": .82, "top": 1}

RoundedButton2:
    id: LEDLights
    font_size: 15
    pos_hint: {"x": .33, "top": .475}
    size_hint: (.34, .2)
    on_press: app.root.current = "advancedcolor"

    Image:

        source: '/home/pi/Desktop/Final GUI/icons/LED.png'
        color: (1,0,0,1)
        center_x: self.parent.center_x
        center_y: self.parent.center_y

RoundedTButton:
    id: leftLamp
    font_size: 15
    pos_hint: {"x": .2, "top": .575}
    size_hint: (.14, .11)
    on_press:
        if self.state == "normal": \
        root.ROn(22)
        if self.state == "down": \
        root.ROff(22)
    on_state:
        if self.state == "normal": \
        leftLamp1.source = '/home/pi/Desktop/Final GUI/icons/lamp.png'
        if self.state == "down": \
        leftLamp1.source = '/home/pi/Desktop/Final GUI/activated icons/lamp.png'


    Image:
        id: leftLamp1
        source: '/home/pi/Desktop/Final GUI/icons/lamp.png'
        center_x: self.parent.center_x
        center_y: self.parent.center_y

RoundedTButton:
    id: rightLamp
    font_size: 15
    pos_hint: {"x": .67, "top": .575}
    size_hint: (.13, .11)
    on_press:
        if self.state == "normal": \
        root.ROn(27)
        if self.state == "down": \
        root.ROff(27)

    on_state:
        if self.state == "normal": \
        rightLamp1.source = '/home/pi/Desktop/Final GUI/icons/lamp.png'
        if self.state == "down": \
        rightLamp1.source = '/home/pi/Desktop/Final GUI/activated icons/lamp.png'


    Image:
        id: rightLamp1
        source: '/home/pi/Desktop/Final GUI/icons/lamp.png'
        center_x: self.parent.center_x
        center_y: self.parent.center_y

RoundedTButton:
    id: leftWC
    font_size: 15
    pos_hint: {"x": .21, "top": .315}
    size_hint: (.13, .11)
    on_press:
        pu = Factory.LWCPopUp()
        if self.state == "normal": \
        root.ROn(17)
        if self.state == "down": \
        pu.open()
    on_state:
        if self.state == "normal": \
        leftWC1.source = '/home/pi/Desktop/Final GUI/icons/WirelessCharging.png'
        if self.state == "down": \
        leftWC1.source = '/home/pi/Desktop/Final GUI/activated
icons/WirelessCharging.png'

    Image:
        id: leftWC1
        source: '/home/pi/Desktop/Final GUI/icons/WirelessCharging.png'
        center_x: self.parent.center_x
        center_y: self.parent.center_y


RoundedTButton:
    id: rightWC
    font_size: 15
    pos_hint: {"x": .67, "top": .315}
    size_hint: (.13, .11)
    on_press:
        pu = Factory.RWCPopUp()
        if self.state == "normal": \
        root.ROn(4)
        if self.state == "down": \
        pu.open()

    on_state:
        if self.state == "normal": \
        rightWC1.source = '/home/pi/Desktop/Final GUI/icons/WirelessCharging.png'
        if self.state == "down": \
        rightWC1.source = '/home/pi/Desktop/Final GUI/activated
icons/WirelessCharging.png'

    Image:
        id: rightWC1
        source: '/home/pi/Desktop/Final GUI/icons/WirelessCharging.png'
        center_x: self.parent.center_x
        center_y: self.parent.center_y

RoundedButton:
    id: whiteNoise
    font_size: 20
    pos_hint: {"x": .439, "top": .63}
    color: (0,0,0,1)
    size_hint: (.13, .11)
    on_press:
        app.root.current = "whitenoise"


    Image:
        source: '/home/pi/Desktop/Final GUI/icons/whitenoise3.png'
        center_x: self.parent.center_x
        center_y: self.parent.center_y

RoundedTButton:
    id: powerbtn
    font_size: 15
    pos_hint: {"x": .53, "top": .205}
    size_hint: (.13, .11)
    on_press:
        root.powerOff()
        app.stop()

    Image:
        source: '/home/pi/Desktop/Final GUI/icons/power.png'
        center_x: self.parent.center_x
        center_y: self.parent.center_y

RoundedButton:
    id: Alarm
    font_size: 15
    pos_hint: {"x": .34, "top": .205}
    size_hint: (.13, .11)
    on_press: app.root.current = "alarmScreen"

    Image:
        id: alarm1
        source: '/home/pi/Desktop/Final GUI/icons/alarm.png'
        center_x: self.parent.center_x
        center_y: self.parent.center_y

RoundedTButton:
    id: leftFan
    font_size: 15
    pos_hint: {"x": .0245, "top": .15}
    size_hint: (.13, .11)
    on_press: root.LFanON(*args)
    on_state:
        if self.state == "normal": \
        leftFan1.source = '/home/pi/Desktop/Final GUI/icons/cpufan1.png'
        if self.state == "down": \
        leftFan1.source = '/home/pi/Desktop/Final GUI/activated icons/cpufan1.png'

    Image:
        id: leftFan1
        source: '/home/pi/Desktop/Final GUI/icons/cpufan1.png'
        center_x: self.parent.center_x
        center_y: self.parent.center_y

RoundedTButton:
    id: rightFan
    font_size: 15
    pos_hint: {"x": .848, "top": .15}
    size_hint: (.13, .11)
    on_press: root.RFanON(*args)
    on_state:
        if self.state == "normal": \
        rightFan1.source = '/home/pi/Desktop/Final GUI/icons/cpufan1.png'
        if self.state == "down": \
        rightFan1.source = '/home/pi/Desktop/Final GUI/activated icons/cpufan1.png'

    Image:
        id: rightFan1
        source: '/home/pi/Desktop/Final GUI/icons/cpufan1.png'
        center_x: self.parent.center_x
        center_y: self.parent.center_y



Slider:
    id: leftSlider
    min:0
    font_size: 20
    size_hint: (.1, .6)
    orientation: 'vertical'
    pos_hint: {"x": .04, "top": .8}
    on_value: LFan.text = str(int(leftSlider.value)) + "%"
    on_value: root.LFanCS(*args)


Slider:
    id: rightSlider
    font_size: 20
    size_hint: (.1, .6)
    orientation: 'vertical'
    pos_hint: {"x": .86, "top": .79}
    on_value: RFan.text =  str(int(rightSlider.value)) + "%"
    on_value: root.RFanCS(*args)

<AdvancedColor>:
name: "advancedcolor"

FloatLayout:

RoundedButton2:
    id: customcolor1
    text: "Custom Color"
    font_size: 30
    size_hint: (.4,.25)
    pos_hint: {"x": .51, "top": .875}
    color: (10/255.0,0,150/255.0,1)
    font_name: "Centaur"
    on_press: app.root.current = "customcolor"

RoundedButton2:
    id: animations
    text: "Animations"
    size_hint: (.4,.25)
    font_size: 30
    pos_hint: {"x": .51, "top": .55}
    color: (10/255.0,0,150/255.0,1)
    font_name: "Centaur"
    on_press: app.root.current = "animations"

RoundedButton:
```

```
        id: clear
        text: "Clear"
        pos_hint: {"x": .5, "top": .19}
        size_hint: (.2, .15)
        font_name: "Centaur"
        font_size: 20
        color: (10/255.0,0,150/255.0,1)
        on_press: root.clearLights()

    RoundedButton:
        id: return
        pos_hint: {"x": .75, "top": .19}
        size_hint: (.2, .15)
        font_name: "Centaur"
        color: (10/255.0,0,150/255.0,1)
        on_press: app.root.current = "main"

        Image:

            source: '/home/pi/Desktop/Final GUI/icons/return.png'
            center_x: self.parent.center_x
            center_y: self.parent.center_y

    Label:
        id: white
        text: "W"
        font_size: 45
        color: (10/255.0,0,150/255.0,1)
        size_hint: (.2, .1)
        pos_hint: {"x": .01, "top": .95}

    Label:
        id: tune
        text: "T"
        font_size: 45
        color: (10/255.0,0,150/255.0,1)
        size_hint: (.2, .1)
        pos_hint: {"x": .2385, "top": .95}

    Slider:
        id: whiteSlider
```

```
            font_size: 20
            min: 0
            max: 255
            size_hint: (.1, .75)
            orientation: 'vertical'
            pos_hint: {"x": .06, "top": .855}
            on_value: root.addWhite(*args)

        Slider:
            id: tunableSlider
            font_size: 20
            min: 0
            max: 255
            size_hint: (.1, .75)
            orientation: 'vertical'
            pos_hint: {"x": .29, "top": .855}
            on_value: root.tuner(*args)

<LEDAnimations1>:
    name: "animations"

    FloatLayout:

        Label:
            id:timeLabel
            color: (10/255.0,0,150/255.0,1)
            font_name: "Centaur"
            font_size: 135
            size_hint: (.4,.2)
            pos_hint: {"x": .45, "top": .98}

        Label:
            id:ampmLabel
            color: (10/255.0,0,150/255.0,1)
            font_name: "Centaur"
            font_size: 75
            size_hint: (.1,.2)
            pos_hint: {"x": .88, "top": .9}
            text: "PM"

        RoundedTButton:
```

```
        id: blink
        group: "animations"
        text: "Blink"
        color: (0,0,0,1)
        size_hint: (.2, .15)
        pos_hint: {"x": .52 ,"top": .75}
        font_size: 25
        font_name: "Centaur"
        color: (0,0,1,1)
        on_state:
            if self.state == "down": \
            blink.color = (1,0,0,1)
            if self.state == "normal": \
            blink.color = (0,0,1,1)

        on_press:
            play.state = "normal"
        on_press:
            if self.state == "down": \
            root.setAnimationState(1)

    RoundedTButton:
        id: comet
        group: "animations"
        text: "Comet"
        color: (0,0,0,1)
        size_hint: (.2, .15)
        pos_hint: {"x": .76 ,"top": .75}
        font_size: 25
        font_name: "Centaur"
        color: (0,0,1,1)
        on_state:
            if self.state == "down": \
            comet.color = (1,0,0,1)
            else: \
            comet.color = (0,0,1,1)
        on_press:
            play.state = "normal"
            if self.state == "down": \
            root.setAnimationState(2)
```

```
RoundedTButton:
    id: chase
    group: "animations"
    text: "Chase"
    color: (0,0,0,1)
    size_hint: (.2, .15)
    pos_hint: {"x": .52 ,"top": .55}
    font_size: 25
    font_name: "Centaur"
    color: (0,0,1,1)
    on_state:
        if self.state == "down": \
        chase.color = (1,0,0,1)
        else: \
        chase.color = (0,0,1,1)
    on_press:
        play.state = "normal"
        if self.state == "down": \
        root.setAnimationState(3)

RoundedTButton:
    id: sparkle
    group: "animations"
    text: "Twinkle"
    color: (0,0,0,1)
    size_hint: (.2, .15)
    pos_hint: {"x": .76 ,"top": .55}
    font_size: 25
    font_name: "Centaur"
    color: (0,0,1,1)
    on_state:
        if self.state == "down": \
        sparkle.color = (1,0,0,1)
        else: \
        sparkle.color = (0,0,1,1)
    on_press:
        play.state = "normal"
        if self.state == "down": \
        root.setAnimationState(5)
```

```
RoundedTButton:
    id: pulse
    group: "animations"
    text: "Pulse"
    color: (0,0,0,1)
    size_hint: (.2, .15)
    pos_hint: {"x": .52 ,"top": .35}
    font_size: 25
    font_name: "Centaur"
    color: (0,0,1,1)
    on_state:
        if self.state == "down": \
        pulse.color = (1,0,0,1)
        else: \
        pulse.color = (0,0,1,1)
    on_press:
        play.state = "normal"
        if self.state == "down": \
        root.setAnimationState(4)

RoundedTButton:
    id: sparklepulse
    group: "animations"
    text: "Sparkle"
    color: (0,0,0,1)
    size_hint: (.2, .15)
    pos_hint: {"x": .76 ,"top": .35}
    font_size: 25
    font_name: "Centaur"
    color: (0,0,1,1)
    on_state:
        if self.state == "down": \
        sparklepulse.color = (1,0,0,1)
        else: \
        sparklepulse.color = (0,0,1,1)
    on_press:
        play.state = "normal"
        if self.state == "down": \
        root.setAnimationState(6)

Label:
    id: colorlabel
    text: "Color"
    font_size: 30
    size_hint: (.2, .1)
    pos_hint: {"x": .035, "top": 1}

Label:
    id: colorValLabel
    text: "N/A"
    font_size: 30
    size_hint: (.2, .1)
    pos_hint: {"x": .035, "top": .275}

Slider:
    id: colorSlider
    font_size: 15
    min: 1
    max: 20
    size_hint: (.1, .6)
    orientation: 'vertical'
    pos_hint: {"x": .08, "top": .9}
    on_value: root.setColor(*args)


Label:
    id: sizelabel
    text: "Size"
    font_size: 30
    size_hint: (.2, .1)
    pos_hint: {"x": .28, "top": 1}

Label:
    id: sizevallabel
    text: "N/A"
    font_size: 30
    size_hint: (.2, .1)
    pos_hint: {"x": .285, "top": .275}

Slider:
    id: sizeSlider
    font_size: 20
    min: 1
    max: 20
    size_hint: (.1, .6)
    orientation: 'vertical'
    pos_hint: {"x": .33, "top": .9}
    on_value: root.setSize(*args)



RoundedTButton:
    id: play
    color: (0,0,0,1)
    size_hint: (.2, .15)
    pos_hint: {"x": .04 ,"top": .15}
    font_size: 25
    font_name: "Centaur"
    color: (0,0,1,1)
    on_state:
        if self.state == 'normal': \
        play1.source = '/home/pi/Desktop/Final GUI/icons/solidplay.png'
        if self.state == 'down': \
        play1.source = '/home/pi/Desktop/Final GUI/activated icons/solidplay.png'
    on_press:
        root.startThread()

    Image:
        id: play1
        source: '/home/pi/Desktop/Final GUI/icons/solidplay.png'
        center_x: self.parent.center_x
        center_y: self.parent.center_y


RoundedButton:
    id: stop
    color: (0,0,0,1)
    size_hint: (.2, .15)
    pos_hint: {"x": .28 ,"top": .15}
    font_size: 25
    font_name: "Centaur"
    color: (0,0,1,1)
    on_press:
            play.state = "normal"
            root.stopThread()

        Image:
            source: '/home/pi/Desktop/Final GUI/icons/solidstop.png'
            center_x: self.parent.center_x
            center_y: self.parent.center_y

    RoundedButton:
        id: return2
        color: (0,0,0,1)
        size_hint: (.2, .15)
        pos_hint: {"x": .52 ,"top": .15}
        font_size: 25
        font_name: "Centaur"
        color: (0,0,1,1)
        on_press: app.root.current = "advancedcolor"

        Image:
            source: '/home/pi/Desktop/Final GUI/icons/return.png'
            center_x: self.parent.center_x
            center_y: self.parent.center_y


    RoundedButton:
        id: page1
        color: (0,0,0,1)
        text: "Rainbow"
        size_hint: (.2, .15)
        pos_hint: {"x": .76 ,"top": .15}
        font_size: 20
        font_name: "Centaur"
        color: (0,0,1,1)
        on_press: app.root.current = "animations2"

<LEDAnimations2>:
name: "animations2"

FloatLayout:

    Label:
```

```
        id:timeLabel
        color: (10/255.0,0,150/255.0,1)
        font_name: "Centaur"
        font_size: 135
        size_hint: (.4,.2)
        pos_hint: {"x": .45, "top": .98}

    Label:
        id:ampmLabel
        color: (10/255.0,0,150/255.0,1)
        font_name: "Centaur"
        font_size: 75
        size_hint: (.1,.2)
        pos_hint: {"x": .88, "top": .9}
        text: "PM"

    RoundedTButton:
        id: rainbow1
        group: "animations"
        text: "Rainbow 1"
        color: (0,0,0,1)
        size_hint: (.2, .15)
        pos_hint: {"x": .52 ,"top": .75}
        font_size: 25
        font_name: "Centaur"
        color: (0,0,1,1)
        on_state:
            if self.state == "down": \
            rainbow1.color = (1,0,0,1)
            if self.state == "normal": \
            rainbow1.color = (0,0,1,1)

        on_press:
            play.state = "normal"
        on_press:
            if self.state == "down": \
            root.setAnimationState(1)

    RoundedTButton:
        id: rainbow2
        group: "animations"
        text: "Rainbow 2"
        color: (0,0,0,1)
        size_hint: (.2, .15)
        pos_hint: {"x": .76 ,"top": .75}
        font_size: 25
        font_name: "Centaur"
        color: (0,0,1,1)
        on_state:
            if self.state == "down": \
            rainbow2.color = (1,0,0,1)
            else: \
            rainbow2.color = (0,0,1,1)
        on_press:
            play.state = "normal"
            if self.state == "down": \
            root.setAnimationState(2)


    RoundedTButton:
        id: rainbowchase
        group: "animations"
        text: "Rainbow Chase"
        color: (0,0,0,1)
        size_hint: (.2, .15)
        pos_hint: {"x": .52 ,"top": .55}
        font_size: 25
        font_name: "Centaur"
        color: (0,0,1,1)
        on_state:
            if self.state == "down": \
            rainbowchase.color = (1,0,0,1)
            else: \
            rainbowchase.color = (0,0,1,1)
        on_press:
            play.state = "normal"
            if self.state == "down": \
            root.setAnimationState(3)

    RoundedTButton:
        id: rainbowcomet
        group: "animations"
        text: "Rainbow Comet"
        color: (0,0,0,1)
        size_hint: (.2, .15)
        pos_hint: {"x": .76 ,"top": .55}
        font_size: 25
        font_name: "Centaur"
        color: (0,0,1,1)
        on_state:
            if self.state == "down": \
            rainbowcomet.color = (1,0,0,1)
            else: \
            rainbowcomet.color = (0,0,1,1)
        on_press:
            play.state = "normal"
            if self.state == "down": \
            root.setAnimationState(4)

    RoundedTButton:
        id: rainbowsparkle
        group: "animations"
        text: "Rainbow Sparkle"
        color: (0,0,0,1)
        size_hint: (.2, .15)
        pos_hint: {"x": .52 ,"top": .35}
        font_size: 25
        font_name: "Centaur"
        color: (0,0,1,1)
        on_state:
            if self.state == "down": \
            rainbowsparkle.color = (1,0,0,1)
            else: \
            rainbowsparkle.color = (0,0,1,1)
        on_press:
            play.state = "normal"
            if self.state == "down": \
            root.setAnimationState(5)

    RoundedTButton:
        id: colorcycle
        group: "animations"
        text: "Color Cycle"
        color: (0,0,0,1)
        size_hint: (.2, .15)
        pos_hint: {"x": .76 ,"top": .35}
        font_size: 25
        font_name: "Centaur"
        color: (0,0,1,1)
        on_state:
            if self.state == "down": \
            colorcycle.color = (1,0,0,1)
            else: \
            colorcycle.color = (0,0,1,1)
        on_press:
            play.state = "normal"
            if self.state == "down": \
            root.setAnimationState(6)

    Label:
        id: speedlabel
        text: "Speed"
        font_size: 30
        size_hint: (.2, .1)
        pos_hint: {"x": .035, "top": 1}

    Label:
        id: speedvallabel
        text: "N/A"
        font_size: 30
        size_hint: (.2, .1)
        pos_hint: {"x": .035, "top": .275}

    Slider:
        id: speedSlider
        font_size: 20
        min: 0
        max: 15
        size_hint: (.1, .6)
        orientation: 'vertical'
        pos_hint: {"x": .08, "top": .9}
        on_value: root.setSpeed(*args)

    Label:
        id: sizelabel
        text: "Size"
        font_size: 30
        size_hint: (.2, .1)
        pos_hint: {"x": .28, "top": 1}

    Label:
        id: sizevallabel
        text: "N/A"
        font_size: 30
        size_hint: (.2, .1)
        pos_hint: {"x": .28, "top": .275}

    Slider:
        id: sizeSlider
        font_size: 20
        min: 1
        max: 20
        size_hint: (.1, .6)
        orientation: 'vertical'
        pos_hint: {"x": .325, "top": .9}
        on_value: root.setSize(*args)



    RoundedTButton:
        id: play
        color: (0,0,0,1)
        size_hint: (.2, .15)
        pos_hint: {"x": .04 ,"top": .15}
        font_size: 25
        font_name: "Centaur"
        color: (0,0,1,1)
        on_state:
            if self.state == 'normal': \
            play1.source = '/home/pi/Desktop/Final GUI/icons/solidplay.png'
            if self.state == 'down': \
            play1.source = '/home/pi/Desktop/Final GUI/activated icons/solidplay.png'
        on_press:
            root.startThread()
        Image:
            id: play1
            source: '/home/pi/Desktop/Final GUI/icons/solidplay.png'
            center_x: self.parent.center_x
            center_y: self.parent.center_y


    RoundedButton:
        id: stop
        color: (0,0,0,1)
        size_hint: (.2, .15)
        pos_hint: {"x": .28 ,"top": .15}
        font_size: 25
        font_name: "Centaur"
        color: (0,0,1,1)
        on_press:
            play.state = "normal"
            root.stopThread()

        Image:
            source: '/home/pi/Desktop/Final GUI/icons/solidstop.png'
            center_x: self.parent.center_x
            center_y: self.parent.center_y

    RoundedButton:
        id: return2
        color: (0,0,0,1)
        size_hint: (.2, .15)
        pos_hint: {"x": .52 ,"top": .15}
        font_size: 25
        font_name: "Centaur"
        color: (0,0,1,1)
        on_press: app.root.current = "main"

        Image:
            source: '/home/pi/Desktop/Final GUI/icons/return.png'
            center_x: self.parent.center_x
            center_y: self.parent.center_y


    RoundedButton:
            id: page1
            color: (0,0,0,1)
            text: "Solid Color"
            size_hint: (.2, .15)
            pos_hint: {"x": .76 ,"top": .15}
            font_size: 20
            font_name: "Centaur"
            color: (0,0,1,1)
            on_press: app.root.current = "animations"


<LEDColor>:
    name: "ccolor"

    FloatLayout:

        ColorPicker:
            id: colorpicker

        RoundedButton:
            id: clear
            text: "Clear"
            font_size: 17.5
            color: (0,0,1,1)
            size_hint: (.15,.15)
            font_size: 20
            font_name: "Centaur"
            pos_hint: {"x": .84, "top": .97}
            on_press: root.clearLights()


        RoundedButton:
            id:set
            text: "Set Color"
            color: (0,0,1,1)
            font_size: 17.5
            size_hint: (.15,.15)
            font_size: 20
            font_name: "Centaur"
            pos_hint: {"x": .51, "top": .97}
            on_press: root.setColor()
        RoundedButton:
            id: return1
            size_hint: (.15,.15)
            pos_hint: {"x": .85, "top": .17}
            font_size: 25
            font_name: "Centaur"
            on_press: app.root.current = "advancedcolor"

            Image:

                source: '/home/pi/Desktop/Final GUI/icons/return.png'
                center_x: self.parent.center_x
                center_y: self.parent.center_y


<AlarmScreen>:

    name: "alarmScreen"

    inp1:  timeInput

    FloatLayout:

        Label:
            id: timeLabel
            font_size: 150
            text: ""
            font_name: "Centaur"
            color: (0,0,1,1)
            size_hint: (.45,.25)
            pos_hint: {"top": .985}

        Label:
            id: ampmLabel
            font_size: 50
            text: ""
            font_name: "Centaur"
            color: (0,0,1,1)
            size_hint: (.05,.25)
            pos_hint: {"x": .44 ,"top": .93}
```

```
Label:
    id: alabel
    size_hint: (.15, .1)
    text: "Alarm:"
    font_name: "Centaur"
    font_size: 35
    pos_hint: {"top": .69}
    color: (0,0,1,1)


TextInput:
    id: timeInput
    font_size: 25
    size_hint: (.25,.12)
    pos_hint: {"x": .16, "top": .69}
    font_size: 40
    font_name: "Centaur"
    color: (0,0,1,1)

Label:
    id: am
    text: "AM"
    size_hint: (.1,.15)
    pos_hint: {"x": .05 ,"top": .54}
    font_size: 35
    font_name: "Centaur"
    color: (0,0,1,1)

CheckBox:
    id: AM
    group: "AMorPM"
    pos_hint: {"x": .13 ,"top": .52}
    size_hint: (.1,.1)
    on_active: root.setAMPM("AM")


Label:
    id: pm
    text: "PM"
    size_hint: (.1,.15)
```

```
        pos_hint: {"x": .27 ,"top": .54}
        font_size: 35
        font_name: "Centaur"
        color: (0,0,1,1)

    CheckBox:
        id: PM
        group: "AMorPM"
        pos_hint: {"x": .35 ,"top": .52}
        size_hint: (.1,.1)
        on_active: root.setAMPM("PM")

    RoundedButton2:
        id: alarm
        pos_hint: {"x": .133, "top": .41}
        size_hint: (.25, .18)
        color: (0,0,1,1)
        font_size: 20
        on_press: app.root.current = "alarmSound"

        Image:

            source: '/home/pi/Desktop/Final GUI/icons/alarmnote.png'
            center_x: self.parent.center_x
            center_y: self.parent.center_y

    RoundedButton:
        id: setAlarm
        text: ("Set")
        size_hint: (.17, .15)
        pos_hint: {"x":.05, "top":.22}
        font_size: 30
        font_name: "Centaur"
        color: (0,0,1,1)
        on_press: root.setAlarm()

    RoundedButton:
        id: cancelAlarm
        color: (0,0,0,1)
        text: "Cancel"
        size_hint: (.17, .15)
```

```
            pos_hint: {"x": .29 ,"top": .22}
            font_size: 30
            font_name: "Centaur"
            color: (0,0,1,1)
            on_press:
                root.turnAlarmOff()
                app.root.current = "main"

        RoundedButton:
            id: one
            text: "1"
            color: (0,0,0,1)
            size_hint: (.15, .15)
            pos_hint: {"x": .51 ,"top": .93}
            font_size: 40
            font_name: "Centaur"
            color: (0,0,1,1)
            on_release: root.timeInput("1")


        RoundedButton:
            id: two
            text: "2"
            color: (0,0,0,1)
            size_hint: (.15, .15)
            pos_hint: {"x": .67, "top": .93}
            font_size: 40
            font_name: "Centaur"
            color: (0,0,1,1)
            on_press: root.timeInput("2")

        RoundedButton:
            id: three
            text: "3"
            color: (0,0,0,1)
            size_hint: (.15, .15)
            pos_hint: {"x": .83,"top": .93}
            font_size: 40
            font_name: "Centaur"
            color: (0,0,1,1)
            on_press: root.timeInput("3")
```

```
RoundedButton:
    id: four
    text: "4"
    color: (0,0,0,1)
    size_hint: (.15, .15)
    pos_hint: {"x": .51, "top": .69}
    font_size: 40
    font_name: "Centaur"
    color: (0,0,1,1)
    on_press: root.timeInput("4")

RoundedButton:
    id: five
    text: "5"
    color: (0,0,0,1)
    size_hint: (.15, .15)
    pos_hint: {"x": .67, "top": .69}
    font_size: 40
    font_name: "Centaur"
    color: (0,0,1,1)
    on_press: root.timeInput("5")

RoundedButton:
    id: six
    text: "6"
    color: (0,0,0,1)
    size_hint: (.15, .15)
    pos_hint: {"x": .83,"top": .69}
    font_size: 40
    font_name: "Centaur"
    color: (0,0,1,1)
    on_press: root.timeInput("6")

RoundedButton:
    id: seven
    text: "7"
    color: (0,0,0,1)
    size_hint: (.15, .15)
    pos_hint: {"x": .51,"top": .45}
    font_size: 40
```

```
        font_name: "Centaur"
        color: (0,0,1,1)
        on_press: root.timeInput("7")

    RoundedButton:
        id: eight
        text: "8"
        color: (0,0,0,1)
        size_hint: (.15, .15)
        pos_hint: {"x": .67 ,"top": .45}
        font_size: 40
        font_name: "Centaur"
        color: (0,0,1,1)
        on_press: root.timeInput("8")

    RoundedButton:
        id: nine
        text: "9"
        color: (0,0,0,1)
        size_hint: (.15, .15)
        pos_hint: {"x": .83, "top": .45}
        font_size: 40
        font_name: "Centaur"
        color: (0,0,1,1)
        on_press: root.timeInput("9")

    RoundedButton:
        id: backspace
        color: (0,0,0,1)
        size_hint: (.15, .15)
        pos_hint: {"x": .51, "top": .22}
        font_size: 40
        font_name: "Centaur"
        color: (0,0,1,1)
        on_press: root.backspace()

        Image:
            source: '/home/pi/Desktop/Final GUI/icons/backspace1.png'
            center_x: self.parent.center_x
            center_y: self.parent.center_y
        RoundedButton:
            id: zero
            text: "0"
            color: (0,0,0,1)
            size_hint: (.15, .15)
            pos_hint: {"x": .67,"top": .22}
            font_size: 40
            font_name: "Centaur"
            color: (0,0,1,1)
            on_press: root.timeInput("0")

        RoundedButton:
            id: colon
            text: ":"
            color: (0,0,0,1)
            size_hint: (.15, .15)
            pos_hint: {"x": .83,"top": .22}
            font_size: 40
            font_name: "Centaur"
            color: (0,0,1,1)
            on_press: root.timeInput(":")


<AlarmSounds>:

    name: "alarmSound"

    FloatLayout:

        Label:
            id: timeLabel
            font_size: 150
            text: ""
            font_name: "Centaur"
            color: (0,0,1,1)
            size_hint: (.6,.25)
            pos_hint: {"x": .15 ,"top": .985}

        Label:
            id: ampmLabel
            font_size: 50
            text: ""
            font_name: "Centaur"
            color: (0,0,1,1)
            size_hint: (.05,.25)
            pos_hint: {"x": .67 ,"top": .93}

        RoundedTButton:
            id: alarm1
            group: "alarms"
            text: "Bird Song"
            color: (0,0,0,1)
            size_hint: (.2, .15)
            pos_hint: {"x": .04 ,"top": .75}
            font_size: 25
            font_name: "Centaur"
            color: (0,0,1,1)
            on_state:
                if self.state == "down": \
                alarm1.color = (1,0,0,1)
                if self.state == "normal": \
                alarm1.color = (0,0,1,1)
            on_press:
                root.loadAlarm("Birdsong.mp3")
                play.state = "normal"

        RoundedTButton:
            id: alarm2
            group: "alarms"
            text: "Early Riser"
            color: (0,0,0,1)
            size_hint: (.2, .15)
            pos_hint: {"x": .28 ,"top": .75}
            font_size: 25
            font_name: "Centaur"
            color: (0,0,1,1)
            on_state:
                if self.state == "down": \
                alarm2.color = (1,0,0,1)
                else: \
                alarm2.color = (0,0,1,1)
            on_press:
```

```
            root.loadAlarm("Early Riser.mp3")
            play.state = "normal"

    RoundedTButton:
        id: alarm3
        group: "alarms"
        text: "Fighter Jets"
        color: (0,0,0,1)
        size_hint: (.2, .15)
        pos_hint: {"x": .52 ,"top": .75}
        font_size: 25
        font_name: "Centaur"
        color: (0,0,1,1)
        on_state:
            if self.state == "down": \
            alarm3.color = (1,0,0,1)
            else: \
            alarm3.color = (0,0,1,1)
        on_press:
            root.loadAlarm("Fighter Jets.mp3")
            play.state = "normal"

    RoundedTButton:
        id: alarm4
        group: "alarms"
        text: "First Light"
        color: (0,0,0,1)
        size_hint: (.2, .15)
        pos_hint: {"x": .76 ,"top": .75}
        font_size: 25
        font_name: "Centaur"
        color: (0,0,1,1)
        on_state:
            if self.state == "down": \
            alarm4.color = (1,0,0,1)
            else: \
            alarm4.color = (0,0,1,1)
        on_press:
            root.loadAlarm("First Light.mp3")
            play.state = "normal"
```

```
RoundedTButton:
    id: alarm5
    group: "alarms"
    text: "Helicopter"
    color: (0,0,0,1)
    size_hint: (.2, .15)
    pos_hint: {"x": .04 ,"top": .55}
    font_size: 25
    font_name: "Centaur"
    color: (0,0,1,1)
    on_state:
        if self.state == "down": \
        alarm5.color = (1,0,0,1)
        else: \
        alarm5.color = (0,0,1,1)
    on_press:
        root.loadAlarm("Helicopter.mp3")
        play.state = "normal"

RoundedTButton:
    id: alarm6
    group: "alarms"
    text: "Helios"
    color: (0,0,0,1)
    size_hint: (.2, .15)
    pos_hint: {"x": .28 ,"top": .55}
    font_size: 25
    font_name: "Centaur"
    color: (0,0,1,1)
    on_state:
        if self.state == "down": \
        alarm6.color = (1,0,0,1)
        else: \
        alarm6.color = (0,0,1,1)
    on_press:
        root.loadAlarm("Helios.mp3")
        play.state = "normal"

RoundedTButton:
    id: alarm7
    group: "alarms"
```

```
        text: "Inception"
        color: (0,0,0,1)
        size_hint: (.2, .15)
        pos_hint: {"x": .52 ,"top": .55}
        font_size: 25
        font_name: "Centaur"
        color: (0,0,1,1)
        on_state:
            if self.state == "down": \
            alarm7.color = (1,0,0,1)
            else: \
            alarm7.color = (0,0,1,1)
        on_press:
            root.loadAlarm("Inception.mp3")
            play.state = "normal"

    RoundedTButton:
        id: alarm8
        group: "alarms"
        text: "Nuclear"
        color: (0,0,0,1)
        size_hint: (.2, .15)
        pos_hint: {"x": .76 ,"top": .55}
        font_size: 25
        font_name: "Centaur"
        color: (0,0,1,1)
        on_state:
            if self.state == "down": \
            alarm8.color = (1,0,0,1)
            else: \
            alarm8.color = (0,0,1,1)
        on_press:
            root.loadAlarm("Nuclear.mp3")
            play.state = "normal"

    RoundedTButton:
        id: alarm9
        group: "alarms"
        text: "Rusty Lake"
        color: (0,0,0,1)
        size_hint: (.2, .15)
        pos_hint: {"x": .76 ,"top": .35}
        font_size: 25
        font_name: "Centaur"
        color: (0,0,1,1)
        on_state:
            if self.state == "down": \
            alarm9.color = (1,0,0,1)
            else: \
            alarm9.color = (0,0,1,1)
        on_press:
            root.loadAlarm("Rusty Lake.mp3")
            play.state = "normal"

    RoundedTButton:
        id: alarm10
        group: "alarms"
        text: "Springtide"
        color: (0,0,0,1)
        size_hint: (.2, .15)
        pos_hint: {"x": .04 ,"top": .35}
        font_size: 25
        font_name: "Centaur"
        color: (0,0,1,1)
        on_state:
            if self.state == "down": \
            alarm10.color = (1,0,0,1)
            else: \
            alarm10.color = (0,0,1,1)
        on_press:
            root.loadAlarm("Springtide.mp3")
            play.state = "normal"

    RoundedTButton:
        id: alarm11
        group: "alarms"
        text: "Submarine"
        color: (0,0,0,1)
        size_hint: (.2, .15)
        pos_hint: {"x": .28 ,"top": .35}
        font_size: 25
        font_name: "Centaur"
        color: (0,0,1,1)
        on_state:
            if self.state == "down": \
            alarm11.color = (1,0,0,1)
            else: \
            alarm11.color = (0,0,1,1)
        on_press:
            root.loadAlarm("Submarine.mp3")
            play.state = "normal"

    RoundedTButton:
        id: alarm12
        group: "alarms"
        text: "Sunny"
        color: (0,0,0,1)
        size_hint: (.2, .15)
        pos_hint: {"x": .52 ,"top": .35}
        font_size: 25
        font_name: "Centaur"
        color: (0,0,1,1)
        on_state:
            if self.state == "down": \
            alarm12.color = (1,0,0,1)
            else: \
            alarm12.color = (0,0,1,1)
        on_press:
            root.loadAlarm("Sunny.mp3")
            play.state = "normal"


    RoundedTButton:
        id: play
        color: (0,0,0,1)
        size_hint: (.2, .15)
        pos_hint: {"x": .1 ,"top": .15}
        font_size: 25
        font_name: "Centaur"
        color: (0,0,1,1)
        on_press: root.playAlarm()
        on_state:
            if self.state == 'normal': \
            play1.source = '/home/pi/Desktop/Final GUI/icons/solidplay.png'
            if self.state == 'down': \
            play1.source = '/home/pi/Desktop/Final GUI/activated icons/solidplay.png'

        Image:
            id: play1
            source: '/home/pi/Desktop/Final GUI/icons/solidplay.png'
            center_x: self.parent.center_x
            center_y: self.parent.center_y


    RoundedButton:
        id: stop
        color: (0,0,0,1)
        size_hint: (.2, .15)
        pos_hint: {"x": .4 ,"top": .15}
        font_size: 25
        font_name: "Centaur"
        color: (0,0,1,1)
        on_press:
            root.stopAlarm()
            play.state = "normal"

        Image:
            source: '/home/pi/Desktop/Final GUI/icons/solidstop.png'
            center_x: self.parent.center_x
            center_y: self.parent.center_y

    RoundedButton:
        id: return2
        color: (0,0,0,1)
        size_hint: (.2, .15)
        pos_hint: {"x": .7 ,"top": .15}
        font_size: 25
        font_name: "Centaur"
        color: (0,0,1,1)
        on_press: app.root.current = "alarmScreen"

        Image:
            source: '/home/pi/Desktop/Final GUI/icons/return.png'
            center_x: self.parent.center_x
            center_y: self.parent.center_y

<WhiteNoise>:
    name: "whitenoise"

    FloatLayout:

        Label:
            id: timeLabel
            font_size: 150
            text: ""
            font_name: "Centaur"
            color: (0,0,1,1)
            size_hint: (.6,.25)
            pos_hint: {"x": .15 ,"top": .985}

        Label:
            id: ampmLabel
            font_size: 50
            text: ""
            font_name: "Centaur"
            color: (0,0,1,1)
            size_hint: (.05,.25)
            pos_hint: {"x": .67 ,"top": .93}

        RoundedTButton:
            id: WN1
            group: "white noise"
            text: "White"
            color: (0,0,0,1)
            size_hint: (.2, .15)
            pos_hint: {"x": .04 ,"top": .75}
            font_size: 25
            font_name: "Centaur"
            color: (0,0,1,1)
            on_state:
                if self.state == "down": \
                WN1.color = (1,0,0,1)
                else: \
                WN1.color = (0,0,1,1)
            on_press:
```

```
            root.LoadSound("White.mp3")
            play2.state = "normal"
            pause.state = "normal"

    RoundedTButton:
        id: WN2
        group: "white noise"
        text: "Pink"
        color: (0,0,0,1)
        size_hint: (.2, .15)
        pos_hint: {"x": .28 ,"top": .75}
        font_size: 25
        font_name: "Centaur"
        color: (0,0,1,1)
        on_state:
            if self.state == "down": \
            WN2.color = (1,0,0,1)
            else: \
            WN2.color = (0,0,1,1)
        on_press:
            root.LoadSound("Pink.mp3")
            play2.state = "normal"
            pause.state = "normal"

    RoundedTButton:
        id: WN3
        group: "white noise"
        text: "Brown"
        color: (0,0,0,1)
        size_hint: (.2, .15)
        pos_hint: {"x": .52 ,"top": .75}
        font_size: 25
        font_name: "Centaur"
        color: (0,0,1,1)
        on_state:
            if self.state == "down": \
            WN3.color = (1,0,0,1)
            else: \
            WN3.color = (0,0,1,1)
        on_press:
            root.LoadSound("Brown.mp3")
            play2.state = "normal"
            pause.state = "normal"

    RoundedTButton:
        id: WN4
        group: "white noise"
        text: "Deep Brown"
        color: (0,0,0,1)
        size_hint: (.2, .15)
        pos_hint: {"x": .76 ,"top": .75}
        font_size: 25
        font_name: "Centaur"
        color: (0,0,1,1)
        on_state:
            if self.state == "down": \
            WN4.color = (1,0,0,1)
            else: \
            WN4.color = (0,0,1,1)
        on_press:
            root.LoadSound("Deep Brown.mp3")
            play2.state = "normal"
            pause.state = "normal"

    RoundedTButton:
        id: WN5
        group: "white noise"
        text: "Beach"
        color: (0,0,0,1)
        size_hint: (.2, .15)
        pos_hint: {"x": .04 ,"top": .55}
        font_size: 25
        font_name: "Centaur"
        color: (0,0,1,1)
        on_state:
            if self.state == "down": \
            WN5.color = (1,0,0,1)
            else: \
            WN5.color = (0,0,1,1)
        on_press:
            root.LoadSound("Beach.mp3")
            play2.state = "normal"
            pause.state = "normal"

    RoundedTButton:
        id: WN6
        group: "white noise"
        text: "Blizzard"
        color: (0,0,0,1)
        size_hint: (.2, .15)
        pos_hint: {"x": .28 ,"top": .55}
        font_size: 25
        font_name: "Centaur"
        color: (0,0,1,1)
        on_state:
            if self.state == "down": \
            WN6.color = (1,0,0,1)
            else: \
            WN6.color = (0,0,1,1)
        on_press:
            root.LoadSound("Blizzard.mp3")
            play2.state = "normal"
            pause.state = "normal"

    RoundedTButton:
        id: WN7
        group: "white noise"
        text: "Creek"
        color: (0,0,0,1)
        size_hint: (.2, .15)
        pos_hint: {"x": .52 ,"top": .55}
        font_size: 25
        font_name: "Centaur"
        color: (0,0,1,1)
        on_state:
            if self.state == "down": \
            WN7.color = (1,0,0,1)
            else: \
            WN7.color = (0,0,1,1)
        on_press:
            root.LoadSound("Creek.mp3")
            play2.state = "normal"
            pause.state = "normal"
```

```
RoundedTButton:
    id: WN8
    group: "white noise"
    text: "Hymns"
    color: (0,0,0,1)
    size_hint: (.2, .15)
    pos_hint: {"x": .76 ,"top": .55}
    font_size: 25
    font_name: "Centaur"
    color: (0,0,1,1)
    on_state:
        if self.state == "down": \
        WN8.color = (1,0,0,1)
        else: \
        WN8.color = (0,0,1,1)
    on_press:
        root.LoadSound("Hymns.mp3")
        play2.state = "normal"
        pause.state = "normal"

RoundedTButton:
    id: WN9
    group: "white noise"
    text: "Train Ride"
    color: (0,0,0,1)
    size_hint: (.2, .15)
    pos_hint: {"x": .76 ,"top": .35}
    font_size: 25
    font_name: "Centaur"
    color: (0,0,1,1)
    on_state:
        if self.state == "down": \
        WN9.color = (1,0,0,1)
        else: \
        WN9.color = (0,0,1,1)
    on_press:
        root.LoadSound("Train Ride.mp3")
        play2.state = "normal"
        pause.state = "normal"
```

```
RoundedTButton:
    id: WN10
    group: "white noise"
    text: "Royal Library"
    color: (0,0,0,1)
    size_hint: (.2, .15)
    pos_hint: {"x": .04 ,"top": .35}
    font_size: 25
    font_name: "Centaur"
    color: (0,0,1,1)
    on_state:
        if self.state == "down": \
        WN10.color = (1,0,0,1)
        else: \
        WN10.color = (0,0,1,1)
    on_press:
        root.LoadSound("Royal Library.mp3")
        play2.state = "normal"
        pause.state = "normal"

RoundedTButton:
    id: WN11
    group: "white noise"
    text: "Sleep Music"
    color: (0,0,0,1)
    size_hint: (.2, .15)
    pos_hint: {"x": .28 ,"top": .35}
    font_size: 25
    font_name: "Centaur"
    color: (0,0,1,1)
    on_state:
        if self.state == "down": \
        WN11.color = (1,0,0,1)
        else: \
        WN11.color = (0,0,1,1)
    on_press:
        root.LoadSound("Sleep Music.mp3")
        play2.state = "normal"
        pause.state = "normal"
RoundedTButton:
    id: WN12
    group: "white noise"
    text: "Thunderstorm"
    color: (0,0,0,1)
    size_hint: (.2, .15)
    pos_hint: {"x": .52 ,"top": .35}
    font_size: 25
    font_name: "Centaur"
    color: (0,0,1,1)
    on_state:
        if self.state == "down": \
        WN12.color = (1,0,0,1)
        else: \
        WN12.color = (0,0,1,1)
    on_press:
        root.LoadSound("Thunderstorm.mp3")
        play2.state = "normal"
        pause.state = "normal"

RoundedTButton:
    id: play2
    color: (0,0,0,1)
    size_hint: (.2, .15)
    pos_hint: {"x": .04 ,"top": .15}
    font_size: 25
    font_name: "Centaur"
    color: (0,0,1,1)
    on_press: root.PlaySound()
    on_state:
        if self.state == 'normal': \
        play1.source = '/home/pi/Desktop/Final GUI/icons/solidplay.png'
        if self.state == 'down': \
        play1.source = '/home/pi/Desktop/Final GUI/activated icons/solidplay.png'

    Image:
        id: play1
        source: '/home/pi/Desktop/Final GUI/icons/solidplay.png'
        center_x: self.parent.center_x
        center_y: self.parent.center_y
RoundedTButton:
    id: pause
    color: (0,0,0,1)
    size_hint: (.2, .15)
    pos_hint: {"x": .28 ,"top": .15}
    font_size: 25
    font_name: "Centaur"
    color: (0,0,1,1)
    on_press:
        root.pause()
    on_state:
        if self.state == 'normal': \
        pause1.source = '/home/pi/Desktop/Final GUI/icons/pause.png'
        if self.state == 'down': \
        pause1.source = '/home/pi/Desktop/Final GUI/activated icons/pause.png'

    Image:
        id: pause1
        source: '/home/pi/Desktop/Final GUI/icons/pause.png'
        center_x: self.parent.center_x
        center_y: self.parent.center_y


RoundedButton:
    id: stop
    color: (0,0,0,1)
    size_hint: (.2, .15)
    pos_hint: {"x": .52 ,"top": .15}
    font_size: 25
    font_name: "Centaur"
    color: (0,0,1,1)
    on_press:
        root.stop()
        play2.state = "normal"
        pause.state = "normal"

    Image:
        source: '/home/pi/Desktop/Final GUI/icons/solidstop.png'
        center_x: self.parent.center_x
        center_y: self.parent.center_y
    RoundedButton:
        id: return2
        color: (0,0,0,1)
        size_hint: (.2, .15)
        pos_hint: {"x": .76 ,"top": .15}
        font_size: 25
        font_name: "Centaur"
        color: (0,0,1,1)
        on_press: app.root.current = "main"

        Image:
            source: '/home/pi/Desktop/Final GUI/icons/return.png'
            center_x: self.parent.center_x
            center_y: self.parent.center_y

<ValidPopUp>:
auto_dismiss: False
title: "Alarm Notice:"
size_hint: (.6, .6)
pos_hint: {"x": .2, "top": .9}
background: 'blackwithborder.jfif'
FloatLayout:
    Label:
        id: alarmlabel
        pos_hint: {"x": .4, "top": .7}
        size_hint: (.2,.2)
        font_size: 35
    RoundedButton:
        id: pUAbtn
        text: "Close"
        color: (0,0,0,1)
        font_name: "Centaur"
        size_hint: (.25,.25)
        pos_hint: {"x": .675, "top": .3}
        font_size: 25
        on_press: pUAbtn.backgroundcolor = (0,0,0,0)
        on_release: pUAbtn.background_color = (0,0,0,1)
        on_release: app.root.current = "main"
        on_release: root.dismiss()

<InValidPopUp>:
auto_dismiss: False
title: "Alarm Notice:"
size_hint: (.6, .6)
pos_hint: {"x": .2, "top": .9}
background: 'blackwithborder.jfif'
FloatLayout:
    Label:
        id: alarmlabel
        pos_hint: {"x": .4, "top": .7}
        size_hint: (.2,.2)
        font_size: 20
    RoundedButton:
        id: pUAbtn
        text: "Close"
        color: (0,0,0,1)
        font_name: "Centaur"
        size_hint: (.25,.25)
        pos_hint: {"x": .675, "top": .3}
        font_size: 25
        on_release: root.dismiss()

<LWCPopUp>:
auto_dismiss: False
title: "Left Wireless Charging Configuration"
size_hint: (.6, .6)
pos_hint: {"x": .2, "top": .9}
background: 'blackwithborder.jfif'
FloatLayout:
    RoundedButton:
        id: thirtyminutes
        text: "30 minutes"
        color: (0,0,0,1)
        font_name: "Centaur"
        size_hint: (.35,.35)
        pos_hint: {"x": .1, "top": .9}
        font_size: 25
        on_press:
            root.timedRelay(1800)
            root.dismiss()

    RoundedButton:
        id: onehalfhour
        text: "1.5 Hour"
        color: (0,0,0,1)
        font_name: "Centaur"
        size_hint: (.35,.35)
        pos_hint: {"x": .1, "top": .4}
        font_size: 25
        on_release:
            root.timedRelay(5400)
            root.dismiss()

    RoundedButton:
        id: onehour
        text: "1 Hour"
        color: (0,0,0,1)
        font_name: "Centaur"
        size_hint: (.35,.35)
        pos_hint: {"x": .55, "top": .9}
        font_size: 25
        on_release:
            root.timedRelay(3600)
            root.dismiss()

    RoundedButton:
        id: on
        text: "On"
        color: (0,0,0,1)
        font_name: "Centaur"
        size_hint: (.35,.35)
        pos_hint: {"x": .55, "top": .4}
        font_size: 25
        on_press:
            root.relayOn()
            root.dismiss()

<RWCPopUp>:
auto_dismiss: False
title: "Right Wireless Charging Configuration"
size_hint: (.6, .6)
pos_hint: {"x": .2, "top": .9}
background: 'blackwithborder.jfif'
```

```
FloatLayout:
    RoundedButton:
        id: thirtyminutes
        text: "30 minutes"
        color: (0,0,0,1)
        font_name: "Centaur"
        size_hint: (.35,.35)
        pos_hint: {"x": .1, "top": .9}
        font_size: 25
        on_press:
            root.timedRelay(1800)
            root.dismiss()

    RoundedButton:
        id: onehalfhour
        text: "1.5 Hour"
        color: (0,0,0,1)
        font_name: "Centaur"
        size_hint: (.35,.35)
        pos_hint: {"x": .1, "top": .4}
        font_size: 25
        on_press: onehalfhour.backgroundcolor = (0,0,0,0)
        on_release: onehalfhour.background_color = (0,0,0,1)
        on_release:
            root.timedRelay(5400)
            root.dismiss()

    RoundedButton:
        id: onehour
        text: "1 Hour"
        color: (0,0,0,1)
        font_name: "Centaur"
        size_hint: (.35,.35)
        pos_hint: {"x": .55, "top": .9}
        font_size: 25
        on_press: onehour.backgroundcolor = (0,0,0,0)
        on_release: onehour.background_color = (0,0,0,1)
        on_release:
            root.timedRelay(3600)
            root.dismiss()
        RoundedButton:
            id: on
            text: "On"
            color: (0,0,0,1)
            font_name: "Centaur"
            size_hint: (.35,.35)
            pos_hint: {"x": .55, "top": .4}
            font_size: 25
            on_press: on.backgroundcolor = (0,0,0,0)
            on_release: on.background_color = (0,0,0,1)
            on_release:
                root.relayOn()
                root.dismiss()

<AlarmPopUp>:
    auto_dismiss: False
    title: "Alarm Notice:"
    size_hint: (.6, .6)
    pos_hint: {"x": .2, "top": .9}
    background: 'blackwithborder.jfif'
    FloatLayout:
        RoundedButton:
            id: alarmoff
            text: "Alarm Off"
            color: (0,0,0,1)
            font_name: "Centaur"
            size_hint: (.35,.35)
            pos_hint: {"x": .1, "top": .65}
            font_size: 25
            on_press: root.turnAlarmOff()
            on_release: root.dismiss()

        RoundedButton:
            id: alarmandlightsoff
            text: "All Off"
            color: (0,0,0,1)
            font_name: "Centaur"
            size_hint: (.35,.35)
            pos_hint: {"x": .6, "top": .65}
            font_size: 25
            on_press: root.turnAlarmandLightsOff()
```

```
          on_release: root.dismiss()

<rainbowAnimationPU>:
    auto_dismiss: False
    title: "Rainbow Animation Notice!"
    size_hint: (.6, .6)
    pos_hint: {"x": .2, "top": .9}
    background: 'blackwithborder.jfif'
    FloatLayout:
        Label:
            id: cd
            color: (0,0,1,1)
            font_name: "Centaur"
            text: "Please Wait..."
            font_size: 65
            pos_hint: {"x": 0, "top": 1.2}
        RoundedButton:
            id: pUAbtn
            text: "Close"
            color: (0,0,1,1)
            font_name: "Centaur"
            size_hint: (.25,.25)
            pos_hint: {"x": .375, "top": .3}
            font_size: 25
            on_release: root.dismiss()


<RoundedTButton@ToggleButton>
    background_color: (0,0,0,0)
    background_normal: ''
    canvas.before:
        Color:
            rgba: (10/255.0,0,50/255.0,1)
        RoundedRectangle:
            size: self.size
            pos: self.pos
            radius: [150]

#Creating Custom small round Button

<WirelessChargerButton>
```

```
    background_color: (0,0,0,0)
    background_normal: ''
    canvas.before:
        Color:
            rgba: (10/255.0,0,50/255.0,1)
        RoundedRectangle:
            size: self.size
            pos: self.pos
            radius: [150]
<RoundedButton@Button>
    background_color: (0,0,0,0)
    background_normal: ''
    canvas.before:
        Color:
            rgba: (10/255.0,0,50/255.0,1)
        RoundedRectangle:
            size: self.size
            pos: self.pos
            radius: [150]

#Creating Custom Large Round Button
<RoundedButton2@Button>
    background_color: (0,0,0,0)
    background_normal: ''
    canvas.before:
        Color:
            rgba: (10/255.0,0,50/255.0,1)
        RoundedRectangle:
            size: self.size
            pos: self.pos
            radius: [175]

#Changing Slider Color to blue
<Slider>:
    canvas:
        Color:
            rgba: (0, 0, 1,1)
        BorderImage:
            border: self.border_horizontal if self.orientation == 'horizontal' else self.border_vertical
            pos: (self.x + self.padding, self.center_y - self.background_width / 2) if self.orientation
== 'horizontal' else (self.center_x - self.background_width / 2, self.y + self.padding)
```

```
            size: (self.width - self.padding * 2, self.background_width) if self.orientation ==
'horizontal' else (self.background_width, self.height - self.padding * 2)
            source: (self.background_disabled_horizontal if self.orientation == 'horizontal' else
self.background_disabled_vertical) if self.disabled else (self.background_horizontal if
self.orientation == 'horizontal' else self.background_vertical)
        Color:
            rgba: root.value_track_color if self.value_track and self.orientation == 'horizontal' else
[0, 0, 1,0]
        Line:
            width: self.value_track_width
            points: self.x + self.padding, self.center_y, self.value_pos[0], self.center_y
        Color:
            rgba: root.value_track_color if self.value_track and self.orientation == 'vertical' else [0,
0, 1,0]
        Line:
            width: self.value_track_width
            points: self.center_x, self.y + self.padding, self.center_x, self.value_pos[1]
        Color:
            rgba: (0, 0, 1,1)
    Image:
        pos: (root.value_pos[0] - root.cursor_width / 2, root.center_y - root.cursor_height / 2) if
root.orientation == 'horizontal' else (root.center_x - root.cursor_width / 2, root.value_pos[1] -
root.cursor_height / 2)
        size: root.cursor_size
        source: root.cursor_disabled_image if root.disabled else root.cursor_image
        allow_stretch: True
        keep_ratio: False
```

Appendix F: Arduino Program

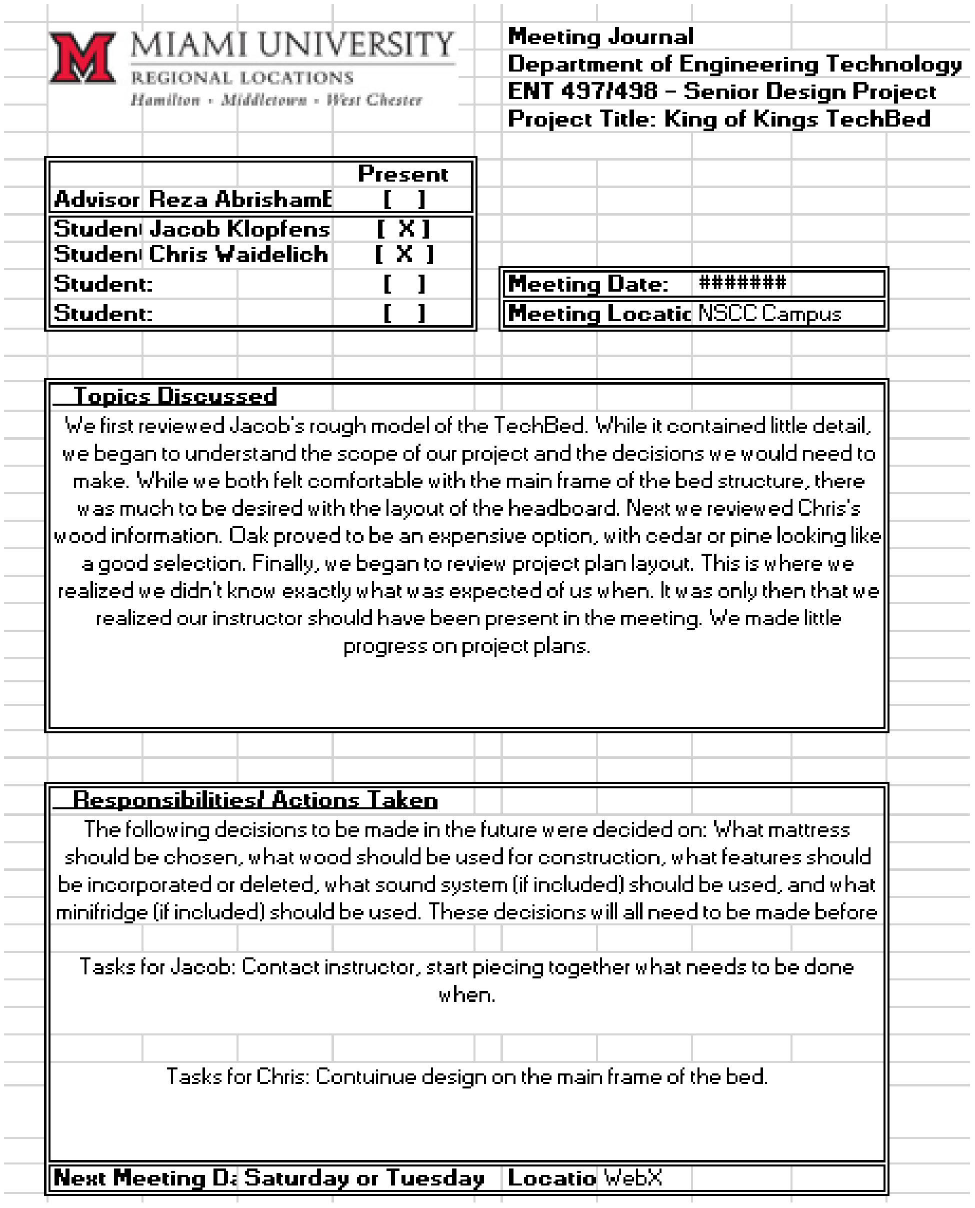
```
#define leftServoA 8
 #define leftServoB 9
 #define rightServoA 11
 #define rightServoB 12
 #include <Servo.h>

 Servo leftServo;
 Servo rightServo;
 int leftServoPin = 5;
 int rightServoPin = 6;

 int leftCounter = 0;
 int rightCounter = 45;
 int aLeftState;
 int aLeftLastState;
 int aRightState;
 int aRightLastState;

 int leftServoPosition = 0;
 int rightServoPosition = 45;


 int leftButton = 10;
 boolean lastButtonLeft = LOW;
 boolean currentButtonLeft = LOW;
 boolean leftLightsOn = false;
 int leftLights = 2;

 int rightButton = 13;
 boolean lastButtonRight = LOW;
 boolean currentButtonRight = LOW;
 boolean rightLightsOn = false;
 int rightLights = 3;

 void setup() {

   pinMode (leftServoA,INPUT);
   pinMode (leftServoB,INPUT);
   pinMode (rightServoA,INPUT);
```

```
  pinMode (rightServoB,INPUT);

  pinMode (leftServoPin, OUTPUT);
  pinMode (rightServoPin, OUTPUT);

  Serial.begin (9600);

  leftServo.attach(leftServoPin);
  rightServo.attach(rightServoPin);
  delay(10);
  leftServo.write(leftServoPosition);
  rightServo.write(rightServoPosition*4);
  delay(1000);
  leftServo.detach();
  rightServo.detach();

  aLeftLastState = digitalRead(leftServoA);
  aRightLastState = digitalRead(rightServoA);

  pinMode(leftButton, INPUT);
  pinMode(rightButton, INPUT);
  pinMode(leftLights, OUTPUT);
  pinMode(rightLights, OUTPUT);
}

boolean debounceLeft (boolean last){
 boolean current = digitalRead (leftButton);
   if (last != current){
       delay (5);
       current = digitalRead (leftButton);
   }
   return current;
   }
boolean debounceRight (boolean last){
 boolean current = digitalRead (rightButton);
   if (last != current){
       delay (5);
       current = digitalRead (rightButton);
   }
   return current;
   }
```

```
void loop() {

   currentButtonLeft = debounceLeft (lastButtonLeft);
   currentButtonRight = debounceRight (lastButtonRight);

   if (lastButtonLeft == LOW && currentButtonLeft == HIGH)
     {
       leftLightsOn = !leftLightsOn;
     }
     lastButtonLeft = currentButtonLeft;

     digitalWrite (leftLights, leftLightsOn);

   if (lastButtonRight == LOW && currentButtonRight == HIGH)
     {
       rightLightsOn = !rightLightsOn;
     }
     lastButtonRight = currentButtonRight;

     digitalWrite (rightLights, rightLightsOn);

  aLeftState = digitalRead(leftServoA);
  aRightState = digitalRead(rightServoA);


  if (aLeftState != aLeftLastState){
    if (digitalRead(leftServoB) != aLeftState) {
      leftCounter --;
    }
    else {
      leftCounter ++;
    }

    if (leftCounter < 0){
     leftCounter = 0;
    }
    if (leftCounter > 40){
     leftCounter = 40;
    }
```

```
 }

 if (aRightState != aRightLastState){
  if (digitalRead(rightServoB) != aRightState){
    rightCounter --;
  }
  else {
    rightCounter ++;
  }

  if (rightCounter < 5){
    rightCounter = 5;
   }
   if (rightCounter > 45){
    rightCounter = 45;
   }

 }

 aLeftLastState = aLeftState;
 aRightLastState = aRightState;

if (leftServoPosition != leftCounter){
  leftServo.attach(leftServoPin);
  delay(10);
  leftServo.write(leftCounter*4);
  delay(20);
  leftServo.detach();

  leftServoPosition = leftCounter;
}
if (rightServoPosition != rightCounter){
  rightServo.attach(rightServoPin);
  delay(10);
  rightServo.write(rightCounter*4);
  delay(20);
  rightServo.detach();
  rightServoPosition = rightCounter;
}
}
```

Appendix G: Weekly Reports



MIAMI UNIVERSITY
REGIONAL LOCATIONS
Hamilton · Middletown · West Chester

Meeting Journal
Department of Engineering Technology
ENT 497/498 – Senior Design Project
Project Title: King of Kings TechBed

		Present
Advisor	**Reza AbrishamE**	**[ ]**
Studen	**Jacob Klopfens**	**[X]**
Studen	**Chris Waidelich**	**[X]**
Student:		**[ ]**
Student:		**[ ]**

Meeting Date:	#######
Meeting Locatic	NSCC Campus

Topics Discussed

We first reviewed Jacob's rough model of the TechBed. While it contained little detail, we began to understand the scope of our project and the decisions we would need to make. While we both felt comfortable with the main frame of the bed structure, there was much to be desired with the layout of the headboard. Next we reviewed Chris's wood information. Oak proved to be an expensive option, with cedar or pine looking like a good selection. Finally, we began to review project plan layout. This is where we realized we didn't know exactly what was expected of us when. It was only then that we realized our instructor should have been present in the meeting. We made little progress on project plans.

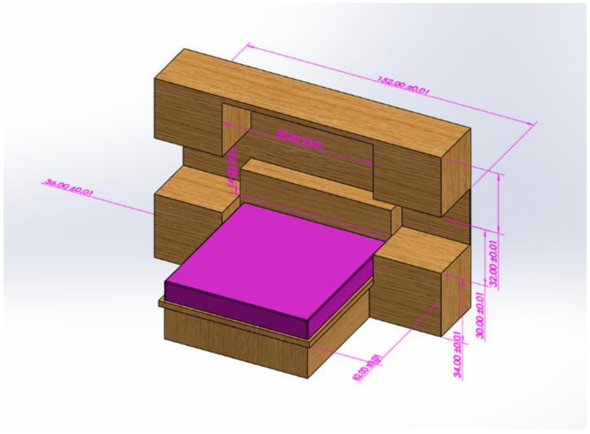
Responsibilities/ Actions Taken

The following decisions to be made in the future were decided on: What mattress should be chosen, what wood should be used for construction, what features should be incorporated or deleted, what sound system (if included) should be used, and what minifridge (if included) should be used. These decisions will all need to be made before

Tasks for Jacob: Contact instructor, start piecing together what needs to be done when.

Tasks for Chris: Contuinue design on the main frame of the bed.

Next Meeting Dɛ	**Saturday or Tuesday**	**Locatio**	WebX



MIAMI UNIVERSITY
REGIONAL LOCATIONS
Hamilton • Middletown • West Chester

Meeting Journal
Department of Engineering Technology
ENT 497/498 – Senior Design Project
Project Title: King of Kings TechBed

		Present
Advisor	Reza AbrishamE	[ ]
Studen	Jacob Klopfens	[X]
Studen	Chris Waidelich	[X]
Student:		[ ]
Student:		[ ]

Meeting Date:	#######
Meeting Locatic	NSCC Campus

Topics Discussed

We began the meeting by breaking down where we are at in the project. At this point, we decided, after what was said in last class, to pour all of our resources into coming up with a design and finishing it out. We then began to dig deeper into the design. Because we wanted to know more about our layout, we decided upon deeper aspects of our product selections, such as sound system and minifridge. This allowed us to begin to make a detailed layout of the headboard, as we begin to figure out where different things will be placed.

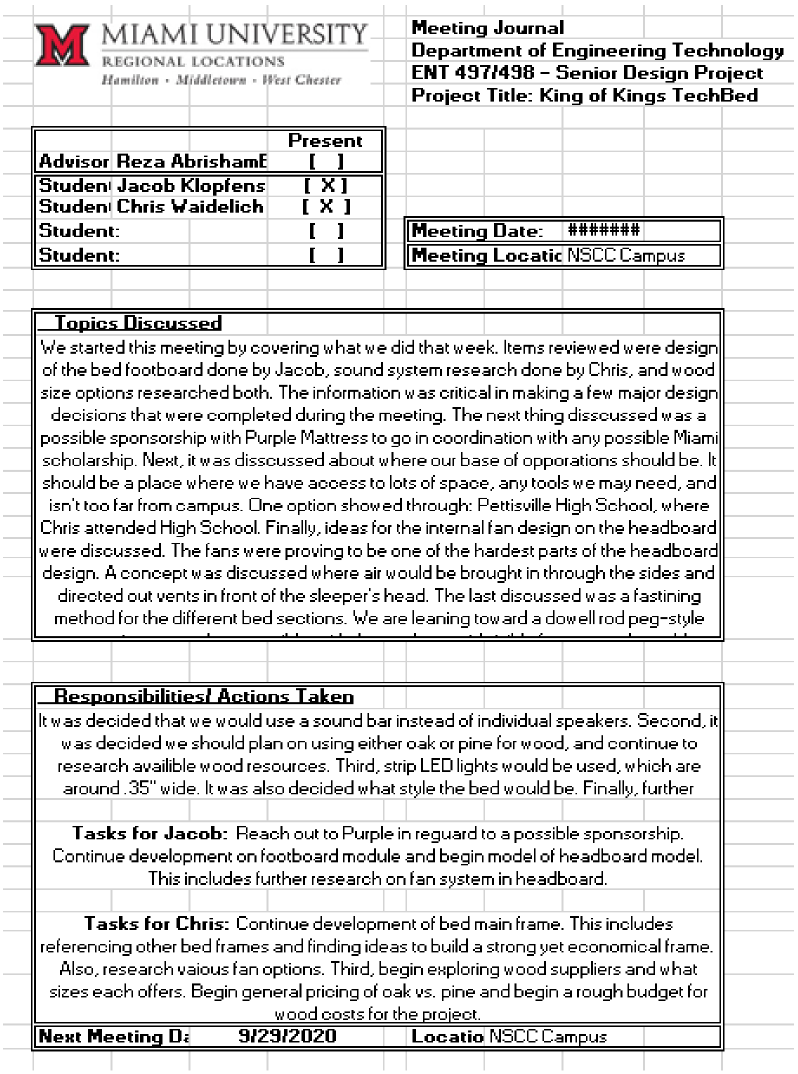
Responsibilities/ Actions Taken

We decided on either a 3 cubic foot minifridge or a 1.7 cubic foot mini fridge, with emphasis on the latter. We also decided on a home theature sound system style, with a 5.1 or 5.2 sound layout. This would allow us to plan for four speakers and a subwoofer. We also decided to add storage drawers to the nightstands where there was room. The other conclusion made during the meeting was proceeding with a 4-pin fan style, which would allow for better control and a quieter experience.

Tasks for Jacob: Continue development of the model. Reduce headboard size, begin to break down frame into individual componants, add footboard and begin footboard design, and begin layout of headboard.

Tasks for Chris: Continue research on sound system, minifridge, and other small tech componants such as projector, lights, and clocks. Give general dimensions for each to Jacob to impliment on the model. Reseach board selection options for the purpose of selecting what kind of framing options we can use standard on the design.

Next Meeting Date: Tuesday	Locatio NSCC Campus



MIAMI UNIVERSITY
REGIONAL LOCATIONS
Hamilton • Middletown • West Chester

Meeting Journal
Department of Engineering Technology
ENT 497/498 – Senior Design Project
Project Title: King of Kings TechBed

		Present
Advisor	**Reza AbrishamE**	**[ ]**
Student	**Jacob Klopfens**	**[X]**
Student	**Chris Waidelich**	**[X]**
Student:		**[ ]**
Student:		**[ ]**

Meeting Date:	**#######**
Meeting Locatio	NSCC Campus

Topics Discussed

We started this meeting by covering what we did that week. Items reviewed were design of the bed footboard done by Jacob, sound system research done by Chris, and wood size options researched both. The information was critical in making a few major design decisions that were completed during the meeting. The next thing disscussed was a possible sponsorship with Purple Mattress to go in coordination with any possible Miami scholarship. Next, it was dissoussed about where our base of opporations should be. It should be a place where we have access to lots of space, any tools we may need, and isn't too far from campus. One option showed through: Pettisville High School, where Chris attended High School. Finally, ideas for the internal fan design on the headboard were discussed. The fans were proving to be one of the hardest parts of the headboard design. A concept was discussed where air would be brought in through the sides and directed out vents in front of the sleeper's head. The last discussed was a fastining method for the different bed sections. We are leaning toward a dowell rod peg-style

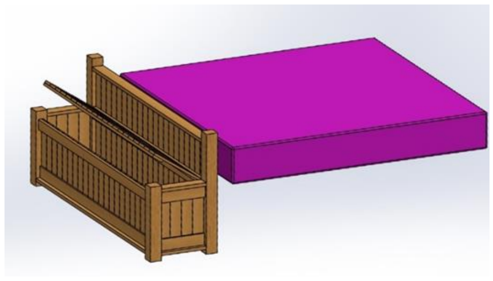
Responsibilities/ Actions Taken

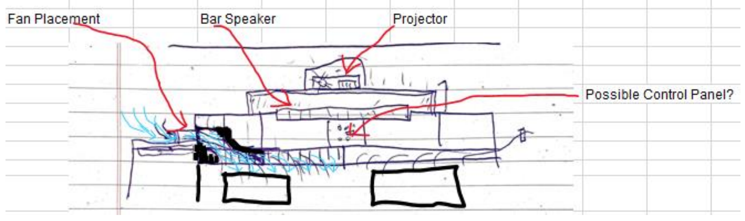
It was decided that we would use a sound bar instead of individual speakers. Second, it was decided we should plan on using either oak or pine for wood, and continue to research availible wood resources. Third, strip LED lights would be used, which are around .35" wide. It was also decided what style the bed would be. Finally, further

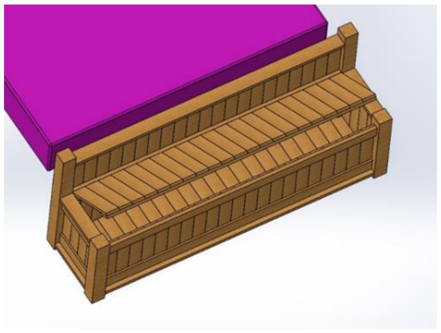
Tasks for Jacob: Reach out to Purple in reguard to a possible sponsorship. Continue development on footboard module and begin model of headboard model. This includes further research on fan system in headboard.

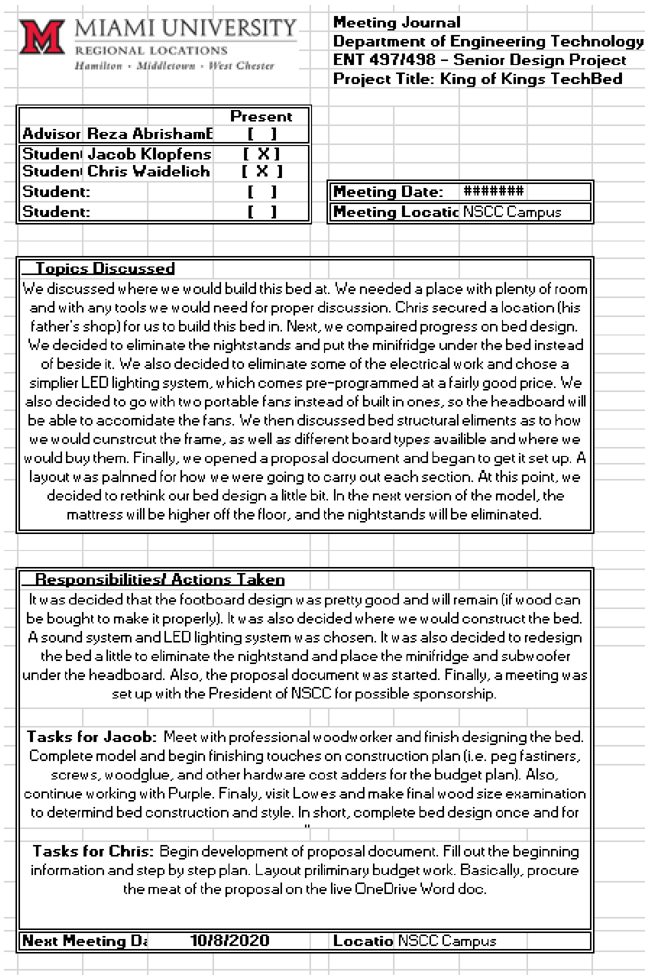
Tasks for Chris: Continue development of bed main frame. This includes referencing other bed frames and finding ideas to build a strong yet economical frame. Also, research vaious fan options. Third, begin exploring wood suppliers and what sizes each offers. Begin general pricing of oak vs. pine and begin a rough budget for wood costs for the project.

Next Meeting Da	**9/29/2020**	**Locatio**	NSCC Campus









MIAMI UNIVERSITY
REGIONAL LOCATIONS
Hamilton • Middletown • West Chester

Meeting Journal
Department of Engineering Technology
ENT 497/498 – Senior Design Project
Project Title: King of Kings TechBed

		Present
Advisor	**Reza AbrishamE**	**[ ]**
Student	**Jacob Klopfens**	**[X]**
Student	**Chris Waidelich**	**[X]**
Student:		**[ ]**
Student:		**[ ]**

Meeting Date:	#######
Meeting Locatic	NSCC Campus

Topics Discussed

We discussed where we would build this bed at. We needed a place with plenty of room and with any tools we would need for proper discussion. Chris secured a location (his father's shop) for us to build this bed in. Next, we compaired progress on bed design. We decided to eliminate the nightstands and put the minifridge under the bed instead of beside it. We also decided to eliminate some of the electrical work and chose a simplier LED lighting system, which comes pre-programmed at a fairly good price. We also decided to go with two portable fans instead of built in ones, so the headboard will be able to accomidate the fans. We then discussed bed structural eliments as to how we would construct the frame, as well as different board types availible and where we would buy them. Finally, we opened a proposal document and began to get it set up. A layout was palnned for how we were going to carry out each section. At this point, we decided to rethink our bed design a little bit. In the next version of the model, the mattress will be higher off the floor, and the nightstands will be eliminated.

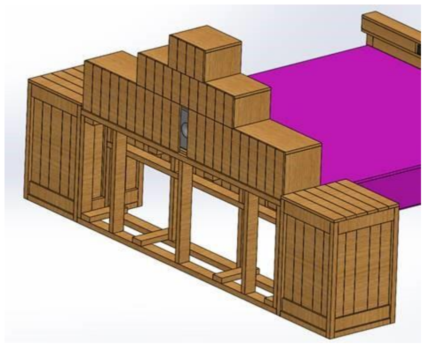
Responsibilities/ Actions Taken

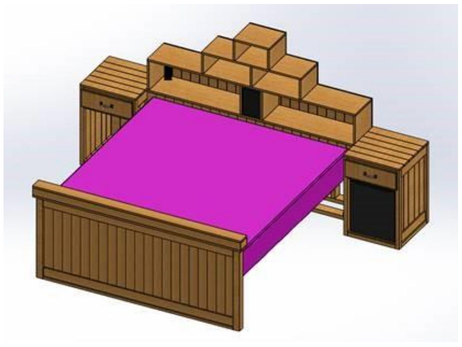
It was decided that the footboard design was pretty good and will remain (if wood can be bought to make it properly). It was also decided where we would construct the bed. A sound system and LED lighting system was chosen. It was also decided to redesign the bed a little to eliminate the nightstand and place the minifridge and subwoofer under the headboard. Also, the proposal document was started. Finally, a meeting was set up with the President of NSCC for possible sponsorship.

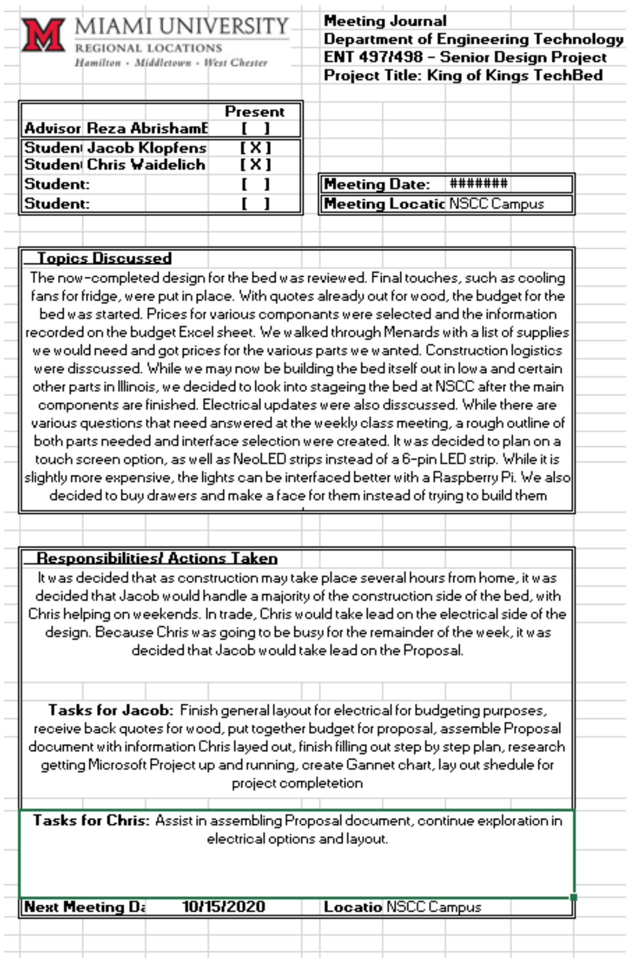
Tasks for Jacob: Meet with professional woodworker and finish designing the bed. Complete model and begin finishing touches on construction plan (i.e. peg fastiners, screws, woodglue, and other hardware cost adders for the budget plan). Also, continue working with Purple. Finaly, visit Lowes and make final wood size examination to determind bed construction and style. In short, complete bed design once and for

Tasks for Chris: Begin development of proposal document. Fill out the beginning information and step by step plan. Layout priliminary budget work. Basically, procure the meat of the proposal on the live OneDrive Word doc.

Next Meeting Da	**10/8/2020**	**Locatio**	NSCC Campus







MIAMI UNIVERSITY
REGIONAL LOCATIONS
Hamilton • Middletown • West Chester

Meeting Journal
Department of Engineering Technology
ENT 497/498 – Senior Design Project
Project Title: King of Kings TechBed

	Present
Advisor Reza AbrishamE	[ ]
Student Jacob Klopfens	[X]
Student Chris Waidelich	[X]
Student:	[ ]
Student:	[ ]

Meeting Date: #######
Meeting Locatic NSCC Campus

Topics Discussed

The now-completed design for the bed was reviewed. Final touches, such as cooling fans for fridge, were put in place. With quotes already out for wood, the budget for the bed was started. Prices for various componants were selected and the information recorded on the budget Excel sheet. We walked through Menards with a list of supplies we would need and got prices for the various parts we wanted. Construction logistics were disscussed. While we may now be building the bed itself out in Iowa and certain other parts in Illinois, we decided to look into stageing the bed at NSCC after the main components are finished. Electrical updates were also disscussed. While there are various questions that need answered at the weekly class meeting, a rough outline of both parts needed and interface selection were created. It was decided to plan on a touch screen option, as well as NeoLED strips instead of a 6-pin LED strip. While it is slightly more expensive, the lights can be interfaced better with a Raspberry Pi. We also decided to buy drawers and make a face for them instead of trying to build them

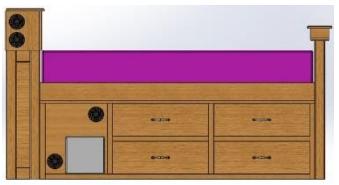
Responsibilities/ Actions Taken

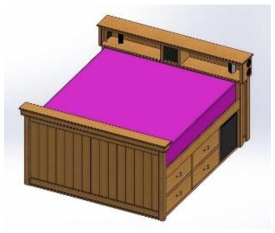
It was decided that as construction may take place several hours from home, it was decided that Jacob would handle a majority of the construction side of the bed, with Chris helping on weekends. In trade, Chris would take lead on the electrical side of the design. Because Chris was going to be busy for the remainder of the week, it was decided that Jacob would take lead on the Proposal.

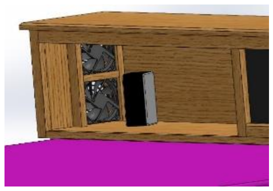
Tasks for Jacob: Finish general layout for electrical for budgeting purposes, receive back quotes for wood, put together budget for proposal, assemble Proposal document with information Chris layed out, finish filling out step by step plan, research getting Microsoft Project up and running, create Gannet chart, lay out shedule for project completetion

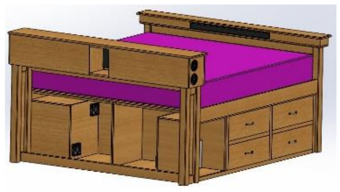
Tasks for Chris: Assist in assembling Proposal document, continue exploration in electrical options and layout.

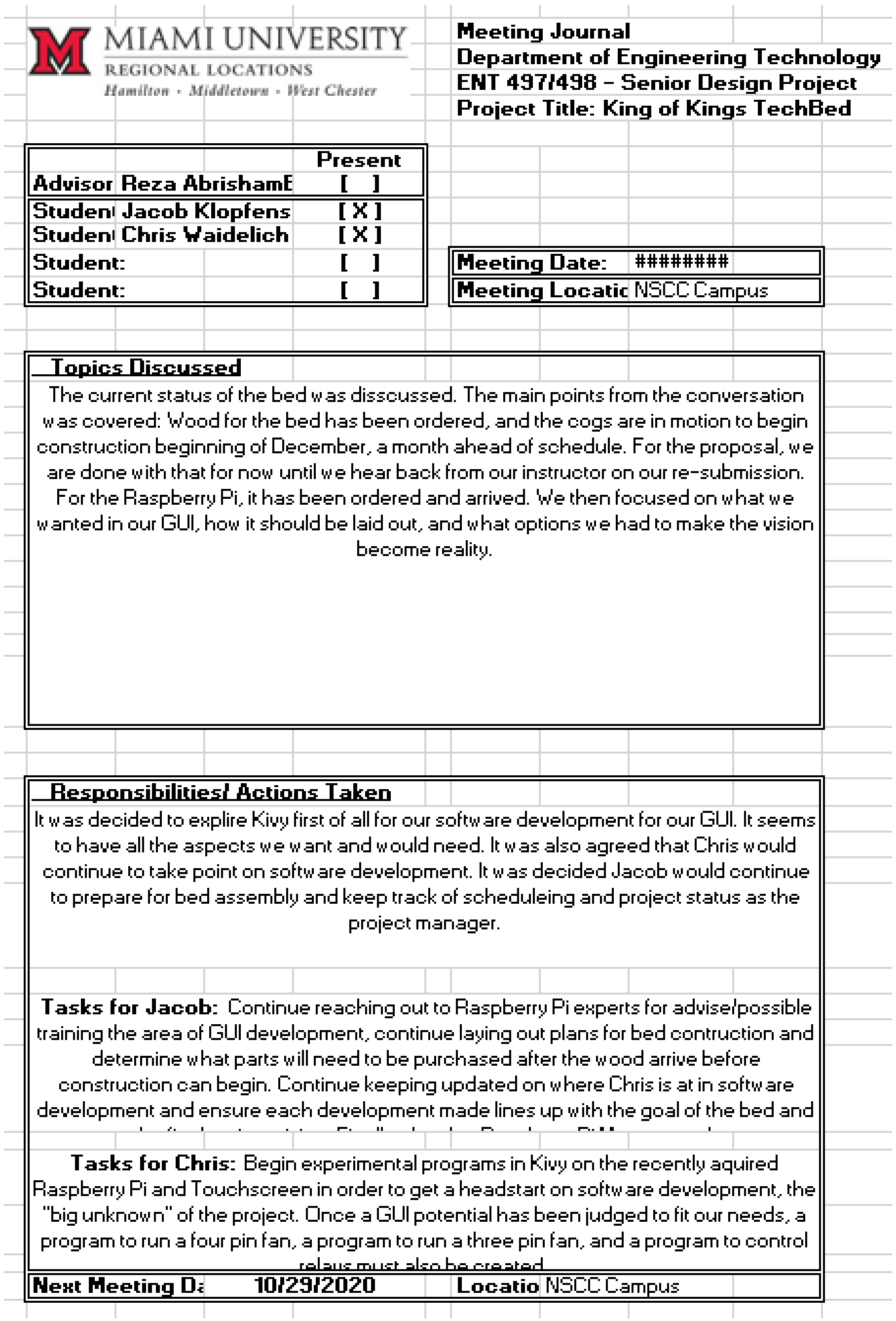
Next Meeting Da **10/15/2020** **Locatio** NSCC Campus











MIAMI UNIVERSITY
REGIONAL LOCATIONS
Hamilton · Middletown · West Chester

Meeting Journal
Department of Engineering Technology
ENT 497/498 – Senior Design Project
Project Title: King of Kings TechBed

		Present
Advisor	**Reza AbrishamE**	**[ ]**
Studen	**Jacob Klopfens**	**[X]**
Studen	**Chris Waidelich**	**[X]**
Student:		**[ ]**
Student:		**[ ]**

Meeting Date:	########
Meeting Locatic	NSCC Campus

Topics Discussed

The current status of the bed was disscussed. The main points from the conversation was covered: Wood for the bed has been ordered, and the cogs are in motion to begin construction beginning of December, a month ahead of schedule. For the proposal, we are done with that for now until we hear back from our instructor on our re-submission. For the Raspberry Pi, it has been ordered and arrived. We then focused on what we wanted in our GUI, how it should be laid out, and what options we had to make the vision become reality.

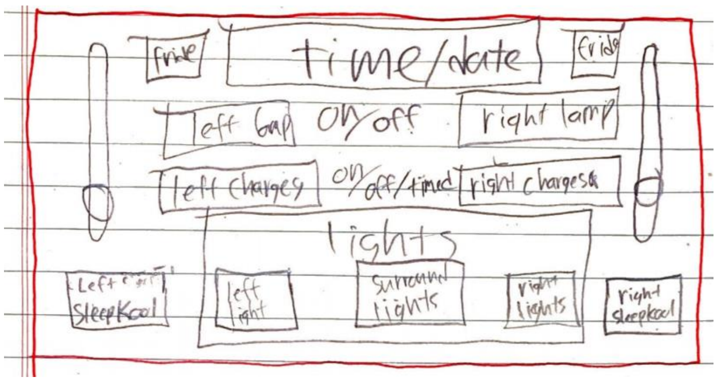
Responsibilities/ Actions Taken

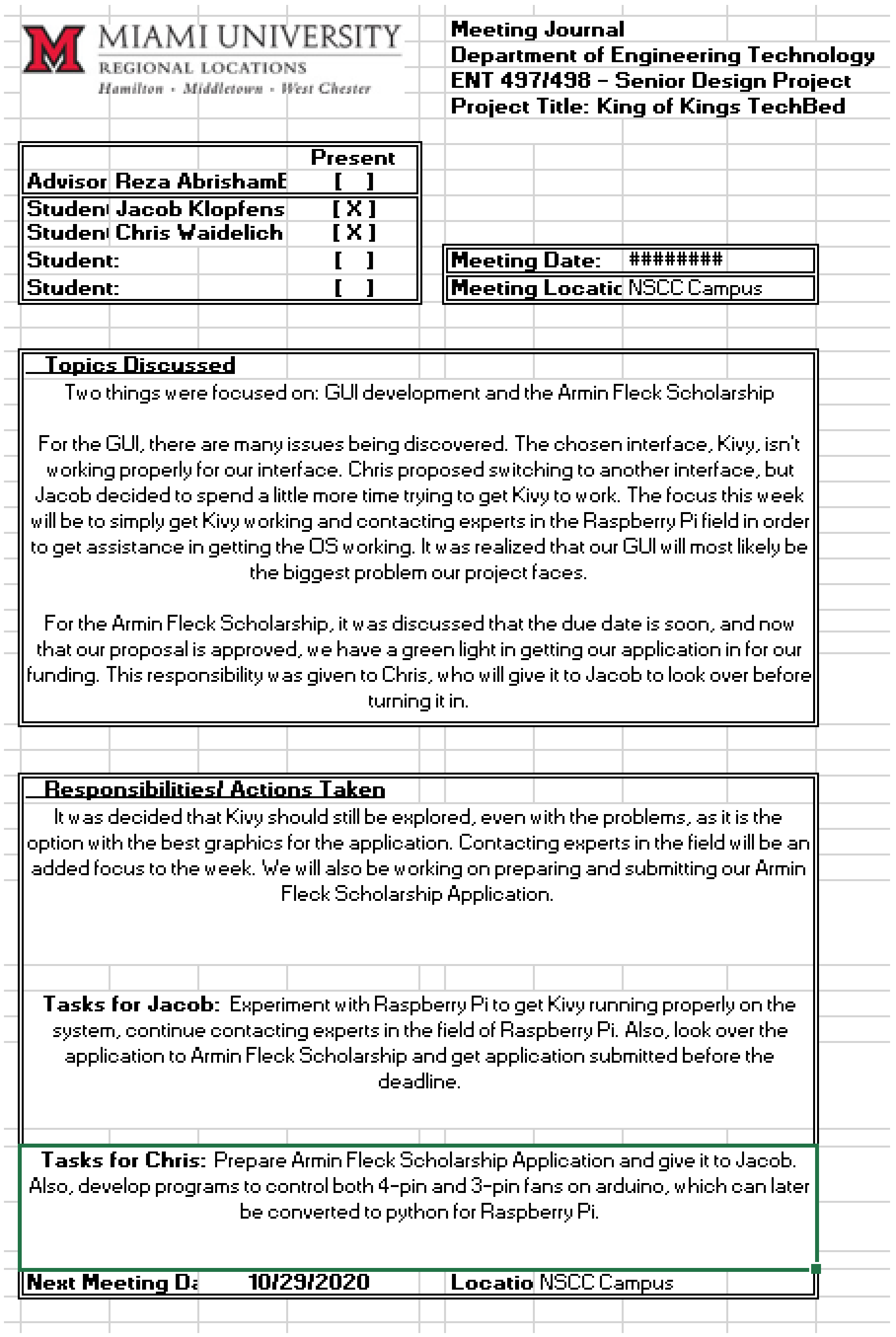
It was decided to explire Kivy first of all for our software development for our GUI. It seems to have all the aspects we want and would need. It was also agreed that Chris would continue to take point on software development. It was decided Jacob would continue to prepare for bed assembly and keep track of scheduleing and project status as the project manager.

Tasks for Jacob: Continue reaching out to Raspberry Pi experts for advise/possible training the area of GUI development, continue laying out plans for bed contruction and determine what parts will need to be purchased after the wood arrive before construction can begin. Continue keeping updated on where Chris is at in software development and ensure each development made lines up with the goal of the bed and

Tasks for Chris: Begin experimental programs in Kivy on the recently aquired Raspberry Pi and Touchscreen in order to get a headstart on software development, the "big unknown" of the project. Once a GUI potential has been judged to fit our needs, a program to run a four pin fan, a program to run a three pin fan, and a program to control relays must also be created

Next Meeting Da	**10/29/2020**	**Locatio**	NSCC Campus





MIAMI UNIVERSITY
REGIONAL LOCATIONS
Hamilton · Middletown · West Chester

Meeting Journal
Department of Engineering Technology
ENT 497/498 – Senior Design Project
Project Title: King of Kings TechBed

		Present
Advisor	**Reza AbrishamE**	[ ]
Student	**Jacob Klopfens**	[X]
Student	**Chris Waidelich**	[X]
Student:		[ ]
Student:		[ ]

Meeting Date:	########
Meeting Locatic	NSCC Campus

Topics Discussed

Two things were focused on: GUI development and the Armin Fleck Scholarship

For the GUI, there are many issues being discovered. The chosen interface, Kivy, isn't working properly for our interface. Chris proposed switching to another interface, but Jacob decided to spend a little more time trying to get Kivy to work. The focus this week will be to simply get Kivy working and contacting experts in the Raspberry Pi field in order to get assistance in getting the OS working. It was realized that our GUI will most likely be the biggest problem our project faces.

For the Armin Fleck Scholarship, it was discussed that the due date is soon, and now that our proposal is approved, we have a green light in getting our application in for our funding. This responsibility was given to Chris, who will give it to Jacob to look over before turning it in.

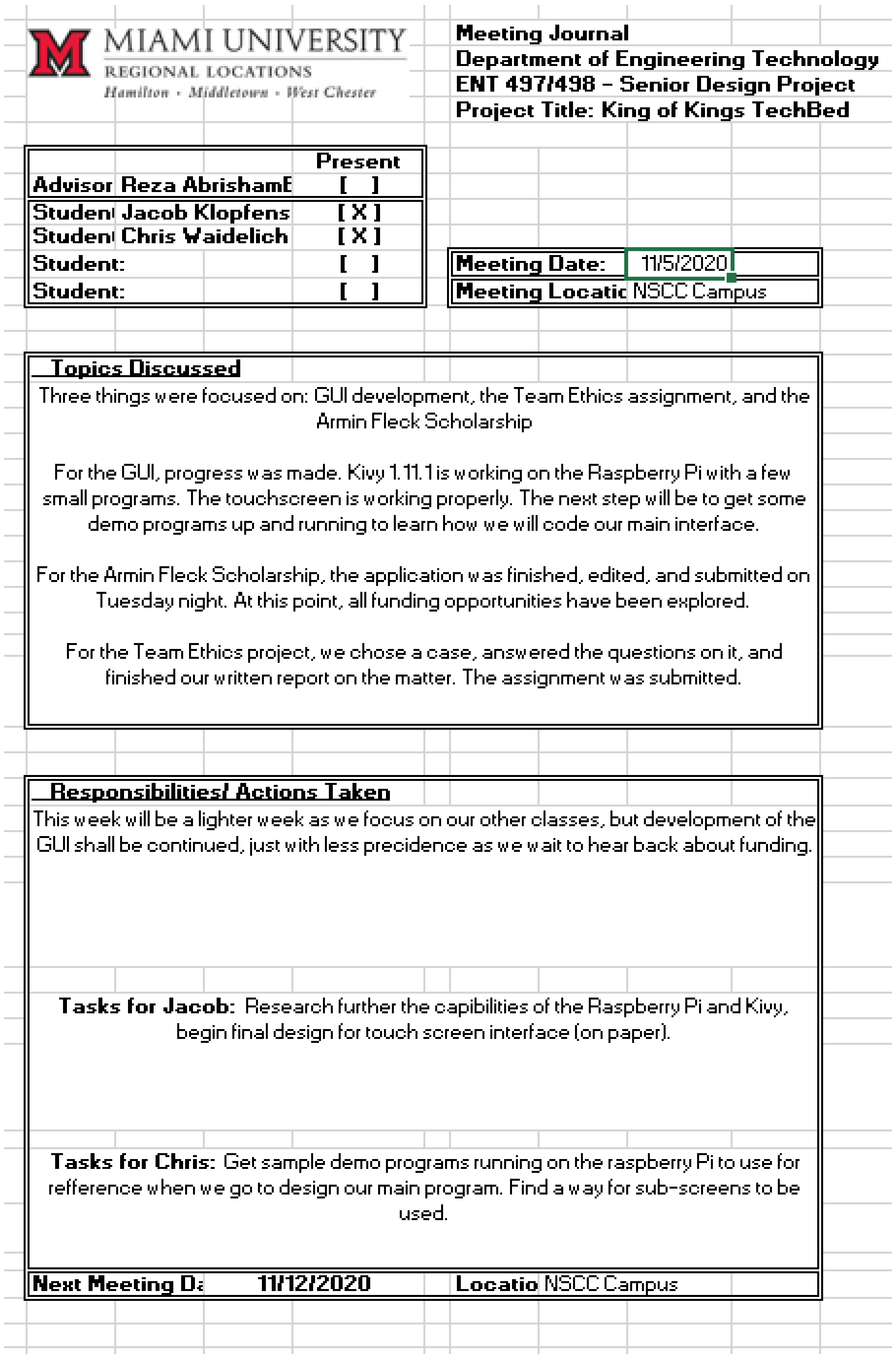
Responsibilities/ Actions Taken

It was decided that Kivy should still be explored, even with the problems, as it is the option with the best graphics for the application. Contacting experts in the field will be an added focus to the week. We will also be working on preparing and submitting our Armin Fleck Scholarship Application.

Tasks for Jacob: Experiment with Raspberry Pi to get Kivy running properly on the system, continue contacting experts in the field of Raspberry Pi. Also, look over the application to Armin Fleck Scholarship and get application submitted before the deadline.

Tasks for Chris: Prepare Armin Fleck Scholarship Application and give it to Jacob. Also, develop programs to control both 4-pin and 3-pin fans on arduino, which can later be converted to python for Raspberry Pi.

Next Meeting Da	**10/29/2020**	**Locatio**	NSCC Campus



MIAMI UNIVERSITY
REGIONAL LOCATIONS
Hamilton · Middletown · West Chester

Meeting Journal
Department of Engineering Technology
ENT 497/498 – Senior Design Project
Project Title: King of Kings TechBed

	Present
Advisor Reza Abrisham	[ ]
Studen Jacob Klopfens	[X]
Studen Chris Waidelich	[X]
Student:	[ ]
Student:	[ ]

Meeting Date:	11/5/2020
Meeting Locatic	NSCC Campus

Topics Discussed

Three things were focused on: GUI development, the Team Ethics assignment, and the Armin Fleck Scholarship

For the GUI, progress was made. Kivy 1.11.1 is working on the Raspberry Pi with a few small programs. The touchscreen is working properly. The next step will be to get some demo programs up and running to learn how we will code our main interface.

For the Armin Fleck Scholarship, the application was finished, edited, and submitted on Tuesday night. At this point, all funding opportunities have been explored.

For the Team Ethics project, we chose a case, answered the questions on it, and finished our written report on the matter. The assignment was submitted.

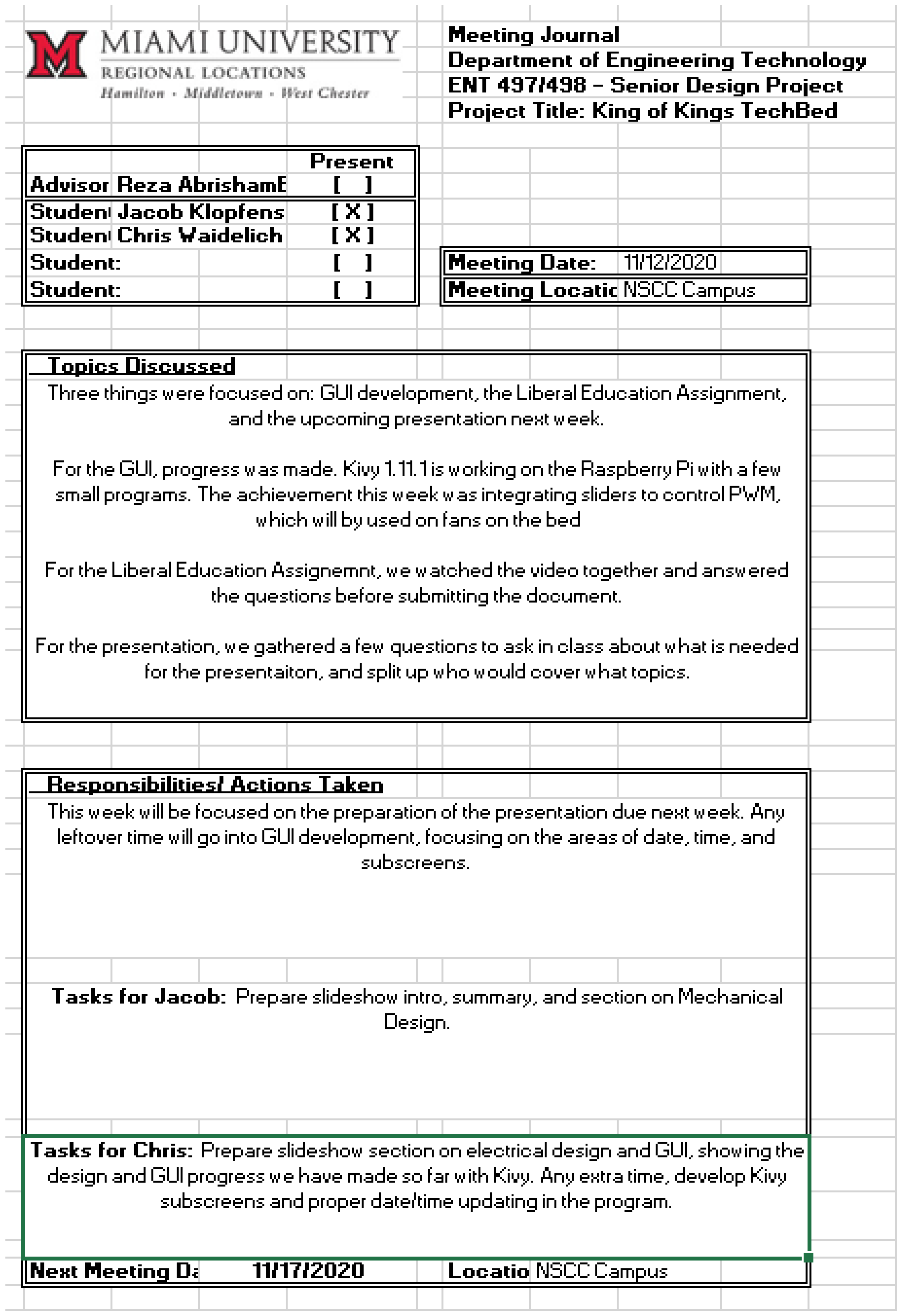
Responsibilities/ Actions Taken

This week will be a lighter week as we focus on our other classes, but development of the GUI shall be continued, just with less precidence as we wait to hear back about funding.

Tasks for Jacob: Research further the capibilities of the Raspberry Pi and Kivy, begin final design for touch screen interface (on paper).

Tasks for Chris: Get sample demo programs running on the raspberry Pi to use for refference when we go to design our main program. Find a way for sub-screens to be used.

Next Meeting Da	**11/12/2020**	**Locatio**	NSCC Campus



MIAMI UNIVERSITY
REGIONAL LOCATIONS
Hamilton · Middletown · West Chester

Meeting Journal
Department of Engineering Technology
ENT 497/498 – Senior Design Project
Project Title: King of Kings TechBed

		Present
Advisor	**Reza AbrishamE**	**[ ]**
Studen	**Jacob Klopfens**	**[X]**
Studen	**Chris Waidelich**	**[X]**
Student:		**[ ]**
Student:		**[ ]**

Meeting Date:	11/12/2020
Meeting Locatic	NSCC Campus

Topics Discussed

Three things were focused on: GUI development, the Liberal Education Assignment, and the upcoming presentation next week.

For the GUI, progress was made. Kivy 1.11.1 is working on the Raspberry Pi with a few small programs. The achievement this week was integrating sliders to control PWM, which will by used on fans on the bed

For the Liberal Education Assignemnt, we watched the video together and answered the questions before submitting the document.

For the presentation, we gathered a few questions to ask in class about what is needed for the presentaiton, and split up who would cover what topics.

Responsibilities/ Actions Taken

This week will be focused on the preparation of the presentation due next week. Any leftover time will go into GUI development, focusing on the areas of date, time, and subscreens.

Tasks for Jacob: Prepare slideshow intro, summary, and section on Mechanical Design.

Tasks for Chris: Prepare slideshow section on electrical design and GUI, showing the design and GUI progress we have made so far with Kivy. Any extra time, develop Kivy subscreens and proper date/time updating in the program.

Next Meeting Da	**11/17/2020**	**Locatio**	NSCC Campus

MIAMI UNIVERSITY
REGIONAL LOCATIONS
Hamilton · Middletown · West Chester

Meeting Journal
Department of Engineering Technology
ENT 497/498 – Senior Design Project
Project Title:

		Present
Advisor	**Reza**	**[ ]**
Studen	**acob Klopfenste**	**[x]**
Studen	**Chris Waidelick**	**[x]**

Meeting Date:	1/29/2021
Meeting Locatic	Chris's House

Topics Discussed

A summary of what was accomplished was covered. The specific entities are as follows:

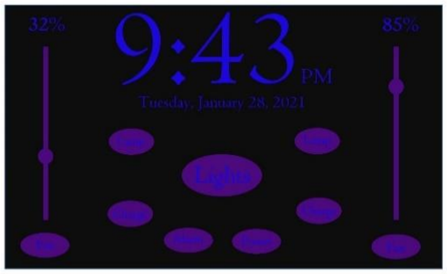
- A look at the current and updated model (specifically, a briefing of the changes made since start of construction
- A review of the parts that have been purchased, along with what final parts need to be purchased yet. It was discovered we have the wrong fans... we have 3-pin fans, we need 4-pin fans
- A design for the GUI was decided upon... the font will be Centaur, the color scheme plurple and blue (night colors), and the layout will have rounded buttons

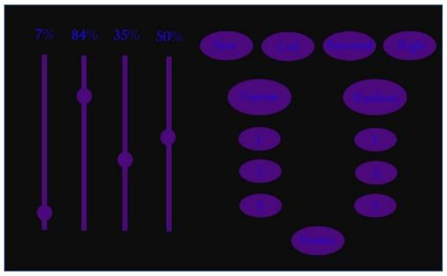
Responsibilities/ Actions Taken

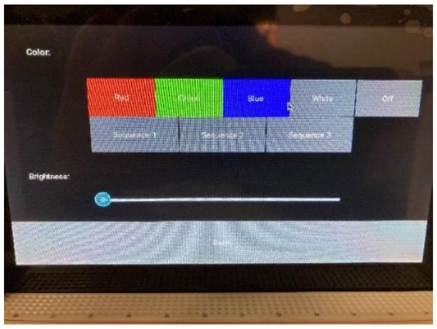
Jacob Klopfenstein:
Jacob will return to Iowa for the first two-three weeks of February to complete the contruction of the bedframe, afterwhich it will be transferred back to Ohio for setup at NSCC in preparation for wiring. Jacob will also oversee the purchase of the replacement fans.

Chris Waidelich:
Chris will continue development of programming options for the light strips, along with preparation programs for the new fans. He will also be responsible for adapting the desired GUI layout and shown in the pictures.

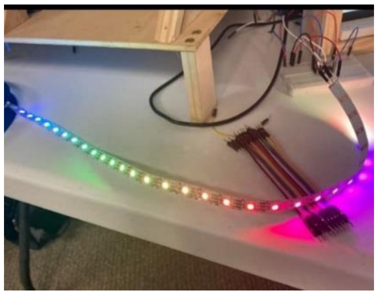
Next Meeting Da	**2/13/2021?**	**Locatio**	**Northwest State Community College**

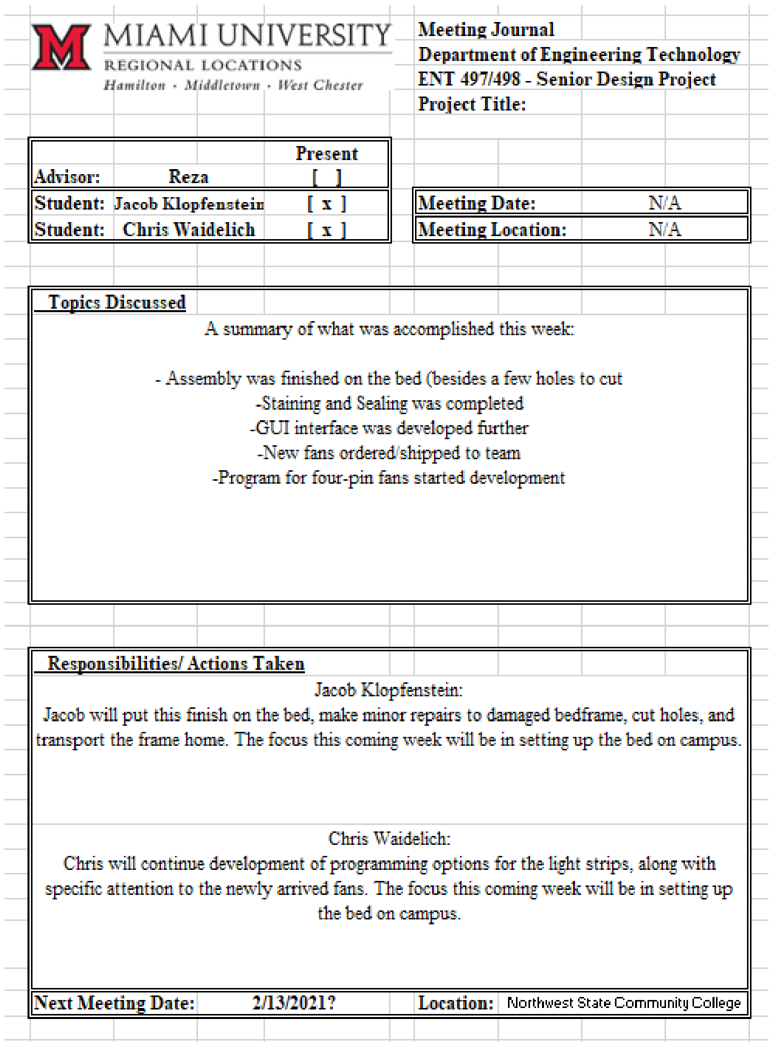






ROGUE





MIAMI UNIVERSITY
REGIONAL LOCATIONS
Hamilton · Middletown · West Chester

Meeting Journal
Department of Engineering Technology
ENT 497/498 - Senior Design Project
Project Title:

		Present
Advisor:	**Reza**	[ ]
Student:	**Jacob Klopfenstein**	[x]
Student:	**Chris Waidelich**	[x]

Meeting Date:	N/A
Meeting Location:	N/A

Topics Discussed

A summary of what was accomplished this week:

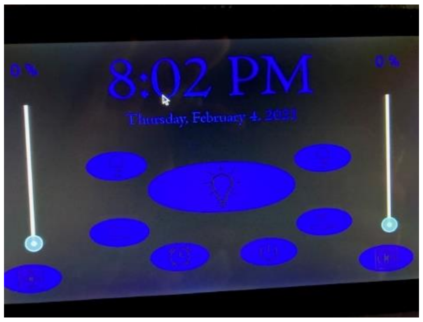
- Assembly was finished on the bed (besides a few holes to cut
-Staining and Sealing was completed
-GUI interface was developed further
-New fans ordered/shipped to team
-Program for four-pin fans started development

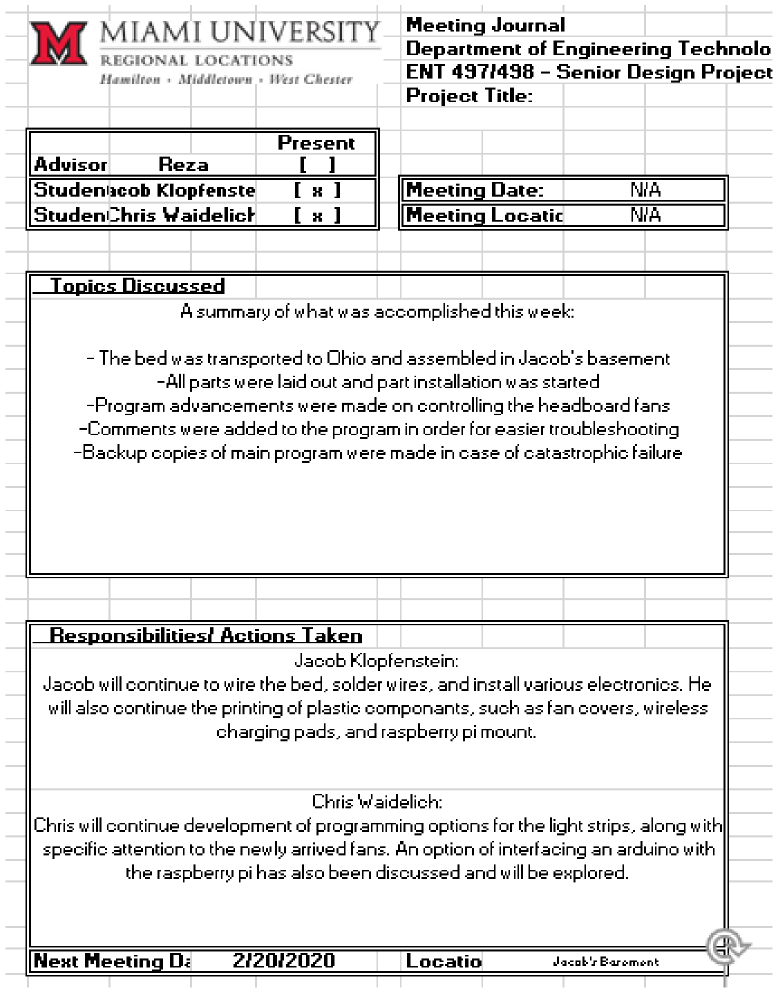
Responsibilities/ Actions Taken

Jacob Klopfenstein:
Jacob will put this finish on the bed, make minor repairs to damaged bedframe, cut holes, and transport the frame home. The focus this coming week will be in setting up the bed on campus.

Chris Waidelich:
Chris will continue development of programming options for the light strips, along with specific attention to the newly arrived fans. The focus this coming week will be in setting up the bed on campus.

Next Meeting Date:	2/13/2021?	**Location:**	Northwest State Community College





MIAMI UNIVERSITY
REGIONAL LOCATIONS
Hamilton • Middletown • West Chester

Meeting Journal
Department of Engineering Technolo
ENT 497/498 – Senior Design Project
Project Title:

		Present
Advisor	**Reza**	[ ]
Studen	**acob Klopfenste**	[x]
Studen	**Chris Waidelich**	[x]

Meeting Date:	N/A
Meeting Locatic	N/A

Topics Discussed

A summary of what was accomplished this week:

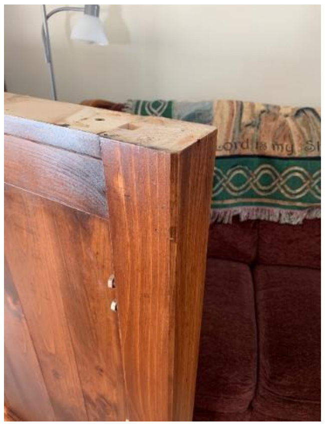
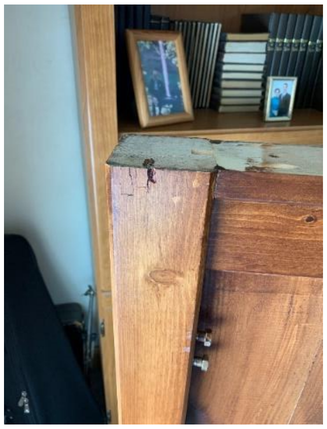
- The bed was transported to Ohio and assembled in Jacob's basement
-All parts were laid out and part installation was started
-Program advancements were made on controlling the headboard fans
-Comments were added to the program in order for easier troubleshooting
-Backup copies of main program were made in case of catastrophic failure

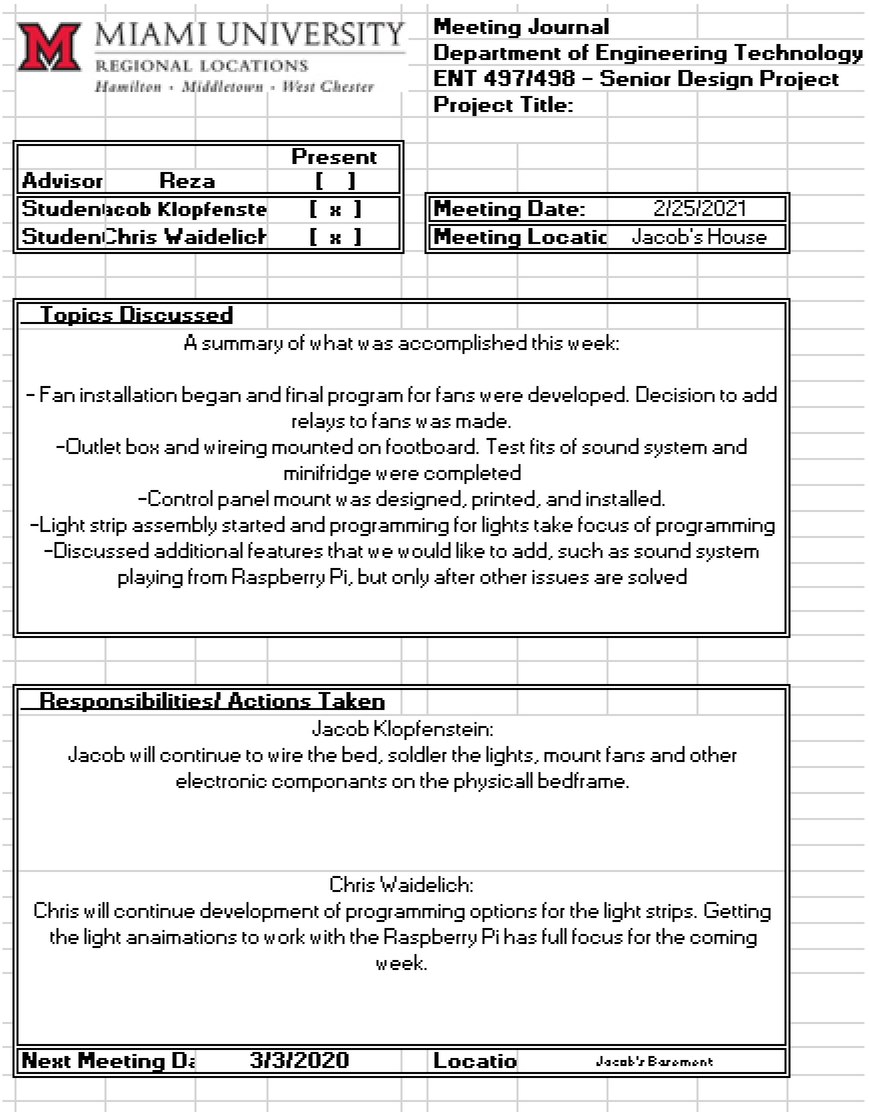
Responsibilities/ Actions Taken

Jacob Klopfenstein:
Jacob will continue to wire the bed, solder wires, and install various electronics. He will also continue the printing of plastic componants, such as fan covers, wireless charging pads, and raspberry pi mount.

Chris Waidelich:
Chris will continue development of programming options for the light strips, along with specific attention to the newly arrived fans. An option of interfacing an arduino with the raspberry pi has also been discussed and will be explored.

Next Meeting Da	**2/20/2020**	**Locatio**	Jacob's Basement



MIAMI UNIVERSITY
REGIONAL LOCATIONS
Hamilton • Middletown • West Chester

Meeting Journal
Department of Engineering Technology
ENT 497/498 – Senior Design Project
Project Title:

		Present
Advisor	**Reza**	**[ ]**
Studen	**acob Klopfenste**	**[x]**
Studen	**Chris Waidelich**	**[x]**

Meeting Date:	2/25/2021
Meeting Locatic	Jacob's House

Topics Discussed

A summary of what was accomplished this week:

- Fan installation began and final program for fans were developed. Decision to add relays to fans was made.
-Outlet box and wireing mounted on footboard. Test fits of sound system and minifridge were completed
-Control panel mount was designed, printed, and installed.
-Light strip assembly started and programming for lights take focus of programming
-Discussed additional features that we would like to add, such as sound system playing from Raspberry Pi, but only after other issues are solved

Responsibilities/ Actions Taken

Jacob Klopfenstein:
Jacob will continue to wire the bed, soldler the lights, mount fans and other electronic componants on the physicall bedframe.

Chris Waidelich:
Chris will continue development of programming options for the light strips. Getting the light anaimations to work with the Raspberry Pi has full focus for the coming week.

Next Meeting Da	**3/3/2020**	**Locatio**	Jacob's Basement

Close Up View of Mounting Brackets

Mounted Touch Screen

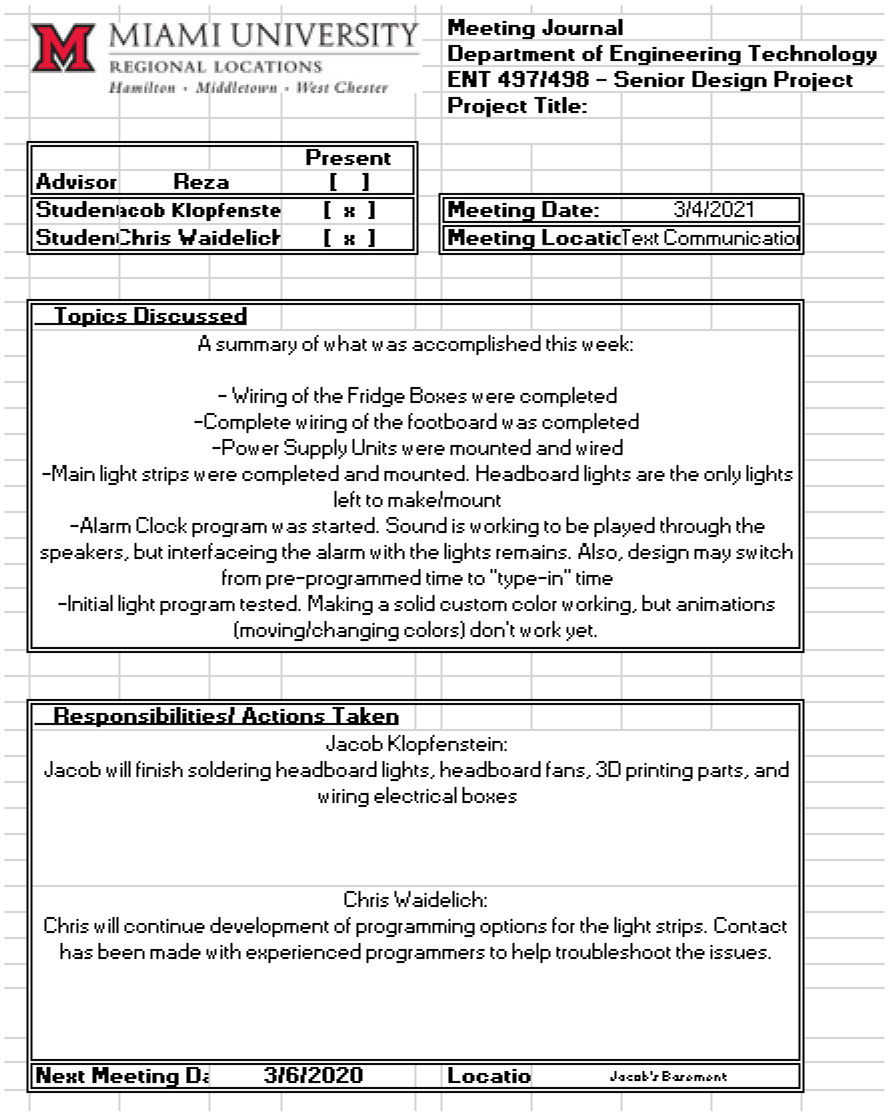
Close Up View of Servo Mechanism

Raspberry Pi/Touch screen Mount from Back

View of Fans/Printed Fan Covers

Mounted Fans Inside Headboard

Soldering Light Strips



MIAMI UNIVERSITY
REGIONAL LOCATIONS
Hamilton · Middletown · West Chester

Meeting Journal
Department of Engineering Technology
ENT 497/498 – Senior Design Project
Project Title:

		Present
Advisor	**Reza**	**[ ]**
Studen	**acob Klopfenste**	**[x]**
Studen	**Chris Waidelich**	**[x]**

Meeting Date:	3/4/2021
Meeting Locatio	Text Communicatio

Topics Discussed

A summary of what was accomplished this week:

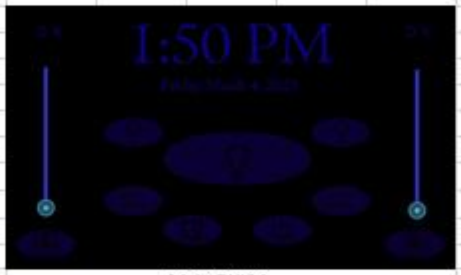
- Wiring of the Fridge Boxes were completed
-Complete wiring of the footboard was completed
-Power Supply Units were mounted and wired
-Main light strips were completed and mounted. Headboard lights are the only lights left to make/mount
-Alarm Clock program was started. Sound is working to be played through the speakers, but interfaceing the alarm with the lights remains. Also, design may switch from pre-programmed time to "type-in" time
-Initial light program tested. Making a solid custom color working, but animations (moving/changing colors) don't work yet.

Responsibilities/ Actions Taken

Jacob Klopfenstein:
Jacob will finish soldering headboard lights, headboard fans, 3D printing parts, and wiring electrical boxes

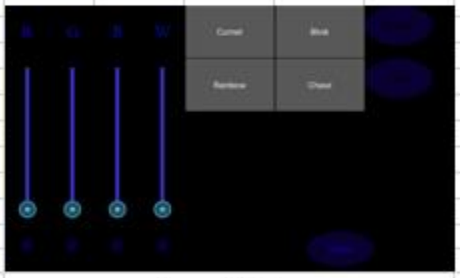
Chris Waidelich:
Chris will continue development of programming options for the light strips. Contact has been made with experienced programmers to help troubleshoot the issues.

Next Meeting Da	**3/6/2020**	**Locatio**	Jacob's Basement



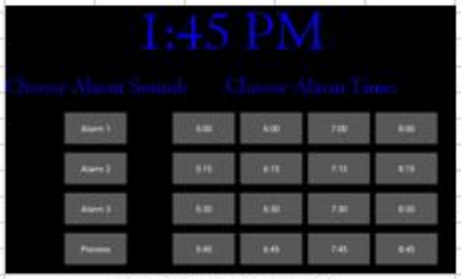
Main GUI Menu

Wired Fridge Box



Current Light Menu

RGBW NeoPixel Lights Installed



Current Alarm Clock Menu

Speaker Bar Installed

Power Supply Units Mounted

MIAMI UNIVERSITY
REGIONAL LOCATIONS
Hamilton · Middletown · West Chester

Meeting Journal
Department of Engineering Technology
ENT 497/498 - Senior Design Project
Project Title:

		Present
Advisor:	**Reza**	[ ]
Student:	**Jacob Klopfenstein**	[x]
Student:	**Chris Waidelich**	[x]

Meeting Date:	3/6/2021
Meeting Location:	Jacob's House

Topics Discussed

A summary of what was accomplished this week:

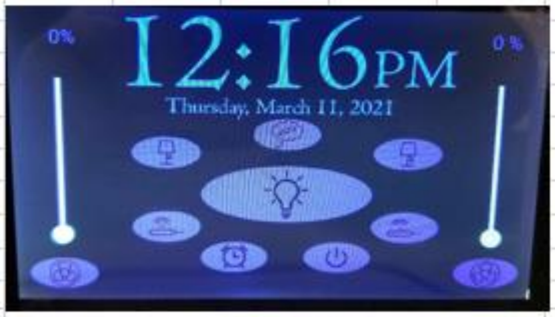
- Left Headboard Fans fully wired
-Power lines for lights created and hooked up
-Headboard lights created and installed
-Color wheel selected and programming began on it
-Alarm Clock menu assembled and tested (NOTE: Needs "Alarm Off" Pop-up programmed)
-Wireless chargers timed menu created
-Fan symbol updated to round instead of square
-Main menu layout updated

Responsibilities/ Actions Taken

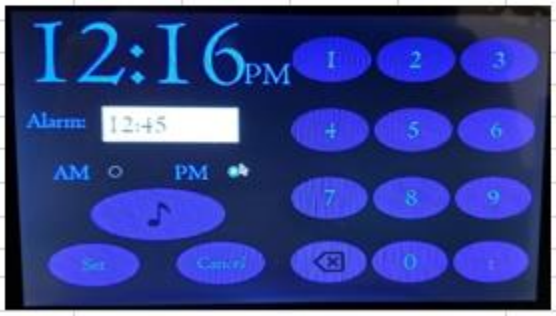
Jacob Klopfenstein:
Jacob will finish soldering headboard lights, headboard fans, 3D printing parts for fan asthetics and aerodynamics, and wiring electrical boxes

Chris Waidelich:
Chris will continue development of programming options for the light strips. First priority is completeing the integration of the color wheel, then rainbow patterns if possible. Development will also continue on alarm clock "Dismiss" popup, toggleing symbols, and alarm & sleep sounds.

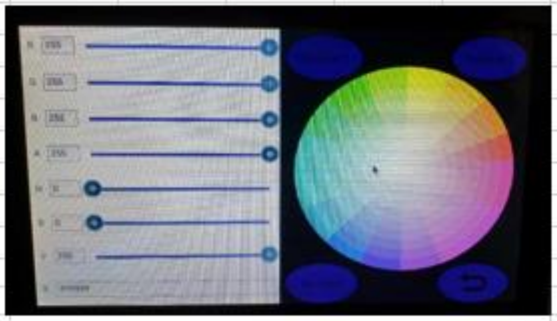
Next Meeting Date:	**3/13/2020**	**Location:**	Jacob's Basement



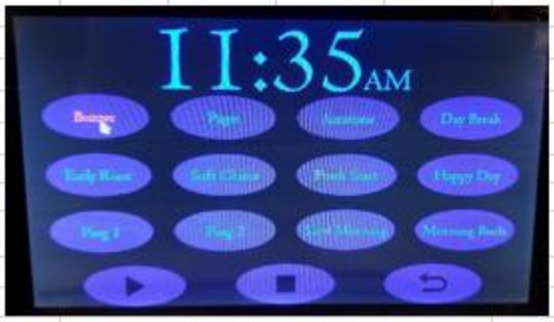
Main GUI Menu



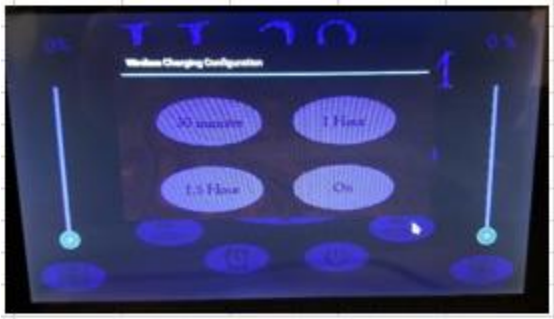
Alarm Screen



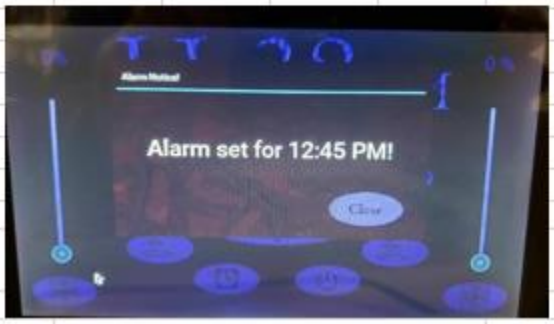
Current Light Menu (W/Color Wheel)



Alarm Sound Select

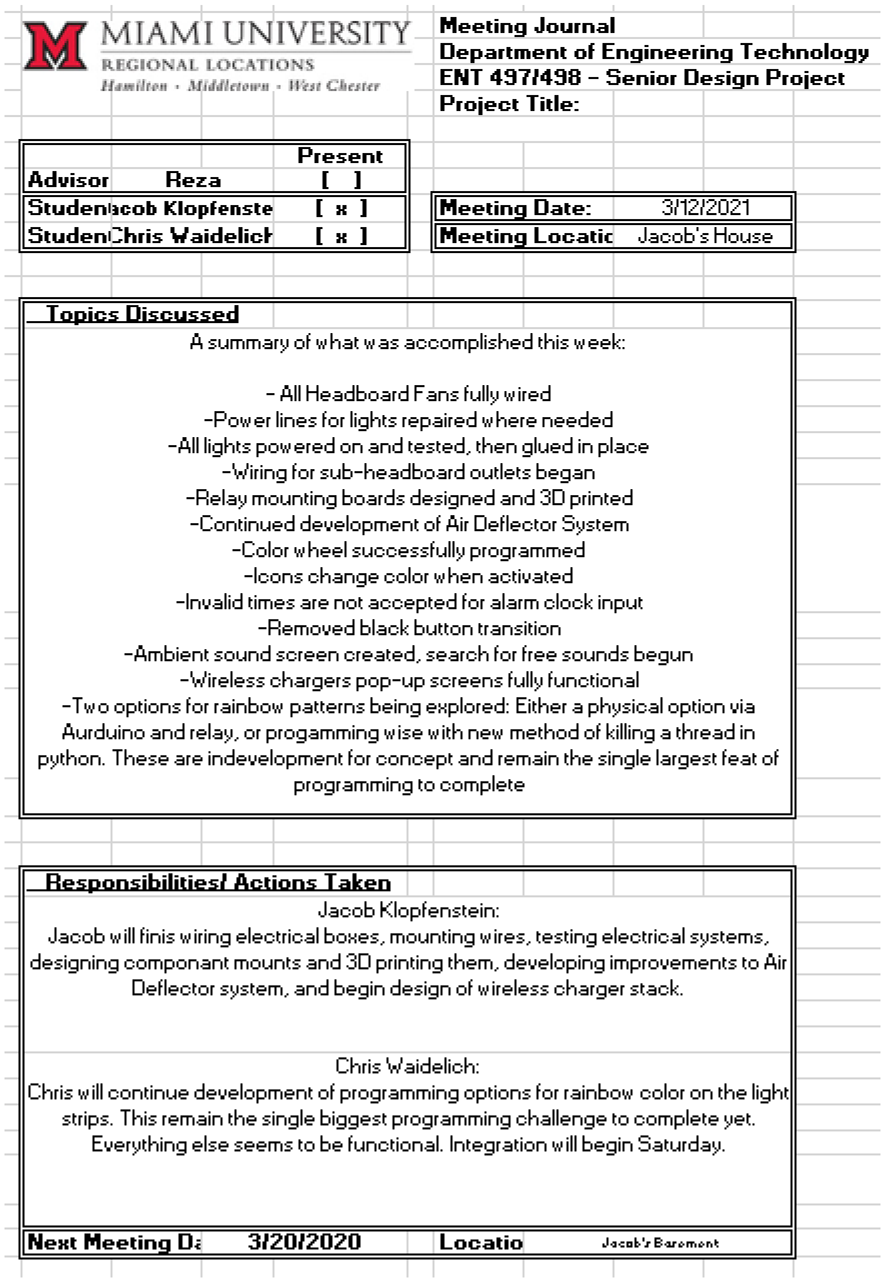


Wireless Charger Menu



Alarm set for 12:45 PM!

Alarm Set Popup



MIAMI UNIVERSITY
REGIONAL LOCATIONS
Hamilton • Middletown • West Chester

Meeting Journal
Department of Engineering Technology
ENT 497/498 – Senior Design Project
Project Title:

		Present
Advisor	**Reza**	**[ ]**
Studen	**acob Klopfenste**	**[x]**
Studen	**Chris Waidelich**	**[x]**

Meeting Date:	3/12/2021
Meeting Locatic	Jacob's House

Topics Discussed

A summary of what was accomplished this week:

- All Headboard Fans fully wired
-Power lines for lights repaired where needed
-All lights powered on and tested, then glued in place
-Wiring for sub-headboard outlets began
-Relay mounting boards designed and 3D printed
-Continued development of Air Deflector System
-Color wheel successfully programmed
-Icons change color when activated
-Invalid times are not accepted for alarm clock input
-Removed black button transition
-Ambient sound screen created, search for free sounds begun
-Wireless chargers pop-up screens fully functional
-Two options for rainbow patterns being explored: Either a physical option via Aurduino and relay, or progamming wise with new method of killing a thread in python. These are indevelopment for concept and remain the single largest feat of programming to complete

Responsibilities/ Actions Taken

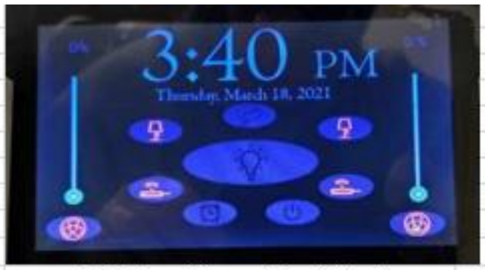
Jacob Klopfenstein:
Jacob will finis wiring electrical boxes, mounting wires, testing electrical systems, designing componant mounts and 3D printing them, developing improvements to Air Deflector system, and begin design of wireless charger stack.

Chris Waidelich:
Chris will continue development of programming options for rainbow color on the light strips. This remain the single biggest programming challenge to complete yet. Everything else seems to be functional. Integration will begin Saturday.

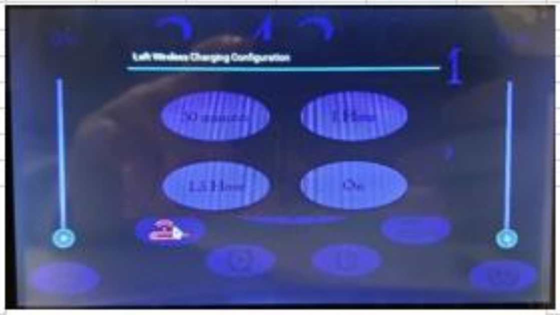
Next Meeting Da	**3/20/2020**	**Locatio**	Jacob's Basement

Beginning of wiring of sub-headboard outlets

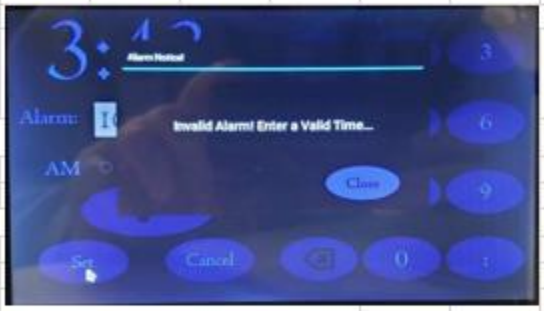
3D Printed Relay Mounts



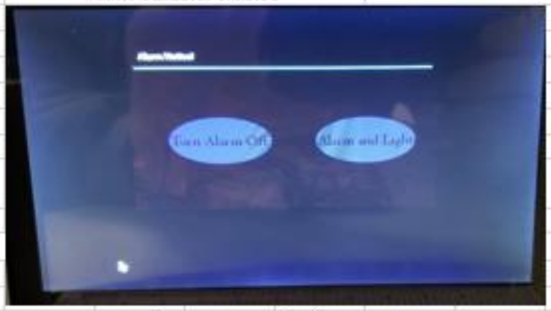
Main Menu (when on, things light up)



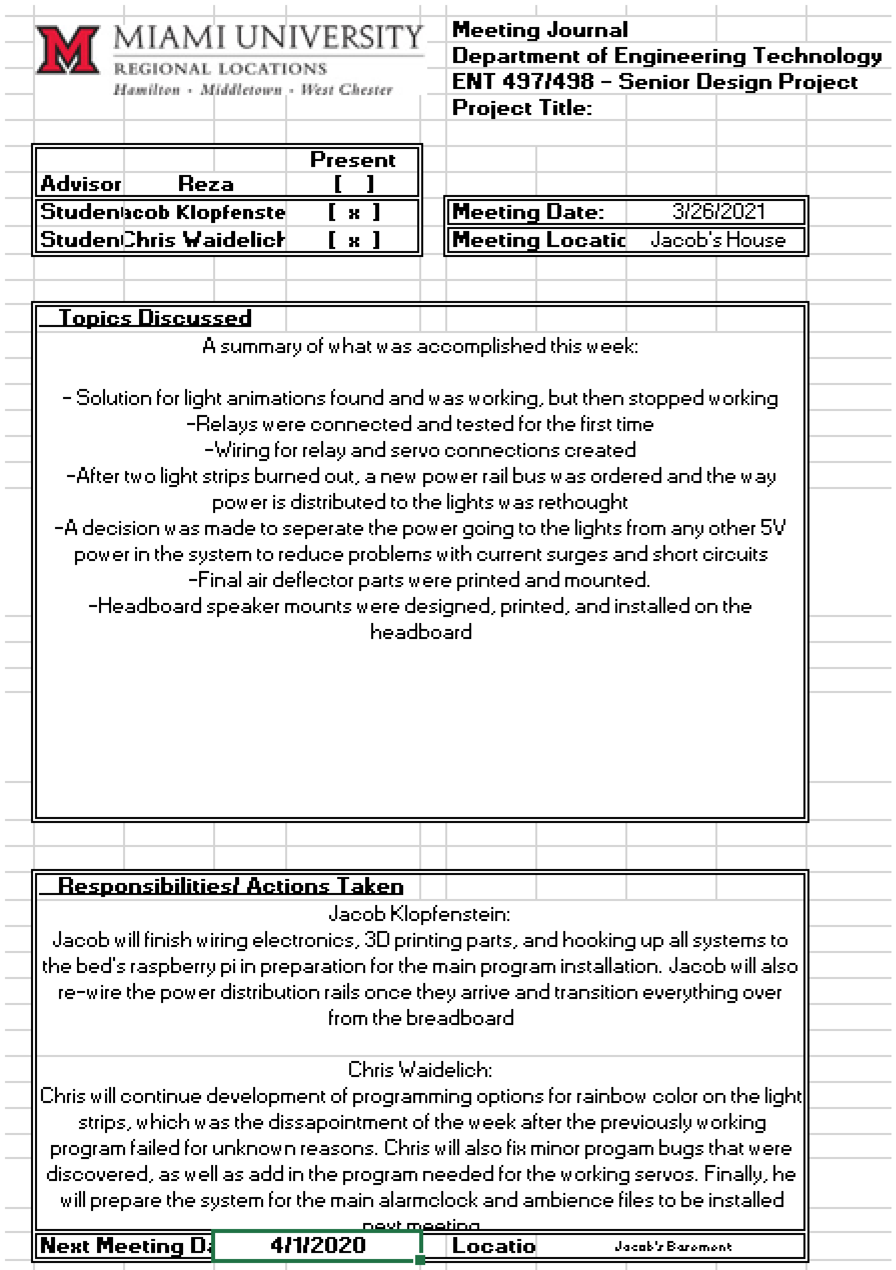
Wireless charger menu



Time validator created



Alarm popup created



MIAMI UNIVERSITY
REGIONAL LOCATIONS
Hamilton · Middletown · West Chester

Meeting Journal
Department of Engineering Technology
ENT 497/498 – Senior Design Project
Project Title:

		Present
Advisor	**Reza**	**[ ]**
Studen	**acob Klopfenste**	**[x]**
Studen	**Chris Waidelich**	**[x]**

Meeting Date:	3/26/2021
Meeting Locatic	Jacob's House

Topics Discussed

A summary of what was accomplished this week:

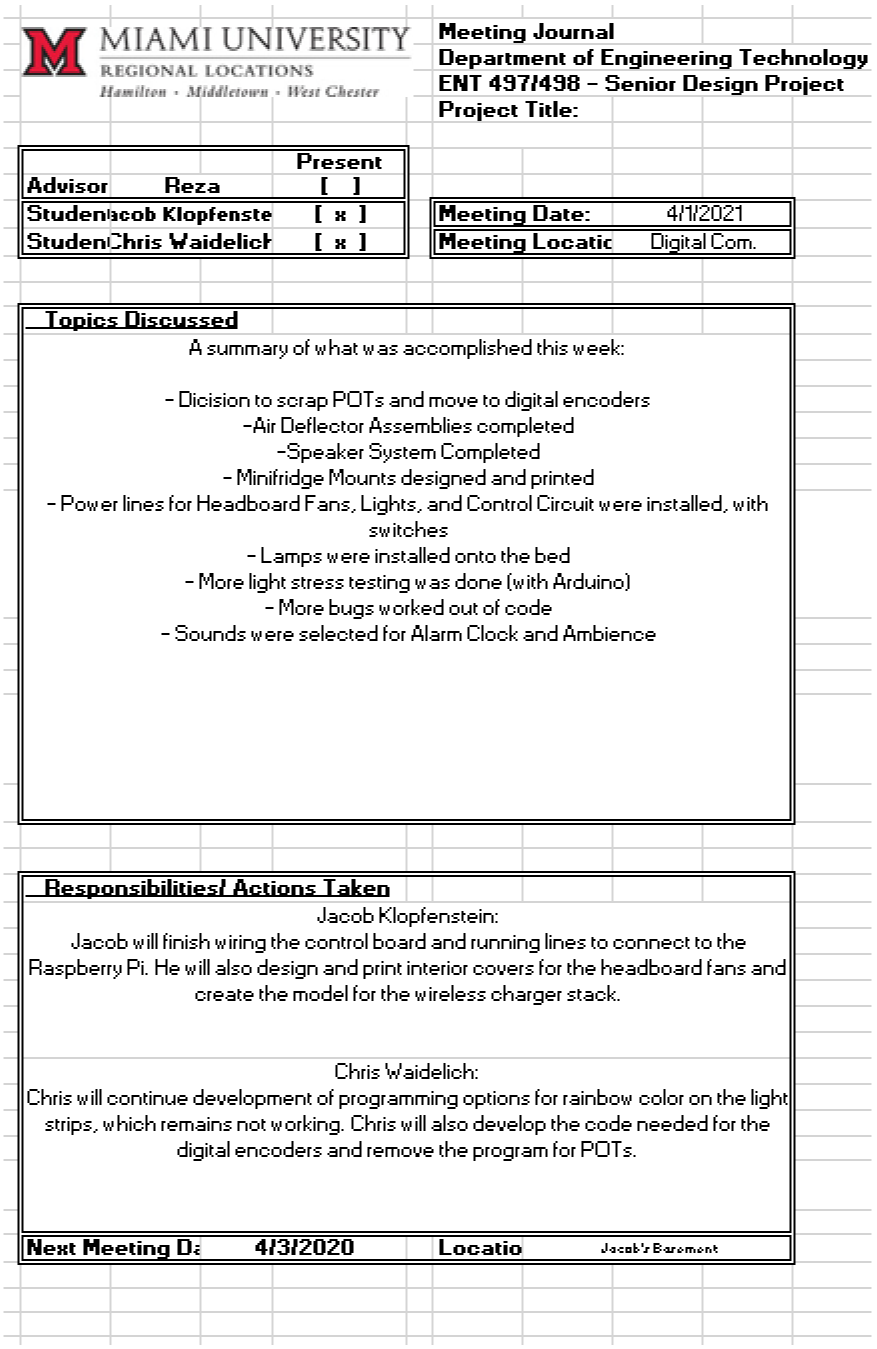
- Solution for light animations found and was working, but then stopped working
-Relays were connected and tested for the first time
-Wiring for relay and servo connections created
-After two light strips burned out, a new power rail bus was ordered and the way power is distributed to the lights was rethought
-A decision was made to seperate the power going to the lights from any other 5V power in the system to reduce problems with current surges and short circuits
-Final air deflector parts were printed and mounted.
-Headboard speaker mounts were designed, printed, and installed on the headboard

Responsibilities/ Actions Taken

Jacob Klopfenstein:
Jacob will finish wiring electronics, 3D printing parts, and hooking up all systems to the bed's raspberry pi in preparation for the main program installation. Jacob will also re-wire the power distribution rails once they arrive and transition everything over from the breadboard

Chris Waidelich:
Chris will continue development of programming options for rainbow color on the light strips, which was the dissapointment of the week after the previously working program failed for unknown reasons. Chris will also fix minor progam bugs that were discovered, as well as add in the program needed for the working servos. Finally, he will prepare the system for the main alarmclock and ambience files to be installed next meeting

Next Meeting Da	**4/1/2020**	**Locatio**	Jacob's Basement



MIAMI UNIVERSITY
REGIONAL LOCATIONS
Hamilton • Middletown • West Chester

Meeting Journal
Department of Engineering Technology
ENT 497/498 – Senior Design Project
Project Title:

		Present
Advisor	**Reza**	**[ ]**
Studen	**acob Klopfenste**	**[x]**
Studen	**Chris Waidelich**	**[x]**

Meeting Date:	4/1/2021
Meeting Locatic	Digital Com.

Topics Discussed

A summary of what was accomplished this week:

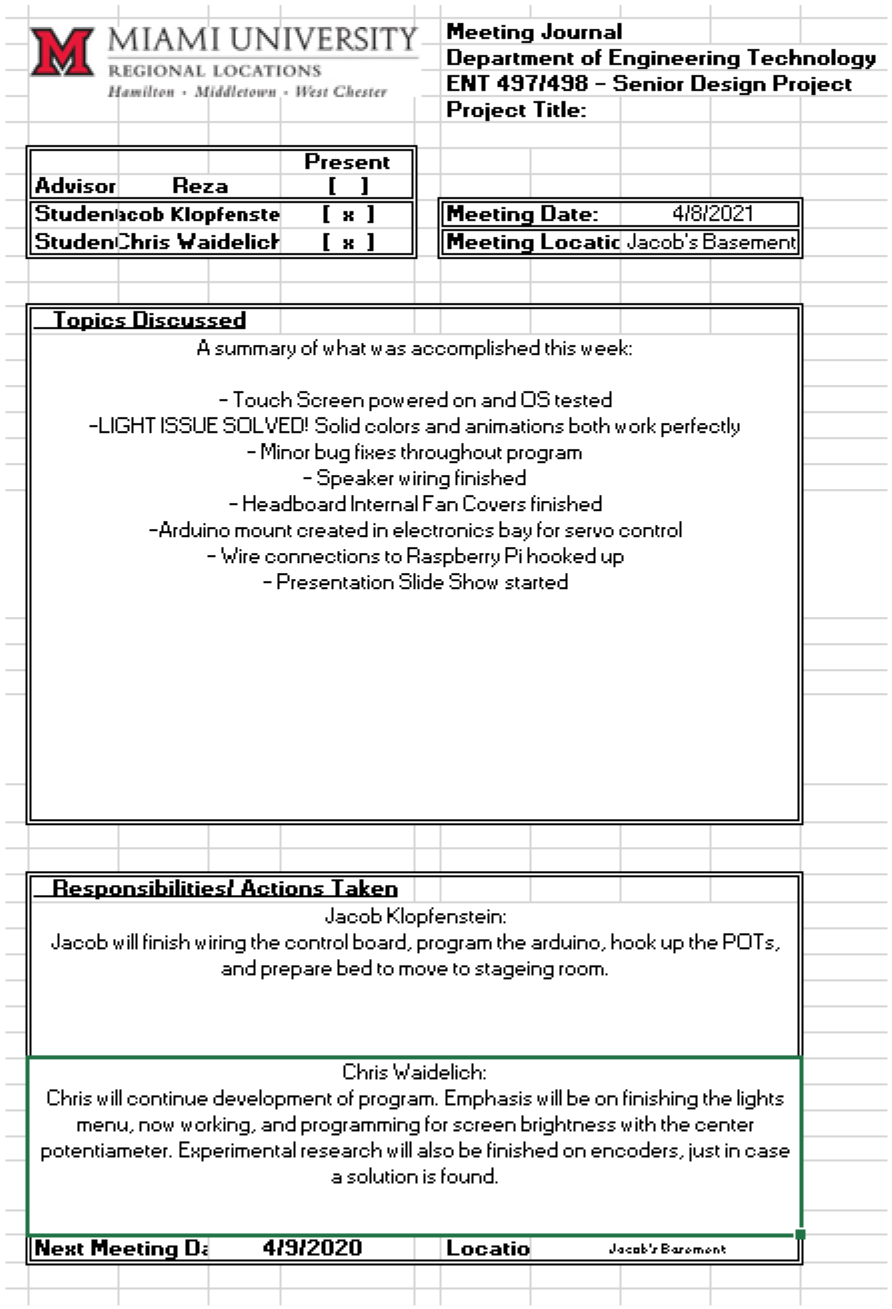
- Dicision to scrap POTs and move to digital encoders
-Air Deflector Assemblies completed
-Speaker System Completed
- Minifridge Mounts designed and printed
- Power lines for Headboard Fans, Lights, and Control Circuit were installed, with switches
- Lamps were installed onto the bed
- More light stress testing was done (with Arduino)
- More bugs worked out of code
- Sounds were selected for Alarm Clock and Ambience

Responsibilities/ Actions Taken

Jacob Klopfenstein:
Jacob will finish wiring the control board and running lines to connect to the Raspberry Pi. He will also design and print interior covers for the headboard fans and create the model for the wireless charger stack.

Chris Waidelich:
Chris will continue development of programming options for rainbow color on the light strips, which remains not working. Chris will also develop the code needed for the digital encoders and remove the program for POTs.

Next Meeting Da	**4/3/2020**	**Locatio**	Jacob's Basement



MIAMI UNIVERSITY
REGIONAL LOCATIONS
Hamilton · Middletown · West Chester

Meeting Journal
Department of Engineering Technology
ENT 497/498 – Senior Design Project
Project Title:

		Present
Advisor	**Reza**	**[ ]**
Studen	**Jacob Klopfenste**	**[x]**
Studen	**Chris Waidelich**	**[x]**

Meeting Date:	4/8/2021
Meeting Locatic	Jacob's Basement

Topics Discussed

A summary of what was accomplished this week:

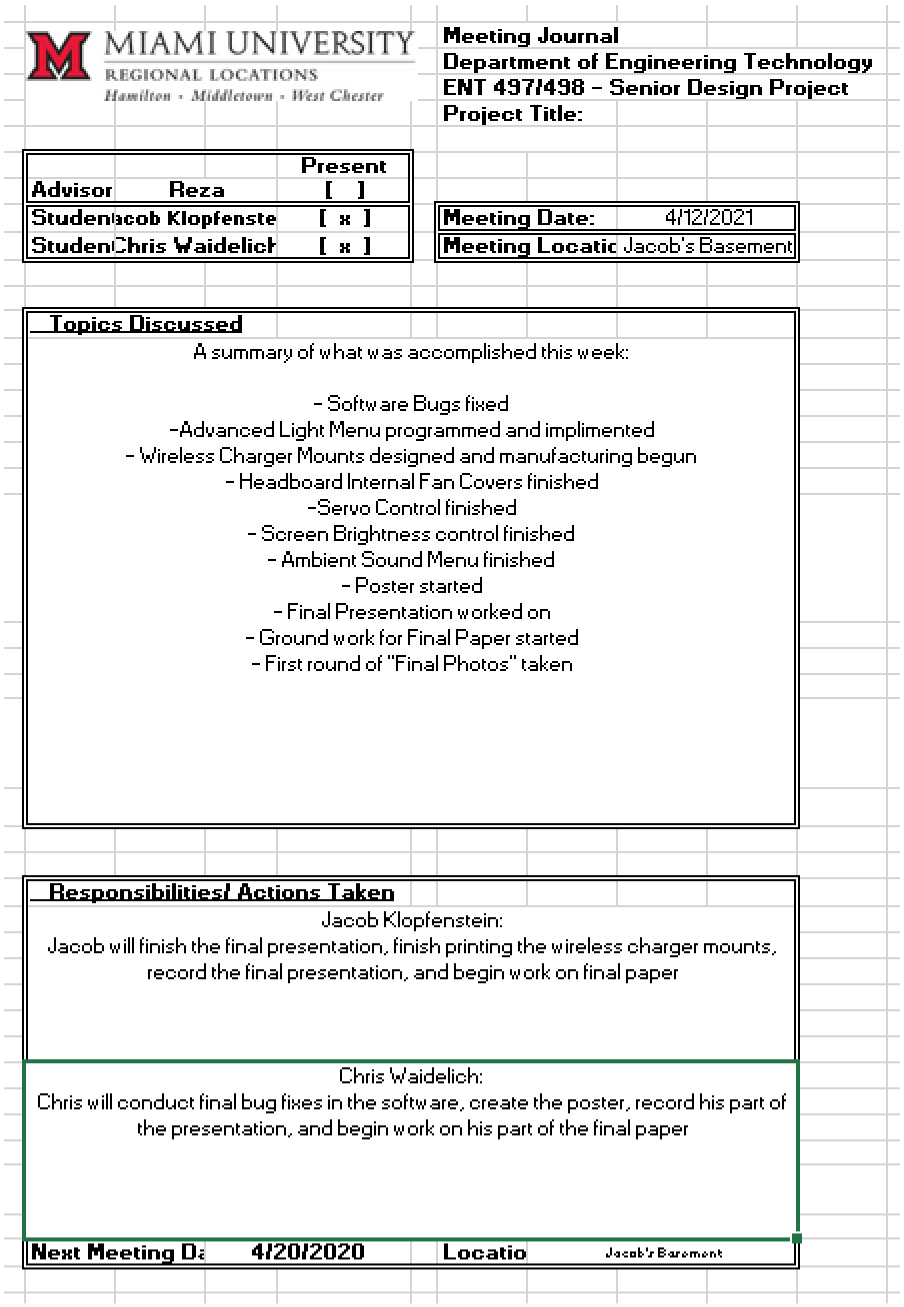
- Touch Screen powered on and OS tested
-LIGHT ISSUE SOLVED! Solid colors and animations both work perfectly
- Minor bug fixes throughout program
- Speaker wiring finished
- Headboard Internal Fan Covers finished
-Arduino mount created in electronics bay for servo control
- Wire connections to Raspberry Pi hooked up
- Presentation Slide Show started

Responsibilities/ Actions Taken

Jacob Klopfenstein:
Jacob will finish wiring the control board, program the arduino, hook up the POTs, and prepare bed to move to stageing room.

Chris Waidelich:
Chris will continue development of program. Emphasis will be on finishing the lights menu, now working, and programming for screen brightness with the center potentiameter. Experimental research will also be finished on encoders, just in case a solution is found.

Next Meeting Da	**4/9/2020**	**Locatio**	Jacob's Basement



MIAMI UNIVERSITY
REGIONAL LOCATIONS
Hamilton · Middletown · West Chester

Meeting Journal
Department of Engineering Technology
ENT 497/498 – Senior Design Project
Project Title:

		Present
Advisor	**Reza**	**[ ]**
Studen	**acob Klopfenste**	**[x]**
Studen	**Chris Waidelich**	**[x]**

Meeting Date:	4/12/2021
Meeting Locatic	Jacob's Basement

Topics Discussed

A summary of what was accomplished this week:

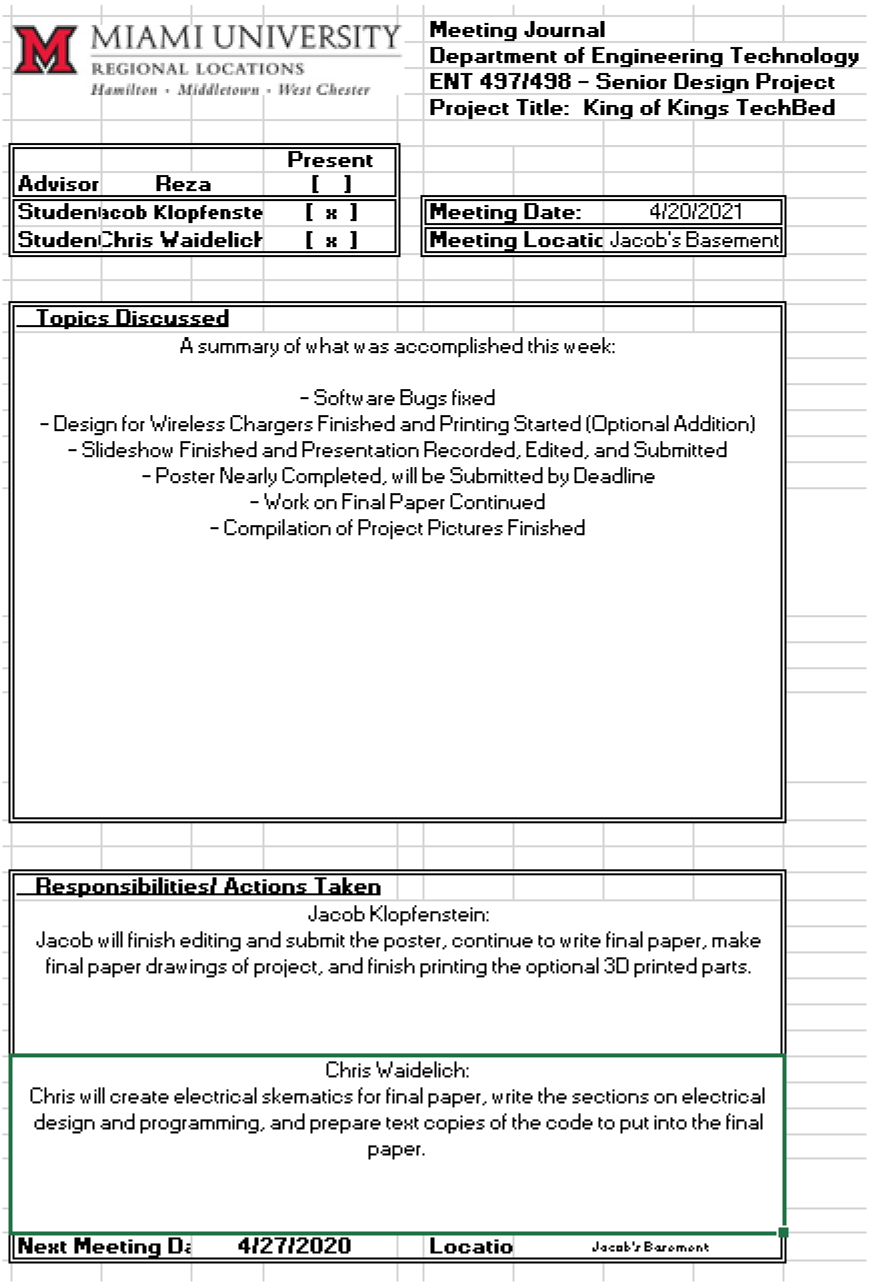
- Software Bugs fixed
-Advanced Light Menu programmed and implimented
- Wireless Charger Mounts designed and manufacturing begun
- Headboard Internal Fan Covers finished
-Servo Control finished
- Screen Brightness control finished
- Ambient Sound Menu finished
- Poster started
- Final Presentation worked on
- Ground work for Final Paper started
- First round of "Final Photos" taken

Responsibilities/ Actions Taken

Jacob Klopfenstein:
Jacob will finish the final presentation, finish printing the wireless charger mounts, record the final presentation, and begin work on final paper

Chris Waidelich:
Chris will conduct final bug fixes in the software, create the poster, record his part of the presentation, and begin work on his part of the final paper

Next Meeting Da	**4/20/2020**	**Locatio**	Jacob's Basement



MIAMI UNIVERSITY
REGIONAL LOCATIONS
Hamilton · Middletown · West Chester

Meeting Journal
Department of Engineering Technology
ENT 497/498 – Senior Design Project
Project Title: King of Kings TechBed

		Present
Advisor	**Reza**	**[ ]**
Studen	**acob Klopfenste**	**[x]**
Studen	**Chris Waidelich**	**[x]**

Meeting Date:	4/20/2021
Meeting Locatic	Jacob's Basement

Topics Discussed

A summary of what was accomplished this week:

- Software Bugs fixed
- Design for Wireless Chargers Finished and Printing Started (Optional Addition)
- Slideshow Finished and Presentation Recorded, Edited, and Submitted
- Poster Nearly Completed, will be Submitted by Deadline
- Work on Final Paper Continued
- Compilation of Project Pictures Finished

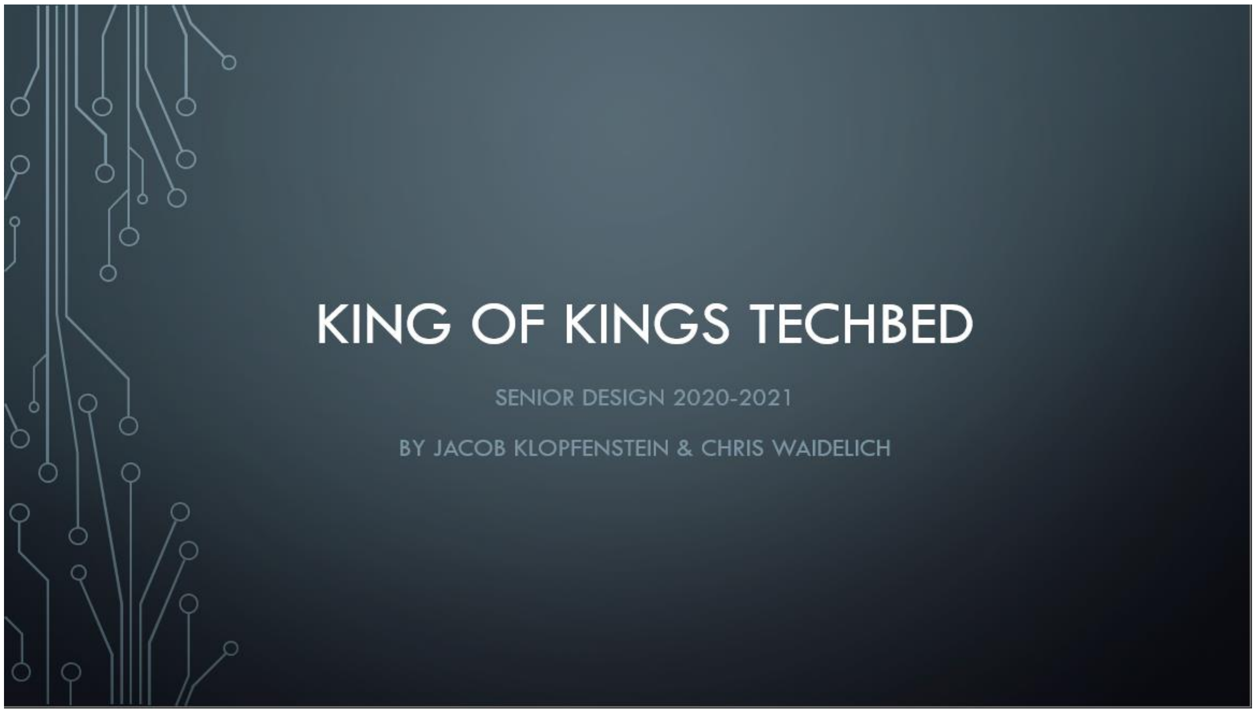
Responsibilities/ Actions Taken

Jacob Klopfenstein:
Jacob will finish editing and submit the poster, continue to write final paper, make final paper drawings of project, and finish printing the optional 3D printed parts.

Chris Waidelich:
Chris will create electrical skematics for final paper, write the sections on electrical design and programming, and prepare text copies of the code to put into the final paper.

Next Meeting Da	**4/27/2020**	**Locatio**	Jacob's Basement

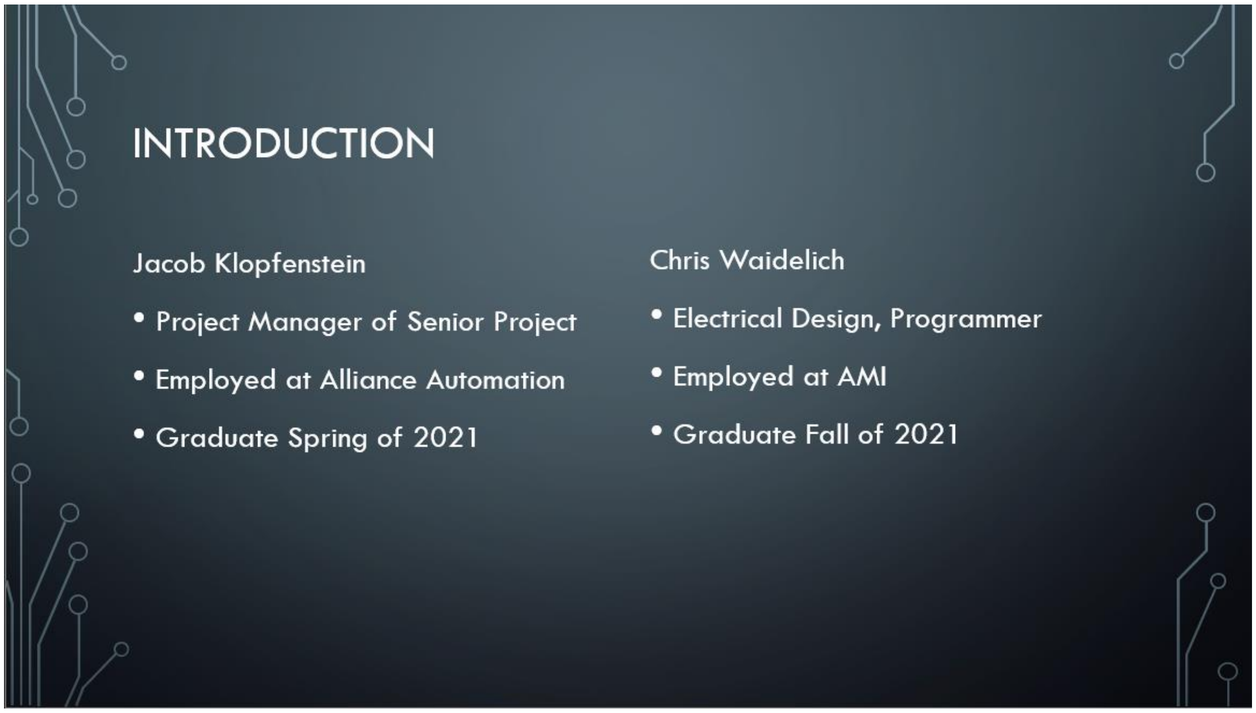
Appendix H: Midterm Presentation



KING OF KINGS TECHBED

SENIOR DESIGN 2020-2021

BY JACOB KLOPFENSTEIN & CHRIS WAIDELICH



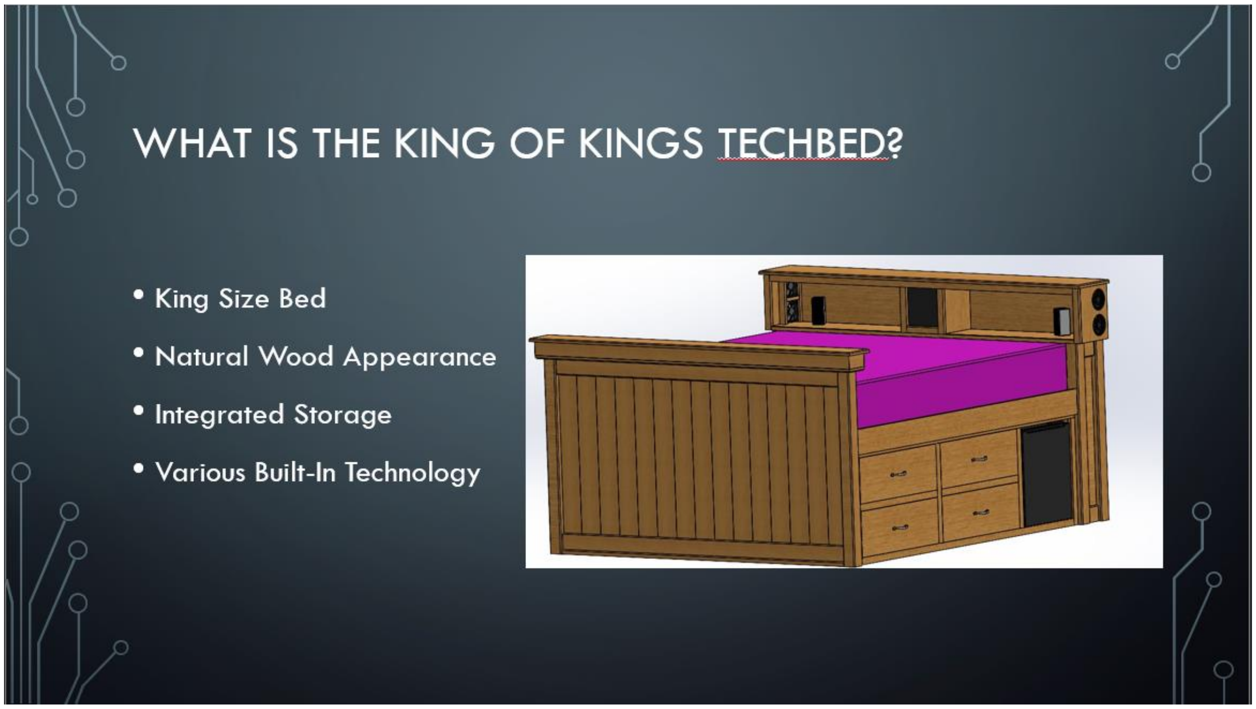
INTRODUCTION

Jacob Klopfenstein

- Project Manager of Senior Project
- Employed at Alliance Automation
- Graduate Spring of 2021

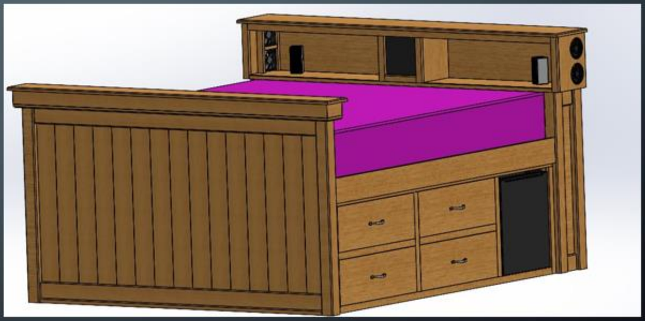
Chris Waidelich

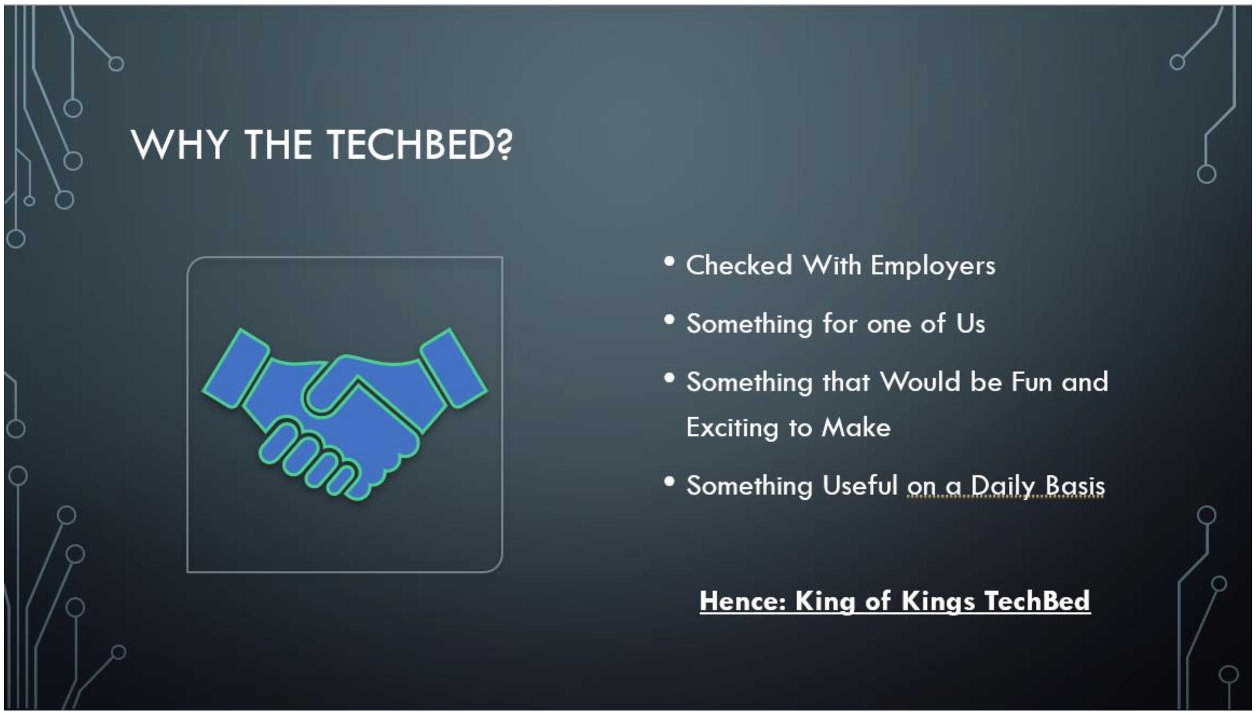
- Electrical Design, Programmer
- Employed at AMI
- Graduate Fall of 2021



WHAT IS THE KING OF KINGS TECHBED?

- King Size Bed
- Natural Wood Appearance
- Integrated Storage
- Various Built-In Technology

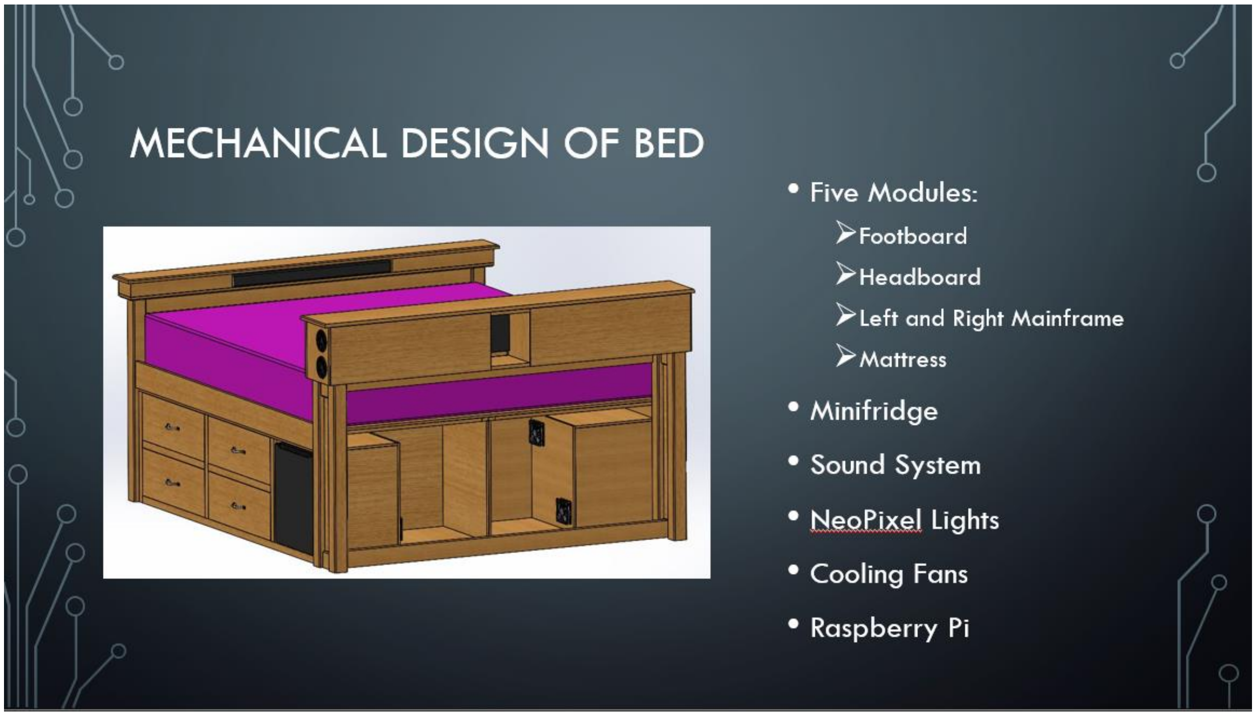




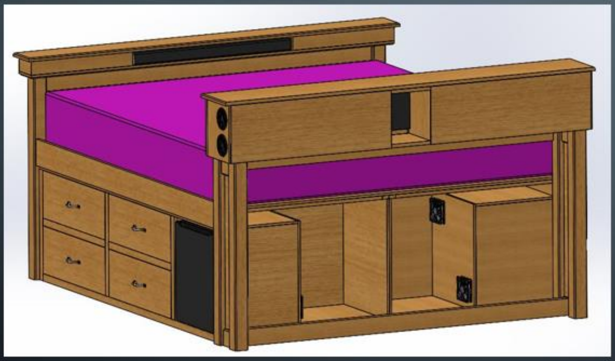
WHY THE TECHBED?

- Checked With Employers
- Something for one of Us
- Something that Would be Fun and Exciting to Make
- Something Useful on a Daily Basis

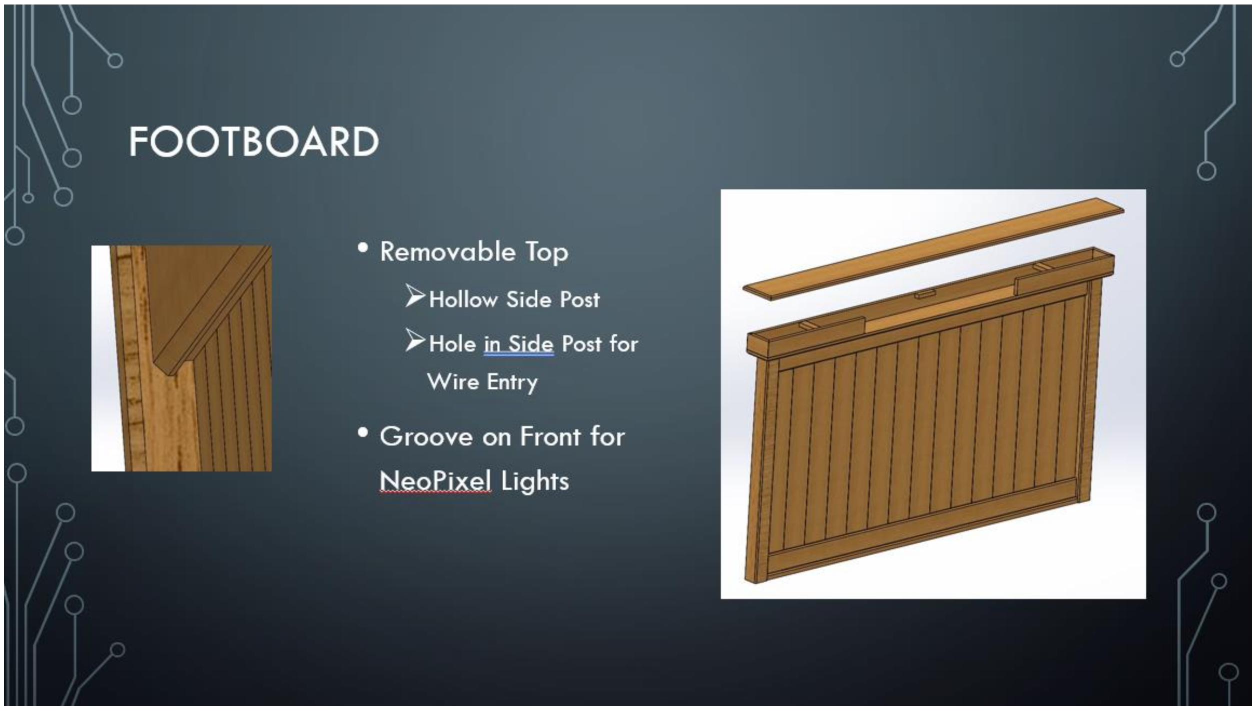
Hence: King of Kings TechBed



MECHANICAL DESIGN OF BED

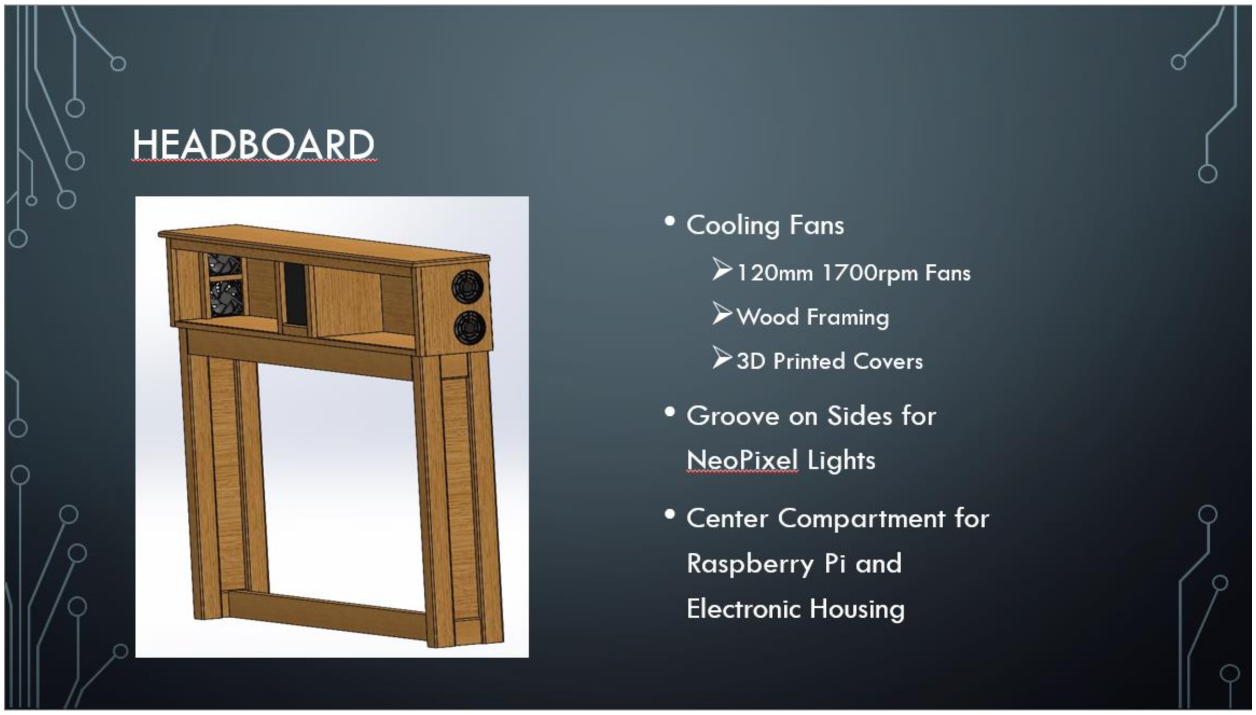


- Five Modules:
  - Footboard
  - Headboard
  - Left and Right Mainframe
  - Mattress
- Minifridge
- Sound System
- NeoPixel Lights
- Cooling Fans
- Raspberry Pi

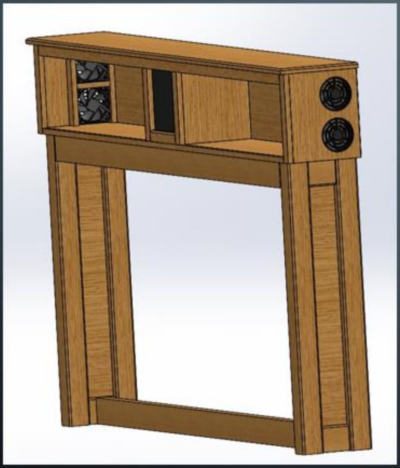


FOOTBOARD

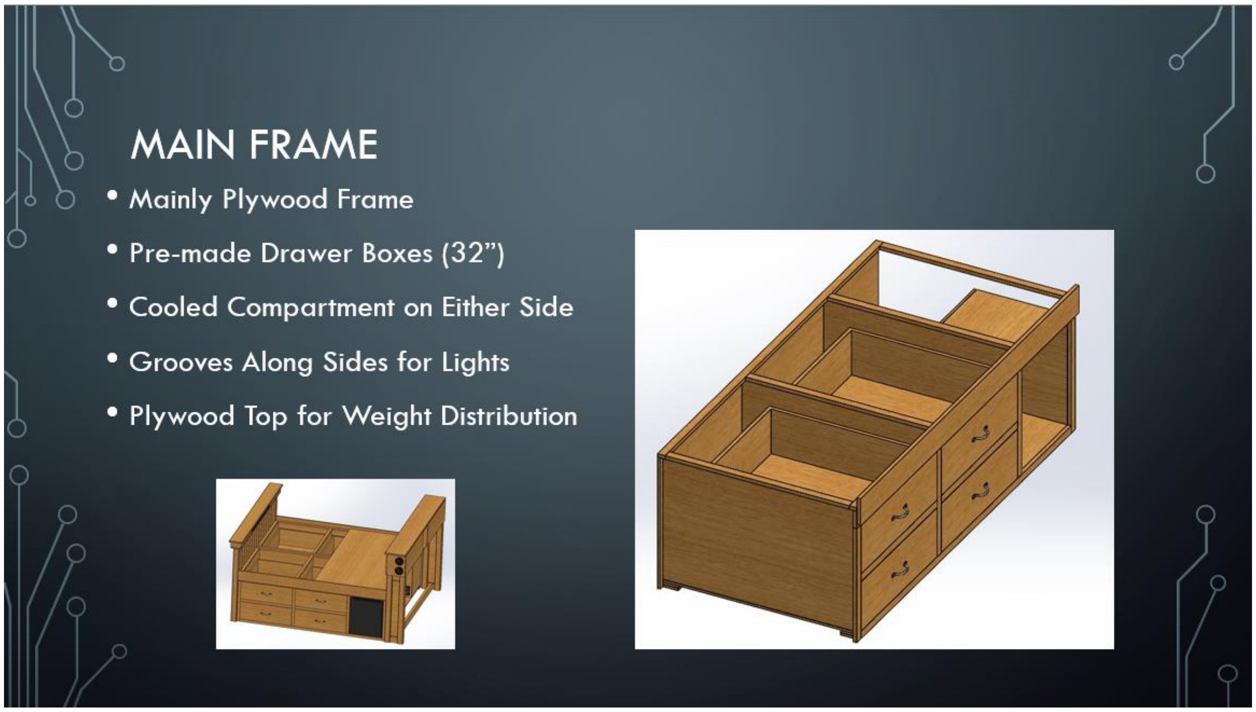
- Removable Top
  - Hollow Side Post
  - Hole in Side Post for Wire Entry
- Groove on Front for NeoPixel Lights



HEADBOARD

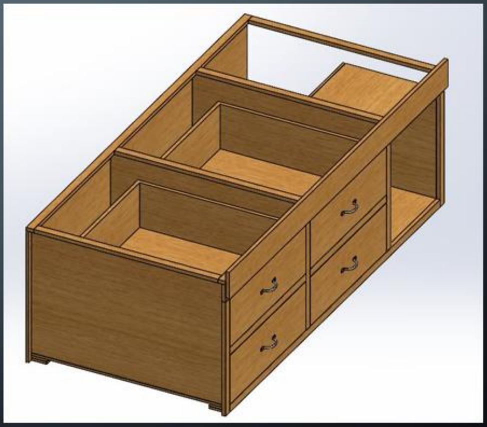


- Cooling Fans
  - ➢120mm 1700rpm Fans
  - ➢Wood Framing
  - ➢3D Printed Covers
- Groove on Sides for NeoPixel Lights
- Center Compartment for Raspberry Pi and Electronic Housing

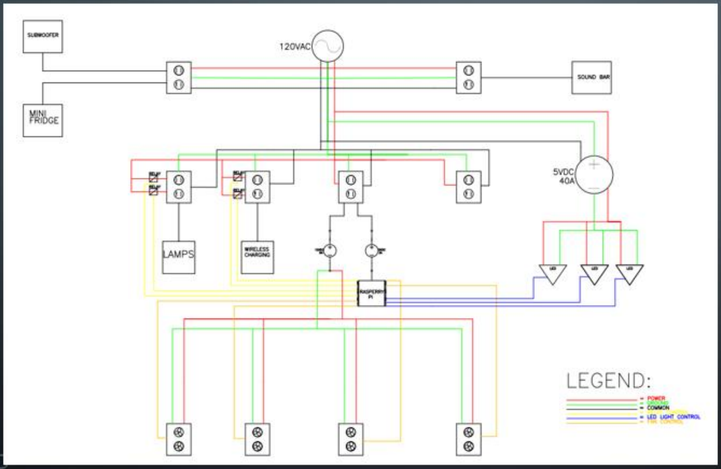


MAIN FRAME

- Mainly Plywood Frame
- Pre-made Drawer Boxes (32”)
- Cooled Compartment on Either Side
- Grooves Along Sides for Lights
- Plywood Top for Weight Distribution



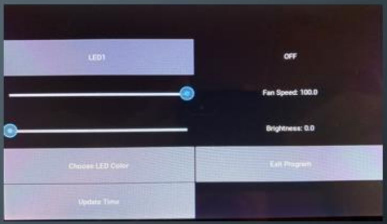
ELECTRICAL DESIGN



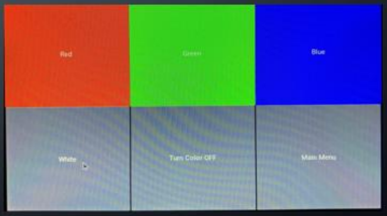
- Single Wall Plug-in
- Four Relay Controlled Plugs
- Four Sets of Fans (Two Sets on Headboard, One set per Cooled Compartment
- Raspberry Pi

GRAPHICAL INTERFACE DESIGN

- Language: **Python**
- Graphical Interface Library: **Kivy**
  - ➢Buttons
  - ➢Sliders
  - ➢Number Readout
  - ➢Various Fonts, Colors, Sizes
  - ➢Main Menu and Sub-Screens



Test Program Progresses

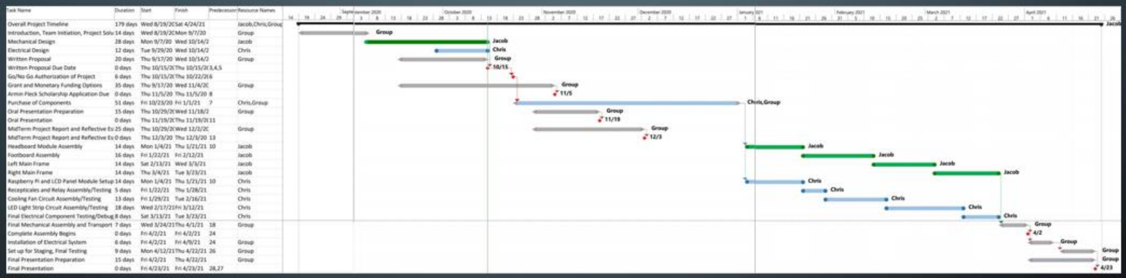


PROJECT BUDGET

- Technology: $2,685
- Electrical Supplies: $250
- Hardware/Fasteners: $415
- Wood/Plywood: $2,000

- Total Budget: ≈ $5,350

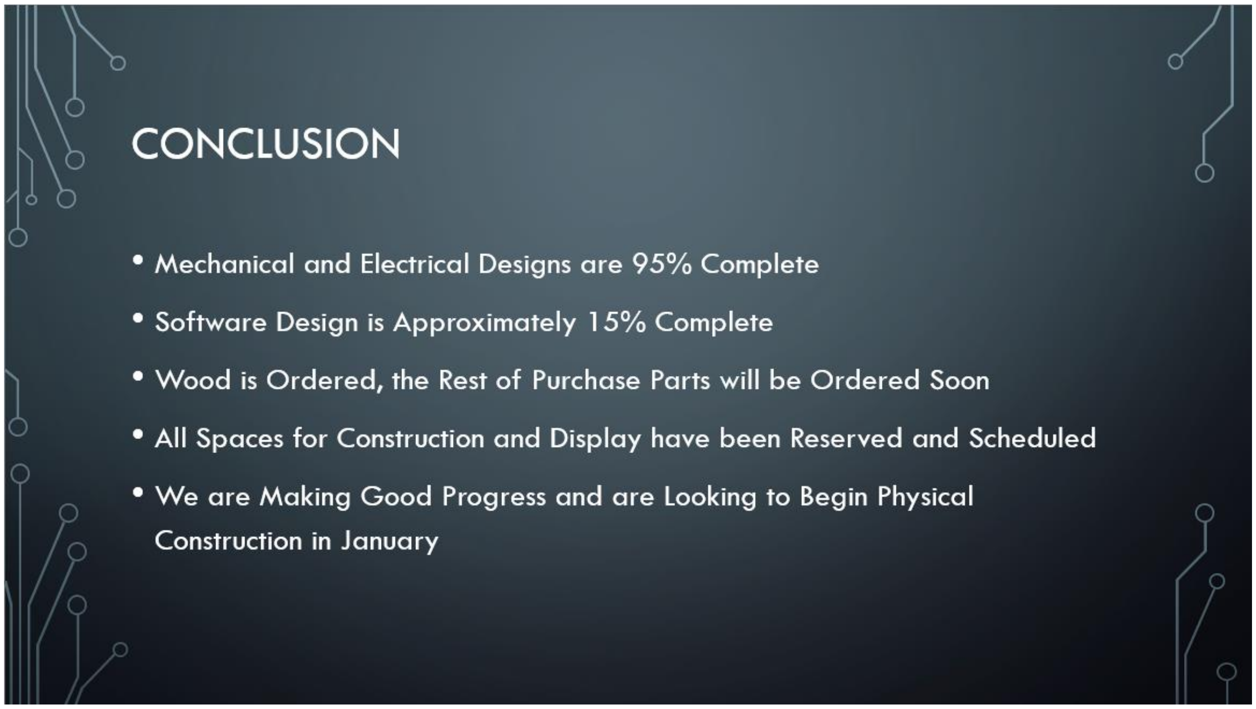
GANNET CHART



Task Name	Duration	Start	Finish	Predecessors	Resource Names
Overall Project Timeline	179 days	Wed 8/19/2C	Sat 4/24/21		Jacob,Chris,Group
Introduction, Team Initiation, Project Solu	14 days	Wed 8/19/2C	Mon 9/7/20		Group
Mechanical Design	28 days	Mon 9/7/20	Wed 10/14/2		Jacob
Electrical Design	12 days	Tue 9/29/20	Wed 10/14/2		Chris
Written Proposal	20 days	Thu 9/17/20	Wed 10/14/2		Group
Written Proposal Due Date	0 days	Thu 10/15/2C	Thu 10/15/2C	3,4,5	
Go/No Go Authorization of Project	6 days	Thu 10/15/2C	Thu 10/22/2C	6	
Grant and Monetary Funding Options	35 days	Thu 9/17/20	Wed 11/4/2C		Group
Armin Fleck Scholarship Application Due	0 days	Thu 11/5/20	Thu 11/5/20	8	
Purchase of Components	51 days	Fri 10/23/20	Fri 1/1/21	7	Chris,Group
Oral Presentation Preparation	15 days	Thu 10/29/2C	Wed 11/18/2		Group
Oral Presentation	0 days	Thu 11/19/2C	Thu 11/19/2C	11	
MidTerm Project Report and Reflective Es	25 days	Thu 10/29/2C	Wed 12/2/2C		Group
MidTerm Project Report and Reflective Es	0 days	Thu 12/3/20	Thu 12/3/20	13	
Headboard Module Assembly	14 days	Mon 1/4/21	Thu 1/21/21	10	Jacob
Footboard Assembly	16 days	Fri 1/22/21	Fri 2/12/21		Jacob
Left Main Frame	14 days	Sat 2/13/21	Wed 3/3/21		Jacob
Right Main Frame	14 days	Thu 3/4/21	Tue 3/23/21		Jacob
Raspberry Pi and LCD Panel Module Setup	14 days	Mon 1/4/21	Thu 1/21/21	10	Chris
Recepticales and Relay Assembly/Testing	5 days	Fri 1/22/21	Thu 1/28/21		Chris
Cooling Fan Circuit Assembly/Testing	13 days	Fri 1/29/21	Tue 2/16/21		Chris
LED Light Strip Circuit Assembly/Testing	18 days	Wed 2/17/21	Fri 3/12/21		Chris
Final Electrical Component Testing/Debug	8 days	Sat 3/13/21	Tue 3/23/21		Chris
Final Mechanical Assembly and Transport	7 days	Wed 3/24/21	Thu 4/1/21	18	Group
Complete Assembly Begins	0 days	Fri 4/2/21	Fri 4/2/21	24	
Installation of Electrical System	6 days	Fri 4/2/21	Fri 4/9/21	24	Group
Set up for Staging, Final Testing	9 days	Mon 4/12/21	Thu 4/22/21	26	Group
Final Presentation Preparation	15 days	Fri 4/2/21	Thu 4/22/21		Group
Final Presentation	0 days	Fri 4/23/21	Fri 4/23/21	28,27	

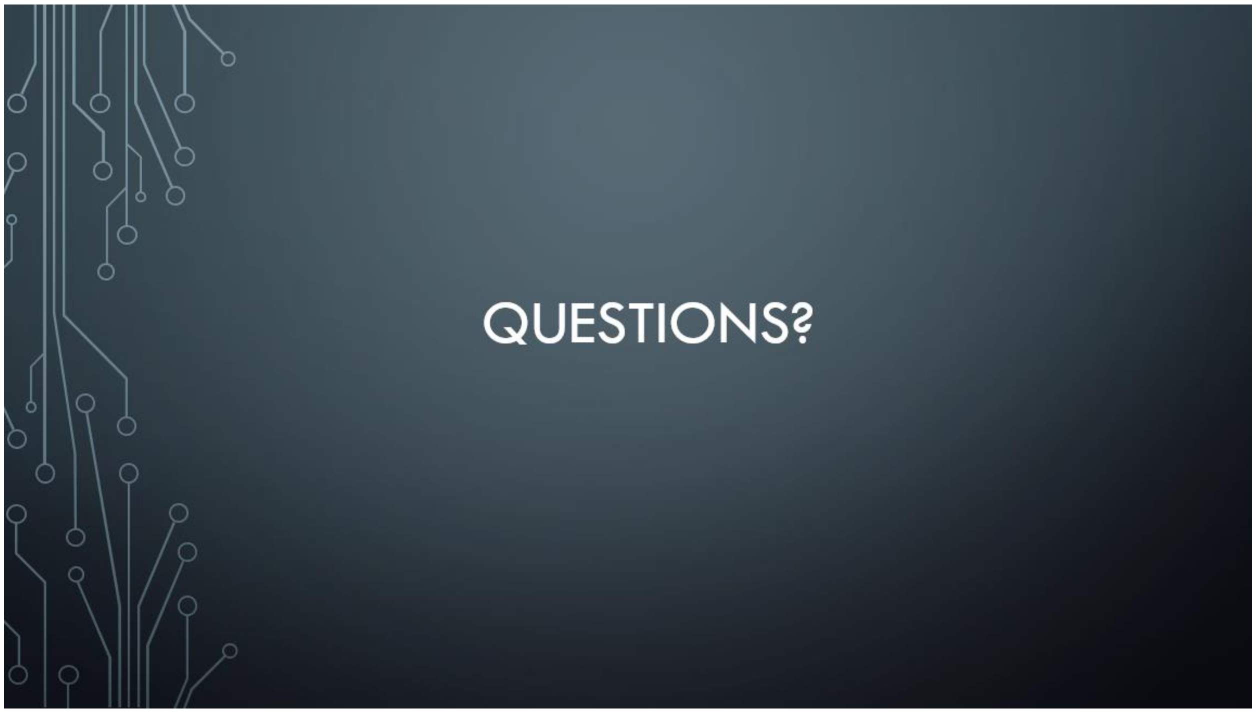
Major Deadlines:

Midterm Report: 12/3/2020 - Electrical Installation: 4/2/2021 - Final Presentation: 4/23/2021



CONCLUSION

- Mechanical and Electrical Designs are 95% Complete
- Software Design is Approximately 15% Complete
- Wood is Ordered, the Rest of Purchase Parts will be Ordered Soon
- All Spaces for Construction and Display have been Reserved and Scheduled
- We are Making Good Progress and are Looking to Begin Physical Construction in January



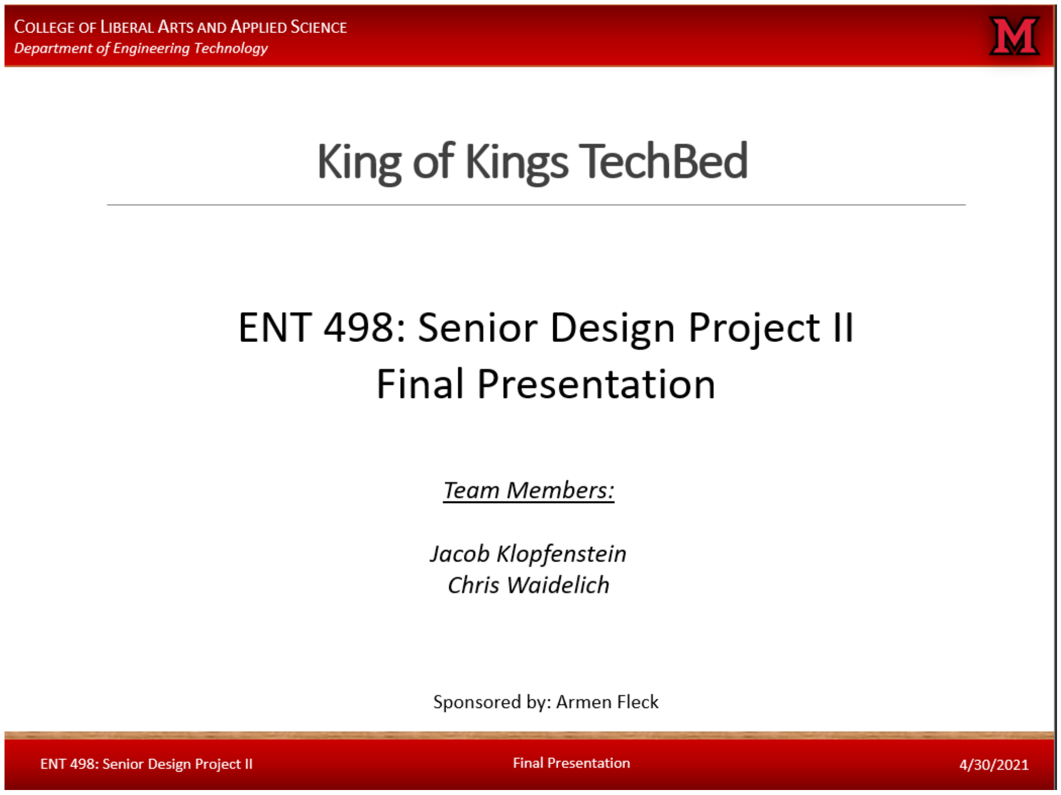
QUESTIONS?

Appendix I: Senior Design 2 Minute Presentation Script

Hello, my name is Jacob Klopfenstein, and along with my partner, Chris Waidelich, we designed and built our project “The King of Kings TechBed”. This is a King Bed meant to be the KING of all king beds, made of white pine to keep a natural wooden appearance while integrating various modern technologies that everyone would love to have on their beds. These technologies include a full surround sound system and subwoofer, RGBW lights built directly into the bedframe, cooling fans in the headboard capable of moving over 400 cubic feet of air per minute, servo controlled air deflectors to direct the air, computer controlled wireless chargers, relay controlled headboard lamps, and of course, a minifridge. All these technology features, save the minifridge itself, are controlled by a raspberry Pi 4B connected to a 7” touch screen with a custom designed user interface. Other important features of the bed include a total of 8 four and a half cubic feet sized drawers under the bed to make efficient use of floor space, and the ability for the bed to come apart into different modules for ease of transportation. The headboard also supports a projector mounted in the center of the headboard to project on the opposite wall for a full cinematic experience. The bedframe is capable of fitting either a regular king or a California King size mattress, depending on which size the user prefers.

And Finally, The touch screen interface utilizes a sleek appearance and calming color palette that matches any bedroom and that any user can easily use and enjoy. Together, me and Chris have redefined how a bedframe should look and function in the new technology age. A big thank you to my colleagues at Miami University for their assistance in the planning and execution of this project, Armin Fleck Scholarship for financial aid towards the project, and to all of you for listening to our presentation on “the King of Kings TechBed”, available in stores nowhere near you.

Appendix J: Final Presentation


COLLEGE OF LIBERAL ARTS AND APPLIED SCIENCE
Department of Engineering Technology

King of Kings TechBed

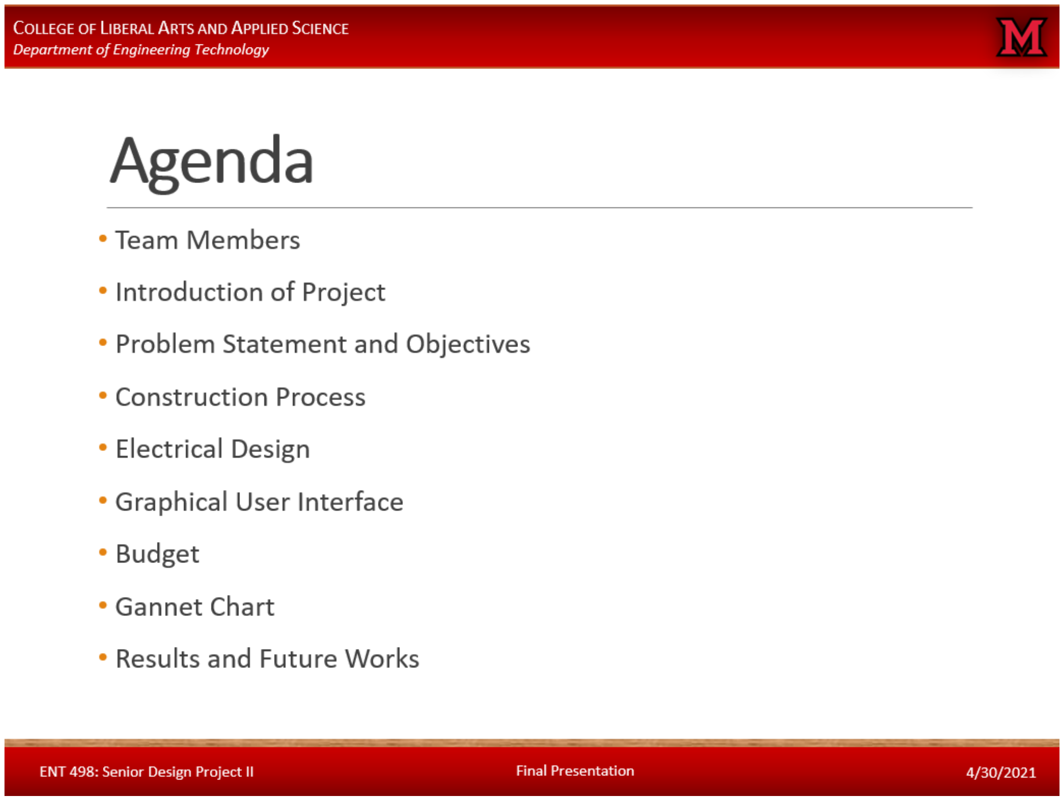
ENT 498: Senior Design Project II
Final Presentation

Team Members:

Jacob Klopfenstein
Chris Waidelich

Sponsored by: Armen Fleck

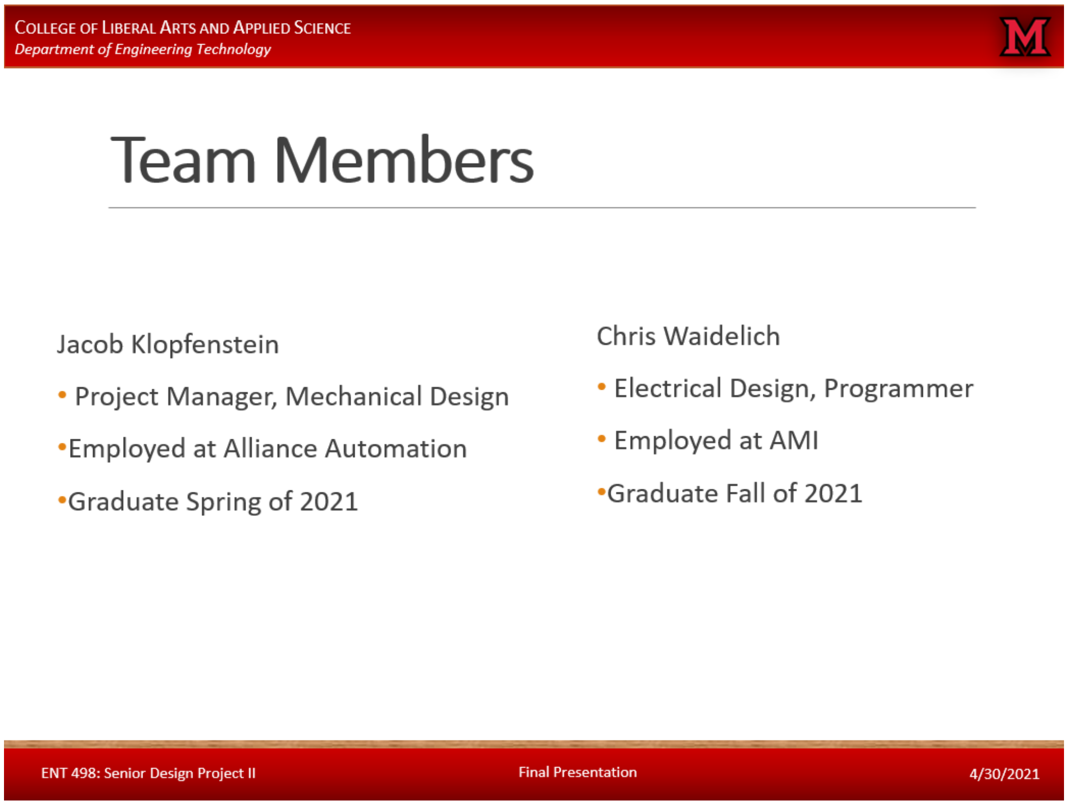
ENT 498: Senior Design Project II Final Presentation 4/30/2021


COLLEGE OF LIBERAL ARTS AND APPLIED SCIENCE
Department of Engineering Technology

Agenda

- Team Members
- Introduction of Project
- Problem Statement and Objectives
- Construction Process
- Electrical Design
- Graphical User Interface
- Budget
- Gannet Chart
- Results and Future Works

ENT 498: Senior Design Project II Final Presentation 4/30/2021



College of Liberal Arts and Applied Science
Department of Engineering Technology

Team Members

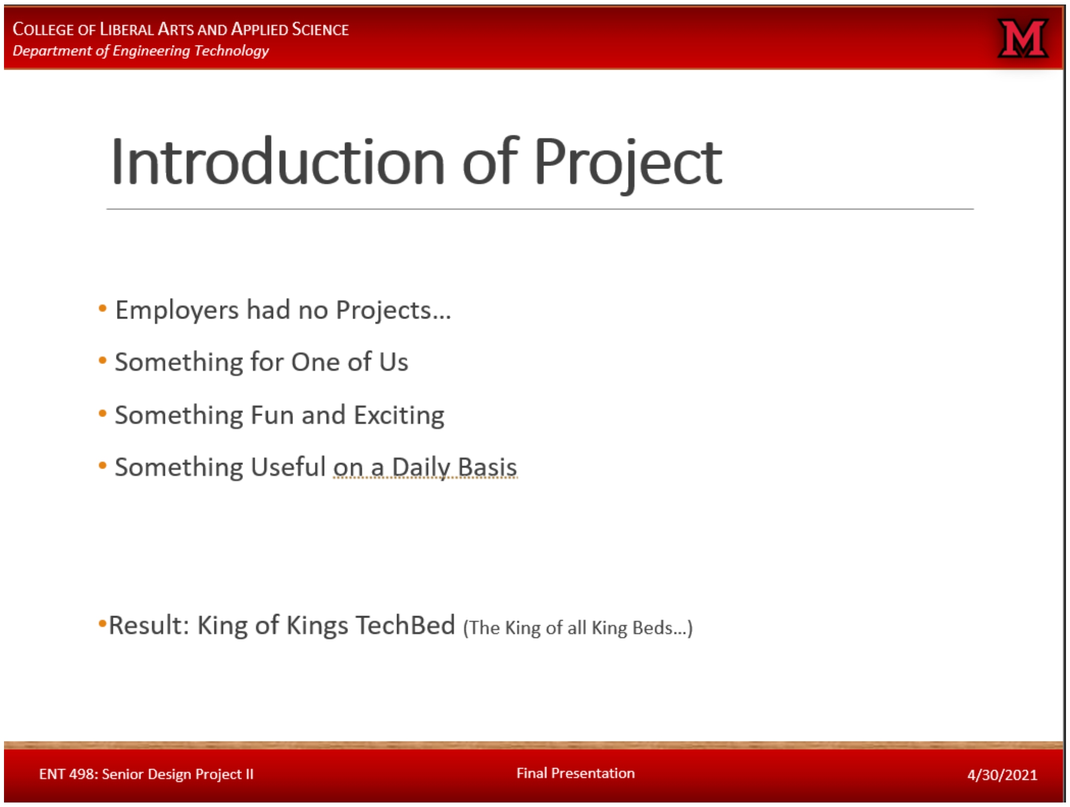
Jacob Klopfenstein

- Project Manager, Mechanical Design
- Employed at Alliance Automation
- Graduate Spring of 2021

Chris Waidelich

- Electrical Design, Programmer
- Employed at AMI
- Graduate Fall of 2021

ENT 498: Senior Design Project II Final Presentation 4/30/2021



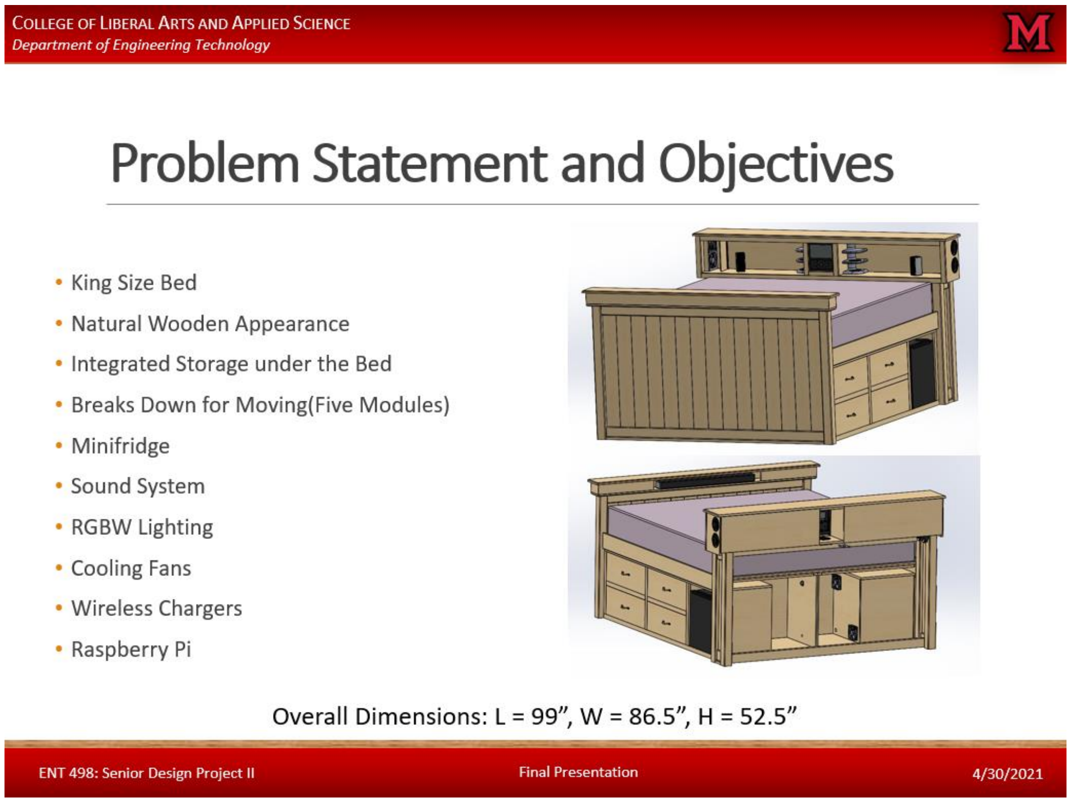
College of Liberal Arts and Applied Science
Department of Engineering Technology

Introduction of Project

- Employers had no Projects…
- Something for One of Us
- Something Fun and Exciting
- Something Useful on a Daily Basis

- Result: King of Kings TechBed (The King of all King Beds…)

ENT 498: Senior Design Project II Final Presentation 4/30/2021



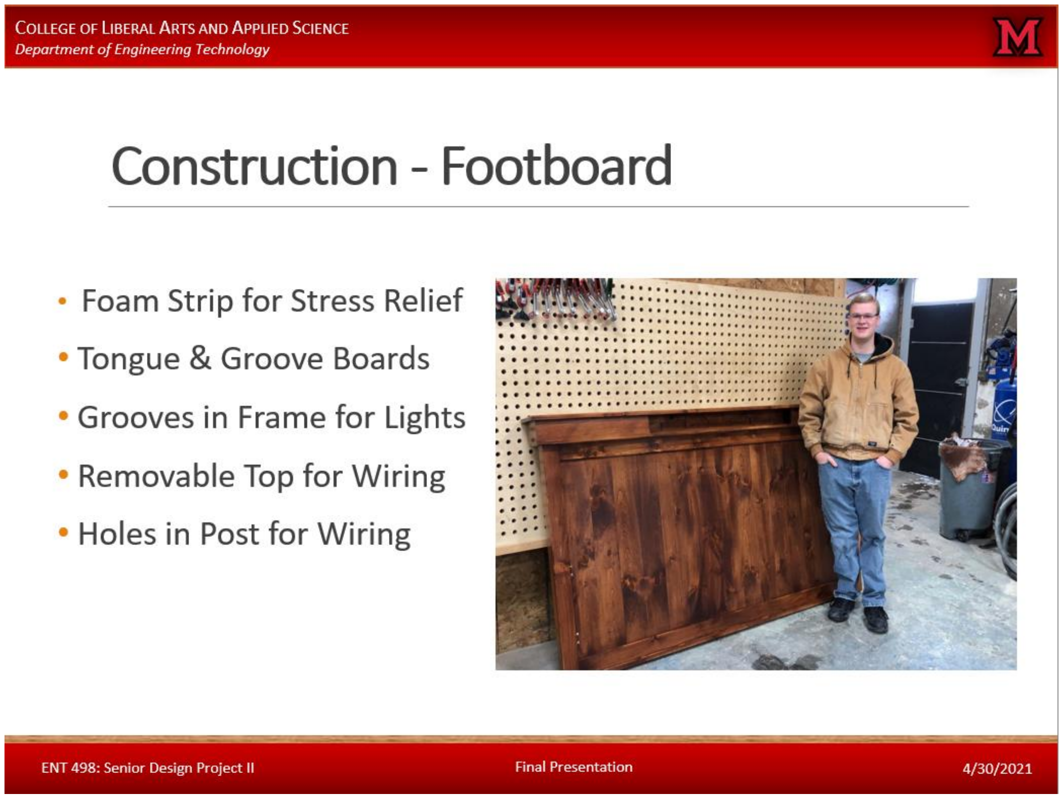
College of Liberal Arts and Applied Science
Department of Engineering Technology

Problem Statement and Objectives

- King Size Bed
- Natural Wooden Appearance
- Integrated Storage under the Bed
- Breaks Down for Moving(Five Modules)
- Minifridge
- Sound System
- RGBW Lighting
- Cooling Fans
- Wireless Chargers
- Raspberry Pi

Overall Dimensions: L = 99”, W = 86.5”, H = 52.5”

ENT 498: Senior Design Project II Final Presentation 4/30/2021

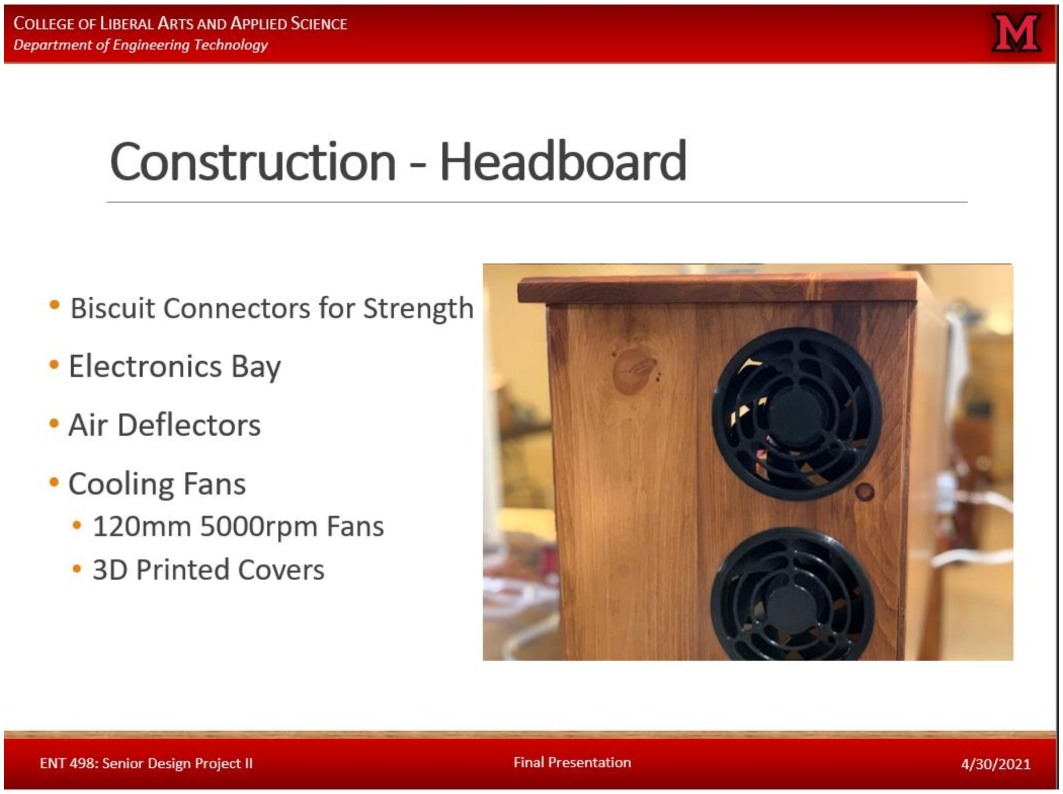


College of Liberal Arts and Applied Science
Department of Engineering Technology

Construction - Footboard

- Foam Strip for Stress Relief
- Tongue & Groove Boards
- Grooves in Frame for Lights
- Removable Top for Wiring
- Holes in Post for Wiring

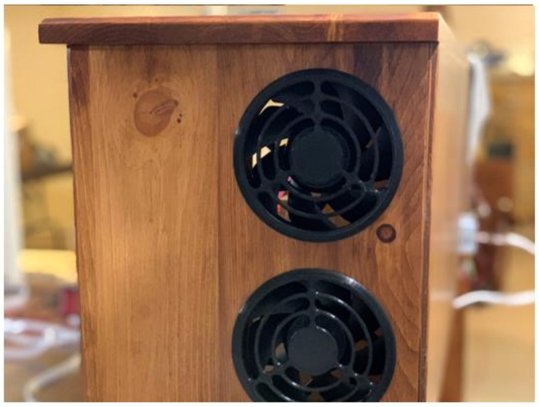
ENT 498: Senior Design Project II Final Presentation 4/30/2021



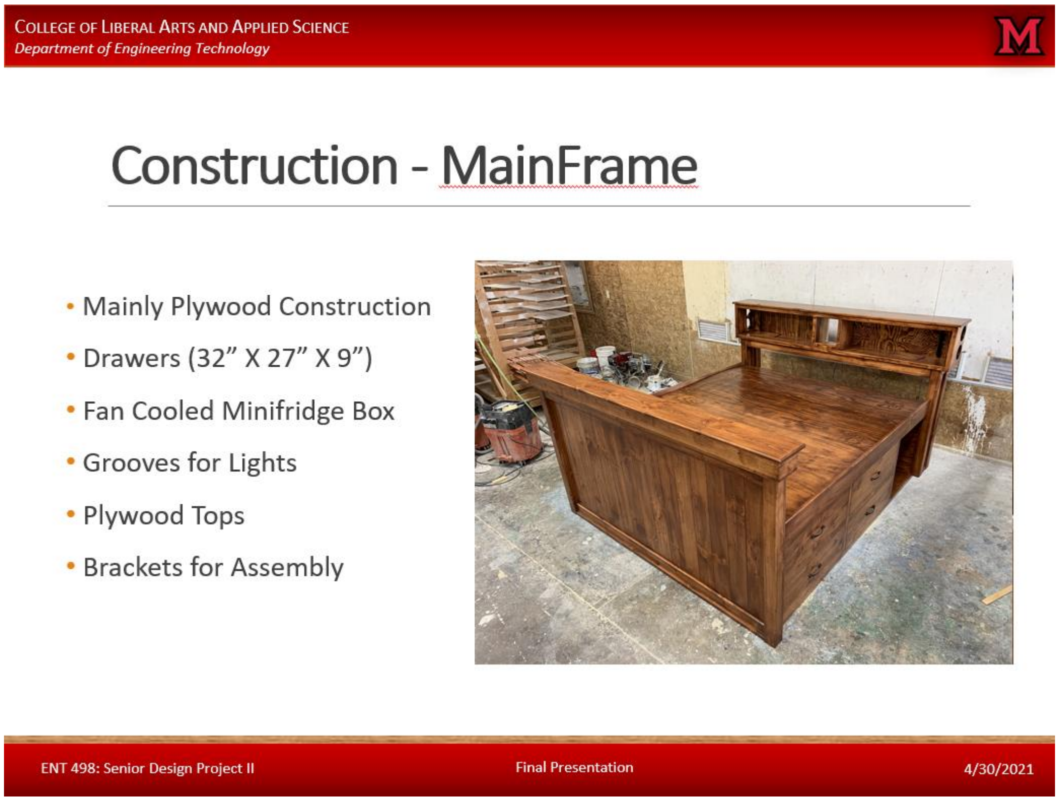
COLLEGE OF LIBERAL ARTS AND APPLIED SCIENCE
Department of Engineering Technology

Construction - Headboard

- Biscuit Connectors for Strength
- Electronics Bay
- Air Deflectors
- Cooling Fans
  - 120mm 5000rpm Fans
  - 3D Printed Covers



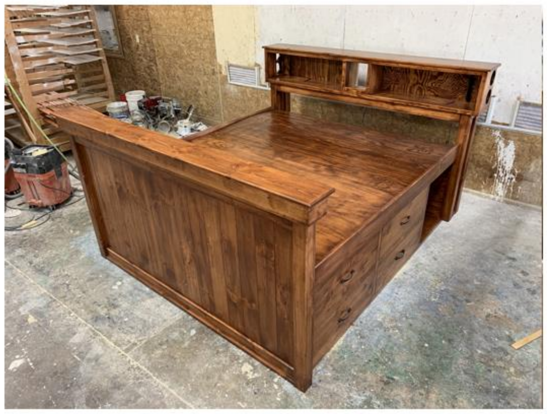
ENT 498: Senior Design Project II Final Presentation 4/30/2021



COLLEGE OF LIBERAL ARTS AND APPLIED SCIENCE
Department of Engineering Technology

Construction - MainFrame

- Mainly Plywood Construction
- Drawers (32” X 27” X 9”)
- Fan Cooled Minifridge Box
- Grooves for Lights
- Plywood Tops
- Brackets for Assembly

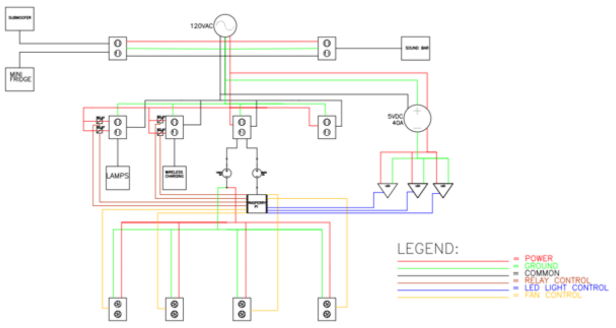


ENT 498: Senior Design Project II Final Presentation 4/30/2021

College of Liberal Arts and Applied Science
Department of Engineering Technology

Electrical Design

- Single Wall Plug-in
- Four Relay-Controlled Outlets
- Four Sets of Fans (Two Headboard, Two Minifridge Boxes)
- Raspberry Pi on Separate Power Supply

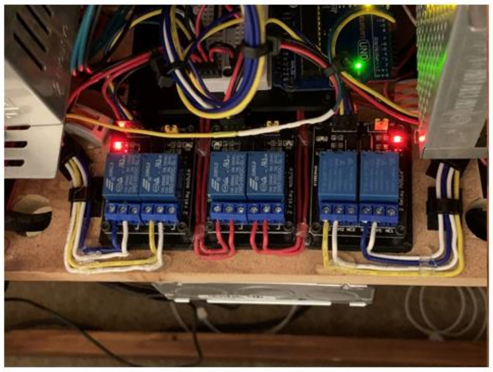


ENT 498: Senior Design Project II Final Presentation 4/30/2021

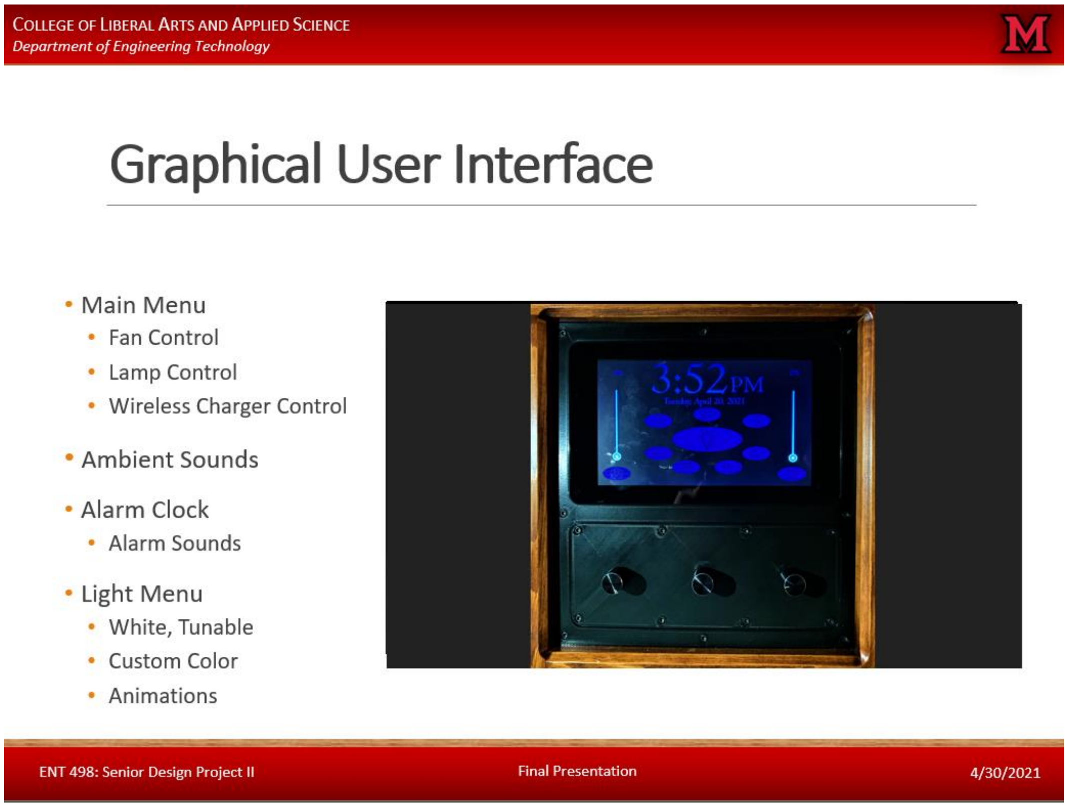
College of Liberal Arts and Applied Science
Department of Engineering Technology

Electrical Design - Wiring

- Removed Minifridge Box Fan Control to Physical Switch
- Added Arduino Uno for Servo Control
- Added Relays to Headboard Fans
- The Three Separate LED Strips Became A Single Strip



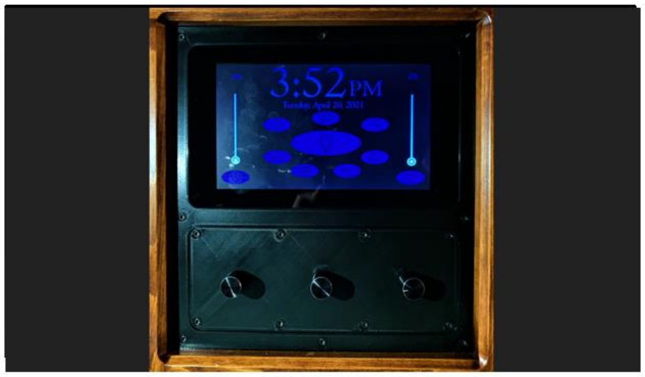
ENT 498: Senior Design Project II Final Presentation 4/30/2021



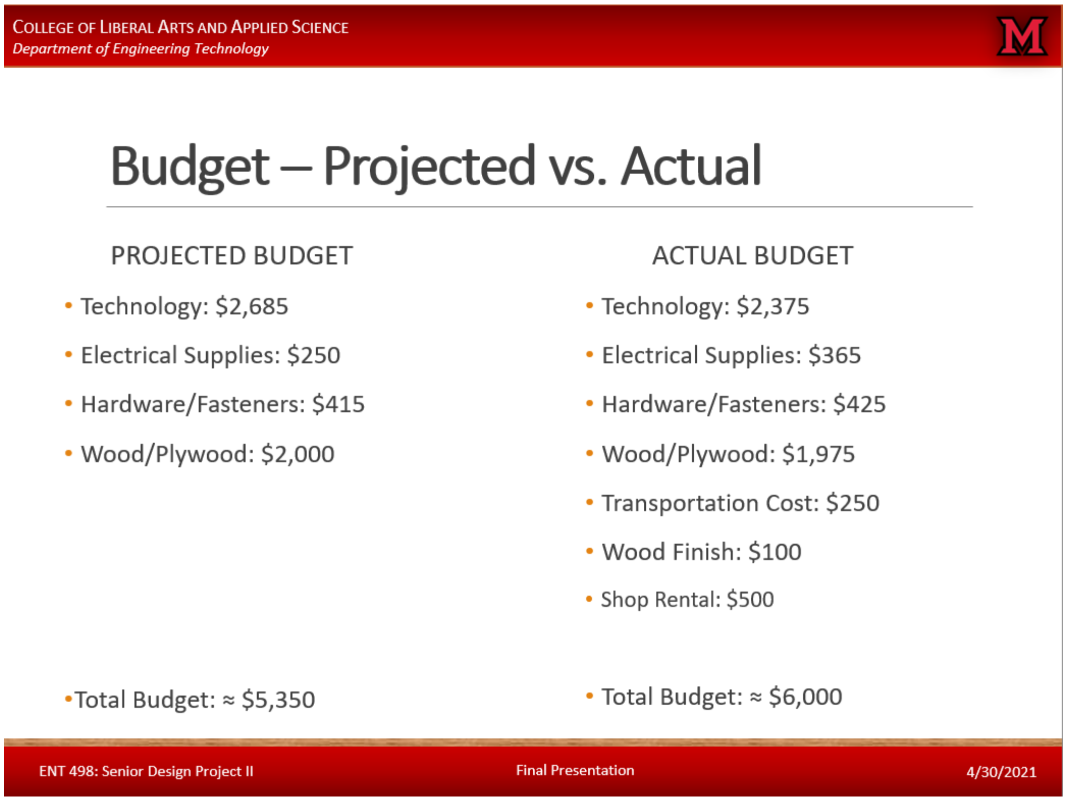
COLLEGE OF LIBERAL ARTS AND APPLIED SCIENCE
Department of Engineering Technology

Graphical User Interface

- Main Menu
  - Fan Control
  - Lamp Control
  - Wireless Charger Control
- Ambient Sounds
- Alarm Clock
  - Alarm Sounds
- Light Menu
  - White, Tunable
  - Custom Color
  - Animations



ENT 498: Senior Design Project II — Final Presentation — 4/30/2021



COLLEGE OF LIBERAL ARTS AND APPLIED SCIENCE
Department of Engineering Technology

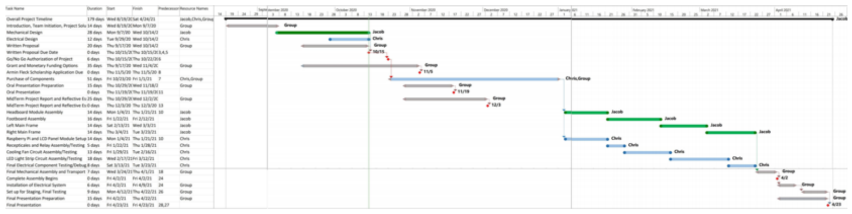
Budget – Projected vs. Actual

PROJECTED BUDGET	ACTUAL BUDGET
Technology: $2,685	Technology: $2,375
Electrical Supplies: $250	Electrical Supplies: $365
Hardware/Fasteners: $415	Hardware/Fasteners: $425
Wood/Plywood: $2,000	Wood/Plywood: $1,975
	Transportation Cost: $250
	Wood Finish: $100
	Shop Rental: $500
Total Budget: ≈ $5,350	Total Budget: ≈ $6,000

ENT 498: Senior Design Project II — Final Presentation — 4/30/2021

COLLEGE OF LIBERAL ARTS AND APPLIED SCIENCE
Department of Engineering Technology

Gannet Chart



Task Name	Duration	Start	Finish	Predecessor	Resource Names
Overall Project Timeline	179 days	Wed 8/19/2C	Sat 4/24/21		Jacob,Chris,Group
Introduction, Team Initiation, Project Solu	14 days	Wed 8/19/2C	Mon 9/7/20		Group
Mechanical Design	28 days	Mon 9/7/20	Wed 10/14/2		Jacob
Electrical Design	12 days	Tue 9/29/20	Wed 10/14/2		Chris
Written Proposal	20 days	Thu 9/17/20	Wed 10/14/2		Group
Written Proposal Due Date	0 days	Thu 10/15/2	Thu 10/15/2	3,4,5	
Go/No Go Authorization of Project	6 days	Thu 10/15/2	Thu 10/22/2	6	
Grant and Monetary Funding Options	35 days	Thu 9/17/20	Wed 11/4/2C		Group
Armin Fleck Scholarship Application Due	0 days	Thu 11/5/20	Thu 11/5/20	8	
Purchase of Components	51 days	Fri 10/23/20	Fri 1/1/21	7	Chris,Group
Oral Presentation Preparation	15 days	Thu 10/29/2	Wed 11/18/2		Group
Oral Presentation	0 days	Thu 11/19/2	Thu 11/19/2	11	
MidTerm Project Report and Reflective Es	25 days	Thu 10/29/2	Wed 12/2/2C		Group
MidTerm Project Report and Reflective Es	0 days	Thu 12/3/20	Thu 12/3/20	13	
Headboard Module Assembly	14 days	Mon 1/4/21	Thu 1/21/21	10	Jacob
Footboard Assembly	16 days	Fri 1/22/21	Fri 2/12/21		Jacob
Left Main Frame	14 days	Sat 2/13/21	Wed 3/3/21		Jacob
Right Main Frame	14 days	Thu 3/4/21	Tue 3/23/21		Jacob
Raspberry Pi and LCD Panel Module Setup	14 days	Mon 1/4/21	Thu 1/21/21	10	Chris
Recepticales and Relay Assembly/Testing	5 days	Fri 1/22/21	Thu 1/28/21		Chris
Cooling Fan Circuit Assembly/Testing	13 days	Fri 1/29/21	Tue 2/16/21		Chris
LED Light Strip Circuit Assembly/Testing	18 days	Wed 2/17/21	Fri 3/12/21		Chris
Final Electrical Component Testing/Debug	8 days	Sat 3/13/21	Tue 3/23/21		Chris
Final Mechanical Assembly and Transport	7 days	Wed 3/24/21	Thu 4/1/21	18	Group
Complete Assembly Begins	0 days	Fri 4/2/21	Fri 4/2/21	24	
Installation of Electrical System	6 days	Fri 4/2/21	Fri 4/9/21	24	Group
Set up for Staging, Final Testing	9 days	Mon 4/12/21	Thu 4/22/21	26	Group
Final Presentation Preparation	15 days	Fri 4/2/21	Thu 4/22/21		Group
Final Presentation	0 days	Fri 4/23/21	Fri 4/23/21	28,27	

Major Deadlines:

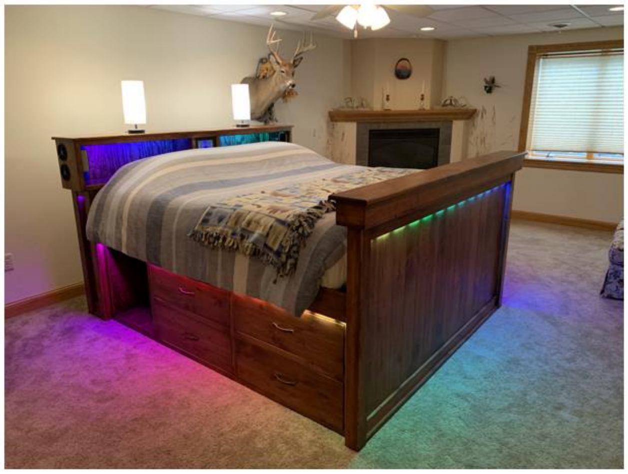
- Projected Construction Start – 1/1/2021
- Actual Construction Start – 1/4/2021
- Projected Construction Finish – 4/1/2021
- Actual Construction Finish – 2/17/2021
- Projected Electrical Finish – 4/12/2021
- Actual Electrical Finish – 4/12/2021
- Projected Project Finish – 4/22/2021
- Actual Project Finish – 4/14/2021

ENT 498: Senior Design Project II Final Presentation 4/30/2021

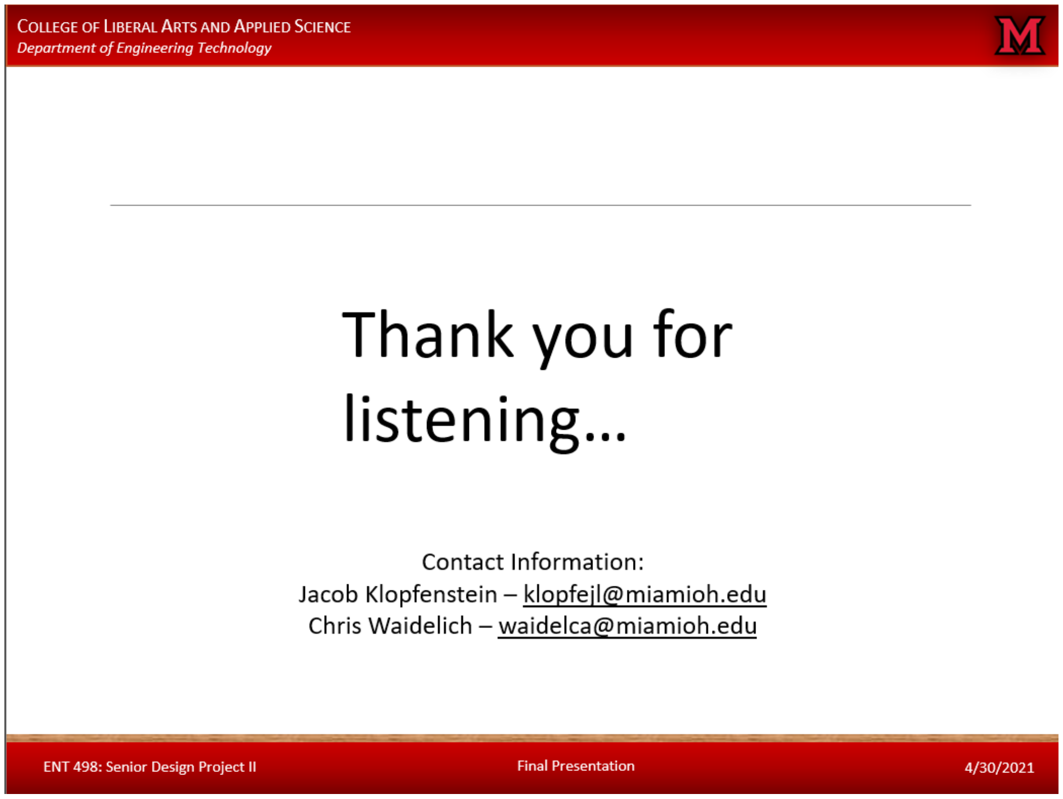
COLLEGE OF LIBERAL ARTS AND APPLIED SCIENCE
Department of Engineering Technology

Results and Future Works

- Completed Bed
- Future Enhancements
  - Automatic Sit-Up
  - Automatic Fans for Minifridge Box and Electronics Bay
  - Different 5V Power Supply
  - Additional Ambient Sounds/Alarms
  - Automatic Screen Dimer
  - Remove Deflectors, Add different internal Fan Guards



ENT 498: Senior Design Project II Final Presentation 4/30/2021



College of Liberal Arts and Applied Science
Department of Engineering Technology

Thank you for listening…

Contact Information:
Jacob Klopfenstein – klopfejl@miamioh.edu
Chris Waidelich – waidelca@miamioh.edu

ENT 498: Senior Design Project II Final Presentation 4/30/2021

Appendix K: Final Project Poster

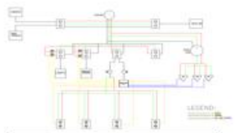
MIAMI UNIVERSITY

King of Kings TechBed

Abstract: The objective of the project was to create a king size bed that had a natural furniture wooden appearance while seamlessly integrating various technology systems into the bedframe. The design must also be capable of being disassembled for relocation purposes. Some of the technology features are a full surround sound system, RGBW strip lighting, wireless chargers, cooling fans with air deflectors, and a minifridge.

Team Members: Jacob Klopfenstein & Chris Waidelich **Advisor:** Reza Abrisham Baf **Sponsor:** Armin Fleck Scholarship

Electrical Design



Electrical Schematic Design

Controlled by Raspberry Pi/Touchscreen

- Headboard Fans, Relays and PWM
- RGBW Lights
- Headboard Lamp Relays
- Wireless Charger Relays
- Sound System (AUX)

Controlled by Arduino

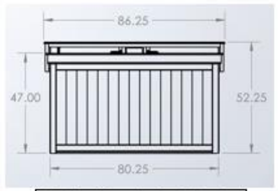
- Air Deflectors/Servo Motors

Hardwired Switches

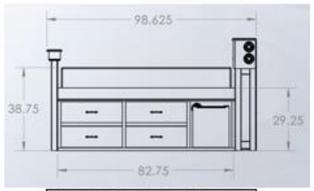
- Minifridge Cooling Fans
- Electronics Bay Cooling Fan
- Minifridge

Interior of Completed Electronics Bay

Mechanical Design



Footboard Perspective Design Drawing



Sideview Perspective Design Drawing

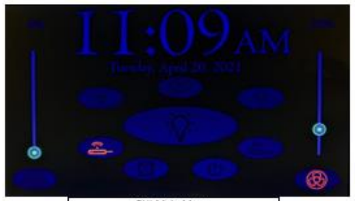
Mechanical Design Details

- The structure is made from White Pine and stained with Chestnut Base
- The headboard and footboard can be removed, and all electrical connections can be unplugged from headboard and footboard for moving purposes
- Air deflectors (not very visible in pictures) are controlled by servo motors mounted under the headboard
- The headboard fans move over 400 cubic feet of air per minute per side at full speed

Finished Bed Corner Perspective

Finished Bed Headboard Perspective

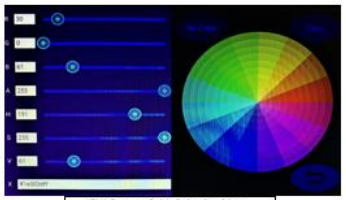
Graphical User Interface



GUI Main Menu

About the Software

- Programming Language of Python
- Graphics Library Kivy
- Symbols Light Up When Activated
- Color Selection Screen has 6.5 million different colors to choose from for the RGBW LED lights
- Twelve Different Ambient Sounds to Fall Asleep To
- Twelve Different Alarms to Wake Up To



GUI Custom Color Selection Menu

Summary and Conclusions: The bed created here utilized skill in mechanical engineering (to design the bed structure and assembly mechanisms), electrical engineering (to design the electrical systems, lighting, and relay controls), and controls engineering (to design the GUI and functioning programs), culminating in a redefined concept of what a modern bedframe should be.

Thank you to Lanz Construction for giving us shop space and access to the necessary tools, Andrew Shultz for his advice and training, and Angela & Mahlon Steiner for providing housing during construction.