ECE 441/442/443
ENGINEERING DESIGN PROJECT
Project Overview

Team 26 - IOT Alarm Clock

Andrew Zellner
Louis Marun
Cade Tillema
Joseph Conrow

Professor Heer

Fall 2023 - Spring 2024

Project Overview

This document serves as a foundational overview of the project, offering insights into the project's depth and scope. From information such as project description and team contacts to protocols, gap analysis, timeline, task list, references and file links, project risks, and a revision table.

1 Overview

1.1 Description

The product is an Internet of Things (IoT) alarm clock that can be configured through a user-friendly website, allowing one to set the clock time, alarm time, and snooze time using a computer/phone connected to the same internet network. In addition, the alarm system will be equipped with a backup power option using batteries and should last 48 hours, providing peace of mind in case of power outages or unplugs. The purpose of this project is to design and build a market-driven product, specifically aimed at parents seeking enhanced control of their children's morning routine to get them on track for school and other important plans. The team will fully design, host, and implement the user-friendly website using HTML, JavaScript, or other programming languages commonly used for web design. The website will send the user's input into an ESP32 module, which is a powerful Wi-Fi/Bluetooth board, and will get that information to the Arduino to get the time displayed on the screen. Similarly, once it is the time of the scheduled alarm, the speaker will buzz. The team will use this opportunity to gain experience with circuit design, hands-on electronics, and internet hosting.

1.2 Team Contacts and Protocols

TABLE I

TEAM CONTACT INFORMATION

Team Member	OSU Email	Role/Contribution
Cade Tillema	tillemca@oregonstate.edu	Timekeeper
Joseph Conrow	conrowj@oregonstate.edu	Reporter
Andrew Zellner	zellnera@oregonstate.edu	Recorder
Louis Marun	marunl@oregonstate.edu	Reflector

This table is for use in regard to understanding the organization and contact information for the involved project team members

TABLE II

TEAM PROTOCOL	
Issue	Protocol
Method of communication	Group chat (text), Discord server
Progress overview	Weekly meeting (Monday @ 1:30 p.m.) to update the team on the current status of each aspect of the project
Allotted time to ask questions and give feedback	The designated time during the weekly meetings
Miscommunication	Ensure that notifications are on for both group chat and Discord
Meeting attendance	Weekly meetings via Discord for more accessible attendance

This table is to explain the construction and methods of operation for the team's project work and organization.

1.3 Gap Analysis

Our project will be designed to fill the gap that is currently in the alarm clock market. Currently, there are few products that have the many functionalities that our alarm clock has. This product will attempt to bring out a new feature and direction compared to other products within the market, which will be based on the remote access enable and controlling feature. Our project will fill the gap in parent-controlled alarm clocks that have specific settings to make sure the parent has control. Through the design's 48-hour backup battery life, the alarm clock will not turn off if unplugged. Another aspect of our project's design is website control. Specific login credentials are required to access the alarm settings. Another key feature of our project is the ability to turn off the physical snooze button on the device, preventing children from delaying the alarm until parental approval.

1.4 Timeline and Task List

TABLE III

TASK LIST

Description of Task	Impact Risk	Expected Work Hours	Due Week	Champion	Actual Work Hours

Market Research Course	10	8	10	Andrew	8
Internal Components (routing/wiring)	7	14	20	Louis	7
Coding Internal Components	5	10	20	Joseph	60
Internet Access	10	20	20	Joseph	15
PCB Design	5	10	20	Louis	10
Display	6	12	20	Louis	5
Display Code	5	10	20	Louis	12
Physical Display (Buttons)	3	6	20	Cade	5
Enclosure Design	2	4	20	Andrew	10
Sound Device and Implementation	5	10	20	Cade	15
Backup battery Implementation	3	6	20	Louis	14
Purchased Components (Dependent)	8	16	20	Cade	13
Purchased Components (Independent)	4	8	20	Cade	12
Graduate	11	Many	30		

This is a running table of ongoing small tasks that must be completed in the order of importance/time

TABLE IV

PROJECT TIMELINE

Task	Week
Team Settlement	1
Market Research	2
Project Development	3
System Requirements	4

Parts Research	5
	6
Order Known Purchased Parts	7
	8
System Blocks	9
PCB Design	10
Purchase all other components	10
Enclosure Design	15
	16
Testing & Debugging	17
System Assembly	18
Enclosure Implementation	21

The overall outlook for a rough basis of deadlines for the project. This will also serve as a guide to keep us on track.

1.6 Revision Table

3/14/2024	Joseph Conrow: Updated the actual hours for the taks list
11/22/2023	Andrew Zellner: Edited and revised content throughout the section.
11/21/2023	Louis Marun: Added to project's description, posted project's description on the student portal, added content to the project overview, and formatted revision table.
11/11/2023	Joseph Conrow: Added table descriptions and fixed feedback comments
11/06/2023	Joseph Conrow: Formatted tables and added Title Page
10/16/2023	Andrew Zellner: Document created, initial content added.

2.2 Risks

TABLE V

RISK ASSESSMENT

Risk ID	Risk Description	Risk Category	Risk Probability	Risk Impact	Action Plan
R1	Parts ordered delayed from Chinese New Year	Schedule	Med	High	- Order parts early - Look into alternative websites such as OSHPARK for PCBs.
R2	Mass power outage	Environmental	Low	High	- Have a backup generator - Use the 48-hour backup battery power option
R3	Internet outage	Environmental	Med	Med	- Use mobile hotspot - Go to campus for internet access
R4	Board Fried	Technical	Med	High	- Double-check the input voltage - Order extra parts in case this happens - Split extra cost between members
R5	Electrical components don't fit in the enclosure	Technical	Low	Med	- Double-check dimensions before printing - Layout components to ensure proper fit
R6	Sick	Safety	High	Med	- Meet on Zoom/Discord - Figure out what needs to be done to still make progress as needed - If unable to finish work due to illness, contact team members to get help and get an

					assignment done on time.
R7	Team member extended unavailability	Schedule	Low	High	- Reach out to instructors - Document progress to minimize severe slowdowns
R8	Software compatibility issues with IoT components	Technical	Low	High	- Look for alternative components - Regulate and maintain a list of updates and version changes

The risk assessment is purposed at stabilizing forthcoming events or mishaps that might affect the timeline and projected outcomes throughout the project

2.4 Revision Table

Date	**Revision**
11/30/2023	Joseph Conrow: Added table descriptions
11/22/2023	Andrew Zellner: Changed spacing between a few sections and edited and revised some sentences in the section.
11/21/2023	Louis Marun: Added risks to the risk assessment table and reformatted the revision table.
11/12/2023	Andrew Zellner: Expanded on self-assessment and made edits to the document
11/11/2023	Joseph Conrow: Added table descriptions
11/07/2023	Joseph Conrow: Reformatted risk assessment table
11/06/2023	Louis Marun: Inserted team-created risk assessment table

Sections 1 and 2 Self Assessment:

For this project's overview, we put a strong emphasis on following the directions given to us. We went through the project document content guide and used it to write our section 2. We left 2.1 empty as mentioned in the guide, we had a table for 2.2 in IEEE format. This is because some of the feedback we got from section 1 is to use IEEE formation for consistency throughout

the document. We ended up deciding to include details on the action plan, in comparison to the risk assessment table format shown in the project document content guide. This helped us solidify our responses in case those risks happen. We also made a revision table for section 2 to help us keep track of any changes made by any group members. Adding a column for the revision number to help the group keep track of how many changes have been made. The corresponding updates we have made to the document through the creation of section 2 have been added to show what changes have been made. In addition, as a final submission for sections 1 and 2, we followed the feedback given to us in the drafts for sections 1 and 2 to fully revise our document making sure that all group members look over the document fully and provide any updates before submitting.

3 Top-Level Architecture

3.1 Block Diagram

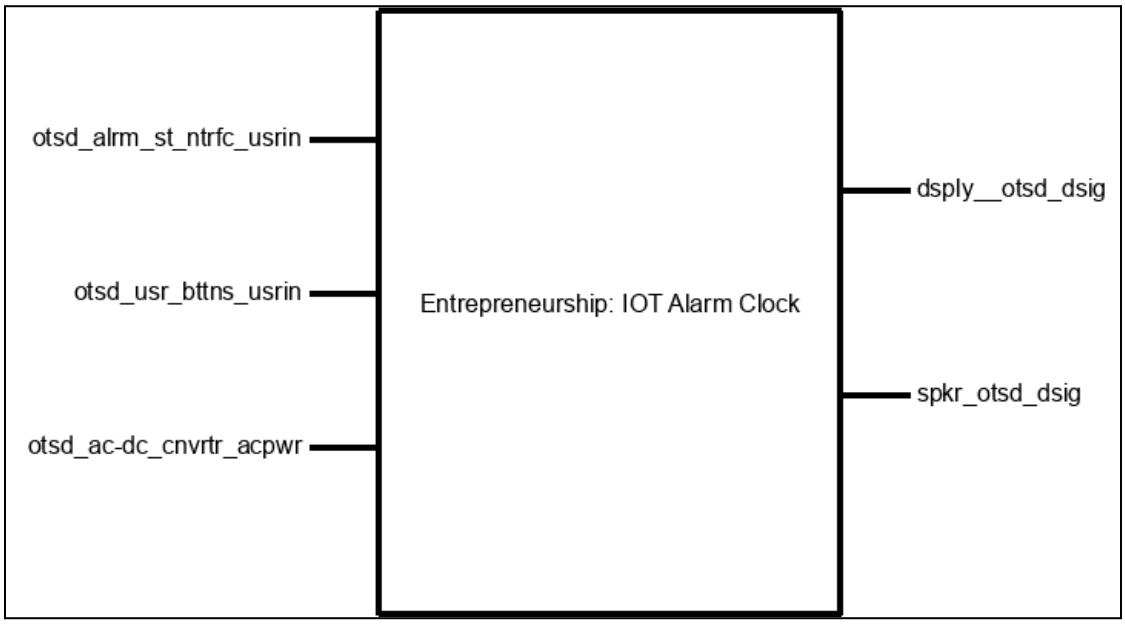


Fig. 1. Black Box Diagram

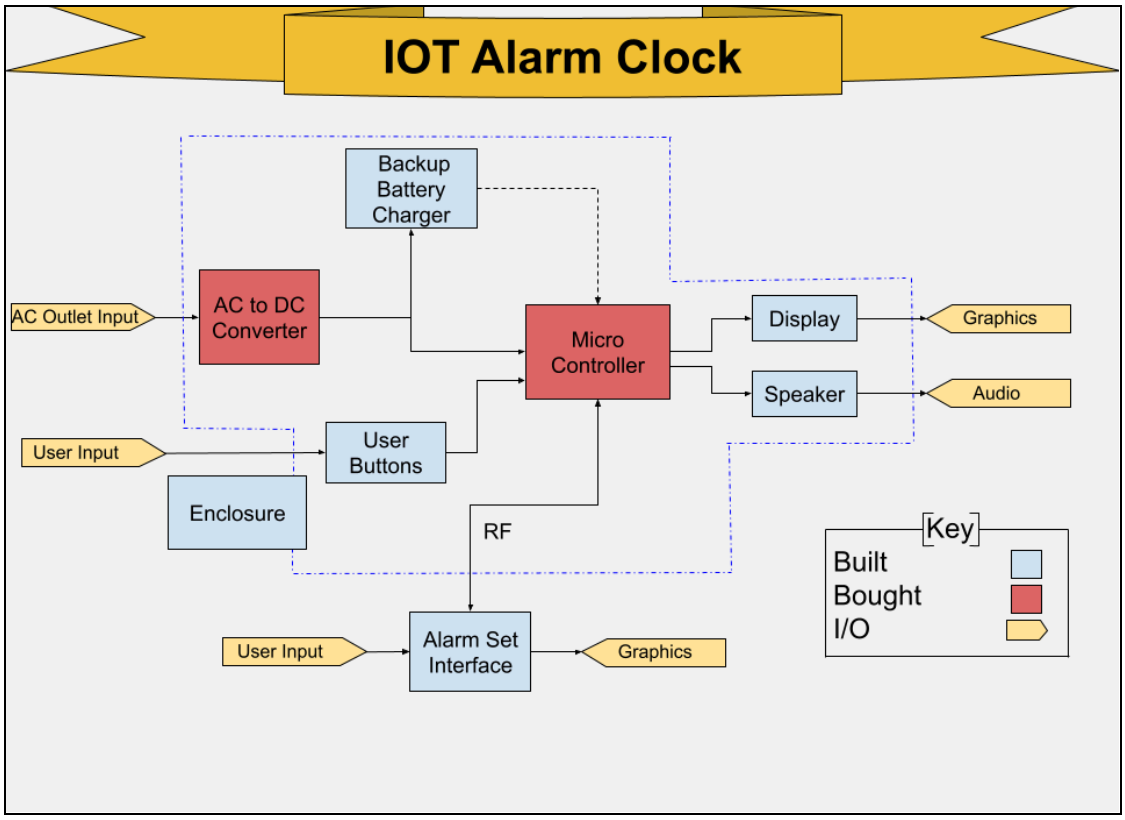


Fig. 2. Top Level Diagram

3.2 Block Descriptions

TABLE VI

BLOCK DESCRIPTIONS

Name	Description
Alarm Set Interface Champion: Andrew Zellner	The alarm set interface will take care of making the internet-based website configure the alarm clock. This interface is the area on the website that allows the user to set an alarm as well as enable or disable the snooze or alarm reset button on the physical device for any alarm that is set. This will mostly be achieved using coding and can be considered a coding block. This will also be achieved through the communication from the website to the microcontroller and then the microcontroller will communicate with the rest of the system. This block also accounts for the credential system in place for accessing the website. This user interface will have one input interface and one output interface. The input interface

	is the user input from the website and the output interface is a signal sent to the microcontroller based on the user input. The speed and frequency of these signals are based on the general concept that you only need to check for an alarm/ button change once every minute. Our alarm clock does not have the capability to be set by seconds. This decision has been made through market research as our customer does not need this. As well as the fact that checking each second would cause a large draw in power and would create difficulties in the system in the way of accomplishing our system requirements. Something else that will be sent to the microcontroller is user credentials. These credentials will be set up with the specific device and need to be entered successfully in order to enter the alarm set interface. To describe the properties of this block, there are two main properties of this block. The first one is an outside source of input as an input to the block. This input will be through mouse and keyboard and will be the user entering the time of an alarm as well as using their mouse to select if they want to turn on or off the user buttons. The second property is a signal that is sent from the interface to the microcontroller. The specific signal that is sent is for the time of alarm that is to be set as well as any information that is sent to the website from the user. That is the contents of the signal. The message of the signal will include what button is being acted on and what action to take on the button. Because these two signals will be sent at different times they are classified as two different input properties. The speed of each message is outlined in the detailed section about the properties, however, it is useful for understanding but not for proving through verification and validation. The website will be implemented through an AWS cloud-based server. This will be accomplished by purchasing a domain through AWS and hosting our page on that server. The domain of the website (listed in file links), is the area where the user will input to the block. This information will then be sent and stored in the database. Where this database will be accessed by the microcontroller to get the user-inputted information.
Enclosure Champion: Andrew Zellner	This block involves designing, building, and implementing the enclosure into our system. This will involve taking into account compact convenience as well as safety objectives. Mainly thinking into the safety side, especially for children interactions, the sharpness of edges, fragility, and electrical exposure will be all factors of the enclosure's design. The main purpose of the enclosure is to ensure the safe and compact nature of a solidly built object that the project's main components will be attached to and associated with. It will be printed using plastics and a 3D printer to encase the stand alone base of the alarm clock system, excluding, in which the system connects to the web interface wireless. It is meant to provide structural support and maintain resistance towards damages made

to the internals. The material chosen to execute this purpose will be PLA, as it is easily accessible, cheap, workable, relatively strong, quick to print, and covers all requirements with decency. The color of choice is non-dependent and can be whichever color is currently maintained by the team. The mounting points that will be included within the enclosure, will be heavily dependent on the exact final specifications of the other blocks and functionalities of the device as a whole. The measurements and fittings will be crucial to a successful protection enclosure. The speaker that will emit the sound will also be protected despite the need for sound waves to transmit. This will be solved through the use of a cage-like mesh design, creating a barrier between the speaker and the outside world, while allowing void gaps in the material to let sound waves travel through without turbulence. The hole for the input power to be inserted, will be a direct hole, with the direct interface recessed into the enclosure's wall, keeping it further from reach of the user or environment. The shape of the design will be smooth and very gradual, minimizing edges and optimizing potential safety concerns. The integration process involves the incorporation of shelving-style slots within the enclosure, specifically crafted to accommodate PCB (Printed Circuit Board) based components. This strategic design choice facilitates a secure and organized placement of the electronic elements, contributing to the overall structural integrity of the system. In addition to the slots, a sophisticated approach is employed, utilizing threaded inserts that can be melted into the PLA (Polylactic Acid) enclosure material. This innovative combination of shelving-style slots and threaded inserts serves a dual purpose. Firstly, it enhances the stability of the mounted components, providing a robust foundation that withstands the dynamics of daily use. Secondly, it ensures a secure fit, preventing any inadvertent displacement of essential elements. The threaded inserts, with their unique ability to meld seamlessly into the PLA material, offer a reliable and durable connection. Moreover, this design methodology facilitates repetitive removal of components, which is particularly advantageous during the prototyping phase. It allows for iterative testing and modifications without compromising the overall structural integrity of the enclosure. This adaptability is invaluable in the development process, enabling the refinement of the alarm clock system based on user feedback and evolving requirements. The dual emphasis on security and the slot method contributes significantly to the overall resilience of the entire system. This robust design approach ensures that the alarm clock system remains resistant to various types of handling and rough usage. Whether in a household with energetic children or in a dynamic environment, the secure mounting mechanism and thoughtfully engineered slots mitigate the risk of damage, enhancing the longevity and reliability of the entire system. As a result, users can confidently interact with the alarm clock,

	knowing that its internal components are securely housed within a design that prioritizes both functionality and durability.
Display Champion: Louis Marun	The display block will be primarily used to show the current time on the clock. The majority of the work that will go into this block is integrating it into the system and coding it to effectively communicate and showcase the current time.In addition, since we are employing a backup battery option, it will be important to manage the current draw of the display, especially since the display we will be using has a high current draw. In terms of interface details, this block has two inputs and one output. The first, and expectedly important, input is power. It is supplied by the microcontroller, which ensures the continuous operation of the display. The DC input voltage will be a fixed 3.3V, with a nominal current of 60mA and a maximum of 80mA. These values were determined by the display's datasheet which serves as the main source of our information about this block, such as wiring diagrams, example codes, and hardware specifications As expected, another input is the digital signal from the esp32, which is how the display gets the information. This information can vary in type, with a logic level of 0 or 1, maximum frequency of 32MHz, and a 10 KHz Pulse Width Modulation Frequency. These values are based on what the esp32 can provide, which were extracted from its datasheet. Lastly, the output, on the other hand, is a signal in the form of visual information, specifically the numerical representation of the current time. The display will be active high, which determines which segment of the display is on. Furthermore, the maximum representation of anything on the display is in 18 bits, with a refresh rate of 70Hz. The integration part is a key aspect of our block development process, with a focus on correctly receiving information from the ESP32. While receiving data from the microcontroller is fundamental, coding the display to process that data is the other major focus of this block. With that being said, it will have a focus on how the display is going to work. Coding the display is important to process the signal from the microcontroller to get the information in numeral values corresponding to the current time. Furthermore, things like resolution scale, color, brightness, and positioning of numbers on the display will be included in the coding aspect of this block. This could get more complex depending on what kind of signal/information the display is receiving from the ESP32, and the mode of the display we use. The display we have chosen has an LCDscreen, and is a higher quality than some of the other ones we say. As a result, it is a little more complicated to program, and we might be using a specific library that fit's its operation, such as U8Lib in contrast to the Adafruit, or vice versa. The primary reason for that is we had many options for displays, but had to consider their pros and cons, such as complexity,

	usefulness, and power draw. As a result, certain displays are more difficult to program than others.
User Buttons Champion: Cade Tillema	The user buttons will be physically present on the enclosure. The reason we decided to separate them is because we need to account within hardware and software functionality of these buttons, which is a part of this block too. The physical buttons will be in an accessible location on the exterior of the enclosure to ensure that the user has no problem pressing them. There will be two buttons in total – one will be used to turn off the sounding alarm and reset it for the next day, while the other button will be used to snooze. The input of this button will be the power provided from the microcontroller and will route it to an input depending on if the button is being pressed.
Backup Battery Charger Champion: Louis Marun	This PCB block will end up making it into our final system, an internet of things alarm clock. It will be used to charge the batteries in the system, for a backup power option. This PCB block has one input and one output. As expected, the only input is DC power. The DC power will come from a different block that will be responsible for converting AC power from the outlet into a DC power, which is used by this PCB block to charge the batteries. This DC input power will be at 5V fixed, but the PCB itself can take anywhere between 3.75V and 6V because of the MCP73831/2 chip that manages the charging functionality of the battery. With this in mind, in case of an emergency power out, the batteries will be fully charged and ready to provide power for the system. The output is DC power to the microcontroller, an esp32. The esp32 can be powered up in different ways, but the current plan is to use the output of this PCB block, which will be 5V, to power up the esp32. This will be sufficient since despite the fact that the esp32 module operates at 3.3V, it has a built-in regulator that steps down the 5V input from this block into 3.3V. As a result, the PCB will have an output header that will directly connect to the Vin and GND pins of the esp32. Furthermore, it has a few more headers for other connections that we might end up using, such as powering up the display, RTC module, and the amplifier for the speaker. This setup, in contrast with directly powering up the microcontroller from the AC-DC converter, is pivotal for ensuring constant operation of the system, especially during critical moments such as a power outage. In addition, this gives the alarm system more portability and usage beyond an ordinary clock that only operates using an electrical outlet.
Speaker Champion: Cade Tillema	This block involves the speaker that will be used to output the sound of the alarm. We will be responsible for the purchasing of the speaker, as well as the physical connection to the microcontroller. Alongside the physical connection we will also need to properly program the

	microcontroller to drive the speaker. Outputting sound will obviously require an amplifier to ensure that the sound is audible. We will look into purchasing a module that includes an amplifier that is a part of a speaker. If this is the case, it will require power from the controller as well as power to drive it.
AC-DC Converter Champion: Joseph Conrow	This is a power supply block where we are converting AC power to DC power. It will be used to supply DC voltage to the batteries that supply the system power. This power supply block has one input and two outputs. As expected, the only input is AC power. The two outputs are the voltage to the backup battery charger and the microcontroller. In order for this power supply to be able to supply power to the system, it needs to be connected to some external power source. For this, we will be inputting 120V 60 Hz AC power from a wall outlet. The output interfaces will be getting a range of DC voltage. This power supply will step down the converted AC to DC voltage inside the range described in the interface properties. The work that will go into this power supply is research on which component will work the best for our system needs. As well as making sure that the power supply will fit into the system. Through various meetings, the team has decided on a converter to fulfill this block. The system's main PCB which has connections from many components currently has a barrel jack which is the connection from the converter. This component does not have any issues with size constraints as all of the conversion and meat of the component is on the plug-in, on the wall outlet. This block will be directly on the inside of the enclosure on one of the walls with a hole that fits the space for the barrel jack cable to connect to the PCB. The specific component that the team is purchasing is the Wall Power Adapter from Pololu Robotics and Electronics. This component was chosen because of its size, hardware specifications, and cost. Looking at the size, as stated earlier, the cable that will be connected to the system does not have any extra hardware or conversion; it is just the cable head. This makes it easier for the enclosure, as there will be a smaller hole needed for the cable to fit through. In comparison to a converter that has the conversion block on the cable end instead of the wall end. In regards to the specifications, looking at the specifications for this component it takes in wall outlet power and outputs 5 volts DC power. This is exactly what the system needs as the battery backup charger takes in 5 volts. Lastly, in regards to the cost, as a team, we are at a great spot with our budget and have some room to spend some extra money. However, in a real project, the goal is to get the system to perform the best for as cheap as possible and you never know what may happen. So, finding this component relatively cheap was great for the team. Another aspect of this component is that it has a cable length of 5 feet, which

	makes our system able to move around and be flexible with what our customers need and want.
Microcontroller Champion: Joseph Conrow	The microcontroller will be the main interconnected piece of the entire project. As it is the main communication device and interfaces between all the other functional blocks. It is crucial to determine the appropriate limitations and functionality that the specific model and module chosen has and can possibly develop. The overall final decision model we chose was the ESP32-WROOM–32 module, AKA “DOIT ESP32 DevKit V1 Wi-Fi Development Board”. This way, with a pre-built chip design, the ESP32 chip itself can easily be accessed by our other components very easily using pin headers. The power supply pin also was able to regulate the specific power demands of our design. As the voltage regulator on the module allowed us to focus all of our power consumption from a single component source, as the method of supply to a standardized system voltage, that is, 5 Volts. Some of the more specific rationale we used to pick this board were guided by their compatibility in communication protocol in reference to our selected project graphical display component, speaker component, internet component, and control components. These modes of communication involved SPI, I2C, PWM, WiFi, and DC signals. These are all protocols that the module chosen can provide with the ESP32 chip soldered on. The power draw of 3.3 volts is also something that the display is able to run off of, which the ESP32 can supply, allowing all of the communications and connections of the display to be rooted to the microcontroller. The connections to the web interface, and thus the user/clock/alarm server database is also to be accessed solely via the microcontroller. The speaker is also able to run and power distinctly sourced from the microcontroller, giving it the correct signals and internet connection to source a radio sound source if such audio source is chosen. The buttons also are a direct interworking interface for the microcontroller, as it can supply itself with a signal voltage and a receiving pin to recognize the state of such user buttons. The inner workings of the microcontroller bring all of these components together to work in harmony. By receiving signals from the database, the display can be correctly displaying the correct time value. It conducts the appropriate sound values and timings for the alarm based on the stored website alarm settings. The microcontroller will be sustainable within the state of offline connection as well. One of the project objectives is to deliver the same functionality despite the source of power or internet connection. The small physical footprint of the chosen microcontroller is also desirable and easy to fit within our designed product enclosure. As this chosen microcontroller module is apt to function within each specific

	interface of the project components, the ability of it is also able to manage, store information, and send it where it needs to go, in a proper and useful manner, allowing the user to experience a clean, and seamless interaction with the product, regardless of which end component they are conducting interaction with, whether that be the graphical clock display, the website interface, the control buttons, or the audio of the speaker.

Explains the inputs and outputs of device blocks, their purpose, and usage

3.3 Interface Definitions

TABLE VI

BLOCK DESCRIPTIONS

Name	Properties
otsd_usr_bttns_usrin	● Timing: After alarm is triggered ● Type: Toggle Switch ● Usability: Press Button
otsd_ac-dc_cnvrtr_acpwr	● Ipeak: 20A ● Vmax: 140V ● Vmin: 100V
otsd_usr_bttns_dcpowr	● Vmax: 5V ● Vmin: 3V ● Vnominal: 3.3V
otsd_alrm_st_ntrfc_usrin	● Other: When the user clicks on the clock displayed on the screen, a slider for each hours, minutes and AM/PM shows on the screen. (Ambiguous wording. Is there a single slider that indicates both hours and minutes, or are there separate sliders for each?) ● Type: The type of this input will be mouse clicking/keyboard stroke entry. (Fix typos.) ● Usability: 9 out of 10 users can use and understand this interface. ●

alrm_st_ntrfc_mcr cntrllr_rf	● Messages: The transmitted data is the time of alarm that is to be set. ● Messages: A message transmitted is the enable/disable for the alarm reset button ● Other: A confirmation message is displayed to the screen once the information of the alarm alarm is successfully sent to the microcontroller (This is not a property of rf communication.)
usr_bttns_mcrcntrl lr_dsig	● Logic_Level: Active High ● Vmax: 5V ● Vnominal: 3.3V
spkr_otsd_envout	● Max Frequency: 21kHz ● Signal: Sound ● Signal: Volume within 10% of desired decibel level
ac-dc_cnvrtr_bckp _bttry_chrgr_dcpw r	● Inom: 53uA ● Ipeak: 513mA ● Vmax: 6V ● Vmin: 3.75V
bckp_bttry_chrgr_ mcrcntrllr_dcpwr	● Inominal: 250mA ● Ipeak: 450mA ● Vmax: 5.25V ● Vmim: 4.75V
ac-dc_cnvrtr_mcrc ntrllr_dcpwr	● Ipeak: 250mA ● Vmax: 3.75 ● Vmin: 2.75V
dsply__otsd_envo ut	● Light: Brightness Level: 3 -+ 2 Lux ● Other: Usability Test: The display should show the correct input time from the esp32 9 out of 10 times. ● Other: User Test: 9 out of 10 people have to be capable of reading the time displayed on the screen from 5 meters away.

mcrcntrllr_dsply__dcpwr	• Inom: 68mA • Ipeak: 83mA • Vmax: 5V • Vmin: 3.3V
dsply__otsd_dsig	• Logic Level: Bit value 0 or 1 • Max Frequency: 1Hz • Vmax: 2.4V
mcrcntrllr_spkr_assig	• Vmin: 2.7V • Power: 1W • Vmax: 5.5V
mcrcntrllr_spkr_dcpwr	• Inominal: 4mA • Vmax: 5.5V • Vmin: 2.7V
otsd_enclsr_envin	• Other: The walls of the enclosure will stay connected when dropped from a height of 3 feet. • Other: 9 out of 10 users will say it is hard to open the enclosure. • Other: The maximum dimensions of the enclosure are 12" x 12" x 12" (L x W x H).

Specifies and address the appropriate interface requirements and characteristics

3.4 References and file links:

3.4.1 References

Arduino Mega 2560 Rev3 — Arduino Official Store

3.4.2 File Links

ESP32 Microcontroller Datasheet

3.5 Revision Table

Date	Revision

3/14/2024	Joseph Conrow: Adjusted uniform formatting
3/12/2024	Andrew Zellner: Updated Block descriptions/interface definitions for the Alarm set Interface and the Enclosure.
1/11/2024	Removed random character in section 3.4.2
12/4/2023	Andrew Zellner: Updated block diagram descriptions, interfaces, interface properties, and added reference.
12/1/2023	Joseph Conrow: Adjusted table and figure formatting and updated interfaces
11/30/2023	Louis Marun: Filled out interface properties for Display and Battery charger blocks, filled out their descriptions,, and added a revision table.
11/30/2023	Cade Tillema: Inserted parts 3.1, 3.2, and 3.3 and included content

4 Validations

4.1 Battery Backup Charger

4.1.1 Description

This PCB block will end up making it into our final system, an internet of things alarm clock. It will be used to charge the batteries in the system, for a backup power option. This PCB block has one input and one output. As expected, the only input is DC power. The DC power will come from a different block that will be responsible for converting AC power from the outlet into a DC power, which is used by this PCB block to charge the batteries. This DC input power will be at 5V fixed, but the PCB itself can take anywhere between 3.75V and 6V because of the MCP73831/2 chip that manages the charging functionality of the battery. With this in mind, in case of an emergency power out, the batteries will be fully charged and ready to provide power for the system. The output is DC power to the microcontroller, an esp32. The esp32 can be powered up in different ways, but the current plan is to use the output of this PCB block, which will be 5V, to power up the esp32. This will be sufficient since despite the fact that the esp32 module operates at 3.3V, it has a built-in regulator that steps down the 5V input from this block into 3.3V. As a result, the PCB will have an output header that will directly connect to the Vin and GND pins of the esp32. Furthermore, it has a few more headers for other connections that we might end up using, such as powering up the display, RTC module, and the amplifier for the speaker. This setup, in contrast with directly powering up the microcontroller from the AC-DC converter, is pivotal for ensuring constant operation of the system, especially during critical moments such as a power outage. In addition, this gives the alarm system more portability and usage beyond an ordinary clock that only operates using an electrical outlet.

While providing power to the system is one of the most important functionalities of this block, another important one is ensuring the system will last 48 hours while being unplugged. As discussed, in case of an emergency power out, this PCB should still be able to provide power to the system using the batteries. Considering that the system will need to stay on for 48 hours, and that the system's maximum current consumption is around 450mA, we needed to find high capacity batteries. As a result, this block includes the battery case that will house the two batteries we are using. The batteries were chosen based on their charge capacity and charging voltage. Each battery has 9900 mAH, which when hooked in parallel, should provide 3.7V with 19800 mAH. This means that the batteries should last around 48 hours if the system is constantly drawing 412.5 mA, which is close lower than our peak current. Since the current draw of our system will peak only when the esp32 is receiving information through wireless/bluetooth, the nominal current draw will be much lower. This means the block should also supply enough power for the system to remain on for 48 hours, given that the batteries are fully charged.

4.1.2 Design

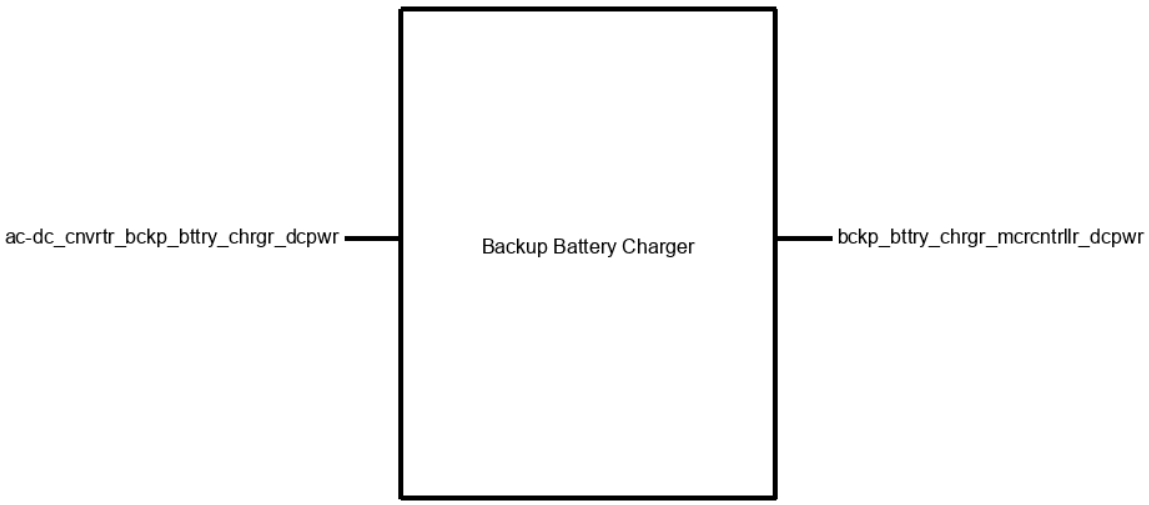


Figure XXX: Backup Battery Charger Black Box Diagram

The figures below show the PCB block's design. The first picture, being the black box diagram, simply shows the interfaces of the block, and how it is connected to two other blocks in the system. The input is the input dc power from the AC-DC converter block. The output is DC power to the microcontroller. Next, figure 2, shows the schematic of the design and highlights the components, connections, and labels of the circuit. Figure 3 is the schematic imported into the PCB editor, which is essential for implementing the designed schematic into a fabricatable circuit that we can use. Finally, figure 4, shows a realistic view of what the PCB will look like after soldering the components on it. Overall, as we go through the figures, the design becomes more realistic and fit to be integrated into our system.

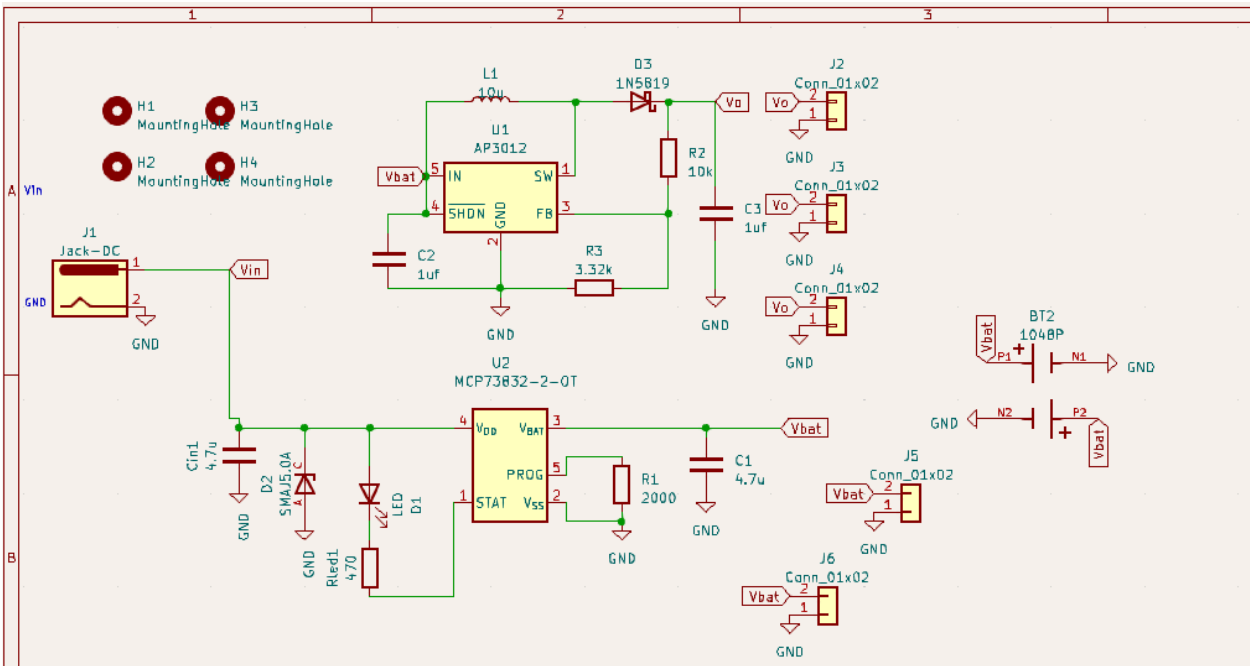


Figure XXX: Design Schematic

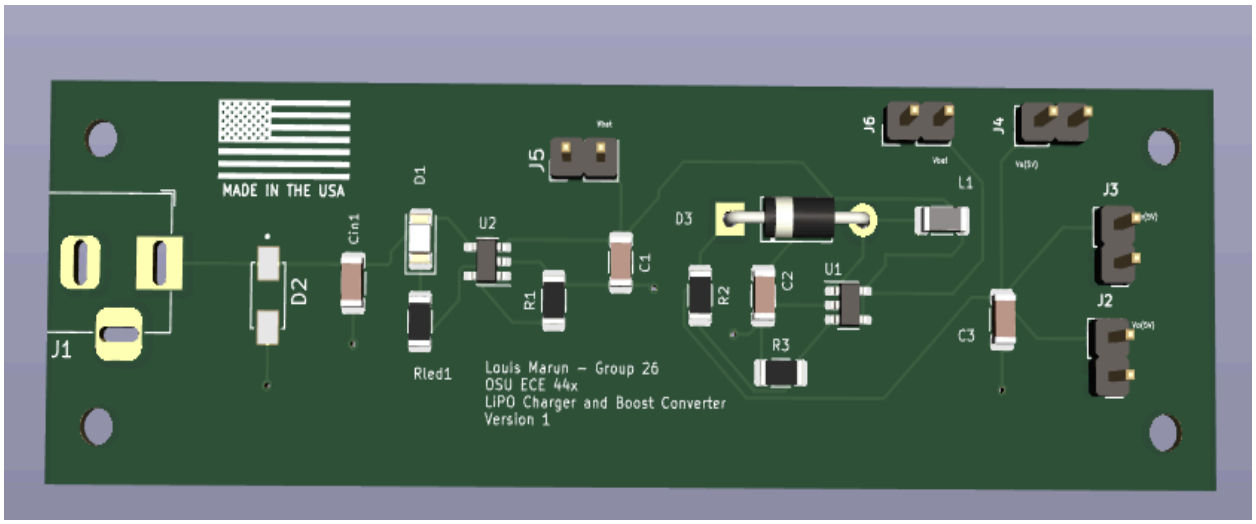


Figure XXX: PCB in PCB Editor

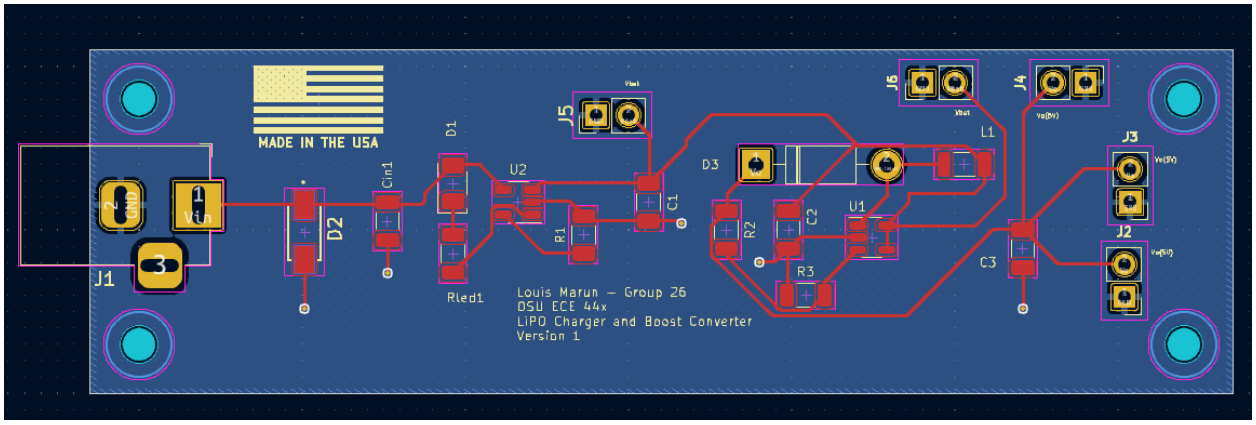


Figure XXX: PCB in 3D viewer

4.1.3 General Validation

The main functionality of this block is that it takes in input from the AC-DC power converter, and uses that to charge the batteries. From there, the 3.7V-4.2V output of the batteries is stepped up by the boost converter and is used as the output DC power to the system. The reason why we chose to power up the system using the batteries instead of directly from the output of the AC-DC converter is to maximize operation. When the system is connected to an outlet, the batteries will be charging and providing power to the system. However, in the case of an emergency outlet, the batteries will not be charging anymore, but they will still provide power to the system. This is important to our system because if there is a power outage in the middle of the night, then the alarm won't buzz and the person won't wake up.

The design of the block fits the needs of the system in many ways. As previously stated, the block takes input power through the barrel jack connector on the bottom left corner of the PCB. This is supplied from the AC-DC power converter block labeled as J1. The output voltage can be seen on the three output headers; at the right corner of the PCB labeled: J2, J3, and J4. This will be a 5 volt output that will directly connect to the esp32 to power it up. Furthermore, there are two more header pins that are at 4V and will be directly connected to the batteries. The output current provided by this block will depend on the draw of the system. This includes the esp32, display, audio amplifier, speaker, and RTC module. The details on the current consumption will be discussed in later sections, but the PCB's boost converter was chosen carefully to be able to provide a maximum of 500mA, which is well above what the system will be drawing.

The PCB's 4V output headers will connect to a battery case with two batteries. This is still a part of the block, which is why this isn't shown as an output in the black box diagram. Two batteries were picked, at 9900 mAH each, to maximize the time the batteries can power up the system. To get more technical, the 5 volt input from the AC-DC converter is stepped down to 4 volts using the MCP chip. This goes directly into the battery case to charge the batteries. Then,

the voltage from the batteries goes through a boost converter that steps up the voltage to 5 volts again, which will be utilized by other blocks in the system, primarily the esp32. In addition, the second chip in the design, the MCP73831/2, sets the charging current of the battery. To ensure that the batteries charge quickly enough, considering the large capacity they have, we chose a value of 2k ohms for R1. This programs the chip to send 500mA as an output to the batteries, which means each battery will be charged with 250mA. With the fact that 500mA is about how large the current through the PCB will get, a trace width of about 0.25mm is sufficient. With this design, the batteries will be constantly charging until they reach max capacity, so that they are ready to be used in case of power outages or in areas with no power outlets. The batteries and their charging case will be purchased, along with all of the components on the PCB.

An alternative solution to this block is to have simply gone with an already existing battery charger. From there, we could have easily soldered some wires to the case and stepped up the 3.7V - 4.2V to 5V using a pre-purchased boost converter PCB module. Considering that the PCB has three 5V outputs, we would also be required to use a protoboard to the number of output voltage pins. While this solution would have been easier and more straightforward, we thought that fully designing this block would give us good experience with PCB design, and since we already worked on a similar PCB in the fall for our tech demo, we thought it wouldn't be too difficult to implement into our system.

4.1.4 Interface Validation

TABLE XXX

Battery Backup Charger Interfaces

Interface Property	Why is this interface this value?	Why do you know that your design details <u>for this block</u> above meet or exceed each property?
bckp_bttry_chrgr_mcrcntrllr_dcpwr : Output		
Inominal: 250mA	This is the amount of current we are expecting the major blocks in our system to draw in typical operation. For the esp32, the nominal current is about 100mA, display is around 60 mA, amplifier is 60 mA, and other components	The design details meet this block because the PCB block will provide as much current as the system will need to use. Considering that this block can provide up to 500mA[3], as long as the system pulls less,

	like RTC and buttons up to 30mA. With all these currents summed up, we get around 250mA [4][5].	this design will be sufficient for the current the system uses.
Ipeak: 450mA	In worst case scenarios, the peak current the esp32 pulls is around 250mA. The display can peak up to 80mA, amplifier around 80mA, and the rest 40 mA [4][5].	Similarly, as long as the batteries are charged, they can supply this amount for the esp32. Given that the current iteration of the PCB can charge the battery, the block should be able to provide 450mA[3].
Vmax: 5.25V	Based on how the PCB was designed, the output voltage is fixed at around 5V. Therefore, the Vmax upper range is around 5.25V and the output voltage won't be higher.	Charging a 4V battery is already a feature of the current PCB design. Therefore, the battery should have no issue providing 5V maximum to the esp32 after the voltage boost using the ACP chip. This should be very close to a fixed 5V output and shouldn't exceed 5.25V
Vmin: 4.75V	Based on how the PCB was designed, the output voltage is fixed at around 5V. As a result, the Vmax lower range should be around 4.75V and the output voltage will not be lower.	Similarly, charging a 4V battery is already a feature of the current PCB design. Therefore, the battery should have no issue providing 5V maximum to the esp32 after the voltage boost using the ACP chip[3]. This should be very close to a fixed 5V output and shouldn't get lower than 4.75V
ac-dc_cnvrtr_bckp_bttry_chrgr_dcpwr : Input		
Inominal: 53uA	When the batteries are not charging, the input current will only be at 53u[2]. This makes since considering that the input current of the PCB is the charging current of the batteries, so if the batteries aren't charging, there won't be a lot of current going into our PCB since the batteries will be powering up the system. This will	The design is capable of taking in this current to charge the batteries, and provide the needed current to the system. This is primarily because the MCP73831//2 can easily take in and provide anywhere between 0 to 500mA of current[2]. In such a case, the current that will be provided into the system will be from the batteries.

	be the case when the battery charge cycle is complete.	
Ipeak: 513mA	The input current of the PCB is based on what current the MCP chip is configured to charge the batteries. For our case, this will be around 500mA since a 2K ohm resistor configures the charging current to that value. [2]].	Similarly to Inominal, the design is capable of taking in this current to charge the batteries, and provide the needed current to the system. As highlighted earlier, the MCP73831/2 chip can be programmed to provide 500mA[2], so the expected Ipeak is well within our design consideration. Furthermore, as previously mentioned, the PCB traces, 0.25mm, are wide enough to sustain this much current[6].
Vmax: 6V	This is the maximum voltage the MCP73831/2 chip can use for input for regulation[2].	This is perfect for our system since the input voltage will be a fixed 5V DC input. This will be supplied from the AC-DC converter block, which is below the Vmax the MCP73831/2 can handle.
Vmin: 3.75V	This is the minimum voltage the MCP73831/2 chip can use for input for regulation[2].	Same thing applies here; the MCP chip takes in a minimum of 3.75V[2]. Therefore, the design details of this block matches with this property since the input voltage is well above the minimum at 5V.

4.1.5 Verification Plan

1) Connect the input barrel jack connector to a power supply for early testing or the AC-DC power converter block for a more advanced testing with other blocks in the system.
2) Feed in the power from the power supply into the barrel jack connector. To verify the input: ac-dc_cnvrtr_bckp_bttry_chrgr_dcpwr, start with showing that the block works at 3.75V input(Vmin) by setting that voltage in the power supply and ensuring that Vbat is at 4V. Then, do the same with a 6V

input(Vmax) and ensure that Vbat is at 4V again. With this, we verified the input voltages.

3) Next, move on to the output and verify bckp_bttry_chrgr_mcrcntrllr_dcpwr. The output of the PCB block will be a fixed 5V output. As a result, verify this part of the feed in the range of input voltage and ensure the output voltage is between 4.75V - 5.25V. After the output voltages are verified, all that is left is to verify input and output current.
4) Use the electronic load to pull around 250 mA and ensure the PCB is properly taking in that current and supplying. This can be done by checking the current reading on the power supply and the electronic load. Next, do the same with Ipeak, and verify that 450mA is going through the block by seeing that the power supply is providing 450mA into the PCB, and that the DC electronic load is set to pull 450mA. With this, we verify Inom for input and output, as well as Ipeak for input and output.

4.1.6 References and File Links

4.1.6.1 References

[1] “The ESP32 devkit power options,” Tech Explorations, https://techexplorations.com/guides/esp32/begin/power/ (accessed Jan. 21, 2024).

[2] MCP73831/2 data sheet - microchip technology, https://ww1.microchip.com/downloads/en/DeviceDoc/20001984g.pdf (accessed Dec. 5, 2023).

[3] General description features - diodes incorporated, https://www.diodes.com/assets/Datasheets/AP3012.pdf (accessed Jan. 22, 2024).

[4] ESP32 series - espressif systems, https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf (accessed Jan. 22, 2024).

[5] Adafruit, https://cdn-shop.adafruit.com/datasheets/MI0283QT-11+V1.1.PDF (accessed Mar. 5, 2024).

[6] “PCB trace width calculator,” DigiKey, https://www.digikey.com/en/resources/conversion-calculators/conversion-calculator-pcb-trace-width (accessed Mar. 5, 2024).

4.1.6.2 File links

N/A

4.1.7. Revision

Date	Revision
3/14/2024	Joseph Conrow: Adjusted uniform formatting
3/5/2024	Louis Marun: Updated sections 1, 2, 3, 4, 5, revision table, and revision statement
3/2/2024	Louis Marun: Updated sections 1, 2, and 4.
1/28/2024	Louis Marun: Edited, updated, and finalized all sections
1/21/2024	Louis Marun: Updated section 3,4, 5, 6, and added to the revision table.
1/18/2024	Louis Marun: Updated section 1, replaced pictures of section 2 with newer PCB version, and updated section 4
12/5/2023	Louis Marun: Filled out sections 5 and 6
12/4/2023	Louis Marun: Made google docs, inserted template for the block, and filled out sections 1-4 and 7

4.2 AC-DC Converter

4.2.1 Description

This is a power supply block where we are converting AC power to DC power. It will be used to supply DC voltage to the batteries that supply the system power. This power supply block has one input and two outputs. As expected, the only input is AC power. The two outputs are the voltage to the backup battery charger and the microcontroller. In order for this power supply to be able to supply power to the system, it needs to be connected to some external power source. For this, we will be inputting 120V 60 Hz AC power from a wall outlet.

The output interfaces will be getting a range of DC voltage. This power supply will step down the converted AC to DC voltage inside the range described in the interface properties. The work that will go into this power supply is research on which component will work the best for our system needs. As well as making sure that the power supply will fit into the system. Through various meetings, the team has decided on a converter to fulfill this block. The system's main PCB which has connections from many components currently has a barrel jack which is the connection from the converter.

This component does not have any issues with size constraints as all of the conversion and meat of the component is on the plug-in, on the wall outlet. This block will be directly on the inside of the enclosure on one of the walls with a

hole that fits the space for the barrel jack cable to connect to the PCB. The specific component that the team is purchasing is the Wall Power Adapter from Pololu Robotics and Electronics. This component was chosen because of its size, hardware specifications, and cost. Looking at the size, as stated earlier, the cable that will be connected to the system does not have any extra hardware or conversion; it is just the cable head. This makes it easier for the enclosure, as there will be a smaller hole needed for the cable to fit through.

In comparison to a converter that has the conversion block on the cable end instead of the wall end. In regards to the specifications, looking at the specifications for this component it takes in wall outlet power and outputs 5 volts DC power. This is exactly what the system needs as the battery backup charger takes in 5 volts. Lastly, in regards to the cost, as a team, we are at a great spot with our budget and have some room to spend some extra money. However, in a real project, the goal is to get the system to perform the best for as cheap as possible and you never know what may happen. So, finding this component relatively cheap was great for the team. Another aspect of this component is that it has a cable length of 5 feet, which makes our system able to move around and be flexible with what our customers need and want.

4.2.2 Design

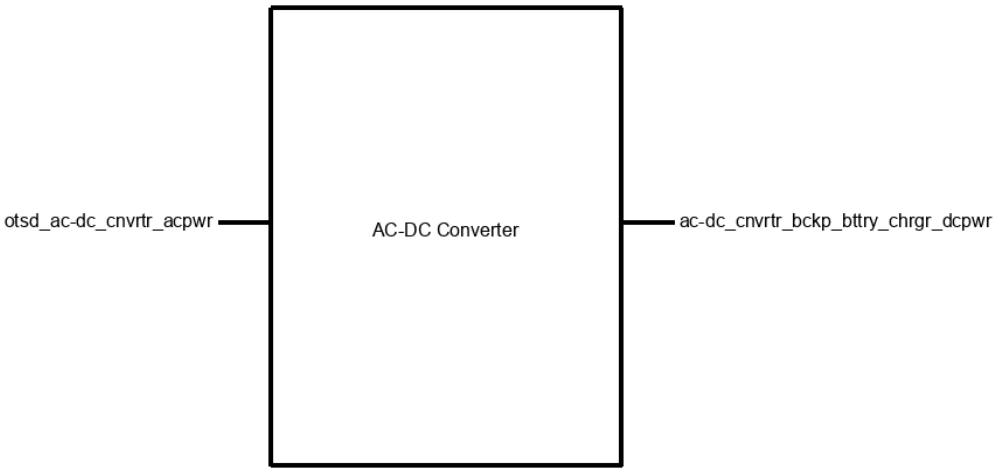


Figure XXX. AC-DC Converter Black Box Diagram

This block, the AC-DC converter, is a purchased block for our system. This block is one of the few cases we have in our system that will not be altered by the group at all. This will come out of the package and go into our system. In the group's mind, building an AC-DC converter was not worth our time and it would be better spent in other places. This block will not have any code or pseudocode as it is just a voltage conversion device. Currently, the group has not yet made a decision on the specific AC-DC converter we are going to purchase

so at this point in time there are no schematics to present. For this converter, we will have a single input of AC power from a wall outlet, that will be converted into DC power and dispersed to the system so it can operate. What is needed for this block to work is the physical device as well as a power cord that can plug into a 15 Amp 120VAC outlet.

4.2.3 General Validation

For this block, the needs of the system are met very easily. The system needs to have an outside source of power. It can not operate on the backup batteries alone. So, we have an AC-DC converter which will provide that outside power. This power will be used to charge our batteries and provide power to the system. Because there are not really any design components besides picking the specific device that works for our system. As many different components will fulfill this requirement we just need to do some research as a team and figure out which one will work the best for us. This can be through different size constraints or how cost-effective the device is.

4.2.4 Interface Validation

TABLE XXX

AC-DC Converter Interfaces

Interface Property	Why is this interface this value?	Why do you know that your design details <u>for this block</u> above meet or exceed each property?
otsd_ac-dc_cnvrtr_acpwr: Input		
Ipeak: 1A	The peak current according to the data-sheet is a max current of 1 Amp.	Since this is a purchased component, and the current is coming from a wall outlet, how this device performs is not up to us as a team but up to the manufacturer and the electrician who set up the outlet.
Inominal: 0.51ma	The nominal current for this interface is based on whatever load it is connected to. Since this is connected to the MCP chip	Based on the datasheet for the Microchip technologies MCP chip the nominal current it can take is the 0.51ma described in the value.

Vnominal: 120V	The nominal voltage from a general wall outlet is 120VAC. According to the Home Depot website, this is the case.	Most wall outlets have this nominal voltage value. Some are slightly larger and some are slightly smaller. But, in general, this is the case. I know this block will accomplish this because of the datasheet for the converter.
ac-dc_cnvrtr_bckp_bttry_chrgr_dcpwr: Output		
Ipeak: 1.5ma	Looking at the datasheet for the Microchip Technology datasheet for the MCP chip the maximum current it can take is 1.5ma.	I will know that my design fits this property by ensuring the device chosen to fulfill this block has this specification. That its max output current is 1.5ma. This can also be done by testing the output current once the device is purchased with a multimeter or similar device.
Inominal: 0.51mA	Looking at the datasheet for the Microchip Technology datasheet for the MCP chip, the nominal current value is the value listed.	I know that this component will work because this is an established company with credibility. The part should match the datasheet in its performance.
Vmax: 6V	Looking at the datasheet for the battery backup charger the maximum voltage it can take is 6V.	I will know that my design fits this property by ensuring the device chosen to fulfill this block has this specification. That its max output voltage is 6V. This can also be done by testing the output voltage once the device is purchased with a multimeter or similar device.
Vmin: 3.75V	Looking at the datasheet for the battery backup charger the minimum voltage it can take is 3.75V.	I will know that my design fits this property by ensuring the device chosen to fulfill this block has this specification. That its max output voltage is 3.75V. This can also be done by testing the output voltage once the device is purchased with a multimeter or similar device.

4.2.5. Verification Plan

1) Plug the power cord into the wall and into the AC-DC converter.
2) Grab a device that can test voltage and current. This could be a DMM or an oscilloscope.
3) Check the voltage across the input terminals and ensure that the input voltage is between the minimum and maximum of 100V to 200V.
4) Check the input current and ensure that the current is less than 20A
5) Check the voltage across the output terminals and ensure that the voltage is in between the minimum and maximum voltage values.
6) Check the current and ensure that it is less than the Ipeak value states in the property.

4.2.6 References and File Links

4.2.6.1 References

[1] "Arduino Mega 2560 Rev3," *Arduino Official Store*. https://store.arduino.cc/products/arduino-mega-2560-rev3#:~:text=Overview (accessed Dec. 08, 2023).

[2] "ESP32 series - mouser electronics," Mouser Electronics, https://www.mouser.com/pdfDocs/esp32_datasheet_en.pdf (accessed Dec. 8, 2023).

[3] "Electrical Outlet Types," The Home Depot, 2020. https://www.homedepot.com/c/ab/electrical-outlet-types/9ba683603be9fa5395fab904ae3e00b

[4] "How Much Electrical Power Do You Need? | Mississippi State University Extension Service," *extension.msstate.edu*. https://extension.msstate.edu/publications/information-sheets/how-much-electrical-power-do-you-need#:~:text=Standard%20wall%20outlet%20voltage%20is (accessed Dec. 08, 2023).

[5] MCP73831/MCP73832 Data Sheet - Microchip Technology, ww1.microchip.com/downloads/en/DeviceDoc/MCP73831-Family-Data-Sheet-DS20001984H.pdf. Accessed 29 Jan. 2024.

[6] "Pololu - Wall Power Adapter: 5VDC, 1a, 5.5×2.1mm Barrel Jack, Center-Positive." Pololu Robotics & Electronics, www.pololu.com/product/1469. Accessed 28 Jan. 2024.

4.2.6.2 File Links

Arduino Mega 2560 Rev3 — Arduino Official Store

Battery Charger Microchip Datasheet

ESP32 Microcontroller Datasheet

MCP Microchip Controller

Pololu AD-DC Converter

Standard Wall Outlet Current Range

Standard Wall Outlet Voltage Range

4.2.7 Revision

Date	Revision
3/14/2024	Joseph Conrow: Adjusted uniform formatting and entered into Project Overview
1/28/2024	Andrew Zellner: Final updates on description and file links
1/26/2024	Andrew Zellner: Updated interfaces and properties based on feedback. Added references to the new datasheets that are involved with this block.
1/25/2024	Andrew Zellner: Updated description and added more detail.
12/7/2023	Andrew Zellner: Added and filled out each section from 1 through 6 for this block.
12/7/2023	Andrew Zellner: Document Created, initial import done from the student portal.

4.3 Alarm Set Interface

4.3.1 Description

The alarm set interface will take care of making the internet-based website configure the alarm clock. This interface is the area on the website that allows the user to set an alarm as well as enable or disable the snooze or alarm reset button on the physical device for any alarm that is set. This will mostly be achieved using coding and can be considered a coding block. This will also be achieved through the communication from the website to the microcontroller and then the microcontroller will communicate with the rest of the system. This block also accounts for the credential system in place for accessing the website. This user interface will have one input interface and one output interface. The input interface is the user input from the website and the output interface is a signal sent to the microcontroller based on the user input. The speed and frequency of these signals are based on the general concept that you only need to check for

an alarm/ button change once every minute. Our alarm clock does not have the capability to be set by seconds. This decision has been made through market research as our customer does not need this. As well as the fact that checking each second would cause a large draw in power and would create difficulties in the system in the way of accomplishing our system requirements. Something else that will be sent to the microcontroller is user credentials. These credentials will be set up with the specific device and need to be entered successfully in order to enter the alarm set interface. To describe the properties of this block, there are two main properties of this block. The first one is an outside source of input as an input to the block. This input will be through mouse and keyboard and will be the user entering the time of an alarm as well as using their mouse to select if they want to turn on or off the user buttons. The second property is a signal that is sent from the interface to the microcontroller. The specific signal that is sent is for the time of alarm that is to be set as well as any information that is sent to the website from the user. That is the contents of the signal. The message of the signal will include what button is being acted on and what action to take on the button. Because these two signals will be sent at different times they are classified as two different input properties. The speed of each message is outlined in the detailed section about the properties, however, it is useful for understanding but not for proving through verification and validation. The website will be implemented through an AWS cloud-based server. This will be accomplished by purchasing a domain through AWS and hosting our page on that server. The domain of the website (listed in file links), is the area where the user will input to the block. This information will then be sent and stored in the database. Where this database will be accessed by the microcontroller to get the user-inputted information.

4.3.2 Design

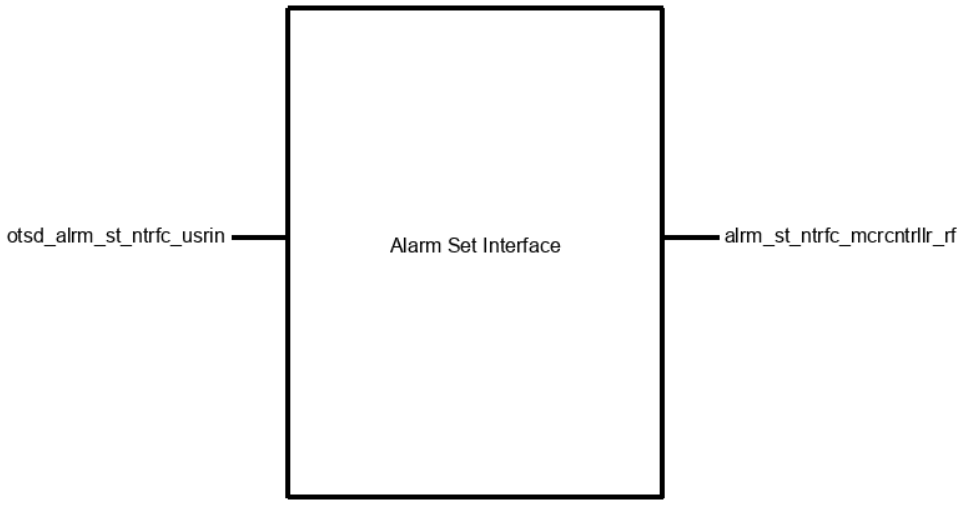


Figure XXX: Alarm Set Interface Black Box Diagram

This block is very theoretical as this is the interface from the setting of the alarm and its communication to the physical device. The alarm set interface is going to be completely code-based. There is no physical aspect to this block, only the input and outputs are physical devices; the block itself is a nonphysical interface. This interface can be inferred through some basic pseudo-code. In general, it will need to take input from a keyboard or mouse for the time that the alarm is going to go off. It will need to send signals to the microcontroller that account for the specific time for the alarm and the alarm settings. These are split into two different signals as they can be sent at different times they do not always need to be sent together. At this point in time, the alarm set interface is finished so no pseudo-code is needed since the code for the interface satisfies the needs of the system. The files as well as the URL for the interface are listed in the File Links section of this document.

4.3.3 General Validation

This specific design meets the needs of this block because it will take input into the specific alarm time as well as the status of the enabling of the two physical buttons on the physical device. Of course, the pseudo-code at this stage is still very new and a very rough draft of what will actually be in the final design. But it provides a good basic outline of what is going to happen. The code will have an interface where the user can enter values. The user will input, and the code will make sure the input is valid, and send signals with the information that was generated by the user input. I chose this type of design because it is simple and will get the job done. This design will be implemented through the already existing website and will serve as a way for the user to enter information on the website. Of course, since this is still in the early stages I do not know how other blocks will interact with this one. Specifically, I do not yet know how the microcontroller will need the signals to be sent so this section is not complete yet. One thing that is pretty clear that will be implemented is how the interface will work. To ensure that user input is valid and easy to send to the microcontroller, we will have two sections that the user is prompted to click on. These two sections will represent hours and minutes where upon clicking all possible integer values will be displayed.

4.3.4 Interface Validation

TABLE XXX

Alarm Set Interface Interfaces

Interface Property	Why is this interface this value?	Why do you know that your design details <u>for this block</u> above meet or exceed each property?

otsd_alrm_st_ntrfc_usrin: Input		
Other: When the user clicks on the clock displayed on the screen, a slider for each hour, minute, and AM/PM shows on the screen.	Since the system will not have alarm times by the second and only by the minute, user input only needs to be checked every minute. The decision to only have minute alarm times was heavily influenced by the team's market research.	Through the database where the website information is stored. The data is gathered once each minute. Because of how the database is set up currently, this property is satisfied.
Type: The type of this input will be mouse mouse-clicking/ keyboard stroke entry.	Based on the style of interface we are implementing it will be easiest to have the input through a mouse click/keyboard stroke. The user will click around to access the different entry boxes and then type into the boxes corresponding to what's required.	On the current website, the code is set up where this is allowed in the different input sections. For entering the username and password, and clicking the clock to change the current alarm time on the screen.
Usability: 9 out of 10 users can use and understand this interface.	This is a very simple interface and the team and I believe that 9 out of 10 people will be able to understand and use this interface.	Through the verification process, this property has been validated. Through two separate occasions, 9 people were able to understand and use this interface.
alrm_st_ntrfc_mcrcntrllr_rf: Output		
Messages: The transmitted data is the time of alarm that is to be set.	This interface is this value because of the general pseudocode that is used to accomplish this as well as the flow that our code has.	The design will fulfill this property because of the code that is currently implemented. Currently, the code supports the communication between the ESP32 and the website relaying information about the alarm times that are set.
Messages: A message transmitted is the enable/disable for the alarm reset button	Because we are dealing with a simple on or off, we want the value sent to be a boolean value as this is the most simple way to show that something is on or off or true or false. The message could also be an integer	The design will fulfill this property because of the code that is being implemented. Currently, communication about the status of the alarm reset button is implemented.

	representing the time of the alarm.	
Other: A confirmation message is displayed to the screen once the information of the alarm alarm is successfully sent to the microcontroller	A simple yes or no, or a box checked or un-checked is a very simple way for the user to be able to turn on or off the buttons. This is the way it is so the interface is kept clean and simple as well as easy to send data to the microcontroller.	I know that the design will meet the expectations of this property because the code currently supports this feature.

4.3.5 Verification Plan

1) Power on a device that can access the internet
2) Go to http://54.190.183.82
3) Select “Configure Alarm”
4) Using a mouse or a touch screen click on the clock in the “Set Alarm Time” area
5) Confirm that a section for hours, minutes, and AM/PM is displayed on the screen
6) Choose a hour, minute, AM/PM
7) Click away from the “Set Alarm Time” area
8) Click in the username box and type in the given username
9) Click in the password box and type in the given password
10) Click the green “Set alarm button”
11) Check to see if a confirmation message shows at the top of the screen.
12) Check to see if it includes the specific alarm time Click “Okay”
13) Check to see if another message is displayed to the screen showing a 1 or 0 representing the status of the reset toggle box.

4.3.6 File Links and References

4.3.6.1 References

[1] “ESP32 Series Datasheet Including.” Available: https://www.mouser.com/pdfDocs/esp32_datasheet_en.pdf

4.3.6.2 File Links

ESP32 Microcontroller Datasheet
Alarm Set Interface Web Page
Code Files

4.3.7 Revision Table

Date	Revision

3/14/2024	Joseph Conrow: Adjusted uniform formatting
3/12/2024	Andrew Zellner: Added entire section to document.

4.4 Enclosure

4.4.1 Description

This block involves designing, building, and implementing the enclosure into our system. This will involve taking into account compact convenience as well as safety objectives. Mainly thinking into the safety side, especially for children interactions, the sharpness of edges, fragility, and electrical exposure will be all factors of the enclosure's design. The main purpose of the enclosure is to ensure the safe and compact nature of a solidly built object that the project's main components will be attached to and associated with. It will be printed using plastics and a 3D printer to encase the stand alone base of the alarm clock system, excluding, in which the system connects to the web interface wireless. It is meant to provide structural support and maintain resistance towards damages made to the internals. The material chosen to execute this purpose will be PLA, as it is easily accessible, cheap, workable, relatively strong, quick to print, and covers all requirements with decency. The color of choice is non-dependent and can be whichever color is currently maintained by the team. The mounting points that will be included within the enclosure, will be heavily dependent on the exact final specifications of the other blocks and functionalities of the device as a whole. The measurements and fittings will be crucial to a successful protection enclosure. The speaker that will emit the sound will also be protected despite the need for sound waves to transmit. This will be solved through the use of a cage-like mesh design, creating a barrier between the speaker and the outside world, while allowing void gaps in the material to let sound waves travel through without turbulence. The hole for the input power to be inserted, will be a direct hole, with the direct interface recessed into the enclosure's wall, keeping it further from reach of the user or environment. The shape of the design will be smooth and very gradual, minimizing edges and optimizing potential safety concerns. The integration process involves the incorporation of shelving-style slots within the enclosure, specifically crafted to accommodate PCB (Printed Circuit Board) based components. This strategic design choice facilitates a secure and organized placement of the electronic elements, contributing to the overall structural integrity of the system. In addition to the slots, a sophisticated approach is employed, utilizing threaded inserts that can be melted into the PLA (Polylactic Acid) enclosure material. This innovative combination of shelving-style slots and threaded inserts serves a dual purpose. Firstly, it enhances the stability of the mounted components, providing a robust foundation that withstands the dynamics of daily use. Secondly, it ensures a secure fit, preventing any inadvertent displacement of essential elements. The threaded inserts, with their

unique ability to meld seamlessly into the PLA material, offer a reliable and durable connection. Moreover, this design methodology facilitates repetitive removal of components, which is particularly advantageous during the prototyping phase. It allows for iterative testing and modifications without compromising the overall structural integrity of the enclosure. This adaptability is invaluable in the development process, enabling the refinement of the alarm clock system based on user feedback and evolving requirements. The dual emphasis on security and the slot method contributes significantly to the overall resilience of the entire system. This robust design approach ensures that the alarm clock system remains resistant to various types of handling and rough usage. Whether in a household with energetic children or in a dynamic environment, the secure mounting mechanism and thoughtfully engineered slots mitigate the risk of damage, enhancing the longevity and reliability of the entire system. As a result, users can confidently interact with the alarm clock, knowing that its internal components are securely housed within a design that prioritizes both functionality and durability.

4.4.2 Design

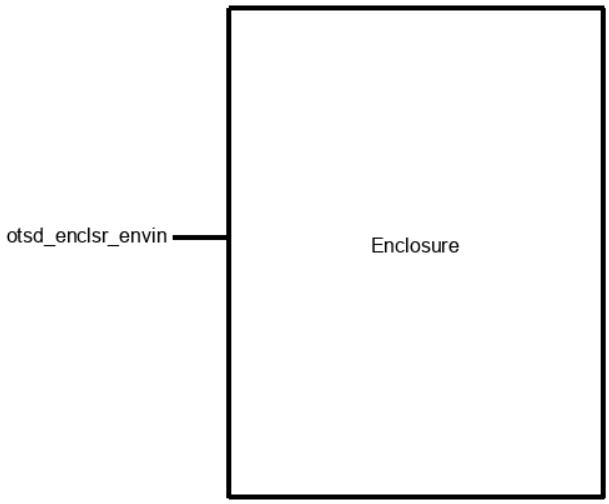


Fig XXX. Enclosure Black Box Diagram

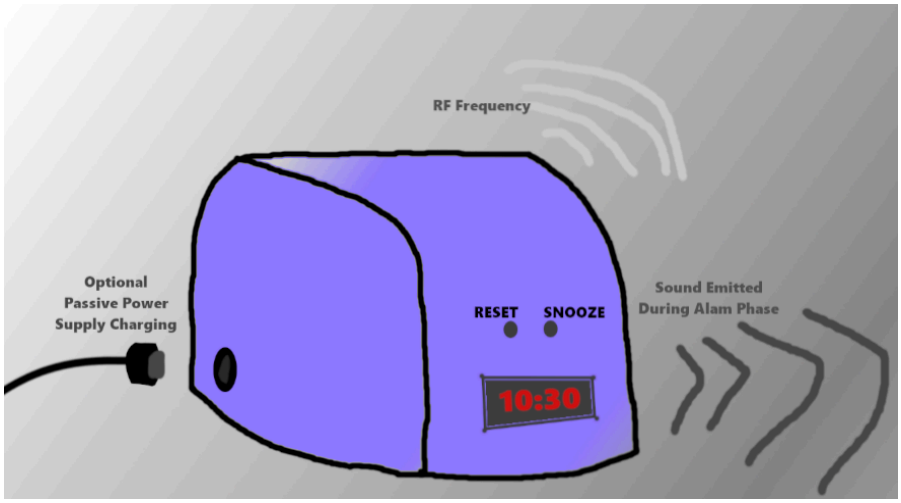


Fig XXX. Rough Draft Enclosure Sketch

The design is not involved within any sort of I/O, as it only allows the physical pass through of I/O and interactions of such other blocks, while not contributing to their functionality. The internals contain the microcontroller, the buttons, the display, the speaker, the charger, and the battery, all working in conjunction and around the enclosure.

The simple and sleek design of the enclosure in Fig XXX. is that it allows the transparent operation of the complimentary blocks, all while containing them and making the product attractive and safe. The rounded edges keep the model compact as well as structurally sound. The lack of pointed corners also keeps it from damaging its surroundings and prevents any possibilities of direct piercing injury to users.

4.4.3 General Validation

The purpose of the enclosure is quite straightforward, and the complexity of the task to be handled is overall simple, yet it can be taken in a way that it has much potential for improvement and additional purpose. The functionality is completely involved by the surrounding of the rest of the device. However, there is room for revision if there so happens to be a convenient feature that can be included in the future. Currently, the enclosure helps keep any type of exposed electricity risk eliminated. The only external wire that will be used will be contained within a save standard charging plug. The enclosure is also internally customized according to the footprint, shape, and layout of the components. This is allowable and adaptable.

4.4.4 Interface Validation

In this specific block, there is no technical interfacing between any other blocks. It is a stand alone functionality that contains physically yet does not interact with other components. The motivations and goal for the block is still yet to be explained. These are such not to be able to pierce skin and also to convert

any type of exposed electrical wires. With the only protruding components being the dc supply signal from the ac-dc converter as well as the buttonholes and the graphical screen to be visible.

TABLE XXX

Enclosure Interfaces

Interface Property	**Why is this interface this value?**	**Why do you know that your design details <u>for this block</u> above meet or exceed each property?**
Otsd_enclsr_envin: Output		
Other: The walls of the enclosure will stay connected when dropped from a height of 3 feet.	The system is meant for children so in case they throw the device we want to make sure it stays together.	Through my block verification this was proven through a drop test. So, this property has been met by the current system.
Other: 9 out of 10 users will say it is hard to open the enclosure.	Because this is a device for children. We want the enclosure to be hard to open to make sure children don't get into the electrical system.	Through my block verification this was proven through a user test. So, this property has been met by the current system.
Other: The maximum dimensions of the enclosure are 12" x 12" x 12" (L x W x H).	Through our market research, most people have a size for their alarm clock less than or equal to a foot by a foot by a foot. So, we want to make sure our enclosure is to this standard.	By measuring the system with a measuring tape, the size is 8x8x5 which is in fact less than 12x12x12.

Specifies and address the appropriate interface requirements and reasoning

4.4.5 Verification Process

As there are no technical specifications and nominal values to measure, it is important to prioritize the overall user experience of this block overall.

1) The safety factor should be tested.This is done by holding the enclosure and rubbing it on a test subject's skin.
2) Ensure there is not any damage done due to absence of points on the model
3) The next objective to accomplish is to ensure that any accompanied electrical wires are not exposed from un-insolation.

4) The only cable protruding from the enclosure should be confirmed to only be the external battery charger plug.

4.4.6 References and File Links

4.4.6.1 References

Inferences to sharpness for skin lacerations

4.4.6.2 File Links

N/A

Revision Table

Date	Revision
3/14/2024	Joseph Conrow: Adjusted uniform formatting
3/12/2024	Andrew Zellner: Updated all aspects of this section to match the student portal.
12/7/2023	Joseph Conrow: Initial content creation

4.5 Microcontroller

4.5.1 Description

The microcontroller will be the main interconnected piece of the entire project. As it is the main communication device and interfaces between all the other functional blocks. It is crucial to determine the appropriate limitations and functionality that the specific model and module chosen has and can possibly develop. The overall final decision model we chose was the ESP32-WROOM–32 module, AKA “DOIT ESP32 DevKit V1 Wi-Fi Development Board”. This way, with a pre-built chip design, the ESP32 chip itself can easily be accessed by our other components very easily using pin headers. The power supply pin also was able to regulate the specific power demands of our design. As the voltage regulator on the module allowed us to focus all of our power consumption from a single component source, as the method of supply to a standardized system voltage, that is, 5 Volts.

Some of the more specific rationale we used to pick this board were guided by their compatibility in communication protocol in reference to our selected project graphical display component, speaker component, internet component, and control components. These modes of communication involved SPI, I2C, PWM, WiFi, and DC signals. These are all protocols that the module chosen can provide with the ESP32 chip soldered on. The power draw of 3.3 volts is also something that the display is able to run off of, which the ESP32 can supply, allowing all of the communications and connections of the display to be rooted to the microcontroller. The connections to the web interface, and thus the user/clock/alarm server database is also to be accessed solely via the

microcontroller. The speaker is also able to run and power distinctly sourced from the microcontroller, giving it the correct signals and internet connection to source a radio sound source if such audio source is chosen. The buttons also are a direct interworking interface for the microcontroller, as it can supply itself with a signal voltage and a receiving pin to recognize the state of such user buttons.

The inner workings of the microcontroller bring all of these components together to work in harmony. By receiving signals from the database, the display can be correctly displaying the correct time value. It conducts the appropriate sound values and timings for the alarm based on the stored website alarm settings. The microcontroller will be sustainable within the state of offline connection as well. One of the project objectives is to deliver the same functionality despite the source of power or internet connection. The small physical footprint of the chosen microcontroller is also desirable and easy to fit within our designed product enclosure. As this chosen microcontroller module is apt to function within each specific interface of the project components, the ability of it is also able to manage, store information, and send it where it needs to go, in a proper and useful manner, allowing the user to experience a clean, and seamless interaction with the product, regardless of which end component they are conducting interaction with, whether that be the graphical clock display, the website interface, the control buttons, or the audio of the speaker.

4.5.2 Design

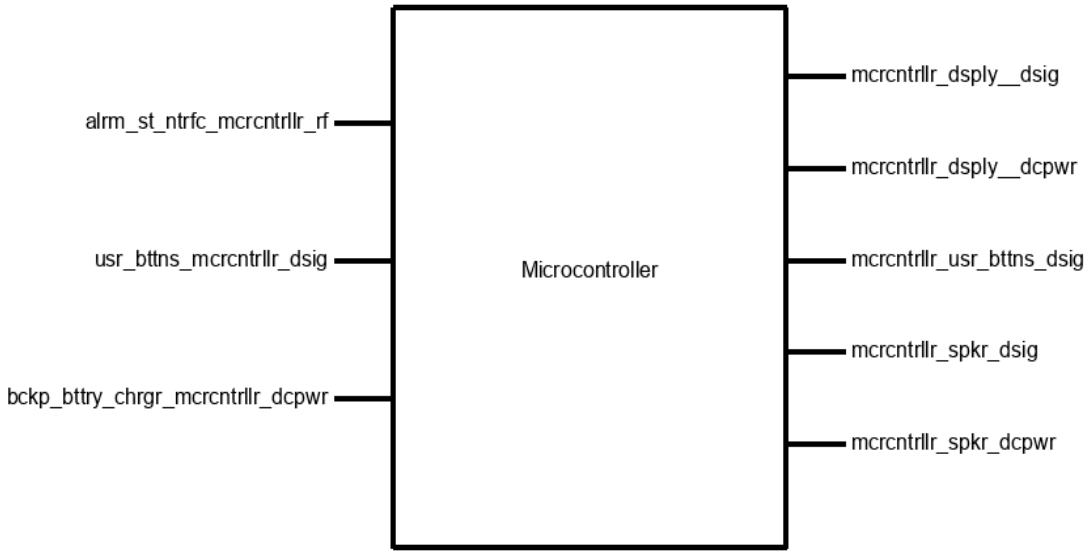


Fig. XXX. ESP32 Black Box

The overall design and implementation steps are based on component specifications. This includes the power demands and communication protocols of each interface. The black box diagram simply refers to the interface setup for inputs and outputs. The ESP32-WROOM-32 Module Pinout explains the selected

module's functionality, and interfacing capabilities, in reference to needs of each component. The Component Connection Pinout is meant to explain the physical connections between the microcontroller and its other components, proving capabilities to maintain connection between each, properly. The ESP32-WROOM-32 Chip DC Output figure is a brief excerpt from the microcontroller module in explanation for proof of the power demands for the ESP32 chip itself, which is comparable with the power regulation of the chip's accompanied module.

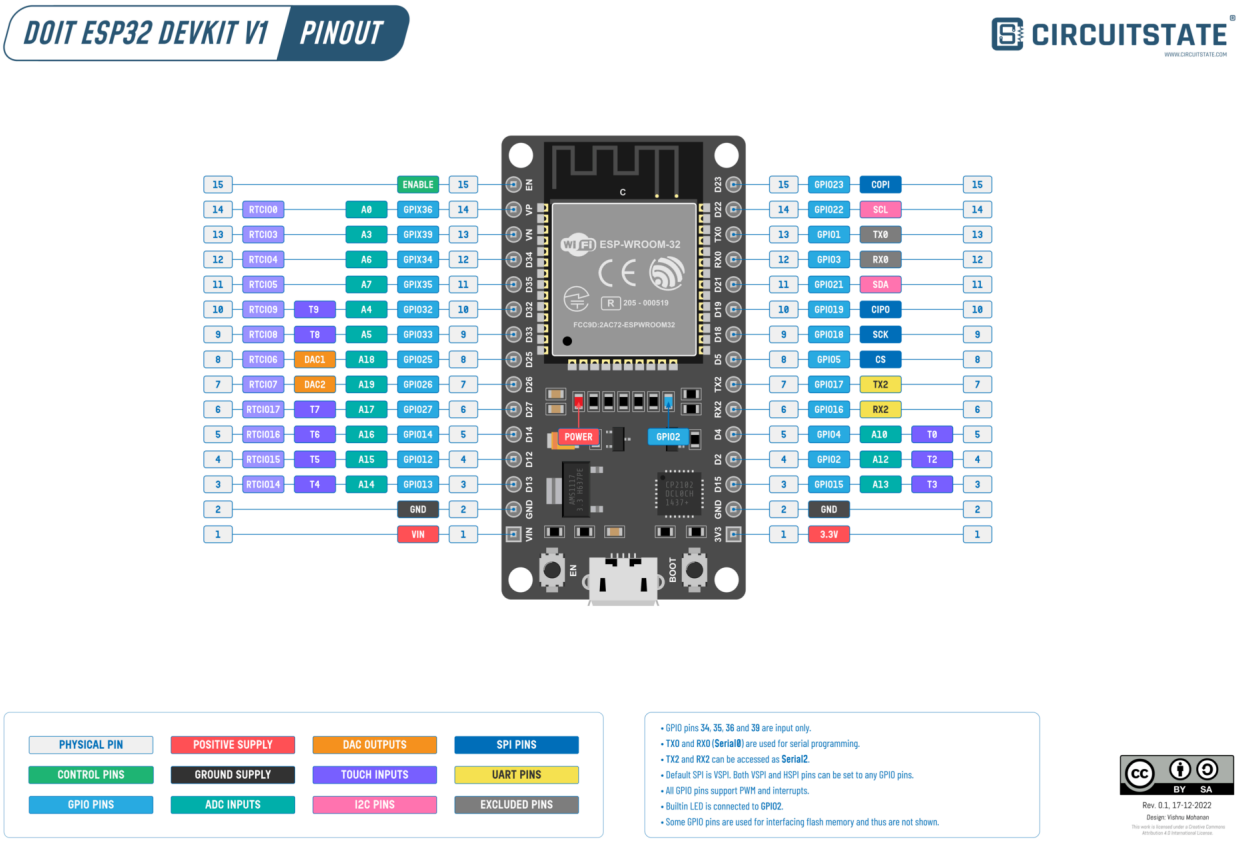


Fig XXX. ESP32-WROOM-32 Module Pinout [1]

Pin	Type	Module Use
1	System	NC
2	Input, System, RTC_IO, Analog, GPIO	Button1_Source
3	Input, System, RTC_IO, Analog, GPIO	Button2_Source
4	System, RTC_IO, Analog, GPIO	Button3_Source
5	System, RTC_IO, Analog, GPIO	Button1_Input
6	System, Touch, RTC_IO, Analog, GPIO	Button2_Input
7	System, Touch, RTC_IO, Analog, GPIO	Button3_Input
8	DAC, RTC_IO, Analog, GPIO	NC
9	DAC, RTC_IO, Analog, GPIO	NC
10	Touch, RTC_IO, Analog, GPIO	NC
11	SPI_CLK, Touch, RTC_IO, Analog, GPIO	NC
12	SPI_MISO, Touch, RTC_IO, Analog, GPIO	NC
13	SPI_MOSI, Touch, RTC_IO, Analog, GPIO	NC
14	GND	NC
15	VIN	NC
16	3V3	Display_Vin, Amp_Vin
17	GND	Display_GND, Amp_GAIN, Amp_GND
18	SPI_CS0, Touch, RTC_IO, Analog, GPIO	Display_CS
19	Touch, RTC_IO, Analog, GPIO	Display_DC
20	Touch, RTC_IO, Analog, GPIO	NC
21	USART_2TX, GPIO	NC
22	USART_2RX, GPIO	Amp_DIN
23	SPI_CS0, GPIO	NC
24	SPI_CLK, GPIO	Display_SCK
25	SPI_MISO, GPIO	Amp_LRC
26	I2C_SDA, GPIO	Amp_BTC
27	USART_0TX, GPIO	NC
28	USART_0RX, GPIO	NC
29	I2C_SCL, GPIO	NC
30	SPI_MOSI, GPIO	Display_MOSI

Fig XXX. Component Connection Pinout

Table 5: Absolute Maximum Ratings

Symbol	Parameter	Min	Max	Unit
VDD33	Power supply voltage	–0.3	3.6	V
I_{output} [1]	Cumulative IO output current	-	1,100	mA
T_{store}	Storage temperature	–40	105	°C

1. The module worked properly after a 24-hour test in ambient temperature at 25 °C, and the IOs in three domains (VDD3P3_RTC, VDD3P3_CPU, VDD_SDIO) output high logic level to ground. Please note that pins occupied by flash and/or PSRAM in the VDD_SDIO power domain were excluded from the test.
2. Please see Appendix IO_MUX of *ESP32 Datasheet* for IO's power domain.

5.2 Recommended Operating Conditions

Table 6: Recommended Operating Conditions

Symbol	Parameter	Min	Typical	Max	Unit
VDD33	Power supply voltage	3.0	3.3	3.6	V
I_{VDD}	Current delivered by external power supply	0.5	-	-	A
T	Operating ambient temperature	–40	-	85	°C

5.3 DC Characteristics (3.3 V, 25 °C)

Table 7: DC Characteristics (3.3 V, 25 °C)

Symbol	Parameter		Min	Typ	Max	Unit
C_{IN}	Pin capacitance		-	2	-	pF
V_{IH}	High-level input voltage		0.75×VDD[1]	-	VDD[1]+0.3	V
V_{IL}	Low-level input voltage		–0.3	-	0.25×VDD[1]	V
I_{IH}	High-level input current		-	-	50	nA
I_{IL}	Low-level input current		-	-	50	nA
V_{OH}	High-level output voltage		0.8×VDD[1]	-	-	V
V_{OL}	Low-level output voltage		-	-	0.1×VDD[1]	V
I_{OH}	High-level source current (VDD[1] = 3.3 V, V_{OH} >= 2.64 V, output drive strength set to the maximum)	VDD3P3_CPU power domain [1, 2]	-	40	-	mA
		VDD3P3_RTC power domain [1, 2]	-	40	-	mA
		VDD_SDIO power domain [1, 3]	-	20	-	mA

Fig XXX. ESP32-WROOM-32 Chip DC Output [2]

Pin Name	Function
VIN	The input of the 3.3V positive voltage regulator. Supply voltage in the range of 4 to 12V.
3.3V	Output from the voltage regulator. You can also supply 3.3V to this pin if you have one. But do not supply both VIN and 3V3 together.
GND	Ground (Negative) supply pins.
ENABLE	This is the reset pin. Connecting this pin to GND will reset the ESP32. This pin is normally pulled-up. The EN button will pull it LOW when you press it.

Fig XXX. ESP32-WROOM-32 Module DC Supply [1]

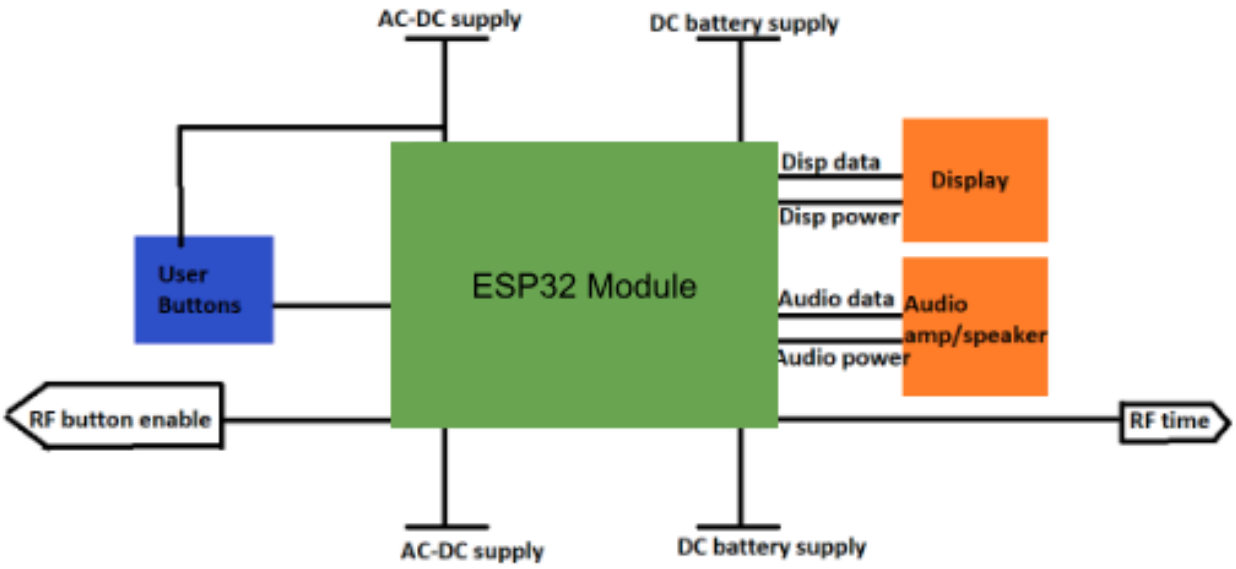


Fig XXX. ESP32 Module Layout

4.5.3 General Validation

The reason that a microcontroller in general was chosen to communicate between all of the components of the alarm clock was that they are quite general purpose and can be utilized in a multitude of ways. The ESP32 was chosen in this role as the main chip due to its versatile input and output setup. It can communicate with the components in SPI, I2C, PWM, and WiFi, just as needed in this project. The display requires SPI and PWM, the speaker requires I2C, and the web server requires WiFi to communicate with.The power output that is capable for the microcontroller is also sufficient for the other components of the project, and can also simply share the load its power source as well.

The compatibility that the ESP32 has with the Arduino framework also provided much software support and the necessary libraries and development environment to complete the product. The community support and content that revolves around the Arduino is something that is hard to compete with, allowing

the project to incorporate clean and smooth incorporation of other creator made libraries to work alongside the custom current project based libraries simultaneously. Thus, completing the proper desired functionality, interconnecting the other components into a fully functional system, capable of accomplishing the promised tasks.

The pinout that the project is designed around is available, and the abundance of pins allows the components attached to the microcontroller to not cause any conflicting needs. This versatility of options is also accompanied by the microcontroller module's ability to take in a variety of voltages. While the chip itself is sustained by a specific voltage of 3.3V, the module incorporates a regulator system to allow voltage to range from 4-12 volts. That way, our battery backup system component can supply the configured voltage of 5 volts.

The code that will be associated with this block will be incorporated into different modular libraries, that each component will be sectioned into. That way the overall code development will be much cleaner, useful for future improvements and quick to fix or change. The reason this is possible is due to the openness of the Arduino software that the ESP32 utilizes, allowing the users to write their own C++ libraries and including them into the flashed program into the ESP32 chip. We chose this framework because it also is useful in dividing labor into each of the developers, creating a simple way to program, while having no dependency on other blocks, creating a versatile software layout. The logic will be simpler overall during integration. The high efficiency and low amount of power draw that the ESP32 will draw is also useful, as the goal of the project is to maintain efficiency and create a reliable backup power source that can maintain nominal function in a non-external power situation.

4.5.4 Interface Validation

TABLE XXX

Microcontroller Interfaces

Interface Property	Why is this interface this value?	Why do you know that your design details <u>for this block</u> above meet or exceed each property?
alrm_st_ntrfc_mcrcntrllr_rf : Input		
Messages: The transmitted data is the time of alarm that is to be set.	The message is simply to indicate successful transmission to the user.	The system is implemented meets such, and verification was completed successfully.

Messages: A message transmitted is the enable/disable for the alarm reset button	The message is simply to indicate successful transmission to the user.	The system is implemented meets such, and verification was completed successfully.
Other: A confirmation message is displayed to the screen once the information of the alarm alarm is successfully sent to the microcontroller	The message is simply to indicate successful transmission to the user.	The system is implemented meets such, and verification was completed successfully.
usr_bttns_mcrcntrllr_dsig : Input		
Logic-Level: Active HIgh	This signal will be used to translate the buttons state	The ESP32 Data sheet Pg15-16 [2] displays the voltage for sensing and output to fit 3.3V
Vmax: 3.6V	3.3 is the nominal, fitting under the 3.6V window	The ESP32 Data sheet Pg15-16 [2] displays the voltage for sensing and output to fit 3.3V
Vmin: 2.5V	3.3 is the nominal, fitting above the 2.5V window	The ESP32 Data sheet Pg15-16 [2] displays the voltage for sensing and output to fit 3.3V
bckp_bttry_chrgr_mcrcntrllr_dcpwr : Input		
Inominal: 102mA +- 2mA	The calculated summation current draw for nominal operation	The load was tested and verified to not exceed, during load conditions.
Ipeak: 104mA +- 2mA	The calculated summation current draw for nominal operation	The load was tested and verified to not exceed, during load conditions.
Vmax: 5.25V	The microcontroller can handle 4-12V, fitting within the range.	The ESP32 Module datapage [1] shown in the design figure, explains the range fits of voltage input ranges 4-12V

Vmin: 4.75	The microcontroller can handle 4-12V, fitting within the range.	The ESP32 Module datapage [1] shown in the design figure, explains the range fits of voltage input ranges 4-12V
mcrcntrllr_dsply__dsig : Output		
Logic-Level: Bit value of 0 or 1	This is a simple PWM method to communicate	The method is tested and is verified.
Max Frequency: 1Hz	The PWM frequency is based on the microcontroller's maximum.	The screen display Pg 176 [3] shows the frequency as compatible. And the ESP32 Pg 1 [2]
Vmax: 2.4V	This signal is set based on the display signal input voltage	The ESP32 [2] shows the voltage as compatible.
mcrcntrllr_dsply__dcpwr : Output		
Inominal: 68mA	Calculated from display data sheet and nominal draw	The nominal was tested and verified even after estimating checking the voltage rating on the ESP32 data sheet [2]
Ipeak: 83mA	The display datasheet tells the peak ratings	The peak was tested and verified even after checking the voltage rating on the ESP32 data sheet [2]
Vmax: 5.0V	This signal is set based on the display’s input supply voltage	The peak was tested and verified even after checking the voltage rating on the ESP32 data sheet [2]
Vmin: 3.3V	This signal is set based on the display input supply voltage	The minimum was tested and verified even after checking the voltage rating on the ESP32 data sheet [2]
mcrcntrllr_usr_bttns_dsig : Output		
Logic-Level: High	This signal will be used to translate the buttons state	The ESP32 Data sheet Pg15-16 [2] displays the voltage for sensing and output to fit 3.3V

Vmax: 3.6V	3.3 is the nominal, fitting under the 3.6V window	The ESP32 Data sheet Pg15-16 [2] displays the voltage for sensing and output to fit 3.3V
Vmin: 2.5V	3.3 is the nominal, fitting above the 2.5V window	The ESP32 Data sheet Pg15-16 [2] displays the voltage for sensing and output to fit 3.3V
mcrcntrllr_spkr_dsig : Output		
Logic-Level: Active High (I2S)	The speaker demands the protocol.	The ESP32 module has I/O pins for such operation [1]
Other: Digital Low (Max Value): 0.6V	The speaker requires such low voltage for low signal	The ESP32 Data sheet Pg15-16 [2] displays the voltage for sensing and output low fit under 0.6V (0V)
Other: Digital High (Min Value): 1.3V	The speaker requires such low voltage for high signal	The ESP32 Data sheet Pg15-16 [2] displays the voltage for sensing and output high to fit above 1.3 (3.3V)
mcrcntrllr_spkr_dcpwr : Output		
Inominal: 2mA	The speaker draw demands no more than such during nominal operation.	The nominal was tested and verified even after estimating checking the voltage rating on the ESP32 data sheet [2]
Ipeak: 20mA	The speaker draw demands no more than such	The peak was tested and verified even after checking the voltage rating on the ESP32 data sheet [2]
Vmax: 5.5V	The speaker draw demands no more than such during nominal operation.	The peak was tested and verified even after checking the voltage rating on the ESP32 data sheet [2]

Vmin: 2.5V	The speaker draw demands no less than such during nominal operation.	The minimum was tested and verified even after checking the voltage rating on the ESP32 data sheet [2]

4.5.5 Verification Process

1) First verify the arduino is receiving power, as the LED should maintain illumination.
2) Unplug the external ac-dc source
3) Verify that battery is supplying to the arduino as well
4) Next the display interactions will be tested in order to properly verify the next few components. This is done by hard coding the display to output a certain number to display.
5) Check if the number is properly displayed as hard coded.
6) When complete, verify button interactions by only outputting on the display when a button is pressed.
7) Do this for each button
8) Check if the audio is properly playing, thus hardcode a signal to be output into the speaker.
9) Connect the the wifi, and verify connection stability
10) Continuously send signals the the web interface
11) Verify via web interface that the signal is being received.
12) Send data from the web interface to the board and display that information on the graphical display.
13) Verify that the same information is being displayed on the the display

4.5.6 References and File Links

4.5.6.1 References

[1] VISHNU MOHANAN, “DOIT ESP32 DevKit V1 Wi-Fi Development Board – Pinout Diagram & Arduino Reference - CIRCUITSTATE Electronics,” *CircuitState*, Dec. 20, 2022. https://www.circuitstate.com/pinouts/doit-esp32-devkit-v1-wifi-development-board-pinout-diagram-and-reference/

[2] “ESP32-WROOM-32 Datasheet.” Available: https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32_datasheet_en.pdf

[3] A. Industries, “2.8" TFT LCD with Touchscreen Breakout Board w/MicroSD Socket,” *www.adafruit.com*. https://www.adafruit.com/product/1770

[4] “Pololu - Wall Power Adapter: 5VDC, 1A, 5.5×2.1mm Barrel Jack, Center-Positive,” *www.pololu.com*. https://www.pololu.com/product/1469

[5] “MCP73831/2 Features.” Available: https://ww1.microchip.com/downloads/en/DeviceDoc/20001984g.pdf

[6] “Rev. 1. 6 BCD Semiconductor Manufacturing Limited Data Sheet General Description,” 2010. Available: https://www.diodes.com/assets/Datasheets/AP3012.pdf

[7] “LM4875 750 mW Audio Power Amplifier with DC Volume Control and Headphone Switch Check for Samples: LM4875 1FEATURES DESCRIPTION TYPICAL APPLICATION CONNECTION DIAGRAM Small Outline Package (SOIC) Mini Small Outline Package (VSSOP) Top View See Package Number D, DGK Figure 1. Typical Audio Amplifier Application Circuit.” Accessed: Mar. 08, 2024. [Online]. Available: https://www.ti.com/lit/ds/symlink/lm4875.pdf

[8] “LM4875 BOOMER 750mW Audio Pwr Amp w/DC Vol Contro & Headphone Switch datasheet (Rev. C),” *ti.com*, May 2013. https://www.ti.com/lit/ds/symlink/lm4875.pdf (accessed Mar. 09, 2024).

[9] “135694,” *jameco.com*, Jul. 29, 2021. https://www.jameco.com/Jameco/Products/ProdDS/135694.pdf

4.5.6.2 File Links

[10] jconrow00, “IOT-Alarm-Clock/Website Interface at master · jconrow00/IOT-Alarm-Clock,” *GitHub*, 2024. https://github.com/jconrow00/IOT-Alarm-Clock/tree/master/Website%20Interface (accessed Mar. 09, 2024).

4.5.7 Revision Table

Date	Revision
3/14/2024	Joseph Conrow: Adjusted uniform formatting
3/12/2024	Joseph Conrow: Table formatting
3/4/2024	Joseph Conrow: Revamped entire block validation
1/26/2024	Joseph Conrow: Revision with final product decision specs
12/7/2023	Joseph Conrow: Initial content creation

4.6 Speaker

4.6.1 Description

The job of this block is to emit a sound into the physical space around the system. This sound will act as the alarm function of the overall system i.e. the alarm clock. As far as the timing of the sound is concerned, that will all be determined by the microcontroller and not from the speaker block. This speaker block only concerns itself with the manipulation of the input, which is an analog signal. From a technical perspective, this block takes an analog signal from the microcontroller and amplifies this signal enabling it to be audible to the human ear. This analog signal is the sinusoidal wave that needs to be amplified in order to be heard. Once the signal is amplified it will be transmitted to the speaker itself that will emit the proper sound. Amplifying the signal requires a module of its own outside of the physical speaker itself. This design uses a MAX98357A audio amplifier and an 8 Ohm, .2W Jameco speaker. The MAX98357A chip is already integrated into a module that provides the necessary interconnections needed for proper function, as well as jumper pins for any input and/or output. The audio amplifier will take in DC power, a bit clock, byte select, gain, and data. The DC power supply voltage for the audio amplifier will, nominally, equal 3.3 volts. This will be supplied by the microcontroller. The input signal, as previously mentioned, is the audio signal provided by the microcontroller. The DC gain input into the audio amplifier will also be provided by the microcontroller and is limited to the supply voltage—in this case, 3.3 volts. Five of the seven jumper pins are for input into the audio amplifier and will be connected to the microcontroller via female to female jumper wires. The other two pins will serve as output for the speaker. These two connections will be connected with female connections at the output of the amplifier and male connections at the speaker. The two connections at the speaker side will be soldered onto speaker terminals. With a supply voltage of five volts, the amplifier is capable of delivering up to 750 mW of power with a Total Harmonic Distortion (THD) of less than 1%. This will ensure both adequate volume as well as sound quality for the alarm. The way in which the module is designed makes it very easy to input from the microcontroller and output to the speaker. The amplifier itself has great advantages such as high potential volume and sound quality. The supply voltage of the amplifier is 2.5 – 5.5 volts which is compatible with the DAC output of the microcontroller which is up to 3.17 volts. The speaker will be fastened to the enclosure in such a way that it is exposed to the outside of the alarm clock. The MAX98357A uses I2S data to communicate the sound for the speaker to amplify. The gain is tied to the ground for a 12 dB output. The I2S protocol enables the speaker to play a wide variety of sounds from simple tones to complex audio such as music.

4.6.2 Design

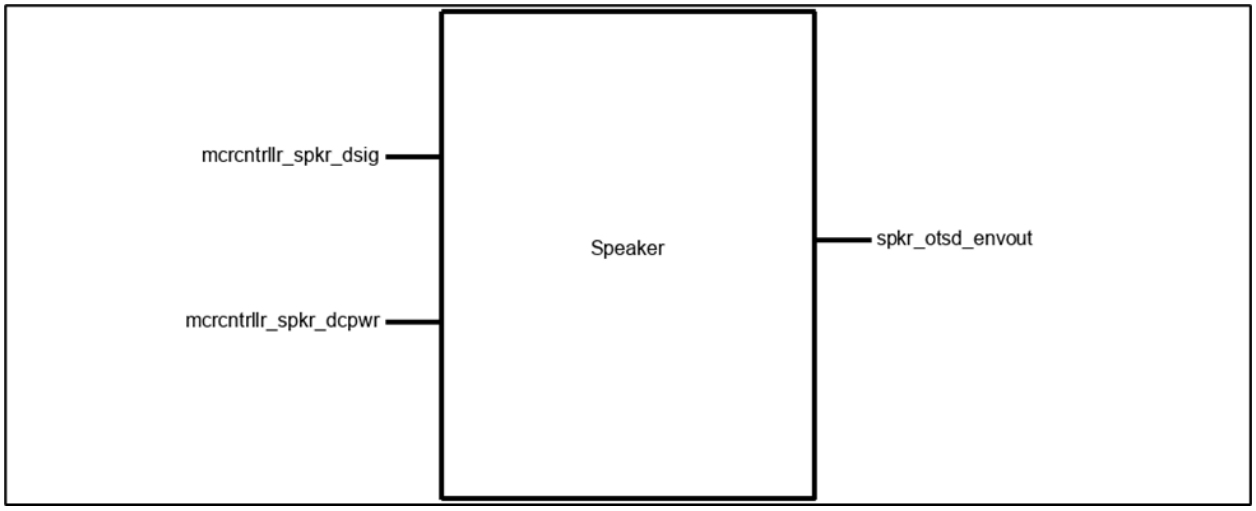


Fig XXX. Speaker Black Box

This block, as previously stated, is composed of the MAX98357A amplifier as well as a speaker shown below in Fig. 1. Due to the demand of the overall system, there is to be a sound emitted when the alarm goes off. The MAX98357A uses I2S protocol to transmit and receive the sound data. This allows much more complex sound to be emitted than if a simple DAC were to be used.

In I2s fashion, the MAX98357A requires several signals to function properly. It needs a bit clock, word select, data, power, and gain. The bit clock (BCLK) is the central clock that the other signals rely on for timing that will connect to pin 26 of the ESP32. The word select (LRC) determines which audio channel the amplifier outputs to and connects to pin 25 of the ESP32. The data (DIN) contains the actual audio information being communicated and connects to pin 22. Power and ground are connected to the 3.3 volt supply from the ESP32. This amplifier was purchased from Adafruit within a breakout board configuration shown below.

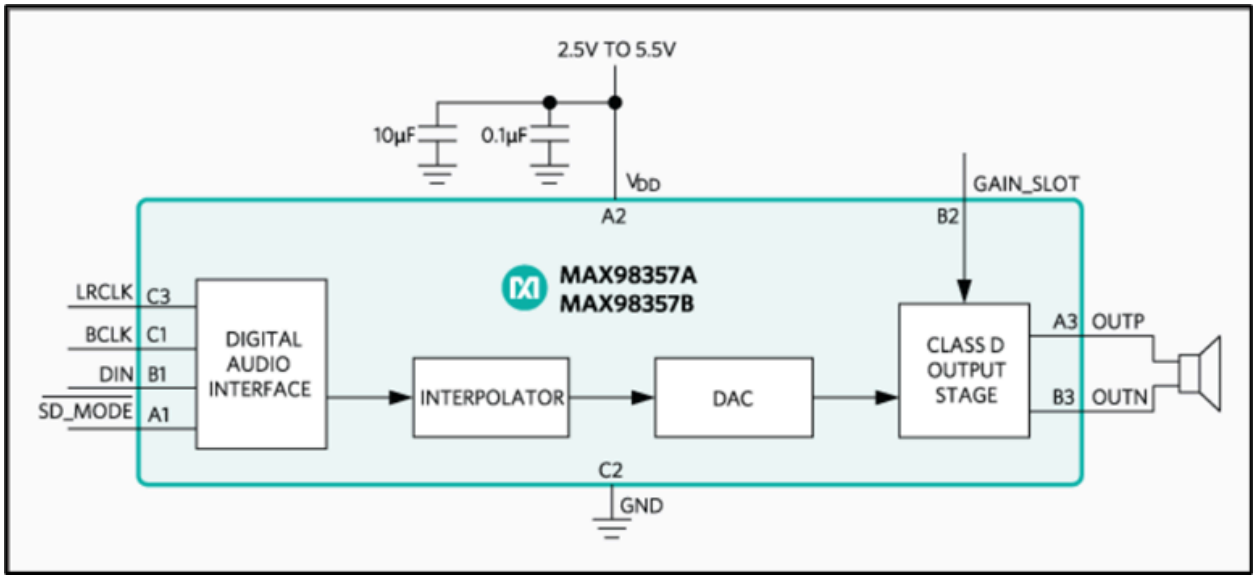


Fig XXX. MAX98357 Block Diagram

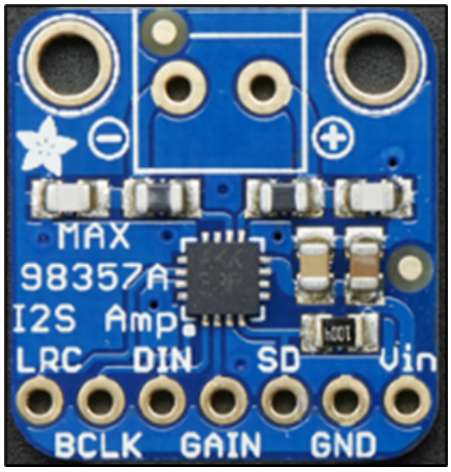


Fig XXX. Adafruit MAX98357A Breakout Board

As can be seen at the top of the board is the pin locations to output to the speaker. This board is easily implemented into the overall system and serves its exact needs. The actual speaker that the amplifier feeds into is a simple 8 Ohm, .2 Watt from Jameco. All inputs and outputs are made using jumper wires.

4.6.3 General Validation

This amplifier and speaker combination meet the needs of thye system very well. Overall, it is a cost-effective, high quality, and versatile option that will ensure the best user experience. It is also highly configurable and easily integrated with the selected microcontroller.

The MAX98357A board makes it very easy to play a wide variety of audio types. The volume of the sound is easily adjustable as well. Due to the I2S interface, playing songs that appeal more to the consumer is available and creates a greater incentive into using the product as a whole. A standard DAC converter was considered for this block. However, the appeal of offering a much greater audio capability was enough to warrant the use of the MAX98357A. One reason this specific amplifier was chosen was due to the popularity of the I2S protocol. It is characteristic of most modern microcontrollers to include the ability of I2S communication making audio transmission much easier. Another great advantage of this amplifier and speaker is the low power consumption. Actively playing sound draws no more than 4mA. Our system is capable of supplying in the 100s of milliamps and this is far under that. Its quiescent current draw is even much less. Additionally, for its incredible capability the amplifier board met the financial restraints of the project.

The benefits of using the selected amplifier and speaker greatly outweigh the downsides. It suits the needs of the system and also features innovative appeal and features to cater to the needs of the user.

4.6.4 Interface Validation

TABLE XXX

Speaker Interfaces

Interface Property	**Why is this interface this value?**	**Why do you know that your design details <u>for this block</u> above meet or exceed each property?**
spkr_otsd_envout : Output		
Other: Max Frequency: 5.5kHz	This was chosen due to the recommendation of the speaker's manufacturer	The amplifier can output sound waves at this frequency. Pg 1 (1).
Other: Signal: Sound	The alarm clock needs an alarm sound.	This is the very function of a speaker.
Other: Min Frequency: 500Hz	This was chosen due to the recommendation of the speaker's manufacturer	The amplifier can output sound waves at this frequency. Pg 1(1).

mcrcntrllr_spkr_dsig : Input		
Logic-Level: Active High (I2S)	This is the way in which I2S functions.	The amplifier utilizes I2S (1).
Other: Digital Low (Max Value): 0.6V	The system is able to meet this requirement.	This is the max value at which a digital zero will read. Pg 7 (1).
Other: Digital High (Min Value): 1.3V	The system is able to meet this requirement.	This is the min value at which a digital one will read. Pg 6 (1).
mcrcntrllr_spkr_dcpwr : Input		
Inominal: 2mA	This nominal current was chosen based on the expected current needs of the system overall	This is the current required by the amplifier when actively outputting sound.
Ipeak: 20mA	This max current was chosen based on the expected current needs of the system overall	Below this value is a safe amount in order to not exceed the demand allotted to this block.
Vmax: 5.5V	This max voltage was chosen based on design for the system.	This is the max value that the amplifier runs on. Pg. 4 (1).
Vmin: 2.5V	This minimal voltage was chosen based on design of the system.	This is the min value that the amplifier runs on. Pg. (1).

4.6.5 Verification Plan

1) The MAX98357A is wired to the ESP32 (see above *Design* section). This supplies 3.3 Volts to the amplifier which validates the Vmax and Vmin.
2) 2. The code is flashed to the ESP32 that results in the speaker outputting a simple tone.
3) 3. Place a multimeter in series with the power between the ESP and MAX98357A.
4) 4. Connect an oscilloscope to the Din pin of the MAX98357A.
5) 5. At this point, the speaker should be outputting an audible sound. A max of 5500Hz and minimum of 500Hz can be adjusted by altering the code.

Measurements:

- The current on the multimeter should display around 2mA
- The oscilloscope should read voltage highs above 1.3V and voltage lows below .6V

- With a frequency meter, frequencies should be measured at 5500Hz and 500Hz

4.6.6 References and File Links

4.6.6.1 References

[1] "MAX98357A." *MAX98357A Datasheet and Product Info | Analog Devices*, www.analog.com/en/products/MAX98357A.html. Accessed 10 Mar. 2024.

4.6.6.2 File Links

N/A

4.7.7 Revision Table

Date	Revision
3/14/2024	Joseph Conrow: Adjusted uniform formatting
3/9/2024	Cade Tillema: Re-wrote block description due to change of amplifier type
3/10/2024	Cade Tillema: Updated interface validations
3/10/2024	Cade Tillema: Refined verification steps
3/10/2024	Cade Tillema: Updated revision table

4.7 Display

4.7.1 Description

The display block will be primarily used to show the current time on the clock. The majority of the work that will go into this block is integrating it into the system and coding it to effectively communicate and showcase the current time.In addition, since we are employing a backup battery option, it will be important to manage the current draw of the display, especially since the display we will be using has a high current draw. In terms of interface details, this block has two inputs and one output. The first, and expectedly important, input is power. It is supplied by the microcontroller, which ensures the continuous operation of the display. The DC input voltage will be a fixed 3.3V, with a nominal current of 60mA and a maximum of 80mA. These values were determined by the display's datasheet which serves as the main source of our information about this

block, such as wiring diagrams, example codes, and hardware specifications As expected, another input is the digital signal from the esp32, which is how the display gets the information. This information can vary in type, with a logic level of 0 or 1, maximum frequency of 32MHz, and a 10 KHz Pulse Width Modulation Frequency. These values are based on what the esp32 can provide, which were extracted from its datasheet. Lastly, the output, on the other hand, is a signal in the form of visual information, specifically the numerical representation of the current time. The display will be active high, which determines which segment of the display is on. Furthermore, the maximum representation of anything on the display is in 18 bits, with a refresh rate of 70Hz.

The integration part is a key aspect of our block development process, with a focus on correctly receiving information from the ESP32. While receiving data from the microcontroller is fundamental, coding the display to process that data is the other major focus of this block. With that being said, it will have a focus on how the display is going to work. Coding the display is important to process the signal from the microcontroller to get the information in numeral values corresponding to the current time. Furthermore, things like resolution scale, color, brightness, and positioning of numbers on the display will be included in the coding aspect of this block. This could get more complex depending on what kind of signal/information the display is receiving from the ESP32, and the mode of the display we use. The display we have chosen has an LCDscreen, and is a higher quality than some of the other ones we say. As a result, it is a little more complicated to program, and we might be using a specific library that fit's its operation, such as U8Lib in contrast to the Adafruit, or vice versa. The primary reason for that is we had many options for displays, but had to consider their pros and cons, such as complexity, usefulness, and power draw. As a result, certain displays are more difficult to program than others.

4.7.2 Design

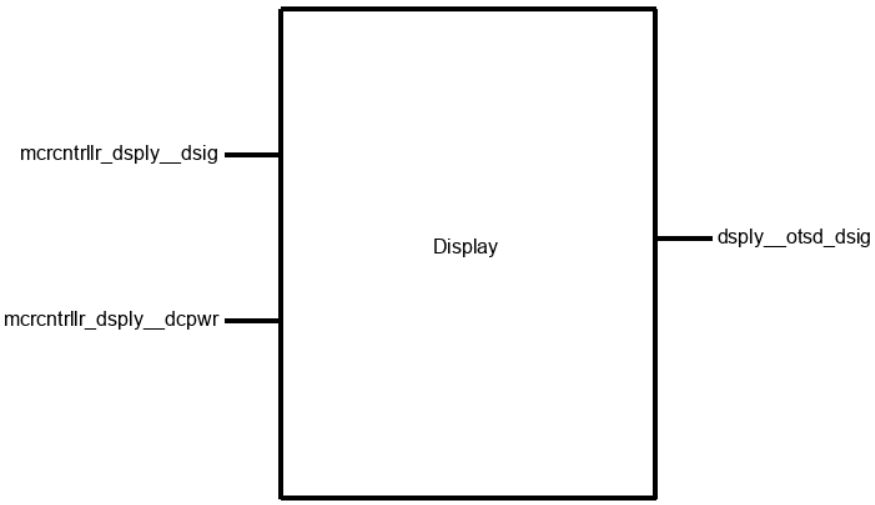


Figure XXX: Display Black Box Diagram

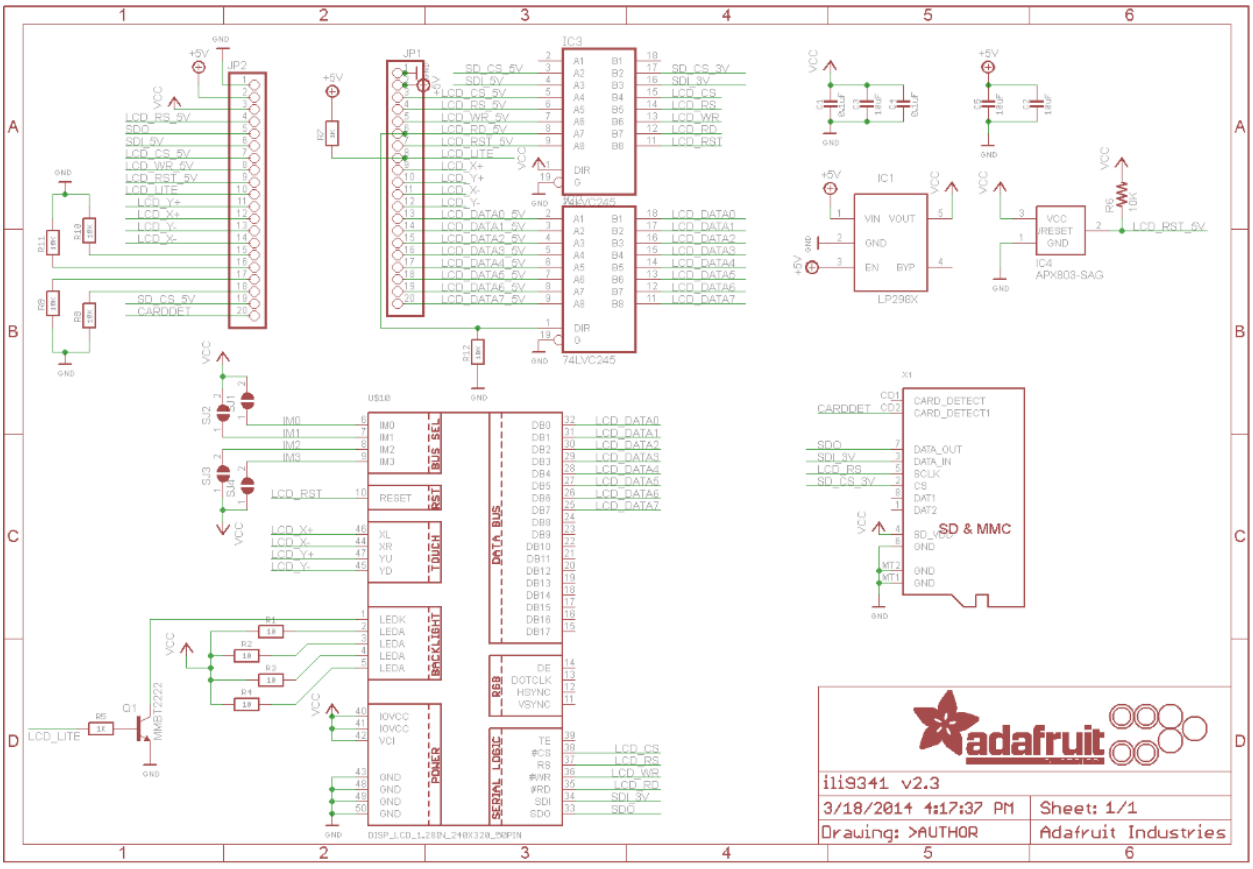


Figure XXX: PCB Schematic from AdaFruit

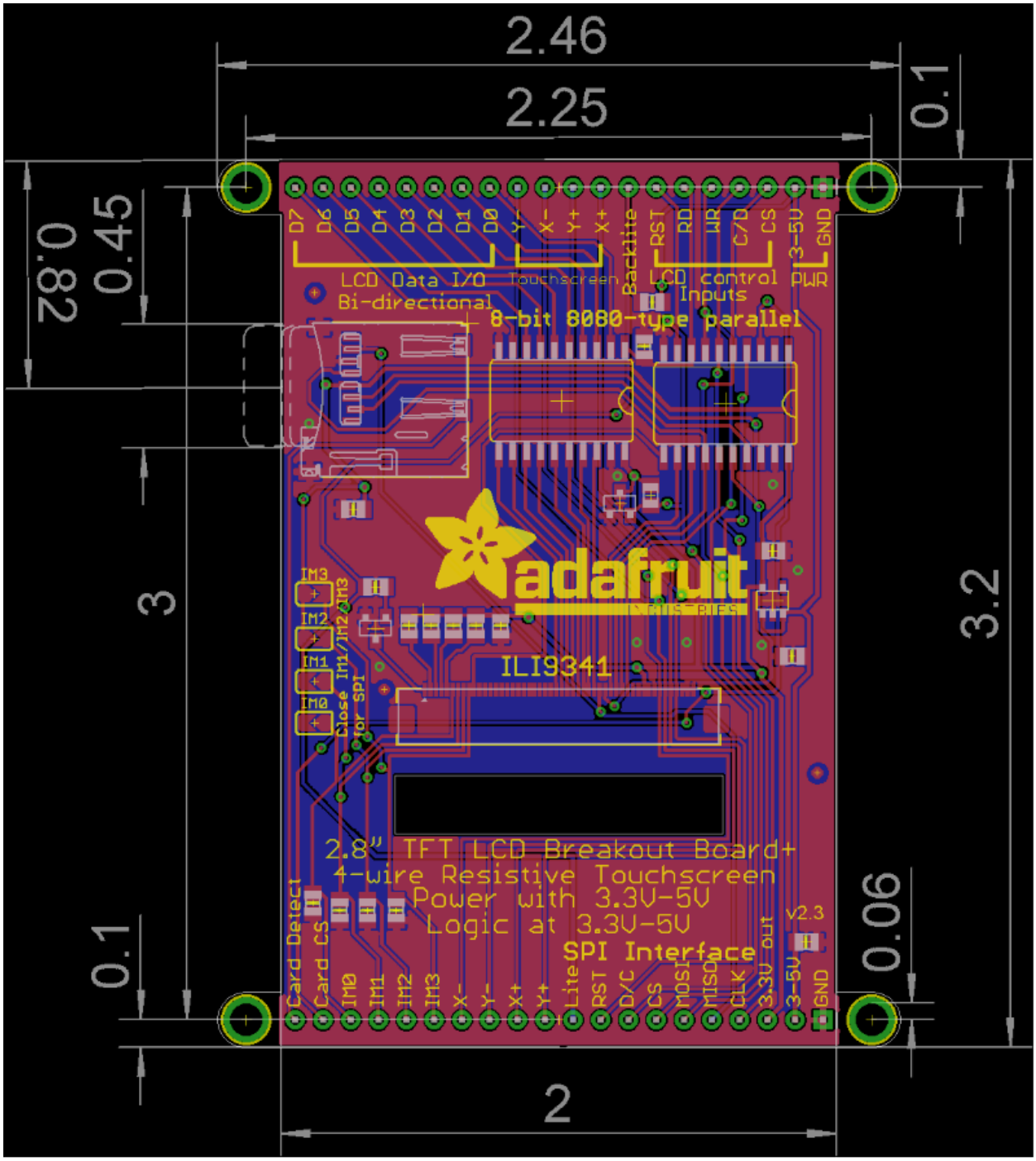


Figure XXX: PCB Layout from Adafruit

4.7.3 General Validation

The pictures above show a black box diagram and a draft pseudo code of the block. The first picture, being the black box diagram, simply shows the interfaces of the block, and how it is connected to two other blocks in the system. It takes in inputs from the microcontroller, the esp32, DC power and digital signal. From there, the output is essentially numerical values of the set current time. The

reason why we chose to implement this block into our system this way is for convenience and efficiency. While we can certainly power up the block from the batteries, it is more straightforward to use the microcontroller as a power source for this block. Furthermore, while we will be using an esp32 in our system, we will use arduino IDE to program the display, which is typically how screens are programmed in similar scale projects. The output of this block is the digital signal that will show the time in numbers. With that being said, the black box diagram shows that the block interfaces are designed in a way to ensure successful implementation of the block into our system, so that it can ultimately end up displaying numerical values that are being provided in the form of digital signals from the microcontroller.

The second aspect of the design shown in the previous section is pseudocode that describes the functionality of the block in more details. The pseudo code has a couple functions to show a possible start of a code implementation. The first step is declaring and initializing some output pins on the esp32 and the display. The initializing function is called into the main function of the program, which is followed by a bool variable sleep. The sleep variable will be utilized set_sleep_mode() function to control the display's sleep mechanics. This function is called a little later in the main function. After setting up the boolean variable, the program calls two functions to display the parsed time in hours and minutes, which ends up showing that current time.This information is stored on the esp32, and is transferred from the alarm set interface based on the user's input, so since it is a different block that stores the data on the esp32, how the time values are sent to the esp32 won't be discussed here. These two functions call the function, sendData(), that parses the data stored in the esp32 to extract the time in hours and in minutes. This variable will contain the user's input of the current time they want to set. For example, if the data from the esp32 is time 11:55 pm, the parser_hours will return 11 and parser_minutes will return 55. From there, we can display them separately with a semicolon in between to make it a readable time by the user. With that being said, this is how we are expecting the code to function, which will help us validate if the display is working as expected in the system. We will send data from the esp32 to the display, and using this code, we will output the time on the screen.

4.7.4 Interface Validation

TABLE XXX

Alarm Set Interface Interfaces

Interface Property	Why is this interface this value?	Why do you know that your design details <u>for this block</u> above meet or exceed each property?

dsply__otsd_envout : Output		
Light: Brightness Level: 3 -+ 2 Lux	This is the brightness of the screen that is determined by the code. It is this value based on how we adjust the brightness level in the code.	This brightness is about what other typical alarm clock systems have for their display's brightness. It isn't too bright to give people difficulty sleeping and isn't too dim to prevent people from reading the time from a few meters away.
Other: Usability Test: The display should show the correct input time from the esp32 9 out of 10 times.	An important aspect of this block is being able to display the time based on the input. As a result, it should show the time 9/10 times based on what is being uploaded to it.	I know that the design details for this block meet this property because the display was made for this.
Other: User Test: 9 out of 10 people have to be capable of reading the time displayed on the screen from 5 meters away.	Similarly, we want people to be able to view the time from 5 meters away or more. This is probably much higher than the average distance someone will be from the alarm clock, so ensuring that they can read it from that far shows that the alarm clock has a good screen for people to use.	The design detail of the block meet this property because the display has a good resolution and a good size, which means that the display is good enough to be used in our system.
mcrcntrllr_dsply__dsig : Input		
Logic-Level: Bit value of 0 or 1	The input digital signal from the microcontroller will have a value of 1 or 0, which will be later converted into a decimal to show on the display. Binary values have to either be 0 or 1.	Since the esp32 can provide binary information, the design meets this property since it is used as the source for the signal.
Vmax: 2.4V	This is the max voltage of the voltage spikes of the digital signal. It is at this value because this is what the esp32 provides at the clock output pin.	This fits into our system since we are regularly working with voltages in these ranges.

Max Frequency: 1Hz	The clock signal will provide a frequency of 1Hz, because this is how often the minimum time change on the display is.	As expected from an alarm system, this fits our needs and offers a standard alarm clock functionality.
mcrcntrllr_dsply__dcpwr : Input		
Ipeak: 83mA	This is the maximum current the LCD display can consume. Therefore, the input current value of the LCD display can't get higher.	The coding aspect won't necessarily tie to this value, but the integration aspect of the block will. The way the block is integrated ensures it can function properly, with a sufficient maximum current of 83mA
Inom: 68mA	This is the typical current the LCD display can consume.	The coding aspect won't necessarily tie to this value, but the integration aspect of the block will. The way the block is integrated ensures it can function properly, with a nominal current consumption of 68 mA.
Vmax: 3.3V	This is the maximum operating voltage of the LCD display. This is a predetermined value to most LCD displays.	Similarly, the block is integrated into the system such that it can utilize the 3.3V max for operation.
Vmin: 2.8V	This is the minimum operating voltage of the LCD display. This is a predetermined value to most LCD displays.	Similarly, the block is integrated into the system such that it can utilize the 2.8V minimum for operation.

4.7.5 Verification Plan

1) Ensure the code compiles on the microcontroller; using the microcontroller for verifying the display works is necessary.
2) Hook up the display to the microcontroller. For early verification stages, use a breadboard and jumper cables.
3) Start with **mcrcntrllr_dsply__dcpwr.** Begin with Vmin and provide 2.8V from the power supply. Ensure that the display still works and is displaying numbers or something of interest. Then, verify that the display works with

3.3V input and see that it is still on. For the Inom and Ipeak, use the electronic load to provide Inom and Ipeak, and ensure the display still works.

4) For the digital signal input from the microcontroller, **mcrcntrllr_dsply__dsig** , an oscilloscope will be handy. Start testing the PWM by providing the value we want in the code. This will make it so that a certain pin will provide the desired PWM frequency. Once the frequency is coded in, use an oscilloscope to show it and then plug that pin into the display and show it works with it. Similarly, each GPIO pin can provide 32MHz frequency. Use an oscilloscope to verify that and plug into the display and show that it works. For the bit value, the frequency is expected to be a square wave, so by reaching a high value or a low value, we can show that the bit value is changing from 0 to 1 as expected. If it worked with the 32MHz frequency test, then it works here.
5) Lastly, the output, **dsply__otsd_dsig,** is primarily the characteristics of the display. As a result, they aren't chosen by us and already set by the manufacturer and will be shown in the datasheet.

4.7.6 Reference and File Links

4.7.6.1 References

[1] Lady Ada, "Adafruit 2.8" and 3.2" color TFT touchscreen breakout V2," Adafruit Learning System, https://learn.adafruit.com/adafruit-2-8-and-3-2-color-tft-touchscreen-breakout-v2/downloads (accessed Jan. 28, 2024).

[2] ESP32 series - espressif systems, https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf (accessed Jan. 22, 2024).

4.7.6.2 File Links

N/A

4.7.7 Revision Table

Date	Revision
3/14/2024	Joseph Conrow: Adjusted uniform formatting
1/28/2024	Louis Marun: Edited, updated, and finalized all sections
1/21/2024	Louis Marun: Updated sections 2, 6, and added to the revision table.

1/18/2024	Louis Marun: Updated sections 1 and 4
12/6/2023	Louis Marun: Added content to section 2, 6, and 7.
12/5/2023	Louis Marun: Filled out sections 5-6.
12/4/2023	Louis Marun: Made google docs, inserted template for the block, and filled out sections 1-4 and 7

4.8 User Buttons

4.8.1 Description

The user buttons will be physically present on the outside of the enclosure. There will be three of these buttons that will serve as an interface between the user and the system. Each press of each button will trigger a different response in the system. The input of each button will be supplied by a signal from the microcontroller. Depending on whether the button is pressed it will allow signal to pass through i.e. if the switch is closed it will short the input of the switch with the output of the switch thus, allowing the signal to pass through. This functionality is known as SPST-NO indicating that there is a single input line and a single output line. NO indicates it is normally open, that is, not connecting the input with the output. The terminal that the output of the switch is connected is a General Purpose Input / Output (GPIO) pin on the microcontroller—which is the case for the pin that is supplying the switch as well. One button will activate the “snooze” feature. This takes effect when the alarm is sounding and a press of the button will postpone the alarm to sound in ten minutes—identical to how most modern alarm clocks function. The next button will be the “alarm reset” function. This takes effect when the alarm is sounding and the user desires to turn off the sounding alarm and reset it for the same time the next day. The last button will alter the brightness. Each press of this button will result in a preset decrement in brightness. A press of this button while it is at its lowest preset value it will roll back to the highest brightness setting. These buttons act as a pulse where the input and output are shorted for as long as the user is pressing down the button. As soon as the button is released the input and output will return to act as an open circuit. It will be the job of the software of the microcontroller to debounce each button press and ensure each press of the button counts as one press and not many. The rating of these buttons is 3A and 125V which is more than enough for our application.

These three buttons will be mounted to the top of the enclosure ensuring easy access for the user to interact with when desiring to trigger one of the specified behaviors of the alarm clock. A lock washer and a nut will screw onto the buttons below the surface of the enclosure. The nut will be tightened until the button is firmly fastened to the enclosure. These buttons will also be clearly

labeled on the enclosure. For simplicity of set up, the buttons will not be soldered directly to a PCB. They will be mounted to the enclosure and will be wired directly to the microcontroller. Jumper wire will be soldered to the leads of each button that carries the signal. The button leads have small holes that the copper wires will be wrapped around and soldered to.

4.8.2 Design

As stated above this block will be responsible not only for securing the physical buttons, but also routing the input signal to the output. Contained in this block is the *E-Switch* SPST_NO pushbutton switch. This pushbutton contains a front panel mount method with a lock washer and nut that screws onto the button clamping it to the enclosure. At the bottom of the button are the two lug-style terminals. The terminal will carry the input while the other terminal will carry the output. The medium in which these signals are carried will be via jumper wires. At the opposing end, in which the wire is attached to the microcontroller, the connection will made with a female jumper attachment. To connect to the button terminals the copper of the wires will be wrapped around the small hole of the terminal and soldered to secure the mechanical/electrical connection.

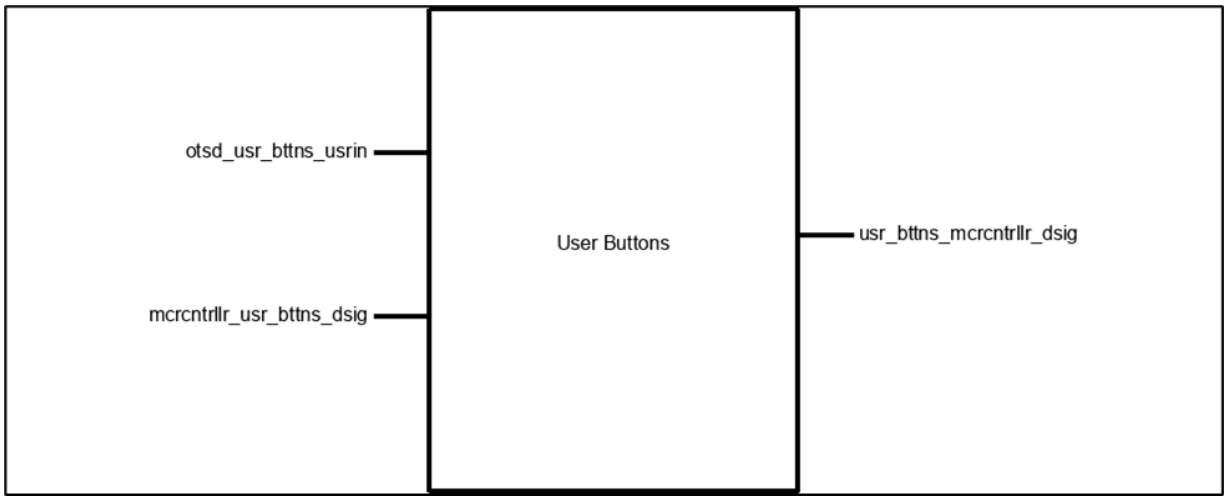


Figure XXX. User Buttons Black Box Diagram
Shows two input interfaces and one output interface.

The dimensions and technical details of the physical button can be found on the datasheet drawing shown here:

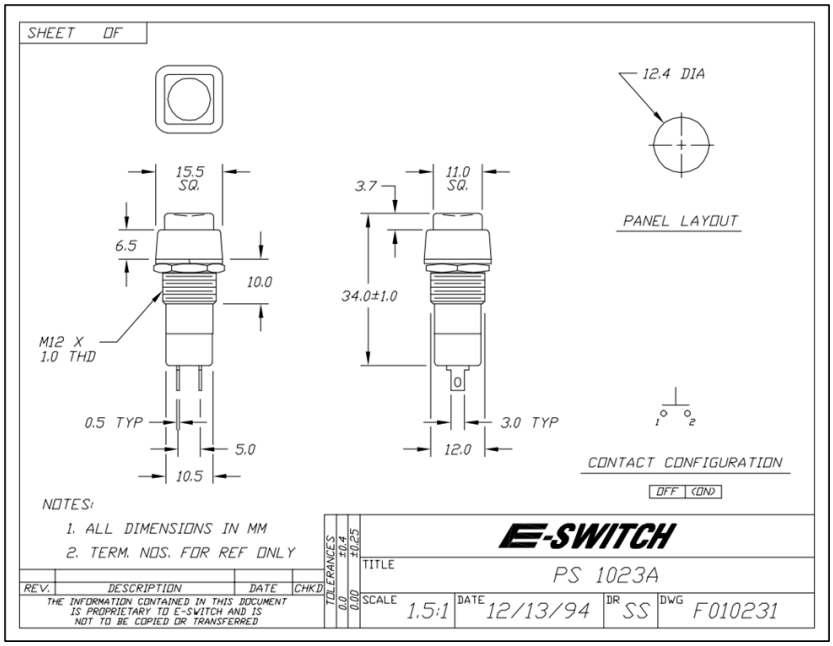


Figure 20. Datasheet Drawing of the PS 1023A pushbutton.

Information contained in fig. 20 is necessary to determine the way in which it will fit on the enclosure such as dimensions and thread size.

4.8.3 General Validation

The needs of the system are not intensive for this block. It requires a way to sense user input. Buttons are one of the most popular way in which this is accomplished. The selected push button is relatively inexpensive, simple, and will be integrated seamlessly into the system making it the ideal choice for the present need.

For its size, this button is significantly less expensive than alternatives. The unit cost is $2.01. Alternatives ranged from $8-$12. Due to this low cost, several extras were able to be ordered in case any break in installation.

The function of this block is simple when compared to other blocks and for this reason a less expensive unit will accomplish what is required of it. Its installation is intuitive and will not require a significant time commitment. Due to the context of the system, the alarm clock will not be exposed to any harsh conditions. Thus, it is appropriate to use plastic buttons.

Due to the style of the button it will fit very well into the system. Both a similar color and right proportion will make the button practical and visually appeasing when integrated into the clock. Its termination style allows for a direct connection to the microcontroller without the need for a mounted PCB in which the button would need to be soldered.

Overall, this button succeeds in fulfilling its role as an inexpensive, simple, and seamless integration into the system. It does not require additional mounting mechanisms, PCBs, or screws making it a cost and time efficient component. An alternative solution would be to design a PCB that would screw in at the top of the enclosure. The buttons would then be soldered to the PCB and jumper wires would direct the signal from the PCB to the microcontroller.

4.8.4 Interface Validation

TABLE XXX

User Buttons Interfaces

Interface Property	**Why is this interface this value?**	**Why do you know that your design details <u>for this block</u> above meet or exceed each property?**
otsd_usr_bttns_usrin : Input		
Timing: After alarm is triggered	At most one time per minute.	Since the buttons are physical they will always be accessible to a user within the vicinity of the clock.
Type: Toggle Switch	Buttons that produce a toggled signal are exactly what this system requires. To activate the snooze or alarm reset functionality all that the user needs to do is press a button and that information will be transmitted to the microcontroller.	These buttons are a toggle because they either let the signal pass through or it does not by remaining open.
Usability: Press Button	Physical buttons were chosen for the convenience of the user. It is a relatively easy action to activate a certain function such as snooze or alarm reset.	This block will work because buttons are as simple as blocking or letting a signal pass through.
mcrcntrllr_usr_bttns_dsig : Input		

Logic-Level: Active High	According to the button's datasheet (1) it is a normally open (NO) switch making it active high.	As mentioned in the design section, the button will be directly wired to the microcontroller with nothing in between. This design ensures that the logic level is, in fact, active high.
Vmax: 5.5V	According to the datasheet of the microcontroller (2) this is the max output level voltage of its digital output pin. This output of the microcontroller also acts as the input of the button.	As mentioned in the design section, the buttons will not alter the signal. It will short the input with the output and thus, the max voltage will be whatever the max of the input is.
Vmin: 3V	According to the datasheet of the microcontroller (2) this is the minimum output level voltage of its digital output pin. This output of the microcontroller also acts as the input of the button.	As mentioned in the design section, the buttons will not alter the signal. It will short the input with the output and thus, the minimum voltage will be whatever the minimum of the input is.
usr_bttns_mcrcntrllr_dsig : Output		
Logic-Level: Active High	According to the button's datasheet (1) it is a normally open (NO) switch making it active high.	As mentioned in the design section, the button will be directly wired to the microcontroller with nothing in between. This design ensures that the logic level is, in fact, active high.
Vmax: 5.5V	According to the datasheet of the microcontroller (2) this is the max output of the digital output voltage level.	As mentioned in the design section, the buttons will not alter the signal. It will short the input with the output and thus, the max voltage will be whatever the max of the input is.

Vmin: 3V	According to the datasheet of the microcontroller (2) this is the minimum output of the digital output voltage level.	As mentioned in the design section, the buttons will not alter the signal. It will short the input with the output and thus, the minimum voltage will be whatever the minimum of the input is.

4.8.5 Verification Plan

1) Connect a DC power supply to the button with voltage set at 3 volts. Positive to one terminal and negative to a 10k Ohm resistor with the other lead of the resistor connecting to the other terminal of the button.
2) With a multimeter, test continuity between the both terminals of the button. There should be no continuity. This verifies “Active High” logic level for both usr_bttns_mcrcntrllr_dsig and mcrcntrllr_usr_bttns_dsig
3) Press and hold down the button and measure across the resistor. The multimeter should read 3 volts. This verifies “Vmin” for both usr_bttns_mcrcntrllr_dsig and mcrcntrllr_usr_bttns_dsig.
4) Repeat steps with power supply at 5.5 volts. With the button held down should read 5.5 volts. This verifies “Vmax” for both usr_bttns_mcrcntrllr_dsig and mcrcntrllr_usr_bttns_dsig.
5) In addition, the way in which the button is pressed verifies the type as “toggle switch”, usability as “Press Button”, and timing for otsd_usr_bttns_usrin

4.8.6 References and File Links

4.8.6.1 References

[1] “PS 1012A,” *digikey.com*, Dec. 13, 1994. https://mm.digikey.com/Volume0/opasdata/d220001/medias/docus/980/PS1023A.pdf (accessed Jan. 27, 2024).

[2] “ESP32 Technical Reference Manual.” Available: https://www.espressif.com/sites/default/files/documentation/esp32_technical_reference_manual_en.pdf#sensor

4.8.6.2 File Links

N/A

4.8.7 Revision Table

Date	Revision
3/14/2024	Joseph Conrow: Adjusted uniform formatting
3/12/2024	Joseph Conrow: Table formatting
1/27/2024	Cade Tillema: Added design, general validation, interface validations, verification plan, and references
1/26/2024	Cade Tillema: Added to description
12/7/2023	Cade Tillema: Adjusted format and numbering
12/7/2023	Cade Tillema: Filled out basic outline information

5 System Verification Evidence

5.1 Universal Constraints

5.1.1 The system may not include a breadboard

There is nowhere in our design of this project that includes the use of a breadboard. Any such responsibilities that could be assigned to a breadboard such as routing wires and interacting between the microcontroller and peripheral devices have been assigned to a custom PCB.

5.1.2 The final system must contain a student-designed PCB

As discussed above all routing of wires between different parts of the system. To reference our block diagram, a PCB will be used in the “Backup Battery” as well as the “User Buttons”. The PCB for the backup battery is a device to recharge a LIPO battery. The PCB for the buttons will be rather simple as it will house several switches that when pressed will message the microcontroller that a button has been pressed.

5.1.3 All connections to PCBs must use connectors.

As opposed to soldering wires directly to any of our PCBs, we will use male/female header pins as necessary.

5.1.4 All power supplies in the system must be at least 65% efficient

We plan on purchasing our power supply and will perform research, analysis, and calculation as necessary to confirm its efficiency.

5.1.5 The system may be no more than 50% built from purchased 'modules.'

As seen in our block diagram, only five modules are purchased: AC-DC Power Supply, Microcontroller, Display, and Speaker. However, most of these modules will be modified in some way by the team. The Microcontroller, Display, and Speaker will all need to be connected and coded in order to properly function in an integrated fashion.

5.2 Requirements

5.2.1 Audio Output

1. **Project Partner Requirement:** The system will output sound at the user's given time and be variable
2. **Engineering Requirement:** The system will output sound within 1 second of the set time at the volume specified within 10% of desired decibels.
3. **Testing Method:** Demonstration
4. **Verification Process:**
    4.1. Set the alarm for a specific time
    4.2. Wait for clock to reach the same time as the alarm was set
    4.3. Listen for speaker output within one second of clock reaching allotted time
5. **Testing Evidence:**

5.2.2 Backup power

1. **Project Partner Requirement:** The alarm clock needs to stay powered while unplugged from wall power.
2. **Engineering Requirement:** The system will operate normally for 48 hours after disconnecting from external power.
3. **Testing Method:** Demonstration
4. **Verification Process:**
    4.1. Plug alarm clock in
    4.2. Ensure batteries are fully charged
    4.3. Unplug alarm clock from outlet power
    4.4. Return clock after 47.99 hours
    4.5. Check if display is still illuminated
5. **Testing Evidence:**

5.2.3 Button Functionality Toggle

1. **Project Partner Requirement:** The system's physical buttons need to be able to be enabled.
2. **Engineering Requirement:** The system will enable/disable the physical user inputs based on website input 95 out 100 times.
3. **Testing Method:** Test
4. **Verification Process:**
    4.1. Ensure the system is on
    4.2. Set an alarm for 1 minute from current time
    4.3. On the web interface, toggle off the snooze and reset button
    4.4. Once the alarm goes off, try to press the buttons
    4.5. Did the buttons work or not?
5. **Testing Evidence:**

5.2.4 Display Time

1. **Project Partner Requirement:** The system needs to display the current time of day.
2. **Engineering Requirement:** The system needs to show the time in hours and minutes accurately to within 5 seconds of global time even when Wi-Fi has not been connected for at least 24 hours.
3. **Testing Method:** Demonstration
4. **Verification Process:**
    4.1. Ensure the system is on
    4.2. Check the currently set time
    4.3. Compare the displayed time with the standard user's time zone (in our case, Pacific Time)
    4.4. Ensure that the delay between the two times is within 5 seconds
5. **Testing Evidence:**

5.2.5 Internet-based interface

1. **Project Partner Requirement:** The system must be accessed via the Internet.
2. **Engineering Requirement:** The system will accept the user's input via the web-hosted site to determine the time of the clock and alarm.
3. **Testing Method:** demonstration
4. **Verification Process:**
    4.1. Log into the website
    4.2. Schedule an alarm time that is 2 minutes from current time
    4.3. Insert the current time you want to display
    4.4. Submit both the scheduled alarm time and current set time
    4.5. Check for a popup confirmation message
    4.6. Does the message say "alarm time and current time currently scheduled?"

5. **Testing Evidence:** https://drive.google.com/file/d/1Lhjgw68hHy-bqi-Ge-X8J-qbpxsq3yBh/view?usp=sharing

Explanation: This requirement is about ensuring that the alarm set interface will take in the user's input for the alarm scheduled time successfully, which is shown by a popup message. The video we provided above shows the verification process in detail. We log into the website and show that setting up an alarm time successfully provides a popup confirmation message.

5.2.6 Physical Buttons

1. **Project Partner Requirement:** The system must have typical physical alarm interface buttons.
2. **Engineering Requirement:** 9 out of 10 users will adjust the system using the snooze button and alarm reset button and state "The buttons were reliable and changed the system as expected".
3. **Testing Method:** test
4. **Verification Process**
   - 4.1. Locate the buttons on the alarm clock.
   - 4.2. Wait for a scheduled alarm time.
   - 4.3. Click on the snooze button to see if it extends the alarm schedule by a few minutes.
   - 4.4. Click on the reset button to see if it works as expected and reset the scheduled alarm time for the next day.
5. **Testing Evidence:**

5.2.7 Safe

1. **Project Partner Requirement:** The enclosure of the system needs to be safe.
2. **Engineering Requirement:** The system will have an enclosure that does not contain sharp or jagged edges with no visible wires.
3. **Testing Method:** Inspection
4. **Verification Process:**
   - 4.1. Look at the system and see if there are any visible wires.
   - 4.2. If there are any visible wires are there any uninsulated wires?
5. **Testing Evidence:** https://drive.google.com/file/d/1kaNfgD-a9-RfYtqXEEtxPQmZxvyMArM3/view?usp=drive_link

Explanation: This requirement is about ensuring that the enclosure does not have any exposed or visible uninsulated wires. The video

provided shows the entirety of the enclosure that it does meet such standard.

5.2.8 Website Credentials

Project Partner Requirement: The website will have a login to access device settings.

Engineering Requirement: 9 out of 10 users will successfully log in to the web-hosted site with valid credentials and state “The username and password gave access to the internet-based interface”.

Testing Method: Test

Verification Process:

Power on an internet-connected device

Go to the website for the alarm clock

Enter incorrect credentials

Check to see if it signed in. If you were signed in this requirement failed. If you were not signed in, continue with the next steps.

Enter the correct credentials

Check to see if you were signed in. If you were signed in this requirement was successful.

Testing Evidence:

5.3 References and File Links

5.4 Revision Table

Date	Revision
3/14/2024	Joseph Conrow: Adjusted uniform formatting
12/4/2023	Louis: Updated testing methods for physical buttons and Internet-based interface
12/4/2023	Joseph Conrow: Updated testing methods and verification process

	for Button Functionality Toggle & Display Time
12/4/2023	Andrew Zellner: Updated engineering requirement for Safe Enclosure. Updated testing method and verification process for Safe Enclosure and Website Credentials.

6. System

6.1 Future Recommendations

6.1.1 Technical recommendations

1) Brain gamma wave monitoring

   The brain wave head set monitoring sensor can be used to gather sleep quality data and find patterns of good and bad pre-sleep routines. That way the users can understand the causes and effects of their actions, sleeping environment, and sleeping routines. The brain waves will give indications for such benefits and downfalls.

2) Responsive camera monitoring

   This camera will be infrared for better viewing in low lighting. The idea is to have a camera on the alarm clock to allow the primary user to look into the child while they are sleeping to ensure they are doing what they are supposed to be doing. An example of this would be making sure the child is going to bed on time instead of looking at their phone.

3) Remote lighting sleep schedule synchronization

   With more time, the website will also include options to schedule when the lights turn on and off in the bedroom. This will require lights that can be controlled using a website/app, which can be synced with the system's website. This can make the whole alarm configuration process more automated and aid parents more.

4) Fully integrated mobile app

   Currently our project supports control of the clock via a website accessed through any device. However, a future feature could include an application that can be

downloaded on a phone to allow the customer to control the clock.

6.1.2 Global impact recommendations

1) Inexpensive material costs lead to a cheaper product which is available to more people.

   Use products that are cheaper but still perform the same. This will allow a cheaper overall cost which allows the product to be sold for a cheaper price. This is a global effect because it allows people of lower class to afford an alarm clock.

2) Purchase thoroughly environmental conscious sourced components

   The awareness of the product's effect on the global climate will lead the search for sourcing sustainable and environmentally friendly components and production lines. This includes shipping and manufacturing in these steps of the lifecycle of the product. By doing our part in the environmental defense force, we can keep the planet alive, clean and healthy for future generations.

6.1.3 Teamwork recommendations

1) Ensure that the work is delegated equally among all members. For example, one group member is responsible for coding the display, one for the speaker, one for the power converter, and the other for the IoT functionality.
2) Among the work that is required, hand out work based on team members prior experience and background knowledge.

6.2 Project Artifact Summary with Links

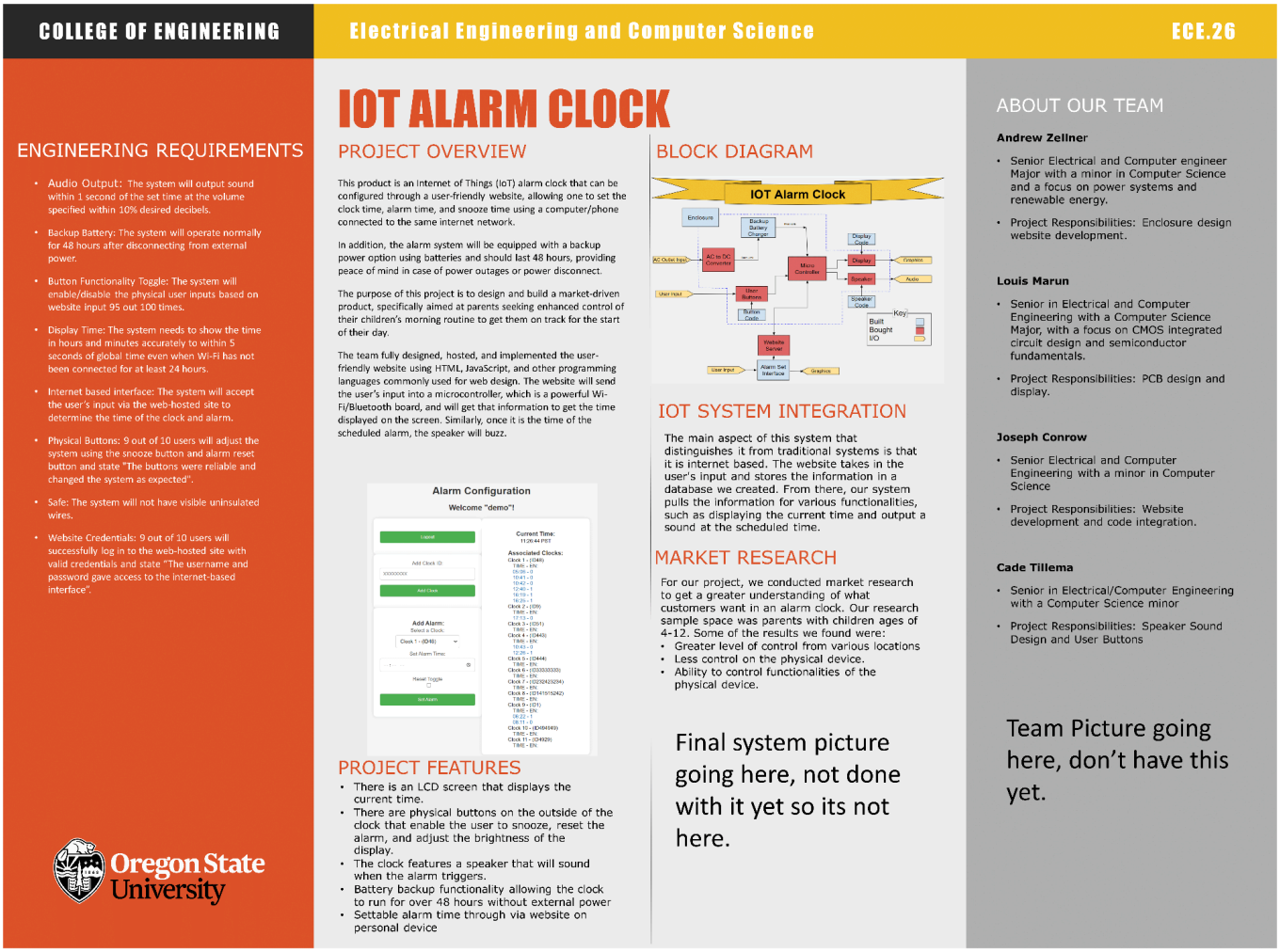
1) Code Link (Arduino & Website): This github repository has both the arduino's library and main file code, as well as the websites HTML and PHP code to manage the database and client requests. The organized folders of each are according to the function of the code section.
   Arduino & Website Code
2) Schematics and mechanical drawings: This link encompasses all of the schematics and mechanical drawings for the enclosure of the

system. These drawings and part files were all made in inventor as well as a exported file for the drawings.
Schematics and Enclosure Files

3) PCB Files: This link includes all the parts that were involved in implementing our PCB into our system. This primarily consists of the parts soldered on our PCB, but also includes the battery holder for the two batteries in our system.
PCB Parts - myLists | DigiKey

6.3 Presentation Materials

Expo Poster Download File



6.4 References and File Links

6.4.1 References

6.4.2 File Links

PCB Parts - myLists | DigiKey
Arduino & Website Code
Schematics and Enclosure Files
Expo Poster Download File

6.5 Revision Table

Date	Revision
4/26/24	Joseph Conrow: Added the specific code folders within the github link
4/26/24	Joseph Conrow: Corrected my spelling error in the revision table
4/26/24	Joseph Conrow: Added the Code portion for the project artifacts
4/26/24	Joseph Conrow: Added the poster for the presentation materials
4/26/24	Joseph Conrow: Changed headers of section 6
4/26/24	Cade Tillema: Added numbering to each section 6 heading
4/26/24	Joseph Conrow: Devised elegant and articulate phrasing of the global impact of our project
4/26/24	Louis Marun: Added Digikey reference and Technical Achievements
4/26/24	Andrew Zellner: Added Achematics and Encloure files as well as updated some descriptions of sections 6.1 and 6.2.
4/26/24	Andrew Zellner: Added Section 6 with sub sections