B.R.E.A.T.H.E: Balanced Responsive Environmental Airflow and Temperature Housing Enhancer

EEL 4915L Senior Design 2

Group 40

Members:

Emily Baross - Electrical Engineering

Kyle Ingleton - Electrical Engineering

Marcus Loyd - Computer Engineering

Patrick Tumbos - Computer Engineering

Reviewers:

Dr. Mike Borowczak, Dr. Sazadur Rahman, Dr. Elizabeth Coln

Sponsor:

Dr. Qun Zhou Sun

Advisors:
ChungYong Chan, Saleem Sahawneh

Table of Contents

1.0 Executive Summary

Improving the environment of living conditions in common households is a goal that industry aims to solve through finding new ways to implement technology. A significant problem includes irregular temperature distribution throughout the household, causing some rooms to be significantly hotter or colder than others. This can cause discomfort and force residents to choose or rely on alternative, less efficient means of temperature control such as ceiling fans and heaters. These inconsistencies can also negatively affect sleep quality due to the low temperature accuracy of these options and other variables such as the sound generated from ceiling fans. In many situations, the problem isn't necessarily due to an air condition unit, but rather ineffective air distribution amongst the rooms of the building. Finding an energy efficient solution to these problems is crucial towards improving the quality of life for residents living in common households, which serves as the main motivation for the development of this project.

BREATHE is a smart ventilation system that aims to provide a solution to these temperature concerns by allowing users to automate their ceiling registers' louver positions, and in turn adjusting the temperature of their room through a control unit. This control unit functions by a user inputting a desired temperature and in turn the device will automatically adjust the angle of the louvers on the vent in accordance with the room's current temperature. BREATHE will consist of two separate units; a control unit which will monitor the environment through sensors and interact with the user, and a vent interaction subsystem that will interface with the vent's louvers and open or close it according to signals from the control unit. The vent interaction subsystem is designed with battery efficiency in mind, reducing the amount of power consumed while the device is operating while also achieving the goal of maintaining a comfortable and desired environment.

This document covers the design process of BREATHE, providing a comprehensive view of the overall project's development. To establish a clear foundation for the development of BREATHE, the content begins with the specification of attainable goals and the objectives to achieve them. This includes the thought process behind the hardware selection, choosing components such as the microcontroller unit, temperature sensor, carbon dioxide sensor, motor and encoder, buzzer and LCD screen. Moreover, the approach regarding the software of BREATHE was also discussed, encompassing areas such as coding environments and graphic libraries. This

also includes the justification of components, graphs and flowcharts of subsystems within the project, and related items. The document further discusses the implementation of our component selection into prototyping and experimentation to deepen our knowledge in these fields. The project was designed to align with our main goals, aiming to optimize elements to meet our design specifications.

2.0 Project Narrative

2.1 Background and Motivation

In today's society, the demand for reliable and efficient home ventilation systems has become paramount. Individuals depend on these reliable home ventilation systems, as they not only establish overall comfort in indoor spaces such as residential, commercial, and industrial spaces, but they also serve to maintain excellent indoor air quality as well as provide the means to regulate temperature. By creating a more effective method of regulating ventilation, BREATHE will be able to sufficiently improve one's health and comfort levels while keeping energy demands low.

Despite advancements in this specific field, there exist some ventilation systems that do not meet the standards that individuals demand in these living spaces. Issues such as inconsistent airflow distribution and poor temperature regulation both contribute to an uncomfortable indoor environment. Moreover, a poor ventilation system can lead to health issues that are problematic to both residential and commercial settings. The circulation of carbon dioxide has been proven to have adverse effects on various environments. For this reason, the importance of proper ventilation not only circumvents the problem of discomfort, but also recognizes the need to monitor the air quality that is being circulated.

Thus, the motivation for this project can be defined by the need to develop an efficient system that directly combats issues that are commonly found in home HVAC systems. This motivation also stems from the team's personal experiences with poor ventilation that results in some rooms with improper heating/cooling systems. Particularly in the tropical state of Florida where rooms can reach upwards of 90 degrees Fahrenheit, many have experienced the discomfort of poorly ventilated rooms. This can become especially unpleasant during the warmer months of the year. The personal struggles in dealing with hot living spaces has reinforced the need to create an efficient system that takes these problems into account.

In developing this project, the team seeks to integrate various electronic and software systems, including environmental sensors and microcontrollers, to intelligently adjust the airflow based on real time environmental data. By taking advantage of these electronic and software systems, air circulation is optimized, thus resulting in a more comfortable living environment.

By addressing both the problems identified in comfortability as well as inefficiencies found in pre-established ventilation systems, the overall aspiration of this project is to bolster the everyday individual's comfortability to their specific needs, facilitate an environment with safe air quality, and contribute to sustainable living. Our motivation leads us to design and implement an efficient and smart ventilation system that not only mitigates the flow of inconsistent and inefficient airflow, but also guarantees an indoor environment that is conducive of a comfortable and healthy living space.

2.2 Previous and Existing products or projects.

2.2.1 Google Home

An existing product on the market that has the ability to adjust temperature via a cell phone application is the Google Home. Google Home has the capability to connect various devices ranging from speakers, doorbell cameras, alarm systems, smoke alarms, Nest thermostats, and beyond. The Nest Thermostat is the device connected through the Google Home app, and this shares the most similarity to BREATHE. With the Nest Thermostat one can view and adjust the temperature of your home's thermostat, toggle from cold to heat settings, select how long the fan should run for, view the humidity percentage of the home, and set a desired temperature schedule per day. It also includes an energy dashboard which displays how much energy is being saved. The device compares the amount of hours of heating and cooling to the previous month, and the daily average of heating and cooling per day. The inspiration we drew from this application as a team was the ability to access temperature readings and manually adjust the temperature output. Our project will be different from this product because we are not modifying the temperature for the entire house, but rather an individual room where our device is placed with a carbon dioxide sensor integrated with our design. BREATHE consists of a manually controlled vent that has the ability to be open or closed in regard to a user's desired room temperature [1].

2.2.2 The Automated Ventilation Controller

One of the first similar projects to catch our eye was the Automated Ventilation Controller project from another senior design group in 2021. The idea was similar, but instead focused on a warehouse environment where accurate temperature control was a necessity for keeping sensitive components and machinery maintained. We conversely wanted to take a different approach which focused more on the comfort of the average person in a residential space. We also wanted to see if we could further optimize the power consumption to make a more approachable product for your average consumer who wouldn't want to replace a battery every few months. We additionally plan to implement a CO_2 sensor which would alert the user of high concentrations of CO_2 in their living space, something that is not acknowledged often as a health risk. We were also very inspired by their convenient mobile app integration, something that we see many other projects incorporate as well. Though we see this as a stretch goal, we do hope to incorporate a mobile app and autonomous vent control in our project as well [2].

2.2.3 Ctrl Flow

Another past project we gained inspiration from is a senior design group that altered fan controls to reach desired room environment quality. Their project, the CTRLFlow, is different from ours in that their main controller was operated through a mobile app connected to a remote cloud server. This is how they would change the speeds of the fans and the angle of the louvers to adjust to the temperatures of the corresponding rooms. Our project will pursue similar goals, however, as mentioned prior, we plan on utilizing a CO_2 sensor to optimize the air quality of the room whilst using power-saving techniques to achieve a stronger battery life. As a stretch goal, one technique we would like to implement is a miniature wind turbine to generate electricity, taking advantage of the AC airflow [3].

2.3 Project Goals

The overarching goal of BREATHE is to design and develop a vent system that will open or close in order to adjust the airflow in a room while monitoring basic environmental conditions such as temperature and carbon dioxide. Using real-time data that is provided by the respective sensors, BREATHE should aim to facilitate a comfortable living space. The fundamental basis of this project's objectives can be simplified to a set of basic, and stretch goals described below.

Basic Goals

1. The vent can be opened or closed remotely from user input.
2. Autonomous control of the vent will be based on user-specified temperature parameters.
3. The temperature of the room in which the vent is located will be regulated based on user input.
4. BREATHE will have a sensor that will sense the Carbon Dioxide levels in a room.
5. A buzzer will alert the user during necessary situations.
6. The power system will ensure a reliable delivery of power to each necessary component.
7. The device will be easy for the user to control and adjust.

Stretch Goals

1. The user can open the vent to a desired angle.
2. BREATHE will incorporate a fan behind the vent to be used as a power generation feature similar to a wind turbine in order to increase battery life of the vent interaction subsystem.
3. BREATHE will incorporate a fan behind the vent to be used in the event that Carbon dioxide levels are high and increase airflow into the room to dissipate excess Carbon dioxide
4. A mobile application will be used as a secondary way to control BREATHE.

2.4 Project Objectives

In essence, BREATHEs goal is to provide a user with an efficient ventilation system that adjusts airflow that is dictated by environmental conditions, ensuring the facilitation of a comfortable living space. With respect to comfortability, BREATHE will be designed to tailor the airflow of the vent in response to any variations found in room temperature. For example, the colder the room temperature, the more restricted the air flow will be and the hotter the room temperature, the less restricted the air will be. In essence, BREATHE will have the capability to attenuate airflow based on sensed room temperatures, thereby facilitating comfortability in a living environment.

BREATHE will provide users with the capability to provide full manual control on the airflow based on their input. A vent interaction sub-system will allow the vent to be adjusted between a fully open and fully closed position through the use of a DC motor, which will be controlled by an MCU. The user will be able to implement these adjustments via a control unit which houses the LCD screen

and the sensors. Alternatively, the user can specify a desired temperature value, which enables an autonomous mode that will fluctuate the vent's state to change based on the current sensor reading of the room's temperature.

BREATHE will have a control sub-system with an LCD screen that takes in user input in conjunction with sensors to provide accurate readings of temperature and CO_2 levels. In order to achieve reliable and accurate readings, an appropriate temperature and CO_2 sensor will be selected. These sensors will be connected to a secondary MCU, which will be responsible for aggregating and interpreting the data gathered from both the temperature and CO_2 sensor.

Furthermore, BREATHE will need a power system that is capable of delivering power to all subsystems including the temperature sensor, CO_2 sensor, DC motor, LCD screen, and both MCUs. Implementing this system will bring challenges, such as finding a suitable battery that will provide adequate power to each component and limiting the battery usage of each subsystem to reduce the inconvenient task of replacing the battery frequently. This problem necessitates finding a bluetooth compatible MCU and sensor with relatively low power consumption that will still satisfy our technical requirements. Modes such as Sleep Mode and Power Down Mode can be integrated in tandem with a reliable power supply section to optimize energy usage.

Next, BREATHE will have a Control unit that is used to control the vent system and display sensor values. The utilization of a touchscreen LCD along with an intuitive interface will create a user-friendly and easy-to-use system. The Control unit will be programmed to display different warnings if necessary, with the help of visual and audio cues. One of the display warnings will be for low battery reading of the vent, which will consist of an intermittent beep and a warning message on the LCD screen. The other display warning will be for a high CO_2 sensor reading. If the CO_2 sensor reads a value of 4500 ppm, the LCD unit will sound an alarm and present a warning message on the screen. A dedicated MCU will be needed to power the LCD screen and manage its associated functions.

Lastly, BREATHE will be Bluetooth compatible, allowing communication between both MCUs involved in the overall system. Intuitively, the MCUs chosen will need to have Bluetooth integrated within them. Bluetooth connectivity will be tested in order to confirm an established and reliable communication method between MCUs over a specified range. Software will be developed for Bluetooth communication protocol to ensure scalability and communication with both the sensors and DC motors to the main Control unit.

2.5 Specifications

Specification	Description	Quantitative Measure
Manual Vent Control	The system should be capable of manual operation that fully opens or closes the vent through user input	Vent can be in 1 of 2 positions, *open or closed*.
Autonomous Vent Control	The system should be able to autonomously adjust the louver position according to the room's temperature and the user's desired temperature threshold.	If the temperature variation is ± *2°F*, the vent should open or close depending on if the variation is above or below the desired temperature set by the user.
Sensor Accuracy	Sensors should maintain high accuracy within a specific range	Temperature reading should be accurate to ±*1°F* and CO_2 reading should be accurate to ±*150 ppm*
Sensor Operating Range	Each sensor should have a sufficiently wide defined operating range to accommodate for extreme conditions.	Temperature sensor operation range should be from *-40* to *176 °F,* whereas the CO_2 sensor operation range should be from *0 ppm to 5000 ppm*
Battery Life	Unit should be able to operate for a certain length of time before a battery replacement.	Minimum duration of *6 months* before needing replacement
Bluetooth Range	Bluetooth communication should work within a specified range	Bluetooth communication should work up to a *10-meter range*

Dimensions	The overall vent system's physical dimensions should fit within a specified range.	*14 x 6 inches*
Audio Alarm System	An audible alarm should be triggered when CO_2 levels exceed a certain threshold. A separate alarm will trigger when the battery is low.	Alarm sounds when CO_2 *exceeds 4500 ppm.* When the battery of the vent interactive subsystem unit is below *10%* capacity it will sound intermittently at a lower pitch and volume. There should be only a ≤10% chance of a false alarm occurring.
Vent Actuation Time	The vent should fully open or close within a defined time limit.	Vent will be able to fully open or close within *5 seconds*

Table 1: Specification Table

2.6 Block Diagrams

2.6.1 Hardware

The system is split into two main sections: the vent interaction subsystem, and the control unit which houses the LCD unit, and sensors. The diagram shows the communication between each of the components, as well as the main interactions between each sub-system. The vent interaction sub-system focuses on delivering proper airflow to the environment by utilizing a motor to adjust louver positions. The control unit focuses on taking input from the sensors and the LCD unit is responsible for sending the control signals to the vent interaction sub-system. The LCD unit will then display information such as temperature and CO2 levels, allowing data to be monitored. It will also trigger an alarm if a hazardous situation occurs. The Hardware Diagram is shown below.

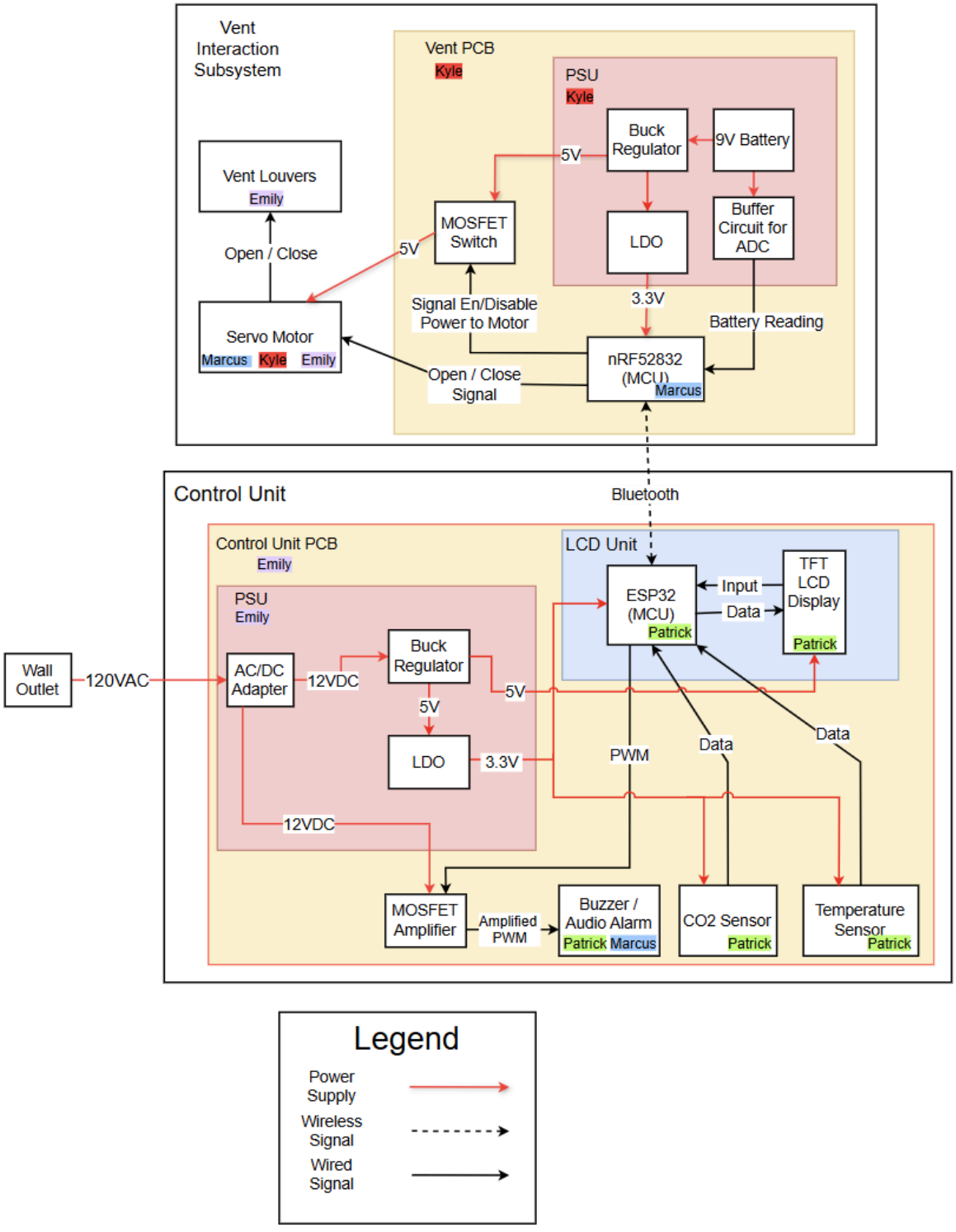


Figure 1. Hardware Diagram

2.6.2 Software

The software use-case flow chart below demonstrates the functional interactions between each of the three subsystems, as well as the work distributed between group members.

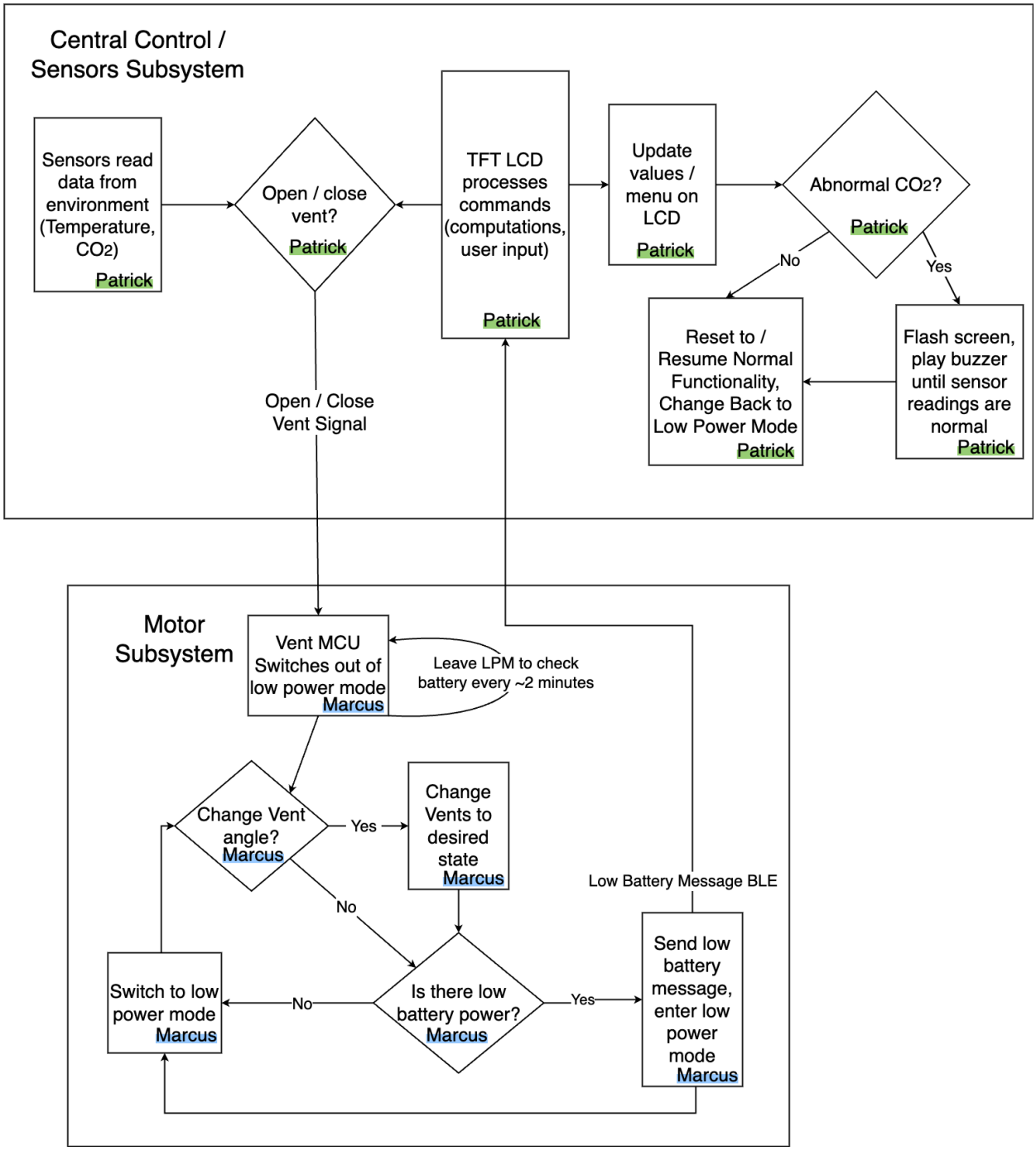
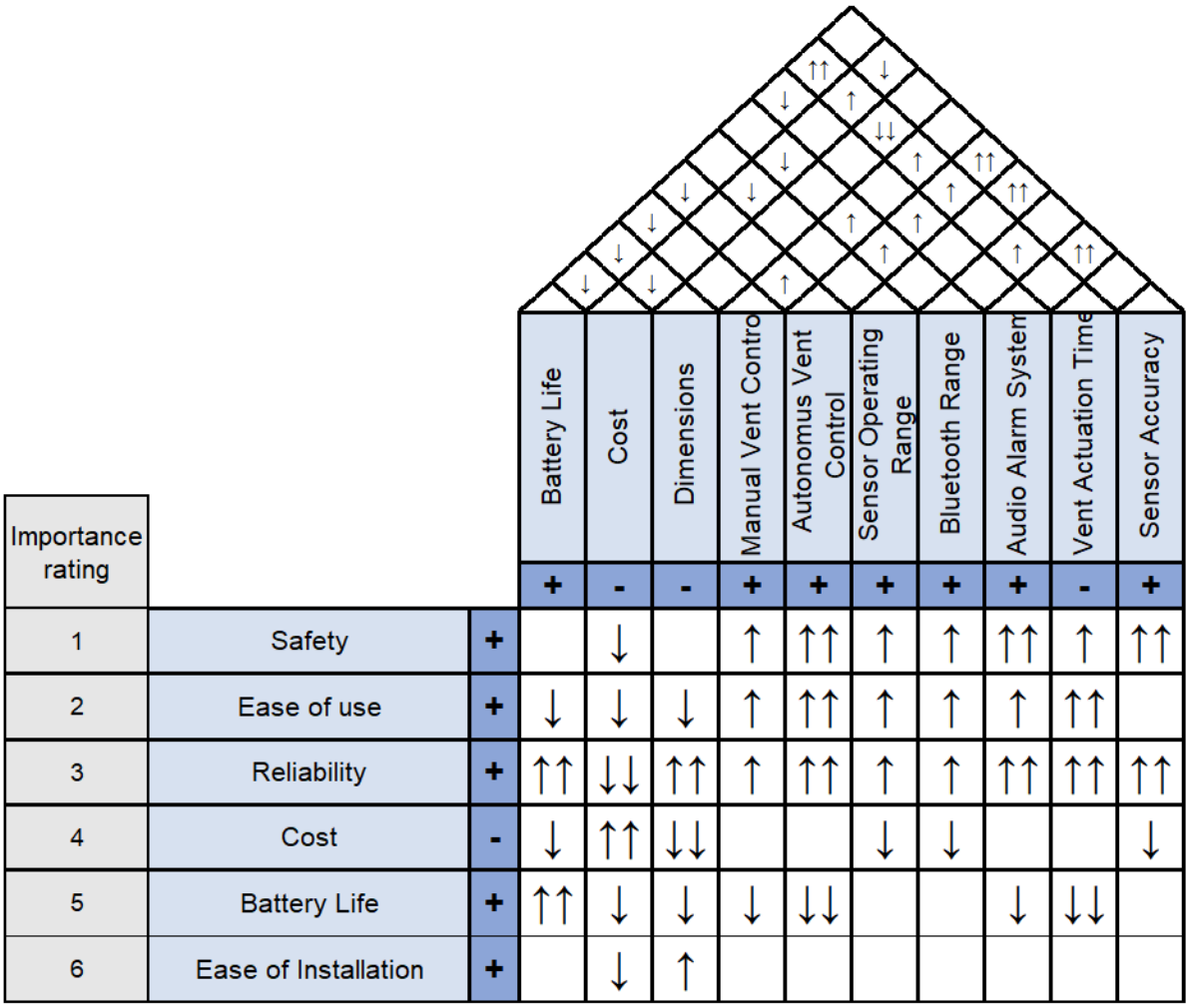


Figure 2: Software Flowchart

2.7 House of Quality

Showcasing the polarities as well as correlations between the functional requirements and the customer requirements the House of Quality diagram is shown below.



Importance rating			Battery Life	Cost	Dimensions	Manual Vent Contro	Autonomus Vent Control	Sensor Operating Range	Bluetooth Range	Audio Alarm System	Vent Actuation Time	Sensor Accuracy
			+	-	-	+	+	+	+	+	-	+
1	Safety	+		↓		↑	↑↑	↑	↑	↑↑	↑	↑↑
2	Ease of use	+	↓	↓	↓	↑	↑↑	↑	↑	↑	↑↑	
3	Reliability	+	↑↑	↓↓	↑↑	↑	↑↑	↑	↑	↑↑	↑↑	↑↑
4	Cost	-	↓	↑↑	↓↓			↓	↓			↓
5	Battery Life	+	↑↑	↓	↓	↓	↓↓			↓	↓↓	
6	Ease of Installation	+		↓	↑							

Polarity		
+	·	-
Positive		Negative

Correlation		
↑		↓
Maximize	Target	Minimize

Figure 3. House of Quality diagram

3.0 Research

This research section offers an overview of the hardware and software components utilized in BREATHE. It details each component's role and emphasizes their function in achieving the project's objectives. The hardware components such as microcontrollers, sensors, LCD displays, and motors are described alongside their specifications and contributions to the system's overall functionality. The software aspect covers the programming languages, development environments, and libraries used for the system's firmware and user interface. By exploring both hardware and software, this section provides insight into the technological framework of BREATHE and how each component is essential for creating an efficient, responsive, and user-friendly smart vent system.

3.1 Microcontrollers

When selecting an MCU, there were three important factors that were considered: power consumption, integrated Bluetooth Low Energy (BLE), and ease of integration. To achieve the system’s desired lifespan of 6 months, it was crucial to choose an MCU that balanced both performance and energy efficiency. Ideally, the desired MCU should have a small form factor, a reasonable operating frequency, and low current consumption in both active modes and low power modes. Equally as important is the inclusion of integrated Bluetooth Low Energy (BLE), since Bluetooth was needed to communicate with the vent interaction subsystem’s MCU. This eliminated the need for an external Bluetooth module, simplifying the system’s design and further optimizing power usage. Lastly, ease of integration was another important factor. Given time constraints and adherence to the project’s specifications, we should prototype the system’s code on associated development boards while the PCB is being developed with the standalone chip. With these factors in mind, several MCUs were considered during the selection process.

ESP32: The ESP32 is considered to be a viable option for the project due to the team’s experience and familiarity with it, as well as its integrated BLE capabilities, affordability, and compact form factor. A drawback to the ESP32 is its relatively high power consumption, since it consumes between 278mA and 335mA [4]. This normally wouldn’t be ideal if the PSU had a limited capacity through a replaceable battery. The control unit in the project is powered through a wall outlet, and so this wouldn’t affect the project’s ability to meet our specifications. The ESP32 also includes two cores operating at around 160

MHz, allowing for fast computational and data processing benchmarks which is necessary for the heavy demands of the control unit. The ESP32 also supports many communication protocols such as SPI, I2C, and UART, allowing it to communicate with the many different peripherals within this project such as the LCD screen and temperature sensor. It also supports outputting PWM signals, allowing for proper operation of the passive buzzer. For the heavy demands of the control unit, this microcontroller is suitable, which is why we chose it.

nRF52840/nRF52832:The nRF52840, and its relative the nRF52832, is a ultra low powered microcontroller that supports BLE. They both utilize an ARM cortex processor operating at a frequency of 64MHz, which allows for fast computation and functionality. The main feature of the nRF52 series that makes it a desirable choice is its low power consumption. In its active mode, it draws about 3.3mA [5] and offers a variety of low power modes, including a low power mode and deep sleep mode, both drawing 2.35uA [5] and as low as 0.4uA [5] respectively. This is achieved by entering a SYSTEM OFF mode, which turns off the CPU. Although this would disable any BLE connections, there is an alternative low power mode known on SYSTEM ON which consumes about 2.35 uA while maintaining full 256 KB RAM retention and without severing BLE connections [5]. For BLE operations, the nRF52's current draw is 6.26 mA during transmission (Tx) and 5.2 mA during reception (Rx) [5]. While these ratings are relatively high, these operations will only last for a short period of time. This means that the nRF52 will allow the system to maintain a very low average current consumption while meeting the computational needs of the project, making it a great choice to meet our specifications.

EFR32BG22: Another option for the project was the EFR32BG22. It has an operation frequency of 76.8MHz, which is appropriate for the vent system's requirements [6]. Moreover, the EFR32BG22 offered efficient power consumption, drawing 42uA/MHz in its active mode. This allows for a sufficient level of flexibility, as the operating frequency can be manually configured to a desired level. Even at the highest possible operating frequency, the EFR32BG22 would only draw 3.23mA [6], being slightly more efficient than the nRF52840. The EFR32BG22 also offers various low power modes. When set to EM2 mode, the MCU consumes 1.94uA and at its lowest consumption mode, EM4, it consumes as low as 0.17uA [6]. One disadvantage regarding this MCU is that configuring it is often more complex than other MCUs. The chip supports multiple operational modes, each requiring specific setup for power management and peripheral functionality. For instance, developers must navigate intricate settings for low-power modes, communication protocols, and peripheral initialization. This complexity is highlighted by the interdependencies among different features; enabling one mode may impact the operation of

others, necessitating careful planning and testing. As a result, the EFR32BG22 offers efficient power consumption, great flexibility and capabilities, while its disadvantages lie in its configuration process [6].

DA14531: A further MCU option is the DA14531. The DA14531 was considered a suitable choice for an MCU due to its very low power consumption. Comparatively, the DA14531 consumes the least amount of power out of every considered MCU, making it the most power efficient selection. It draws an impressive 830 µA in its active mode and offers multiple low power options. Its Extended Sleep Mode draws 1 µA and Hibernation Mode draws 270 nA. Its operating frequency is also the lowest out of every MCU being considered, being at only 16MHz [7]. Additionally, it has an impressive form factor, being very compact at 1.7 × 2.05 mm [7]. Despite these advantages, the DA14531 has some limitations such as its processing power. The DA14531 uses an ARM Cortex-M0+, which is considerably less powerful than the EFR32BG22's ARM Cortex-M33 or the STM32's ARM Cortex-M4. Likewise, the DA14531 houses only 48 KB of RAM and lacks flash memory. While these limitations may not pose concerns for basic sensor data collection and transmission, they can impact the performance and functionality of the LCD screen. The LCD display typically requires a certain amount of memory and processing power to handle data updates. The limited RAM and lower processing power of the DA14531 could restrict the responsiveness of the LCD interface, leading to slower performance or difficulties in handling complex graphics. This constraint could also limit the ability to implement features such as animations, real-time data updates, or a user-friendly interface with multiple menus or options. Nonetheless, the DA14531 is by far the most power-efficient choice for a MCU, however it may lack the necessary processing power needed for the entire system [7].

CC2640R2F: The CC2640R2F was another potential MCU option for the project. Like the other aforementioned MCU's, it has low power consumption. Its active mode consumes 1.45 mA + 31µA/MHz, and at its maximum operating frequency it'll draw only 2.938mA, making it slightly more efficient than many of its alternatives. In its standby mode, which is one of several low power modes, draws 1uA, and its shutdown mode draws 0.15uA. While the CC2640R2F is offered as a SoC, it is also offered as a development board, which adheres to an ease of prototyping. However, a downside of the CC2640R2F is its limited memory. Although the CC2640R2F has flash memory, its value is considerably less than other options, only being at 128KB. Moreover, it only has 28 KB of SRAM. The lack of memory that the CC2640R2F has poses a similar concern with the DA14531, where it might not be powerful enough to handle the LCD

screen. Furthermore, the development board of the CC2640R2F is quite expensive, costing $29 [8].

STM32: Another viable option for an MCU was the STM32. Like previously mentioned MCUs, the STM32 not only has an operating frequency of 64MHz, but also has relatively low current draw. Its active mode consumes 53uA/MHz. Therefore at its maximum operating frequency of 64MHz, the STM32 will draw 3.39mA. A significant advantage the STM32 has is its extensive selection of low power modes. For instance, Stop Mode 2 draws only 0.5uA while retaining RAM and allows for a fast wake-up time. Likewise, the STM32 provides the availability of software libraries, such as TouchGFX. TouchGFX is a powerful graphics library designed for the STM32, and is optimized for touchscreen interfaces. Access to TouchGFX streamlines the development process, as it provides a wide range of pre-built functions and graphical elements. Despite its advantages in power-efficiency, the STM32's active mode does have slightly higher power consumption compared to the other MCUs, especially the DA14531 and the CC2640R2F. However, the variation in comparison is minimal, typically ranging only a few milliamperes. A more notable downside is cost, as STM32 development kits can be quite expensive compared to alternatives. For instance, the STM32WB5MM Development Kit costs $25 [9].The STM32 stands as a strong choice due to its performance and low power consumption.

MCU	Frequency	Run/Active Mode Current Consumption	LPM Current Consumption	BLE Tx/Rx Current Transmission
ESP32-C3	160MHz	TX: 278 mA - 335 mA RX: 84 mA - 87 mA	Light Sleep: 130 µA Deep Sleep: 5 µA Modem Sleep: 13 mA - 28 mA	RX: 84 mA - 87 mA TX: 278 mA - 335 mA
nRF52XX	64 MHz	Run Mode (Using DC/DC Regulator + Flash Memory): 3.3mA	Deep Sleep (System OFF Mode): 0.4 µA - 1.86 µA Low Power (System ON, wake up from interrupt): ~2.35 µA	RX: 6.26 mA TX: 5.2 mA

STM32WB09XX	64 MHz	Run Mode: ~2.679 mA - ~3.85 mA	Low Power Sleep Mode (1 MHz): 1.023 µA - 12.929 µA Shutdown Mode: 20 nA	RX: 3.6 mA TX: 4.9 mA
DA14531	16 MHz	Run Mode: 830 µA Run Mode with CPU Idle: 5 mA	Extended Sleep Mode: 1 µA Hibernation Mode: 270 nA	RX: 2.2 mA TX: 3.5 mA
EFR32BG22	76.8 MHz	Active Mode (EM0): 42 µA/MHz	Low Power Mode (EM2): 1.94 µA Lowest Mode (EM4): 0.17 µA - 0.43 µA	RX: 2.5 mA TX: 3.4 mA
CC2640R2F	48 MHz	Active Mode: 1.45 mA + 31 µA/MHz	Idle Mode: 650 µA Standby Mode: 1 µA Shutdown Mode: 0.15 µA	RX: 5.9 mA TX: 6.1 mA

Table 2: MCU Comparisons

3.1.1 Final MCU Selection and Justification

In conclusion, the selection of the MCU for the smart vent system involved a thorough analysis of various options, each with its own strengths and weaknesses in terms of power consumption, ease of integration, overall ability to process tasks and data in a timely manner, and various communication protocols such as SPI and Bluetooth Low Energy (BLE) capabilities.

The EFR32BG22 showcased superior efficiency at higher frequencies but presented a complex configuration process that could impact development time constraints. On the other hand, the DA14531 excelled in power efficiency, with the lowest consumption rate, but its limited processing capabilities raised concerns about operating the LCD screen effectively. The CC2640R2F provided low power consumption and easy prototyping but was limited by its memory capacity. The STM32 offered a combination of performance and efficiency, and showed to be a strong potential candidate. However, other MCUs offered lower current consumption, which was a necessity given B.R.E.A.T.H.E's constraints.

Likewise, the limited GPIO pins and uncertain ease of implementation makes this choice weaker compared to the alternative options. Finally, the ESP32 and nRF52 were shown to be the strongest candidates for the roles that were required for their respective applications. The ESP32 is shown to be a very strong MCU with a high processing speed and variety of pins, allowing for seamless functionality of multiple necessary peripherals such as the TFT LCD screen and sensors. Although the MCU does consume a higher amount of current compared to the other options, this will not be an issue since there will not be a limited power supply such as a battery on the unit that the ESP32 will be integrated on. The nRF52 is compatible with the Arduino IDE which will allow for easier prototype integration, and also supports the Nordic SDK which has a lot of documentation. Similar to the ESP32, it also supports BLE and PWM signals which allows for operations of the motor while maintaining a continuous connection with another MCU. It also consumes a very low amount of current, which maximizes the battery life of the vent unit. These two MCUs allow the project to perform all of the necessary functions, providing a basis for BREATHE to reach its goals and specifications.

3.2 Sensor Selection

3.2.1 Introduction

BREATHE will incorporate two key sensors: a temperature sensor and a CO2 sensor. These sensors are integral to the smart vent system, as they will continuously monitor environmental conditions in order to ensure optimal airflow. The temperature sensor will ensure the vent system adjusts ventilation based on current room conditions while the CO2 sensor will monitor the air quality, where both are crucial for a safe and comfortable living indoor environment. The collected data is sent to the MCU for further processing and control, allowing the vent to open or close autonomously based on real-time room conditions. This data is also displayed on an LCD screen, giving users the option to manually override the system if needed. Moreover, collection of reliable data from the sensors will improve safety since the sensors will be able to detect dangerous levels of temperature or CO2. This will allow the system to adjust airflow accordingly or alert the user, thus helping to prevent any potential dangers.

3.2.2 Selection Criteria

When selecting suitables sensors for the project, there are several criteria that need to be considered. One important aspect is accuracy. The sensor must

provide precise readings so that the system can adjust airflow based on accurate environmental data.. In adherence to the project's specifications, the temperature sensor should have an accuracy of ±1°F and the CO2 sensor should be accurate to ±50 ppm. This will ensure the system performs as expected. Another crucial factor is its operating range. The sensor must accommodate both a wide range of temperatures and CO2 levels to ensure proper operation under various environmental conditions. According to the project specifications, the temperature sensor must function within a range of -40°F to 176°F, while the CO2 sensor must function within a range of 0 ppm to 5000 ppm. Having a wide temperature range allows the system to remain effective in both extreme heat and cold, ensuring it can monitor and control ventilation in nearly any situation. Additionally, a wide CO2 range allows the system to effectively detect and respond to varying air quality levels, further promoting a healthy indoor environment.

3.2.3 Types of Temperature Sensors

With respect to temperature sensors, there are several different types with each offering distinct advantages. These sensors vary in terms of accuracy, integrated features, and communication protocols. The ideal sensor will involve carefully evaluating these factors and making trade-offs based on specific requirements of the vent system. For instance, higher accuracy may come at the cost of a slower response time, while a more expensive sensor may come with greater accuracy. Thus, the objective is to balance these considerations to select the most suitable sensor for the application.

Analog temperature sensors: Analog temperature sensors output a continuous voltage signal that is linearly proportional to temperature [10]. Moreover, they are relatively simple to integrate with MCUs. They offer reasonable accuracy, typically ranging from ±0.28 to 0.56°C/±0.5 to 1°F and exhibit low power consumption. Likewise, they have a typical operating range from -50°C to 150°C/-58°F to 302°F [11]. However, generally over time collected data tends to drift which can lead to inaccuracies and interrupt operations, therefore calibration can be used to mitigate this.

Digital temperature sensors: Digital temperature sensors convert temperature to digital signals. They consist of a sensing element that detects temperature and conditions the detection through signal conditioning. Digital temperature sensors offer similar benefits to analog temperature sensors in the sense that they offer low power consumption, have similar accuracy, and are easy to integrate with MCUs. They also have a similar operating range, typically spanning from -40°C

to 125°C/-40°F to 257°F. However, they are generally more expensive than Analog Temperature Sensors [11].

Thermocouples: are versatile sensors suitable for measuring extreme temperatures. They consist of two dissimilar metals joined at one end, creating a voltage that varies with temperature. Thermocouples can operate over a wide range, typically from -200°C to 1250°C/-328°F to 2282°F, but they require additional signal conditioning to produce accurate readings. Their advantages include high-temperature capability and durability, but they may have lower accuracy compared to other sensor types, ranging from ±1 to 2°C/±1.8 to3.6°F [11].

Thermistors: are temperature-sensitive resistors that change resistance with temperature. They are highly sensitive and accurate to ±0.1 to 0.2°C/±0.18 to 0.36°F within a limited temperature range, typically from -50°C to 150°C/-58°F to +302°F. While they offer high precision and low power consumption, thermistors can be nonlinear and require more complex calibration methods to maintain accuracy [11].

RTDs (Resistance Temperature Detectors): use the principle of resistance change in metals to measure temperature. They offer high accuracy ranging from ±0.1 to 0.5°C/±0.18 to 0.9°F, stability, and a wide operating range, generally from -200°C to 850°C/-328°F to +1562°F. However, they require precise signal conditioning and a constant current source for accurate measurements, which can make integration moderately complex. RTDs are also more sensitive to lead resistance, often needing 3-wire or 4-wire configurations to reduce errors. Additionally, RTDs tend to be more expensive and can be less robust than thermocouples in harsh environments [11].

Sensor Type	Accuracy in °F	Temperature Range in °F	Ease of Integration
Analog	±0.5-1°F	-58°F to +302°F	Easy
Digital	±0.5-1°F	-40°F to +257°F	Easy
Thermocouples	±1.8-3.6°F	-328°F to +2282°F	Difficult, requires signal conditioning
Thermistors	±0.18-0.36°F	-58°F to +302°	Moderate, requires calibration

RTDs	±0.18-0.9°F	-328°F to +1562°F	Moderate, may require external ICs

Table 3: Sensor Type Comparisons

3.2.4 Temperature sensor type selection

Upon further research, it has been decided that analog and digital temperature sensors are the most suitable options for the smart vent system. These two sensors offer simplicity in integration and sufficient compatibility with the ESP32, making them the most practical choices. Digital temperature sensors are particularly advantageous due to their straightforward implementation. They typically communicate with the ESP32 via communication protocols such as I2C and SPI, which eliminates the need for additional circuitry. Since necessary procedures for sensing such as signal conditioning and analog to digital conversion occur within the sensor itself, no external signal processing components are needed. This not only significantly reduces design complexity, but also improves accuracy by minimizing external factors such as noise.

Similarly, analog temperature sensors provide simple integration, requiring only a direct connection to the MCU. The ESP32 built-in ADC efficiently handles the conversion of the analog signals into digital values, eliminating the need for additional external signal processing circuitry. Likewise, they provide a linear output that is easily interpreted by the ESP32, further simplifying system design.

Thermistors are relatively easy to integrate as well, often requiring only a single resistor or voltage divider. However, their nonlinear output can complicate obtaining accurate readings, introducing a layer of complexity in calibration and signal interpretation.

RTDs provide high accuracy, however require more elaborate external circuitry. Components such as an external constant current source,signal conditioning components, and possibly dedicated RTD IC’s are needed in order to handle amplification and changes in resistance. These additional elements will unnecessarily complexify the implementation of the sensors, making them less desirable to choose as a temperature sensor.

Lastly, thermocouples were considered to be unsuitable for the smart vent system. One significant disadvantage is its form factor; a thermocouple is not ideal for the smart vent system due to its relatively loose and larger form factor, which complicates its integration with other components such as the LCD display. In conclusion, analog and digital sensors offer the best balance of

simplicity, accuracy, and ease of integration for this specific smart vent system. Their reduced need for external components and ease of communication with the ESP32 make them the most efficient and practical choices.

3.2.5 Temperature Sensor Options

Outlined below are analog and digital temperature sensors that are under consideration for the smart vent system:

1. The LM35 is a precision analog temperature sensor developed by Texas Instruments and was considered as a viable temperature sensor for various reasons. It is known for its linear voltage output directly proportional to temperature in Celsius, which disregards the need of complex signal processing or conversions. This allows for an ease of integration with components such as microcontrollers, as less computational resources and memory would be utilized. Moreover, This sensor is designed for applications requiring precise and reliable temperature measurements, with built-in calibration that eliminates the need for implementing external calibration methods. This is a very substantial advantage, as this also reduces software design complexity. The LM35 also provides an accuracy of ±0.5°C/0.9°F. and operates over a range of 4V to 30V [12]. It also has an impressive temperature range, spanning from -55°C to 150°C/-67F to 302F, which surpasses the smart vent's temperature range specifications.Despite these advantages, one potential downside is found in its operating range. The smart vent's selected MCU, the ESP32, operates at a voltage rating of 3.3V, while the LM35 requires a minimum operating voltage of 4V. While the LM35 could be connected to an external power supply and share a ground with the ESP32, this approach may require using two separate voltage rails in the LCD unit. This setup would increase the board footprint and add to the overall cost. Despite the challenge posed by its higher minimum operating voltage, the LM35 remains a strong candidate due to its precision, reliability, and ease of integration.

2. The TMP36 is another suitable option for a temperature sensor, offering many of the same advantages as the LM35. This low-voltage, precision sensor also outputs a voltage linearly proportional to the Celsius temperature, simplifying integration with microcontrollers and other electronic components. Like the LM35, it has integrated self-calibration measures that ensures consistent and reliable temperature readings over extended periods of time. Although its accuracy is slightly lower than that of the LM35, it provides sufficient accuracy ranging from ±1 to ±2°C/±1.8

to ±3.6°F. Moreover, it has a very reasonable operating range, spanning from -40°C to 125°C/-40°F to 257°F, exceeding the smart vent's specifications. It also has an operating voltage range of 2.7 V to 5.5 V, ensuring compatibility with the 3.3V supply of the ESP32. Additionally, the sensor is stable when driving large capacitive loads and is designed for reliability across various environmental applications, ensuring consistent performance in the smart vent system.The sensor also benefits from low self-heating characteristics, which help maintain accurate readings without significant temperature drift [13]. Overall, the TMP36's ease of integration, adequate accuracy and operating range, and stability and self-heating characteristics make it a reliable option for a temperature sensor.

3. The DHT20 is a temperature and humidity sensor developed by Aosong Electronics. Although less than others, it has an operating range of -40°C to 80°C or -40°F to 176°F, still making it suitable for environments beyond the smart vent's specified range. Likewise, it provides sufficient temperature sensing accuracing, being identical to the LM35 at 0.5°C/±0.9°F, and comes pre-manufactured with full calibration [14]. With a voltage range between 2.2 and 5.5V, the DHT20 is well-suited for integration with the ESP32, similar to the TMP36, as it can operate within the MCU's 3.3V range without requiring external power sources. Additionally, since the DHT20 is a digital sensor, it communicates using the I2C protocol, which is natively supported by the ESP32, further simplifying integration. Moreover, the DHT20 has a very fast response time compared to the LM35 and TMP36. It reaches 63% of the final temperature value in just 0.8 seconds, whereas the LM35 takes between 20 to 30 seconds, and the TMP36 requires under 50 seconds in still air conditions. This rapid responsiveness makes the DHT20 highly suitable for real-time temperature monitoring applications [14].

4. SHT30: The SHT30 is a digital temperature sensor developed by Sensiron and was considered a suitable sensor for the smart vent. It integrates seamlessly with many MCUs including the ESP32, as it has an operating range from 2.15V to 5.5V, and uses the I2C protocol. Moreover, it offers an operating range similar to that of the previously mentioned sensors, with an operational range of -40°C to 125°C/-40°F to 257°F and an idle range of -40°C to 160°C/-40°F to 302°F. It comes fully calibrated and linearized, delivering temperature-compensated digital output with impressive accuracy of ±0.1°C [15]. The sensor also has a relatively fast response time, having a thermal time constant of approximately 2 seconds to reach 63% of its final value. However, a notable consideration

is that the SHT30 combines both temperature and humidity sensing, which may lead to increased power consumption and greater sensitivity to environmental factors such as moisture levels, potentially introducing inaccuracies in readings. In contrast, dedicated temperature sensors like the DHT20 or LM35 focus solely on thermal measurements, making them less susceptible to variations in humidity, airflow, and other environmental conditions.

3.2.6 Final Selection and Justification

After extensive research and consideration, the DHT20 was chosen as the sensor used for the smart vent. While each considered sensors were all suitable options for the project and shared similar benefits, the DHT20 distinguished itself with its ease of integration and rapid response time. When compared to its analog counterparts, the DHT20 simplifies integration through its I2C communication protocol, which is natively supported by the ESP32. More specifically, the DHT20's operating range is compatible with the ESP32, unlike the LM35, simplifying its integration within the smart vent system. Additionally, the DHT20 boasts the fastest response time of all the sensors, reaching 63% of its final value in just 0.8 seconds. With a reasonable accuracy of ±0.5°C/±0.9°F, the DHT20 meets the project's specifications and comes fully calibrated, removing the need for any external calibration. Despite having a slightly lower operating range than its alternatives, it still adheres to BREAHTE's requirements. Ultimately, the DHT20's combination of power efficiency, rapid response time, and reliable performance makes it the ideal choice for the smart vent system.

3.2.7 Summary Table of Sensor Specifications

Below is a table summarizing important specifications of each sensor.

Feature	LM35	TMP36	DHT20	SHT30
Accuracy	±0.5°C	±1°C to ±2°C	±0.5°C	±0.1°C
Temperature Range	-55°C to 150°C/-67°F to 302°F or	-40°C to 125°C/-40°F to 257°F	-40°C to 80°C /-40°F to 176°F	-40°C to 160°C/-40°F to 302°F Or -40°C to 125°C/-40°F to 257°F

Operating Voltage	4V to 30V	2.7V to 5.5V	2.2V to 5.5V	2.15V to 5.5V
Time Constant (Response Time)	20-30 seconds (still air)	<50 seconds (still air)	0.8 seconds	2 seconds

Table 4: Sensor Specifications Summary

3.2.8 Types of CO2 Sensors

Generally, there is less variety of CO2 sensors than temperature sensors. Unlike temperature sensors, which have a wide variety of types and are used in many different industries and applications, CO2 sensors are more specialized, focusing mainly on gas detection needs. This specialization results in fewer types of CO2 sensors overall because the applications are narrower, targeting specific industries like HVAC, environmental monitoring, and industrial safety. Moreover, CO2 sensors generally consume significantly more current than temperature sensors and are more expensive. Below are the most common CO2 sensors used:

Infrared (NDIR) Sensors: Non-Dispersive Infrared (NDIR) sensors measure amounts of wavelengths such as infrared light that is absorbed by CO2 molecules in the hair [6]. It operates by allowing air to flow into the sensor chamber. The sensor emits infrared light at a wavelength specific to CO2, typically around four microns. On the opposite end, a detector measures the amount of infrared light that passes through the air sample. If carbon dioxide is present, it absorbs a portion of the light, reducing the amount that reaches the detector. The level of absorption is directly related to the concentration of CO2 in the air—the more CO2, the more light is absorbed [16]. NDIR sensors offer several advantages such as high accuracy, typically in the range of ±50 ppm or better, moderate to fast response times, and minimal maintenance requirements. However, NDIR sensors are typically more expensive than other CO2 sensors.

Electrochemical Sensors: Electrochemical sensors detect CO2 by measuring changes in voltage or current using electrodes to determine how much CO2 is in the air [17]. A chemical reaction occurs when CO2 molecules interact with an electrolyte or electrode material inside the sensor, resulting in an electrical signal that relates to the concentration of CO2. Electrochemical sensors exhibit numerous advantages, such as high sensitivity, compact form factor, and sufficient energy efficiency. They also generally exhibit adequate accuracy, ranging from ±50-100 ppm. However, one drawback that electrochemical

sensors have is that they have a slower response time compared to other sensors. This is due to its reliance on the chemical reaction between the gasses and electrodes as well as additional signal processing conversions.

Metal Oxide Semiconductor Sensors (MOS): Metal Oxide Semiconductor (MOS) don't directly measure CO2, rather they use the resistivity of metal compounds to estimate the concentration of CO2 in the air. Compounds such as SnO2 (tin oxide) and WO3 (tungsten trioxide) are used as the sensing material to detect gasses [17].When CO2 or other gasses come into contact with the metal oxide surface, chemical reactions occur that alter the sensor's electrical resistance. This change in resistance is measured and used to estimate the concentration of CO2 in the air. Due to the way these sensors work, they offer benefits such as enhanced sensitivity, cost effectiveness, and fast response times. However, they are less accurate than other CO2 sensors, falling within a range of around ±100-200 ppm. Furthermore,due to their cross sensitivity to other gasses, it may in some cases lead to inaccuracies in measuring CO2 concentrations.

Photoacoustic Sensors: Photoacoustic CO_2 sensors operate by emitting modulated light into a chamber containing air. When the light is absorbed by CO_2 molecules, it causes them to heat up and expand, creating a pressure wave (sound wave) that is detected by a microphone. The strength of the sound wave correlates to the concentration of CO_2. These sensors exhibit excellent accuracy, with a typical range spanning from ±10-20 ppm, and are stable, but they tend to be more expensive and consume more power, making them less ideal for cost-sensitive or low-power applications.

Sensor Type	Response Time	Accuracy	Advantages	Disadvantages
Infrared (NDIR)	Moderate to Fast	±50 ppm	High accuracy, reliable	Larger size, higher cost
Electrochemical	Slow	±50-100 ppm	High sensitivity, compact, energy efficient	Slower response time due to chemical reactions
MOS	Fast	±100-200 ppm	Fast response time, cost-effective, sensitive	Sensitivity to other gasses may lead to inaccuracies

Photoacoustic	Moderate	±10-20 ppm	Highly accurate, stable	High cost, higher power consumption

Table 5: Carbon Dioxide Sensor Type Comparisons

3.2.9 CO2 sensor type selection

Upon further analysis, it has been decided that NDIR sensors will be the best fit for the smart vent system. They provide the ideal balance of accuracy and reliability, making them suitable for long-term use. While electrochemical sensors offer compactness and sensitivity, their slower response time makes them less optimal for this application. Similarly, metal oxide semiconductor (MOS) sensors may offer fast response times and cost-effectiveness, but their cross-sensitivity to other gasses could result in measurement inaccuracies. Given the smart vent system's need for accurate, real time CO_2 monitoring, NDIR sensors, such as the MH-Z19B or SenseAir S8, offer the most appropriate solution.

3.2.10 CO2 Sensor Options

Outlined below are various CO2 sensors that were considered for the smart vent system:

MH-Z19B: The MH-Z19B is a commonly used NDIR CO2 sensor that provides many advantages. It offers sufficient accuracy of ±50 ppm and has a range spanning from 0 to 5000 ppm, aligning with the specifications of the smart vent system. It also allows easy integration with the ESP32, as it communicates using UART and Pulse Width Modulation (PWM). However, a downside is that the has a relatively slow response time, that being of 120 seconds to reach 90%, of the final reading, which may limit its effectiveness in an environment where rapid CO2 level changes can occur [18].

SEN0159: The SEN0159, which uses the MG-811 sensor, is another NDIR CO_2 sensor being considered for the smart vent. It features a high sensitivity to CO2 while exhibiting less sensitivity to alcohol and Carbon Monoxide, which ensures both reliability and minimal interference. It also features an onboard heating circuit optimized for sensor performance and includes a built-in conditioning circuit for signal amplification. Wth a measurement range extending from 0 to 10,000 ppm, it surpasses the MH-Z19B in terms of range. It boasts an impressive response time of 20 seconds, significantly faster than both the SenseAir S8 and the MH-Z19B. The accuracy is consistent with the MH-Z19B,

but its broader range makes it better suited for environments with higher CO_2 concentrations. Overall, the SEN0159's faster response time and broader ppm range make it a strong contender in scenarios where quick detection and a wider range are necessary [19].

SenseAir S8: Another sensor we considered was the SenseAir S8. It is an NDIR CO_2 sensor with a measurement range spanning from 0 to 20000 ppm, which is higher than both the MH-Z19B and the SEN0159. The response time is around 30 seconds, which is faster than the MH-Z19B but slower than the SEN0159. It communicates using UART, making integration with the ESP32 straightforward. The SenseAir S8's accuracy of ±0.02% volume CO_2 ±3% of reading can be a downside, especially in low CO_2 environments, where the fixed error of ±200 ppm may result in significant deviations from the true value. Additionally, at higher concentrations, the combined error can exceed ±350 ppm, potentially affecting the smart vent system. Overall, the SenseAir S8 provides an ease of integration, though its slower response time and possible inaccuracy problems are limiting factors compared to the SEN0159 [20].

SCD41: The SCD41 is an advanced NDIR CO_2 sensor designed for high accuracy and compact applications. It provides a measurement range of 0 to 40,000 ppm, significantly surpassing the other sensors, which makes it suitable for environments with extreme CO_2 levels. In terms of accuracy, it offers an impressive precision of ±(40 ppm + 5% of reading), making it one of the most accurate sensors in its category. It also has a response time of 60 seconds, faster than the MH-Z19B but slightly slower than the SEN0159. The combination of high accuracy and a wide ppm range makes the SCD41 a great choice for demanding applications, although its higher price point could be a limitation for budget-conscious projects [21].

3.2.11 Final Selection and Justification

The SCD41 has been chosen for the smart vent system due to its exceptional performance in critical areas such as accuracy, measurement range, and reliability. With a measurement range of 0 to 40,000 ppm, the SCD41 is uniquely capable of handling a wider spectrum of CO_2 concentrations compared to other sensors, making it ideal for diverse environments, including those with potentially high CO_2 levels.

In terms of accuracy, the SCD41 provides a precision of ±(40 ppm + 5% of reading), ensuring reliable measurements that are crucial for effective air quality management. This level of accuracy enhances the system's ability to respond to varying CO_2 levels promptly, improving overall performance and effectiveness.

Moreover, the SCD41 boasts a response time of 60 seconds, which allows for timely adjustments based on real-time CO_2 measurements.

While the SCD41's price point is higher than some alternatives, its combination of high accuracy and extensive range justifies the investment. The capabilities of the SCD41 make it a robust choice for applications that demand precision and reliability in monitoring CO_2 levels, ensuring that the smart vent system operates effectively in maintaining optimal air quality.

	MH-Z19B	SenseAir S8	SCD41	SEN0159
Accuracy	± 50ppm	±0.02% volume CO_2 ±3% of reading	± 40 ppm	± 50 ppm
PPM range	0 to 5000 ppm	0 to 20,000 ppm	0 ppm to 40,000 ppm	0 to 10,000 ppm
Response Time	T90 < 120 s	Varies, typically very fast (ms)	60 seconds to reach 63%	< 3 minutes for 90% step change
Cost	$29-$38	$46-$59	$37	$49

Table 6: Carbon Dioxide Sensor Comparison

3.3 LCD Screen selection

3.3.1 Introduction

The LCD screen plays an important role in the smart vent system by serving numerous essential functions. It will display environmental data gathered from the sensors, providing users with real time information. Additionally, the LCD screen will also act as the control mechanism for the vent system, allowing users to manually adjust the vent cover if deserted. Lastly, the LCD screen will also display warnings when hazardous levels of either CO2 or temperature are detected, making sure the smart vent system warns users of potentially dangerous environmental conditions.

3.3.2 Selection Criteria

When selecting an LCD screen, several criteria must be considered to ensure optimal performance and usability. Like other elements of BREATHE, such as the MCU and sensors, low power consumption is a crucial factor for the LCD. Given that the LCD screen will likely be used for extended periods, it is important to choose a power-efficient display. A display that consumes minimal power will extend the overall battery life of the system, making it ideal for long-term operation in various environments.

Moreover, it is essential for the LCD screen to integrate seamlessly with the ESP32 microcontroller. This integration involves compatibility with communication protocols, such as SPI or I2C, allowing for efficient data transfer and control. An LCD that can easily interface with the ESP32 will not only streamline development, but also enhance system performance. This can include the availability of graphic libraries that facilitate quick implementation and configuration.

Another important criterion is touchscreen capability. A touchscreen LCD provides an intuitive user interface, allowing users to navigate through the system efficiently. The interactivity of a touchscreen can lead to a friendly user experience, making it easier to access various features and environmental data.

Lastly, form factor is another critical element. It is essential to find a balance between visibility and compactness. The LCD should be large enough to ensure that information is easily readable, yet not so large that it obstructs other components or hinders overall functionality. A well-designed form factor enhances user interaction while maintaining an efficient use of space.

3.3.3 Types of Touch Screen LCDs

There are numerous types of touch screen each with its own technology and applications. Each type of touchscreen has its own unique advantages and limitations, which makes choosing a desirable type essential for the smart vent system.

Resistive Touchscreen: Resistive touch screens are one of the most common types of touch screen LCDs, with applications found in medical and military settings [22]. Likewise, they typically can be used with a pointing device, such as a stylus. They utilize changes in resistance caused by pressure applied to the screen for accurate touch positioning [22]. They are designed with multiple layers, including a top flexible layer and bottom rigid layer, each coated with conductive material. When pressure is applied, the layers make contact,

allowing the system to measure resistance changes and determine the touch coordinates. Resistive touchscreens can be further divided into two distinct types:

- *Four-Wire Resistive Touch Screen:* This design uses four distinct wires to measure touch coordinates. It features a top and bottom conductive layers of resistive film [22], each layer connected with two wires. These layers are separated by spacers, and when the screen is touched, they make contact at the touch point, allowing the system to detect the X and Y coordinates by measuring voltage changes across the four wires.
- *Five-Wire Resistive Touch Screen*: This design has a very similar structure to the four-wire type, however it uses an inner and outer conductive layer with a fifth wire connected to it. Similar to the four-wire type but includes an additional wire for improved accuracy and durability. When touched, the conductive layer on the glass panel makes contact with the sensing layer, creating a change in resistance near the touch point. This variation is measured by the circuit to calculate the exact touch coordinates [22].

Resistive Touchscreens have advantageous characteristics, with one being their durability. They have a strong resistance to dust, oil, and moisture. A second advantage is high accuracy. This is highlighted in the five-wire type, as its fifth wire helps measure resistance values more accurately and provides more precise touch position detection [22]. Lastly, it requires low touch pressure, making it highly accessible and convenient for users. However, a disadvantage can be found in the four-wire configuration, which is its susceptibility to wear. The top conductive layer can develop cracks over time due to frequent use, leading to inaccuracies. This issue can be mitigated by opting for the more durable five-wire version. Below is a table comparing the two types of resistive touch screens.

Type	Description	Advantages	Disadvantages
Four-Wire Resistive	Uses four distinct wires connected to conductive layers to detect touch	Cost-effective and simple design	Prone to wear and reduced accuracy over time due to frequent use
Five-Wire Resistive	Includes an additional fifth wire for improved measurement accuracy and durability.	More precise, longer-lasting than the four-wire type.	More expensive compared to the four-wire version.

Table 7: LCD Resistive Touch Screen Type Comparisons

Capacitive Touchscreen: Capacitive touchscreens are undoubtedly the most common type of touchscreens used. They are utilized in a plethora of applications, such as today's smartphones, tablets, and laptops. Capacitive touchscreens consist of a glass panel coated with a conductive layer and an induction electrode. When untouched, the electric field formed by the induction electrode remains uniform. Touching the screen with a finger or stylus alters the field, adding capacitance at the touch point. The induction electrode detects this change, and the control circuit measures it to determine the touch coordinates [22]. Capacitive Touchscreens can be further categorized into four types:

- *Surface Capacitive Touch*: This type consists of a single ITO conductive layer on a glass panel with a protective coating. Four electrodes are connected to the corners. Touching the screen allows charge to ground through the user's body, which enables the controller to determine the touch position. However, it only supports single-touch detection.

- *Projection Capacitive Touchscreen:* The self-capacitive design includes two ITO conductive layers, with a control circuit applying driving signals to small coils. When touched, the screen senses changes in the electric field and determines the touch position. However, it lacks the multi-touch capability of projection capacitive screens.

- *Self-Capacitance Type:* The self-capacitive design includes two ITO conductive layers, with a control circuit applying driving signals to small coils. When touched, the screen senses changes in the electric field and determines the touch position. However, it lacks the multi-touch capability of projection capacitive screens.

- *Mutual Capacitance Type:* This type also has two ITO layers, but it excels in multi-touch accuracy and sensitivity. The control circuit detects changes in the electric field when touched, and the screen can differentiate between multiple touch points. Mutual capacitance has superior wiring simplicity, fast acquisition speed, low power consumption, and excellent anti-interference capability, making it ideal for interactive applications.

Given the variety of capacitive touch screens, each carries their own advantages and disadvantages. For instance, projection capacitive touch screens support multitouch, but are more complex and costly than surface capacitive touch screens. Likewise, mutual capacitance completely outperforms self-capacitance touch screens across a variety of metrics. They have more simple wiring, have fast acquisition speed, low power dissipation, have a higher signal stability, and support multitouch.

Furthermore, capacitive touch screens generally have several advantages over resistive touch screens. They are much more sensitive than resistive touch screens, responding effortlessly to light and gentle points of contact, making them easier and more intuitive to use. Additionally, they have a longer service life, as they don't rely on physical pressure of a point of contact for operation. Moreover, a distinguishable advantage is that some capacitive touch screens can support multi-touch capabilities, which can allow for operations such as pinching and zooming. Lastly, mutual capacitance offers a great advantage, that being low power dissipation, which can be very beneficial to the smart vent system. The biggest disadvantage that capacitive touch screens have is that they are more expensive than resistive touch screens due to complex technology involved in their construction, such as multi-layered designs.

The following table presents a comparison of capacitive touch screens.

Type	Description	Advantages	Disadvantages
Surface Capacitive	Single ITO conductive layer with electrodes at the corners of a glass panel	Durable and reliable for basic touch applications	Limited to single-touch detection
Projection Capacitive	Dual ITO layers that support multi-touch and high sensitivity	Multi-touch capability, fast response time	Higher cost and complexity
Self-Capacitance	Two ITO layers, detects touch by sensing electric field changes	Simple wiring and fast touch detection	Lacks multi-touch capabilities
Mutual Capacitance	Excels in detecting multiple touch points accurately with two ITO layers.	Supports multi-touch, high signal stability, low power consumption, fast response	More expensive due to advanced technology

Table 8: LCD Touch Screen Type Comparisons

Acoustic Wave Touch screens: Acoustic wave touch screens are another specialized type of touch interface, with more niche applications compared to

capacitive and resistive screens. These screens are commonly used in environments such as ATMs, information kiosks, and touch-screen arcade machines [23]. Acoustic wave touch screens are classified into different types:

- *Surface Acoustic Wave (SAW) Touch Screens:* The most common variation of this particular type are surface acoustic wave (SAW) touch screens. They use ultrasonic transmitters and receivers to calculate x and y coordinates. When touched, the screen disrupts acoustic waves, allowing a point of contact to be detected. As per their application, they are excellent for large sized screens with persistent durability, however they are susceptible to contaminants and random unintended touches.

- *Bending Wave Scheme:*Another variation includes the bending wave scheme, which detects sound waves generated by tapping the touch screen. This is further categorized into two approaches, acoustic pulse recognition (APR) and dispersive signal technology (DST).

    - *Acoustic Pulse Recognition (APR):* APR touch screens use piezoelectric transducers to detect touch positions based on stored wave data. While effective, APR requires an extensive prior process to store bending wave data across the screen. This approach is prone to errors in touch location estimation, supports only single-touch input, and depends on sufficient tapping strength [23].

    - *Dispersive Signal Technology (DST):* DST processes measured bending wave data without pre-stored information, but suffers from similar limitations, including single-touch support, high tapping strength, and computational demands [23]. Similar to APR touch screens, they can only detect single touch inputs.

There are numerous advantages and drawbacks of acoustic wave touch screens. As mentioned earlier, they are highly durable, allowing for a long lifespan and resistance to wear. Moreover, they possess higher sensitivity than resistive touch screens, as they depend on acoustic wave disruption, rather than physical pressure. Lastly, they can also detect some non conductive materials, such as microfibers or wool. However, unlike resistive touch screens, they are more susceptible to contaminants, such as dirt or moisture. Additionally, they are also prone to errors in touch detection, such as false touches or unresponsiveness, as a result of variance in data processing. Specific to the DST configuration, they require significant processing power to process bending wave data, which can in turn lead to more power consumption.

A table comparing the two types of acoustic wave touch screens is shown below.

Type	Description	Advantages	Disadvantages
Surface Acoustic Wave (SAW)	Uses ultrasonic transmitters and receivers to detect touch through wave disruption	Ideal for large screens, durable, good sensitivity	Susceptible to contaminants such as dirt and moisture
Bending Wave Scheme (APR/DST)	Detects sound waves generated by touch and processes wave data	Durable with a unique detection method	Requires high tapping strength, supports only single-touch, computationally demanding

Table 9: Acoustic WaveLCD Touch Screen Type Comparisons

Optical Touch Screens: Another type of touch screens are optical touch screens. They have similar applications to acoustic touch screens, such as arcade machines, kiosks, and aviation [24]. They work by placing IR transmitters and receivers on opposite sides of the screen. When a touch blocks the IR light path, the x and y coordinates are calculated by detecting which receivers don't receive the signal. However, this design requires a raised bezel for the IR components, and multiple touches can result in ghost touches. There are two primary types of optical touch screens:

- *Planar Scatter Detection (PSD):* This method sends IR light through a waveguide under total internal reflection (TIR). When touched, the TIR is interrupted, and scattered light is detected to determine the touch location. PSD supports multi-touch and offers high image clarity but requires significant computational power for large displays.
- *Frustrated Total Internal Reflection (FTIR):* FTIR also uses TIR, but the touch location is determined by the IR light that escapes toward the opposite side of the touch. This light is captured by cameras or vision sensors, which generate images to identify the touch points.

Benefits of optical touch screens include their ability to support multiple touch points, their compatibility with large displays, and excellent visual clarity due to the absence of additional layers that can obstruct the view. However,

disadvantages include the need for a raised bezel to accommodate IR transmitters and receivers, They are also prone to ghost touches, where unintended inputs may be registered due to multiple simultaneous touches. Moreover, the extraction of touch locations for larger screens requires significant computational power, increasing the overall system demand.

Displayed below is a table that compares the two types of optical touch screens.

Type	Description	Advantages	Disadvantages
Planar Scatter Detection (PSD)	IR light is sent through a waveguide and interrupted by touch, scattering light to determine location	Supports multi-touch, high image clarity	Requires significant computational power for large displays
Frustrated Total Internal Reflection (FTIR)	Detects touch by capturing IR light that escapes during touch using cameras or vision sensors	Multi-touch support, excellent visual clarity	Complex setup, higher computational demands

Table 10: Optical LCD Touch Screen Type Comparisons

3.3.4 LCD type selection

After further analysis, the best suited options were undoubtedly either resistive touch screens or capacitive touch screens. Capacitive touch screens are advantageous due to their high sensitivity and ability to support multi-touch interactions, allowing for a friendly and seamless user experience. Their long lifespan and durability make them suitable for environments where the system will be in frequent use. However, the higher cost of capacitive screens compared to resistive options is a consideration. On the other hand, resistive touch screens offer durability and accuracy at a lower price point, making them a viable alternative. Although acoustic wave touch screens and optical touch screens have their benefits, they simply aren't suitable for the smart vent system. One major drawback is their typically large size, which is unnecessary for this application. A more compact interface would be sufficient to display essential environmental data and controls. Moreover, it would be difficult to even interface one with an ESP32, as they require additional hardware. Acoustic wave touch screens would require a specialized controller, and optical touch screens would require hardware such as IR transmitters, receivers and cameras. Lastly, these hardware would consume large amounts of power, making the overall vent

system inefficient. Conclusively, acoustic wave and optical touch screens are not only practically unsuitable for the smart vent system, but also introduces additional overhead and complexity. Conversely, resistive and capacitive touch screens present the most effective solutions for the smart vent system, balancing performance, cost, and ease of integration while ensuring a user-friendly interface.

3.3.5 LCD Options

DFR0669 3.5" IPS Capacitive Touchscreen Display: The DFR0669 3.5" IPS capacitive touchscreen display is a considered option for a LCD touchscreen. This module has a resolution of 480x320, utilizes the ILI9488 driver IC, and uses SPI for communication. An advantage that the DFR0669 has is that it uses the GT911 chip instead of a I2C connection, which allows the screen to handle multiple points of contact effectively. Moreover, it offers a reasonable resolution and size which can be particularly advantageous for displaying user-friendly interfaces. It offers an operating voltage between 3.3 to 5V [25].

2.8" Capacitive TFT Touch Shield: The 2.8 Capacitive TFT Touch Shield is another capacitive display that is being considered for BREATHE. Due to its smaller form factor, it has a smaller resolution than the DFR0669, being at 240x320 [26]. Likewise, it uses an ILI9341 driver, similar to the WaveShare 3.2" LCD screen. Moreover, its FT6206 capacitive touch screen driver, which communicates through SPI, provides a multitude of advantages such as self-capacitive sensing, auto calibration, and an integrated MCU [27]. Similarly to the 3.2" WaveShare LCD screen, it offers 262K colors and has an operating voltage of 3.3V [26].

WaveShare 4" Resistive Touch LCD: The WaveShare 4" Resistive Touch LCD is a resistive touch screen with a resolution of 480x320 [28]. It uses the ILI9486 driver [29], communicates through SPI, and utilizes a XPT2046 touch controller . Moreover, its backlight pin is controlled through an adjustable PWM signal, allowing flexibility in designing the LCD screen's brightness. In addition, it has a micro SD slot, allowing for users to store photos.

WaveShare 3.2" Resistive Touch Screen: WaveShare also offers a 3.2" Resistive Touch LCD screen. It has a resolution of 320x240 and uses a ILI9341 driver [30] , which supports 262K colors instead of the 16.7 million offered by the NT35510 in the 4" model, providing sufficient color depth for simpler graphic needs. The ILI9341 also has a more basic configuration of tearing effect control compared to the NT35510 as a result of its lower resolution [31]. However, like the NT35510 driver, it also supports on-chip RAM and gamma correction, both

allowing for a smooth graphical display. Like the 4'', The WaveShare 3.2'' LCD screen also uses the XPT2046 touch screen control, using the SPI communication protocol as mentioned earlier. Lastly, this LCD display also operates at 3.3V, allowing for easy interfacing with the ESP32.

3.3.6 Final Selection and Justification

After further consideration, it has been concluded that the WaveShare 4" Resistive Touch Screen LCD was the best option for BREATHE. A large factor that justified this LCD screen over its counterparts was its form factor, as it provided a sufficient balance between display area and compactness, allowing information to be clearly displayed without overwhelming the physical design of the LCD and sensor system. Moreover, the screen provided the largest resolution of all the LCD screens considered besides the DFR0669 3.5", allowing for vibrant visuals that will contribute to a user-friendly interface. In comparison, the smaller 2.8" capacitive and 3.2" resistive options offered lower resolutions (240x320 and 320x240, respectively) and fewer colors, making them less ideal for detailed data visualization. Furthermore, the XPT2046 resistive touch controller allows for precise, pressure-sensitive input. Conclusively, the WaveShare 4" LCD was found to be the most substantial and versatile display choice for BREATHE.

Feature	DFR0669 3.5" Touch Screen	2.8" TFT Touch Shield	WaveShare 4" LCD Screen	WaveShare 3.2" LCD Screen
Touch Type	Capacitive	Capacitive	Resistive	Resistive
Resolution	480x320	240x320	480x320	320x240
Color Depth	262k Colors	262k Colors	262k Colors	262k Colors
Communication Protocol	SPI	SPI	SPI	SPI
Operating Voltage	3.3V to 5V	3.3V	5V	3.3V
Display Driver IC	ILI9488	ILI9341	ILI9486	ILI9341

Table 11: Touchscreen LCD Comparison

3.4 Buzzer Selection

3.4.1 Introduction

As mentioned before, BREATHE will require a buzzer to alert the user when dangerous levels of CO2 or temperature are detected. When the temperature or CO2 sensors exceed a predefined safety threshold, the MCU will immediately signal the buzzer to emit an audible alarm, alerting the user of any potential danger. Once CO2 or temperature levels stabilize, the buzzer will turn off, allowing the system to resume normal operations.

3.4.2 Selection Criteria

When selecting a buzzer it is important to select one that will be audible enough for the user to hear. It must produce a sound loud enough to be heard clearly in various indoor settings, even in cases where there may be background noise such as HVAC systems or household activities. Moreover, the selected buzzer should be compact enough to fit within the system, without adding unnecessary bulk to the design.

3.4.3 Types of buzzers

Buzzers are essential components when producing an audible alert is necessary. It’s applied in many systems such as alarm systems, appliances, and doorbells. While there are numerous models of buzzers available, the range of buzzer types are quite limited, leading to fewer considerations being made when selecting an actual type of buzzer. Nonetheless, it's important to consider the advantages and disadvantages of these buzzers to determine the most viable option.

Piezoelectric Buzzers: One of the most common variations of buzzers are piezoelectric buzzers, which use the piezoelectric effect to generate sound. These buzzers are known for their high efficiency and low power consumption, making them ideal for battery-operated devices like the BREATHE system. Piezoelectric buzzers exhibit numerous advantages, such as a long lifespan that can reach upwards of 10,000 hours [32]. Additionally, they produce zero interference since they do not generate RF noise, which ensures that they won't disrupt other circuits. However, a notable downside is that they can be susceptible to hazardous temperatures or changes in humidity [33]. Extreme temperatures or high humidity levels can affect the piezoelectric material’s properties, which can lead to inefficiencies.

Magnetic Buzzers: Along with piezo buzzers, magnetic buzzers are the other most common type of buzzer [34]. Magnetic buzzers function by passing an electrical current through a coil of wire to create a magnetic field. This current attracts a flexible ferromagnetic disk, generating sound as the disk oscillates similarly to a speaker cone. As current fluctuates, the disk moves back and forth, producing audible alerts. These buzzers are current-driven devices that require a voltage source, with the current flow determined by the applied voltage and the coil's impedance. Magnetic Buzzers offer numerous benefits similar to piezoelectric buzzers, such as a long service life, compactness, and cost effectiveness [35]. However, they are less energy efficient than piezoelectric buzzers, and can consume upwards of 100mA or even more. Likewise, due to the moving parts such as the vibrating diaphragm in magnetic buzzers, they are generally more prone to wear and tear than their piezoelectric buzzers counterparts. This translates to a shorter operational lifespan [36].

	Piezoelectric	Magnetic
Power Efficiency	Higher	Lower
Vulnerable to damage from environment (Extreme Temperature and Humidity)	Possible	Possible
Operational Lifespan	Higher	Lower due to potential wear and tear

Table 12: Buzzer Type Comparisons

3.4.4 Buzzer Type Selection

Due to the limited kinds of buzzers, both magnetic buzzers and piezoelectric buzzers will be taken into consideration. As discussed, both provide valuable advantages as well as various disadvantages in relation to BREATHE's specific requirements. Given the project's focus on energy efficiency and long-term reliability, piezoelectric buzzers may be a more viable option. However, if a higher sound output is needed, then magnetic buzzers could be preferred, despite their higher current consumption. Ultimately, the decision will be concluded based on balancing the tradeoffs between both buzzers in order to ensure both optimal system performance and adherence to BREATHE's constraints.

3.4.5 Buzzer Options

PS1240: The PS1240 is a buzzer developed by [37] that has a compact form factor, measuring at just 12.2x6.5mm [37]. Moreover, it has a sound pressure level of 70dB and a frequency of 4kHz [37]. Although this level is outperformed by other magnetic buzzers, it is still an acceptable level that is adequate enough in indoor environments. Its wide operating voltage range of 3V to 30V [37] also allows for an easy integration with the ESP32, as it operates at 3.3V. Another substantial advantage is that it is very affordable, typically priced around $0.50 to $0.60 [32], which aligns very well with BREATHE's budget constraints. However, besides its lower sound output to magnetic buzzers, possible disadvantages include its susceptibility to environmental factors such as high humidity or temperature levels, which can impact both the sound quality and operationality of the buzzer. Despite this, the PS1240 serves as a practical choice for BREATHE for its acceptable sound output, voltage range, and affordability.

CMI 1295: Another piezoelectric buzzer that will be considered is the CMI 1295. It has a similar form factor, measuring at a dimension of 12 x 9.5 mm [38]. Likewise, its sound level outperforms the PS1240, offering a sound pressure level of 85 dBm. This may be more effective in scenarios where a user is not present in the room where the sensor system is located or if loud background noise is a concern. It also is very affordable, pricing at about $1.21. However, the CMI operates at a much higher operating range, being at 8V [38]. This would require an additional DC-DC step down converter, whereas the PS1240 simply needs a connection with the ESP32. Moreover, the CMI 1295 uses much more current than the PS1240, consuming about 30mA at 8V [39]. Lastly, like other piezoelectric buzzers, it is also susceptible to environmental factors like temperature and humidity. That said, the CMI 1295 is a strong alternative for applications where louder output takes precedence over power efficiency and ease of integration.

Jameco 5V Magnetic Buzzer: The Jameco 5V magnetic buzzer is a magnetic buzzer that will be taken into consideration. It has a similar size to both the PS1240 and CMI 1295, sizing at 12x7.5mm [40]. Moreover, it has an operating voltage range from 3 - 7V, which like the PS1240, will facilitate an easy integration with the ESP32. It has a frequency of 3.31KHz ± 300Hz and also offers a very audible output sound, that being at 83dB [40]. The Jameco 5V Buzzer is also very affordable being priced at $0.59, falling into a similar price range with the PS1240 [40]. However, it has the same power consumption as the CMI1295, that being at 30mA [40]. Likewise, since it is a magnetic buzzer, it may potentially have a shorter life span than the piezoelectric buzzers

mentioned, however this problem may not be detrimental as the buzzer will only be used in emergency situations. Conclusively, the Jameco 5V magnetic buzzer offers advantages that both the CMI1295 and PS1240 have such as an ease in integration as well as audible output sound, however it falls short in power consumption.

3.4.6 Final Selection and Justification

We decided that the PS1240 would be the most suitable option for BREATHE for a number of reasons. Although all three buzzers share traits such as affordability and compact form factor, the PS1240 appeared to have the least drawbacks.

One advantage is that its sound output to current consumption ratio is the best out of the three buzzers. Despite being slightly quieter than the Jameco 5V magnetic buzzer and the CMI 1295, it is still acceptable within the context of the project. Additionally, it draws the least amount of current, further facilitating a power-efficient system. Lastly, its voltage range allows for an easy implementation with the ESP32 and other peripherals on the control unit with only one voltage rail.

	PS1240	CMI 1295	Jameco 5V Magnetic Buzzer
Dimensions	12.2 x 6.5mm	12 x 9.5 mm	12 x 7.5 mm
Operating Voltage	3V	8V	5V
Current Consumption	~5mA	30mA	30mA
Sound Pressure Level	70 dB	85 dB	83 dB
Frequency	4 kHz	3.6 kHz	3.31KHz ± 300Hz

Table 13: Buzzer Comparison

3.5 Motor Selection

3.5.1 Motor Type Comparisons

	AC Motors	Brushless DC Motor	Brushed DC Motor	Servo Motor	Stepper Motor
Torque Output	High	Medium	Low	High	Low
Speed Output	Very High	High	High	Moderate	High
Power Consumption	Very high	Moderate - High	Moderate - High	Moderate - High	Moderate - High
Heat Generation	High	Low	High	Low	Low
Energy Efficiency	Good	Good	Bad	Good	Good
Built-in Motor Controller	No	No	No	Yes	No

Table 14: Motor Type Comparisons

When selecting a motor it's very important to take into account the specifications for the project. In order for BREATHE to be able to open or close its louvers while also maintaining an efficient, persisting battery life, it's important for the selected motor to be capable of supplying the necessary amount of torque while also fitting size requirements and utilizing an average power consumption that is as low as possible. It is possible to utilize mechanical designs in conjunction with our electrical parts to improve characteristics of the vent, such as speed and size, to help the project align with the specifications.

AC motors generally have a higher speed and torque output at the cost of an exhaustive power consumption [41]. This could be practical if we have time to pursue our stretch goal of implementing a device capable of acting as both a wind turbine and an emergency fan capable of blowing a high amount of air into a room full of CO_2. This is because we would be able to blow a generous amount of clean air into the room without having to worry too much about the power consumption. The benefits of an AC motor would not be necessary for adjusting the louvers, and the power consumption would make the choice cause more harm than good.

Comparing a brushless DC motor with a brushed DC motor gives benefits and drawbacks on both sides, however a brushless DC motor would fit our specifications better. Generally, brushed DC motors are cheap and provide the same functionality at the cost of a weaker lifespan, louder sound. Due to the design of the brushes, these motors generate more heat and require more power due to internal friction [42]. The power consumption contradicts our energy efficiency specification, and so for our project it would be best to avoid these motors. A brushless DC motor, on the other hand, generally costs more but provides a longer lifespan and better energy efficiency, making it a better choice if we decide to use a DC motor [42].

Servo motors offer a high torque and a small size, which fit our specifications. It functions by taking a PWM signal and adjusts its angle accordingly. This would make coding the functionality very straight forward; the basic goal of completely opening and closing the louvers and the stretch goal of opening to a desired angle would have minimal changes. The project will rely on the servo's high torque to move the louvers, and will also require a mechanical design that is capable of opening and closing the louvers through the servo's range of motion. There are also existing 360 degree servo motors which can provide continuous movement and adjust speed according to the PWM signal [43]. These motors tend to have a slower rotational speed but higher torque, which can be beneficial for opening and closing the vent's louvers.

Stepper motors utilize a magnetic force that can be manually controlled against multiple coils to create motion. These motors can output a different range of torque and speed, each having an inverse effect on the other. In other words, if we wanted to achieve a high torque then we would be sacrificing speed, and vice versa [44]. BREATHE's specifications show that a high torque is more important than speed, however we will still need to be fast enough to open and close the vent within the specified amount of time.

When comparing stepper motors and servo motors, both have their own applications and associated advantages and disadvantages in terms of power consumption. If there is no torque required for holding a specific position then stepper motors are more power efficient. On the other hand, if a lot of torque is required for holding a position then servo motors will consume less power [45]. Ultimately it'll depend on the amount of force generated from the AC's wind. Since most available vents do not have additional mechanisms to ensure louver angles are held, it's likely that the air pressure will be enough to move the vent by itself and so additional support from the motors wouldn't be necessary to hold desired angles. Taking this into account a servo motor will likely be able to

provide us the best amount of torque while reducing the amount of total power consumption.

3.5.2 Servo Motors Comparisons

	SG92	HS-5645MG	MG996R	Futaba 3004
Torque Output (Newton-Meters)	0.245	1.01	0.92	0.314
Voltage Rating	4.8V - 6V	4.8V - 6V	4.8V - 6V	4.8-6V
Current Rating	100mA - 300mA	350mA - 450mA	170mA - 900mA	400mA
Stall Current (at 4.8V)	600mA	2.4A	1.3A	1A
Dimensions	0.9055 x 0.4803 x 1.063 inches	1.59 x 0.77 x 1.48 inches	1.6024 x 0.7756 x 1.689 inches	1.6 x 0.80 x 1.4 inches
Speed (s / 60 degrees)	0.1	0.23	0.2	0.23
Weight	9g	60g	55g	36.85g
Price	~$2-3	$45.99	~$4.75	$18

Table 15: Comparison between servo motors

There are two types of servo motors: continuous movement and regular. Continuous movement servos function just like DC motors, but intake a PWM signal and adjust its speed accordingly. Regular servos tend to only open around 180 degrees. They also intake a PWM signal and adjust their angle according to the value [46]. It is also possible to make edits to a normal servo motor to enable 360 degree rotation [47]. Most vent designs can fully open and

close through 180 degree rotation, so the servo's limited rotation would not be a problem. The following paragraphs will talk about a few different options for potential servo motors.

The SG92 is a small micro servo capable of 180 degree rotation. It only weighs 9g and the dimensions are 0.9055 x 0.4803 x 1.063 inches, making it friendly with the size constraints of many different mechanical designs. It rotates at a speed of 60 degrees every .1 seconds at an operating voltage range of 4.8V, making it friendly with the timing requirement of BREATHE. Although there isn't an official current rating on the datasheet, some sources indicate through experimentation that it consumes about 100-300mA on average and can use up to 600mA depending on the loading force. The main problem with this servo is that it can only supply a force of 0.245 Nm [48]. This has a chance of not being enough torque to open and close the vent. The motor will require experimentation with different vents and their associated friction, but it would provide a very flexible solution to mechanical constraints.

Another potential servo motor is the HS-5645MG. This motor is more on the expensive end, costing about $46. This motor intakes a minimum of 4.8V and can provide torque of up to 1.01 Nm, which can help if our vent has a very high amount of friction. The motor uses on average 450mA, and can use up to 2.4A depending on the amount of force required. The motor can also move at a speed of 60 degrees every 0.23 seconds, making it well within our time constraints. The physical dimensions are 1.59 x 0.77 x 1.48 inches, making it a bit larger than the SG92. This means that it may not be as flexible for positioning and may require some mechanical tricks to translate the servo's force towards moving the vent's louvers. Aside from this, it weighs 60 grams which is heavier than the SG92 [49]. There is a chance that the weight may affect the vent's balance, but if the vent is bolted to a wall or ceiling then this would likely not cause a problem.

Another potential servo motor is the MG996R. This motor weighs 55 grams and has dimensions of about 1.6024 x 0.7756 x 1.689 inches, making it slightly bigger than the other two servo motors. This unit has an operating voltage between 4.8V and 6V, and can produce about 0.92Nm. The official datasheet mentions that it consumes anywhere from 500mA to 900mA, but this is off of 6V. Other sources indicate through experimentation that it typically consumes about 170mA, and has a stall current of about 1.3A when operated at 4.8V [50]. We initially believed that although if the motor stalled for a little bit of time it wouldn't affect the system, however we found that this stall current affected the voltage regulators due to its input limits. Although the MG996R initially seemed to be the most reliable option, we found that after testing the stall currents would affect

the voltage regulators. In the end we decided to switch to a Futaba s3004 servo motor due to its capabilities to open and close the vent, as well as the smaller stall current allowing for flawless integration with the power supply PCB board.

Servo motors don't necessarily need a motor controller since they're typically already embedded within. That being said, these motors can still intake current even when not operating. Although the power consumption is minimum, it can still affect our overall battery life if gone unnoticed. This problem can be solved if we utilize a switch controlled by the MCU that can enable and disable the connection between the servo motor and the power supply. That way the motor will not be pulling current from the battery when it is not in use, saving an accumulated amount of power overtime.

3.6 Mechanical Systems

Due to the nature of this project, we will need to find ways to transfer the motor's rotational motion into the louver's rotating rod. The main issues involve size constraints and potential torque required to overcome the friction of the vent. If we put the motor on the side and there is not enough space to fit the vent into the wall or ceiling, then the product will not work as intended and we will therefore fail our specifications. There are a few systems that can be implemented to translate the motor's force into the desired location, while also potentially increasing the torque at the cost of speed if necessary.

3.6.1 Rack and Pinion

Rack and Pinion gears allow for a rotational gear to push a rod forward, essentially converting rotational motion into linear motion. This is commonly referred to as a linear actuator, which uses a motor such as a stepper motor and allows for the rod to deliver forward force. Looking at many vent designs, users are able to move a slider up or down to open / close the vent. This translates to rotating the louver rod, taking advantage of the prebuilt design to increase the input torque [55]. This way we will be able to save a great amount of battery life due to less overall power consumed overtime. This will be dependent on the chosen vent's frame, as it will need to be able to support a linear movement with a leveraged handle.

3.6.2 Chained Gears

Chains can also be utilized to connect two gears together. This implements the same logic as connecting two gears together, but without having to have contact with positioning directly next to each other. This also ensures that the

two gears move in the same direction. If one gear has more teeth than the other, then this will implement a difference in gear ratio and therefore change the values of speed and torque. These two variables have an indirect relationship with each other; if one increases then the other decreases by the same factor and vice versa. If gear 2 has double the amount of teeth than gear 1, then we can say there is a gear ratio of 1:2. This means that the speed of gear 2 is half of gear 1, but gear 1 has doubled the amount of torque of gear 1. This design can be helpful if we need to implement more torque for the louvers and if we need to position the motor a distance away from the louver's rotational rod [55].

3.6.3 Bevel Gears

Bevel gears act similarly to other gears, but they also allow for rotational motion to be transferred in another axis direction. They have similar torque and speed conversions depending on the number of teeth. The gears allow for the redirection of force by being angled at 90 degrees. If we need to position the motor where the shaft cannot be aligned in the same plane as the louver's rotational rod, then this system will be a great way to allow flexibility for positioning the motor in other parts of the vent [55].

3.6.4 Worm Gears

Worm gears allow for motors to translate high amounts of speed into higher amounts of torque. It works by rotating a cylinder with thread against a bigger gear with a large amount of teeth. It's common for these devices to multiply an input torque by 30 times, and in some cases even greater than 300 times. The downside is that increasing the torque by a multiplied constant causes the speed to be divided by that same constant. In other words, if we multiply an input torque by 30 times then it will require 30 rotations for a single rotation. For some servo motors with lower RPM this can cause movement of the vent's louvers to take a really long time, which can cause the product to fail the timing specifications. For other motors that have a very high RPM but a very low torque rating, this can be a perfect implementation [55].

3.6.5 Mechanical Design Selection

Ultimately we decided to proceed with a rack and pinion mechanism to support the effort of opening and closing the vent. It's a very straight forward design that allows us to take advantage of the vent's original design and extra torque from its lever, allowing the motor to utilize less applied torque and therefore saving power. Since the mechanism utilizes a gear, it allows for the use of a servo

motor to fully open and close the vent through a 180 degree rotation that translates to linear movement to cover the entire lever.

3.7 Power selection

The power section of both the vent interaction subsystem and the control unit is extremely important to ensure all of our peripherals receive sufficient power that is within their respective manufacturer specifications. For the control unit, since it will likely be in close proximity to wall outlets, some type of AC-DC power adapter unit would be sufficient for powering it. Part of what led to this decision was the LCD's power consumption, which would likely drain any battery relatively quickly considering it needs to be on most of the time (it would be a little inconvenient if the user had to wake up the LCD each time just to see the temperature). To reduce unneeded complexity this was the best option for the control unit. For the vent interaction subsystem, however, due to its location it will be impossible to power it using anything but a battery. This makes it especially important to design a power system that will be efficient enough to ensure the battery lasts at least 6 months as previously specified.

Both systems will need some type of voltage regulation present to ensure all peripherals receive proper voltages. For the control unit, most of the peripherals including the MCU will need a 3.3V input. The LCD screen, however, requires a 5V rail for power. As for the vent interaction subsystem, the same issue arises as the motor requires 5V whereas the NRF52832 has a max input voltage of 3.6V and a minimum input voltage of 1.7V. This means there will need to be two separate voltage rails in the vent interaction subsystem. It is important to note that there are important things to consider besides just efficiency and current draw. Cost is also a factor to consider. Overall, whatever method is used to provide power to all the peripherals in the vent interaction subsystem in particular will need to be balanced between all these considerations.

3.7.1 Battery Types:

The most important decision to make regarding the vent interaction subsystem's power supply design is the type of battery we will use. Since we want the subsystem to be powered for as long as possible without the user having to change the batteries, we need to prioritize battery capacity. This also needs to be balanced with cost and overall size since the vent cover only allows a small amount of space to place components.

One primary concern is that we will need a primary or disposable battery since there would be no ideal way of charging a secondary battery in the vent. Lithium

metal batteries tend to have the highest capacity for primary batteries, but they also tend to cost more. Another advantage is that they are less prone to leakage than common Alkaline batteries. Note that we are referring to these batteries as Lithium-metal to properly differentiate them from secondary Lithium-metal batteries. Lithium-metal batteries often have various different chemical mixtures in them depending on the brand and type so we are using it as a catch-all term to avoid confusion.

Alkaline batteries in contrast have the most limited capacity and are prone to leakage which can easily corrode and damage sensitive components if they are left in for too long. As a result, we likely will not consider this as a reasonable battery type to use. The only benefit to Alkaline as opposed to Lithium-metal batteries is overall cost and availability. Alkaline batteries also can be slightly smaller depending on the brand.

	Alkaline	Lithium-metal
Cost	Cheapest battery type	Balanced Cost
Size	Relatively small size depending on model, comparable overall to most lithium-metal batteries	Relatively small size depending on model, comparable overall to most alkaline batteries
Availability	Easily accessible in any department store	Easily accessible in any department store

Table 17: Comparison between different types of batteries

Overall, Lithium-metal batteries seem to offer the best ratio between capacity and size. The fact that they are also easy to find in stores also makes this option very attractive both for testing and for the user.

The two Lithium-metal batteries that will be considered will be the CR123 3V battery as well as a lithium 9V battery. The CR123 is rated for a capacity of 1500mah and the 9V is rated for 1200mah. These batteries provided the most capacity whilst being easily attainable and moderately sized.

3.7.2 Voltage Regulation

Voltage regulators are commonly used to provide a stable voltage output from a variable voltage source to meet the specific voltage requirements of each component. Both subsystems will need voltage regulators as the voltage range of each of our components isn't simple to match with a battery or wall adapter. This is especially important for our batteries since their discharge voltage can

decrease over time. There are two general types of voltage regulators: Linear and switching regulators.

3.7.2.1 Linear Voltage Regulators

Basic linear voltage regulators consist of a circuit utilizing an op-amp with a closed feedback loop and a bipolar transistor to stabilize the output voltage. This circuit allows quick adjustments when the load current changes. Linear regulators commonly come in IC packages that either have output voltages that are adjustable using a voltage divider, or output voltages that are fixed to that model of regulator.

The main issue with these regulators is efficiency. Basic linear regulators typically have a dropout voltage of around 2.5V, which means the input voltage must be at least 2.5V more than the output voltage to guarantee the output is stable. These regulators have an input power that is directly proportional to the output power, meaning that higher the voltage difference between input and output causes less efficiency in the regulator. These regulators are therefore most efficient when $V_{in} = V_{out}+V_{Dropout}$. More importantly, the extra power from this difference between the input and output is dissipated as heat, so using such a regulator would likely require having a heatsink attached to it. The main upsides to a linear regulator are cost, ease of use, and lack of switching noise which is common in switching regulators. Regulators such as the LM78xx series and others are so ubiquitous and cheap that they are commonly considered jellybean components.

3.7.2.2 Low Dropout Regulators

One solution to the dropout voltage requirement of basic linear regulators is a low dropout voltage regulator. Low dropout regulators allow us to use voltages closer to the output voltage whilst also wasting less power given that our input voltage is very close to the output voltage. These regulators can commonly reach up to 90% efficiency given this condition. The main downside to this is that the higher the input voltage is the less efficient the regulator becomes, which is the same as the behavior found in basic linear regulators. There are many 3-3.6V batteries we could use to power our components, but if the discharge voltage were to drop below $V_{out} + V_{Dropout}$ then our output voltage would not be stable. Trying to use batteries with higher voltages would also reduce the efficiency. Ultimately this option is more useful than basic linear regulators but careful considerations would need to be made regarding battery choice. Once more, heat is also a factor, and this will require careful considerations in regards to IC package type to ensure the temperature of the device is kept under control.

3.7.2.3 Switching Regulators:

Switching regulators utilize an inductor, a diode, and a FET controlled by a switching IC to turn the FET on or off depending on if the desired output voltage has been reached [62]. Due to the nature of the design, switching regulators are capable of high efficiency regardless of input voltage and it can be used in many more conditions such as with an input voltage that is lower than the output voltage (boost configuration). Switching regulators typically have above 90% efficiency and a typical regulator IC likely wouldn't be too much more expensive than a linear regulator.

The main downside to this is that the design requirements are quite strict and finding proper capacitors and inductors that fit the datasheet's requirements would cost more than a comparable linear regulator circuit. There is also added complexity due to the fact that efficiency is also dependent on the output current, and the overall output current can be limited depending on the input voltage and the IC. Thankfully our devices don't consume much current, so this shouldn't be much of an issue. The presence of switching noise is another downside as it requires more careful component choices and design considerations to mitigate the noise. Thankfully modern DC-DC switching ICs are very reliable [62].

	Linear Regulator	Switching Regulator
Cost	Due to the low component count and basic nature of the design, cost is generally low.	Higher component count and more complex and robust ICs generally result in higher cost.
Efficiency	Not Fixed, dependent on difference between V_{in} and V_{out} Can be a maximum 60% for a basic linear regulator and 98% for an LDO. There is no minimum	Typically fixed and only slightly dependent on output current. Tends to be above 90% but can reach a minimum of around 70% at low current draw
Design Complexity	Simplistic	Slightly Complex
Size	Low component count consisting of mostly tiny SMD components. Takes up little space on PCB.	Requires anywhere from 6-15 depending on the type of regulator and design. Utilizes an inductor which can increase PCB footprint drastically.

Table 18: Comparison between different types of voltage regulators

Due to their high efficiency, switching regulators are likely to be a fitting choice for BREATHE's power supply design. Despite the complexity and increased footprint, the fact that switching regulators operate at a mostly fixed efficiency is helpful for Lithium-metal batteries which can drop in discharge voltage over time. This also allows less concern for thermal concerns as less power is able to be dissipated as heat. This is especially critical with the control unit, where dropping down from 12V to 5V may draw significant amounts of heat due to the power loss.

3.7.3 Types of switching regulators

There are three main types of switching regulators that we can incorporate into the design of our power system. The type of regulator we use is also going to partly decide what battery we use and is also integral to optimizing the overall battery life of the vent interaction subsystem.

For the sake of simplicity, the battery life calculations for this section will not account for low power modes and other general discontinuities in current draw, though low-power mode features will be mentioned. The calculations will be entirely based off of an 400mA continuous current load with a 5V output rail. This is because the motor will require a voltage of at least 5V and has a maximum current load of 400mA. Obviously, the motor's current load will be discontinuous, but this scenario is merely to offer a simple and fair comparison. The LCD screen's current draw on the control unit will not be used in this comparison since it is not battery powered. This will also mostly ignore common lithium battery phenomena that occur with continuous current loads such as reduced capacity or reduced discharge voltage. This is to once again provide an easily understandable comparison as a baseline.

3.7.3.1 Boost Converter (TPS 61023)

A boost converter, sometimes referred to as a step-up converter, is a switching regulator that allows a voltage source to output more voltage than it can typically provide.

This means that if we used a common battery with a 3V supply we would still be able to supply 3.3V or even 5V to the components that need it. This would also be safe in case the battery's discharge voltage were to fall below 3V nominal. This would be beneficial for battery sources with high capacity but low output voltage.

There is, however, a major downside to this design due to the nature of how a boost converter is able to output a higher voltage. Since in a switching regulator the power must be conserved (i.e. $P_{in} = P_{out}/\text{efficiency}$), the boost converter will

draw more current from the source compared to the output [63]. For example, if we were to use a battery that supplies nominal 3V and we needed an output of 5V at 400mA, then

$$P_{out} = (5V)(0.4A) = 2W$$

According to the TPS61023 datasheet, with these input and output characteristics the efficiency of the regulator would be about 94% [64]. So for the input power:

$$P_{in} = 2W/0.94 = 2.128W$$

Our source current draw would therefore be:

$$I_{in} = 2.128W/3V = \sim 0.7093A$$

This isn't great for overall battery life regardless of the capacity of the battery. Batteries can be optionally stacked in parallel to increase capacity, but even then the current draw of a boost converter is generally unsuitable for a battery source if longevity is a main priority. Boost converters are also typically less efficient compared to buck or buck-boost regulators, but they all tend to be around 90-95% efficiency regardless.

3.7.3.2 Buck converter (AP63205-WU)

A buck converter, sometimes referred to as a step-down converter, is a switching regulator commonly used to output a lower voltage from a higher voltage source. This allows us to potentially use a 9V battery or several other battery types in series.

One main advantage of using a buck converter, especially in comparison to boost converters, is how the converter handles input and output currents. Just like the boost converter, power must be conserved from input to output [65]. For a voltage source such as a 9V battery, if we were to once again calculate for a 5V output with 400mA current draw:

$$P_{out} = (5V)(0.4A) = 2W$$

According to the AP63205WUdatasheet, the efficiency at 400mA is about 94% [66]. So

$$P_{in} = 2W/0.94 = 2.128W$$

$$I_{in} = 4.255W/9V = 236.4mA\sim$$

Essentially, in this configuration a buck converter would give us a 400mA output while only drawing nearly half that at the source. This would be highly beneficial for the vent interaction subsystem where battery life is of utmost importance. Using a 9V battery this would result in a very reasonable battery life despite the high current draw of the motor. One thing to note, however, is that this reduction in source current is dependent on the difference in voltage between the battery and the desired V_{out}. Another important factor to consider is a battery's discharge voltage over time. Batteries will not have a consistent output voltage across their entire lifespan, instead they will slowly discharge less voltage until they reach their "cutoff voltage" which is when they will stop discharging entirely. This cutoff voltage will vary but tends to be within the same range depending on the battery's nominal output voltage. For the 9V battery example that was used in this section, the cutoff voltage tends to be around 5.5-6V. This means the buck-converter will work across the battery's entire lifespan since the regulator only requires V_{in} to be equal to $V_{out}+I_Q$(quiescent current). For batteries that have a cutoff below that requirement, they will not be able to function across their entire lifespan. This would in turn decrease overall battery life.

3.7.3.3 Buck-Boost converter (TPS63070)

The last major topology we considered for the power section is the buck-boost topology. This is a combination of the two previous switching regulator designs that allows for voltages both above and below the desired output voltage. This is helpful for the exact scenario stated in the prior section: a lithium battery that will have a discharge voltage lower than the output voltage after a certain period of time. Instead of that battery becoming unusable before its rated cutoff, the buck-boost topology would allow the battery to be usable in boost mode until it eventually dies. Of course, this would result in increased source current draw which isn't optimal for high battery life. However, this would still be ideal for use with said batteries as opposed to a buck converter. This begs the question why not use a buck-boost topology as opposed to a regular buck-converter? Buck-boost topologies are often paired with Lithium-ion batteries and have many built-in protections specific to them [67]. Unfortunately, buck-boost converters tend to cost more and require more components to work, which will make the PCB larger as a result. This makes a buck-boost converter practically useless for any battery that can't fall below the desired output voltage. Moreover, whilst buck-boost topologies are ideal for rechargeable battery types such as LiPo and Li-ion batteries, for disposable batteries they have little use when compared to a regular buck-converter. As a note, the efficiency of this buck-boost converter is practically the same as the other options [67].

3.7.3.3.1 A Quick Note on SEPIC converters:

SEPIC converters are a variation on buck-boost converters that can reduce board size by reducing the overall parts count. SEPIC converters work by chaining a boost converter to an inverted buck-boost converter. This has a few other benefits such as a proper 0V shut down mode and a non-inverted output (buck-boost converters have inverted outputs that need to be accounted for in the design) [68]. They are also less prone to switching noise.

The main issue with SEPIC converters is poor efficiency in comparison to buck-boost converters and a higher IC cost. SEPIC converters therefore will not be considered in this design.

	TPS 61023 Boost converter	AP63205WU Buck converter	TPS55289 Buck-Boost converter
Efficiency (5V 400mA output)	94%	94%	94%
Source Current (5V 400mA output)	709.3mA	236.4mA	237mA
Quiescent Current	20uA	22uA	760uA
IC Cost	$0.78	$1.38	$6.60

Table 19: Comparison of Switching Regulators

Overall, the buck converter with a 9V battery seems to be the most reasonable option for our power section due to its high battery life, low cost, and moderate size. Both size and battery life are the main concerns for BREATHE considering the components have to fit within a narrow space and the batteries need to last at least 6 months before needing replacement.

3.7.4 Lithium-metal Battery Choice:

As previously stated, the method of regulation will also directly tie into what battery we must use in the vent interaction subsystem. Due to the fact that both a buck and buck-boost topology are very similar and reasonably useful depending specifically on what battery is used, this section will cover battery choices that match either scenario in which one is needed or more useful as opposed to the other. Boost-converters have been understandably ruled out as they would draw far too much extra current. This section will use battery life calculations that are extremely simplified and once again based off of a

hypothetical continuous 400mA 5V output for the motor. There are two main considerations for Lithium-metal battery types with high capacity and enough voltage to be used in conjunction with a buck regulator: A 9V battery and a CR123 battery.

3.7.4.1 9V Battery in Buck Topology:

A typical 9V Lithium-metal battery has a capacity of around 1200mAh. This is relatively high considering both the size and availability of these particular batteries. Since the source current using a 9V battery for BREATHE's chosen buck-converter has already been calculated as 236.4mA, this value can be used to calculate an estimated battery life. This battery life would be around 5 hours and 4 minutes. Hypothetically, let's say it takes the motor 5 seconds to open or close the vent, and it only will be powered for those 5 seconds. In this case the motor could move 3648 times before the battery would run out (ignoring quiescent current and the MCU's power draw). Overall, this option is extremely efficient and very suitable for the vent interaction subsystem. Notably with a buck-converter this battery would be able to function across its entire lifespan.

3.7.4.2 CR123 Battery in Buck-Boost Topology

Another option is using two CR123 batteries in series to get a nominal 6V with 1500mah capacity. In terms of size, this option would take up more space since two batteries in series would be slightly longer than one 9V battery, though they would have similar depth. This is also coupled with the fact that A style batteries require an enclosure with spring contacts to properly use them, potentially taking up more space. As for battery life, using our previous example

$$P_{in} = 2W/0.96 = 2.083W$$

$$I_{in} = 4.17W/6V = 347.2mA\sim$$

Using this configuration, the overall battery life would be around 2 hours and 9 mins. This performance is significantly worse than that of the 9V battery which makes this option less sensible for our purposes, especially since it takes up more space. Another factor that hasn't been touched upon is discharge voltage. These batteries have a cutoff at around 2V, so there is a possibility it is still operational but discharging under 3V [69]. Hypothetically, the battery could be somewhat old and discharging at 2.4V, in this case it would not be able to supply a 5V output. This is in contrast to the 9V which has a cutoff discharge voltage of around 6V, meaning it could still provide power across the entire range of its lifespan [70]. This means using a buck-boost topology would be necessary to extend the battery life to accommodate the whole range of the battery.

To add to the previous calculation, using the datasheet we can find the discharge voltage corresponding to the Ah performance. From the Duracell datasheet we find that from 1.4-1.5Ah the discharge voltage is below 2.5 per battery, meaning that the input voltage will be below the output voltage and the regulator will be in boost mode [69]. Therefore, without a buck-boost converter the actual battery life is around 4 hours and 19 minutes. However, using a buck-boost converter we can get the full capacity of the battery

Calculating for the total battery life with lowest discharge voltage before cutoff (2.1V)

$$I_{in} = 2W/4.2V = 476.2mA\sim$$

This allows an extra 12 minutes bringing us a total of 2 Hours and 31 minutes of battery life as opposed to just 4 hours with a typical buck converter when accounting for the drop in discharge voltage. This isn't necessarily bad, but performance is still much worse than that of the 9V battery with a buck converter.

	9V Battery	2 CR123 Batteries in Series
Source Current Draw(@400mA 5V output)	236.4mA	476.2mA
Battery Capacity	1200mAh	1500mAh
Battery Life (@400mA 5V continuous output)	5 hours and 4 minutes	4 Hours and 31 minutes
Cost	$17.42	$12.47

Table 20: Comparison of lithium ion battery types

3.7.5 Providing Split Rails:

As previously outlined, the power section will require two different voltage rails to properly supply power to both the MCU and the motor in the vent interaction subsystem. Thankfully, the MCU can take anywhere from 1.7-3.6V whilst the motor needs anywhere from 4.8-7.2V to function properly. In terms of current consumption, the MCU seems to draw at maximum 130mA, it is important to note it will likely never consume anywhere near this much current, but using this as a baseline means the MCU will be able to draw as much as it will ever need. Conversely, the motor will draw a maximum current of 400mA. Whatever

solution chosen to compensate both output rails will also need to compensate for these maximum ratings just in case. It was already determined that a buck converter is the most reasonable solution, at least for the first stage of voltage regulation, but for providing our second voltage rail there are three unique solutions with their own respective tradeoffs. At this stage we can determine the specific ICs we will use for the entire power section in the vent interaction subsystem.

3.7.5.1 Chaining The Buck Converter into an LDO:

One option is to use an LDO on the output of the main voltage rail on our buck converter to step-down to the required voltage needed for the MCU. Previously, it was determined that linear voltage regulators would not be suitable for the vent interaction subsystem since the efficiency is dependent on the voltage difference between input and output and we could not guarantee that it would be above even 70% when using the batteries that are most ideal for this application. Buck converters also can provide a higher output current whilst sourcing less from the battery depending on the voltage difference. Conversely, an LDO would only source exactly what the output is drawing in addition to the device's quiescent current. For our main voltage rail, which would step down 9V to 5V (ignoring discharge voltage loss), a buck converter is easily the best solution. However, when stepping 5V down to 3.3V the difference isn't nearly as large, especially when accounting for the fact that the MCU is usually drawing only a dozen or so milliamps at any given time. Whilst power consumption and efficiency are quite important downsides for this option, the tradeoffs are a lower overall footprint and lower cost.

A potential LDO to consider in this case would be the XC6206. This LDO has a quiescent current of about 1uA and only requires 2 capacitors in addition to the IC itself [71]. It features overcurrent protection as well as overtemperature protection as well. Stepping down 5V to 3.3V, the efficiency of this regulator would be about 66%. Using the maximum current draw of the MCU at 200mA, we can calculate the amount of power dissipated as heat as well as the proper package to use to ensure thermal reliability. We can also roughly calculate battery life using our previously established metrics.

$$P_D = (5)(.200) - (3.3)(.200) = 0.34W$$

According to the datasheet [71], a SOT-89 would easily be able to handle this extra power dissipation if the MCU were to ever hit its maximum current draw. The battery life of this solution (ignoring the motor) would be around 60 hours if we estimate the MCU draws 20mA consistently.

3.7.5.2 Using Two Buck Converters:

This option is the obvious conclusion for providing two voltage rails. The main downside of this being the footprint would be double that of a single buck converter circuit, which may potentially make it more difficult to fit the vent-subsytem's PCB in the actual vent cover. Thankfully, many of the other components in the vent interaction subsystem have very small footprints, making it feasible to fit this scheme inside the vent. One large advantage of this would be cost, dual-output converters tend to cost a bit more compared to a single buck-converter. Whilst it may cost more than a buck-converter chained into an LDO, this scheme would be much more efficient and as a result increase battery life. Moreover, this scheme would be very easy to test in case one regulator were to go bad.

It is important to note the best scheme for using two buck converters would be combining them in parallel as opposed to series. In series, there would be more current draw from the battery on the 3.3V rail since it would be sourcing from 5V instead of 9V. Moreover, switching noise may be more of an issue when using a buck converter as a source for another buck converter. Reducing noise may require a more complex design and a more precise layout. Due to this, combining them in parallel would be preferred.

3.7.5.3 Using a Dual-Output Buck Converter:

Another potential option is utilizing a PMIC that has two buck converters integrated into one package. This would reduce the space concerns with having to use two ICs with their own relatively large components counts. Typical dual-output buck converters will have component counts comparable to a single buck-converter. This would substantially reduce the overall footprint. The main downside to this is cost, whilst an LDO and single buck-converter combination would be very inexpensive, a typical dual-output buck converter IC is very costly.

A great contender for this option would be the LTC-3622. This dual-output buck converter was one of the only ones on the market that had a somewhat lower cost, fulfilled our current output requirements, and maintained good efficiency across a wide range of output current loads. This IC also comes in leaded packages which will inevitably make testing simpler before we integrate the design into a full-fledged PCB. Curiously, the LTC-3622 also provides two optional modes of operation for each output, one that would reduce ripple voltage at the cost of some efficiency at lower current loads and another which would increase efficiency at the cost of ripple voltage. This would be extremely

useful for debugging issues with either the MCU or motor which may be due to ripple.

3.7.5.4 MCU Input Voltage Options:

Since the ESP32 has a wide input voltage range between 1.8V-3.6V and there aren't any extra peripherals which would require 3.3V, there are a few considerations to make when determining what voltage output should be sent to the MCU. Thankfully, the LTC-3622 datasheet specifies the efficiency of the regulator at different output voltages. Using this, we can compare the source current draw depending on whether we use a 2.5V output or a 3.3V output (note: these are the only two shown in the datasheet, 1.8V is also shown but efficiency plateaus far too low at that voltage) [72].

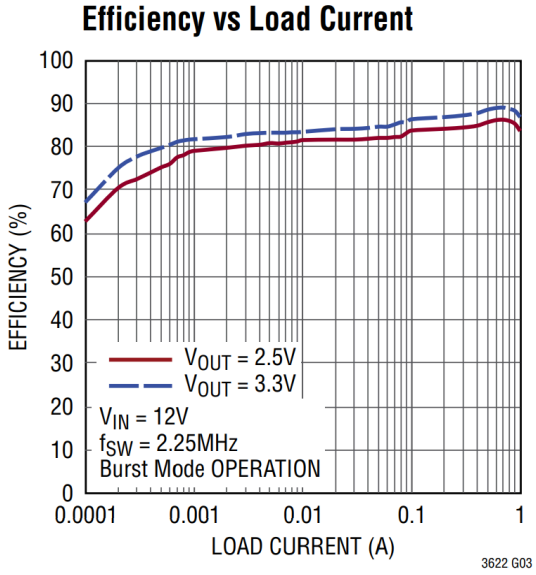


Figure 4: Efficiency vs Load Current of LTC-3622 at 2.5V and 3.3V output [72]

(This image is used with permission from Analog Devices, Inc. ("ADI"), © 2024.)

For the sake of simplicity we can calculate the source current draw for 20mA. At 3.3V:

$$P = (3.3 *.02)/(.8) = 0.0825W$$

$$I_{in} = 0.0825/9 = 9.16mA$$

At 2.5V:

$$P = (2.5 *.02)/(.83) = 0.0602W$$

$$I_{in} = 0.0602/9 = 6.68mA$$

As shown above, a 2.5V output would be ideal for the MCU output as it draws a little less current from the source due to the voltage difference. Heat issues due to efficiency would be negligible since the two output voltages are relatively close in efficiency anyways.

It is also important to note efficiency differences between the AP63205WU and the LTC3622. The AP63205WU has around 94% efficiency at 400mA (near max current draw of the motor) and the LTC3622 has around 89% efficiency at 400mA. This difference is nearly equal across most of the graph's range, except for extremely low current loads where the LTC3622 is actually more efficient.

	AP63205WU*Buck Converter(5V) + MCP1700 LDO(3.3V)*	*LTC-3622 Dual-Output Buck Converter(5V and 2.5V)*	*Two* AP63205WU*Buck Converters in Parallel(5V and 3.3V)*
Efficiency (5V 400mA output)	94%	89%	94%
Efficiency (3.3V/2.5V 20mA output)	66%	83%	91%
Source Current Draw (@3.3V and 2.5V with 20mA output)	20mA	6.7mA	8.05mA
Source Current Draw (@5V 400mA output)	236.4mA	249.7mA	236.4mA
Total Source Current Draw (neglecting quiescent current)	256.4mA	489.8mA	244.45mA
IC Cost	$3.22	$9.71	$5.44
Footprint	Marginally larger footprint than	Smallest footprint and lowest	Largest footprint, nearly double the

	dual-output regulator due to having to use two ICs	component count	area of dual-output IC.

Table 21: Comparison between power supply options

Considering cost, source current draw, and footprint it is evident that using two buck converters in parallel would be the most effective solution for BREATHE. Whilst the LTC-3622 is a reasonable option, the extra source current draw and high cost makes it a less than ideal option. Moreover, it would be much easier to test two separate buck converters in case issues arise.

3.8 Communication Protocols

3.8.1 Serial Communications

There are several serial communication protocols that serve to ensure reliable data transfer, efficient power usage, and real time functionality. Each of which brings their own unique advantages best suited for various tasks. Below is a list of the most common serial communication protocols used.

UART: Universal Asynchronous Receiver-Transmitter (UART) is a widely used serial communication protocol that enables point-to-point data exchange between two devices. Unlike synchronous protocols such as SPI or I2C, UART is asynchronous, meaning it doesn't require a shared clock between the devices. Instead, data transfer relies on setting a common baud rate on both ends, which synchronizes data transmission and reception. Each UART data packet includes start and stop bits that signal the beginning and end of each byte, ensuring the receiving device correctly interprets the data without a shared clock signal [73]

UART communication uses two primary lines, TX (transmit) and RX (receive), enabling simple, bidirectional data flow. Data is transferred in a serial format, one bit at a time, with error-checking options like parity bits that help detect errors during transmission. UART's simplicity and widespread support make it ideal for low-speed data transfer applications, such as communicating between microcontrollers or interfacing with modules that don't require high data rates or complex handshaking protocols. However, UART's asynchronous nature limits its effectiveness over longer distances, as baud rate mismatches can lead to data errors [73].

In embedded systems like BREATHE, UART is especially useful for debugging and status reporting, where data is sent to a terminal or computer for monitoring. This protocol's low overhead and easy setup make it ideal for applications that don't require high bandwidth or multiple-device connectivity. While limited to single-device connections, UART's simplicity, reliability, and low resource requirements make it a versatile and valuable communication option for point-to-point data exchanges in embedded systems [73].

I2C: The Inter-Integrated Circuit (I2C) protocol, developed by NXP (formerly Philips), is a popular synchronous, multi-master, multi-slave communication protocol commonly used in embedded systems for connecting low-speed peripherals like sensors, memory devices, and ADCs. I2C requires only two lines—SDA (Serial Data) and SCL (Serial Clock)—making it efficient in terms of wiring and well-suited for applications where multiple devices share a single communication bus. Each device on an I2C bus has a unique address, allowing the master to communicate with specific devices without requiring extra control lines [74].

I2C supports a range of data rates, from standard mode at 100 kbps to fast mode at 400 kbps, with high-speed mode reaching up to 3.4 Mbps. It operates as a half-duplex protocol, where data can only be transmitted in one direction at a time. The protocol also includes acknowledgment bits, enabling error detection by confirming successful data transfers between devices. This reliable communication mechanism allows I2C to handle various applications with multiple sensors or peripheral devices, despite its slower data rate compared to other protocols like SPI [74]

For BREATHE, I2C is particularly useful for connecting environmental sensors to a microcontroller, allowing multiple sensors to share the same communication lines. This scalability and simplicity are ideal for applications that require multiple devices on a single bus without complex wiring. Although I2C's slower data rate may not be ideal for bandwidth-intensive tasks, it excels in low-power, low-speed data collection, making it a practical and efficient choice for sensor integration in embedded systems [74].

SPI: The Serial Peripheral Interface (SPI) is a high-speed, synchronous communication protocol widely used in embedded systems for connecting microcontrollers with peripherals such as sensors, displays, and memory modules. SPI operates in a full-duplex mode, allowing simultaneous data transmission and reception, which is essential for high-performance applications. It uses a master-slave architecture with four primary lines: MOSI (Master Out Slave In), MISO (Master In Slave Out), SCLK (Serial Clock), and SS

(Slave Select). The master device generates the clock signal (SCLK), enabling precise timing and coordination for data exchange with the slave devices. SPI's streamlined protocol and clock-driven communication allow data rates of several Mbps, providing efficient performance for bandwidth-intensive applications [75, 76] .

One of SPI's significant advantages is its flexibility and simplicity in data exchange, achieved through dedicated communication lines that reduce bus contention. Since SPI does not have an addressing scheme, each additional device requires a unique SS line, which the master uses to select and communicate with specific peripherals. This method offers rapid data access, making SPI ideal for applications requiring real-time data transfer, such as driving graphical displays or reading high-speed ADCs. However, since each device needs a separate SS line, systems with multiple devices can become more complex to manage, especially on PCBs with limited space [75].

SPI's speed and efficiency make it highly suitable for applications requiring frequent and large data transfers. For BREATHE, SPI is especially advantageous for interfacing with components like LCD screens, where high-speed data refresh is critical for smooth visual performance. Despite its wiring limitations, SPI's fast, full-duplex communication capabilities make it one of the most effective protocols for embedded applications requiring substantial data handling and minimal latency [76].

3.8.2 Serial Communication Final Selection and Justification

In the smart vent system, each communication protocol—UART, I2C, and SPI will serve different roles that will ensure reliable operation. UART will primarily be used for debugging and status reporting, allowing the system to send diagnostic information to a connected computer or terminal for monitoring. For instance, UART can be used to test the output of a sensor by s values to a terminal. Its simplicity and low resource requirements make it ideal for this purpose. For instance, UART can be used to test the output of the CO2 sensor by sending its output values to a terminal. Likewise, I2C will facilitate communication between the ESP32 microcontroller and multiple environmental sensors, such as the CO_2 and temperature sensors. With its multi-master, multi-slave capability and shared bus architecture, I2C minimizes wiring complexity while enabling the seamless integration of multiple peripherals. Finally, SPI will be employed for interfacing with the capacitive touchscreen LCD. Its high-speed, full-duplex communication will allow for quick data refresh rates, enabling smooth and responsive visuals for displaying environmental data and user feedback.

Together, these protocols provide a robust foundation for the BREATHE system, optimizing communication efficiency across its various subsystems.

Below is a table comparing each type of wired serial communication.

Feature	UART	I2C	SPI
Communication Type	Asynchronous	Synchronous	Synchronous
Clock Requirement	No (asynchronous operation)	Yes (shared clock line: SCL)	Yes (master provides clock: SCLK)
Number of Lines	2 (TX, RX)	2 (SDA, SCL)	4 (MOSI, MISO, SCLK, SS)
Topology	Point-to-point	Multi-master, multi-slave	Single-master, multi-slave
Full-Duplex Capability	No	No	Yes
Speed/Data Rate	Limited by baud rate, less effective over longer distances	100 kbps (Standard Mode), up to 3.4 Mbps (High-Speed Mode)	High-speed communication, data rates up to several Mbps
Error Checking	Start and stop bits, optional parity bits for error detection	Acknowledgment (ACK) bits confirm successful data transfers	None mentioned, relies on application layer

Table 22: Comparison between wired serial communications

3.8.3 Wireless Communication

Bluetooth 5.0: Bluetooth is a wireless communication protocol designed for short-range, low-power data transfer. The introduction of Bluetooth 5 significantly enhanced the protocol, offering improved data rates, range, and broadcast capacity over previous versions. Bluetooth operates in the 2.4 GHz ISM band and uses adaptive frequency hopping (AFH) to minimize interference from other wireless signals. Bluetooth 5 offers three key modes: a high-speed

mode for fast data transfer (up to 2 Mbps), an extended range mode to support longer-distance connections, and an advertising mode that allows devices to broadcast data to multiple devices simultaneously without a dedicated connection [77].

Bluetooth 5 expanded the functionality of the protocol by increasing the maximum range and enabling more flexible data transmission options. Devices can now transmit data over distances of up to four times that of Bluetooth 4.2, with adjustments available between range and data rate to fit specific application needs. Bluetooth's flexibility makes it suitable for applications like IoT, smart home, and wearable devices where reliable short-range communication is essential. Enhanced broadcast capabilities in Bluetooth 5 also enable devices to communicate with multiple receivers simultaneously, allowing a single device to share information broadly, such as in beacon applications [77].

Bluetooth's low power consumption is another advantage, particularly in Bluetooth Low Energy (BLE) applications where devices are designed to transmit small amounts of data infrequently. This feature is ideal for applications that require minimal power usage, such as sensors, health monitors, and remote controls. For embedded systems like BREATHE, Bluetooth is beneficial for data-sharing applications that require secure, low-power connectivity, especially in devices that need to operate on battery power for extended periods. Bluetooth 5's enhancements in speed, range, and broadcast capabilities make it a highly versatile protocol for modern wireless communication needs [77].

Bluetooth Low Energy: Bluetooth Low Energy (BLE) is a wireless communication protocol designed to support low-power applications in areas such as IoT and wearable devices. Similar to traditional Bluetooth, BLE operates in the 2.4 GHz ISM band, utilizing a frequency-hopping mechanism to reduce interference and ensure reliable communication. However, BLE differs from traditional Bluetooth by optimizing energy efficiency, allowing devices to operate on small batteries for extended periods. This makes BLE suitable for applications that require periodic data transfers, such as sensor readings and device control, rather than continuous data streaming [78].

BLE is structured around the Generic Attribute Profile (GATT), which organizes data into services and characteristics for efficient data management. GATT allows devices to communicate in a structured format that is easy to standardize across various device types, from simple sensors to complex multi-function devices. BLE also supports various connection modes, including broadcast and connection-oriented communication, allowing devices to balance between

conserving power and maintaining robust connectivity based on the application's needs [78].

The low energy consumption of BLE is achieved through brief data transmission intervals and a low-duty cycle, where devices can remain in low-power states between transmissions. This power-saving feature makes BLE highly suitable for applications in smart homes, health monitoring, and remote control systems, where maintaining long battery life is critical. Compared to traditional Bluetooth, BLE prioritizes energy efficiency over data rate, achieving a balance that supports frequent but brief data exchanges while preserving battery life [78].

Wi-Fi: Wi-Fi, based on the IEEE 802.11 standard, is a widely used wireless communication protocol that provides high-speed data transfer for local area networks. Operating primarily in the 2.4 GHz and 5 GHz frequency bands, Wi-Fi allows multiple devices to connect to a central access point, which allows for efficient data sharing and internet connectivity within a local area. The protocol supports data rates ranging from several Mbps to several Gbps, depending on the specific 802.11 variant , making it ideal for applications requiring substantial bandwidth, such as video streaming, file transfer, and real-time data monitoring [79].

Wi-Fi uses Orthogonal Frequency-Division Multiplexing (OFDM) to divide the wireless channel into multiple subcarriers, improving spectral efficiency and reducing interference from overlapping channels. This approach, combined with Multiple Input Multiple Output (MIMO) technology, allows Wi-Fi to support high data throughput and multi-device connectivity. MIMO enables multiple antennas to transmit and receive signals simultaneously, increasing data rates and coverage by leveraging spatial multiplexing. As a result, Wi-Fi networks can handle a high density of devices, which is particularly valuable in environments like homes, offices, and industrial facilities [79].

While Wi-Fi offers significant advantages in speed and range, it also has higher power requirements than protocols like BLE, making it less suitable for low-power IoT devices. Wi-Fi's power consumption makes it more practical for applications where devices are either plugged in or require occasional charging. For embedded systems like BREATHE, Wi-Fi is best suited for applications involving large data transfers or real-time data access but may not be optimal for battery-powered devices needing long-term, low-power operation. Wi-Fi's versatility and high throughput make it an essential protocol for applications that require fast, reliable connectivity within local networks [79].

3.9.4 Serial Communication Final Selection and Justification

BLE (Bluetooth Low Energy) is the most suitable option for transferring environmental data, such as temperature and CO_2 levels, within the BREATHE'S interconnected MCUs. This is because BLE is specifically designed for low-power applications, making it highly efficient for systems that require extended operation on limited power, such as a battery-powered smart vent. Unlike traditional Bluetooth, which is optimized for continuous data streaming, BLE prioritizes energy efficiency by transmitting data in short bursts and maintaining a low-duty cycle. This aligns perfectly with the nature of environmental data, which does not require constant, high-bandwidth communication but rather periodic updates.

Additionally, BLE operates in the 2.4 GHz ISM band and uses the Generic Attribute Profile (GATT) to structure data transmission into services and characteristics, enabling a straightforward and standardized method for transferring sensor data between devices. This ensures reliable communication while conserving power, as only relevant data is exchanged.

Another key advantage of BLE is its support for both broadcast and connection-oriented modes, allowing the system to balance power conservation and robust connectivity. For BREATHE, the connection-oriented mode can securely transfer temperature and CO_2 levels to another MCU, ensuring data integrity and accuracy without the need for high-speed or high-power transmissions.

The ESP32 and nRF52832 microcontrollers, which will serve as the foundation of the BREATHE system, and are highly compatible with BLE. Their hardware and software flexibility allow seamless integration with BLE communication protocols, making them well-suited for both short-range communication and their respective use cases, such as efficient power management and high data processing capacity. While the ESP32 is capable of supporting other protocols such as Wi-Fi or standard Bluetooth, BLE remains the optimal choice. Its low power consumption, structured data management, and ability to maintain efficient and reliable communication over short distances make it the best choice for the BREATHE system's data-sharing requirements. BLE ensures minimal power usage while still providing the necessary functionality to transmit critical environmental data effectively.

Feature	Bluetooth 5.0	BLE	Wi-Fi
Frequency Band	2.4 GHz ISM	2.4 GHz ISM	2.4/5 GHz
Data Rate	Up to 2 Mbps (high-speed mode)	Optimized for low data rates	Several Mbps to several Gbps, depending on the 802.11 variant
Range	Up to 4x the range of Bluetooth 4.2	Short to medium range, optimized for low-power applications	High range within a local area network
Number of Lines	2 (TX, RX)	2 (SDA, SCL)	4 (MOSI, MISO, SCLK, SS)
Connection Modes	High-speed, extended range, and advertising modes	Broadcast and connection oriented modes	Centralized access point connecting multiple devices
Multi-Device Connectivity	Enhanced broadcast capabilities allow communication with multiple devices	Connection-oriented or broadcasting for flexibility	Supports multi-device connectivity through MIMO technology

Table 23: Comparison between power supply options

3.9 Software research

3.9.1 Coding Languages

Due to the coding environment of the ESP32 and nRF52832, the selection of coding languages must be optimized for embedded system development. Factors such as limited memory, performance demands, and compatibility with the MCU must be taken into account. Moreover, the coding environment should exhibit real-time qualities since BREATHE requires predictable and timely responses. As a result, object oriented programming languages should be avoided due to them introducing computational overhead in real time embedded

environments [80]. Features such as inheritance, polymorphism, and function encapsulation can result in unpredictability and inefficiency, especially in resource-constrained environments such as embedded systems. Moreover, languages that have garbage collection should entirely be avoided, as the garbage collector runs whenever the system decides its appropriate, which is not only unpredictable [80], but very resource intensive [81]. Thus, these factors inherently narrow down the choices of languages suitable for embedded development, especially when dealing with real-time operations and resource constrained environments. Ultimately, careful language selection is crucial to ensure that the BREATHE project meets its performance requirements while effectively utilizing the limited resources of the microcontrollers.

C: C is undoubtedly a strong choice for embedded system development. Not only is it the most common language used in embedded system development [82], but it has a plethora of advantages that make it arguably the most viable option over other languages. One advantage that C has is that it has direct access to hardware and peripherals. This is crucial for embedded applications that require control over devices such as communication interfaces or sensors [80]. Moreover, C is able to produce compact and efficient code, and is able to efficiently utilize an MCU's resources when needed. This is important for embedded systems due to limited memory and performance constraints. C also remains as a ubiquitous choice for embedded systems, as it is the standard language used in not only the ESP32 and nRF52832, but also in Arduino, TI, and many other microcontrollers. Further, C has its own variation, Embedded C [80], that is specifically optimized for embedded development. In summary, C remains as a substantial choice for a coding language due to its relationship with hardware, efficient utilization of resources, and ubiquity.

C++: C++ shares many of the same advantages that C has when it comes to embedded system development. Commonalities between the two include their close access to hardware, efficient utilization of resources, and widespread use in embedded systems [83]. A distinction that C++ has over C is that it is an object oriented programming language. Although there are concerns regarding the utilization of OOP in embedded systems, C++ can still serve as a powerful tool in embedded systems development when used carefully—by minimizing the use of OOP features and prioritizing low-level control.

MicroPython: MicroPython is an implementation of the Python programming language tailored for microcontrollers and embedded systems, much like how Embedded C is a derivative of C. One advantage that MicroPython has is its simplistic syntax. Since it is based on Python, it allows for code to be written that is not only more readable, but much simpler to develop compared to

traditional languages such as C or C++. Furthermore, it has built-in libraries that make configuring peripherals like I2C much simpler than using C or C++ [84]. Another benefit that MicroPython has is its integrated Read-Eval-Print Loop (REPL), which allows for quick iterations and testing of code. This feature is very beneficial in an embedded environment, as it allows for fast debugging and testing [codeinst]. Additionally, MicroPython is supported in many ESP32 boards, which can facilitate easy integration [85]. Despite its advantages and differences to C and C++, MicroPython has its disadvantages. C and C++ outperforms Micropython in terms of performance since it is an interpreted language rather than a compiled one [deep]. Therefore, it has slower execution time which can pose concerns to the real-time nature of BREATHE. Moreover, despite it being optimized for embedded environments, it's not only more taxing on memory than C and C++, but also has limited low level access to hardware. Lastly, MicroPython uses garbage collection, however it can be manually triggered or even disabled [86]. Conclusively, while MicroPython has disadvantages like performance and resource limitations, it offers several compelling benefits such as rapid development and ease of use, making it a possible choice for BREATHE.

Java: While traditionally associated with desktop and enterprise applications, Java has established a presence in embedded environments, albeit in a very small capacity compared to other languages [82]. Like MicroPython and Embedded C, Java ME is a version of Java specifically optimized for embedded systems and mobile devices [87]. It provides APIs and tools designed for resource-limited devices such as *Connected Limited Device Configuration* (CLDC) [88], *Mobile Information Device Profile* (MIDP) [89], and Java ME SDK. A distinction that Java has over other programming languages is its portability, specifically through its *Write Once, Run Anywhere* (WORA) feature. WORA allows code that is written in Java to run on any platform that supports the Java Virtual Machine (JVM) without modifying any source code [90].This means that the same Java code can be used across various architectures such as ARM, x86, and RISC without the need to adjust the code for specific hardware architectures, unlike languages like C or C++. While Java can be tailored for embedded environments, it still has its disadvantages. Not only does it use garbage collection, which cannot be disabled unlike in MicroPython, but it also has a substantially restrictive nature due to its reliance on the JVM. While this isn't inherently disadvantageous, it can introduce concerns within the context of embedded system development. The JVM introduces memory overhead, which can cause slower execution times and be a significant limitation for resource-constrained environments [91]. Additionally, the abstraction from the JVM limits direct access to hardware, making it difficult to optimize performance

and interact with peripherals. Consequently, these restrictions can complicate development efforts and restrict the applicability of Java in many embedded scenarios, making lower-level languages like C more favorable for such applications [91]. While Java has some features tailored for embedded development, its reliance on the JVM introduces many concerns, including memory overhead, garbage collection, and limited hardware access.

Based on the requirements of the smart vent system as well as the embedded environment of an ESP32 and nRF52832, C was decided as the best choice as the programming language for BREATHE. Its direct hardware access, efficient utilization of memory, as well as its substantial support and ubiquity in embedded development make it a very strong choice as a programming language, especially given its real-time nature. C++ was another suitable choice, however it would require careful management of resources to avoid any performance deficiencies. MicroPython offers many advantages like simple syntax and built in libraries, however it is outperformed by C in terms of runtime and management of memory. Lastly, Java also offers some benefits, however is inferior to C in terms of resource management and real-time capabilities.

	C	C++	MicroPython	Java
Object Oriented	No	Yes	Yes	Yes
Garbage Collection	No	No	Yes, can be manually enabled/ disabled	Yes
Real-Time Capabilities	Excellent, deterministic and predictable behavior	Good, can be deterministic and predictable, requires careful resource management	Limited, slower performance may inhibit real time capabilities	Poor, garbage collection and JVM inhibits real time capabilities

Advantages	Direct hardware access, efficiently uses resources, ubiquitous in embedded systems	Direct hardware access, OOP capabilities (if used efficiently), ubiquitous in embedded systems	Simplified syntax, rapid development and prototyping, built in libraries	Portability, wide range of libraries
Disadvantages	Lack of OOP capabilities	Potential overhead from OOP	Performance limitations, interpreted nature, garbage collection (can be disabled)	Performanc e limitations, memory overhead, garbage collection

Table 24: Comparison between power supply options

3.9.2 Development Environments for the ESP32

There are several development environments and tools available for programming ESP32 microcontrollers. Each offers various features and capabilities that provide essential resources for coding, debugging, and optimizing the performance of the ESP32.

ESP-IDF: The ESP-IDF is a development framework made specifically to interact with the capabilities of the ESP32, including freeRTOS integration, various communication protocols (SPI, I2C, etc), WIFI, BLE, and power management to name a few [92]. It is also compatible with Visual Studio Code, allowing for seamless development to be very organized and visually appealing for those who are comfortable with the coding environment [92]. It also features a lot of example code, helping users to dive into the learning curve. It is an industry tool that many developers use to generate compact and robust code for IoT applications [92].

PlatformIO: Another development environment platform for the STM32 is PlatformIO. Rather than being made specifically for the STM32, PlatformIO is more generalized, supporting a wide range of microcontrollers, including Arduino, ESP32, and Nordic [93]. Furthermore, PlatformIO has an extensive library management, ensuring an ease of searching, installing, and updating libraries [94]. Moreover, it offers support for hardware debuggers designed for the STM32, such as the ST-Link. Although not made solely for the STM32,

PlatformIO's extensive set of tools for embedded development makes it a respectable choice for a development platform.

Arduino IDE: Although associated with Arduino boards, it's actually possible to develop software for various other supported boards using the ArduinoIDE. This is possible through different additional libraries, including ESP32 support [95]. Additionally, the development process can be streamlined by leveraging existing Arduino libraries for peripheral interfaces such as I2C and UART, and abstracting them allowing for quick development times. It also allows for operation of the ESP32's internal functions including BLE connectivity via abstracting the internal functions, allowing for fast and clean code [95].

Wokwi: Wokwi, while not a dedicated IDE, can be a powerful tool for simulating and prototyping projects involving the ESP32. It is an online electronics simulator that can be used to simulate both microcontrollers and electronic components [96]. Wokwi not only provides an extensive selection of electrical components such as sensors, breadboards, and LCD screens, but also provides MCUs such as the ESP32, Arduino, and STM32. This can be very beneficial, as it allows for an ease of creating and testing circuit diagrams, facilitating rapid prototyping. While Wokwi won't be used for actually integrating the ESP32 within BREATHE, it is a very useful tool in prototyping and simulating theoretical implementations involving the ESP32 and other electronic components.

Upon further research, the Arduino IDE was chosen due to its compatibility with our MCUs and its ease of integration. It offers a very wide variety of libraries and tools that ease the efficiency of the development process, making it the best option for managing BREATHE's requirements. It also provides a quick means of uploading compiled code to the MCU's flash, allowing code to be immediately tested after being developed. Arduino files can also easily be exported as a .hex file, allowing for multiple ways for uploading code to the flash of MCUs on the PCB. There are also a lot of resources and documentation online for common libraries, allowing for the development process to focus on high level algorithmic development and low level hardware testing. Overall, the selection of the Arduino IDE provides the necessary components to efficiently develop and prototype the smart vent system.

Development Platform	Description	Key Features
ESP-IDF	A platform designed specifically for ESP32 MCUs	ESP-IDF provides extensive libraries for RTOS, communication protocols, and BLE connections.

PlatformIO	A general purpose environment supporting many types of MCUs including the ESP32	Extensive library management for easy searching, installing, and updating of libraries. Supports ESP32 debuggers like ST-Link
Arduino IDE	A platform designed for Arduino boards, but can be configured for various supported MCUs (such as the ESP32).	Supports existing Arduino libraries for peripherals (I2C, UART) and debugging tools. Suitable for simpler MCU development.
Wokwi	An online simulation tool for MCUs and electronic components	Useful for rapid prototyping and circuit simulation with components like sensors and displays. Allows visualization of potential ESP32 applications without real hardware, though it's not intended for full development or integration.

Table 25: Table comparing different software environments for the ESP32

3.9.3 Development Environments for the nRF52832

Similar to the ESP32, the nRF52832 has a wide variety of tools that can be used for development processes. Each tool has their own advantages and disadvantages, and each one is evaluated when considered for the development of BREATHE.

3.9.3.1 Nordic SDK

The Nordic SDK contains a large variety of libraries and example code for users to utilize the nRF51 and nRF52 series to its fullest extent [96 <-> 97]. It allows for full support of the chip, allowing for PWM outputs, low power modes, and BLE connectivity, which are the main desired functions for the vent unit.

3.9.4 Graphic Libraries

The ESP32 microcontroller family is widely recognized for its flexibility and powerful performance. One particular advantage that the ESP32 has is that it is highly compatible with various graphical libraries, providing smooth integration for creating a friendly and intuitive user interface. Its compatibility with popular

libraries such as TouchGFX, STemWin, and LVGL allows for an ease of development for a wide range of graphical applications.

TouchGFX: Designed specifically for STM32, TouchGFX is a powerful, free graphic software framework that enables the creation of high-quality graphical user interfaces (GUIs). It comprises the TouchGFX Designer, an intuitive drag-and-drop tool for easy graphic building, and the TouchGFX engine, which offers optimized rendering capabilities. A massive advantage of TouchGFX is that it has automatic code generation, which automatically generates the necessary code once the GUI is designed, significantly simplifying the development process. Additionally, TouchGFX supports a variety of customizable widgets, dynamic interactions, and comprehensive text handling, making it easy to develop user-friendly applications. Fully integrated within STM32Cube MCU Packages and compatible with STM32CubeMX, TouchGFX ensures seamless co-development of graphics and main applications in a unified environment, making it a power choice for creating high-quality graphical applications [97].

TFT_eSPI: While not as robust as other open source graphic libraries, TFT_eSPI is a simple but efficient graphics library optimized specifically for the ESP32. Moreover, it is designed for TFT LCD displays, supporting a wide range of them on the market. It offers many built in functions that allows users to create sprites, shapes, and custom colors. Likewise, it has integrated touch screen support, allowing users to designate areas on their screen for touch-based operations.

LVGL (Light and Versatile Graphics Library): LVGL is an open-source graphical library designed for embedded systems, including the STM32 family. It is optimized for resource-constrained environments, enabling the creation of visually appealing user interfaces with low memory usage. Like STemWin and TouchGFX, LVGL supports a wide variety of graphical widgets, such as sliders, buttons, and charts. Moreover, it supports more advanced features, such as animations. Its efficient rendering capabilities, including anti-aliasing and real-time widget management, help save RAM by dynamically creating only currently visible elements [99].

Graphic Library	Description	Key Features
TouchGFX	A GUI framework specifically for STM32, enabling high-quality graphical interfaces.	TouchGFX Designer for drag-and-drop GUI design, automatic code generation, a variety of widgets, and optimized rendering. Seamlessly integrates with STM32CubeMX for co-development of graphics and applications.
TFT_eSPI	A graphics library optimized for the ESP32 and other MCUs to interface with SPI-based TFT displays.	Offers sprite based graphics, support for touch input, custom fonts, and support for many different graphic drivers.
LVGL	Open-source library optimized for embedded systems, including STM32, to create user interfaces in resource-limited environments	Provides various widgets (sliders, buttons, charts), supports animations, and has efficient rendering features with real-time widget management and RAM-saving capabilities.

Table 26: Graphics Libraries Comparison

4.0 Relevant Standards and Design Constraints

4.1 Design Standards

This section will discuss the relevant standards applied in the project, particularly those associated with communication protocols, air quality, PCB development, and software development

With the design and implementation of BREATHE, adherence to industry standards is crucial. Standards ensure reliability and safety across the various components and technologies that will be used within the smart vent system. By following established guidelines for aspects like data transmission and sensors, the system will meet important performance and safety requirements. Additionally, compliance with regulatory frameworks governing wireless communication protocols, such as Bluetooth BLE, and the electrical integrity of

system components will ensure seamless integration. This adherence guarantees the system's operational stability, longevity, and compliance with technical regulations.

4.1.1 Communication Standards

Bluetooth Low Energy (BLE) Standard: Bluetooth Low Energy (BLE) is a wireless communication standard that is designed for short range and low power operation [100]. BLE operates in the 2.4 GHz band and has a very low range of power which ranges from 0.01mW to 100mW [101]. BLE also leverages several methods that enhance its reliability, such as a frequency hopping transceiver, amplitude shifting keying modulation and shaped binary frequency modulation, as well as frequency division multiple access (FDMA) and time division multiple access (TDMA).

BLE is heavily based on the architecture of the general Bluetooth core system, leveraging a dual stack system that includes both traditional bluetooth and bluetooth low energy. At a high level, this system consists of both a host and a controller. The host layer manages numerous high-level protocols such as *Attribute protocol/Generic attribute profile* (ATT/GATT) and the *Security Manager Protocol* (SMP). These protocols handle device discovery, data exchange, and security in BLE communications Moreover, it also includes a *Logical Link Control and Adaptation layer* (L2CAP), which is responsible for managing the flow of data between the host and controller. The controller section manages both BLE and traditional bluetooth operations. More specifically, the BLE controller handles the physical transmission of data and manages links with other BLE devices through the Link Manager and Link Controller [101].

BLE also employs several procedures for device communication. It begins with the advertising procedure, which allows devices to broadcast data without connections, making use of both the primary and secondary channels to mitigate congestion of data. Next, the scanning procedure listens for unidirectional broadcasts of data from any advertising devices casting data. Within this procedure are scanning devices, which can request additional data and initiate connections. The discovering procedure allows devices to find any other ones over a given proximity. Once discovered, devices will use connected mode to maintain communication. Finally, periodic advertising is used to synchronize broadcasts for data consistency, and decision-based scanning is used to optimize data handling by filtering out any packets that each device is not using.

The Bluetooth Low Energy (BLE) standard significantly enhances the development of the smart vent system by providing energy efficient and reliable communication between multiple devices. This standard is extremely viable, as it will allow the two different microcontrollers to communicate together effectively while saving a significant amount of power. The aforementioned features mentioned also highlight BLE's positive impact on the smart vent system. Its low range of power facilitates a longer life span, especially since the MCUs will communicate often with each other. Moreover, the modulation methods that BLE employs, such as its frequency hopping transceiver and amplitude shift keying modulation mitigates signal interference and fading, which helps protect the integrity of communication between the two MCUs. FDMA and TDMA also assist in preserving the robustness of signals being transmitted by reducing signal congestion and interference that are transmitted from devices.

The previous features mentioned highlight BLE's positive impact on the smart vent system. Its low range of power facilitates a longer life span, especially since the MCUs will communicate often with each other. Moreover, the modulation methods that BLE employs, such as its frequency hopping transceiver and amplitude shift keying modulation mitigates signal interference and fading, protecting communication integrity. FDMA and TDMA also assist in preserving the robustness of signals being transmitted by reducing congestion and interference that are transmitted from devices.

Furthermore, the BLE architecture enhances the design of BREATHE. In the host layer, ATT/GATT protocols allow for an efficient transmission of data by defining a simple protocol to discover, group, and access data services and characteristics. Additionally, the L2CAP layer helps guarantee smooth communication and data flow between devices by reducing latency and fastening system response. This aspect is particularly advantageous, as it allows the MCU that manages the environmental sensors to quickly relay data to the MCU responsible for controlling the vent. Subsequently, the architecture of BLE also integrates numerous security features such as encryption and authentication using a method called *Counter with Cipher Block Chaining-Message Authentication Code* (CCM) [101], which ensures data integrity and packet security [101]. While security isn't necessarily a big area of concern, it remains an important consideration to ensure the data being transmitted is protected.

BLE procedures also enhance the functionality of BREATHE. The advertising procedure allows for unidirectional broadcasts to be made between the two MCUs, which eliminates the need for a physical connection between them. This

allows for the two MCUs to communicate wirelessly, making it both practical and efficient for indoor environments such as residential and commercial spaces. Moreover, the scanning and discovering processes enable quick identification and connection between devices, facilitating seamless interaction between the microcontrollers. In the connecting mode, a reliable and continuous link between the MCUs will be able to be established. Also, a method called *connection subrating* [101] can help with reducing power consumption by allowing the MCUs to reduce their active duty cycle without sacrificing communication strength, contributing directly to a longer life span. Additionally, decision-based scanning optimizes data handling by filtering irrelevant packets, which further conserves power.

In conclusion, adhering to the Bluetooth Low Energy (BLE) standard is pivotal for the successful operation of the smart vent system. Its utilization of efficient communication protocols as well as reliable data transmission methods help the interactivity between the two microcontrollers, and its architecture supports reliable data exchange while maintaining security. Overall, BLE's capabilities not only improve operational efficiency but also contribute to a longer lifespan for the smart vent system, making it an ideal standard to conform to.

I²C Standard: The I2C standard is based on the UM10204 I²C-bus specification and user manual. It is one of the most common communication protocols, being implemented in over 1000 different ICs that are manufactured by more than 50 companies. I2C has distinguishable characteristics compared to other methods of communication. One feature is its dual-bus topology. As its name suggests, it uses two wires, serial data (SDA) and serial clock (SCL) to carry information between any device connected to them. Furthermore, each line allows for bi-directional data transfers, which contrasts with other protocols like SPI or UART where separate lines are used for transferring data. Another distinguishable trait are software addressable devices with unique addresses. While this feature is found in other communication methods, I2C's differentiates itself by using a master/slave device relationship that doesn't require a dedicated master device. Moreover, I2C supports multiple masters and includes collision detection and arbitration that prohibits devices from interfering with each other. I2C will greatly benefit the smart vent system, as both the temperature sensor and CO2 sensor utilize this protocol for communication. Its simplicity and straightforward design allows for the seamless integration of multiple slave devices. By using I2C, these sensors can easily be interfaced with the ESP32 microcontroller, reducing the complexity of wiring and allowing each device to bidirectionally read and write to each other.

SPI Standard: The Serial Peripheral Interface (SPI) is a synchronous serial communication protocol used to connect a master device, typically a microcontroller, to one or more slave devices. It operates using a 4-wire interface consisting of numerous transmission methods. The first one is MOSI (Master Out Slave IN), which is for data sent from the master to the slave. Secondly, SPI uses MISO (Master In Slave Out) for data sent from the slave to the master. Next, is SCK (Serial Clock), which is generated by the master to synchronize data transmission. Lastly is CSB (Chip Select), a low active signal that selects the slave devices for communication. Data is transmitted in a master-slave configuration where commands and data are shifted bit by bit, with the most significant bit (MSB) sent first. Communication begins when CSB is pulled low and ends when CSB is pulled high, allowing for both read and write operations. The maximum SPI clock frequency is typically 500 kHz, enabling data transfer rates of up to 6600 samples per second per channel for specific commands. While both I2C and SPI utilize a master-slave relationship when transmitting data, SPI differentiates itself from I2C by using full-duplex communication rather than half-duplex. This means that data can be sent and received simultaneously, resulting in lower latency. In BREATHE, the SPI interface will be crucial for enabling communication between the microcontroller and LCD touch screen. This high-speed protocol will facilitate rapid data transfer, ensuring that user inputs and sensor data can be displayed and processed in real time. The fast response time provided by SPI will enhance the user experience by allowing for smooth and responsive touch interactions, which is essential in applications that require immediate feedback [102].

4.1.2 PCB Design Standard

IPC-2221 Generic Standard on Printed Board Design: IPC-2221 is a widely recognized standard regarding PCB design. It provides basic information regarding the general requirements for PCB design as well as discusses design recommendations and principles, such as material selection power distribution considerations, electrical performance, component routing, and general placement requirements [103]. The standard ensures that PCB designs meet critical requirements for manufacturability, reliability, and performance for many applications. Since BREATHE will incorporate two separate PCBs—one for the LCD screen and sensor system to display information and gather environmental data, and the other for the motor system to adjust the vent cover—ensuring the proper design for both is critical. IPC-2221 provides essential guidelines for optimizing these PCBs, from managing signal integrity and thermal dissipation on the sensor and LCD PCB to addressing noise control and current-carrying

capacities for the motor PCB. These design principles will enhance performance, minimize interference, and ensure reliable operation [103].

For the PCB with the MCU, sensors, and LCD screen, IPC-2221's guidelines on component placement, trace routing, and power integrity not only help make the design of the PCB board more organized, but also aid in managing the varying power and signal requirements for each part. For example, with respect to component placement, the standard recommends that through-hole parts should be mounted on the side of the PCB board opposite to the side that will be in contact with the solder. It also considers orientation and accessibility, stating that components should be mounted parallel to the edges of the printed board [103], and electronic components should be spaced so that they are not obscured by other components. IPC-2221 also provides considerations regarding the power and signal integrity concerns. For example, it indicates that a low RF impedance should be maintained throughout the DC power distribution, as a design with an improperly incorporated ground can lead to RF emissions. Likewise, the standard also recommends isolating circuits with large frequency differences (i.e, low frequency and high frequency circuits in order to minimize possible interference [103].

Additionally, the standard highlights that power supply design must take adequate thermal dissipation into account. Section 7 outlines thermal management requirements, and emphasizes several factors such as the method of heatsink mounting, soldering temperature requirements, and use of dielectric materials [103]. For the PCB with the motor, power consumption will likely be the highest as a result of the motor's stall current. By adhering to the IPC-2221 standard, incorporating a power supply within the PCB that is conscious of thermal dissipation is important. Section 7 outlines thermal management requirements, and emphasizes several factors such as the method of heatsink mounting, soldering temperature requirements, and use of dielectric materials [103]. IPC-2221's recommendations on trace widths and clearances also help prevent overheating and ensure current-carrying capabilities are sufficient for the motor. Moreover, the PCB with the motor may face issues managing noise from the motor's switching operations and protecting the board from electrical transients that could damage the MCU. The standard has information that can address this concern, as section 6.4.6 deals with managing switching noise, and recommends techniques such as the integration of decoupling capacitors and smaller vias to handle this. This is particularly relevant for mitigating noise from the motor's switching operations and protecting the MCU from electrical transients [103].

Overall, by adhering to IPC-2221 guidelines, BREATHE will ensure that both PCBs will be designed for optimal performance, reliability, and safety, addressing key considerations such as thermal dissipation, noise management, and power distribution, which are essential for the system's long-term functionality.

4.1.3 CO2 and Temperature Standards

NIOSH Standard for CO_2 Exposure: The National Institute for Occupational Safety and Health (NIOSH) provides guidelines detailing the numerous impacts of CO2 in a workplace environment. It specifies the recommended airborne exposure limit over a 10-hour work shift is 5000 ppm. Moreover, it also indicates that a concentration of 30000 ppm should not be exceeded at any time within a 15-minute work period. NIOSH also emphasizes the risks associated with being exposed to elevated levels of CO2. When CO2 levels exceed 5000 ppm, NIOSH highly recommends using a NIOSH approved air-supplied respirator or an auxiliary breathing apparatus. Additionally, when CO2 exceeds 40000 ppm, NIOSH classifies this as life-threatening and strongly advises to use a full facepiece and an emergency escape air cylinder on top of a NIOSH approved air-supplied respirator. NIOSH also provides information about both short term and long term effects of exposure to dangerous levels of CO2. Short term effects outline symptoms of both initial contact and prolonged exposure. Initial contact symptoms may include irritation of skin and eyes, and prolonged exposure may result in headaches, difficulty in breathing, and death. Long term effects include an increased risk of cancer and reproductive hazards. Lastly, NIOSH provides instructions on how to manage CO2 in workplace settings. This includes how to handle CO2 leaks, protocols on operating and storing CO2, and workspace safety practices when working in environments with elevated CO2 levels. Understanding the risks of high levels of CO2 as well as knowing when CO2 levels start to become dangerous is very important. It not only provides guidance on defining what is considered dangerous for the CO2 sensors, but also helps determine what is a sufficient range for a reliable CO2 sensor. Moreover, knowing the risks of hazardous CO2 levels highlights the importance of prioritizing the safety of users who use the smart vent, justifying the inclusion of features like LCD screen warnings and a buzzer to alert users [104].

OSHA 2017 Standard for CO2 Levels: The Occupational Safety and Health Administration provides similar guidelines to NIOSH regarding the exposure limits of CO2. Instead of using the *Recommended Exposure Limit* as a metric for defining a recommended limit, it uses the *Permissible Exposure Limit* to determine the maximum safe threshold for CO2 levels. It is the same as NIOSH, being at 5000 ppm, however it is measured during a 8 hour work period rather

than a 10 hour work period. Furthermore, along with the NIOSH standard, understanding the exposure limits as described by OSHA standards helps further contextualize what is a sufficient range to configure the CO2 sensors [104].

ASHRAE 62.1 Standard: The American Society of Heating, Refrigerating, and Air-Conditioning Engineers guidelines for indoor air quality highlight various implications regarding CO2 levels. It emphasizes that CO2 concentrations above 1000 ppm may indicate potential inefficient ventilation methods and a higher average of CO2 levels may be an indication of poor air mixing. Moreover, inconsistent levels of CO2 that are measured overtime may result in misleading low readings. Elevated CO2 levels can indicate factors such as increased occupancy, insufficient air exchange rates, and poor air distribution. Due to a possible fluctuation in CO2 levels, the standard highlights the importance of choosing a well calibrated and reliable CO2 sensor. The SCD41 adheres to this, as it comes pre-calibrated and doesn't require external calibration. Likewise, the standard highlighting variables like poor air mixing and inefficient ventilation allows for extraneous variables to be mitigated by recommending the use of tools such as temperature profiles to remove any potential CO2 measurement discrepancies. This will ensure accurate and reliable environmental data [105].

ASHRAE Standard 55: ASHRAE Standard 55 details various temperature ranges in regards to maintaining thermal comfort in indoor environments. It focuses on the balance between air and radiant temperature while also taking factors such as clothing insulation and metabolic rate into account. One method that the [106]. ASHRAE standard utilizes the *Analytical Comfort Zone Method,* which establishes metrics such as clothing insulation, metabolic rate, and airspeed to determine the comfort in a room with respect to temperature. For instance, ASHRAE states that at a clothing insulation level of 1.0 is correlated to a cooler temperature range spanning from 68°F to 77°F). Conversely, a lower clothing insulation level of 0.65 is correlated to a temperature range spanning from (73.4°F to 80.6°F). Moreover, the *Elevated Air Speed Comfort Zone Method* takes airspeed into account, indicating that each temperature limit fluctuates when airspeed exceeds 0.2 m/s. The information provided by the ASHRAE Standard greatly contextualizes what temperature ranges are considered comfortable and uncomfortable. By understanding the information given by this standard, BREATH's software can define acceptable temperature thresholds, thus allowing for the vent to more accurately change the airflow going into indoor environments [106].

4.1.4 Software Standards

C Standard: As of 2024, the C programming language standard is based on C18 (ISO/IEC 9899:2018) [107]. However, the most recent standard in development is C23 (ISO/IEC 9899:2024), which is expected to release sometime in October or November of 2024. The C standard is very important as it defines the core specifications needed to write reliable programs. The standard outlines numerous aspects, such as C's syntax, which are the formal rules for writing code, and constraints that determine what is allowed or not allowed within the language. Moreover, the C standard also specifies the representation of both input and output data, as well as various functions that C provides [107]. Since BREATHE's software will be developed in C, it is important to closely adhere to C's standards. Doing so ensures reliability by enforcing rules that not only prevent undefined behavior, but also makes sure that fundamental concepts like memory management and standard library functions behave consistently within implementations. More particularly, this is crucial in embedded environments, where even minor inconsistencies can lead to significant errors.

4.1.5 Safety Standards

IEC 61508 Standard: The IEC 61508 standard, developed by the International Electrotechnical Commission outlines aspects to be considered when electrical/electronic/programmable electronic (E/E/P) components are utilized to carry out some kind of safety function [108]. Two central objectives of this standard are to facilitate the development of safety within systems that use E/E/P components as well as enable the development of other safety standards in sectors where such standards are lacking [108]. For instance, section 6 details various responsibilities and actions that should be considered if a system intends on incorporating some kind of safety function. For example, subsection 6.2.9 indicates that procedures will be developed for maintaining accurate data with respect to hazards/hazardous events, safety functions, and the safety of E/E/PE systems [108]. This directly applies to BREATHE, as both the CO2 and temperature sensor will be responsible for obtaining accurate environmental data for the user. Furthermore, the selected sensors (the DHT20 and SCD41) have been optimized for obtaining very accurate readings, and will have software-defined thresholds established based off of standards such as ASHRAE and OSHA. These considerations not only have not guaranteed both the reliability and accuracy of each sensor, but also facilitate immediate safety responses whenever hazardous conditions are detected. By adhering to these guidelines, BREATHE will be able to facilitate an environment that is prepared to handle hazardous situations.

IEC 60364 Standard for Low voltage electrical installations: The IEC 60364 standard provides requirements for initial and periodic verification of electrical installation regarding systems with low voltage. This is especially important for BREATHE, as every component operates within a 3.3 to 5 voltage range. (verify this group). IEC 60364 covers numerous requirements such as installation verification, tests for verification, and fault protection. For example, section 6.4 outlines initial verification of a system, stating that every installation of a component must be verified during development before it is finalized [109]. It further emphases that precautions must be done in order to verify danger and damage won't be caused to people, livestock, or equipment, even if the circuit is defective [109]. Moreover, testing methods such as testing for polarity are outlined. Section 6.4.3.6 stresses to verify that wiring has been correctly connected to socket-outlets and similar accessories during a polarity test.

Fault protection is another area covered by IEC 60364, focusing on verifying that the installation has sufficient protective measures in place to protect the system and users from electrical faults such as short circuits. More specifically, section 6.4.3.7.1 stresses to verify the characteristics and effectiveness of protective devices or methods that contribute to them such as voltage regulation, including methods like simple visual inspections or looking at current ratings for fuses [109]. Properly functioning fault protection ensures that dangerous conditions like electric shock or fires are mitigated. Electrical installations of each component within BREATHE must be thoroughly inspected to ensure both the safety and operational reliability of BREATHE. Verifying aspects such as potential damage or defective components is essential. Moreover, preventing dangerous incidents like electric shocks or fires remains imperative to ensure the safety of users and the continued reliability of the system under all operating conditions.

4.2 Design constraints

As engineers we will all have to deal with design constraints for the rest of our lives. Constraints are what fundamentally define our capabilities and boundaries within a design. Depending on the project at hand, constraints can range from the time given to complete a project, or the budget of a project to technical constraints like the size of a component, power supply capacity or mechanical capabilities. As such they are the most important part to any project.

4.2.1 Time Constraints

One of the main constraints of this capstone project is time. This is primarily due to the fact that the team only has two semesters to bring their design to a

complete and fully working product. These time constraints as a team begin with the fact that we only have about a week and a half to think of an idea for this project. With that being said, not only does that involve research as to what ideas can be feasible and implemented within the two-semester time frame but it also involves various discussions as a team to find out our respective goals to achieve within a capstone project along with our capabilities as engineering students.

At the start the team is only given two weeks to think of an idea, decide how to appropriately implement the concept and write a ten page report detailing such along with determining nine key engineering objectives to achieve within the limited time frame. The team then has one month to research and select different components to facilitate their designs and complete a sixty page report detailing such discoveries. The remaining time in the first semester, which is one month, consists of designing the hardware and software for their design, ordering their chosen components, prototyping and initial testing along with writing the remainder of the one hundred and twenty or more page report. Most engineering students aren't dedicating their sole time to senior design and creating their designs because they are taking other classes, labs and even working jobs or internships alongside senior design. Time management is such a critical component of senior design and can really weigh on a student's morale at times.

Shipping times for certain components can also arise complications with stock of items, back ordering and general transportation delays. Living in Florida weather also plays into the effect of the time constraints of our project, with the fall semester falling in peak hurricane season, weather delays can impact component shipping and research for the team. Due to some recent hurricanes, classes were canceled for a week and a half and most of our team members even lost power for several days while classes were canceled. This greatly impacted productivity as the 'free' days off from classes that could have been used to completely focus on research and documentation writing were wasted due to having no power and internet connectivity.

Aside from these aforementioned perspectives of time constraints, when comparing the actual time given for this team to complete this capstone project to the engineering industry where they have years of experience and time to complete research and development before even beginning to prototype it's clear to see how astounding it is to see these senior design projects come to life in such a compact amount of time.

4.2.2 Budget Constraints

Budget is yet another crucial constraint within engineering projects. Not only does this team have a budget constraint of two thousand dollars, but we have a moral obligation as young college students to make our product affordable for the consumer. As a team we are very grateful for the opportunity to have Dr. Qun Zhu Sun sponsor our design, but it is also very important for us to not take advantage of Dr. Sun's generosity. As such we have been careful to not only be considerate and select the most adequate components for our design but to also factor in cost so as to not exceed budget constraints. It is imperative that we consider such constraints in order to guarantee the success of the project.

One of the most expensive components to consider in our project is the Microcontroller Unit selection. After a careful analysis of seven different MCU's based on power consumption, Bluetooth Low Energy, ease of integration, and cost, we selected the ESP32 and the nRF52832. Originally we planned to select the STM32WB55 mainly because of its low power consumption both in sleep and active modes. After initial testing, we found that there weren't many pins on the MCU, and that the ESP32 and nRF52832 was a better alternative due to their documentation and community support. After changing the design for the Control Unit to be powered by a wall outlet, we found that power was only a concern for the Vent unit. Though the cost is much more than similar MCUs, we found this expense to be necessary to ensure the vent interaction subsystem can last as long as possible off of battery power.

Another component that we found to be quite expensive is the carbon dioxide sensor. This was one of the more niche components to look for, but it was necessary for the core of our project so finding an accurate sensor was very important. Unfortunately, possibly due to their niche nature, many of the carbon dioxide sensors we found were very expensive, with most of them running above $40. We determined most of these were accurate enough for our needs, so we decided to choose the cheapest one that was still reputable.

One of the most important components of our project is a functional motor, a component that can easily become expensive depending on its specifications. It's crucial to choose a proper motor to control the vent's movements while also maintaining a reasonable power consumption. The cost of these motors range heavily to provide a combination of either large amounts of speed or torque. Examining different motors and mechanical designs, we've managed to find a selection of motors that optimized the amount of power necessary while minimizing the cost.

Although having a sponsored senior design project has many advantages and truly lifts the financial burden off of the students' shoulders, it can also create some limitations. For example, as our sponsor is a member of the University of Central Florida Faculty, they not only are teaching several classes but they are the director of the Smart Infrastructure Data Analytics Laboratory, the associate editor of the Institute of Electrical and Electronics Engineers (IEEE) transactions of Smart Grids, the secretary of the IEEE Smart Buildings, Loads, and Customer Systems (SBLC) committee, and are currently participating in 3 separate research projects; This makes our sponsor hard to get in contact with at times and as such causing further delays. With our heavily decorated sponsor juggling multiple roles they don't necessarily have time to go over every single component selection and cost with a fine-toothed comb, this puts the paramount job of critically analyzing our selections with respect to budget in our hands. This responsibility not only educates us on the principles of budgets but prepares us for the real world of engineering.

4.2.3 Size constraints

Yet another design constraint to consider is size constraints. Since this project involves automating the louvers of a ceiling vent it is critical that as a group we consider the dimensions of the vent and its available space in the duct in order to guarantee as per our specifications that our vent design can fit inside a standard vent duct. With that being said each component that we researched and selected needed to be as small as possible. For this, we tried to use batteries that were sized to fit into the vent while still being able to last as long as we specified. We also needed to limit our PCB footprint, which requires using small component packages and fewer components if possible.

4.2.4 Ethical Constraints

With sourcing the components, we also have to consider the Ethical Constraints. Not only as a team of students pursuing this project to fulfill our degree requirements, but also as future engineers wanting to create something that can be useful and helpful to society. As a team we also chose to only source our components from companies that we consider to be of high societal and ethical value. It is important that as members of society that we don't contribute or promote unethical working environments such as companies that use child labor, slavery, or poor working environments. The team is also actively working to protect users personal data such that our temperature or carbon dioxide sensor readings are adhering to piracy laws and we can ensure data security and confidentiality. Since battery life is so pivotal for our project, as a team we also are aware of choosing ethical environmentally conscious components that

minimize our energy footprint. Low power mode is a crucial aspect to our project's longevity being successful and something we take

5.0 Comparison of ChatGPT with other Similar Platforms

With the recent boom in Artificial intelligence (AI) within the last couple of years in the technology sector, it's no wonder that so many different uses for AI have become easily available to everyone. Depending on who one would ask in the general population, the viewpoints of artificial intelligence can range from, totally for it to completely against it. Large Language Models (LLMs) are the most commonly used AI platforms today. One of the more widely known LLM platforms is ChatGPT, but following closely behind them are Microsoft Copilot, Google Gemini, Meta AI and more.

5.1 Large Language Models

Large language models are a type of Artificial Intelligence algorithm which use deep learning and enormous data sets in order to "understand, summarize, generate and predict new content" [110]. LLMs work by training the models, one of the training methods is called 'unsupervised learning', which is essentially a type of machine learning in which the model derives relationships between different concepts without the need for human intervention [110]. These models are also trained using information such as various articles, books, and websites to really understand how language works and how humans communicate. The various LLMs don't think for themselves though, they utilize the patterns they have learned in order to formulate the best response.

5.2 Microsoft Copilot

Copilot was created by Microsoft and is the successor to Cortana. Cortana began as a 'virtual assistant' helping PC users perform basic tasks such as getting the current weather report, answering questions, and setting reminders. Cortana was Microsoft's response to Apple's widely known Siri where it would also pull information from the users' local device or connect to the internet for more information. Microsoft saw how Cortana had fallen behind with respect to its competitors and so they began developing Copilot. Copilot was released to the public on November 1st, 2023, and was marketed to be "your everyday AI companion" [111]. Coming in Stock with Windows 11 Copilot was also integrated with all Microsoft 365 products, Edge, and Bing. With this update, Microsoft also incorporated the AI within several of its applications such as, paint, photos, the snipping tool, Clipchamp, and notepad.

Some advantages to Copilot are improved productivity, and integration within the Microsoft ecosystem. The amalgamation of Copilot within Microsoft 365 greatly benefits the user experience and allows the user to focus on the creative personal aspects of their work by giving copilot the monotonous tasks, such as “drafting emails, generating reports, and analyzing data” [112]. This seamless integration provides the capability of a user to be more efficient and productive with their time so as to not worry about the little things such as, did they format that email properly, or is their presentation in PowerPoint of high quality that their company requires. Some disadvantages include high cost, connectivity issues, and privacy concerns. If a consumer already has a Microsoft subscription, then Copilot is automatically included, although if a user wants to just use the full version of Copilot, then they would need to pay either a monthly fee of $30 or a yearly fee of $360 [111]. As far as connectivity issues are concerned for Copilot, it seems that a constant internet connection is required, which greatly reduces the flexibility of this platform. For users that often work in environments with unstable internet connections or offline environments such as airplanes then their capabilities with copilot are limited.

5.3 Google Gemini

Google Gemini, created by Google AI is a powerful AI chatbot that was developed and released in February 2024 as a response to its fierce competitor ChatGPT. Google Gemini has four separate large language models implemented within its AI they are, Gemini Ultra, Pro and Nano [113]. Each LLM within Google Gemini is set with different versions of the platform, to correspond with the sizes and capabilities. The main capabilities of this platform involve answering questions, generating code and summarizing information. Currently Google Gemini is integrated within the google ecosystem with applications such as YouTube, Gmail, the Google website, and Google Maps [113]. While the basic version of this platform is free, the advanced versions of Gemini cost around $20 dollars a month.

Advantages to Google Gemini include variety and google application integration. Since Google Gemini can complete and process multiple types of user inputs such as images, questions, and suggestions, its variety is on clear display as it can analyze an image that incorporates text at the same time. Furthermore, the ease of Google Gemini being incorporated within applications that are used daily benefit users without them even realizing. For example, you can open the google website to complete a search and alongside the link results to your query a Google Gemini response will automatically generate and often provide the answer the user is looking for without them having to look through various links

to find what they need. Disadvantages to this platform would be that because it works best within the google ecosystem it limits its ability and usefulness in other application settings.

5.4 Meta AI

Meta AI is an artificial intelligence driven chatbot assistant developed under the parent company of Meta Platforms Inc. which was formerly known as Facebook. The AI.Meta website describes Meta as an AI that, “helps you learn, create and connect in new ways” [114]. Meta boasts that with their platforms users are able to talk with recognizable voices and share images in order for them to learn more about their surroundings [114]. The application was built using the Llama 3 large language model which is a program that can recognize and generate text and images. Meta AI was first announced in September of 2023 when it was pre-installed in the Ray-Ban Meta smart glasses. Since then it has been gradually released to the public across the globe and among various Meta social media applications. Meta AI is currently available for free to all users and is embedded within Facebook, Instagram, Messenger and WhatsApp along with its own website Meta.ai [114]. Some of the ways Meta AI can help users is by automating text message responses, help with planning events, generate suggestions and images within a conversation, answer questions and provide insights during chats [115].

Meta AI at first glance has several advantages such as user experience convenience and business benefits, however a major disadvantage is privacy. Over half of the world’s population are users of various social media platforms and Facebook, which has Meta AI integrated, is still the most widely used social media platform of them all. With this large usage of social media, it is no secret that the world is constantly giving and receiving information within this space. As such, content creation is inevitable if one is a social media user and with Meta AI’s abilities, content creation has never been so significantly easier. Meta AI can create and edit unbelievable images, make captions for said images, along with generating suggestions to make the users current creations even better. The business benefits to this platform are also noteworthy in that Meta AI can manage content by scheduling the content to be posted at the peak times where it can be seen by the largest audience, it can also supply the creator with insights on how the posts are being interacted with by the audience. All of these tools that Meta AI has under its belt greatly optimizes a creator's marketing efforts. The main disadvantage with this platform though is its user privacy. It is well known that Facebook, now known as Meta, has gone to court and been fined exuberant amounts of money for their privacy violations. It only makes

sense that since they have been so careless in the past that Meta AI doesn't fall too far from the tree so to speak. Meta AI currently utilizes public Facebook and Instagram images as a part of training for its learning model, and there are current issues with certain countries not being allowed to 'opt out' of these mass content scrapings. Living within a somewhat ethical society this violation of privacy deters new users from utilizing Meta AI and ultimately making Meta AI an ethically flawed platform.

5.5 ChatGPT

ChatGPT is an artificial intelligence AI chatbot that has exploded into everyone's homes all over the world. The artificial intelligence research company Open AI created ChatGPT and released the platform to the general public in November 2022 [116]. The GPT in ChatGPT stands for "Generative Pre-trained Transformer", and much like the abbreviation suggests it refers to how the platform processes requests and formulates responses" [116]. The way Open AI trains the ChatGPT platform is through a machine learning (ML) technique called "reinforcement learning through human feedback" (RLHF) [116]. Reinforcement learning through human feedback performs tasks asked by humans and incorporates their feedback in a reward function in order to align with "human goals, wants and needs" [117]. This way of RLHF training is implemented in ChatGPT with a thumbs up or thumbs down button that a user can select to rate the prompt responses. ChatGPT's capabilities have progressed to great extent, in its initial roll out it was mainly a question and answer platform with answers ranging from somewhat correct to extremely wrong. The current knowledge cutoff date for the platform is late 2023 and it can now go way beyond simple questions and answers. Since conception of the platform one of the new duties ChatGPT can perform is to run python code directly to assist with data analysis, simulations, graph plotting and calculations. ChatGPT can now also accept attachments for evaluation and create images based on detailed descriptions. One of the most significant ways ChatGPT has improved is with its ability to retain and reference previously shared information with the platform, this has extraordinarily enhanced the user experience and efficiency of conversations. It is clear to see how ChatGPT has dominated the AI chatbot space and become almost a household name in regards to AI. Additionally, this platform has now incorporated real time internet browsing for the most up to date information.

It's safe to say that ChatGPT is a multifaceted tool which can be extremely beneficial to anyone. Advantages to using ChatGPT include enhanced user efficiency, diversity, and personalization. Disadvantages of Open AI's ChatGPT include misinformation and privacy concerns. With all of Open AI's upgrades to

ChatGPT user efficiency has skyrocketed, the platform's ability to quickly handle mundane tasks, and answer any questions that come its way have freed up a great deal of time for users to focus their attention to more important parts of their projects. This platform has become the most famous AI chatbot for its diversity, it has a range of abilities from image generation, to solving complex math problems. ChatGPT's personalization features are also a plus due to the platform's ability to reference the prior conversations a user has had with the platform with this knowledge ChatGPT can tailor its responses based on the users preferences [116]. Although there are a lot of advantages with using ChatGPT misinformation is still a concern, inaccurate or unverified information is highly possible with the current training data used. Even though Privacy is taken seriously at Open AI the retention policies in effect could also potentially expose sensitive information and thus users should be weary when sharing such information.

Some of the ways ChatGPT could assist us in our senior design group would be in deciphering various high level language documents, examining pdf's to locate crucial information, suggest components for selection, give basic ideas on how to implement certain designs, along with proofreading and assisting with debugging and troubleshooting. On the other hand, ChatGPT could also hinder our learning experience by holding us back from free thinking, if we are constantly asking how to do x, y, z, then we will lose the critical and outside the box thinking that can fuel innovation in design and fundamentally limit progress within society. It also can be quite risky to heavily rely on ChatGPT due to the fact that inaccurate information is highly plausible.

5.5.1 ChatGPT Prompt Examples

This section reflects on ChatGPT's responses viewable in appendix E to our team's prompts in order to actively show the advantages and disadvantages of using Open AI's ChatGPT.

5.5.1.1 Prompt 1

In regards to the prompt and response in appendix E Prompt 1 it was actually quite helpful in getting a base understanding of the different types of voltage regulators and how they would be beneficial in a battery powered device. Some of the regulators mentioned are even the same ones we used as examples prior to searching the prompt. The only potential criticisms of this response would be the fact that it is very generalized and lacks some of the nuances of power and input current calculations for some of the switching converters. However, as a starting point ChatGPT could be very useful if we had no prior knowledge of voltage regulators [118].

5.5.1.2 Prompt 2

In regards to the prompt and response in appendix E Prompt 2 although ChatGPT outlines valid and applicable standards regarding temperature and CO2, it did provide a European standard, which is not applicable to BREATHE, which is based in the United States. The group's focus needs to align with relevant local regulations and guidelines, such as those established by OSHA and ASHRAE in order to adhere to various compliances within its intended environment. Moreover, relying on standards specific to a different region could lead to discrepancies in performance expectations, as foreign standards may provide information that is incongruent to those found in the United States [119].

5.5.1.3 Prompt 3

In regards to the prompt and response in appendix E Prompt 3 it unfortunately contains multiple inaccuracies. Taking a look at each point, ChatGPT stated that in low power mode, the SCD41 typically consumes 70uA [120]. However, according to the official SCD41 datasheet, it actually consumes 3.2mA and has a maximum consumption of 3.5mA [21]. As one can see, this is a huge inconsistency, as the difference between the value that ChatGPT gave and the one provided by the official SCD41 datasheet is significant. Moreover, when comparing ChatGPT's claim that the SCD41's measurement mode uses up to 120uA as well as a peak current of 2.5mA to the datasheet, it is clear that this response is also inaccurate. Based on the SCD41 datasheet, the peak current ranges from 175 - 205 mA and the SCD41's measurement mode consumes 15 - 18mA. Again, this difference in values highlights major discrepancies between ChatGPT and the datasheet. Thus, it is fair to say that when ChatGPT is in use, its contents should be verified as it cannot be accurate indefinitely.

6.0 Hardware Design

In order to create a successful senior design project and meet the course's guidelines, a Printed Circuit Board (PCB) with significant peripherals is required. To meet these specifications we decided that the most efficient way to bring our design to life was to split Breathe into two main subsystems we call, the Control Unit and the Vent interaction Subsystem. With these subsystems in mind, two PCBs are needed.

With the intention of creating two PCBs from scratch we needed to select suitable software to design them before they could be sent to be fabricated. We decided to use the free software suite KiCad to layout the PCB schematics and board design. We considered other alternatives such as EAGLE, Fusion 360, and

Altium. Although these programs are well-documented and exhibit many tools and features, we found KiCad to better align with our team's workflow and preferences due to both its accessibility and ease of use. Since we determined that KiCad was the most user friendly and accessible we were then able to begin the designing process. The ESP32 is the MCU we will be utilizing for the Control Unit and the NRF52832 is the MCU being used for the Vent Interaction Subsystem so for the design procedure to begin we needed to heavily reference both the datasheet and various application notes to design basic necessities needed for a proper functioning system. We also relied on various websites as references to easily select which pins are capable of performing the actions needed and how many were available.

6.1 Control Unit

The control unit consists of some of the major data peripherals of BREATHE, those being the LCD screen, temperature sensor, carbon dioxide sensor, and the buzzer. A block diagram of how this subsystem works can be seen below in Figure 5.

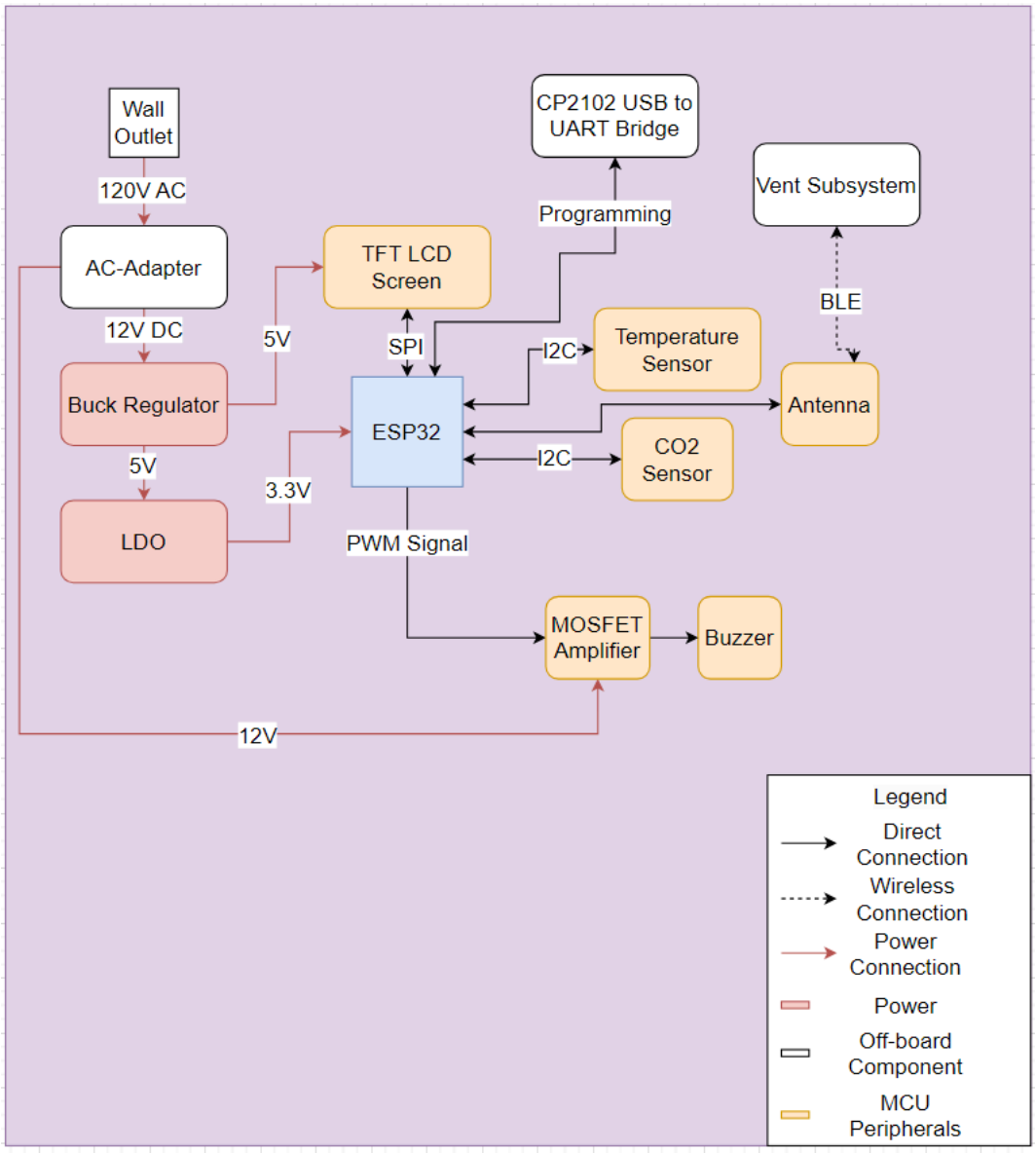


Figure 5: Hardware Block Diagram for Control Unit

6.1.1 MCU

For the ESP32 we are using for the control unit MCU, very few accommodations are needed to adapt the ESP32 WROOM32 module to work efficiently. A 22uF bulk capacitor and a 10nF decoupling capacitor are added to the VDD supply to ensure clean power delivery.

A 10k ohm resistor and an additional 10nF decoupling capacitor are put near the enable pin which leads to the SPST push button which will be used to reset the ESP32 when needed. An additional debouncing capacitor is placed in the button

loop to prevent accidental additional button presses. The boot pin also includes a similar set up. We also have additional UART pins routed out to a connector for use with an external USB to Serial converter which will allow us to flash the ESP32 WROOM32 module.

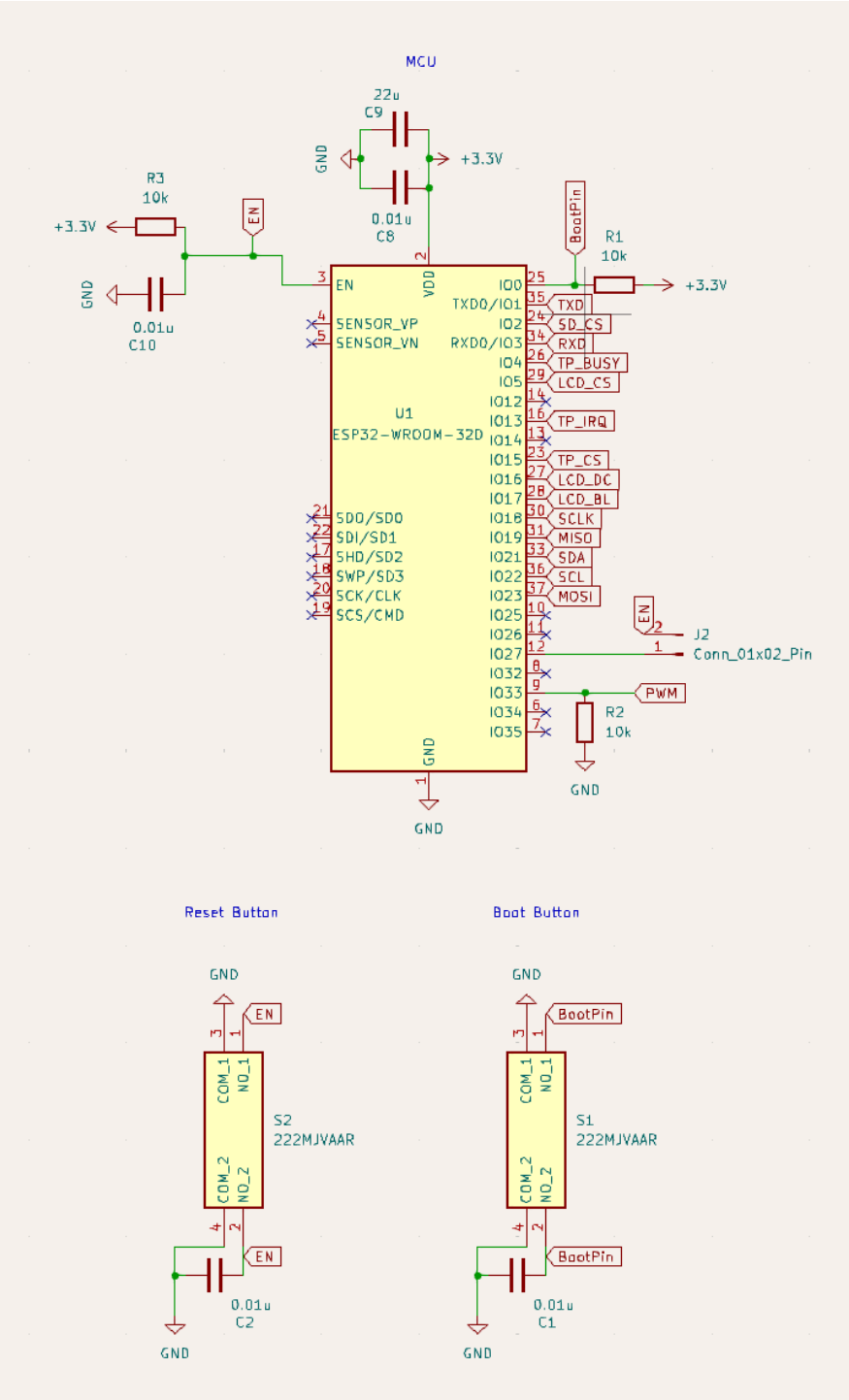


Figure 6: Control Unit MCU Isolated Schematic

6.1.2 LCD Screen

Once we made the appropriate changes to our MCU selection and our LCD screen brand we were able to properly decide how we wanted to incorporate the LCD. The options were to either incorporate the LCD screen into our PCB or connect to it via pin headers. We ultimately decided that the best decision would be to connect the screen via wires and a pin header. Following the ESP32 data sheet and the user manual for the 4 inch LCD screen, an Interface table is presented in the LCD user manual with descriptions as to what each pin needs. A slide switch was also added in order to be able to correctly flash code to the ESP32 so that the 5V needed for the LCD screen did not interfere with the 3.3V powering the MCU. Cross referencing both the data sheet and the user manual the pin configuration schematic with respect to the needs of the LCD screen can be seen below in Figure 7.

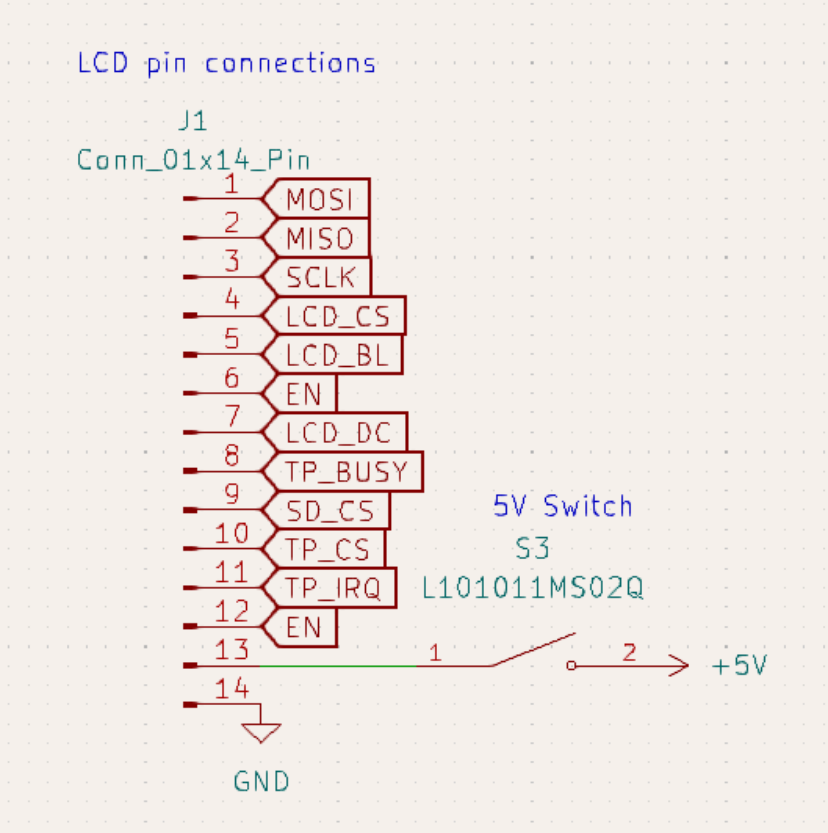


Figure 7: LCD Pin connections

6.1.3 Temperature Sensor

To begin the temperature sensor schematic, we had to take a look at the DHT20 temperature sensor datasheet to understand its working conditions and limitations. The supply voltage for the temperature sensor is 3.3V which is

denoted by the arrow pointed upwards and connected to the input voltage of the sensor (VDD). The remaining pins are SCL, SDA and Ground which are all connected respectively to the appropriate MCU pins and ground. The schematic for the DHT20 sensor is shown below in Figure 8.

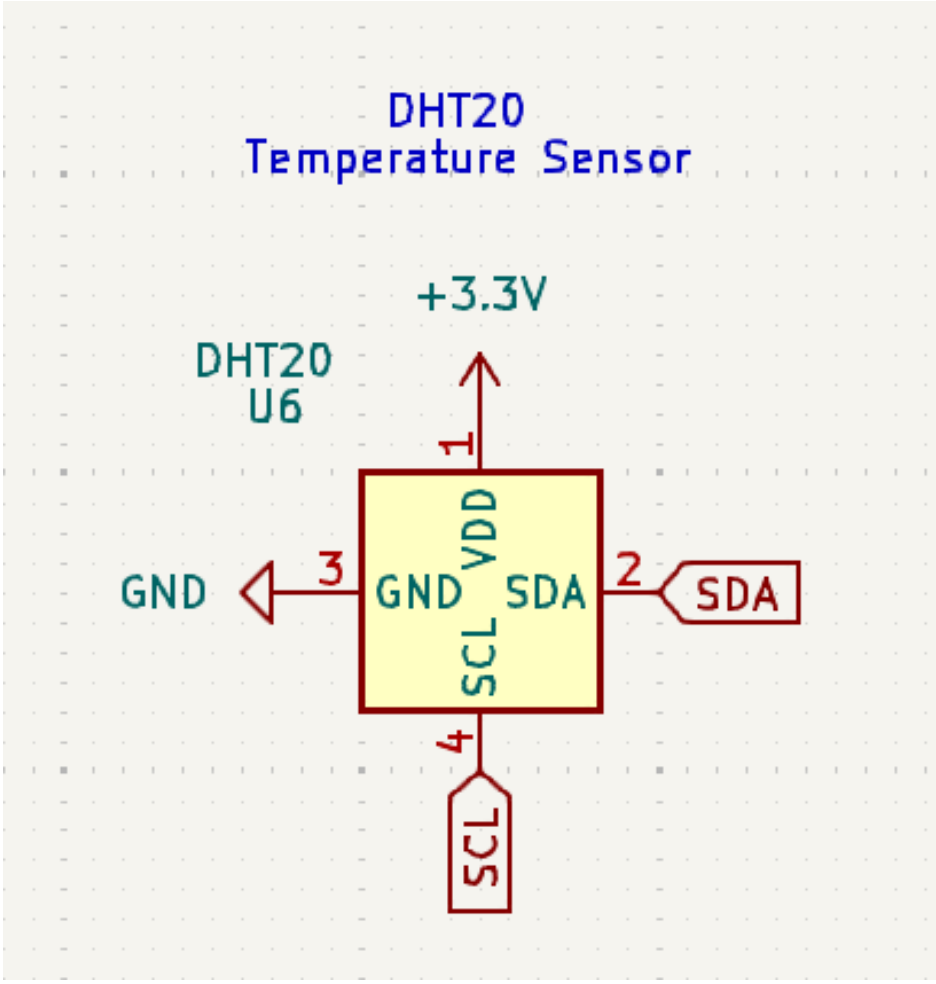


Figure 8: Temperature Sensor Schematic

6.1.4 Carbon Dioxide Sensor

The SCD41 Carbon Dioxide Sensor schematic process was very similar to how we designed the temperature sensor. To begin we examined the SCD41 data sheet to understand its working conditions and limitations. As soon as we were able to understand the main ideas of how the sensor worked we could move forward with schematic design. In the data sheet, a section titled interface specifications explained the best practice for setting up this sensor and was followed with a figure entitled typical application circuit. This typical application circuit was extremely beneficial in guiding the schematic design for the carbon dioxide sensor. This section went on to explain how the VDD and VDDH pins

were to always be kept at the same voltage, detailed how the SDA and SCL pins were for transferring data to and from the sensor, and synchronizing the I2C communication between the MCU and sensor. The data sheet then goes on to say that the SDA and SCL lines should be connected to external pull up resistors. We followed all of the data sheets recommendations and the schematic for the sensor can be seen below in Figure 9.

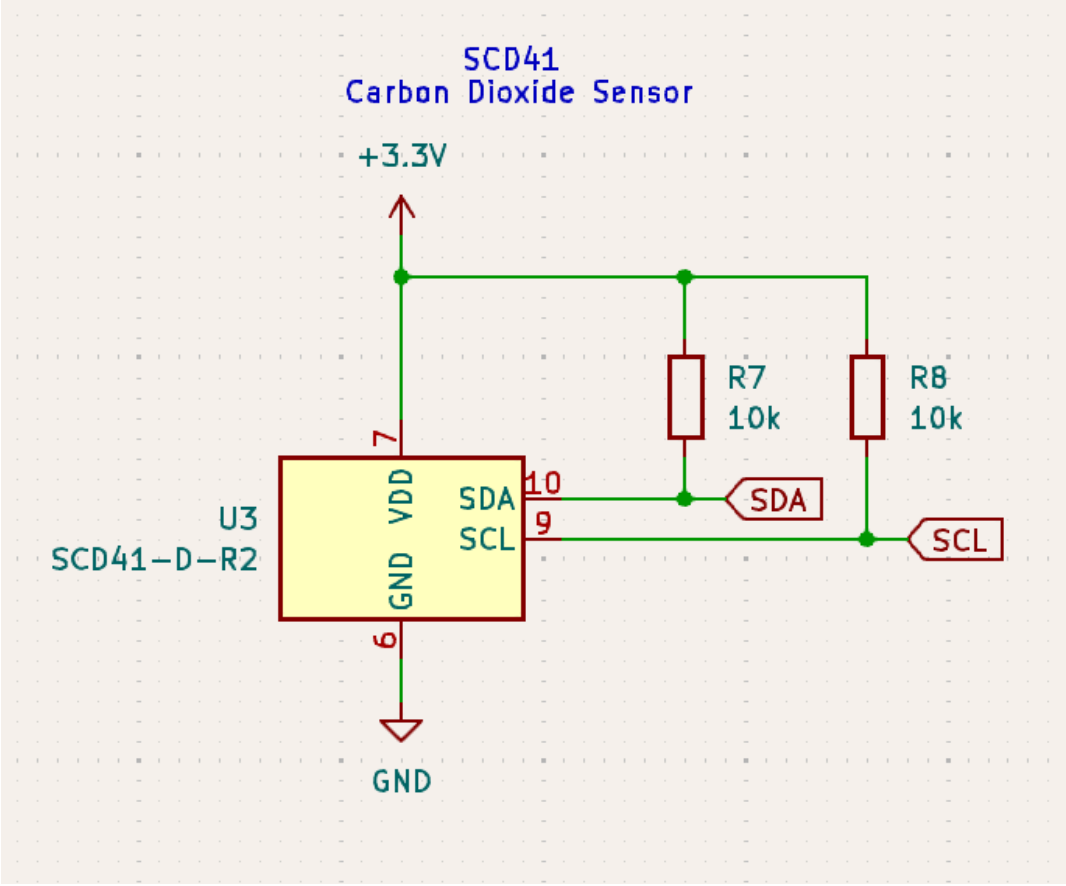


Figure 9: Carbon Dioxide Sensor Schematic

6.1.5 Buzzer

For the piezo buzzer, which will alert the user of excess Carbon Dioxide, or low battery life, the setup requires a PWM signal as an input. Since we wanted the buzzer to be louder, with the help of Dr. Weeks we implemented a MOSFET amplifier with the addition to a 12V input as seen in Figure 10. The PWM output of the ESP32 acts as a control to switch the MOSFET from saturation to cutoff, and the 12V input biases the buzzer to 12V when high. A resistor is added between each node of the buzzer to prevent one node from floating when the MOSFET is in cutoff.

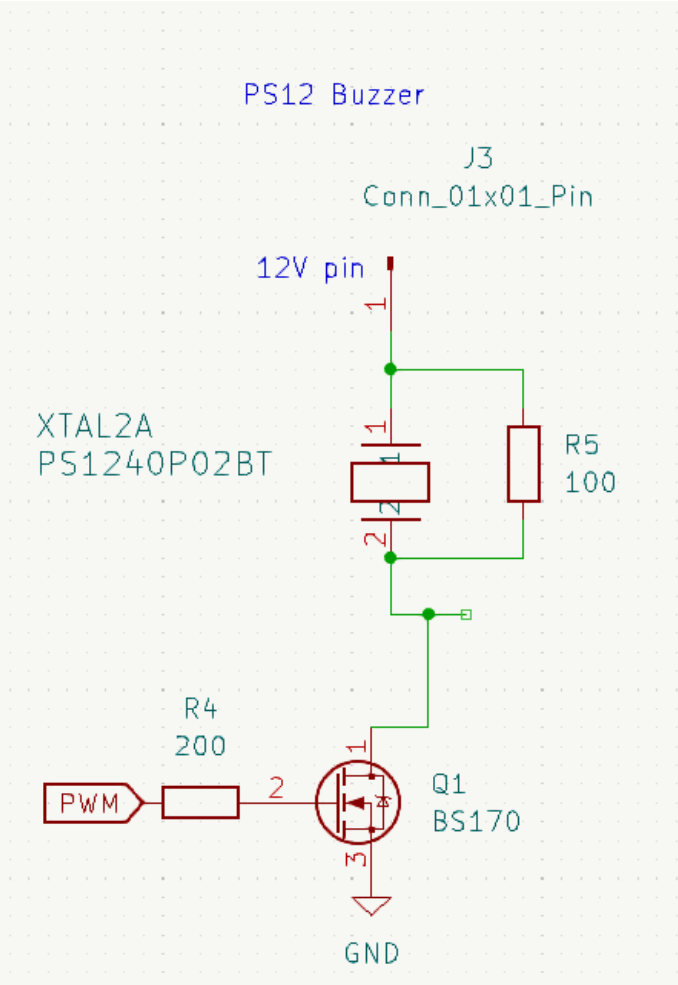


Figure 10: Buzzer Schematic

6.1.6 Power Design for Control Unit

For the control unit, the power section is designed to accommodate all of our peripherals without too much complexity. This unit will be on a separate PCB known as the Control until Regulator board and utilizes a 12V AC-DC adapter which will convert the 120V AC power from a wall outlet into a 12V DC output which will power the entire control unit. These adapters are plentiful and allow for a lot of extra headroom for our various peripherals. Another advantage is that this gives us the 12V supply for the Mosfet amplifier for the buzzer. This adapter will be connected through a standard DC barrel jack where it will then be regulated into two distinct voltage rails.

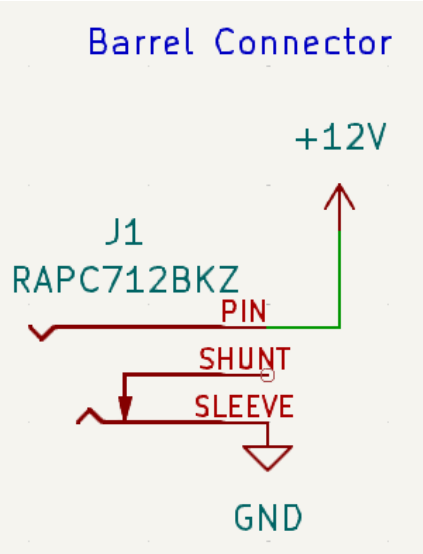


Figure 11: AC-DC Adapter Barrel Connector

Even though the LCD screen that will be incorporated into the control unit utilizes 5V logic for all of its main drivers and peripherals, the regulator board uses an LDO to regulate that 5V input to 3.3V. The dropout voltage of this regulator is relatively high, and since the MCU has a maximum input voltage of 3.6V, running both the MCU and the LCD screen off of one voltage rail is not feasible unless we were to take the LDO out of circuit by modifying the LCD. This means we need a separate 3.3V and 5V rail to power both the LCD and MCU respectively. It is important to note 3.3V was chosen as the input voltage for the MCU in this section because the CO_2 sensor requires a 3.3V input.

Because using a linear regulator to step down 12V to 5V would result in 59% of the power from this conversion to be dissipated as heat, it seemed more reasonable to use a switching regulator to deliver a stable 5V output to the LCD screen. Using a linear regulator in this manner would have likely required a heatsink, and the additional heat generated from this inefficient regulation could make the results of the temperature sensor less accurate as the enclosure would get warmer.

However, to provide the 3.3V output to the MCU we decided to use an LDO in series with the 5V buck-regulator instead of using another buck-regulator mostly to limit the design complexity. LDO's require less components and less design overhead when laying out the design on a PCB. Since power consumption is not an issue on the control unit, the increased efficiency of a buck converter doesn't hold as much impact. As long as a suitable package is chosen that can properly sink the extra heat generated due to efficiency losses (34% of the total power)

this option is perfectly suitable for powering the MCU. Moreover, LDO's are widely used in series with switching regulators to reduce noise, so this option would give the MCU a much cleaner power source.

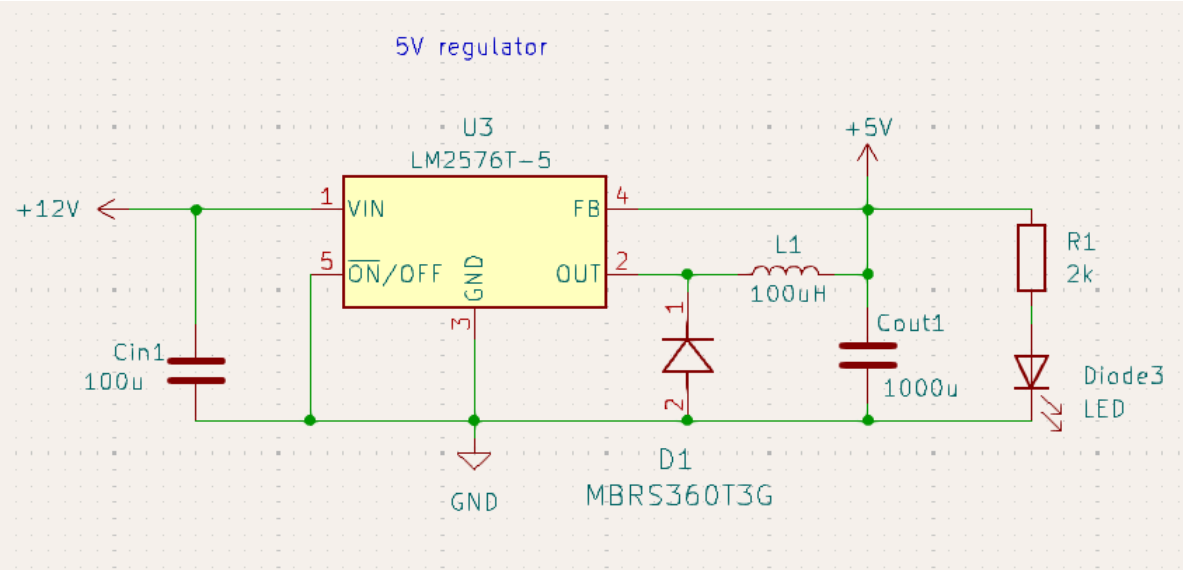


Figure 12: 5V Buck-Converter Schematic

The layout of the buck-converter mostly follows the datasheet's application circuit with some notable tweaks to suit the control-unit's specific needs [66]. The SEL pin was set with a specific resistance which would configure the voltage output to 5V. The schematic follows the datasheet's recommended capacitor and inductor values for voltage outputs above 0.8V. The enable pin EN is tied to the voltage supply at the input pin to ensure the regulator is always on. We added an LED with a suitable current-limiting resistor to give us a visual indicator to tell if the buck-converter is working properly and outputting current.

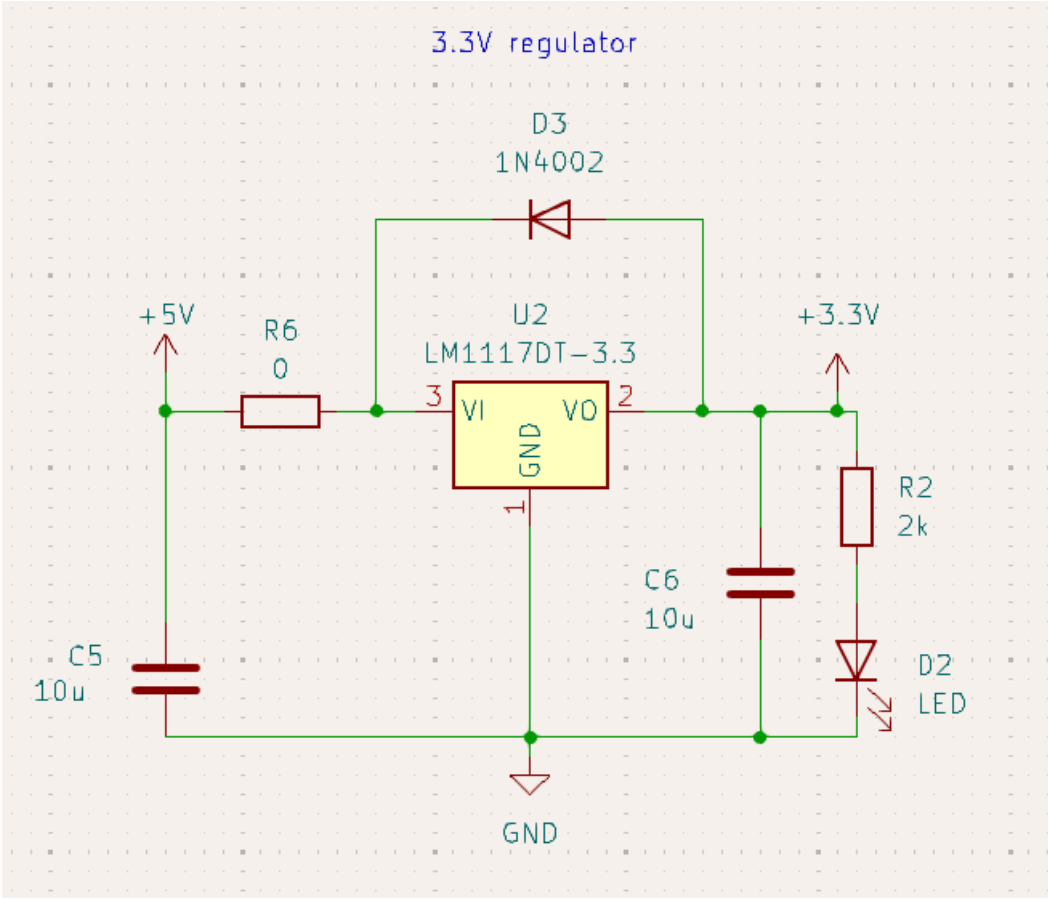


Figure 13: LDO Regulator for MCU

The LDO takes the 5V output from the buck-regulator at its VI input pin. Two capacitors are placed from the input and output to ground as specified in the datasheet [125]. Once more, an LED is placed at the output to give us an indicator that the regulator is working properly. Additionally, a 1N4002 diode is added between the input and output pins as added protection as specified in the datasheet.

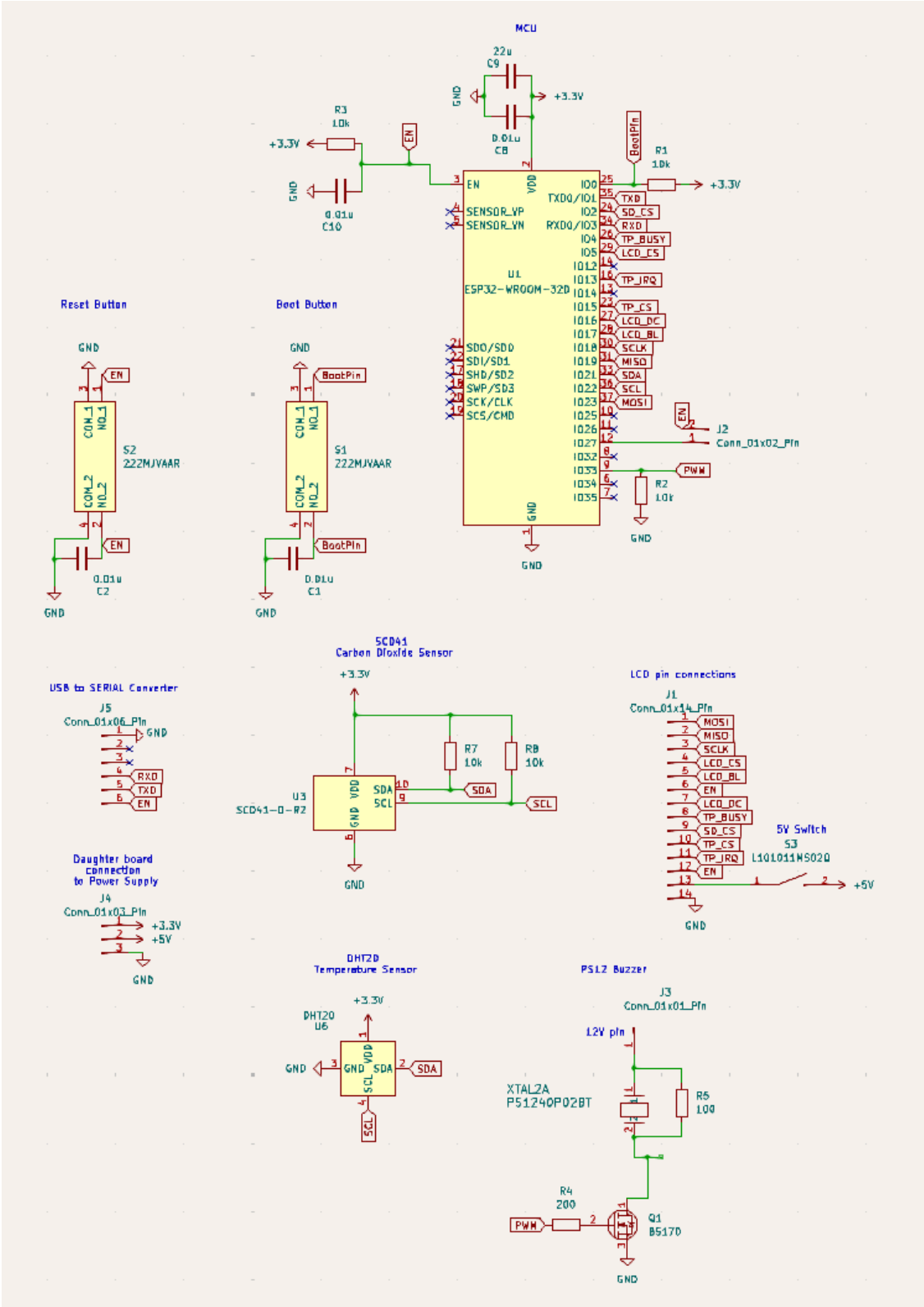


Figure 14: Overall Schematic of Control Unit PCB

6.2 Vent Interaction Subsystem Design

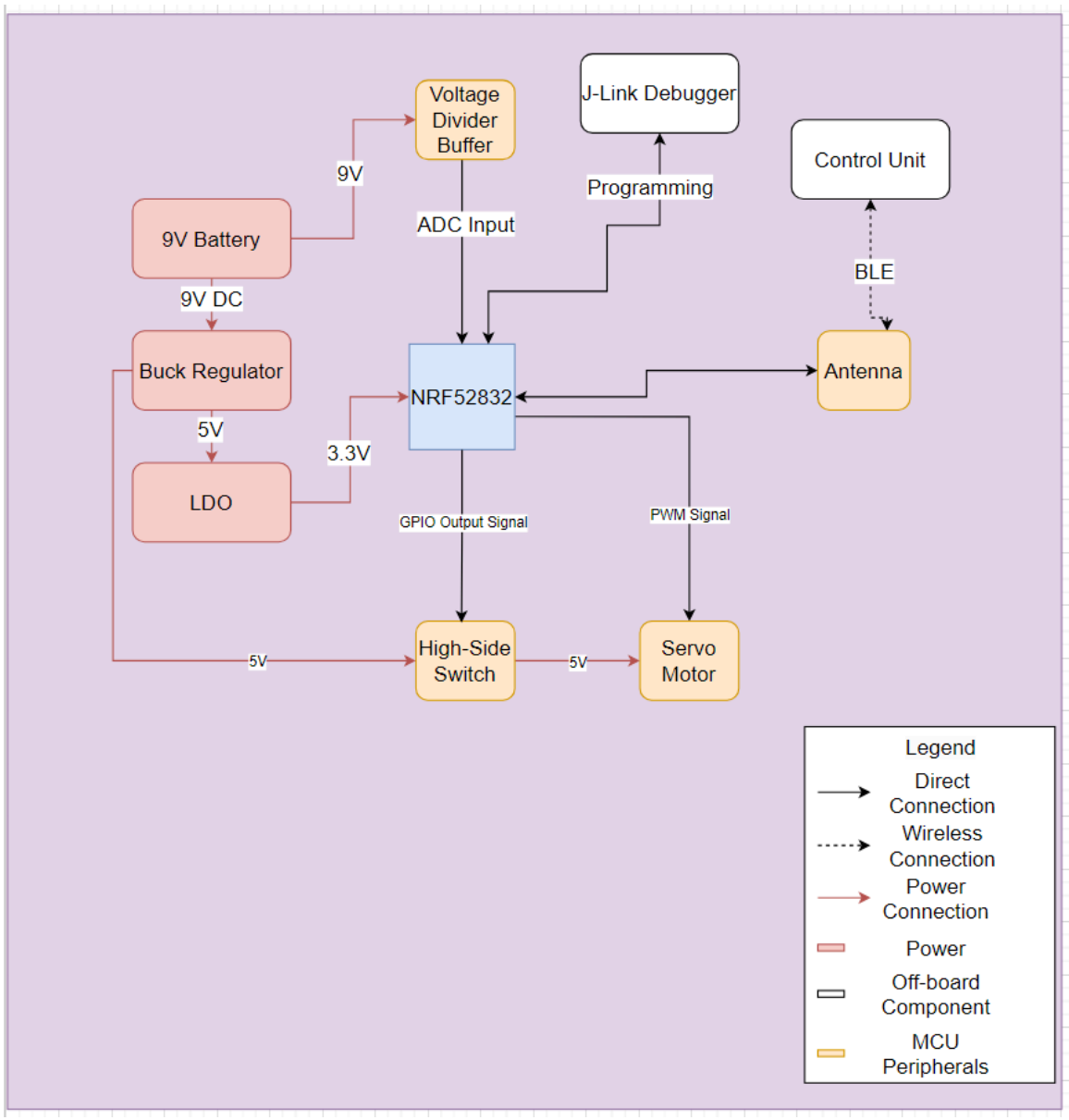


Figure 15: Vent Interaction Subsystem Hardware Block Diagram

6.2.1 MCU

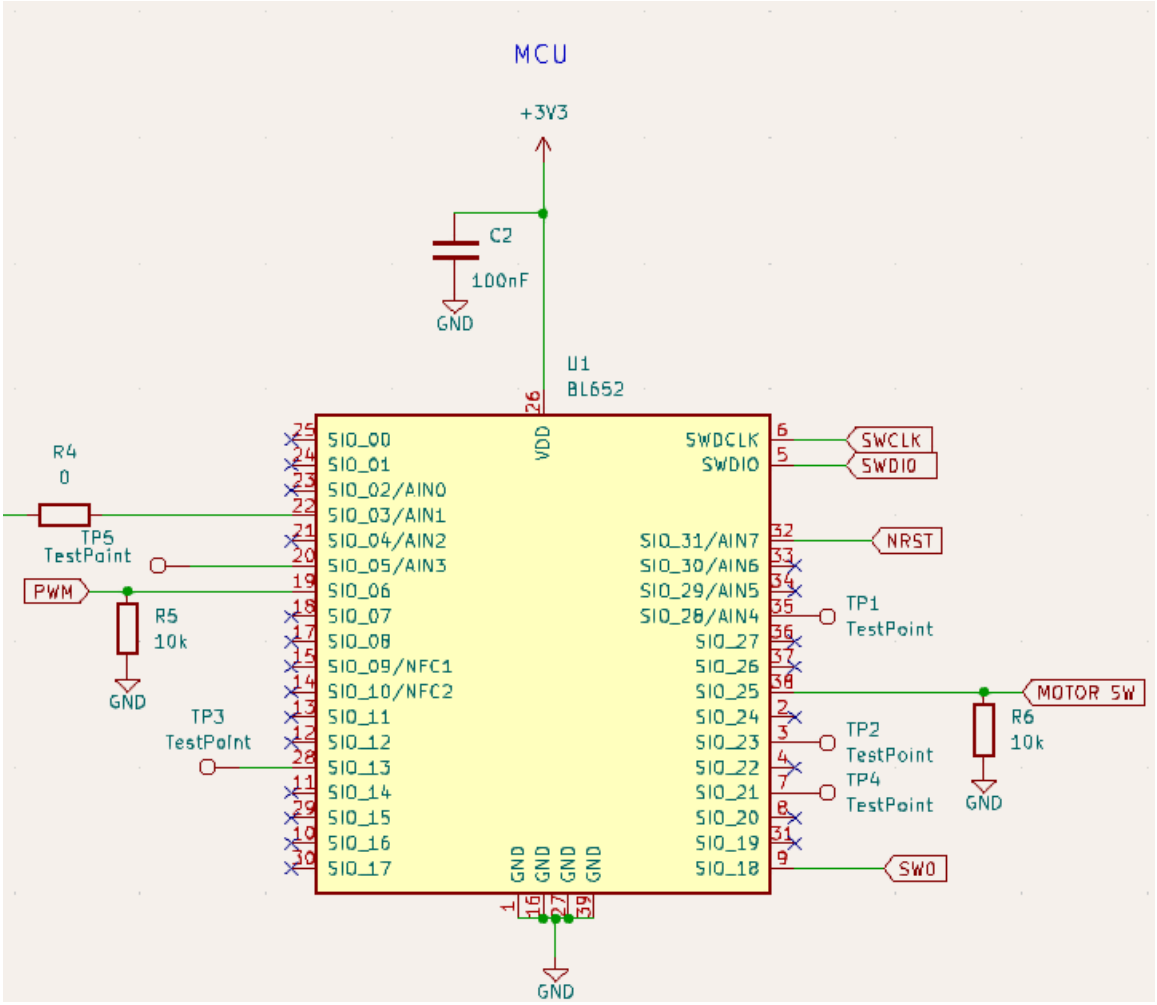


Figure 16: Vent Interaction Subsystem MCU Schematic

For the vent-subsystem, we opted to utilize the NRF52832 as opposed to the ESP32 meaning the board design and setup will be a bit different. The NRF52832 has many different integrated modules to choose from that incorporate different solutions for bluetooth antennas (such as UFL, Chip Ceramic, and PCB trace antennas) and also have differing pinouts and integrated coupling capacitors and crystal oscillators. The module we chose is the BL652-SA01 which includes all the aforementioned features we would want in a module for this MCU. Due to the heavily integrated design of this module, we only needed to add one decoupling cap on the 3.3V input for the unit to work. We also added test pads to some GPIO pins just in case we needed them later. We did end up using some of these for backup boards for our motor's PWM input.

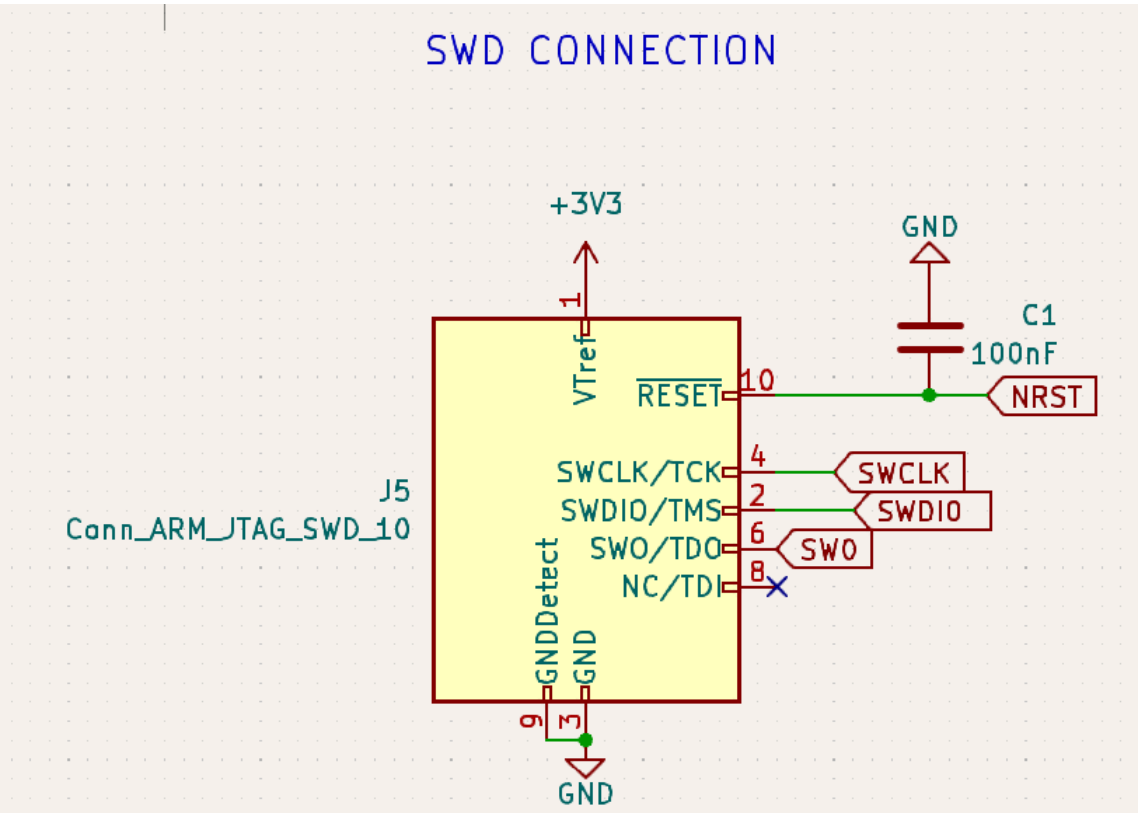


Figure 17: Vent Interaction Subsystem MCU Debugger Schematic

To flash code to the MCU, an external debugger needs to be utilized. The Segger J-Link debugger was used to flash all code to the NRF52832 as Nordic recommends. The J-Link primarily uses SWD to communicate with and flash the MCU, requiring usage of the SWD connector seen above. The reset pin is routed to the reset of the SWD connector, with an additional decoupling capacitor. The SWCLK, SWDI0, and SW0 trace pins are all routed to the SWD connector as well from the MCU.

6.2.2 Servo Motor and High-Side Switch

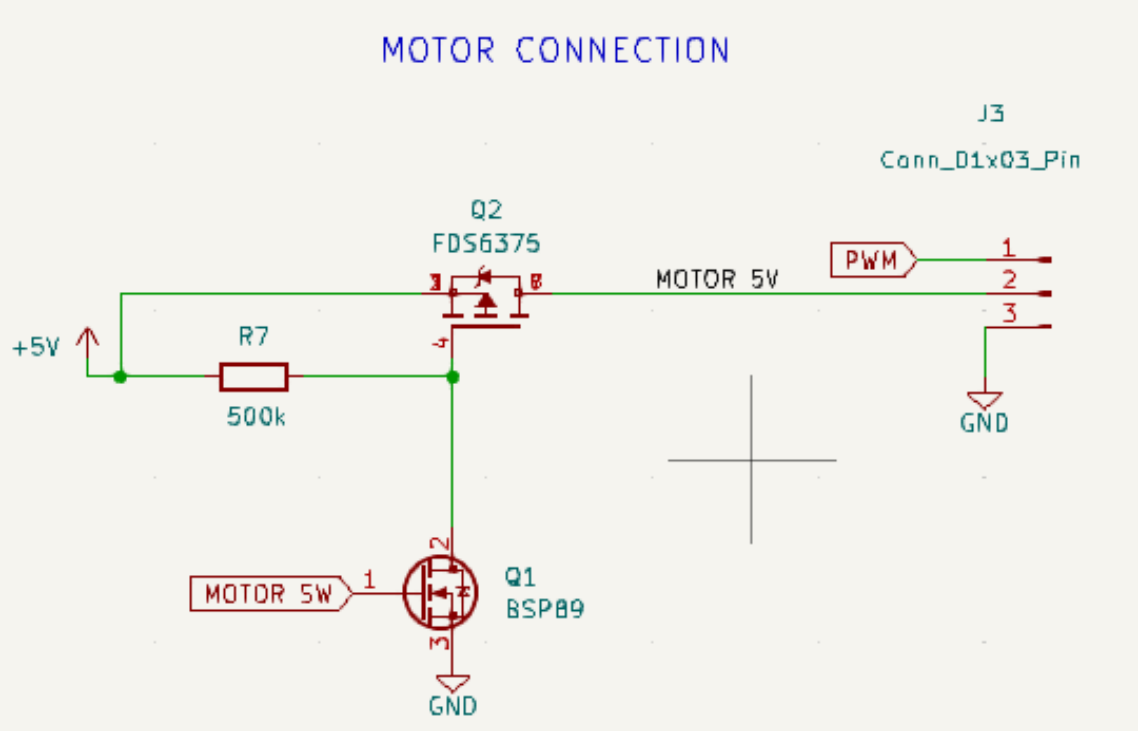


Figure 18: Motor Connection/High-side Switch Schematic

For the servo motor, we implemented a high side switching circuit with two MOSFETs and a 500kΩ resistor. The first MOSFET is N-channel and biased so that a single GPIO output (@3.3V) will saturate it and therefore pull the gate of the FDS6375 (a P-channel MOSFET) to ground. By default, the gate of this FET is pulled up to 5V using a 500kΩ resistor. When Q1 is saturated, Q2 also becomes saturated and 5V is passed to pin 2 of the motor. This switch allows the motor to be turned off during stall conditions and also cuts out idle current draw from the motor when not in use. Additionally, a PWM output on the MCU is also tied to pin 1 to move the motor.

6.2.3 Power Design for Vent Interaction Subsystem

For the vent interaction subsystem, we need two voltage rails once more, one 5V to power the servo motor and another rail to power the MCU. As previously outlined, we opted to use a buck regulator to provide the 5V rail and an LDO in series to supply the 3.3V rail for the MCU.

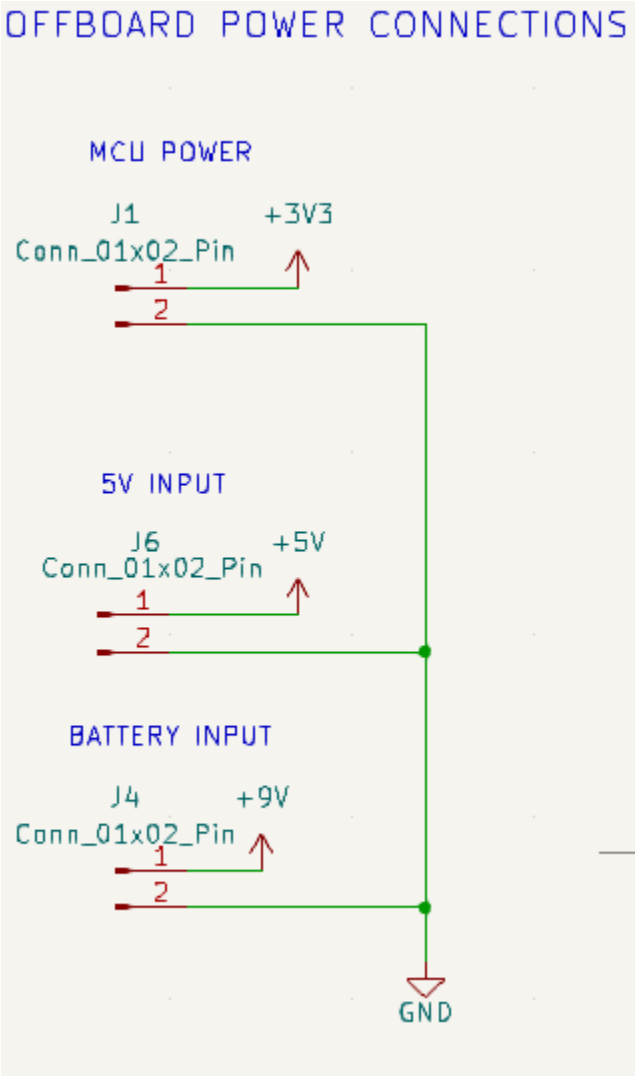


Figure 19: Vent Interaction Subsystem Input Power Connections

Board Connections: To ensure power supply faults don't interfere with the main board, the power supply is on a separate board which connects to the main vent unit using JST connectors.

Battery Input: For the battery input, two test pads are placed on the power supply board which will allow us to solder the positive and ground of the battery connector to the board.

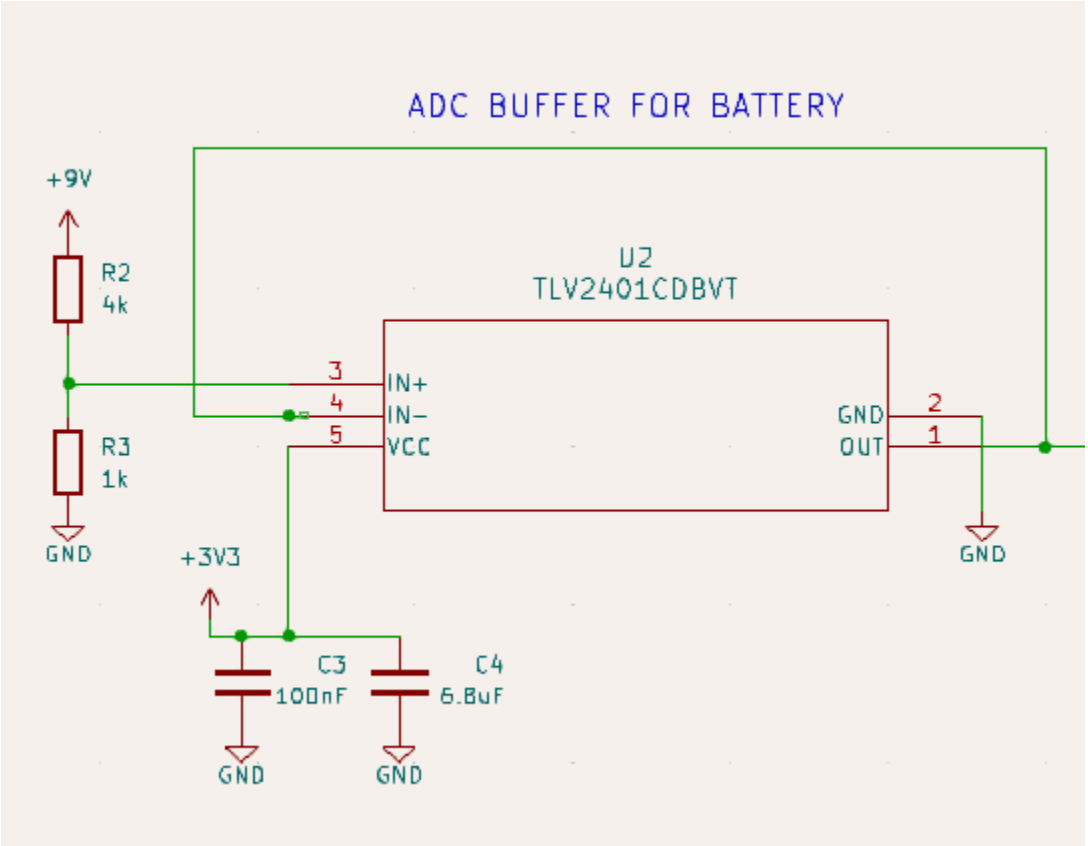


Figure 20: Battery Life Indicator Schematic

Battery Life Indicator: One crucial element in the design of the vent interaction subsystem is being able to notify the user of the battery life. In particular, we want to make sure the user is aware if the battery life is low and it needs to be replaced. Thankfully, the NRF52832 has an analog to digital converter (ADC) which can read analog voltages and convert them into bits which can be read and stored. This can be used to read the voltage value and determine the battery life in relation to the cutoff voltage of the battery and its rated nominal voltage. This value is sent via bluetooth to the control unit and, if it is near the cutoff voltage, the buzzer in the control unit will alert the user of low battery. There is one notable issue, the battery is 9V but the MCU is powered using 3.3V. 3.3V in this case is the reference voltage for the ADC and the ADC cannot read voltages greater than this reference voltage. This means it wouldn't be able to read the voltages at all, even when the battery is far below its cutoff voltage. This, however, can be easily solved by using a simple voltage divider using two resistors. This solution would give us a voltage with a known ratio that is just low enough to be within the ADC's operating voltage range. We will use a ratio of 4.9:1 which will ensure our maximum battery life would correspond to a voltage of 1.8V, far below the ADC's voltage reference. There is another caveat though, if

the impedance of the two resistors in the voltage divider is too close to the input impedance of the ADC then the voltage at that pin will be less accurate. The NRF52832 datasheet doesn't quite specify the ADC's input impedance [122]. This poses a problem with our design as even if we were to use 400Ω and 100 Ω resistors in this divider (extremely low values) the voltage output may still be slightly inaccurate. This may seem trivial, but since our ratio is so large the cutoff voltage is only 0.6V below the nominal voltage out of the divider, so these small inaccuracies are amplified in some regard.

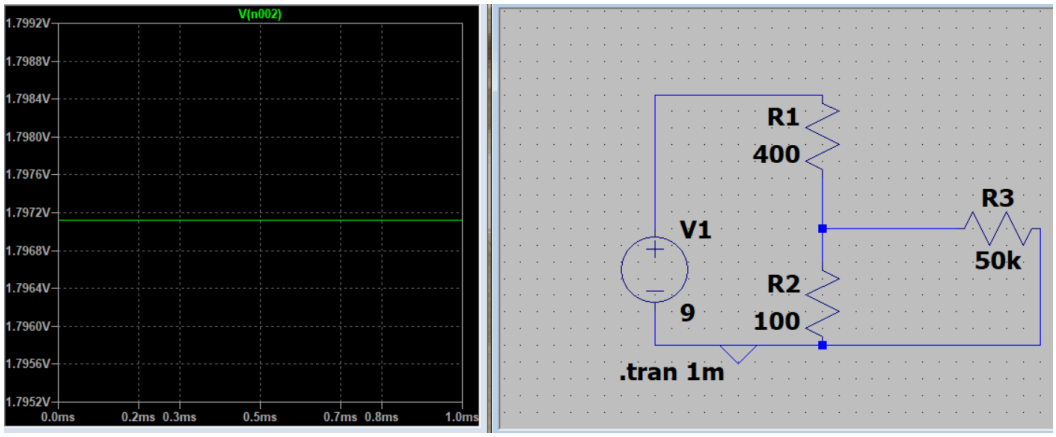


Figure 21: LT-Spice Simulation of Voltage Divider with 50kΩ input impedance

There is another simple solution to this, however, and that is to use an op-amp at unity gain to buffer the output. Since an op-amp has a very high input impedance, very little current will go to its input and the voltage would therefore be more accurate to the ratio given by the voltage divider. In this case our supply can either be the MCU supply or any other voltage supply on the board, but we chose the MCU's supply since this circuit will be closer to the MCU.

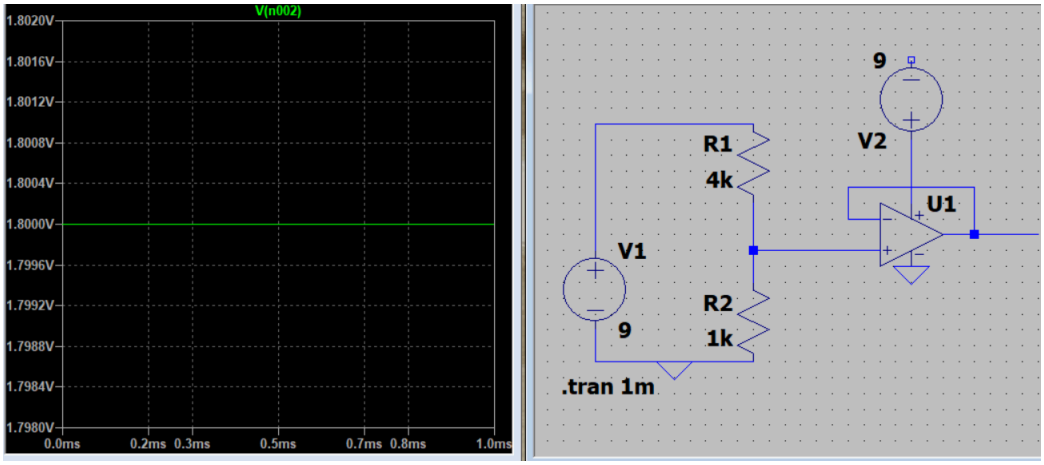


Figure 22: LT-Spice Simulation of Buffered Voltage Divider

In practice, we switched the values from 1kΩ and 4kΩ to 1MΩ and 3.9MΩ for two specific reasons. The first is that it's very difficult to find a resistor that is exactly 4kΩ . Secondly, the lower impedance on the input of the op amp resulted in more current consumption from the op amp. Raising this value to 1MΩ and 3.9MΩ fixed the issue, resulting in a current draw of only 1-2uA as opposed to 1.8mA when using the 1kΩ and 4kΩ resistors. Additionally a 6.8uF bulk capacitor along with a decoupling capacitor were added to the supply input of the op amp for stability. The TLV2401 ws chosen specifically due to its ultra-low supply current, making it a perfect buffer for the vent unit.

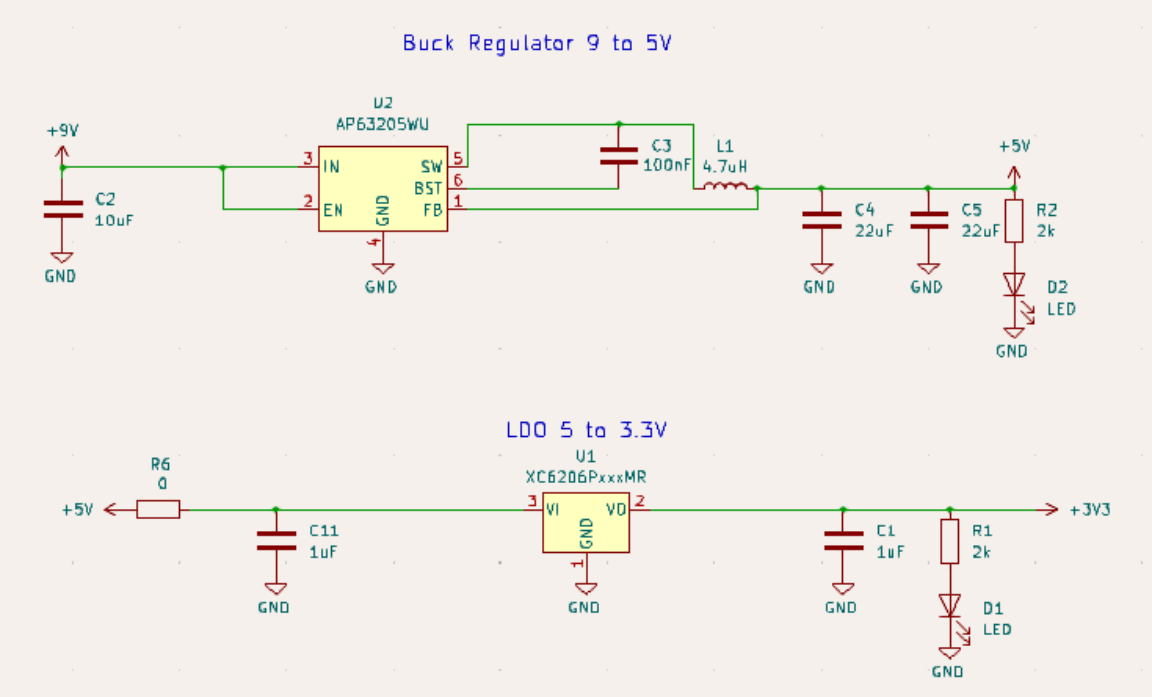


Figure 23: Vent Interaction Subsystem Power Supply Schematic

Buck-regulator: For the vent unit, the power supply design was quite simple. The AP63205 buck regulator datasheet was followed closely for our given output voltage, adding the necessary input capacitor, output capacitors, and bootstrap capacitor. The LDO placed in series with the buck regulator was also very simple, only requiring one 1uF capacitor on both the input and output. Both regulators feature a diode with a current limiting resistor on each output to make initial testing easier. These were removed when testing current consumption, however, since the diodes draw excess current.

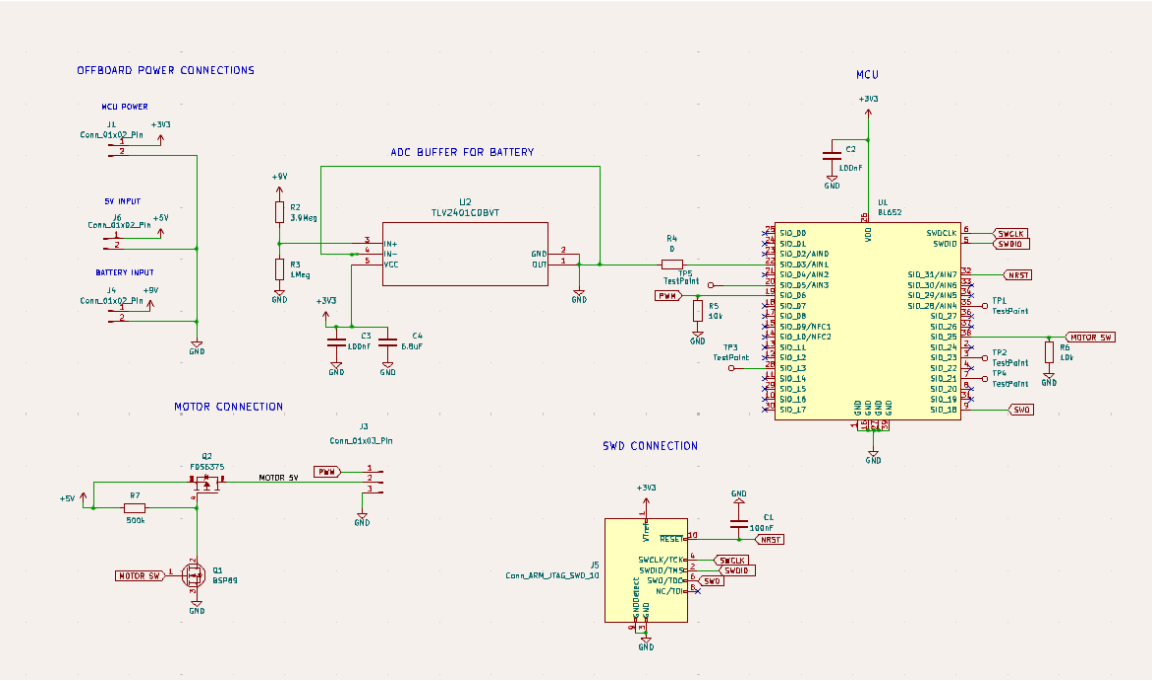


Figure 24: Vent Interaction Subsystem PCB Overall Schematic

7.0 Software

7.1 Setting up the Development Environment

To be able to upload code to the nRF58232 and the ESP32, the development environments must first be established. The ESP32 utilizes the Arduino IDE, and therefore it must be recognized via USB for code to be uploaded. The nRF52832 can use the Arduino IDE, but as mentioned in a previous section we are only using it for setting up a prototype. After this was finished, we needed to set up the Nordic SDK with Visual Studio Code. This required a connection with the J-Link EDU MINI, to allow code to be uploaded to the flash of the nRF52832 via SWD.

Configuring the pins is relatively straightforward in both development environments. If we want to set up an ADC, in the Arduino IDE we just simply configure the pin to input and then set up the input to read to a string. In the Nordic SDK the logic is similar where a GPIO pin is configured to input, but then the timers must be configured before it can be set to input. After these pins are configured, we can then implement the logic for each of the units.

7.2 Hardware Abstraction Layer

Similar to the STM32, the Arduino IDE and the Nordic SDK acts as a hardware abstraction layer. This allows for interaction with the hardware from the software generated from the Arduino IDE and the Nordic SDK. Once pins are configured and functions are called to act on the logic, the background logic allows for them to access the internal components of the MCU to work. This allows for us to interact with the peripherals including the servo motor, LCD screen, and temperature sensor.

The Hardware Abstraction Layer (HAL) is a crucial component of microcontroller development, providing a high-level interface for interacting with the hardware. It abstracts the complexity of register-level programming, allowing developers to configure and control peripherals using simple and intuitive APIs. This approach significantly reduces development time and lowers the barrier for engineers to work with advanced microcontrollers like the ESP32. Instead of needing detailed knowledge of the microcontroller's internal architecture, developers can focus on designing application-level logic, relying on HAL to handle low-level hardware interactions.

HAL is especially important for ensuring portability and scalability in embedded systems development. Since microcontrollers have a wide range of variants with different features and capabilities, HAL provides a unified API that works across multiple devices within the ESP32 family. This consistency allows developers to migrate code between different models with minimal changes, enabling seamless upgrades or modifications to meet evolving project requirements. Furthermore, HAL's modular design makes it easier to add new peripherals or update existing configurations, fostering flexibility and adaptability in system design.

Another critical advantage of HAL is its ability to streamline peripheral initialization and configuration. For example, setting up communication protocols such as I^2C, UART, or SPI requires configuring multiple registers, which can be error-prone and time-consuming. HAL simplifies this process by encapsulating these operations into pre-defined functions, ensuring correctness and reducing debugging efforts. Additionally, HAL supports advanced features like interrupt handling, DMA (Direct Memory Access), and low-power modes, making it an indispensable tool for maximizing the performance and efficiency of ESP32-based systems. By leveraging HAL, developers can focus on creating robust and feature-rich applications while minimizing the complexity of hardware management.

The system clock configuration establishes the timing foundation for the microcontroller, ensuring that all peripherals and internal processes run in sync and meet timing requirements. Peripheral clocks are then configured to provide the necessary clock signals to individual subsystems like I^2C, UART, and GPIO, enabling communication and functionality for specific components, such as the DHT20 sensor or the LCD module. The initialization of the HAL abstracts the low-level hardware details, providing a standardized interface for developers to access the microcontroller's features without delving into the intricacies of register-level programming.

These foundational configurations are universal across subsystems, making them a critical step regardless of the specific functionality being implemented. By standardizing the initialization process, the ESP32 ensures that each subsystem integrates seamlessly into the overall design, maintaining consistency and reliability throughout the development process.

7.3 Clock Configuration

The Arduino IDE and Nordic SDK is a powerful tool that simplifies the setup of ESP32 nRF52832 microcontrollers, including the critical task of clock configuration. Clock configuration is essential for determining the speed and timing of the microcontroller's operations, as it sets the frequencies for the CPU, memory, and peripherals. The Arduino IDE automatically generates the necessary initialization code based on user-defined settings, such as selecting internal or external oscillators (e.g., HSI, HSE, LSI, LSE) and configuring the Phase-Locked Loop (PLL) for frequency scaling. It also ensures proper configuration of clock sources for peripherals, allowing for precise synchronization and optimal performance. The Nordic SDK requires the configuration file to specify the clocks being used. Once these are declared, the logic for implementing these can be used through the internal functions provided by the Nordic SDK. For example, the Nordic SDK can set up the required internal RC clock, which is required when we do not have an external oscillator.. This automation saves significant development time by handling the complex relationships between system and peripheral clocks, ensuring that the configuration adheres to ESP32 and nRF52832 specifications and meets application-specific requirements. By leveraging the Arduino IDE and Nordic SDK, we can confidently integrate customized clock settings without the risk of manual configuration errors, enabling efficient and reliable system performance.

7.4 Temperature Sensor

The supported Arduino and Nordic SDK functions used for our functionality initializes the internal peripherals and system components that the ESP32 and nRF52832 needs for proper operation. When we call these functions, the development environments ensure that various peripherals such as I2C and SPIare directly driven by the clock, allowing for us to interact with the temperature sensor, LCD screen, and the CO_2 sensor.

Interacting with the serial monitor required for UART communication to be established. This is handled through the serial.begin function, where we are able to start the baud rate and change this in the serial monitor. This allowed for us to debug and edit the code accordingly.

The main program logic runs inside a sensor handling function, where the system initializes and begins continuous operation. An initial message is sent to the UART terminal using printf(), indicating that the system is starting the temperature reading process. Inside the infinite loop, the program continuously reads the temperature data from the DHT20 sensor. The read temperature function is called, which interacts with the DHT20 sensor via I^2C. In this application, the ESP32 microcontroller acts as the I^2C master, and the DHT20 sensor acts as the slave device with the address 0x48. The function begins by sending a start condition, followed by the 7-bit slave address (0x48) and a write bit (0). This combination signals the DHT20 that the microcontroller intends to write to the device. After receiving an acknowledgment (ACK) from the DHT20, the microcontroller sends the register pointer value, which specifies the internal register to access. In this case, the pointer is set to the temperature register, typically 0x00 or 0x01 depending on configuration.

After the register pointer is set, the function sends a repeated start condition, followed by the slave address (0x48) and a read bit (1), indicating that the microcontroller now wants to read data from the specified register. The DHT20 responds by sending two bytes of data. The first byte contains the most significant bits (MSB) of the temperature value, while the second byte contains the least significant bits (LSB). Once the two bytes are received, the microcontroller sends a stop condition, signaling the end of the I^2C transaction.

These two bytes are combined to form a 16-bit raw value, where the MSB represents the integer portion of the temperature, and the LSB represents the fractional portion. This raw value is then multiplied by the DHT20's resolution (0.0625°C per bit) to compute the temperature in Celsius. The precise use of start, repeated start, and stop conditions, along with the slave address and

register pointer, ensures accurate and reliable data communication between the ESP32 and the DHT20.

However, the handling of all the low-level operations is handled by the hardware specific DHT library, allowing all of it to be encapsulated into a simple read function.

The function reads two bytes of data from the sensor's temperature register and combines them into a single 16-bit raw value. This value is then converted to a temperature in Celsius by multiplying by 0.0625, as specified by the DHT20 datasheet. The temperature is then converted to Fahrenheit using the following formula:

$$F = \frac{9}{5}C + 32$$

Moreover, a delay of 1 second using the delay function is introduced between each reading to ensure the program does not overwhelm the sensor and to provide time for the sensor to stabilize between readings. The code includes several instances of error handling to ensure proper operation. For example, if the I2C communication fails to read the data from the DHT20 sensor, an error message is printed to the terminal, and the program returns a placeholder value (set to -273.15) to indicate a failure in the temperature reading. Additionally, the Error_Handler() function is invoked whenever a critical failure occurs, such as when the I2C initialization or UART initialization fails. Likewise, the Error_Handler() function enters an infinite loop, allowing the issue to be troubleshooted.

In summary, the software software for the ESP32 microcontroller reads temperature data from a DHT20 sensor over the I2C communication protocol and outputs the readings via UART to a terminal. It also includes appropriate error handling and peripheral initialization to ensure that the system runs smoothly and can handle unexpected situations. The code allows for easy communication with the DHT20 sensor and provides a stable loop for continuous temperature monitoring with periodic readings.

Below is a flowchart demonstrating how the software for the DHT20 sensor works.

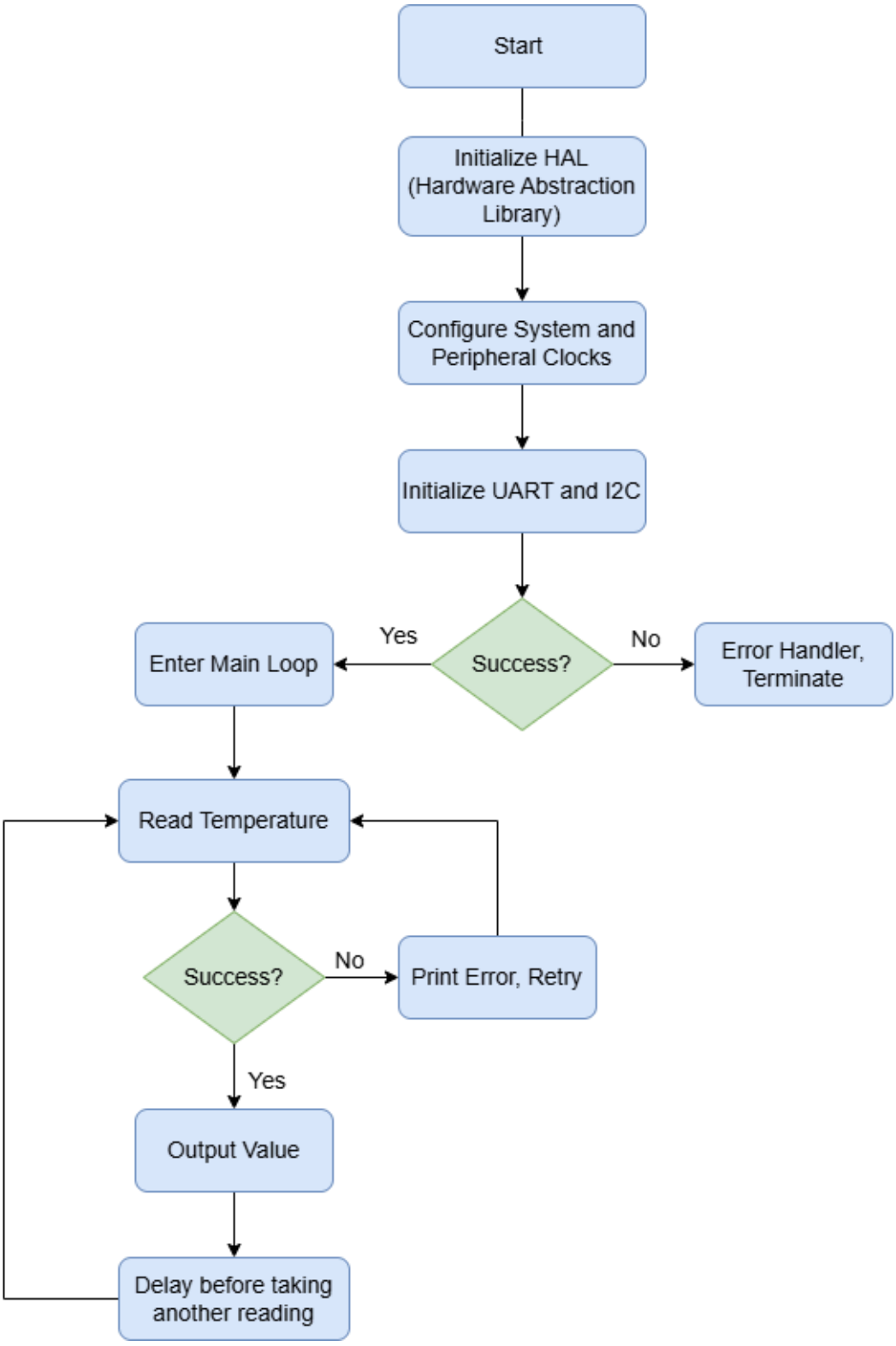


Figure 25: DHT20 Temperature Sensor Flowchart

7.5 CO2 Sensor

The software for the CO2 sensor utilizes the Sensirion SCD41 sensor to measure the concentration of carbon dioxide found within the environment. Since the SCD41 uses I2C as its communication method, it will be configured very similarly to the temperature sensor. The process begins by initializing various libraries, the wire library and a SCD41 specific library.

The delay was chosen to be 5 seconds since the SCD41 sensor requires sufficient time to perform its measurement and transmit the data to the ESP32 microcontroller. The SCD41 operates in periodic measurement modes, where it requires several milliseconds to process environmental data and generate an accurate reading. Additionally, the sensor takes time to stabilize and respond to read requests, especially in conditions where environmental parameters are changing. Given the measurement cycle times and the sensor's processing requirements, a 5 second delay ensured that the sensor had enough time to complete its internal calculations, transmit the data reliably, and avoid any overlap or communication errors with subsequent readings. This delay helps maintain communication integrity and prevents the system from overwhelming the sensor with too many requests, allowing for efficient data acquisition.

After the delay, the function retrieves the raw measurement data from the sensor. This data, received as two bytes representing the raw CO_2 concentration, is processed into a human-readable value in ppm by combining the bytes into a 16-bit integer and converting it to a floating-point number. Similar to the DHT20, error handling measures were implemented. If a communication failure occurred during the transmission or reception of data, the function prints an error message using printf and returns an invalid value to signal the failure. This allows for any errors to be quickly identified.

The subsystem operates in a continuous loop within the main function, repeatedly calling the read function to obtain updated CO_2 readings. The results are transmitted over UART for display in the serial console, providing real-time monitoring. If an error is detected, an appropriate message is displayed to notify the user. This continuous operation ensures consistent air quality monitoring and enables integration with systems for automated responses, such as ventilation control or safety alerts. By leveraging these functions, the CO_2 sensor effectively fulfills its role in maintaining safe indoor air quality and contributing to occupant health and safety. Like the DHT20, all of the low level operations described are also abstracted into a simple read function.

Below is a flowchart demonstrating how the software for the SCD41 sensor works.

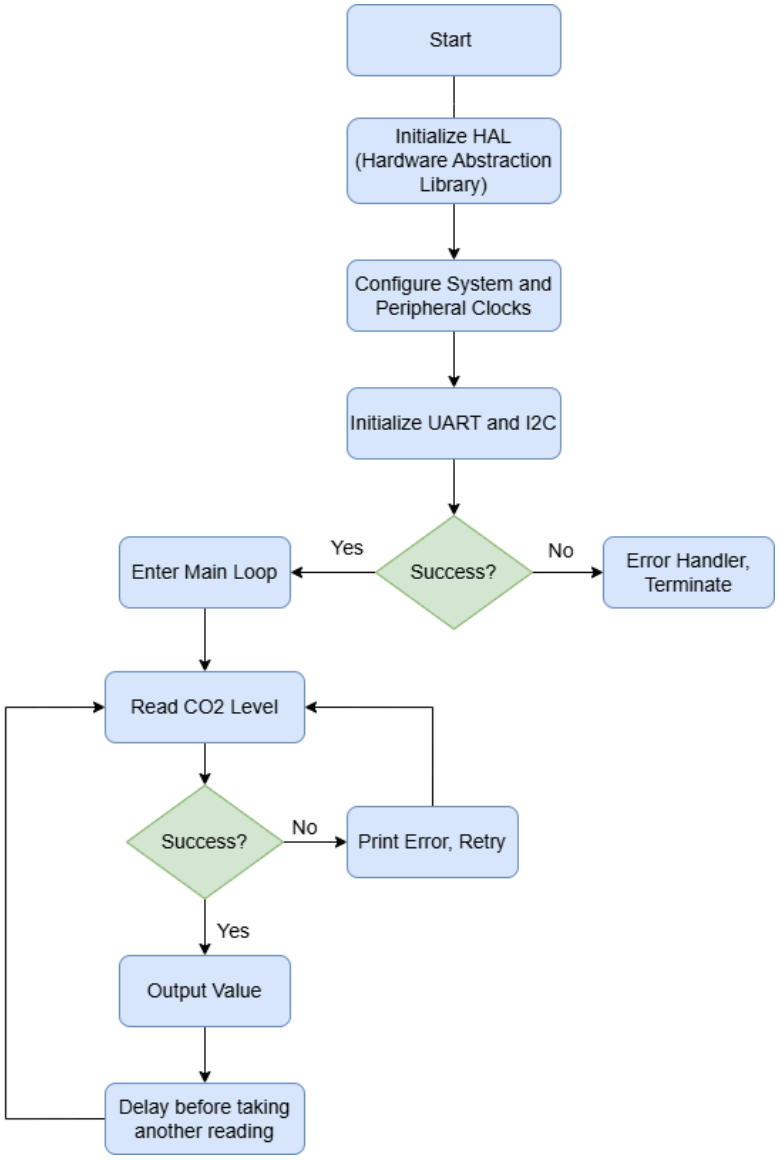


Figure 26: SCD41 CO2 Sensor Flowchart

7.6 BLE Testing

Bluetooth Low Energy (BLE) communication requires a robust and precise hardware setup to ensure reliable data transmission and reception. To test the BLE capabilities of the ESP32 and the nRF52832, two separate software were created in order to implement a robust Bluetooth Low Energy (BLE) communication system, with one functioning as the BLE server and the other as

the BLE client. We implemented the same UUIDs on both MCUs, allowing for them to be connected. This was as simple as implementing a string to the codes of each when using the Arduino IDE, however we found there were a few changes that needed to be implemented when looking at the Nordic SDK. First, we had to break up the string into an array of bytes in little endian format. After this was corrected, we were able to notice the UUID on the Nordic SDK app. Once this was verified to work, we found that the characteristic UUID had to be implemented differently. On the Nordic SDK, it takes in a 4 byte string which is implemented in the middle of the UUID. This made us change the UUIDs on both units to support this implementation, and this was found to be effective. s.

The server code initializes the system and configures the RADIO and RADIO_TIMER peripherals to operate as a BLE peripheral. It continuously advertises its availability to nearby BLE clients, broadcasting essential information such as its UUID and connection parameters. This setup allows a BLE client to discover the server and establish a connection. The server also processes data requests from the client and manages responses, acting as the provider of information or services. The necessary functions ensure the RADIO peripheral is prepared for transmitting and receiving BLE packets, while precise timing to align with BLE's strict timing protocols. These components enable the server to operate as a reliable and responsive BLE device.

The client code, on the other hand, is designed to initiate communication with the server. It begins by scanning for available servers broadcasting their advertisements. Once it discovers a server that matches its predefined criteria, the client establishes a connection. The functions in the client code mirror those in the server code, ensuring the client's hardware is similarly configured for BLE operations. Once connected, the client can exchange data with the server, either by sending requests for information or subscribing to notifications. The client also plays a key role in managing the BLE connection, including determining the connection interval and ensuring efficient communication.

Together, the server and client codes implement a full BLE communication system. The server continuously advertises and awaits connections, while the client actively scans for available servers and manages the connection process. Both codes utilize the clock peripherals to ensure reliable packet transmission and precise timing synchronization, which are fundamental to BLE's low-power, high-efficiency communication. By working in tandem, the two codes enable the seamless exchange of data, making this system suitable for a wide range of wireless communication applications.

More specifically, BREATHE will be using BLE to communicate between the control and the vent interaction subsystem, where environmental data and user input can be transmitted from the control MCU to the vent MCU in order to dictate the vent's behavior.

Below is a software diagram showcasing how the software responsible for BLE testing between two microcontrollers works.

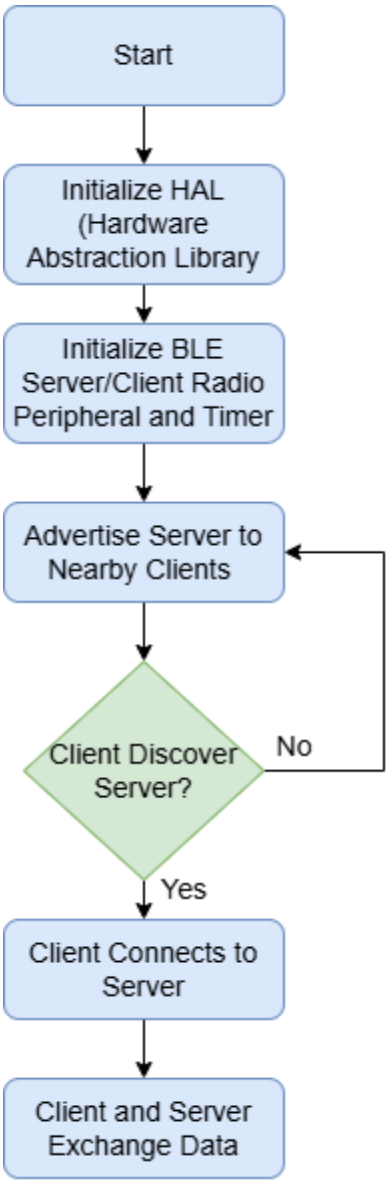


Figure 27: BLE Testing Flowchart

7.7 Buzzer

The electromagnetic buzzer acts by hooking up one terminal to a PWM signal from the MCU, and the other terminal to ground. The duty cycle of the PWM can cause the buzzer to turn on or off, and the frequency of the signal can affect the tone of the buzzer. The PWM signal is achieved by assigning the output pin to a timer, which is any of the analog or supported digital pins on the ESP32. Under the hood, the functions in the Arduino IDE set the mode to countermode up, which means the timer increments up until a specific value is met before resetting back to 0 and causing an event to occur, which in this case is turning the timer on or off. The volume of the buzzer is dependent on the amplitude of the waveform. Although this is not adjustable through the ESP32 by itself, the default sound through the development board is sufficient.

When implemented in the final design, the MCU will be waiting for a signal from the control unit to be transmitted via bluetooth. When it receives the signal, it'll store the message in a buffer and interpret it. The incoming message will either respond with a power alert or a CO2 alert. If the MCU receives a CO2 alert, it will send a PWM signal with a very high amplitude. This will be achieved through interfacing with the PSU through an additional unit such as an Op amp. For the power alert, the unit will send a PWM signal with a lower amplitude, which would utilize the same initial logic without additional interfacing with the PSU to adjust the signal.

Below is a diagram describing how the buzzer will be used.

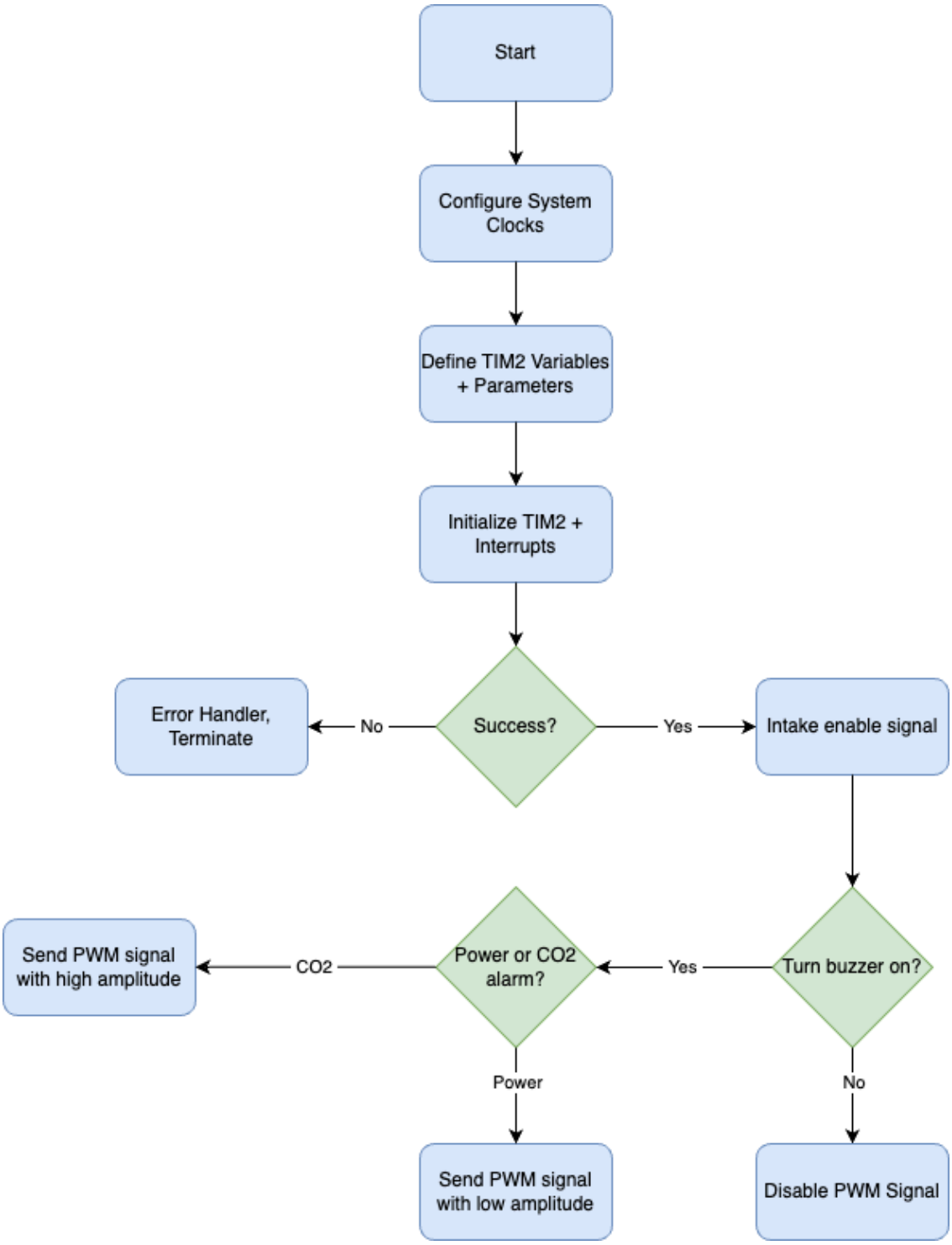


Figure 28: Buzzer Flowchart

7.8 Motor

BREATHE will be utilizing a servo motor to interface with the vent. To manipulate the actions of the motor, the MCU will be utilizing similar logic and have the same initial setup as the buzzer. Servo motors intake a PWM signal and adjust the angle of the motor's rod according to the pulse width. The nRF52832 will use an initialized AIN2 to enable PWM output, and then hook it up to a pin.This will then allow the MCU to adjust the pulse width being sent to the motor, allowing us to adjust the angle of louvers.

Every servo motor has a unique period that is required for enabling movement. For the Futaba S3004, a period of 50hz is required. By initializing this when setting up the Timer 2 clock in the appropriate function, we are able to interact with the motor and receive the desired movements.

Below is a diagram showcasing the logic for initializing and utilizing the motor.

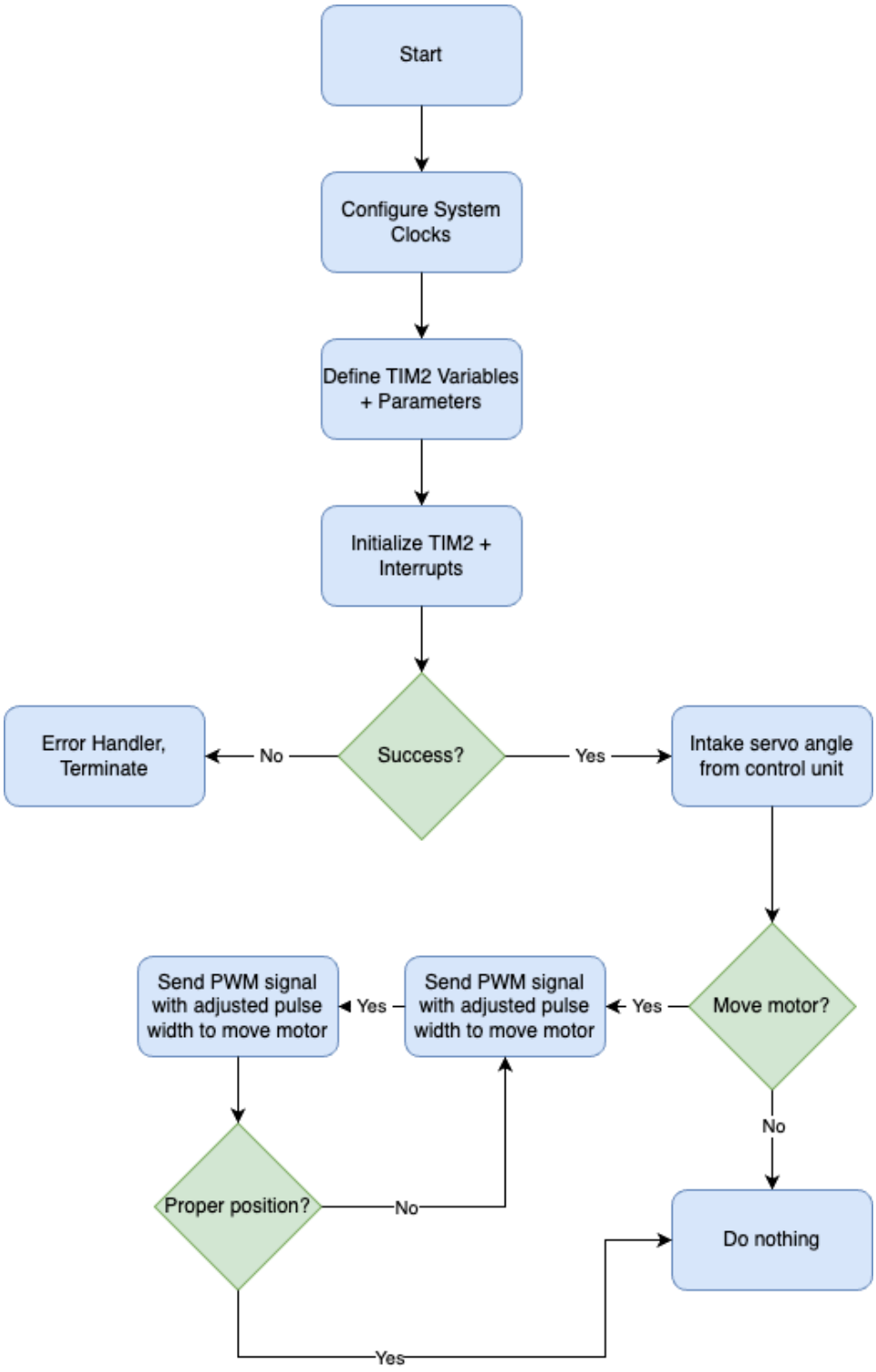


Figure 29: Motor Flowchart

7.9 User Interface System

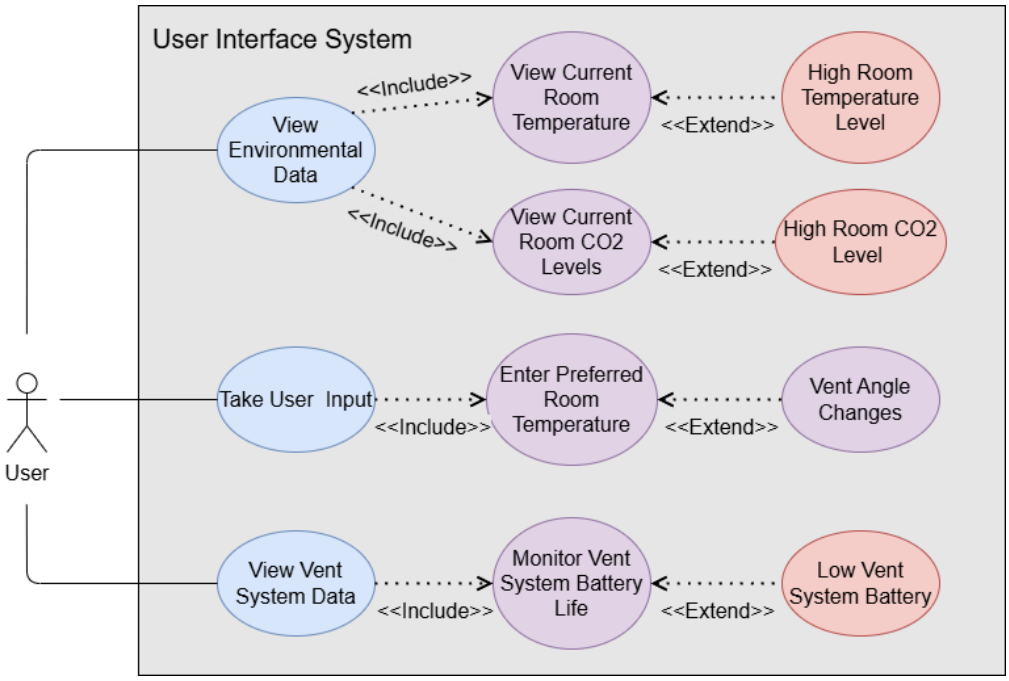


Figure 30: User Interface Use Case Diagram

As displayed above, the Use Case Diagram for the Smart Vent System outlines the interactions between the user and the system, as well as the system's response to user input. The diagram centers around the user, who interacts with the system to monitor environmental data and provide preferences for room conditions. The system allows the user to view environmental data, including the current room temperature and CO2 levels, and it provides the ability to determine if these conditions exceed acceptable thresholds. Additionally, the User can take input, such as entering their preferred room temperature, which directly influences the system's behavior.

Based on the user's input, the system adjusts the vent angle to maintain the desired room conditions. This is represented in the diagram by the "Vent Angle Changes" use case, which is triggered by the user's preferences or by the environmental data such as high room temperature or CO2 levels. If the user inputs a preferred temperature, the system adjusts the vent to help maintain that temperature. The use of the <<extend>> relationship indicates that the vent angle change is contingent upon specific conditions, such as the user's input or when room temperature exceeds a predefined threshold.

In addition to the user-driven features, the system also monitors its battery life and alerts the user if the battery level is low, ensuring continuous operation. The "Monitor Vent System Battery Life" use case is linked to a "Low Vent System Battery" notification, signaling the user when the system's battery is nearing depletion. This provides the user with feedback on system health, enabling timely intervention if needed. These functionalities ensure that the system operates reliably and provides the user with an intuitive, interactive experience.

It is important to note that the system retains autonomous functionality. Even though the user can input preferences, the Smart Vent System is still capable of operating automatically in the background to maintain optimal conditions. If the user does not provide specific input, or if the system detects that conditions like room temperature or CO2 levels are beyond acceptable limits, the vent adjusts automatically to bring the room back to comfortable levels. The autonomous features are seamlessly integrated, and the system will only omit this automatic behavior if the user explicitly chooses to rely solely on manual control. This flexibility allows the user to balance automation with manual preference, making the system adaptable to different needs.

In summary, this use case diagram effectively captures the various components of the Smart Vent System, highlighting both user-driven actions and system responses to user input. It shows how the system responds to environmental data and user preferences to maintain optimal room conditions, making the system both responsive and user-friendly.

Within software, the TFT_eSPI graphics library was leveraged to create B.R.E.A.T.H.E's GUI. Thankfully, TFT_eSPI offered many built in functions that streamlined development and helped translate initial prototypes for B.R.E.A.T.H.E's GUI from Figma to the actual LCD screen. Data from the sensors is displayed to the screen and is updated whenever new data is polled. Moreover, buttons for each operation, such as increasing and decreasing the user-set preferred temperature as well as toggling between autonomous and manual mode were created using various shape rendering functions. These buttons were also given touch screen capabilities. TFT_eSPI allows for this by simply defining an area's x and y coordinates and using its built-in tft.getTouch function. Moreover, custom fonts were possible by converting downloaded fonts into a hexadecimal array, and then simply placing that array into its own unique header file.

7.10 Additional Software Design Considerations

7.10.1 Manually Implementing Floating Point Numbers In Nordic SDK

Initially, the Nordic microcontroller did not support floating-point operations, which presented a significant challenge for implementing the BLE communication system, particularly for sending ADC values for the battery life.. The ADC outputs its measurements in a float format, However, when it was implemented, we found that we could not transmit floats, nor could we easily convert them into a string.

To address this, it was necessary to manually implement functions to convert the floating point numbers into strings. This was achieved by declaring a buffer string, then going through the first few decimals of the float, followed by the whole number, then reversing it to have it as the desired format. Once the float was converted into a string successfully, we were able to transmit this to the ESP32 as desired.

7.10.2 Clearing Flash

During the development and testing phases of the Smart Vent System, clearing the flash memory periodically was a crucial step to ensure reliable operation. Flash memory in embedded systems like the nRF52832is used to store firmware, calibration data, and other configuration settings. As part of the iterative development process, frequent updates to the code and settings meant that reprogramming the device and ensuring the flash memory was cleared of outdated or incorrect data was necessary. Failure to do so could result in conflicts between old and new code, causing unpredictable behavior or system crashes. This was particularly important for ensuring that new firmware versions would not interact with residual data from previous versions, which could lead to errors or faulty sensor readings.

Clearing the flash memory was essential for several reasons. First, it helped avoid data corruption, as old data from previous firmware versions could remain in the flash memory unless explicitly erased. This could lead to conflicts with the new code, especially if the old data was incompatible with the new version. Second, clearing the flash ensured that the system always started with a clean slate each time new firmware was loaded, which helped maintain consistency and prevent issues from leftover data. Third, periodic clearing helped prevent flash wear, which is important when frequently updating firmware, even though modern flash memory has wear-leveling mechanisms to mitigate excessive writes. Finally, during development and debugging, clearing the flash allowed

the team to test new code versions without the risk of residual data affecting the system's behavior.

To manage this process, the nrfjprog software used with the J-link EDU Mini was used. This software tool, part of the Nordic tools, provided a reliable interface for programming, erasing, and reading the flash memory of nRF52832 microcontrollers. The nrfjprog allows the user to connect to the nRF52832 microcontroller via a ST-Link programmer or USB-to-serial interface, depending on the hardware configuration. Once connected, the tool enabled the erasure of the flash memory, either entirely or by erasing specific sectors that needed to be cleared. This was a critical part of the reprogramming process, ensuring that outdated or corrupt data did not interfere with the new firmware.

After the flash was cleared, new firmware was flashed onto the microcontroller using the Segger Embedded Studio. The software also provided a verification process to ensure that the new code was written correctly and without errors. During debugging sessions, Segger Embedded Studio allowed the development team to monitor the system's behavior in real time, checking for any issues that might arise due to flash corruption or residual data. This ensured that each firmware iteration was installed properly and that the system was ready for further testing or deployment. Conclusively, the ability to clear and reprogram the flash memory using nrfjprog played an important role in maintaining the integrity of the development process and ensuring that the Smart Vent System operated as expected.

7.10.3 Memory Management for the Control Unit

Memory management on the Control Unit wasn't anticipated to be a concern, especially due to the fact that the Control Unit's partition scheme was set to Huge APP, theoretically giving B.R.E.A.T.H.E more flexibility in terms of memory constraints. However, at the end of B.R.E.A.T.H.E's development, it unfortunately became a problematic occurrence. Notably, due to the majority of the ESP32's heap being consumed, there became problems with rendering graphical elements onto the screen, with some sprites being pushed to the top left of the screen and other elements disappearing altogether. An initial solution to this drawback was to create separate classes for each sprite. This proposed solution did work initially, causing the re-rendering of elements of the top left of the screen to disappear, however it was not anticipated that establishing a BLE connection between the control unit and the vent unit was a significant contender in using heap space. Moreover, given the constraints of using object

oriented programming in an embedded system, unique classes were found to consume significant portions of memory. To address these concerns, the color depth for all sprites was set to 8 rather than 16. This was a massive revelation, as even after an established BLE connection and post-rendering of all sprites, B.R.E.A.T.H.E's control unit heap space increased nearly by 70 KB. Likewise, by putting all sprites into one single class like how it was initially, an additional 20 KB of heap space was saved.

8.0 System Fabrication and Prototype Construction

When designing the PCBs for our project we understood that we would need to separate it into two separate PCBs, one for the Control Unit and one for the Vent Interaction Subsystem. In doing so there are certain key aspects one must take into account before making their PCB layout. For starters a detailed system schematic is crucial for PCB design and since we have detailed our process for the schematic designs in chapter 6 by using KiCad the next step is to begin the boards layout process.

First, we needed to decide on which company would fabricate and ship our PCBs as this will decide the board stackup and the design constraints. The stackups and design constraints are typically unique to each fabrication company. We chose to use JLCPCB as they are widely used, have reasonable pricing, and have decent quality control. We also chose to use a 4 layer board as it doesn't add too much cost and will make routing easier. This is also important for our buck regulator as the layout examples utilize a 4-layer board and straying away from this may not be ideal [125].

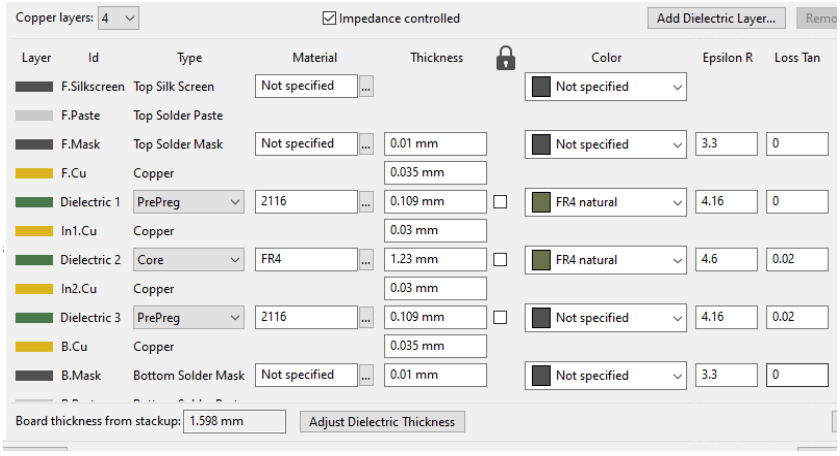


Figure 31: KiCad Board Stackup

JLC PCB also has specific design constraints that can be incorporated into KiCad which will be useful when running the final DRC check to show the constraints are being met [126]. These design constraints need to be met so JLC PCB can also properly fabricate the board. These standards were incorporated into our design with a little extra slack above the minimum design constraints to reduce cost a little since using purely the minimum constraints would add to the cost of fabrication.

Once the setup of our board layout is complete we can begin with organizing components. At first when you open the PCB editor all of the components you used in your schematic editor sheet are imported and clustered together and organized by component type not connections. A good rule of thumb for organization sake is to select the specific schematic you'd like to work on in the schematic editor which will highlight all the components pertaining to that part in the PCB editor and then you can move them to a less congested area to begin the layout. The best practice when it comes to the PCB layout is to start with the MCU since it is the most critical component and then move on to organizing the decoupling capacitors as close to their MCU pin connections as feasible. Once the layout is organized per connection and schematic guidelines, you can then begin to route your connecting components together. First we needed to establish our two inner layers to be dedicated ground planes and the bottom layer to be used for routing vias and power.

8.1 Control Unit PCB

Since the Control unit consists of the most peripherals it was by far the most challenging PCB to design. Following the steps explained in the introduction above is the majority of how the layout design came from. However, since the placement of the peripherals was also something to consider in terms of designing it added an extra step to consider. Since the main visual aspect of the control unit consists of the LCD screen it was important to consider that the housing unit size and PCB for this unit were proportional to the screen and didn't look outlandish. The LCD screen is 95mm by 62mm and the PCB behind it could only be at most about 50mm larger. The actual dimensions of the designed pcb are 137mm by 113mm and thus we successfully designed within the bounds of our specifications. As far as the peripheral component placements went, the buzzer was chosen to be in a higher configuration on the board so as to be able to project the sound at an altitude closer to the user's ears. Next would be the temperature and carbon dioxide sensor which were chosen to be placed next to each other on the right side of the board to not

interfere with the buzzer or power supply sections. A component layout and 3D view of the control unit PCB can be seen below in Figures 32 and 33.

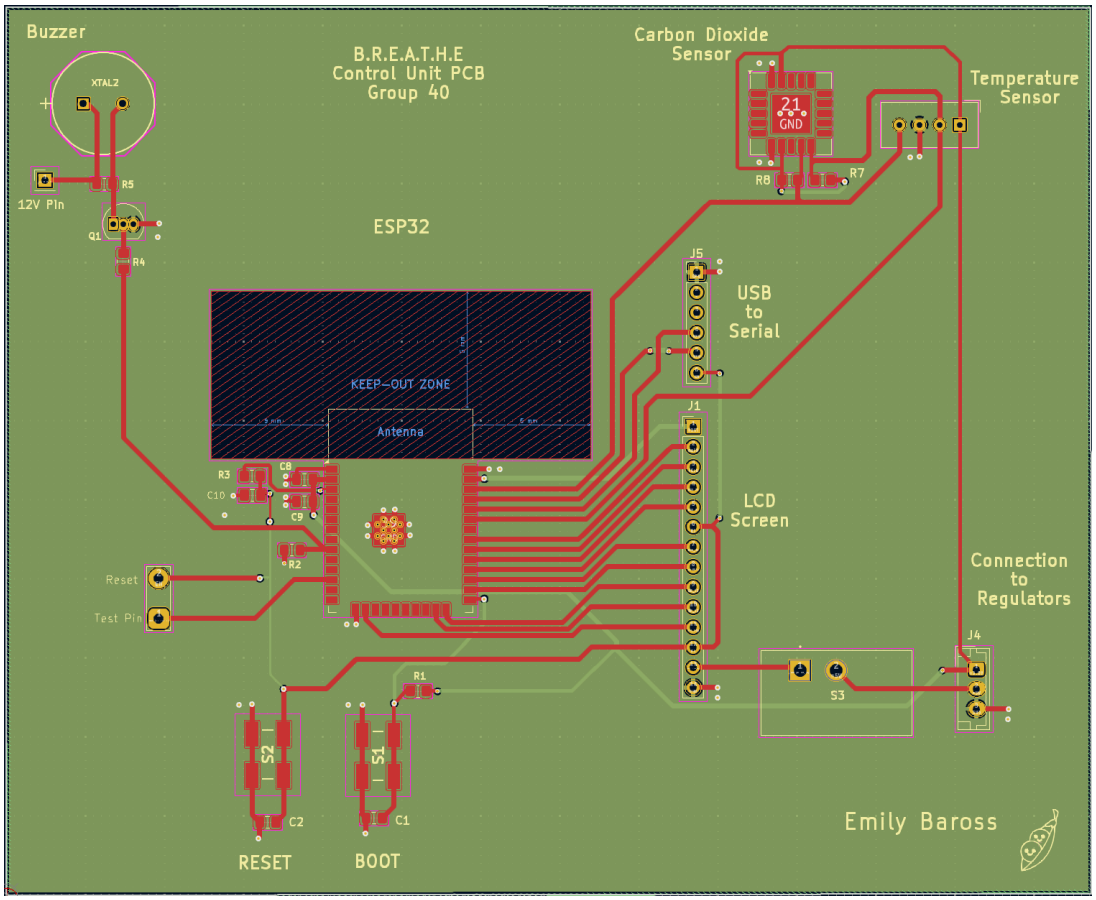


Figure 32: Control unit PCB Layout

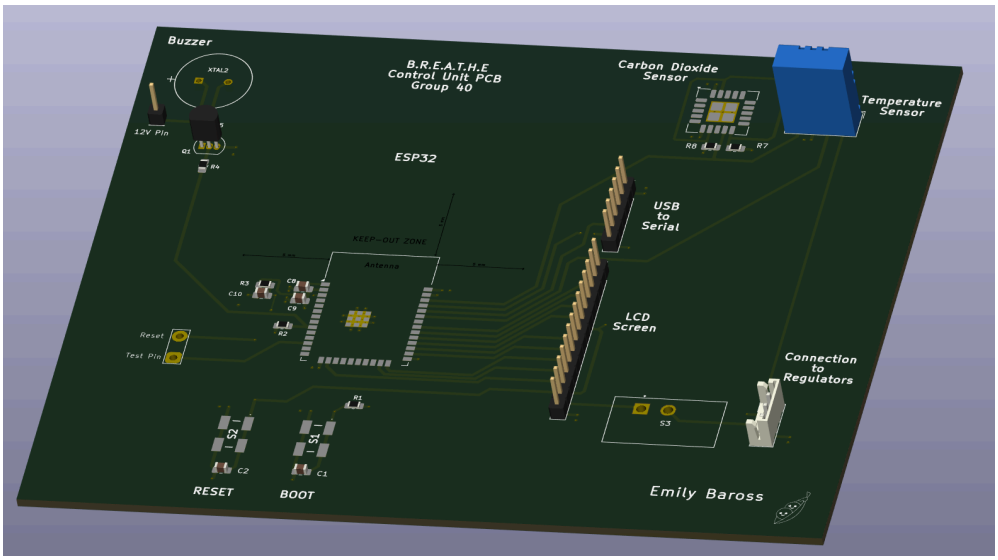


Figure 33: 3D Model of Control Unit PCB

8.1.1 Control Unit Power Supply PCB

For the control unit, we opted to utilize a 2-layer board stackup as it would reduce cost and the LM2576 regulator is known to be insensitive to board layout, so the benefits of a 4-layer stackup were unnecessary in this design. The top copper layer is used to route signals while our bottom layer is a dedicated ground layer. Large copper ground pours on the top layer were used whenever applicable to provide a stable ground and some thermal resistance. Stitching ground vias were also liberally placed across these pours as well to reinforce this. We also made sure to use a short and wide copper pour to connect the output node to the inductor on the board to keep the current loop relatively small and reduce any parasitic inductance. We used mostly through-hole capacitors and a large through-hole inductor partly for simplicity and also because it was easier to find through-hole electrolytic capacitors as opposed to SMD solid-state ones. The inductor value we needed was also easier to find in a through-hole package. Finally, the 5V and 3.3V rails are routed to a 3-pin JST connector which connects to the input of the main control unit board.

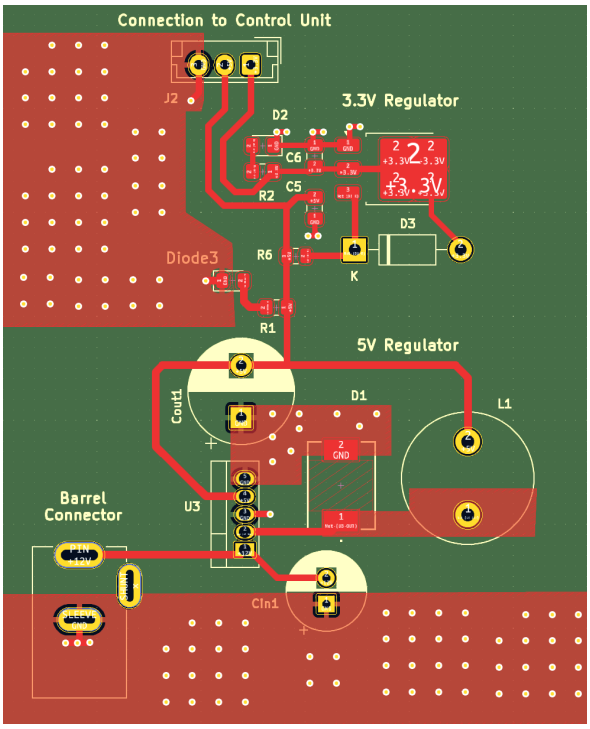


Figure 34: Control Unit Power Supply PCB

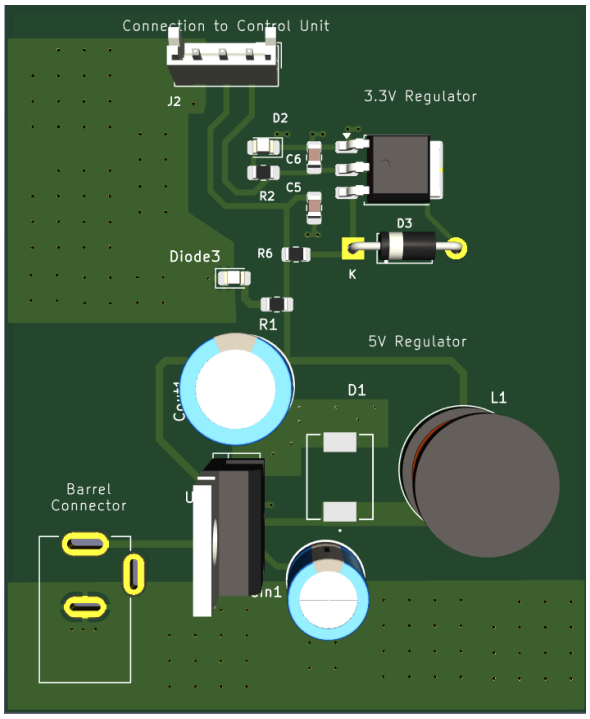


Figure 35: 3D Model of Control Unit PCB

8.2 Vent Interaction Subsystem PCB

Comparatively to the Control Unit PCB, the vent interaction subsystem had much fewer peripherals, and luckily the MCU also did not require a complex layout. Decoupling capacitors were placed closely to the MCU whilst bulk capacitors were generally placed more liberally. GPIO test points were routed as solder pads near the edge of the board. The MCU itself was placed with its antenna close to the edge of the board to maximize BLE range, and an accompanying no-copper zone was placed in the antenna area as specified by the manufacturer. Ground vias were also placed under the MCU to dissipate heat. The MOSFETS are connected to a pin header which connects to the motor off the board. One thing of note is that the footprint is for a right angle pin header which was actually a mistake. JST connectors are marked out to provide the 3.3V and 5V rail as well as the 9V battery input for the ADC. This PCB utilizes a 4-layer board stackup in order to provide low inductance ground connections on the two inner layers. The bottom layer is purely used as a signal layer.

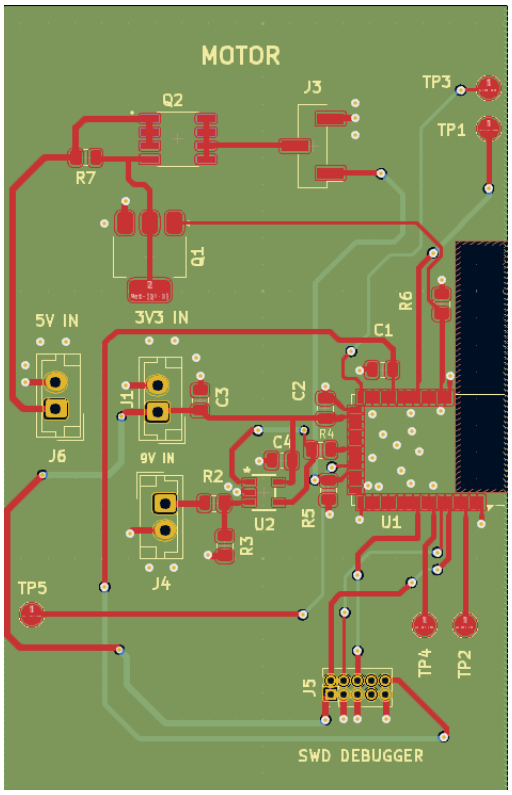


Figure 36: Vent Interaction Subsystem PCB Layout

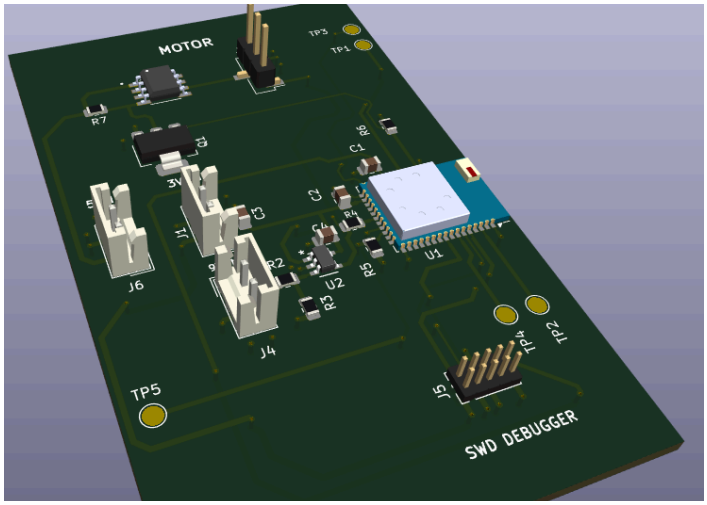


Figure 37: 3D Model of Vent Interaction Subsystem PCB

8.2.1 Vent Interaction Subsystem Power Supply PCB

Similarly to the control unit, the vent interaction subsystem also houses its power supply on a separate PCB. This PCB utilizes the AP63205 buck regulator and the XC6206 LDO to provide the 5V and 3.3V rails to the main unit through

JST connectors. The battery is connected to the power supply board using two solder pads which will route out to a connector offboard. Additionally, there is a JST connector to route out the 9V battery to the main board to be supplied to the ADC on the vent unit's MCU.

In the same fashion as the main vent PCB, a 4-layer board stackup was used with the two inner copper layers being designated ground layers. The input capacitor of the AP63205 was placed relatively close, and we made sure to make the copper pour going from the SW pin to the inductor as wide and short as possible. Large top copper pours were used for ground connections to help with thermals and provide a more stable ground connection.

All of the precautions described in the PCB design section of the regulator's datasheet were followed as closely as possible and traces were made to be 0.6-0.75mm as well to ensure the regulator's maximum current could be supplied to the outputs. [66].

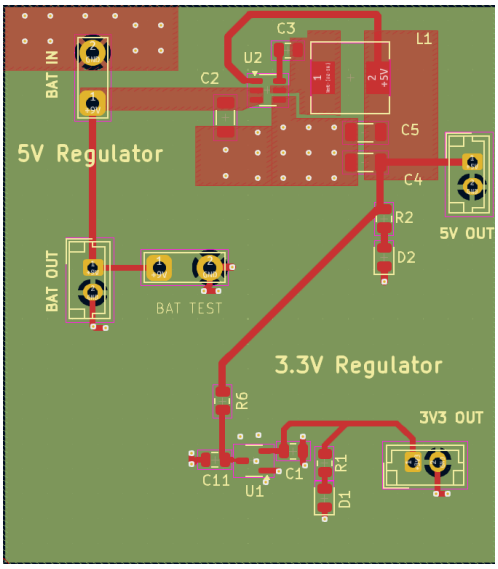


Figure 38: Vent Interaction Subsystem Power Supply PCB

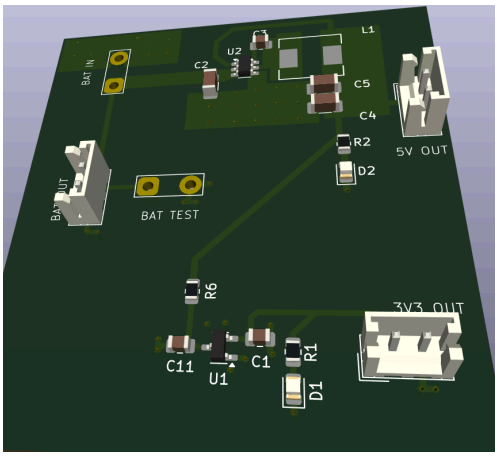


Figure 39: 3D Model of Vent Interaction Subsystem Power Supply PCB

8.3 Prototype Construction

8.3.1 Control unit

For the control unit, the goal is to create a simplified box that can be mounted onto a wall at eye level for the user where they can easily access the LCD screen to make temperature adjustments needed without having to open an enclosure to access the screen. A 3D prototype visual can be seen in Figure 40 of this enclosure. The lid of the enclosure will have the LCD screen and 3 drilled holes in front of the sensors and buzzer. These holes will act as an opening for the sensors to get more accurate readings of the environment's temperature and carbon dioxide levels as well as an opening for the buzzer's sound to more easily escape from the enclosure. We ultimately decided to 3D print this enclosure and the final design can be seen in Figure 41.

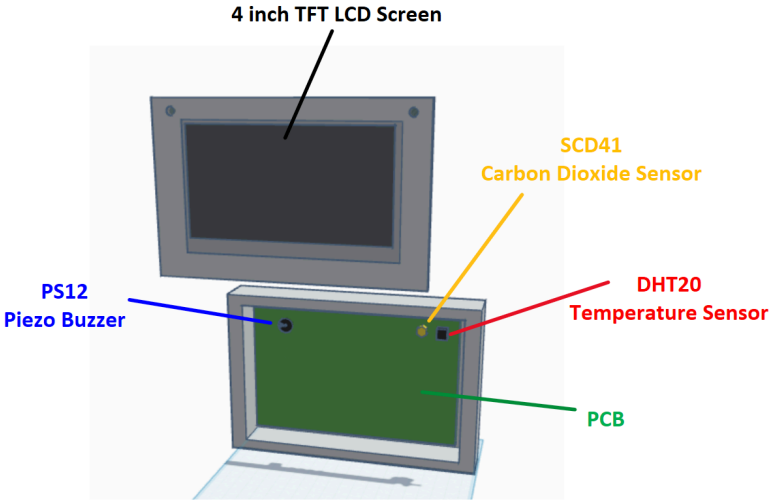


Figure 40: Prototype design for Control Unit

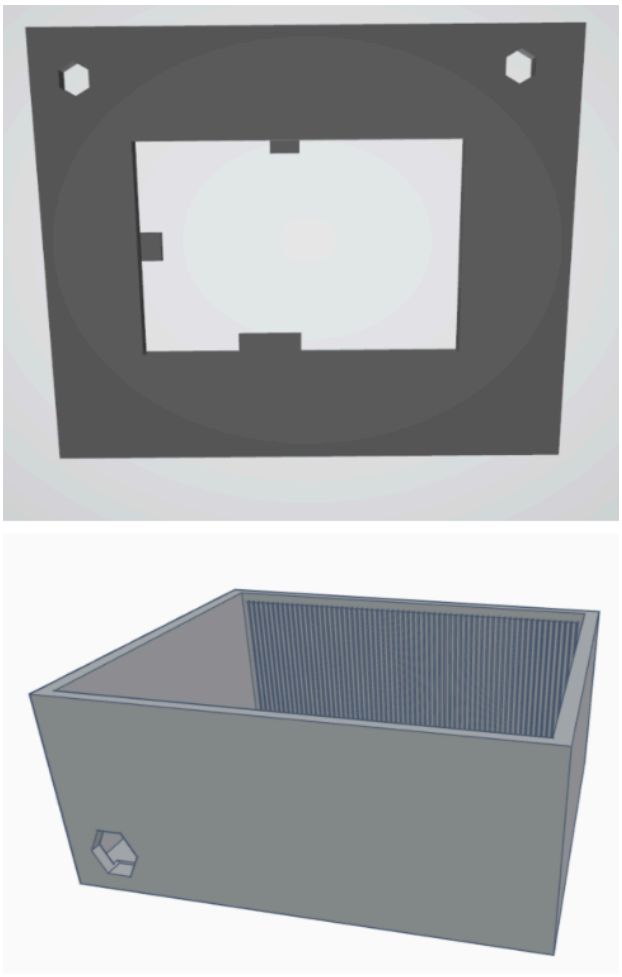


Figure 41: Final design for Control Unit

8.3.2 Vent Interaction Subsystem

For the Vent Interaction Subsystem, the main goal was for this to be compact enough to fit inside of the ceiling duct with our vent. In Figure 42, you can see our prototype idea for how the components will fit in the vent. This view is from the back side of the vent where the louvers are located with our components located beside them. The 9V battery, servo motor, and PCB are all closely placed on one side of the louvers thus showing how important a compact design is. We used a pre-existing vent anyone can purchase from a home improvement store and attached our components via a 3D printed design that was completely custom to what we needed and it worked most efficiently. In Figure 43, the final 3D printed design can be seen.

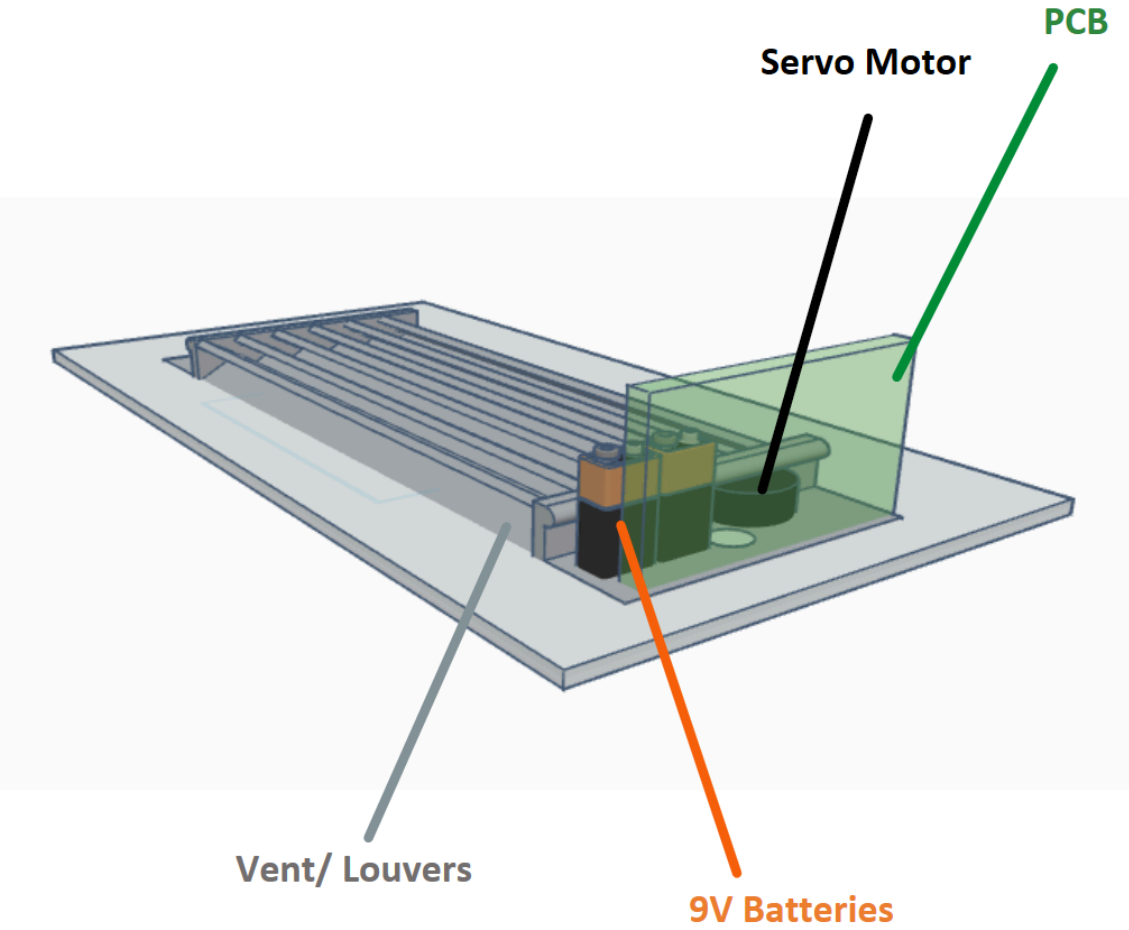


Figure 42: Prototype design for Vent Interactive Subsystem

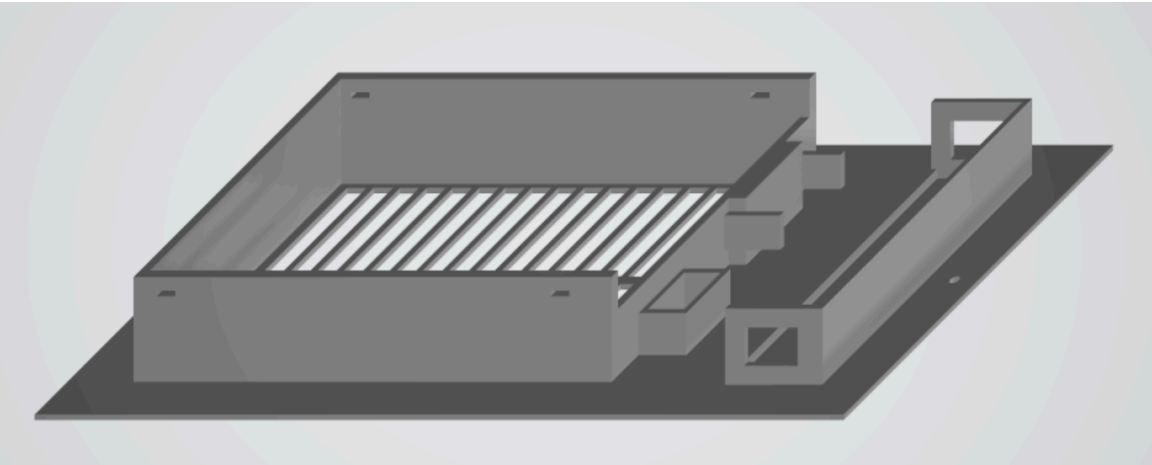


Figure 43: Final design for Vent Interactive Subsystem

9.0 System Testing and Evaluation

9.1 Intro

There's a certain level of programming to go along with each of our hardware components which in a way intertwines both components of testing. Both hardware and software required thorough testing to ensure the system functions as intended. While hardware testing involved tasks like verifying connections, measuring voltages, and checking signal integrity, developing the software for each hardware component was equally essential to ensure seamless integration. This included managing communication protocols such as I2C or SPI and ensuring accurate data flow between devices. Together, these processes allow us to identify and resolve issues more effectively, as problems in functionality can arise from either hardware faults or software misconfigurations. This combined approach ensures that both the hardware and software aspects of the system work seamlessly in unison.

9.2 ESP32 / nRF52832 Software Testing

In order to test the ESP32 and nRF52832, we used the ESP32-WROOM and Adafruit nRF52 development boards and utilized the Arduino IDE and Nordic SDK to generate code. We were able to create and test code while referencing the datasheet and the sample code from the IDE. First we downloaded the Arduino IDE and the Nordic SDK. From here we were able to download the necessary firmware to our board. We then referenced the datasheet for the ESP32 and nRF52832 to see the ports and their corresponding pins. After creating a sample ESP32 and nRF52832 project we used the Hardware Abstraction Layer (HAL) to generate the necessary code to configure the registers for the GPIO pins, and enable the port. First we went to the project's .ioc file to open the diagram with the chip. We then configured the chip to enable output for a GPIO pin and toggled the pin every 500 milliseconds. After hooking up the pin to an LED we were confident that we could interface with the pins on the board, and that the board could perform basic functionality.

In order to perform tests on the development boards, we utilized the Nordic SDK and Arduino IDE to develop software. Despite the small documentation for the board, creating and testing code was possible through referencing both the datasheet and sample code provided from the Nordic SDK, as well as example code from the libraries in the Arduino IDE. We broke down our goals into smaller subgoals and aimed to work our way up. For example, to test the temperature

sensor we first had to ensure that our board could run basic code and print to a terminal before looking into retrieving data from the sensor.

9.3 Bluetooth Testing

In order to test the BLE capabilities of the ESP32 and nRF52832, we implemented the sample code available through the IDE to establish the bluetooth connection between the two development boards. We first tested the bluetooth compatibility of the boards by creating code by referencing the BLE_Client and BLE_Serverr sample code. We then uploaded the code to the board and established a connection to a mobile phone through the Nordic SDK Connectapp. By doing so we were able to send a signal to the board and toggle an LED. Afterwards, we drafted and tested code on both boards to establish a client and server relationship. These files enabled BLE on both boards and would cause them to search for a compatible board and connect. Once they are connected many functionalities can be done to prove connectivity, such as pressing a button on the server to send a signal to the client and enable an LED.

9.4 Carbon Dioxide Sensor Testing

9.4.1 Software Testing

Software testing for the CO_2 sensor involved developing and verifying the code needed to initialize and communicate with the sensor. As aforementioned in the software section, a crucial portion of testing the software was configuring an I2C communication between the CO_2 sensor and the ESP32.

During software testing for the CO_2 sensor, an issue was encountered where the system failed to read CO_2 data, with the error message "Failed to read CO_2 data" consistently appearing. After debugging, it was determined that the issue stemmed from trying to poll data from the sensor too fast. Since the response time of the SCD41 was relatively slow, especially compared to the DHT20, it couldn't collect readings within milliseconds.

To resolve the issue, we simply increased the time it took to gather readings from 500 ms to 2 seconds.

This debugging process highlighted the importance of ensuring proper configuration of communication protocols and thorough testing under various conditions to identify and resolve such issues.

9.4.2 Hardware testing

Once the initial coding was complete, the next step was to conduct the hardware testing of the CO_2 sensor. Our plan for testing was to subject the CO_2 sensor to multiple different environments to ensure its measurement capabilities and ultimately implement a pass fail criteria for the results.

To begin testing, our group started off by placing the CO_2 inside of an occupied UCF library study room. Results showed that the CO_2 levels were around 1100 ppm (parts per million) which is accurate to what the average levels of a small enclosed room would be with poor air circulation, and thus it passed this initial test. Following this, we moved the sensor outside of the study room to the third floor of the library to which we hypothesized would give us a decrease in ppm levels. However, to our surprise they were actually elevated and reached about 1300 ppm. After some consideration, we concluded that the crowded 3rd floor had elevated CO_2 levels during midterm exam week. It became clear that the increased levels were due to the significantly higher number of people on the floor at that time. We then proceeded to walk outside of the library to an open area near trees and the ppm values decreased immensely to around 300 ppm, which also passed since 300 ppm is the average natural concentration of an outdoor environment. With these results, we deemed that overall the CO_2 successfully passed each test case.

9.5 Temperature Sensor Testing

9.5.1 Software testing

To test the software for the temperature sensor, it was crucial to establish an I2C connection and ensure we could print information to a terminal, similar to the carbon dioxide sensor. Initially, there was an issue with printing data through UART to the terminal, as the I2C connection worked, but the terminal displayed garbage text. The temperature reading could be partially seen, however it was obstructed by a jumbled mix of letters and ascii symbols instead of the temperature reading. This issue was traced to a mismatch in configuration between the settings in the code and the terminal software. Specifically, the baud rate established in the code did not match the ArduinoIDE's terminal baud rate. After successful debugging of the UART configurations in the software, we could proceed to hardware testing.

9.5.2 Hardware testing

For hardware testing of the DHT20 temperature sensor, our plan consisted of programming the sensor in conjunction with the ESP32 development board. Since the sensor doesn’t have visual or audible ques for showcasing successful readings, it required a programmable aspect to view the readings of the temperature in a Tera Term terminal window. After successfully implementing and testing the software for the DHT20 temperature sensor, we moved on to hardware testing. Our plan involved programming the sensor to work along with the ESP32 development board kit. As mentioned earlier, the DHT20 sensor relied on outputting data to a serial monitor window, allowing us to monitor and verify the sensor's output during testing.

Once programming was completed and troubleshooting was finalized, we were able to get consistent readings of temperature. We first tested if the sensor could read a consistent value of the ambient temperature of the room it was located in and then we proceeded with doing small tests like breathing hot air onto the sensor to see if the temperature increased and fanning the sensor to ensure the temperature would decrease. These small tests proved to be successful and so we were then able to take the sensor into the senior design lab to take advantage of the technologies present for a more drastic temperature shift. Inside the senior design lab we connected the sensor to the appropriate pins on our ESP32 development board kit and waited until the temperature sensor was able to read an accurate ambient reading of the lab. Once ambient temperature was reached on the sensor, we turned on the heat gun in the lab and pointed it to our sensor to see if the sensor could recognize a drastic increase in temperature of the air around the sensor. The tests produced in our original room and in the senior design lab proved to be accurate which assured us for picking the appropriate and correct temperature sensor for our project.

9.6 Hardware and Software Testing of The Buzzer

Completion of the hardware testing of the PS1240 Piezo Buzzer was one of the simplest hardware testing completed. Within the product description of the Piezo buzzer, it boasts its ability to work with a 3 to 30 Volt peak-to-peak square wave input with zero DC bias and within the frequency range of 2K to 10K Hz. The test plan we chose to go with was to feed an initial voltage source to the buzzer and observe the outcome, from there we are to clarify the input signal with a varying peak-to-peak square input voltage and appropriate frequency range. Once the initial check is complete the test plan continues with incorporating the development kit for the ESP32 and programming the desired

output signal through PWM. By adjusting the duty cycle of the signal we were able to get different sounds from the buzzer.

Initial testing then began when we attached the buzzer to a breadboard with one pin in ground and the other pin connected in series with the Rhodes & Schwarz NGE100 power supply for an input voltage of 3 Volts. This did not work due to there being no square wave input and so we continued with the next phase. For the next phase we disconnected the Rhodes & Schwarz NGE100 power supply and modified the circuit in the breadboard to connect the Piezo buzzer with one pin to ground then the other pin in series with the Keysight EDU33212A Waveform generator that provided a 3 Volt peak-to-peak square wave input with zero DC bias and 2 KHz frequency. This next phase produced an appropriate sound to which we were happy with. We then listened for differences in volume and pitch when varying the input voltage from 3 to 30 Volts and then varying the frequency from 2KHz to 10KHz.

The next step in the testing process was to connect it to the ESP32 development kit and use code to utilize the button on the board and adjust the duty cycle. First we experimented and attempted to output DC to the buzzer, however this produced a low, brittle sound. After looking at the board's datasheet, we found that we were likely outputting a sine wave with minimal variation which would explain the sound output. Next we looked into inputting a square wave, which could be achieved through a PWM signal. We configured the TIM2 on the NUCLEO-WB09KE, which allowed us to send a PWM output through a pin on the board. After hooking up the buzzer to this pin and common ground we were able to get a solid, desired loud noise. We were then able to cycle through different duty cycles by connecting a button interrupt with different percentages for the input voltage. We found that by changing the duty cycle we achieved different sounds, however there were minimal changes to the volume.

Since it would be ideal to have different volumes for different buzzer alarms, we decided to test out an op-amp to amplify the PWM signal going to the buffer. Notably, we did not utilize the BJT driver circuit and instead plugged the positive and ground of the PWM signal directly to the buzzer. This did not cause any notable issues, but we will incorporate and test this driver circuit in the final design. Since we had only a 5V supply at the time of testing, we could only get the amplified PWM signal to make the buzzer 1dB louder. The final design will incorporate a PWM signal with a higher amplitude with a 12V supply, so it should be significantly louder if we were to max out the gain before clipping.

9.7 Servo Motor Testing

Interfacing the motor with the nRF52832 required a bit of research. We initially powered the motor with 5VDC and sent in a PWM signal, however this did not affect the movement of the motor's rod. Traditionally, servo motors are able to be in compliance with hobbyist boards such as the Arduino series and ESP32s by implementing a servo library and initializing the motor through pre-existing functions. We were not able to find a reliable servo library in compliance with the nRF52832 MCU, and so we attempted to send a raw PWM signal to make it move. After doing some research, we found out that servo motors require a specific period in order to interpret input signals, and that the Futaba S3004 motor requires a 50 hz period. By adjusting this value in the code we were able to get the motor moving to the desired angle through the development board.

9.8 LCD Screen Testing

For testing the LCD screen, we simply integrated it with an Arduino Uno to verify its display. After confirming this, we began testing by using sample code to check its touch screen capabilities. This included using code that would display a small square and would move positions if it detected any touch from the user. After this was confirmed, we proceeded to translate our figma prototypes to the screen. This included printing sensor data to the screen, creating both CO2/Low Battery visual warnings, and creating both autonomous and manual mode buttons.

9.9 Plan for Senior Design 2

At the time, when we were developing code on the STM32WB09Ke, we planned on incorporating all of the subsystems into one comprehensive unit, and implementing our plans for combining the PSU and MCU in both the control unit and the vent. We planned on polishing our PCB design and ordering it as soon as possible so we could test and ensure that the subsections meet the desired functionality. We were also planning on finalizing the code that will be implemented on both subsystems, being able to combine all of the individual components into one master script that will also implement the logic as shown in the flowcharts. Once we had both of these accomplished, we would look into uploading the code onto the flash of the MCU on the PCB.

Once we were able to configure the pins to our MCU and turn the LCD screen on, we were planning on being able to print characters and ensure touch screen capabilities work as intended. We were also going to use the graphics library to aid in development of the user interface. The screen will also be able to cycle

through different menus, with each one having their own design. Once we are certain that the LCD screen works and is compatible with the graphics library and the STM32, we will connect it with the logic of the rest of the software design and ensure that two way communication and data interpretation is possible between the LCD screen and the MCU.

In order to upload code onto the flash of the STM32WB09KE we will need to implement a method on each board that will allow us to interface with the MCU to program and debug it as necessary. The STM32 can be programmed through JTAG or SWD (Serial Wire Debug). ST offers several debuggers under the ST-LINK product line. The ST-LINK V2 and V3 lines are the main considerations for our design considering the other products in the line are obsolete. The ST-LINK V2 is a popular and cheap option with a few caveats. Firstly, we plan on using SWD to interface with the STM32 as this is what ST recommends and it requires less pins than JTAG (10 as opposed to 20). Unfortunately, the ST-LINK V2 is only capable of SWD through a JTAG socket, requiring 20 pins on the board to be able to interface with the MCU. There is, however, a work-around involving the use of a dedicated JTAG to SWD adapter which would allow us to only need the standard 10 pins for SWD on the board. This would add cost and may add some headache if the adapter is found to be non-functional. Tagconnect also offers officially promoted third-party cables that only require small pads to be etched into the PCB as opposed to an actual pin header. This option, however, tends to be quite costly and also may come loose from the board much easier than using pin-connecters.

The ST-LINK V3 is a little more modern and utilizes the proper 10-pin SWD socket without the need for an adapter like the ST-LINK V2. Unfortunately, the ST-LINK V3 does not support application voltages under 3.3V, which would pose issues when interfacing with the vent interaction subsystem which has a 2.5V supply. This means using the ST-LINK V2 with a proper adapter will be the method we use to program our MCUs moving forward.

In addition to having the previous sections working, we will also test the battery life of the system. We will accomplish this by having all of the necessary components available and powered, then examining and interpreting the power consumed over a period of time. Once we find out the average power consumption for every mode the system will be in, we will be able to approximate the average time spent in each mode and calculate the life expectancy of the vent before a battery replacement is necessary.

In SD2 we were also planning on interfacing our components with a vent design. We were evaluating the pros and cons of different vent designs, and once we

find the most optimal design we are going to test function compatibility to meet our specifications. There are many mechanisms that we can experiment with to allow BREATHE to open and close the vent, however our main focus will be dedicated towards the electrical and software aspects.

10.0 Administrative Content

10.1 Budget

Item Description	Price	Quantity
NRF52832 DevKits	$24.95	2
NRF52832 module	$9.29	5
ESP32 WROOM 32	$5.30	3
PCBs	$9.74	40
PCB Additional Components	$390.07	N/A
Amazon Carbon Dioxide Monitor	$39.99	1
Carbon Dioxide Sensor	$37.10	4
Temperature Sensor	$4.50	6
LCD Screen	$41	2
Batteries	$18	5
Buzzer	$0.56	10
Sample Vent	$26.60	1
Servo Motor	$7.20	5
J-Link Debugger	$60	1
ESP32 Debugger	$6	2
Total Cost	$1,448.68	

Table 27: Budget Table

For this project, we were allotted a budget of $2000, and ended up spending around ¾ of said budget. A large portion of the budget went towards the PCB and additional components, due to the fact that our boards needed revisions and tariffs made these subsequent boards more costly. The carbon dioxide sensors also were the most expensive single component on any of the boards, possible because of its niche use and the fact that it directly measures carbon dioxide rather than using other methods. The LCD screen also was quite expensive due to its large size and pin integration. These cost-adders including our multiple backups in case of hardware issues (which we occasionally did run into) made our budget much more inflated than what we originally expected in SD1.

10.2 Project Milestones

Task	Task Description	Start Date	Anticipated End Date
Group Formation	Form a group before the second lecture *Group Formation*	8/20/2024	8/22/2024
Project Idea Creation	Formulate a project idea	8/22/2024	8/30/2024
Divide and Conquer Document	Finish at least 10 pages of the D/C document	8/30/2024	9/6/2024
Divide and Conquer Meeting with Professor	Discuss divide and conquer document and take note of criticism and plan to make changes	9/10/2024	9/10/2024
Divide and Conquer Revisions	Discuss criticisms given and make religions to the Divide and Conquer Document	9/10/2024	9/27/2024
60 Page Report Draft	Finish our group's 60 page report draft	9/28/2024	10/25/2024
60 Page Report	Discuss criticisms given	10/26/2024	11/8/2024

Revision	and make revisions to the 60 page report		
Final Report and Mini Demo Video	Complete the Final Report and Mini Demo Video for SD1	11/9/2024	11/26/2024

Table 28: Senior Design I Milestones

Task	Task Description	Start Date	Anticipated End Date
PCB Testing	Start PCB testing and ensure they all function as desired	2/24/2025	3/10/2025
System Integration and Testing, incorporate stretch goals if possible	Start integrating all subsystems together and begin testing their functionality. Begin implementing stretch goals if realistic and possible. Need some time to identify and correct errors on both the software and hardware side.	3/10/2025	3/31/2025
Finalize project prototype	Finalize implemented all subsystems together into one, cohesive project.	3/31/2025	4/10/2025
Practice Project Demo	Begin practicing a simulated project demonstration to make sure everything works as anticipated	4/10/2025	4/15/2025
Possibly implement optimizations (if necessary and or possible)	If a system can be improved, begin implementing new optimizations to BREATHE.	3/31/2025	4/10/2025
Finalize any documents	Finalize any documents needed	4/01/2025	4/15/2025

Group Practices Final Presentation	Practice Final presentation as a group	4/13/2025	4/13/2025
Final Presentation	Present final project to SD committee.	4/15/2025	4/15/2025

Table 29: Senior Design II Milestones

10.3 Work Distribution Table

Responsibilities	Primary Member	Secondary Member
Power Supply Design	Kyle Ingleton	Emily Baross
Vent Interaction subsystem PCB Design	Kyle Ingleton	Emily Baross
Control Unit PCB Design	Emily Baross	Kyle Ingleton
Vent Interaction Subsystem Software Design	Marcus Loyd	Patrick Tumbos
Control Unit Software Design	Patrick Tumbos	Marcus Loyd
LCD and GUI Design	Patrick Tumbos	Marcus Loyd
Carbon Dioxide Sensor Testing	Patrick Tumbos	Emily Baross
Temperature Sensor Testing	Patrick Tumbos	Emily Baross
Motor Testing and Configuration	Marcus Loyd	Emily Baross
Vent Interaction Subsystem Enclosure Design/Testing	Emily Baross	Marcus Loyd
Control Unit Enclosure Design/Testing	Emily Baross	Patrick Tumbos
Buzzer configuration and Testing	Marcus Loyd	Kyle Ingleton

Table 30: Work Distribution Table

11.0 Conclusion

In conclusion, BREATHE offers a user-friendly and efficient solution for improving the environmental conditions of common rooms in households. By allowing users to input their desired conditions of their rooms, they are able to comfortably live without worries of temperature variations. This can significantly improve quality of life due to not having to rely on less effective alternative means of temperature control such as ceiling fans. Although this product can benefit consumers, like any technology there is still a lot of room for improvement. This can include stretch goals such as implementing a mobile app to allow users to have a more convenient user interface with the control unit.

In summary, although the STM32WB09KE was originally selected and utilized for testing, the ESP32 and nRF52832 was chosen as the MCU going forward due to its robust documentation and plethora of pins. The temperature and carbon dioxide sensors that were selected were the DHT20 and SCD41 respectively, as they both provided a strong balance between efficiency, ease of integration with the ESP32, and reliability. Moreover, the PS1240 piezoelectric buzzer was selected for its ability to output a clear and audible sound and was amplified with the use of an operational amplifier. Likewise, the motor selected was the MG966R servo motor due to its high capability of torque and speed. Additionally, the chosen methods of voltage regulation were an LDO and buck converter which were compliant in the vent interaction subsystem’s long term power needs. Various software tools were used, such as the Arduino IDE and the Nordic SDK in the development of BREATHE's software. Serial communications such as UART and I2C made interfacing and testing the sensors possible, and the use of BLE allowed for the MCU’s to communicate with each other efficiently. Lastly, each component underwent thorough testing to ensure that they worked optimally.

By working on this project the group was able to build and expand our foundational skills to prepare us for industry. We were able to apply the skills we gained from class to a practical project, guiding us throughout the development process all the way from system design and research to prototyping and experimentation. This project has also introduced us to different technologies utilized in industry, such as KiCad and the Nordic SDK. The challenges that we came across had us consider many different possible options and we evaluated each one before picking the most optimal solution. The biggest challenges we faced were finding ways to reduce power consumption for the vent interaction subsystem and have each subsystem interact with other subsystems, however

by learning and utilizing the past knowledge and experiences we were able to design BREATHE to accommodate for these.

Appendix A. References

[1] Google, "Understand Google Smart Home Compatibility," *Google Home*, Sept. 6, 2024. [Online]. Available: https://home.google.com/what-is-google-home/.

[2] J. Mansy *et al.*, "CtrlFlow," *CtrlFlow - Group 17*, 2021. [Online]. Available: https://www.ece.ucf.edu/seniordesign/fa2020sp2021/g17/#team.

[3] W. Dominguez *et al.*, *The Automated Ventilation Controller*, 2021. [Online]. Available: https://www.ece.ucf.edu/seniordesign/su2021fa2021/g15/.

[4] Espressif Systems, “ESP32 Datasheet,” [Online]. Available: https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf

[5] Seeed Studio, “XIAO BLE - Nano BLE MCU (nRF52840) Product Specification,” [Online]. Available: https://files.seeedstudio.com/wiki/XIAO-BLE/Nano_BLE_MCU-nRF52840_PS_v1.1.pdf

[6] Silicon Labs, “EFR32BG22 Datasheet,” [Online]. Available: https://www.silabs.com/documents/public/data-sheets/efr32bg22-datasheet.pdf

[7] Renesas Electronics, “DA14531 Datasheet,” [Online]. Available: https://www.mouser.com/datasheet/2/698/REN_DA14531_3v6_DST_20220921-3075800.pdf?srsltid=AfmBOorMdf0l8dpkwq7trmmbND0V8eaWM_KtjzQYGPqeMxzAb45Qv5en.

[8] Texas Instruments, “CC2640R2F Datasheet,” [Online]. Available: https://www.ti.com/lit/ds/symlink/cc2640r2f.pdf

[9] STMicroelectronics, “STM32WB09KE Datasheet,” [Online]. Available: https://www.st.com/resource/en/datasheet/stm32wb09ke.pdf

[10] Analog Devices, “Analog Temperature Sensors,” [Online]. Available: https://www.analog.com/en/product-category/analog-temperature-sensors.html#category-detail

[11] ChatGPT, “Temperature Sensor Datasheet,” [Online]. Available: https://chatgpt.com/share/671b9dc5-e954-8002-90ff-2f9c39eec5a7

[12] Texas Instruments, “LM35 Precision Centigrade Temperature Sensors,” [Online]. Available: https://www.ti.com/lit/ds/symlink/lm35.pdf

[13] Analog Devices, “TMP35/36/37 Datasheet,” [Online]. Available: https://www.analog.com/media/en/technical-documentation/data-sheets/TMP35_36_37.pdf

[14] Texas Instruments, “DHT20 Datasheet,” [Online]. Available: https://www.ti.com/lit/ds/symlink/DHT20.pdf?ts=1729788269632&ref_url=https%253A%252F%252Fwww.google.com%252F

[15] Sensirion, “SHT3x Series Datasheet,” [Online]. Available: https://sensirion.com/media/documents/213E6A3B/63A5A569/Datasheet_SHT3x_DIS.pdf

[16] Kaiterra, “Carbon Dioxide Sensors,” [Online]. Available: https://learn.kaiterra.com/en/air-academy/carbon-dioxide-sensors

[17] J. Zhang and T. Liu, “Electrochemical Sensors for Gas Detection,” in *Encyclopedia of Sensors*, Springer, 2023. [Online]. Available: https://link.springer.com/referenceworkentry/10.1007/978-3-031-16338-8_22-1

[18] Winsen, “MH-Z19B CO_2 Sensor,” [Online]. Available: https://www.winsen-sensor.com/d/files/infrared-gas-sensor/mh-z19b-co2-ver1_0.pdf

[19] Farnell, “SEN01590 CO_2 Sensor Datasheet,” [Online]. Available: https://www.farnell.com/datasheets/3176108.pdf

[20] CO2 Meter, “S8 CO_2 Sensor Datasheet,” [Online]. Available: https://www.co2meters.com/Documentation/Datasheets/DS-S8-3.2.pdf

[21] MikroElektronika, “SCD41 Datasheet,” [Online]. Available: https://download.mikroe.com/documents/datasheets/SCD41%20Datasheet.pdf

[22] Nelson Miller Group, “What Is a SAW Touch Screen and How Does It Work?” [Online]. Available: https://nelsonmillergroup.com/resources/what-is-a-saw-touch-screen-and-how-does-it-work/

[23] C. J. Schubert, et al., “An Overview of Touch Screen Technologies,” *National Library of Medicine*, 2021. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8309784/

[24] Fortec, “The Impact of Optical Bonding on Touchscreens,” [Online]. Available: https://fortec.us/blog/the-impact-of-optical-bonding-on-touchscreens-apollo-display-technologies/#:~:text=Applications%20Across%20Industries,user%20experiences%20and%20drives%20innovation

[25] DFRobot, “3.5 inches 480 x 320 TFT LCD Capacitive Touchscreen,” [Online]. Available: https://wiki.dfrobot.com/3.5inches_480_320_TFT_LCD_Capacitive_Touchscreen_SKU_DFR0669

[26] Adafruit Industries, “2.8 TFT Touch Shield v2 - Capacitive or Resistive,” [Online]. Available: https://www.adafruit.com/product/1947.

[27] FTDI, “FT6206 Capacitive Touch Controller Datasheet,” [Online]. Available: https://www.buydisplay.com/download/ic/FT6206.pdf.

[28] Waveshare, “4inch Resistive Touch LCD,” [Online]. Available: https://www.waveshare.com/4inch-tft-touch-shield.htm

[29] HP InfoTech, “ILI9486” [Online]. Available: https://www.hpinfotech.ro/ILI9486.pdf

[30] Waveshare, “3.2inch 320x240 Touch LCD (D),” [Online]. Available: https://www.waveshare.com/wiki/3.2inch_320x240_Touch_LCD_(D).

[31] ILI Technology Corporation, “ILI9341 TFT LCD Driver Datasheet,” [Online]. Available: https://cdn-shop.adafruit.com/datasheets/ILI9341.pdf.

[32] Futtr Buzzer, “Advantages of Piezoelectric Buzzer,” [Online]. Available: https://futtrobuzzer.com/advantages-of-piezoelectric-buzzer/

[33] American Piezo, “Top Uses of Piezoelectricity in Everyday Applications,” [Online]. Available: https://www.americanpiezo.com/blog/top-uses-of-piezoelectricity-in-everyday-applications/.

[34] CUI Devices, “Buzzer Basics,” Mouser, 2023. [Online]. Available: https://www.mouser.com/catalog/additional/CUIDevices_buzzer_basics_mouser.pdf

[35] FBElec, “Advantages and Disadvantages of Magnetic Buzzers,” [Online]. Available:

https://www.fbelec.com/blog/advantages-and-disadvantages-of-magnetic-buzzers.html

[36]"GPT Share Link," *ChatGPT*, [Online]. Available: https://chatgpt.com/share/671b87ca-ba4c-8002-9bdb-e9b638033ae9

[37] TDK, “Piezoelectronic Buzzer Catalog,” [Online]. Available: https://product.tdk.com/system/files/dam/doc/product/sw_piezo/sw_piezo/piezo-buzzer/catalog/piezoelectronic_buzzer_ps_en.pdf

[38] CMI, “CMI 1295 & 1285T Datasheet,” [Online]. Available: https://www.mouser.com/datasheet/2/1628/cmi_1295_1285t-3509445.pdf

[39] "GPT2 Share Link," *ChatGPT*, [Online]. Available: https://chatgpt.com/share/671b8899-4010-8002-b778-b46ad24fbe83

[40] Jameco, “Jameco 2337711 Datasheet,” [Online]. Available: https://www.jameco.com/Jameco/Products/ProdDS/2337711.pdf

[41] Bradley, “AC and DC Motors: Differences and Advantages | Types of Electric Motors,” *Gainesville Industrial Blog*, Jul. 01, 2019. https://www.gainesvilleindustrial.com/blog/ac-dc-motors/

[42] Monolithicpower.com, 2024. https://www.monolithicpower.com/en/learning/resources/brushless-vs-brushed-dc-motors

[43] Jameco, “How Servo Motors Work | Servo Motor Controllers,” *Jameco.com*, 2019. https://www.jameco.com/jameco/workshop/howitworks/how-servo-motors-work.html

[44] “Speed - Torque Curves for Stepper Motors,” *Oriental Motor U.S.A. Corp.* https://www.orientalmotor.com/stepper-motors/technology/speed-torque-curves-for-stepper-motors.html

[45] “AMCI : Advanced Micro Controls Inc :: Stepper vs Servo,” *www.amci.com*. https://www.amci.com/industrial-automation-resources/plc-automation-tutorials/stepper-vs-servo/

[46] P. Dixit, “What Are The Types of Servo Motors | Robu.in,” *Robu.in | Indian Online Store | RC Hobby | Robotics*, Apr. 20, 2020. Available: https://robu.in/types-of-servo-motor/

[47] “How to Make ANY servo rotate 360° - EASY and FAST,” www.youtube.com. https://www.youtube.com/watch?v=JhHSXCLsN4k

[48] “Servo Motor Micro SG92R,” *ProtoSupplies*, Jul. 06, 2020. https://protosupplies.com/product/servo-motor-micro-sg92r/

[49] “HS-5645MG High Torque Metal Gear 24T Digital Sport Servo,” *Hitecrcd.com*, 2024. https://hitecrcd.com/hs-5645mg-high-torque-metal-gear-digital-sport-servo/

[50] “Servo Motor MG996 360 Degree Continuous Rotation,” *ProtoSupplies*. https://protosupplies.com/product/servo-motor-mg996-360-degree-continuous-rotation/

[51]E. P. Company, “Article | What Is an Encoder?,” *www.encoder.com*. https://www.encoder.com/article-what-is-an-encoder

[52]“EC10E1220501 Product information | EC10E Series | Through shaft Encoder | Encoders | Products Search | Products & Technologies | Alps Alpine,” *Alpsalpine.com*, 2024. https://tech.alpsalpine.com/e/products/detail/EC10E1220501/ (accessed Oct. 25, 2024).

[53]*bourns*,Mar.31,2015. https://www.bourns.com/docs/Product-Datasheets/ace.pdf

[54]“Series 292 20mm Optical Ring Encoder Datasheet,” *ctscorp*, Dec. 10, 2020. https://www.ctscorp.com/Files/DataSheets/Encoders/encoders-292-datasheet.pdf

[55]K. Nice and K. Hall-Geisler, “How Gears Work,” *howstuffworks*, Jun. 09, 2023. https://science.howstuffworks.com/transport/engines-equipment/gear.htm

[56] https://www.howstuffworks.com, “How Gears Work,” *HowStuffWorks*, Nov. 16, 2000. https://science.howstuffworks.com/transport/engines-equipment/gear.htm

[57] “Primacy E-Books,” *Primacy E-Books*, Jun. 09, 2022. https://www.primacyebooks.com/lesson/chain-drive/ (accessed Oct. 25, 2024).

[58] “Bevel Gears,” *Stahl Gear & Machine Co.*, Mar. 23, 2023. https://stahlgear.com/project/bevel-gears/ (accessed Oct. 25, 2024).

[59] “As the Worm Turns: How Worm Gears Control Speed in Industrial Machinery,” *blog.wcbranham.com*, Mar. 28, 2023. https://blog.wcbranham.com/worm-gears-regulate-speed

[60] D. Knight, “Introduction to linear voltage regulators,” DigiKey, https://www.digikey.com/en/maker/tutorials/2016/introduction-to-linear-voltage-regulators

[61] “The Fundamentals of LDO Design and Applications,” The Fundamentals of LDO Design and Applications | Analog Devices, https://www.analog.com/en/lp/001/fundamentals-of-ldo-design-and-applications.html#:~:text=A%20low%20dropout%20regulator%20(LDO,controlled%20by%20the%20error%20amplifier

[62] “Switching regulator,” Switching Regulator | Analog Devices, https://www.analog.com/en/resources/glossary/switching_regulator.html#:~:text=A%20switching%20regulator%20is%20a,in%20a%20feedback%20control%20loop

[63] F. D. Stasi, Working with Boost Converters, https://www.ti.com/lit/an/snva731/snva731.pdf?ts=1727865371311&ref_url=https%253A%252F%252Fwww.google.com%252F

[64] TPS61023 3.7-A Boost Converter with 0.5-V Ultra-low Input Voltage, https://www.ti.com/lit/ds/symlink/tps61023.pdf?ts=1629351001794&ref_url=https%253A%252F%252Fwww.mouser.de%252F

[65] F. D. Stasi, DC/DC Converter Datasheets - Quiescent Current Demystified: Part Two, https://www.ti.com/lit/ta/ssztcb3/ssztcb3.pdf?ts=1727875659535&ref_url=https%253A%252F%252Fwww.google.com%252F

[66] Diodes AP63200-AP63201-AP63203-AP63205 Datasheet, https://www.diodes.com/datasheet/download/AP63200-AP63201-AP63203-AP63205.pdf

[67] TPS63070 2-V to 16-V Buck-Boost Converter With 3.6-A Switch Current, https://www.ti.com/lit/ds/slvsc58b/slvsc58b.pdf?ts=1726744288114&ref_url=https%253A%252F%252Fwww.google.com%252F

[68] S. Keeping, “The SEPIC option for battery-power management,” DigiKey, https://www.digikey.com/en/articles/the-sepic-option-for-battery-power-management#:~:text=The%20most%20notable%20downside%20of,the%20extra%20capacitor%20and%20inductor

[69] Lithium/Manganese Dioxide Battery CR123A, https://www.duracell.com/wp-content/uploads/2021/06/CR123_US.pdf

[70] 9V Energizer L522, https://data.energizer.com/pdfs/l522.pdf

[71] “XC6206 series,” XC6206 | TOREX SEMICONDUCTOR LTD., https://product.torexsemi.com/en/series/xc6206.

[72] LTC3622 Datasheet, https://www.analog.com/media/en/technical-documentation/data-sheets/LTC3622-3622-2-3622-23-5.pdf

[73] Texas Instruments, "Universal Asynchronous Receiver/Transmitter (UART) Peripheral Guide," [Online]. Available: https://www.ti.com/lit/ug/sprugp1/sprugp1.pdf?ts=1731375239972.

[74] NXP, "I2C-bus specification and user manual," [Online]. Available: https://www.nxp.com/docs/en/user-guide/UM10204.pdf.

[75] Analog Devices, "Introduction to SPI Interface," [Online]. Available: https://www.analog.com/en/resources/analog-dialogue/articles/introduction-to-spi-interface.html.

[76] Mouser Electronics, "SPI Interface Specification," [Online]. Available: https://www.mouser.com/pdfdocs/tn15_spi_interface_specification.PDF.

[77] Bluetooth SIG, "Bluetooth 5 Core Specification," [Online]. Available: https://www.bluetooth.com/wp-content/uploads/2019/03/Bluetooth_5-FINAL.pdf.

[78] Silicon Labs, "Fundamentals of Bluetooth Low Energy," [Online]. Available: https://www.silabs.com/documents/public/user-guides/ug103-14-fundamentals-ble.pdf.

[79] Tektronix, "Wi-Fi Overview: 802.11 Physical Layer and Transmitter Measurements," [Online]. Available: https://www.tek.com/en/documents/primer/wi-fi-overview-80211-physical-layer-and-transmitter-measurements.

[80] A. Weeks, UCF, “Course Notes,” [Online]. Available: https://webcourses.ucf.edu/courses/1441572/pages/notes

[81] K. T. Wang, “Embedded Java: A Survey of the Current State,” *IEEE Transactions on Computers*, vol. 55, no. 5, pp. 655-668, May 2006. [Online]. Available: https://ieeexplore.ieee.org/document/1300326

[82] VDC Research, “Research Insights,” [Online]. Available: https://blog.vdcresearch.com/.a/6a0115714871cc970c0134875578fa970c-pi

[83] Qt, “C for Embedded: Advantages, Disadvantages, and Myths,” [Online]. Available: https://www.qt.io/blog/c-for-embedded-advantages-disadvantages-and-myths

[84] MicroPython, “I2C – Inter-Integrated Circuit,” [Online]. Available: https://docs.micropython.org/en/latest/library/machine.I2C.html

[85] MicroPython, “MicroPython for STM32,” [Online]. Available: https://micropython.org/stm32/

[86] MicroPython, “Garbage Collection,” [Online]. Available: https://docs.micropython.org/en/latest/library/gc.html

[87] Oracle, “Java ME Overview,” [Online]. Available: https://www.oracle.com/java/technologies/javameoverview.html

[88] Oracle, “Java ME CLDC API,” [Online]. Available: https://docs.oracle.com/javame/8.0/api/cldc/api/index.html

[89] Oracle, “MIDP Programming and Packaging,” [Online]. Available: https://www.oracle.com/technical-resources/articles/javame/midp-programming-packaging.html

[90] TechnoReply, “Comparing Java with Other Programming Languages,” [Online]. Available: https://technorely.com/insights/comparing-java-with-other-programming-languages-when-and-why-to-choose-java

[91] A. Obregon, “Java in Embedded Systems: Opportunities and Limitations,” *Medium*, 2023. [Online]. Available: https://medium.com/@AlexanderObregon/java-in-embedded-systems-opportunities-and-limitations-cd0128c9f2c2#:~:text=Resource%20Constraints,pose%20challenges%20in%20such%20environments

[92] STMicroelectronics, “STM32CubeF0,” [Online]. Available: https://www.st.com/en/embedded-software/stm32cubef0.html

[93] PlatformIO, “PlatformIO Frameworks,” [Online]. Available: https://registry.platformio.org/search?t=platform

[94] PlatformIO, “Library Manager Dependencies,” [Online]. Available: https://docs.platformio.org/en/latest/librarymanager/dependencies.html

[95] STMicroelectronics, “How to program and debug the STM32 using the Arduino IDE,” [Online]. Available: https://community.st.com/t5/stm32-mcus/how-to-program-and-debug-the-stm32-using-the-arduino-ide/ta-p/608514

[96] Wokwi, “Wokwi - Online Electronics Simulator,” [Online]. Available: https://docs.wokwi.com/#:~:text=Wokwi%20is%20an%20online%20Electronics,Arduino%20Uno%20%22Hello%20World%22

[97] STMicroelectronics, “TouchGFX Designer,” [Online]. Available: https://www.st.com/en/development-tools/touchgfxdesigner.html

[98] STMicroelectronics, “STM32Cube™,” [Online]. Available: https://www.st.com/en/embedded-software/stemwin.html/1000#

[99] LVGL, “Features,” [Online]. Available: https://lvgl.io/features

[100] Bluetooth, “Bluetooth Technology Overview,” [Online]. Available: https://www.bluetooth.com/learn-about-bluetooth/tech-overview/

[101] Bluetooth, “Bluetooth Core Specification Version 6.0,” [Online]. Available: https://www.bluetooth.com/specifications/specs/core60-html/

[102] Mouser Electronics, “SPI Interface Specification,” [Online]. Available: https://www.mouser.com/pdfdocs/tn15_spi_interface_specification.PDF?srsltid=AfmBOopTL2eHqlp6_2lnDhkGZdVwD3d9PZegYLbNcUnEoZU-W27fPsDc.

[103] IPC, “IPC-2221A: Generic Standard on Printed Board Design,” [Online]. Available: https://www-eng.lbl.gov/~shuman/NEXT/CURRENT_DESIGN/TP/MATERIALS/IPC-2221A(L).pdf

[104] NIOSH, “Carbon Dioxide: General,” [Online]. Available: https://nj.gov/health/eoh/rtkweb/documents/fs/0343.pdf.

[105] ASHRAE, “ANSI/ASHRAE Standard 62.1-2022: Ventilation for Acceptable Indoor Air Quality,” [Online]. Available: https://static1.squarespace.com/static/6320b844c3820725e4d5688f/t/6372af076022e56f815dc7f5/1668460297956/ASHRAE+62.1-2022+%281%29.pdf.

[106] ASHRAE, “ANSI/ASHRAE Standard 55-2017: Thermal Environmental Conditions for Human Occupancy,” [Online]. Available: https://www.ashrae.org/file%20library/technical%20resources/standards%20and%20guidelines/standards%20addenda/55_2017_d_20200731.pdf.

[107] ISO/IEC, “Programming Languages — C,” [Online]. Available: https://www.open-std.org/jtc1/sc22/wg14/www/docs/n2310.pdf.

[108] NobelCert, “IEC 61508-1: Functional Safety of Electrical/Electronic/Programmable Electronic Safety-Related Systems,” [Online]. Available: https://nobelcert.com/DataFiles/FreeUpload/IEC%2061508-1-2010.pdf.

[109] IEC, “IEC 60364-6: Electrical Installations of Buildings – Part 6: Verification,” [Online]. Available: https://cdn.standards.iteh.ai/samples/21340/5b6773dd9a0c417e97501490c483a487/IEC-60364-6-2016.pdf..

[110] S. M. Kerner, “What is a large language model (LLM)? – TechTarget Definition,” *WhatIs.com*, Sep. 2023. https://www.techtarget.com/whatis/definition/large-language-model-LLM

[111] Y. Mehdi, “Announcing Microsoft Copilot, your everyday AI companion,” *The Official Microsoft Blog*, Sep. 21, 2023. https://blogs.microsoft.com/blog/2023/09/21/announcing-microsoft-copilot-your-everyday-ai-companion/

[112] “Microsoft Copilot Pros and Cons The Balancing Act,” *redresscompliance.com*, Dec. 24, 2023. https://redresscompliance.com/microsoft-copilot-pros-and-cons-the-balancing-act/

[113] A. B. published, “What is Google Gemini? Everything you need to know about Google’s next-gen AI,” *TechRadar*, Dec. 09, 2023. https://www.techradar.com/computing/artificial-intelligence/what-is-google-gemini

[114] Meta AI, “Meta AI,” *ai.meta.com*, 2024. https://ai.meta.com/meta-ai/

[115] Meta, “Meet Your New Assistant: Meta AI, Built With Llama 3,” *Meta*, Apr. 18, 2024. https://about.fb.com/news/2024/04/meta-ai-assistant-built-with-llama-3/

[116] A. Hetler, “What is ChatGPT? Everything You Need to Know,” *TechTarget*, Dec. 2023. https://www.techtarget.com/whatis/definition/ChatGPT

[117] “What is RLHF? - Reinforcement Learning from Human Feedback Explained - AWS,” *Amazon Web Services, Inc.* https://aws.amazon.com/what-is/reinforcement-learning-from-human-feedback/

[118] “Chatgpt,” ChatGPT, https://chatgpt.com/share/671b82b8-4e4c-8005-b009-fdd90139746e.

[119] “ChatGPT,” *Chatgpt.com*, 2024. https://chatgpt.com/share/671b6e09-8530-8002-ab2b-66526c5bd37d.

[120] “ChatGPT,” *Chatgpt.com*, 2024. https://chatgpt.com/c/671b7ae5-d6e0-8002-8818-d3a3dc41d2e5.

[121] STMicroelectronics, "How to develop RF hardware using STM32WB microcontrollers," Application Note AN5165, Jan. 2021. [Online]. Available: https://www.st.com/resource/en/application_note/an5165-how-to-develop-rf-hardware-using-stm32wb-microcontrollers-stmicroelectronics.pdf

[122] NRF52832 product specification V1.8, https://www.mouser.com/datasheet/2/297/nRF52832_PS_v1_8-2942485.pdf

[123] “Piezoelectronic buzzers Without oscillator circuit Pin terminal PS series REACH SVHC-Free RoHS.” Accessed: Nov. 06, 2024. [Online]. Available: https://product.tdk.com/en/system/files/dam/doc/product/sw_piezo/sw_piezo/piezo-buzzer/catalog/piezoelectronic_buzzer_ps_en.pdf

[124] “LM1117 800-mA, Low-Dropout Linear Regulator.” Available: https://www.ti.com/lit/ds/symlink/lm1117.pdf

[125] “Controlled Impedance Calculator - JLCPCB,” *Jlcpcb.com*, 2024. https://jlcpcb.com/pcb-impedance-calculator/ (accessed Nov. 26, 2024).

[126] “PCB Manufacturing & Assembly Capabilities - JLCPCB,” *jlcpcb.com*. https://jlcpcb.com/capabilities/pcb-capabilities

Appendix B - Copyright Permissions

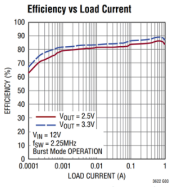
Request for Permission to Use Image from MAX77505 Datahseet

KI Kyle Ingleton
To brand@analog.com
Cc Patrick Tumbos; Marcus Loyd

Reply | Reply All | Forward
Tue 11/26/2024 3:56 AM

Dear Analog Devices,

We are a group of students from the University of Central Florida working on our senior design project, and we would like to request permission to use an image from the MAX77505 datasheet. The image would be used solely for educational purposes in our project materials, with appropriate credit is given to your company.
The image and a link to it are attached below for reference.



https://www.analog.com/media/en/technical-documentation/data-sheets/max77505.pdf

Thank you very much for your time and consideration, and we look forward to your response.
Best regards,
Emily Baross
Kyle Ingleton
Marcus Loyd
Patrick Tumbos

SA Scarpulla, Alyssa <Alyssa.Scarpulla@analog.com>
To: Peterson, Susan <Susan.Peterson@analog.com>; Kyle Ingleton
Cc: Patrick Tumbos; Marcus Loyd; Brand <brand@analog.com>

Wed 12/4/2024 2:02 PM

Retention: UCF Delete after 10 Years (10 years) Expires: Sat 12/2/2034 2:02 PM

Start reply with: Thank you! | Permission granted! | Wonderful! Thank you so much!

Hi Kyle,

We appreciate you reaching out for permission to use our copyrighted image.

This email serves as written permission for using the image from the MAX77505 datasheet in your senior design project. Please include the following attribution under the photo (if feasible) "This image is used with permission from Analog Devices, Inc. ("ADI"), © 2024."

Please let me know if you have any additional questions.

Best,
Alyssa Scarpulla
Associate Counsel, Intellectual Property
Email alyssa.scarpulla@analog.com

ANALOG DEVICES

analog.com

Appendix E - ChatGPT prompts and outcomes

Prompt 1:

"What voltage regulator would be best for a battery to last as long as possible"

Response:
"Choosing the right voltage regulator to maximize battery life depends on the specific requirements of your project, such as the input and output voltages, current consumption, and efficiency needs. Here are some key points to consider:

1. Types of Voltage Regulators:

- Linear Regulators (LDO): Simple and inexpensive, linear regulators work well for low-current applications but are inefficient when the input voltage

is much higher than the output. The excess energy is dissipated as heat, leading to faster battery drain.
- Switching Regulators (Buck/Boost Converters): These are much more efficient than linear regulators, especially for higher currents and when the input voltage is significantly different from the output. They can step down (buck) or step up (boost) the voltage.

2. Factors Influencing Battery Life:

- Efficiency: Higher efficiency means less power wasted as heat, prolonging battery life. Switching regulators typically have efficiencies of 80-95%, while linear regulators can be as low as 50% or less in some cases.
- Quiescent Current: This is the current the regulator consumes even when the load is minimal. For battery-operated devices, a regulator with low quiescent current is ideal.
- Input Voltage Range: Make sure the regulator can handle the full range of the battery voltage as it discharges.

3. Recommended Voltage Regulators for Maximizing Battery Life:

- Buck Converter: If your battery voltage is higher than the required output voltage, a buck converter (step-down) is suitable. Look for one with high efficiency and low quiescent current, such as the Texas Instruments TPS62125.
- Boost Converter: If your battery voltage is lower than your required output voltage, a boost converter (step-up) is needed. The Analog Devices LTC3539 is an efficient option with low quiescent current.
- Buck-Boost Converter: If your input voltage can fluctuate above and below your desired output, a buck-boost converter like the Texas Instruments LM2623 or the Analog Devices LT8495 is a good choice.

4. Ultra-Low Quiescent Current Regulators:

- If your project is low power and spends a lot of time in sleep mode, choose a regulator with ultra-low quiescent current, such as the Microchip MCP1700 (LDO) or the Texas Instruments TPS62743 (buck converter).

Selecting a regulator that balances efficiency and low quiescent current will help extend battery life." [114].

Prompt 2:

“Give me standards regarding air quality using co2 and temperature sensors”

“Response (ChatGPT):When it comes to air quality standards involving CO_2 and temperature sensors, several key standards and guidelines can be referenced:

1. NIOSH (National Institute for Occupational Safety and Health)
CO_2 Exposure Limits: NIOSH recommends a ceiling limit of 3,000 ppm (parts per million) for CO_2 in workplace environments to avoid potential health risks.
Heat Stress: Guidelines for temperature in occupational settings recommend maintaining temperatures to prevent heat stress, particularly in high-temperature work environments.

2. OSHA (Occupational Safety and Health Administration)
Air Quality Standards: OSHA does not set a specific CO_2 standard but follows NIOSH recommendations for safe exposure levels.
Thermal Stress: OSHA's standards emphasize maintaining acceptable thermal environments in workplaces to protect employees from heat-related illnesses.

3. ASHRAE (American Society of Heating, Refrigerating and Air-Conditioning Engineers)
Indoor Air Quality (IAQ): ASHRAE Standard 62.1 (Ventilation for Acceptable Indoor Air Quality) suggests maintaining CO_2 levels below 1,000 ppm in indoor spaces to ensure acceptable air quality.
Temperature Guidelines: ASHRAE recommends maintaining indoor temperatures between 68°F and 75°F (20°C to 24°C) in winter and 75°F to 80°F (24°C to 27°C) in summer for optimal comfort.

4. EPA (Environmental Protection Agency)
Indoor Air Quality Guidelines: The EPA provides general recommendations for maintaining indoor air quality, including monitoring CO_2 levels and ensuring good ventilation practices to control pollutant levels.

5. WHO (World Health Organization)
Air Quality Guidelines: WHO recommends keeping CO_2 levels below 1,000 ppm in indoor settings to minimize health risks associated with poor air quality.
Temperature Recommendations: WHO suggests that thermal comfort in indoor environments should be managed to promote health and productivity.

6. EN 13779 (European Standard)

Ventilation for Non-Residential Buildings: This standard specifies requirements for ventilation systems, emphasizing maintaining CO_2 levels below 1,000 ppm and managing temperature for comfort and health.

7. ISO (International Organization for Standardization)
ISO 16000 Series: These standards relate to indoor air quality and provide guidelines for measuring various pollutants, including CO_2. They also include recommendations for maintaining thermal comfort.

Practical Applications
When implementing CO_2 and temperature sensors in your smart vent system, consider adhering to these standards to ensure that your system promotes a healthy indoor environment. Monitoring CO_2 levels and maintaining optimal temperature can help mitigate health risks and enhance overall air quality" [113].

Prompt 3:

"How much current does scd41 consume?"

Response: "The SCD41 CO_2 sensor typically consumes around 70 µA in low-power mode and up to 120 µA during measurement mode. It also has a peak current of about 2.5 mA during warm-up. These values make it suitable for battery-powered applications where low power consumption is crucial" [114].

We have utilized LLM for drafting, outlining, comparing, summarizing, and proofreading purposes.