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

Emily Baross, Kyle Ingleton, Marcus Loyd, Patrick Tumbos

Dept. of Electrical Engineering and Computer Science, University of Central Florida, Orlando, Florida, 32816-2450

***Abstract* — Improving the living conditions in indoor living spaces has been a goal that industries have been working to solve. Particularly, irregular temperature distribution throughout indoor living spaces presents a significant issue, causing both discomfort and inconsistent room temperatures. B.R.E.A.T.H.E (Balanced Responsive Environmental Airflow and Temperature Housing Enhancer) is a smart ventilation system that has been specifically designed to address these concerns. By implementing an autonomous functionality that adjusts vent louvers according to indoor room temperature as well as giving users the ability to manually open or close them, B.R.E.A.T.H.E gives users more control over their indoor climate. Moreover, B.R.E.A.T.H.E also features a Carbon Dioxide alert system and a user-friendly interface for complete system control. This paper discusses the design methodology, component selection, and software implementation behind B.R.E.A.T.H.E.**

***Index Terms* — Autonomous, Bluetooth Low Energy(BLE), Carbon Dioxide, Indoor Air Quality, Smart Vent.**

I. Introduction

Reliable and efficient home ventilation is essential in today's society. These systems ensure comfort, maintain indoor air quality, and regulate temperature across residential, commercial, and industrial spaces. By optimizing ventilation control, B.R.E.A.T.H.E aims to enhance health and comfort while minimizing energy consumption.

B.R.E.A.T.H.E is a smart ventilation system that aims to provide a solution to these temperature concerns by incorporating both an autonomous feature that adjusts the vent's louver position based on environmental conditions and letting users manually open or close the vent. B.R.E.A.T.H.E will have two subsystems consisting of the control unit and the vent interaction subsystem, shown in Fig. 1.The control unit consists of a temperature and carbon dioxide sensor that will periodically gather environmental data and determine if the vent should open or close. Moreover, it includes an LCD screen that provides users with an intuitive interface while allowing full control of the system. Likewise, the control unit will also have a buzzer that will alert the user of hazardous CO_2 levels or low battery levels. In order to keep energy efficiency in mind, the control unit will be powered via wall power so that the user won't have to constantly buy new batteries for it. The control unit will communicate with the vent unit through the use of Bluetooth Low Energy(BLE). The vent interaction subsystem will be controlled by the control unit and through the use of a motor attached to the vent louvers to open and close the vent. Moreover, since this subsystem is battery powered, low power mode optimizations will be applied in order to maximize battery life.

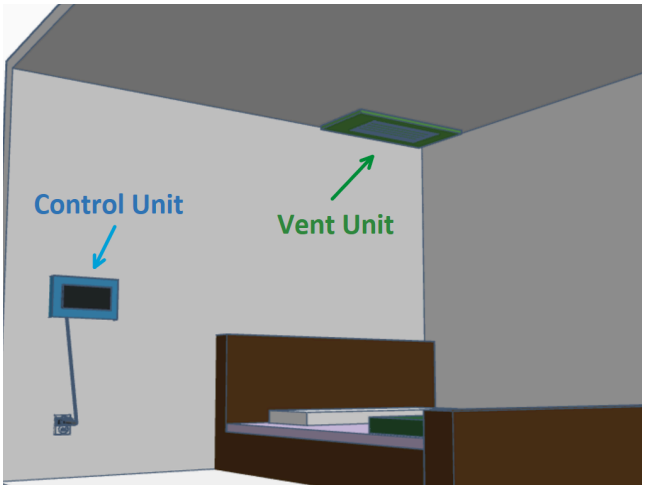

Fig. 1. B.R.E.A.T.H.E 3D system overview

II. Specifications

The specifications for B.R.E.A.T.H.E balance technical precision with user-friendly design, ensuring accuracy and intuitive design. With nine key specifications its robust design meets all requirements set by the Accreditation Board for Engineering and Technology(ABET).

The technical specifications for B.R.E.A.T.H.E are:

1) Manual Vent Control: The system should be capable of manual operation that fully opens or closes the vent through user input.

2) 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, meaning 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.
3) Vent Actuation Time: The vent should fully open or close within a defined time limit. The time limit for actuation is 5 seconds.
4) Bluetooth Range: Bluetooth communication should work within a specified range. The specified range for Bluetooth communication is that it should work up to 10 meters.
5) Sensor operating Range: Each sensor should have a sufficiently wide defined operating range to accommodate for extreme conditions. The chosen operating range for the Temperature sensor is -40 to 176 °F. The chosen operating range for the Carbon Dioxide sensor is 0 ppm to 5000 ppm .
6) 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.
7) 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. The Alarm will be triggered when the Carbon Dioxide levels exceed 4500 ppm, and an alarm will also be triggered when the battery of the vent interactive subsystem unit reaches below 10% capacity.
8) Battery Life: Unit should be able to operate for a certain length of time before a battery replacement. The minimum duration of battery life before a replacement is needed is 6 months.
9) Dimensions: The overall vent system's physical dimensions should fit within a specified range. The entire vent device should be able to fit within a 14 by 6 inch air duct.

Our most critical specifications that we will demo for presentation purposes will include, the autonomous vent control feature, the audio alarm system, and the battery life. The autonomous vent control feature will be shown by setting the desired temperature of the room on the control unit and then manipulating the temperature of the environment around it via a hair dryer on the heat setting and then seeing how increasing the temperature by 2°F will in turn actuate the vent autonomously to open all the way. The audio alarm system will be showcased in two ways since there are two different alarms, the first will be for high carbon dioxide levels and for demo purposes we will lower the threshold to be 1800 ppm so as to not wait for such drastic conditions at 4500 ppm and increase the CO_2 around the control unit, and then for the low battery alert we will disconnect the 9V battery in the vent and connect our vent interaction subsystem to a power supply from the senior design lab and lower the voltage output to around 6V as that is the battery's rated cutoff voltage as specified by the manufacturer datasheet. Finally, for demonstration purposes, to present the 6 month battery life calculation for the vent interaction subsystem, we will need to measure the average current consumption of each peripheral in the system and calculate the expected battery life while factoring in the time spent using said peripherals. This is because the MCU in the vent interactive subsystem will be idling the majority of the time, meaning we have to account for a discontinuous current load on our battery when determining how long it will last in typical use.

III. Hardware Selection

When selecting hardware for B.R.E.A.T.H.E it was important to the team that each component was carefully selected in order to fully meet our project specifications. As such, in the following text we will explain what component was chosen and why it was selected.

A. Microcontroller Unit

When selecting a Microcontroller Unit (MCU), there were three important factors to consider: power consumption, BLE, and ease of integration. To be able to accomplish the desired power consumption specification of a 6 month battery life we chose to select 2 different MCU's. For the control unit, since battery life was not crucial, we chose the ESP32, and for the vent interactive subsystem where the 6 month battery life was critical we chose the nRF52832.

The ESP32 was considered the most viable option for the team given our experience with it, its BLE capabilities, affordability, and compact form factor. Since as a team we decided to have the control unit be plugged into a wall outlet for power, the ESP32's high power consumption of 278-335 mA was not an issue. This MCU also includes two cores operating at around 160 MHz, allowing for fast computational and data processing benchmarks which are necessary for the heavy demands of the control unit. Additionally, the ESP32's support for a wide range of communication protocols such as SPI, I2C, and UART only further expanded our capabilities and ensured our myriad of peripherals the ability to communicate, making it an ideal choice for our project's needs.

For the vent interactive subsystem, the location of the vent essentially forced us to use battery power as opposed to grabbing power from an outlet as was done with the control unit. It was determined the ESP32 consumes far too much power when transmitting and receiving data through BLE, which led us to finding a different MCU that was just as easy to integrate but with less power consumption. The NRF52xx series was one of our first choices as it advertises extremely low power Bluetooth functionality and extensive low power modes. We chose the NRF52832 to run the vent interaction subsystem as it was one of the only MCU's in the NRF52xx series that had available modules with integrated chip/PCB antennas and internal crystal oscillators whilst still being reasonably simple to solder.

The low power BLE functionality does come at a cost, however, with the clock speed and available RAM which is 64MHz and 64 KB respectively. This doesn't pose much of an issue though, since the vent-interactive subsystem is only responsible for relatively simple and light tasks.

On the other hand, we save quite a lot of battery life. In its active mode, the NRF52832 draws about 3.3mA and offers a variety of low power modes, including a low power mode and deep sleep mode, both drawing 2.35µA and as low as 0.4µA respectively[1]. The nRF52832 will primarily be in low power mode, as this allows it to maintain a BLE connection while also utilizing a very low current consumption. This is significantly less demanding compared to the ESP32 and allows us much more room to reach our battery life specifications.

B. Temperature Sensor

In order to meet the specifications of ±1°F temperature sensor accuracy and an operating range of -40 to 176 °F, the team considered four separate sensor models with varying ranges and accuracies. The DHT20 sensor boasted a 0.5°C (±0.9°F) accuracy, a -40°F to 176°F temperature range, and came pre-manufactured with full calibration. Additionally, it has a voltage range between 2.2V and 5.5V, which is well-suited for integration with the ESP32. Since the DHT20 is a digital sensor that communicates using the I2C protocol that is natively supported by the ESP32, and also reaches 63% of the final temperature value in just 0.8 seconds, selecting this temperature sensor was the obvious and well-informed choice.

C. Carbon Dioxide Sensor

The Carbon Dioxide Sensor was also held to high standards like the temperature sensor and required an operating range of 0 ppm to 5000 ppm and ±50 ppm sensor accuracy. Again, the team considered four different sensor options and discovered the SCD41 advanced Non-Dispersive Infrared (NDIR) sensor, which provides a measurement range of 0 to 40,000 ppm, and an impressive precision of ±40 ppm making it the clear choice due to its highly accurate capabilities.

D. LCD Screen

To provide an attractive and easily accessible means of allowing the user to both control and receive information from the system, we chose to use a touchscreen LCD screen. When choosing a specific screen, the team carefully considered the type of touchscreen technology, screen size, and resolution as well as the communication protocols utilized by the built-in LCD and touchscreen drivers.

Though they are less precise than other options on the market, we found resistive touch screens to be perfect for use with the control unit due to its widespread commercial use and low cost. The team found that the consumer likely would not appreciate or notice more precise and responsive touch capability and that it would likely not add to the function of the system as a whole.

When it comes to resolution and screen size though, we needed to find a balance between making the unit itself compact while retaining the clarity of the graphics and information contained on the LCD screen. We also needed to make sure these graphics were not too crowded so that the user would be able to interact with the LCD screen without making unintended selections.

Our final considerations were based on the communication protocol. It was necessary to utilize a protocol that was compatible with our MCU and simple to use. I2C and SPI were preferred as they are supported by the ESP32 and utilize very few GPIO pins.

Weighing all these specifications, we decided to use the WaveShare 4" LCD Screen as it gave us adequate screen real estate and resolution whilst using SPI as its main protocol, reducing the amount of pins we need to interact with on the MCU to control it. This LCD screen is also easy to develop as it routes pins out to regular pin headers instead of utilizing a ribbon connector like many other options on the market.

E. Buzzer

While choosing a suitable buzzer, reliability and adequate sound volume were important characteristics for the team. A buzzer that would be loud enough to efficiently alert the user of either a toxic level of Carbon Dioxide or a low battery life made this peripheral an important one in regards to user safety. We opted to use the PS1240 piezo buzzer, as it has a long life expectancy and is very simple to integrate.. The PS1240 operates at a minimum voltage of 3V and requires a PWM signal to

function, which we can easily get from an analog pin on the ESP32. Since we found that using the buzzer at 3.3V to be a little too quiet to properly alert the user of potential danger, we decided to additionally use a MOSFET amplifier circuit to give the buzzer a louder noise floor.

F. Motor

To properly move the vent louvers to open or close the vent on autonomously, the team needed to consider different motors that would be able to achieve this whilst still allowing our other specifications to be sufficiently met. The motor would need to have enough torque to move the vent louvers and also have low power consumption in order to preserve battery life.

As a team, we decided servo motors were a perfect fit for this project as their built in drivers were effective for the use case of the project and they provided enough torque to adequately move the louvers. Of the many different models of servo motors, we needed to select one that would have sufficient torque and a size small enough to reasonably fit inside of the vent. We initially chose the MG996R, but our final choice was the S3004 servo motor as it presented a good balance between torque, power consumption, and size. It only draws ~40-200 uA depending on the mechanical load and retains a torque of 0.314 Newton-Meters. Although the current draw is a little high, it wouldn't have much of an impact on the overall power consumption due to only briefly operating with long interval breaks in between. Additionally the stall current isn't high enough to damage the linear regulators, which is the main reason why we chose this.

G. Power Supply

Two types of power sources will be utilized for the B.R.E.A.T.H.E system: the control unit will be using a wall outlet as its power source while the vent unit will need to operate off of battery power. This necessitates two different approaches to each subsystem's power supply scheme.

Due to the vent subsystem's location, we needed a reliable and high capacity battery to power the peripherals on this unit and ensure the user doesn't have to replace the battery very often. The location furthermore ruled out the use of rechargeable batteries, as it would be difficult to find a convenient method of charging the batteries. This boiled our choice of battery down to purely primary (non-rechargeable) batteries. We chose to go with a lithium-metal battery as they have relatively high capacities and are not very prone to leakage.

For the vent interaction subsystem, we need at least 5V to power all of our peripherals, and this necessitates a battery with an operating voltage at least 500mV or so above 5V. With consideration to size and capacity, we chose a 9V lithium metal battery as our primary voltage source for the vent subsystem. The batteries 1200mAh capacity is very respectable for a disposable battery, and its cutoff voltage is high enough that we can use the full range of the battery before it dies.

Both the control unit and the vent interactive subsystem require a 5V and 3.3V rail to supply power to our various peripherals. To do this, we will need an efficient voltage regulation scheme to provide clean power to each unit. The specific regulators used between each unit are different since the primary voltage sources differ greatly. The control unit is powered through a 12V AC-DC power adapter connected to a wall outlet, whereas the vent unit is fully powered by a 9V battery. Our battery specifications therefore require a better t voltage regulation scheme to maximize battery life. Just in case of potential power-related issues, we also decided to have the power supplies of both units on different PCBs connected via wires to the main boards.

For the control unit, we used the LM2576 switching regulator to provide the 5V rail for the LCD screen and an LM1117 Low Dropout (LDO) regulator to provide 3.3V to the MCU and other peripherals. A switching regulator was chosen in this instance to avoid potential thermal issues when stepping down 12V to 5V. The LM2576 fulfilled our current requirements whilst being efficient enough to avoid thermal concerns. It is also not nearly as sensitive to PCB layout as other regulators, making it robust and more than sufficient for the power supply on this unit.

For the vent unit, however, we needed even more efficiency and a much lower quiescent current to have any chance of meeting our battery life specifications. As a result, the team decided on using the AP63205 to provide the 5V rail to power the motor and the XC6206 LDO to supply 3.3V to the MCU. The AP63205 is more efficient than the LM2576 over a wide range of output currents whilst boasting a low 22μA quiescent current [2]. This is much more reasonable for the vent subsystem compared to the 5-10mA of quiescent current the LM2576 draws [3]. The AP63205 is also not very sensitive to PCB design, though it does use much smaller components. We opted to stick with an LDO for the 3.3V rail on this unit in order to lessen the chance of potential issues since buck regulators can be quite sensitive. The LM1117 was not chosen for this unit due to the 1.7-5mA quiescent current it draws [4]. Alternatively, the XC6206 draws only 1μA of quiescent current, making it perfect for maximizing battery life [5].

IV. Hardware Block Diagram

A visual to show how all of the hardware components involved within B.R.E.A.T.H.E come together is displayed in Fig. 2. To begin, the control unit takes in 120VAC from a wall outlet and then via an AC adapter it is converted to 12VDC which connects to our power supply PCB through a barrel connector. From there the 12V gets sent to the buck regulator that steps down the voltage to 5V for the LCD screen and for the LDO. The LDO drops down the voltage again to 3.3V to provide power to the ESP32 (MCU) and remaining components such as the Carbon Dioxide Sensor and the Temperature Sensor. Afterwards, the ESP32 interprets data coming in from the two sensors and from the touchscreen LCD, and with this data it will then determine whether to send data back to the LCD, send a wireless Bluetooth signal to the Vent interaction subsystem, or a PWM signal to the MOSFET amplifier for the Buzzer.

The nRF52832 uses BLE data received from the ESP32 to determine if it should open or close the vent using a PWM signal. If it is necessary to turn on the motor, it will send a GPIO output to the MOSFET switch to connect the motor to the power supply. Diving deeper into the vent interaction subsystem's layout, The power supply consists of a 9V battery which provides a 5V rail through the use of a buck switching regulator. That regulator is then further dropped down to 3.3V via an LDO which supplies power to the NRF52832. To accurately read battery life and alert the user of a low battery, we tied a buffered voltage divider circuit to an ADC input on the NRF52832 which will periodically read the value and send an alert to the ESP32 through BLE if the battery is near its cutoff voltage. The last main element of this subsystem is a MOSFET switching circuit that goes between the 5V output and the servo motor. This is a high side switch which will cut power to the motor when not in use. This prevents the motor from stalling and consuming excess current while it holds a specific angle. As outlined previously, this switch is controlled by the NRF52832.

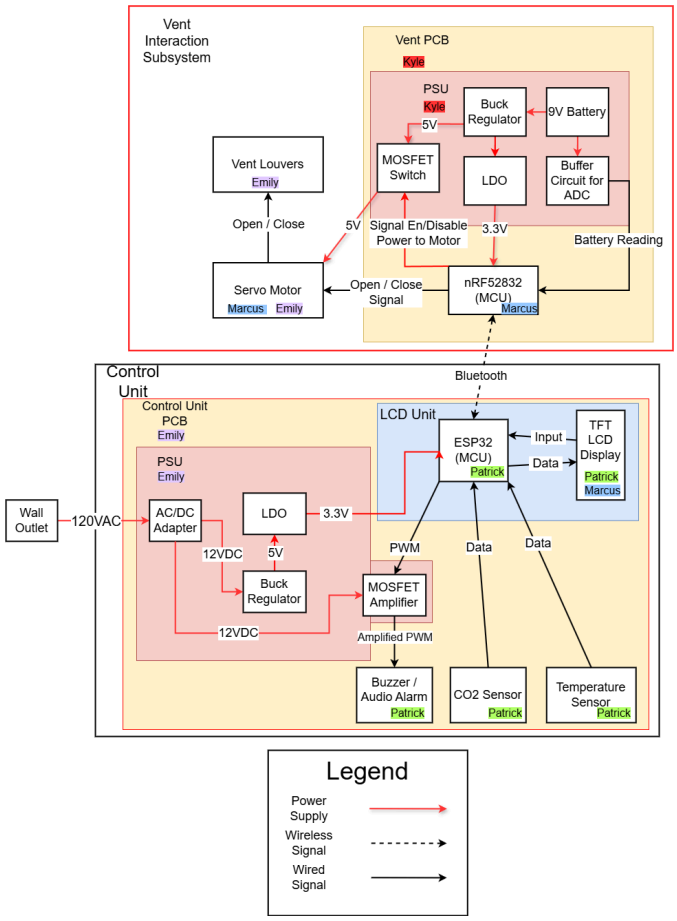

Fig. 2. B.R.E.A.T.H.E Hardware Block Diagram

V. Software

The software for B.R.E.A.T.H.E can be broken down into two main portions: development for the Control unit which utilizes an ESP32, and development for the Vent unit which utilizes an nRF52832. Each unit has their own unique development environments due to the individual constraints each one had to focus on, as well as the advantages and disadvantages that each environment is associated with.

The two units have a few similarities, including the use of Bluetooth Low Energy (BLE), which is the source of communication between the two. This involves a publisher-subscriber model, where each unit subscribes to a common service to establish an initial connection with one another. Once they are connected they are able to create and subscribe to common characteristics where they can upload information to. Once this is done, the other unit will automatically gain an alert to check the information and be able to interpret it. This is useful

because it allows the MCUs to communicate with each other while maintaining efficient power consumption.

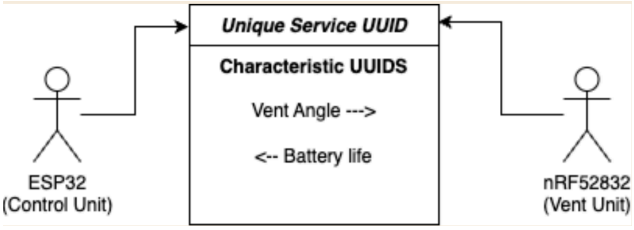

Fig. 3. Picture describing the publisher-subscriber relationship between the two MCUs through BLE.

A. Development Environments for the nRF52832

For the nRF52832, we had two main IDE's to choose between when developing code: the Arduino IDE, and the SEGGER IDE paired with Visual Studio Code. The Arduino IDE contains a high level of abstraction, allowing us to utilize the many libraries and quickly develop and test code on the MCU. A major downside is the higher power consumption that comes with it. Due to the high abstraction, the nRF52832 consumes a higher current consumption. Additionally, in order to run the Arduino libraries, Adafruit firmware was necessary, which draws about an additional 4mA. In our use case, an increase of even 1mA of current would significantly decrease the battery life of the vent unit. Due to these reasons the Arduino IDE was only used for quick prototyping, allowing for subsystem integration testing to be performed.

For the final software integration, the Nordic SDK was used in conjunction with SEGGER IDE and Visual Studio Code. This software development kit was created by the manufacturer of the MCU and allowed for the use of libraries and example code to develop bare metal code, allowing for complete control over the MCU and utilization of low power modes. According to the datasheet, the nRF52832 consumes about 1.2 µA when in System ON mode with full RAM retention [1]. By using the Nordic SDK, the MCU in testing has shown to consistently use less than 1 mA of current, allowing for the battery life specification to be met.

The SEGGER IDE allowed for code to be flashed onto the nRF52832 with the J-Link EDU Mini, and Visual Studio Code allowed for easy text manipulation and file navigation, both of which are necessary when working with the Nordic SDK. Project files are saved as '.emProject', which includes directories and files that are referenced in the build configuration similar to a cmake file. The SEGGER IDE can reference these '.emProject' files and automatically import the referenced directories and files, allowing for easy compilation and flashing. Visual Studio Code lacks the necessary tools for efficient compilation and flashing, but it has incredibly fast and user-friendly file navigation and development tools that allows for the code to be created and edited in a timely manner. For these reasons, these two development environments were used for developing the demonstration code.

B. Software Architecture for the Vent Unit

The software logic for the nRF52832 is relatively straightforward. It stays in low power mode until it receives a vent status message from the Control unit. When this happens, it sends a HIGH signal to the switching MOSFET circuit which allows for the motor to connect to the 5V line, and then sends a PWM signal directly to the motor to allow for the angle of the louvers to be adjusted. Once this is achieved, it goes back to low power mode. Every 2 minutes it temporarily leaves low power mode to check the current battery life via the NRF52832's onboard ADC. It then publishes this information to the 'Battery Life' characteristic before going back into low power mode until another event is triggered.

The Nordic SDK provided lots of libraries made specifically for setting up the BLE connections. A premade UUID was generated and denoted in little endian format, which was then passed into the *sd_ble_gatts_service_add* function to establish an initial service to search for. From there, the characteristics were added by initially creating a ble_uuid_t struct, and passing the UUID into the .uuid parameter. Read and write permissions are also assigned in this struct. They are then passed into a ble_gatts_attr_t struct, which is allocated the appropriate memory and then passed into the *sd_ble_gatts_characteristic_add* function to associate it with the service. From there, the on_write function handles any incoming information, which will primarily be for the vent angle in B.R.E.A.T.H.E. After verifying that the message is received from the vent angle characteristic, it will call and pass the appropriate vent angle into the *move_vent* function, which will send the corresponding angle to the motor.

The servo logic in the software consists of two main tasks: sending a GPIO signal to the MOSFET switch, and enabling / adjusting the PWM signal sent directly to the motor. The GPIO signal is relatively straight forward and consists of configuring the pin to output via the nrf_gpio_cfg_output function, followed by setting it once the nRF52832 receives the angle characteristic, then disabling it once the signal has been sent. For the PWM signal, an instance 'PWM1' is created and associated with TIMER1 through the *APP_PWM_INSTANCE* function.

The GPIO 6 pin is configured through the APP_PWM_DEFAULT_CONFIG_1CH function, which associates the pin with a 200 Hz frequency. The PWM1 and config instances are then passed into the app_pwm_init function to fully set up the functionality. From there, it can be enabled or disabled when needed. In B.R.E.A.T.H.E's use case, it will disable the PWM signal after 1 second, as this would be enough time to get the vent louvers to the desired state.

Additionally, the battery life is monitored through an ADC and the corresponding status will be transmitted to the control unit via the 'Battery Life' characteristic. The ADC works by configuring a separate timer, TIMER2, to enable sampling periods. Once the timer is configured, the nRF52832 then initializes the ADC by configuring the AIN1/A1 pin to read voltage, and the *saadc_callback* function is called anytime an interrupt occurs that requests data. The function handles the logic for reading the battery voltage by reading the raw ADC value, then converting it into a float after multiplying it with the proper scale factor. This is obtained through testing. By hooking up the ADC to a power supply and watching the actual voltage vs the observed voltage, we are able to divide the two and get the appropriate scaling factor. This value is then converted to a string, and the logic for interpreting the value will be taken care of on the control unit.

For all of the above, an updated configuration file is necessary. This contains lot of *#define* statements which enable or disable peripherals, such as specific timers. Working with this can be tricky as conflicting configuration files would cause the nRF52832 to reset whenever it would come across an error. This is why it is very crucial to have an accurate configuration file, so that the drivers can properly function with the intended purpose. Additionally, the Vent unit does not include an external crystal on the module or the PCB. Due to this, the configuration file has to ensure that the nRF52832 utilizes its internal RC oscillator by defining *NRF_SDH_CLOCK_LF_SRC*, otherwise it will result in an error and cause the MCU to continuously attempt to reset.

C. Software Architecture for the Control Unit

The following describes the software design and methodology behind the control unit subsystem.

The software retrieves data from both the SCD41 and DHT20 via I2C. Hardware-specific libraries for both sensors were also utilized to simplify integration. These libraries were particularly useful as they provided built-in functions that handled data retrieval, error handling, and I2C processes. Since each sensor had different response times, a consistent delay value of two seconds was chosen in order to synchronize the readings taken. Moreover, since the default temperature readings that are gathered by the DHT20 are read in Celsius, the readings were converted to Fahrenheit using the following formula:

$$F = C \times \frac{9}{5} + 32 \tag{1}$$

The sensor values would then be outputted to the LCD screen.

Moreover, the LCD screen was responsible for displaying two key alerts: low battery and elevated CO_2 levels. The implementation of the CO_2 warning was done by establishing a predefined threshold and adding hysteresis to prevent rapid fluctuations and false positive readings (i.e, a sudden spike in CO_2 levels that quickly return to normal on the next reading). This was done by storing the five most recent readings into an array and calculating their average. Then, the average value is compared to the predefined threshold. If the reading is above the threshold, then the buzzer will trigger and display a bright red alert box on the LCD screen, alerting the user to high CO_2 levels. Once the average CO_2 level subsides below the threshold, the buzzer is then turned off.

The system also monitors battery levels by periodically transmitting analog voltage readings via BLE. The nRF52832 reads the battery voltage of the vent subsystem at regular intervals and sends these values to the control unit. Similar to the CO_2 alert, hysteresis was implemented by storing the last 10 transmitted values in an array. The system then calculates the average of these values and compares it to a predefined threshold. If the average falls below the threshold, the ESP32 would continue to check subsequent readings through a 5-second interval. If these readings remain below the threshold, the buzzer would activate.

Since elevated CO_2 levels can pose a serious safety concern, a more urgent and loud alarm tone was selected. A loop was created to generate a linear frequency sweep from 4000 Hz to 5500 Hz over a span of one second. This alarm would continuously play until the average CO_2 level would go below the threshold. For the low battery alert, a less urgent tone was made. Musical notes were referenced and transcribed into defined frequencies for the buzzer. Subsequently, they were arranged to play *Beethoven's 5th symphony*. Given that this alert isn't necessarily as dangerous as high CO_2 levels, an option on the LCD Screen to stop the alarm would appear for users to press.

The Control unit also allows users to open or close the vent either autonomously or manually. If the user wishes to control the vents autonomously, the vent mechanism acts based on a user-defined threshold. Whenever the

threshold is exceeded by ±2 degrees Fahrenheit or higher, the vent subsystem will either send an open or close signal via BLE to the control subsystem. Conversely, in manual mode, the user is able to press an open or close button on the LCD screen. Based on the selected option, a corresponding string is sent via BLE to the vent subsystem, where it is processed accordingly.

The TFT_eSPI was a crucial library that handled much of the UI implementation and functionality of the LCD screen. A wide range of pre-defined functions made rendering shapes, colors, and sprites straightforward and efficient.

In order for the TFT_eSPI library to be interfaced with the LCD screen, a user setup file had to be configured. Various parameters such as the SPI pins (MISO, MOSI, and SCLK), touch pins, and graphics controller had to be defined in order for the LCD screen to work. Without properly setting these parameters, the LCD screen would not initialize or function correctly. Additionally, optional features like display rotation, screen resolution, and font loading could be adjusted in the setup file to further customize the display behavior and performance.

The font selection given by TFT_eSPI was quite limited, but fortunately it supported the implementation of custom fonts. Custom fonts were created by converting a downloaded font into an array of hexadecimal values. However, since the converted arrays were thousands of lines long, it consumed a significant amount of memory. At a certain point, the control unit program couldn't be compiled since the ESP32's memory constraints were far exceeded. This posed a major concern, as initially the ESP32's flash and RAM could not accommodate BLE handling, sensor integration, and display assets all at once. To mitigate this, the partition scheme was changed in order to allocate more memory for the application. By increasing the available program storage space through a custom partition layout, the system was able to successfully compile and run without sacrificing crucial features.

Touch input was handled by the TFT_eSPI library by using a provided function and defining an area that the touch can be applied to. Visual elements such as boxes, background colors, and sprites were also rendered using functions in the TFT_eSPI library.

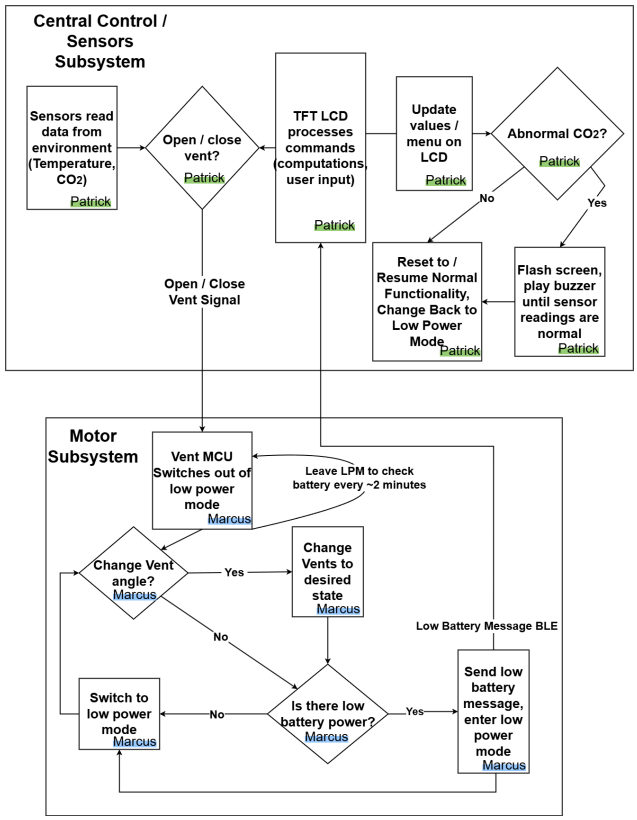

Fig. 4. B.R.E.A.T.H.E Software Flowchart

VI. Conclusion

The B.R.E.A.T.H.E smart ventilation system is a device that aims to give consumers more control over the comfort of their own home. By controlling the state of a room's air passageway, users can monitor and achieve their desired temperature. Through the use of multiple sensors, B.R.E.A.T.H.E can measure the room's environment and perform various tasks, such as detect when the vent louver angles should be adjusted for optimal temperature, perform a loud audio alert in the scenario of high CO_2 and also perform an audio alert based on low battery. B.R.E.A.T.H.E also uses smart and efficient power saving techniques to reduce the power consumption of the vent subsystem, keeping the unit functional for as long as possible whilst keeping the consumer from having to buy new batteries very often.

In the future, this project can be expanded by implementing multiple B.R.E.A.T.H.E units in different rooms of a household due to the scalability of the foundation of the project.

B.R.E.A.T.H.E aims to integrate these features in an easily accessible, non-intrusive, and attractive package that will not require adapting to existing household HVAC

systems or having to make changes to household wiring. Furthermore, this project provides better options for controlling household ventilation for those who physically cannot adjust the vent louvers themselves.

Acknowledgement

The authors wish to acknowledge the guidance and support of the professors who have helped throughout this journey. Firstly, we would like to thank Dr. Qun Zhou Sun for graciously sponsoring this project and initially presenting the idea to our team. We would also like to thank Dr. Chung Yong Chan, Dr. Arthur Weeks, and Dr. Saleem Sahawneh for their mentorship and advice while designing and building this system. And lastly, we would like to thank our committee members Dr. Mike Borowczak, Dr. Sazadur Rahman, and Dr. Elizabeth Coln for taking their time to review our project.

References

[1] Nordic, "NRF52832 Product Specification v1.8," nRF52832 datasheet, Nov. 2021. https://www.mouser.com/datasheet/2/297/nRF52832_PS_v1_8-2942485.pdf?srsltid=AfmBOorL5PcxrWRiNmH8xARxF6n_ncGPFGY5BFUb7w_xPYCyco2UJ8w

[2] Diodes Incorporated, "3.8V TO 32V INPUT, 2A LOW IQ SYNCHRONOUS BUCK WITH ENHANCED EMI REDUCTION", AP63200 / AP63201 / AP63203 / AP63205 Datasheet, November 2024 https://www.diodes.com/assets/Datasheets/AP63200-AP63201-AP63203-AP63205.pdf

[3] Texas Instruments, "LM2567xx Series SIMPLE SWITCHER Power Converter 3-A Step-Down Voltage Regulator", LM2576/LM2576HV Datasheet, Mar. 2023

[4] Texas Instruments, "LM1117 800-mA, Low-Dropout Linear Regulator", LM1117 Datasheet, Jan. 2023 https://www.ti.com/lit/ds/symlink/lm1117.pdf

[5] Torex, "Low ESR Cap. Compatible Positive Voltage Regulators", XC6206 Datasheet. https://product.torexsemi.com/system/files/series/xc6206.pdf