

1 Cover Page

Senior Design Group 10: Home Security II

David Garcia Gonzalez, Gabriel Quintero, Jason Maxwell, Jared Bowling

2 Table of Contents

[1 Cover Page](#)

[2 Table of Contents](#)

[3 Introduction](#)

[4 Detailed System Requirements](#)

[5 Detailed project description](#)

[5.1 System theory of operation](#)

[5.2 System Block diagram](#)

[5.3 Sensors Subsystem](#)

[5.4 Main Hub Subsystem](#)

[5.5 Power Subsystem](#)

[5.7 User Interface Subsystem](#)

[6 System Integration Testing](#)

[6.1 Describe how the integrated set of subsystems was tested.](#)

[6.2 Show how the testing demonstrates that the overall system meets the design requirements](#)

[7 Users Manual/Installation manual](#)

[8 To-Market Design Changes](#)

[9 Conclusions](#)

[10 Appendices](#)

3 Introduction

Keeping one's home safe is a challenging problem within the United States. According to the FBI 2.5 million burglaries are committed annually, or in other words, a break-in occurs every 26 seconds. Once a burglary has been committed, the chances of the case being solved are very slim with police only solving around 13% of reported cases. Due to this, home security systems are one of the most vital methods of combating criminals. Oftentimes its presence alone serves as a deterrent as it was found that 83% of potential burglars check for a security system before any attempt to break-in is made. Besides break ins another threat facing house holds are fires. The U.S. fire administration in 2021 estimated that there was 353,500 fires among residential homes costing \$8,855,900,000 in damages. However, many homeowners within the United States do not have a form of a home security system to protect against these threats. National surveys found that 46.9% of people don't have a home security system installed in their home. One of the common factors cited for lacking a home security system was cost. Many of the popular mainstream security systems cost hundreds of dollars such as Ring making it unaffordable for low income families.

As such our group set out to provide a security system that would not only accurately identify these threats but also a system that was cost-effective. This would be achieved through allowing the system to be modular in nature. If you want to combat burglars you can purchase a motion sensor now to detect break-ins and later on purchase a gas sensor to detect fires. Through this method, the system would be more accommodating to customers of all backgrounds. This was unlike many of the existing security systems which often required a "set" to be purchased raising the cost with some sensors that home-owners might not even need. As this would primarily aid those with lower incomes, most of the system work was designed on the backend. This would make it easier to install and use for those who might not be as technologically literate as they would be interacting with a user-interface on a website. Not only would this allow for the user to check on their home's status wherever they are but it also would allow for notifications to be sent directly to them through email.

Overall the system was able to meet the requirements we set out to do. The total cost of the system was roughly half of the other systems on the market as well as the system's ability to detect and quickly notify the user when a sensor was set off. One area that could be improved upon is the power system. As cost was a priority, rechargeable batteries were utilized in order to power the sensors. As a result of this design choice, the time needed before the system needs to be recharged is smaller compared to other market options. However, the batteries still provide a reasonable lifetime expected of a security system lasting around half a year. In addition we mitigated the hassle of checking on the batteries by having the system notify the customer when a sensor battery is low.

4 Detailed System Requirements

The first requirement of our system is to provide the users the ability to mix and match the home security system that fits their needs. To meet this requirement, we created a system that is composed of two elements: a main hub that provides the connectivity between the sensors and the user interface, and a collection of sensors that can be selected by the user to fit their needs, depending on the house size (i.e. bigger houses that require more sensors) and concerns (i.e. more concerned about break-ins than fires). Users buy a main hub and then select which and how many of the sensors they want.

The second requirement is to provide the users the ability to easily install the main hub and the sensors. The main hub needs to be able to connect to the user local network. The sensors need to connect to the MAC address of the ESP NOW ESP32 in the main hub. This information needs to be provided by the user after purchase. Users should be able to open their phones, connect to their newly bought device (whether a main hub or a sensor) and add the important information for the device operation (MAC address, user ID, SSID, password). After this, the device should be able to join the home security network and work seamlessly.

The third requirement is to provide the users the ability to manage the home security system anywhere with an internet connection. This requires the design of a backend structure for all the home security information, a hosted front-end user interface accessible from the browser, and an API to request and update the database information, from the main hubs and the users.

The fourth requirement is to provide sensors that accurately measure the home security parameters that the users are interested in. Currently, we are working on three sensors: a magnetic sensor that provides the user security against break-ins in opened doors and windows, a movement sensor and a gas sensor that measures both fire hazards and gas intoxications.

5 Detailed project description

5.1 *System theory of operation*

To achieve the requirements laid out previously, the following system was designed. Figure 1 contains an overview of the different high level components that integrate the system. The system starts with the sensors, these are wireless, battery powered, small boards that perform a unique and specific function related to the security of the home (like gas monitoring, movement sensor and magnetic sensors for doors and windows). The wireless sensors allow the user to place them easily around the house, without having to worry about wiring everything to a main system that becomes costly and harder to install without professional help. This makes the entire system more accessible and ergonomic to customers. The sensors consist of essentially two parts, a sensing piece and a processing unit in charge of communication with the Main Hub. When the sensors detect an event, they are triggered and send the notification to the Main Hub.

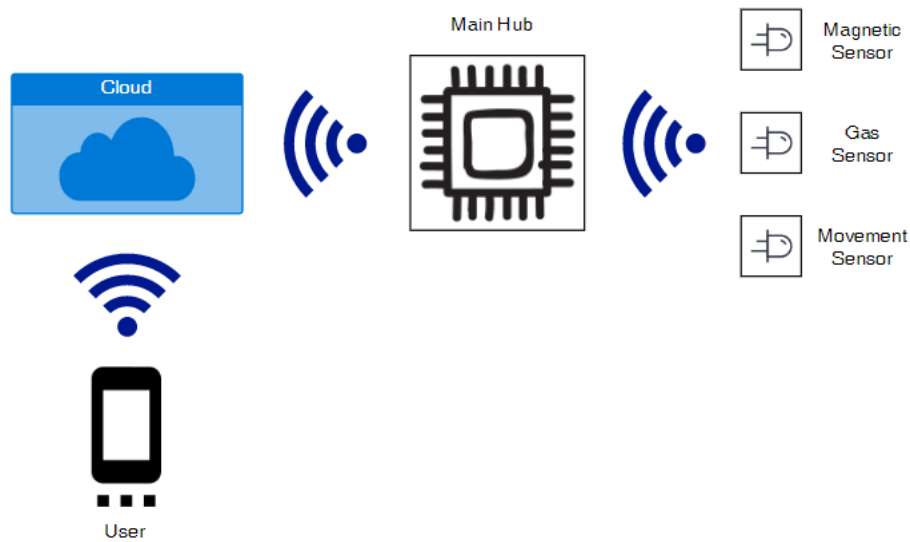


Figure 1. Diagram of Operation.

At this point in the operation flow the sensor event that was triggered arrives at the Main Hub, which collects all the sensor information across sensor nodes in the home and is in charge of communicating with the Home Security database. Upon receiving the alert from the sensor, the Main Hub has a set of API calls that lets it trigger the specific sensor that was triggered.

The Cloud component of the system contains two subsystems that will be explored in more detail later in the report: the user interface and the backend. The Main Hub calls the API to report a sensor trigger and changes the status for that sensor in the database and notifies the user of the sensor trigger.

The User receives email notifications regarding the status of their sensors, but it can also access the website for the system and check the status of the different sensors, as well as disarm and arm the sensors.

5.2 System Block diagram

The system is split into five subsystems: the sensor, the main hub, the power subsystem, the backend and the user interface. Each of these subsystems will be explored, including the components that form them and the interfaces between each of them. Figure 2 shows a high level diagram of the subsystems and which ones are connected to each other.

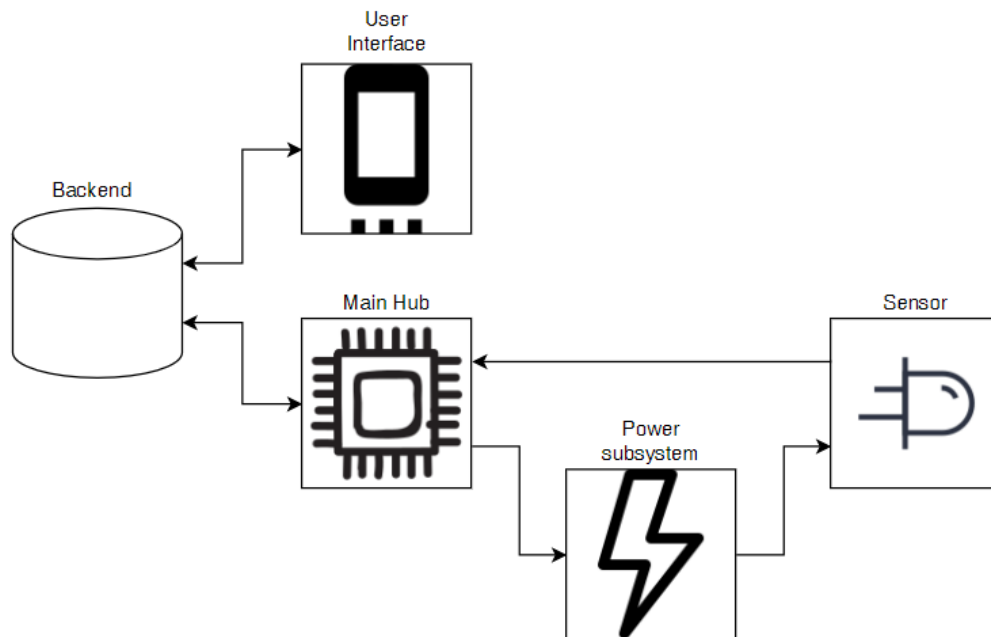


Figure 2. Subsystem diagram.

The sensor subsystem consists of the sensor boards, specifically the joint operation of the sensing element and the processing unit that was described above. It receives environment data, like the presence of a particular gas in the air, the movement of an intruder across a door, or the opening of a window or door. That data is digitized so an event of interest can be detected and trigger the sensor. The processing unit is in charge of receiving that data from the sensing element and when it is triggered, it communicates with the Main Hub the event. Since the sensor is battery powered, once the battery has been depleted it can be charged by connecting the sensor node to the Main Hub. An

important detail of the sensor nodes is that the client might purchase the sensors separated from the Main Hub, this means that the sensor board requires an initial configuration that allows the user to register the Main Hub information (in this case the MAC address) in a client-friendly way. Figure 3 shows the interfaces for the sensor subsystem.

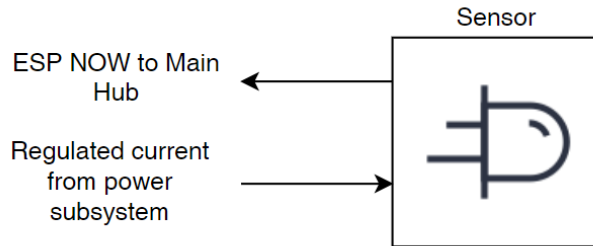


Figure 3. Interfaces for the sensor subsystem.

The Main Hub subsystem is in charge of coordinating the different sensors and executing the HTTP requests to send the sensor triggers back to the database. It consists of two processing units, connected by an I2C line: one is in charge of receiving the communications from other sensors and the other communicates with the API. The Main Hub then is connected via ESP NOW communication protocol with all the sensor nodes and by WiFi with the system backend. Figure 4 shows the interfaces for the Main Hub subsystem.

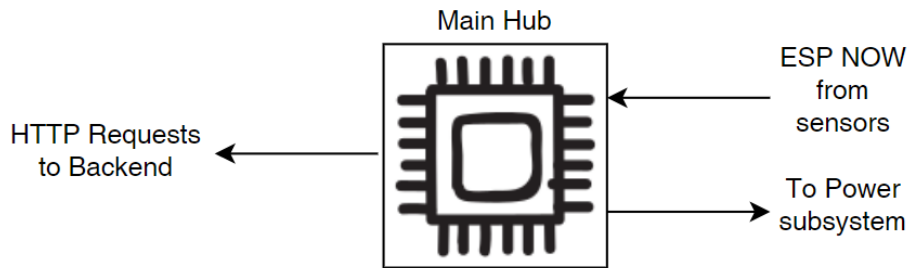


Figure 4. Interfaces for the Main Hub subsystem.

The power subsystem cuts across the physical design and it has elements in both the sensor nodes and the main hub. It consists of the battery of the sensor node, the battery charger chip and the

power distribution for that chip and battery coming from the Main Hub. The selected battery in this iteration for the sensor boards is a 3.7 V 6000mAh battery. This means that the 0.5 C (half-capacity) charging current for the battery is 3 A, which needs to be provided from the Main Hub power cord and regulated by the charging chip. Figure 5 shows the interfaces for the Power subsystem.

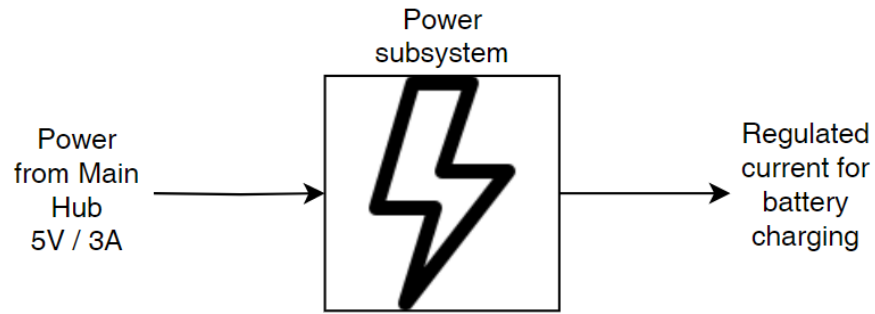


Figure 5. Interfaces for the power subsystem.

The Backend subsystem has a wide set of responsibilities and constitutes the most complex subsystem, whose architecture will be covered later in the report. This subsystem is completely defined in software. From the outside, it is just a collection of API routes that allow the user interface and the Main Hub to execute HTTP requests and receive or send relevant system information, like arm/disarm sensors, receive triggers, etc. In the inside, the architecture needs to hold the information of each user and the sensors, ensure secure transactions between the user and the database and allow the information to be updated by the user interface and the main hub. Figure 6 shows the interfaces for the backend subsystem.

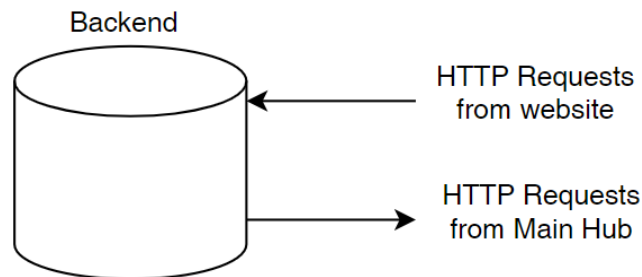


Figure 6. Interfaces for the backend subsystem.

The user interface subsystem is the client-facing interface that allows the user to check the status of the sensors, add new sensors, register and change the status of the sensors. It is fully defined in software. It contains the following features: secure sign-up/sign-in to ensure that the users data is secure and the home security system is blinded by malicious actors, continuous automatic update and display of the sensors' status and acknowledgement of sensor triggers and arming and disarming capabilities. All the functionality that requires interacting with the database is done by the API routes explained above through HTTP requests. Figure 7 shows the interfaces for the user interface subsystem.

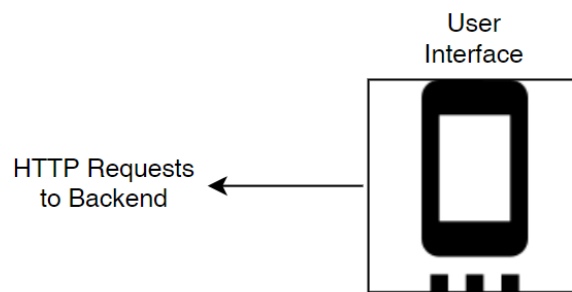


Figure 7. Interfaces for the user interface subsystem.

5.3 *Sensors Subsystem*

The three most important requirements that the sensor subsystem must meet are accurately detecting the events relevant to the security of the home, providing wireless communication with the Main Hub, and being battery powered to allow for easy placement. These three requirements frame our design decisions both in hardware and software for this subsystem.

Figure 8 shows the high level schematic for the sensor subsystem, the device specific schematic can be found at the end of the document. The sensors we chose require 5 V input for the gas and the movement sensor. Since the selected battery provides 3.7 V, a boost converter is used to raise the voltage from 3.7 V to 5 V. The ESP32 in the Sensor subsystem also requires constant 3.3 V, so a regulator is placed between the ESP32 and the battery. Additional elements for the subsystem include header pins for programming, battery connection, charging connection to the Main Hub and sensor attachment.

Simplicity is one of our design principles in this project, so we decided that users should be able to receive easy to understand and analyze information about their home status, so most of the sensors are binary oriented, either an event is detected and the sensor is triggered, or an event is not detected and the sensor does not trigger. This means that the sensors communicate with the processing unit by a single digital line, that either is held high or low to signal the trigger of the sensor.

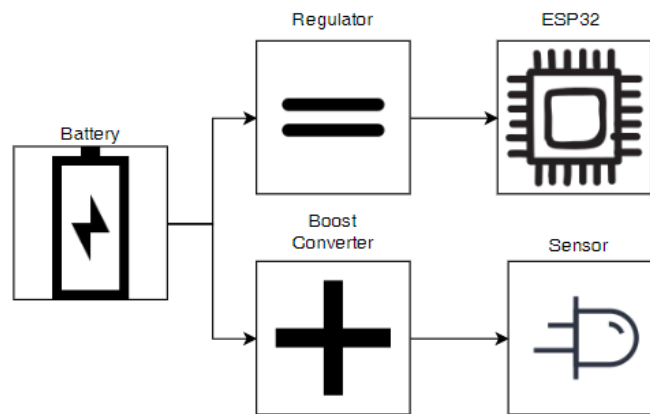


Figure 8. Schematic for the Sensor subsystem.

Since some of these sensors might be placed in zones of the house that are not easily accessible, and that accessing and manipulating these sensors should be done the least possible to guarantee a seamless user experience, the battery life becomes a key factor. The current draw of the sensor needs to be reduced as much as possible to extend the battery life. This drives our first decision of operating the sensor ESP32 in deep sleep mode most of the time. In deep sleep mode, most of the peripherals

of the ESP32 are turned off, including the high current consuming WiFi and ESPNOW antenna module. The ESP32 allows to define wake up sources while in deep sleep. Two sources achieve two different functionalities: a GPIO source to wake up the ESP32 when the sensor trigger line goes high/low and a timer wake up to check battery voltage and emit low battery communication.

The decision to use deep sleep as a method for saving power frames the way the code works. Figure 9 includes the code flow for the sensor node. On wake up (or reboot), the first check is whether the sensor has been configured. If the sensor has not been configured, the sensor becomes an AP access point and serves a simple website at 192.168.4.1. The client connects to the sensor using their device and opens the website. The website prompts them to add information essential for the operation of the sensor subsystem, which in this case is the MAC address of the Main Hub. When the MAC address has been registered the value is persisted into the device’s flash memory, making it available after power down. Finally the ESP32 reboots.

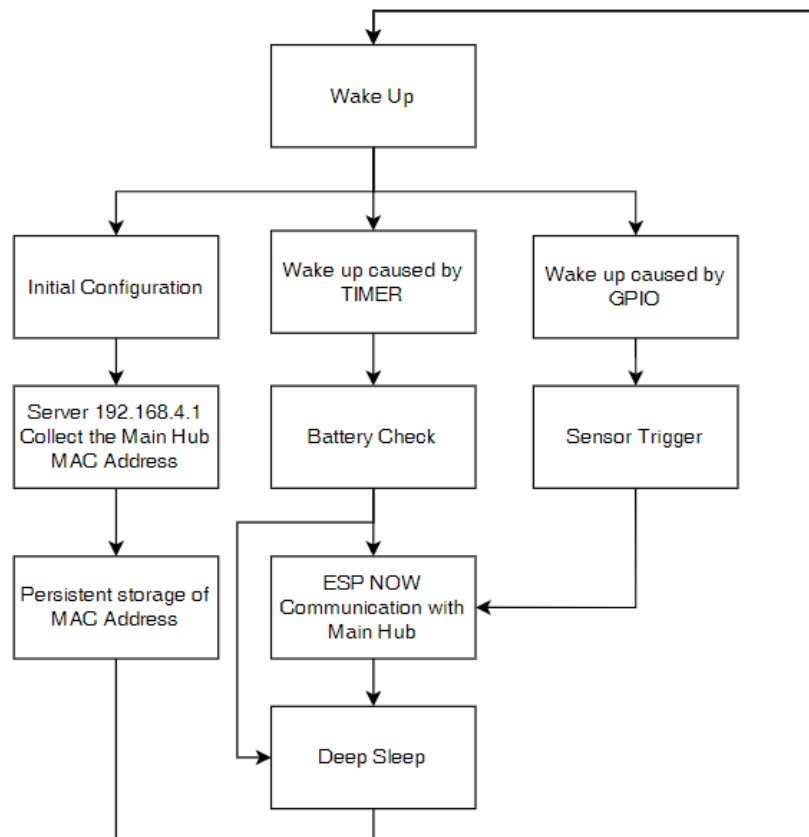


Figure 9. Code flow for the node subsystem.

If the sensor has already been configured, it checks the wake up source. The ESP32 registers the wake up source after exiting deep sleep mode and depending on the reason for exiting deep sleep it executes one of two flows. First, if the wake up was caused by the assigned GPIO pin going high/low, then that corresponds to the sensor getting triggered. The corresponding trigger message is sent through ESP NOW to the Main Hub and the device goes back to deep sleep mode.

If the wake up was caused by the timer, then it is a battery check. The ESP32 is programmed to wake up occasionally to do a battery check and ensure that the device is working properly. The battery check is done using the ADC1 in the ESP32. It executes an analog measurement of the battery voltage and depending on the result the charge level of the battery is determined. Since the range of voltages for the ESP32 goes from 0 V to 3.3 V and the expected voltages from the battery go from 3 V to 4.2 V, a resistor divider is placed to measure half of the battery voltage, which has the added benefit of operating in the linear region of the ESP32 ADC. In this case, if the voltage is lower than 1.55 V (3.1 V for the battery voltage) then the battery is low and a message is sent to the main hub to report it to the user. If the low battery message was previously sent and the voltage is above the 1.55 V threshold then the battery level is not low anymore and the sensor messages the Main Hub to report the battery level is not low anymore to the user.

With this in consideration, the current in transmission mode for the ESP32 is around 120 mA while it is around 2 mA in deep sleep. The next factor in reducing power consumption is the time spent transmitting the events to the main hub. This factor is essential in two decisions regarding this subsystem: the communication protocol and the use of two ESP32s in the Main Hub. For the communication protocol, ESP NOW allows for communication between two ESP32s when the MAC address of the receiver is known. The communication protocol is kind-of WiFi transmissions without most of the headers and overhead that comes with more robust protocols. For low-count

systems like home security, ESP NOW provides a good framework for fast communication. Using ESP NOW marks the transmission time of the ESP32 from wakeup to sleep in around 60 milliseconds. The use of two ESP32s in the Main Hub is explained in the following section.

Testing for this system was performed in the development board. The first test is regarding the functionality of the sensors. Each sensor was tested under the conditions it is intended to detect, like liberating butane from a lighter and checking whether the sensor was triggered. The second test is regarding the deep sleep mechanism of the ESP32. The test involved checking whether the ESP32 was adequately waking up under the different conditions, both timer triggered and GPIO triggered. The third test involved the ESP NOW communications between two boards and checking whether the communication was successful. The fourth test involved checking the initial configuration of the ESP32, whether the access point was established, the page served and the data collected and persisted. The final test is an integration test across all the elements explained before.

5.4 Main Hub Subsystem

The Main Hub subsystem needs to meet the following requirement: it should receive the communications from the different sensors in the home security system via ESP NOW and communicate with the backend to update the sensor information in the home security database.

Figure 10 shows a high level overview of the Main Hub schematic. Power is provided from a power cord connected to the wall, providing 5 V with a maximum current of 4 A. Since the Main Hub also routes power to charge the sensor batteries, it requires the higher current rating to deliver 3A of charging current to the sensor boards. Power from the cord is regulated down to 3.3 V, which are then the voltages supplied to the two ESP32s that form this board. The design decision on the use of two ESP32s will be explained later in the section. The two ESP32s communicate with each other via I2C, and each one with the respective subsystem using WiFi and ESP NOW.

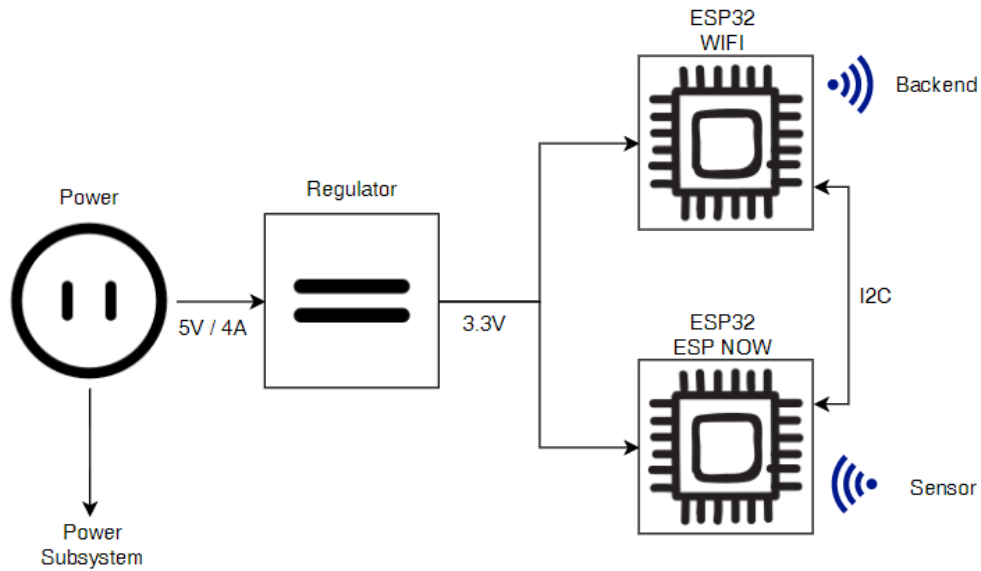


Figure 10. Schematic for Main Hub.

The first design of the Main Hub contemplated using only one ESP32, which would do both the WiFi communication and the ESP NOW communication. The difficulty arises since most modern routers use channel switching to continuously operate in the least crowded channel. Since channel switching means the WiFi channel cannot be static in the ESP32, then the sensor board before executing the ESP NOW communication needs to find out what channel the Main Hub is operating. This adds overhead to the system. In particular, tests suggested that around 95% of the time consumed in the transmission on the sensor side was spent trying to match the Main Hub WiFi channel. With two ESP32s, one can handle the dynamic changing channel of the WiFi and the other the simpler and faster ESP NOW communication. To link both ESP32s, an I2C channel between both is set-up.

With the design of the board better explained, the next part is to go over the code flow for each of the ESP32s. The first one to be covered is the ESP NOW unit. Figure 11 shows the code flow for the ESP NOW unit. The code consists of two flows, one that covers the process from boot to the

execution of the loop routine and an interrupt flow caused by the reception of a ESP NOW communication.

The device configures the I2C communication between the ESP NOW unit and the WiFi unit. The ESP NOW is the master device and the WiFi device is the slave. In the main loop routine, a queue is routinely checked for new alerts from the ESP NOW communication. If there is a new event in the queue, it identifies the board ID (each sensor board contains a unique ID) and the message (which classifies the communications in sensor trigger, high battery and low battery). It then sends the message through I2C communications to the WiFi unit. The second part of the code is related to the ESP NOW communications, and it is structured like an interrupt. Upon receiving the ESP NOW transmission, the message is read and added to the event queue described above. This queue is the way to connect the interrupt-driven part of the code with the sequential part of the code.

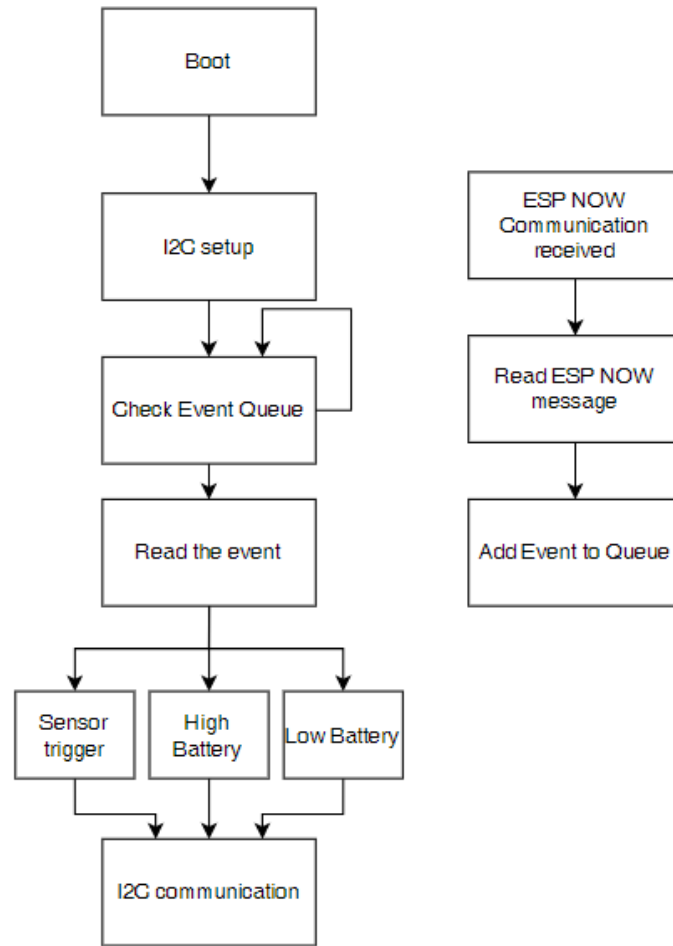


Figure 11. Code flow for Main Hub ESP NOW unit

The second unit is responsible for all WiFi communications with the backend of the system. It consists of another ESP32 with the following code flow. Figure 12 shows the code flow for the WiFi unit. Similarly to the ESP NOW unit, the WiFi code flow is divided into an interrupt-driven part for the I2C communications and a sequential part for WiFi communications. Both parts are again connected using a queue.

When the ESP32 starts up, the initial configuration of the device is checked. In a similar way to the Sensor board, if the device has not been configured then the ESP32 becomes an access point and serves a server at 192.168.4.1 to collect the user information, such as user ID, SSID and SSID

password. This step connects the Main Hub with the local network and the Internet. Once the data is persisted, the device is rebooted.

If the device is already configured, the I2C is set up and the event queue is checked sequentially. If there is an event in the queue, the message is read. The contents of the message determines what API route is used to update the home security database. The device calls the appropriate API route with the board ID to update the sensor information in the database. The interrupt-driven section is in charge of managing the reception of I2C communications from the ESP NOW unit. When the communication is received, it encodes the board ID and the message into the queue for the main flow to handle.

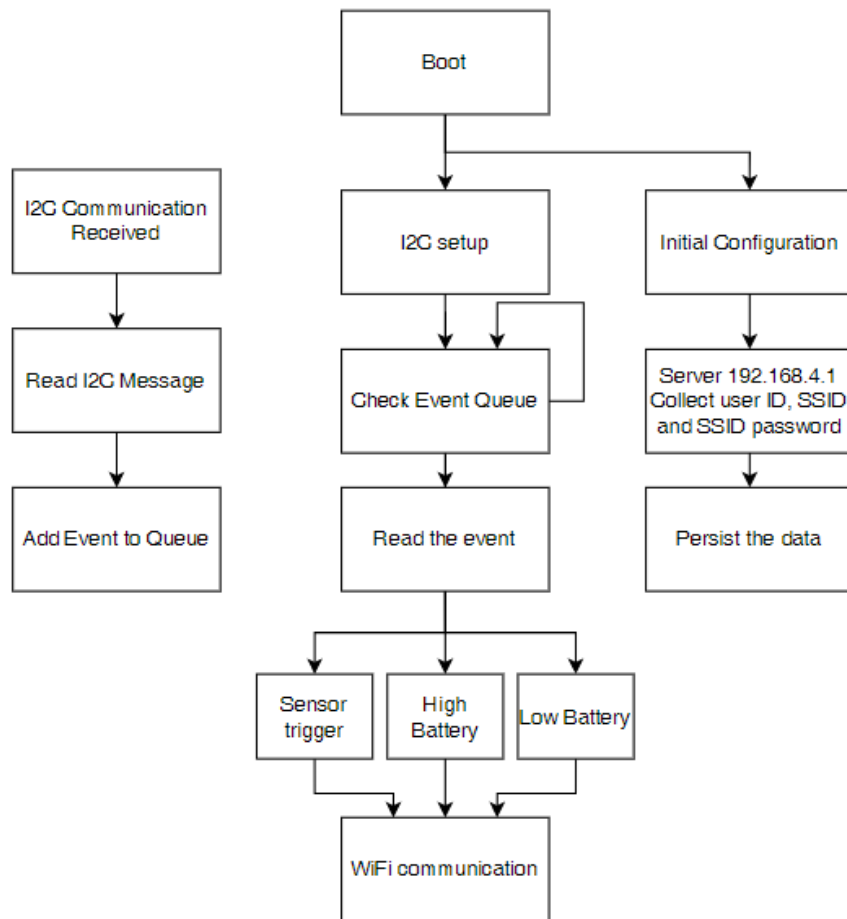


Figure 12. Code flow for Main Hub WiFi unit.

5.5 Power Subsystem

The Power subsystem is in charge of providing to the sensors the capability of recharging the battery. The current battery for the sensor boards is a 3.7V 6000mAh battery. This means that 1C for this battery is 6 A (for fast charging) and 0.5 C is 3 A (for standard charging). The current version of the power subsystem works for standard charging in order to limit the maximum current rating on the AC adapter. Figure 13 shows the high level schematic for the Power subsystem. A more extended view on the Power subsystem is shown in the full Main Hub and sensor schematics.

The functionality of the subsystem is simple. The power cord from the Main Hub provides a maximum of 4 A of current at 5 V. This is connected to a battery charging chip that supplies the 3A explained before to the sensor via a double-ended micro USB cable. When the user gets a notification that the battery level is low, they plug the sensor to the Main Hub in the provided port and it charges the sensor. The charging chip detects the low voltage from the battery and charges it completely.

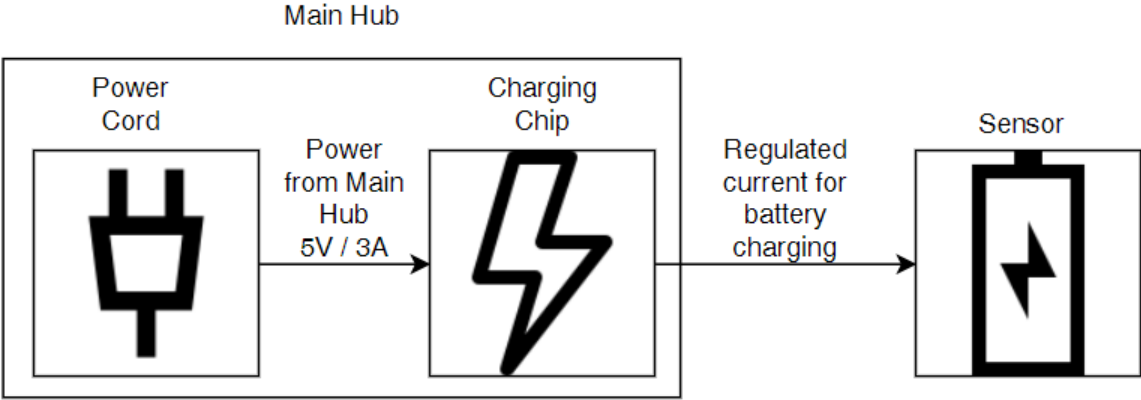


Figure 13. High level design for Power subsystem.

This subsystem, although basic, fills our requirement of providing users with the ability to have battery powered sensors in an ergonomic way, since they can charge the sensors again without having to buy new batteries, just by plugging it to the Main Hub.

5.6 Backend Subsystem

The Backend subsystem is the software architecture that allows the home security system to aggregate the user data and sensor status and make them widely available for users anywhere in the world with a connection to the Internet. Figure 14 shows the basic architecture of the Backend. The backend connects with two other subsystems, the Main Hub and the User Interface. Both these

subsystems communicate with the backend via HTTP requests (GET, POST, PUT) to the defined API routes.

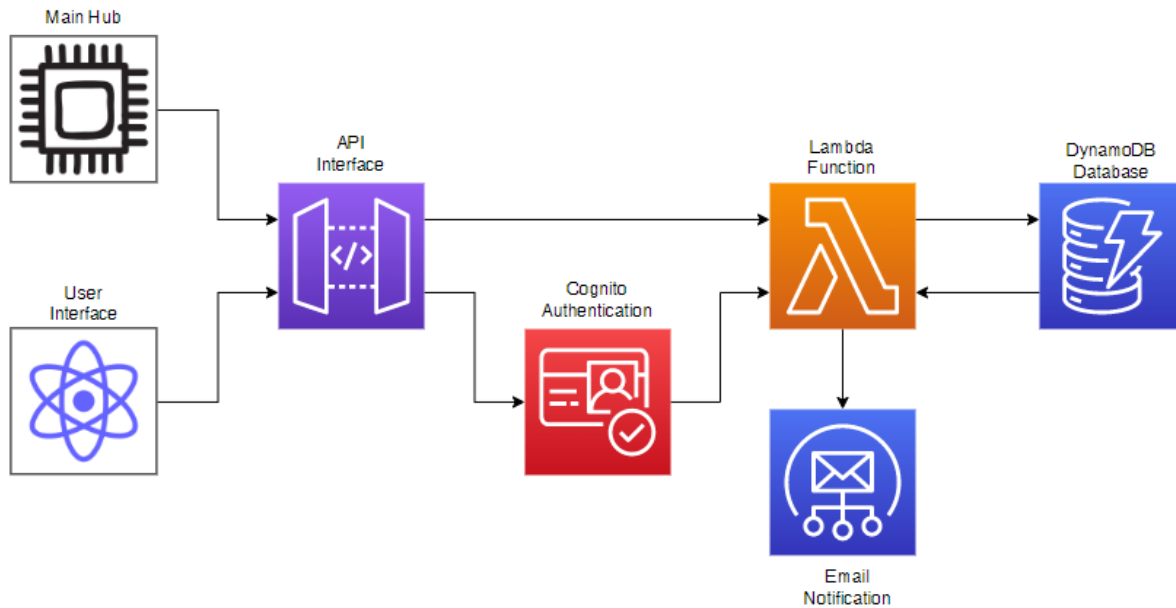


Figure 14. Backend Subsystem architecture.

The entire backend architecture is built using Amazon Web Services tools that allow for seamless integration. The HTTP requests from the Main Hub and the User Interface are received by the API Gateway, which defines the different available routes and the level of authentication for each one. A more detailed description of the routes and the security of each one will be provided later. After the API call is executed and categorized by the API Gateway, if the call does not require authentication, then it gets processed directly by the Lambda function associated with the call. If the call requires authentication, AWS provides an authentication service called Cognito that is connected to the users' database.

The Lambda function is the computing element that executes the logic of the API calls. The structure of the Lambda function follows a popular JavaScript framework called Express to define the different routes and the responses to the requests. The Lambda function is also responsible for

the communication alerts to the user that happen via email. The reason JavaScript was selected is because most of the User Interface was already going to be programmed with JavaScript, so keeping JavaScript for all aspects of the Backend and User Interface provided consistency across the two subsystems.

The Lambda function finally connects the information received from the HTTP requests to the API routes to the database. The database contains user information (like name, email and phone) and sensor information (like sensor name, sensor type, sensor status). The database selected for this project is DynamoDB, which is a type of NoSQL database. NoSQL databases provide the benefit of being able to dynamically change their structure if the need arises in the project to change the database structure without requiring to change all previously-existing elements. The storage mechanism also closely follows the object structure in JavaScript, providing familiarity to the team members in charge of designing the backend. The database structure will be covered in more detail later in the report. To summarize, the backend consists of API routes that could be authenticated, handled by Lambda functions that interact with a NoSQL database.

Generally, there are three API routes depending on the objective of the call. Figure 15 shows the three different routes. The user route requires authentication since it has access to the user's personal information and the ability to disarm and arm the sensors. This route is used by the User Interface to display the user information when they log in to the website and to arm/disarm the sensors. The hub route and battery routes do not require Cognito authentication since they are less vulnerable to a malicious attack. The hub route is in charge of triggering the sensors, but it cannot disarm or arm the sensor. This is the route used by the Main Hub to trigger the sensor. The battery route is the one in charge of reporting changes in the battery status, either low or high battery, and it is also called by the Main Hub.

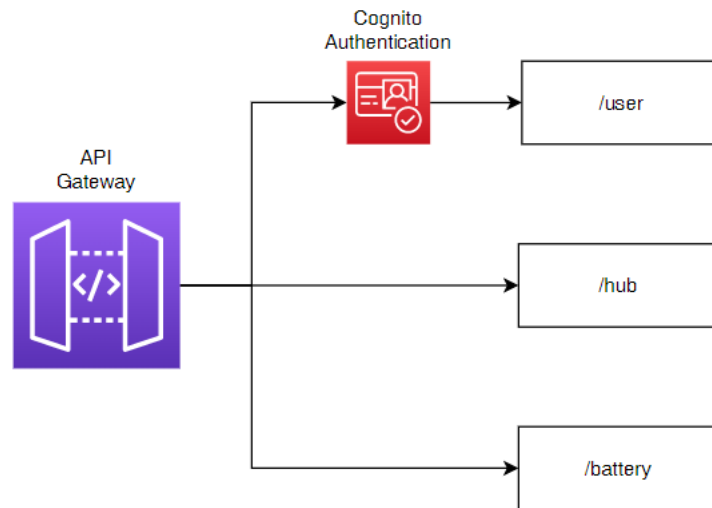


Figure 15. API Routes.

The database structure is the final description of the backend architecture. Each user has some personal information associated with their user ID. This information is their name, email and phone number. The personal information allows the system to display the information about the system in a more friendly way and includes points of contact to the user in case of emergency for email alerts. Each user also has a sensor object associated with it, which holds the information of all sensors. Each sensor has a name that aids the user in recognizing and differentiating the sensors, the status that indicates whether the sensor is armed, disarmed or triggered, the sensor type (magnet, movement or gas), the battery status (high/low) and the trigger date (the last time the sensor was triggered).

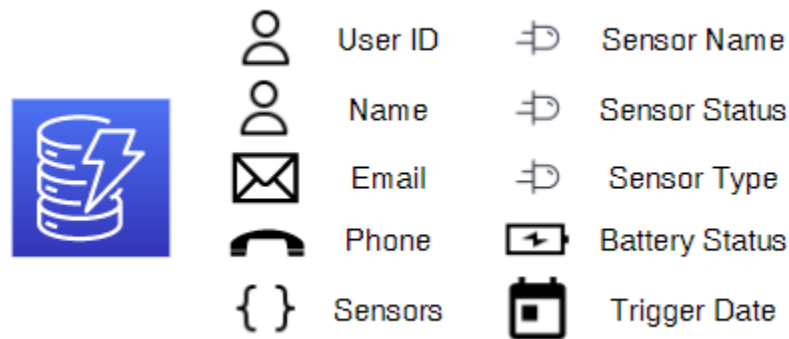
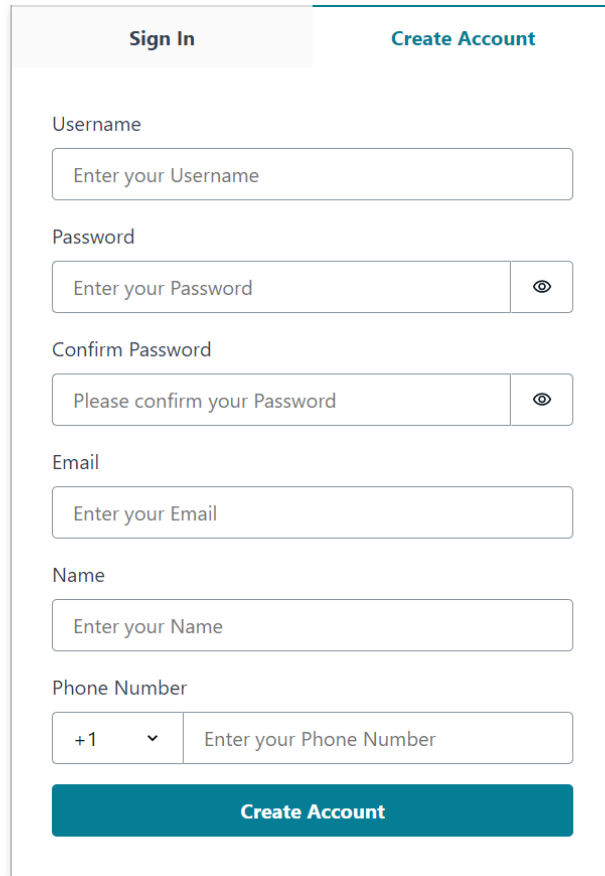


Figure 16. Database structure of the system.

5.7 *User Interface Subsystem*

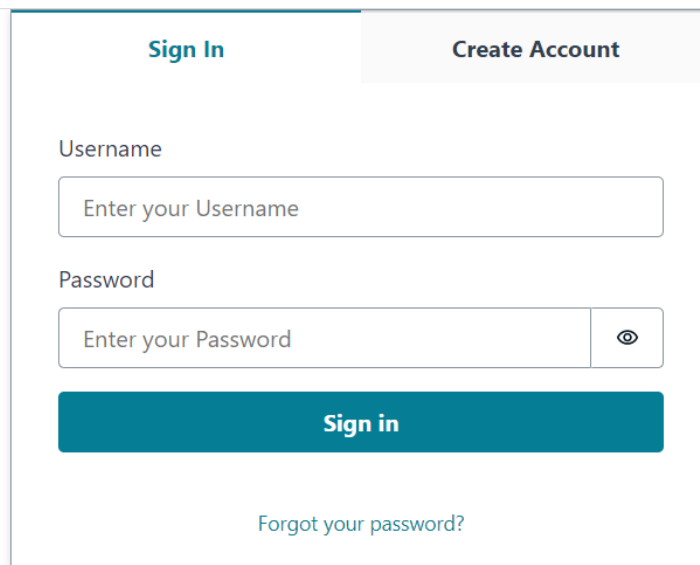
The User Interface subsystem allows the user to interact with the sensor data. The front-end framework for the User Interface is React (JavaScript). Some team members already had previous experience with this framework. React also allows for easy hierarchical design and easy updates to the User Interface data. To speed up the process of the component design for the User Interface, Cloudscape Design, a React component library was used. Cloudscape allows stylized components that help create a visually appealing User Interface without a lot of design experience.

The User Interface needs an authentication and authorization process to ensure that the data is safe and that malicious actors cannot disarm the home security system. To fulfill the authentication and authorization requirement for the User Interface, there is a Cognito Authenticator built on top of the User Interface. As described in the previous subsystem, Cognito is a user-based authentication system based on AWS, which allows the creation of user pools to add security layers to our applications. Figure 17 and 18 shows the User Interface authenticator for both sign in and sign up.



The image shows a 'Sign Up' form with two tabs: 'Sign In' and 'Create Account'. The 'Create Account' tab is active. The form contains the following fields: Username (text input), Password (password input with a visibility toggle), Confirm Password (password input with a visibility toggle), Email (text input), Name (text input), and Phone Number (text input with a country code dropdown set to '+1'). A teal 'Create Account' button is at the bottom.

Figure 17. User Interface Sign Up.



The image shows a 'Sign In' form with two tabs: 'Sign In' and 'Create Account'. The 'Sign In' tab is active. The form contains the following fields: Username (text input) and Password (password input with a visibility toggle). A teal 'Sign in' button is below the password field. At the bottom, there is a link that says 'Forgot your password?'.

Figure 18. User Interface Sign In.

The User Interface uses HTTP requests to get both the user information (such as the user's name) and the sensor status. The HTTP requests are executed periodically when the application is open every 5 seconds. Figure 19 shows the main view of the User Interface. Each sensor is displayed with their corresponding name, type and status. Users can “Acknowledge” triggered sensors, “Arm” disarmed sensors and “Disarm” armed sensors.

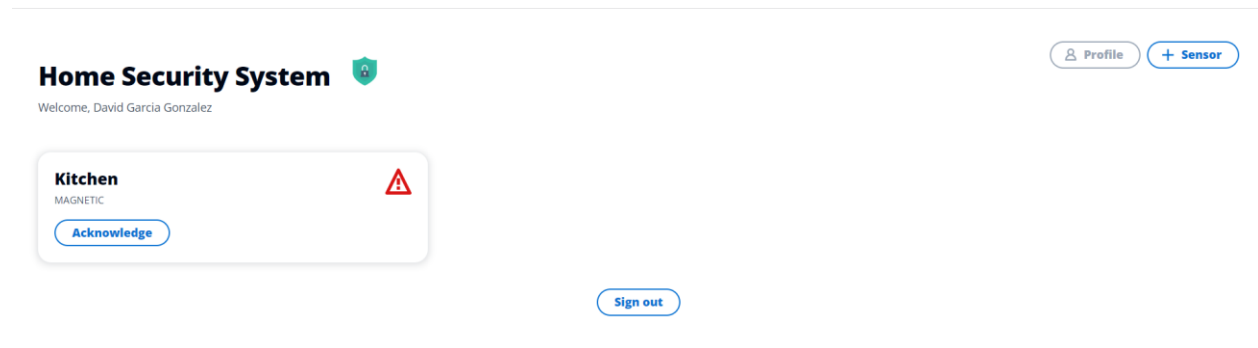


Figure 19. User Interface main view.

Users can also add new sensors to the system. Figure 20 shows the sensor registration form that allows users to add sensors. It lets the user add a personalized name to the sensor, the provided sensor ID and the sensor type.

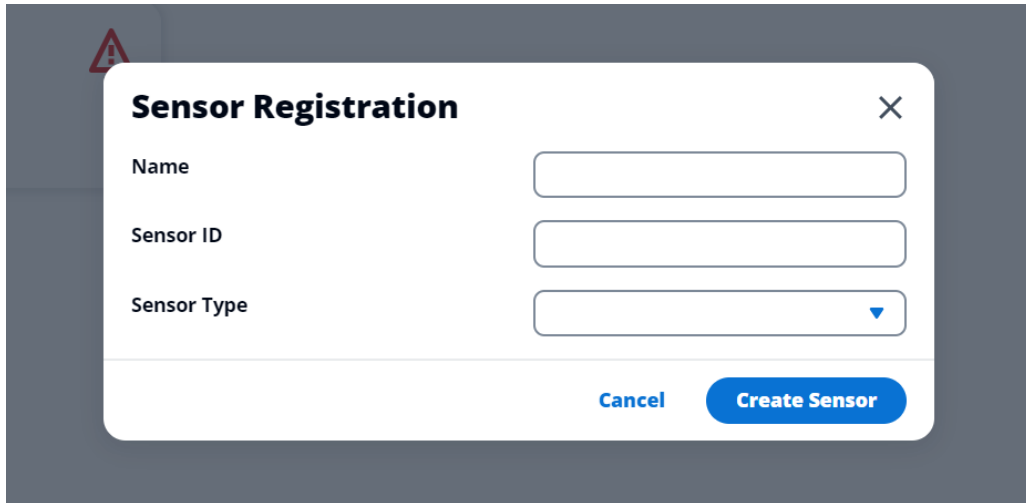
A screenshot of a 'Sensor Registration' form. The form is white with rounded corners and a close button (X) in the top right. It contains three input fields: 'Name', 'Sensor ID', and 'Sensor Type'. The 'Sensor Type' field is a dropdown menu. At the bottom right, there are two buttons: 'Cancel' and 'Create Sensor'.

Figure 20. Sensor registration form.

6 System Integration Testing

6.1 *Describe how the integrated set of subsystems was tested.*

Before manufacturing, the entire system was tested using two ESP32 development boards and an Arduino board. Since the system contains three ESP32s and there were only two development boards, the Arduino board was used to mock the behavior of the third ESP32.

Two tests were conducted. The first tested the connection between the sensor board and the ESP NOW unit of the Main Hub. The WiFi unit of the Main Hub was mocked with the Arduino board to simulate the reception of the I2C command that corresponds to the sent message from the sensor board. The flow for the test was as follows: the sensor board was triggered with one of the three sensors, using the serial port to confirm that the sensor board woke up from deep sleep and that the ESP NOW communication was successful. The Main Hub ESP NOW unit received the message and relayed it to the mocked WiFi unit (Arduino) via I2C. This was all checked using the serial ports of the ESP32s and the Arduino.

The second test had the ESP NOW Main Hub unit as the mocked element by the Arduino. The Arduino created a mock I2C transmission corresponding to the reception of an ESP NOW transmission from the sensor board. The WiFi Main Hub unit received the I2C communication and executed the HTTP request to the API. The User Interface was used to confirm the end to end reception of the trigger.

6.2 *Show how the testing demonstrates that the overall system meets the design requirements*

The first requirement described at the start of the report is providing users the ability to mix and match the home security system that fits their needs. This was accomplished by allowing for a complete user-friendly flow to connect a new sensor to the existing Main Hub board. This initial configuration fulfills the requirement of allowing users the ability to easily install the Main Hub and the sensors. The User Interface fulfills the requirement of providing users the ability to manage the home security system anywhere with an internet connection.

7 Users Manual/Installation manual

Note that directions to set up the main hub and sensor products are identical. However, the main hub should always be set up before sensor products.

1. Power on your products by opening the container and sliding the power switch.
2. In a WiFi-enabled device (phone, tablet, PC, etc.) , connect to Home Security. It should show up as a WiFi network you can connect to.
3. Using your internet browser, navigate to 192.168.4.1 .
4. Input your network credentials in the form you are redirected to (form shown in figure 21):

ESP Wi-Fi Manager

Not secure | 192.168.4.1

ESP Wi-Fi Manager

SSID

Password

IP Address

Gateway Address

Figure 21. Form to configure network credentials into products.

5. Close the confirmation page and wait for your new sensor to show up in your monitoring page. Working sensors should look like this in your main page:

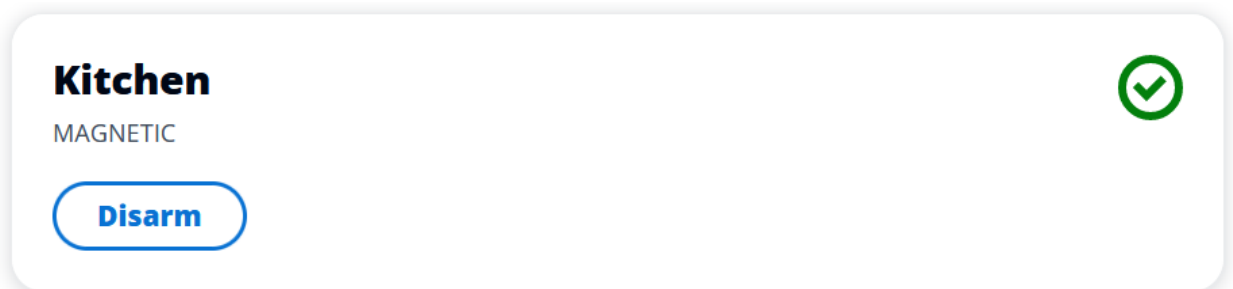


Figure 22. Example image of armed sensor

6. Close the container and visit your home security system main page and you should be ready to go!

8 To-Market Design Changes

There are some unexpected, minor issues we had to get around in order to make the design functional. The layout for the boost converter chip. Fortunately, this chip drives sensor modules, so a single-core wire between the correct node in the sensor board and the sensor module fixed that problem. Ideally, we would want the layout to be perfect since wires are more unreliable than traces. Additionally, the battery connector for the sensor had different dimensions to the ordered socket. Again, we got around this issue by making our own through hole connectors. However, these could be more unreliable, and making this from scratch could be wasteful in an electronic supply chain.

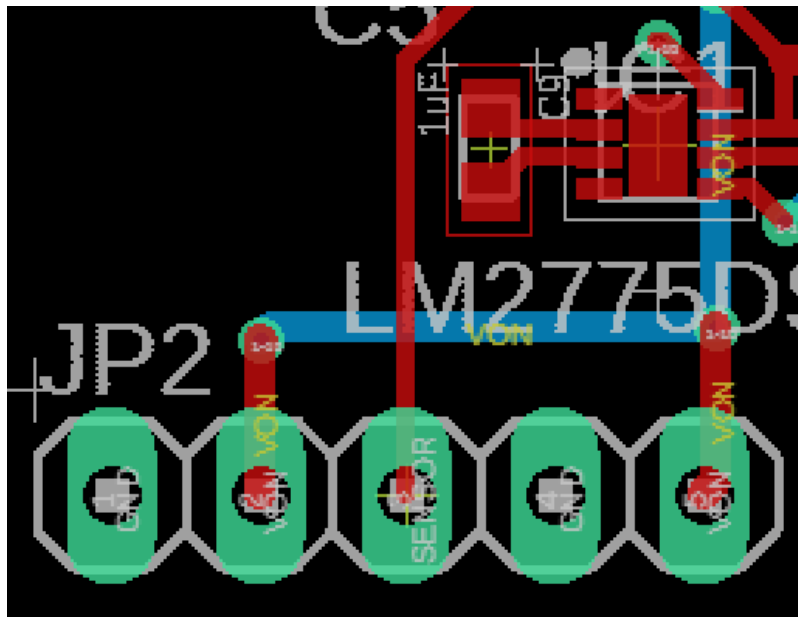


Figure 22. Picture of sensor header pins being wired to V on instead of V boost.

Some design improvement ideas came to mind once the design was assembled and functional. For instance, the batteries were cumbersome in assembly because of their weight and size. We acknowledge that there is a tradeoff between battery life and product size. It is important that sensor

nodes are as light and small as possible because their visibility affects their usability. Visible sensors can be disconnected by intruders, so a smaller, lighter battery is worth exploring in further revisions. Additionally, weight of the battery shortens the lifetime of the mount used in the mechanical setup of the sensors. The other design improvement is automating the programming procedure. Ideally, all the firmware for this product should be uploaded with minimal effort by a technician. This task would entail creating a graphical user interface that a technician can use to upload the firmware and flash the memory. Flashing the memory is necessary in order to store the network credentials in non-volatile memory. Finally, it would be worth exploring the idea of designing a board without a boost converter because the magnetic sensor does not need one. This is worth doing if more types of sensors that operate with 3.3 logic are added to the setup. The reasoning behind designing a general-purpose sensor board that supports all 3 sensors was to avoid creating another board and make mass-scale assembly more swift. Additionally, this makes the project scalable since more and more sensors can be supported without any board redesign. Choosing a board is one less step in a supply chain and it is plausible that the reduction in operational cost is greater than that of buying extra components.

9 Conclusions

Overall the system provides a robust security system with additional features that make it appealing to the end user. With every user living with a unique set of circumstances, they bring with them a unique set of needs for their security which this system is able to provide for. Through its modular nature, sensors are able to be removed and added to the system with relative ease. The user interface not only makes the installation process easier, but also more accessible to those who might not be as technologically literate as their peers. Furthermore, the notification system allows for customers to check on the status of their home even while they are away giving an increased feeling of security. Finally, the power supply has been optimized to improve the battery life of the system reducing the annoyance that comes with the need to recharge batteries.

10 Appendices

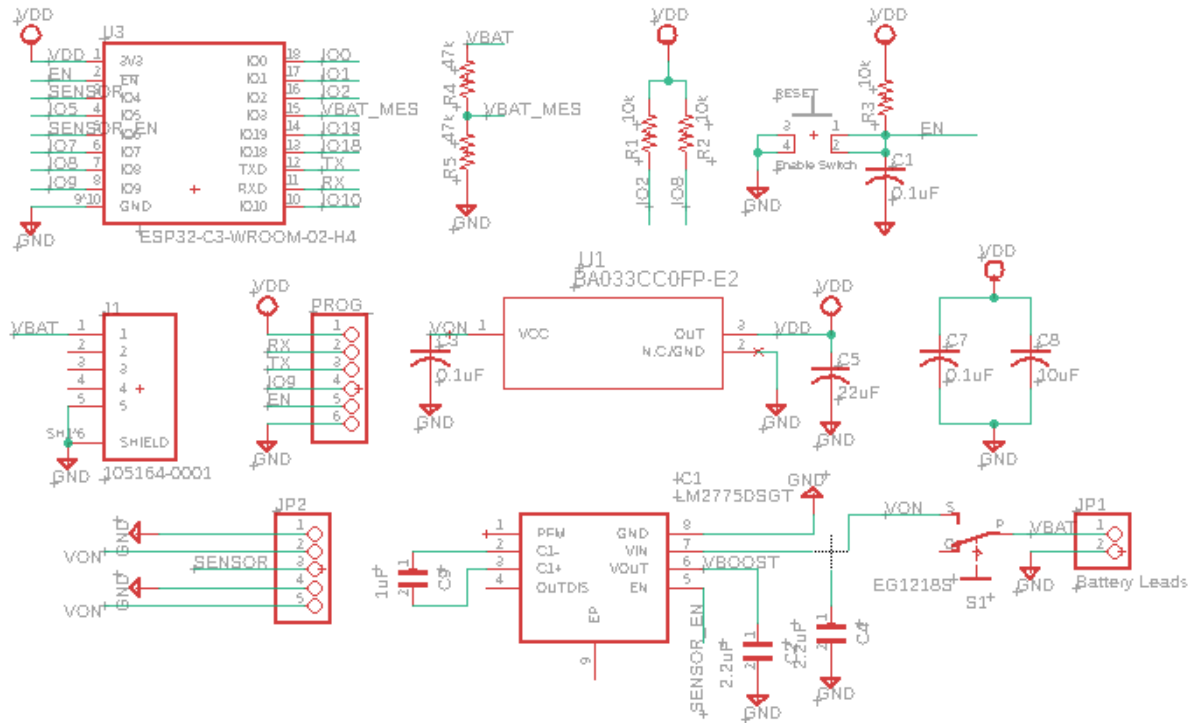


Figure 23. Schematic for general-purpose sensor board.

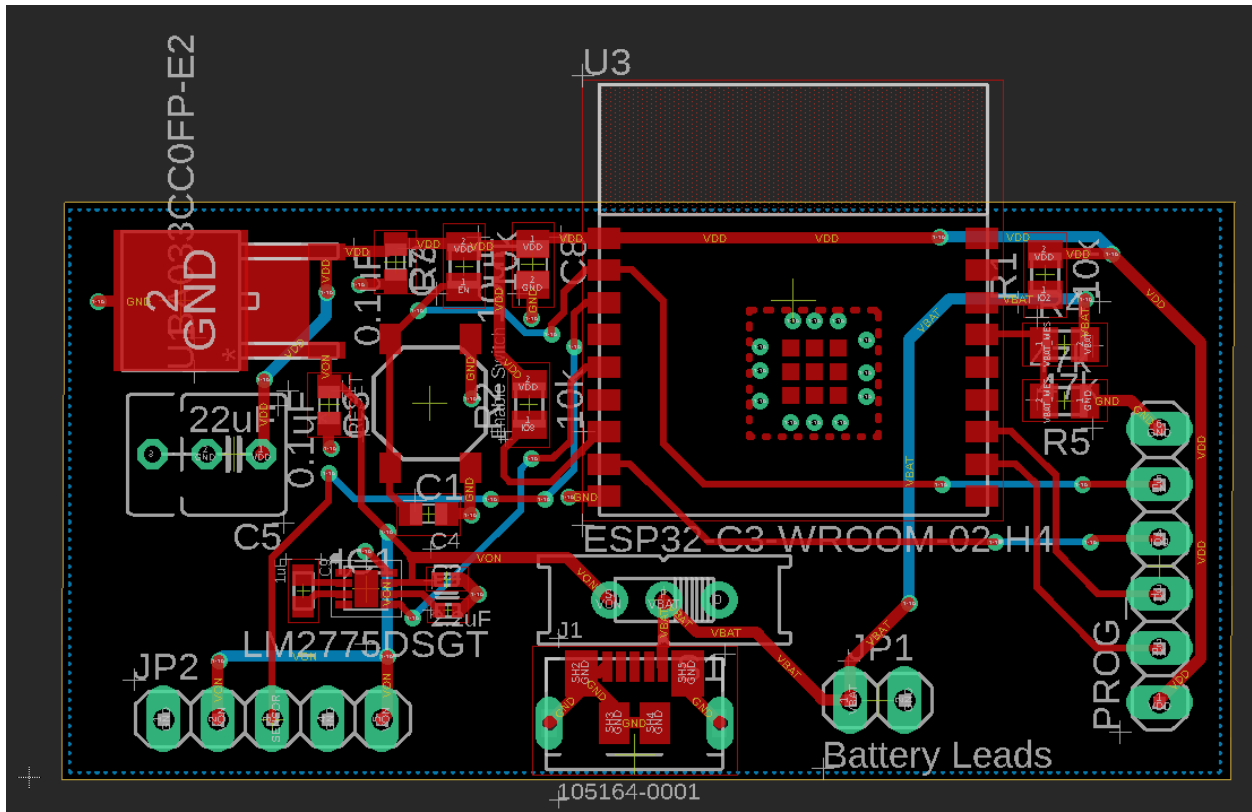


Figure 24. Board layout for general-purpose sensor board.

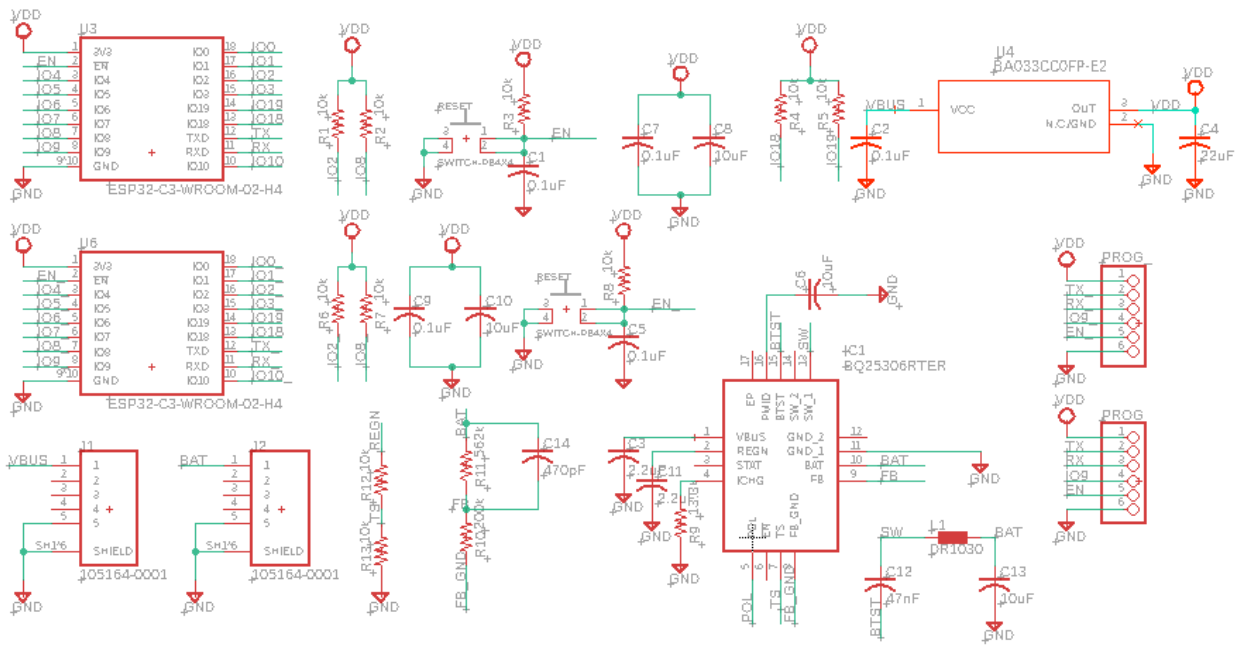


Figure 25. Schematic for main hub board.

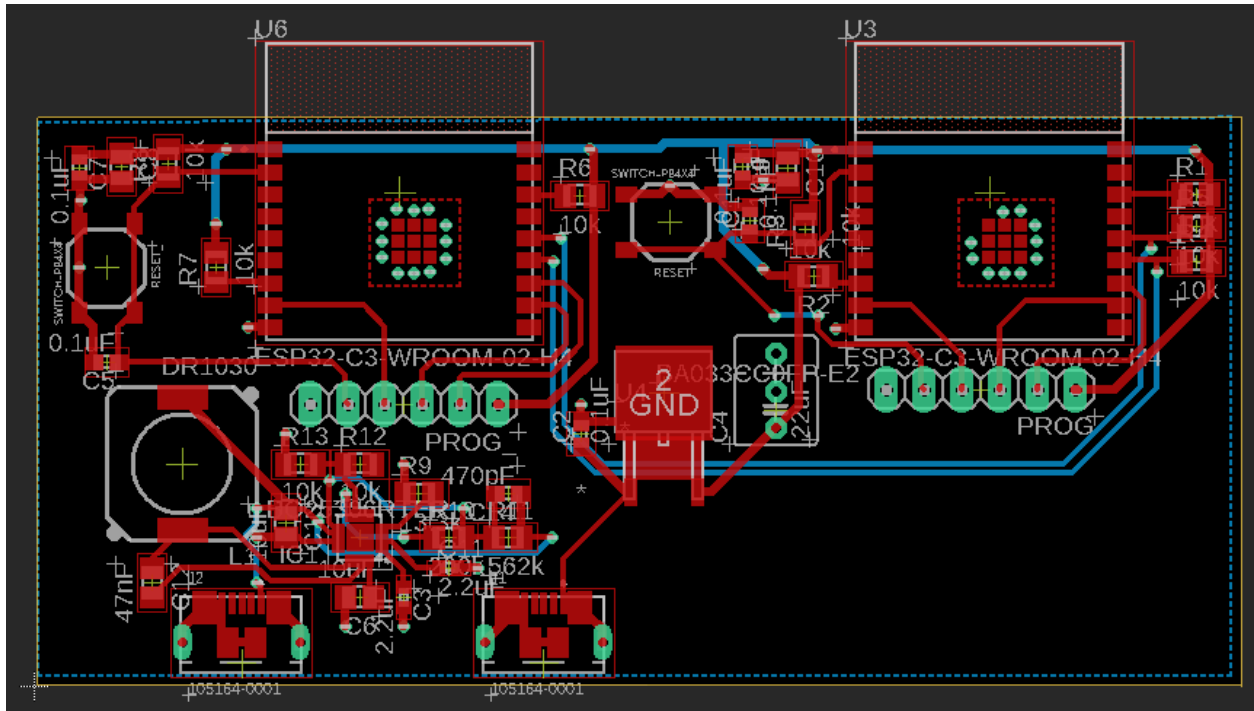


Figure 26. Board layout for main hub board.