# H.A.L.O.
# Hall-effect Array for Live Observation

Magnetic Field Imaging

EEL4914 - Senior Design 2
Group 18

**College of Engineering and Computer Science**

UNIVERSITY OF CENTRAL FLORIDA


***Authors***

| | |
|---|---|
| Ryan McCord | *Electrical Engineering* |
| Gabriel Martin | *Computer Engineering* |
| Aidan Kennedy | *Electrical Engineering* |
| Brogan Dougherty | Electrical Engineering |

***Review Committee***

| | |
|---|---|
| Dr. Enxia Zhang | *Assistant Professor* |
| Dr. Jaesung Lee | *Assistant Professor* |
| Dr. Piotr Kulik | *Assistant Professor* |

***Advisor***

Dr. Chung Yong Chan

# Table of Contents

# List of Tables



# List of Figures

# 1: Executive Summary

H.A.L.O. (Hall-effect Array for Live Observation) is a portable magnetic field imaging system developed to capture and visualize magnetic field distributions in real-time using an array of sensors. Designed as a standalone, user-friendly platform, H.A.L.O. enables intuitive analysis of magnetic field intensity and distribution in the form of a heat map. The system can be utilized in educational and research settings to better understand magnetic field behavior in everyday objects or new technologies.

The sensor array consists of 64 Texas Instruments DR5053VA sensors arranged in an 8x8 matrix. Four CD74HC4067 analog multiplexers (MUXs) are used to sequentially scan sensor outputs, which are then digitized by an ADS7883SDBVR 12-bit ADC. An ESP32-WROOM microcontroller chip handles the multiplexing and data acquisition process. All captured data is then transmitted over UART to a Raspberry Pi 4, which renders the real-time heat map using a Python-based graphical user interface (GUI) displayed on a TFT LCD.

The system is powered by two single-cell lithium-ion batteries connected in series, providing a total voltage of 7.4V. Power is managed by the MP2615 battery management IC, which also enables the charging of the system through a 12V DC jack. Two switching buck regulators are used to supply the system components. The LM2576 was chosen to step down to 3.3V for the ESP32 and sensor array PCBs, while the LM2679 provides 5V for the Raspberry Pi. The hardware for this system is modular, meaning that it consists of five custom PCBs: Sensor Array, ESP32 Microcontroller Module, Battery Management System, 3.3V Regulator, and 5V Regulator.

The system is enclosed in a durable, transportable housing, allowing it to be used in various environments without relying on any external power supply. H.A.L.O. has been designed with scalability in mind, as an expanded array of sensors would allow for higher spatial resolution in magnetic field mapping. By combining multiple 8x8 arrays, users could analyze more complex or larger magnetic environments. With our main focus on real-time processing portability, and ease of use, H.A.L.O. offers a cost-effective and accessible solution for visualizing magnetic field behavior and strength.

This senior design project was developed by Group 18, which consists of four team members: Gabriel Martin, majoring in Computer Engineering, and Ryan McCord, Aidan Kennedy, and Brogan Dougherty, all majoring in Electrical Engineering.

# 2: Project Description

## 2.1 Background and Motivation

Magnetic fields are present in many everyday items and electrical systems, yet measuring and visualizing them remains a challenge. Traditional tools for magnetic field measurement, such as handheld Gaussmeters or single point sensors, often lack spatial resolution and dynamic feedback. These methods are typically limited to static measurements and do not provide an intuitive and dynamic view of magnetic behavior.

Hall-effect sensors provide a cost-effective solution for measuring magnetic fields, as they generate an analog voltage output proportional to the magnetic field polarity and strength. While individual Hall-effect sensors are commonly used for position sensing or proximity detection, they cannot individually provide a visualization of magnetic fields. To visualize complex magnetic behavior, especially those involving time-varying fields, there still remains a need for a system that can perform high-resolution magnetic field mapping in real-time.

This project addresses that need through the implementation of an 8x8 array of Hall-effect sensors. When a magnetic field is present, the polarity and strength present at each of the 64 sensors will be displayed in the form of a color-coded heat map. By sampling and displaying data in real time, the system enables users to observe how magnetic fields look and change over time and space. This has direct relevance for applications such as identifying electromagnetic interference in PCBs or providing students with an intuitive understanding of a typically complex subject.

The motivation for this project stems from the lack of readily available tools capable of magnetic field visualization. Gaussmeters can only read magnetic field intensity at a specific point, while more effective systems such as MRIs are incredibly expensive and are not for public use. Our system aims to bridge this gap by combining precise sensing, portability, and real-time display. It is designed to be accessible and user friendly, making it a useful and essential resource for students and engineers looking to directly interact with magnetic field dynamics.

## 2.2 Overview of Project Goals

The primary goal of this project is to develop a real-time magnetic field visualization system that can detect and display magnetic field strength using an

array of Hall effect sensors. By providing a clear visual representation, we hope to make magnetic field analysis more accessible and user-friendly. The visual representation is comprised of a heat map of the magnetic field where different gradients correspond to different field strengths.

We created a compact, lightweight, and reliable device that can be used by engineers, researchers, and educators working with electronics. The device is easy to use in a variety of environments, primarily in classrooms and laboratories, while maintaining high accuracy and responsiveness.

*Goals:*

- Create a system capable of visualizing magnetic fields.

*Objectives:*

- Use an 8x8 grid of Hall-effect sensors to capture magnetic field measurements across a region of space.
- Use MUXs and an analog to digital converter (ADC) to ready the data for processing.
- Process the data with an ESP 32.
- Transmit the processed data to a Raspberry Pi Model 4 B that controls the display.
- Display the data on an LCD screen with capacitive touch that will allow the user to interact with the guided user interface (GUI).
- Visualize the magnetic field via two modes.
    - a heat map mode that shows larger magnitudes as lighter colors, lower magnitudes as darker colors, and colors readings associated with the north pole as red and the south pole as blue
    - a raw data that displays the units of magnetic field strength. The user can choose between mT or G as the displayed unit.

## 2.3 Project Features and Functionalities

*Overall functionality*

The overall functionality of the project utilizes Hall effect sensors to map magnetic fields and display them as a 2-dimensional heat map for intuitive readability. The device updates the screen at a rate of 10 Hz to show any changes in the magnetic field in real time.

*Sensor Matrix*

The sensor matrix is made up of 64 Hall effect sensors in an 8 by 8 configuration. We utilized analog Hall effect sensors instead of digital ones because it makes visualizing the heat map more comprehensive and allows for

different field strengths to be represented instead of just having the sensors detect the presence or polarity of said field. When analog Hall effect sensors come into contact with a magnetic field the electrons and holes separate in them creating a voltage that is proportional to the strength of the magnetic field. The analog outputs of the sensors feed into several multiplexers which then feed into an analog-to-digital converter to send the data to the MCU.

*MCU*

The MCU interfaces with an analog-to-digital converter which allows us to take the analog data coming from the multiplexers and convert it to digital signals that can be processed by the MCU. The MCU is reading the voltage value of the Hall effect sensors and assigning a field value to them. This information is then be transmitted from the ADC, to the MCU, and to our display screen

*Display*

The display receives data from the MCU and shows the 2D map of the field. It is a colored display using red to represent a region of north polarity and blue to represent a region of south polarity, brighter colors corresponding to higher field values. The display has 64 such regions that can change colors according to the Hall effect sensor from which it is receiving information. The display has a refresh rate of 10 Hz which allows for real-time updates from the sensors to show any changes in the magnetic field such as moving a magnet across its surface or rotating its polarity.

The display is able to switch between showing the heat map, or displaying raw values. It can also do both of these at the same time. The raw values can be displayed in mT or in gauss

The display is a touchscreen to allow for user inputs. The user will be able to use the screen to change between displays, choosing which graph they want to see and how they want the data to be presented. The user is able to change if they are viewing a graph showing amplitude or the raw data values. The screen also allows the user to save the data by pressing record while a flash drive is inserted into one of the USB slots. This data can then be viewed on a computer.

*Flash Drive Connection*

The device comes with a USB port on the side that can be used to mount a flash drive that allows users to save raw data values from the device to be viewed in an excel sheet on their computer. This excel sheet includes the data from each sensor and the time since the user pressed record in milliseconds.

*Software*

The software for the project is responsible for acquiring sensor data, processing it, and updating the visual heat map on the display in real time. The software is designed to meet a 10 Hz display refresh rate while sampling a 8 x 8 sensor array (64 sensors) with sufficient accuracy and speed. The microcontroller firmware is developed in C, utilizing the ESP32's existing libraries for communication protocols such as UART or SPI.

An ADC is used to convert the analog signals from the Hall effect sensors (routed through multiplexers) into digital values. Each sensor in the array is sampled individually. Given the need for real-time performance, it is critical that both the ADC conversion rate and the MCU's clock speed are sufficient to complete all 64 sensor readings within 1/10th of a second.

The system was designed to minimize noise so individual sensor calibration proved not to be necessary because of the accuracy of most of the sensors. We found that the sensors performed consistently regardless of their position on the board.

The data is sent from the MCU to a Raspberry Pi 4 which is responsible for driving the display. Communication between the Pi and the MCU is done via UART.

The mT values are directly mapped to a set of bins corresponding to intensity levels. Each bin is associated with a specific color, with 16-24 possible colors available to represent different ranges of magnetic field strength. The min and max sensor values are about -11.5 and 11.5, ensuring that the bin sizes accurately represent the range of measured voltages. At this stage, minimal processing beyond binning and scaling is performed on the sensor data, as the main objective is to maintain a high refresh rate for the heat map. The values acquired from the ADCs at this stage are all that is required for the comprehensive development of the various display modes outlined in the display section above.

The software's main loop is optimized to sequentially sample each sensor, process the ADC values, update the corresponding bins and values, and transmit the color-coded data to the display. The clock speed of the MCU is set at the default which is sufficient to ensure that all sensor readings, storage operations, and display updates are completed within the time frame required for a 10Hz refresh rate. This approach guarantees that the heat map remains accurate and responsive to changes in the magnetic field. The Pi uses the binned ADC data to generate a color-coded heat map. Each sensor's ADC

reading is mapped to one of the predetermined color bins. The display is partitioned into 64 regions, each representing a sensor's reading, which together form the complete 2D visualizations of the magnetic field.

## 2.4 Requirement Specifications

Table 2.5: List of Requirements & Stretch Goals

| Parameter | Purpose | Value |
|---|---|---|
| Number of Tiles in Display | Visualization resolution | ≥ 64 tiles |
| Field Size Detected | Magnetic field visualization | ≥ 64 $cm^2$ |
| Field Intensity Range Detected | Magnetic field visualization | [-8, 8] mT |
| Detection Distance | Accurate measurements | [1 cm, 2 cm] |
| Screen Refresh Rate | Expedient display of data | ≥ 10 Hz |
| Length of Recording | Value storage for analysis | ≥ 30 s |
| Power Consumption | Small energy footprint | < 5 W |
| Power Efficiency | Small energy footprint | > 75% |
| Accuracy of Detection | Accurate measurements | ± 1 mT |
| Operating Temperature | Convenience | < 140° F |
| Latency | Expedient data transmission | < 200 ms |
| Weight | Convenience | < 5 lbs |
| Size | Convenience | < 10 $cm^2$ |
| Cost | Affordability | < $250 |

## 2.5 House of Quality

The House of Quality provides insights into the trade-offs between various engineering requirements in H.A.L.O. Many of these requirements are interconnected, meaning improvements in one area could come at the expense of another. For example, increasing the refresh rate of the magnetic field mapping system may demand higher processing power, which could lead to increased power consumption and have an impact on our battery life. Similarly, reducing the system's size would require components to be placed closer together, increasing the risk of electromagnetic interference, which could degrade overall performance. Understanding these relationships allows us to make better-informed decisions in our design to ensure we create a well-balanced system that meets technical and logistical goals.

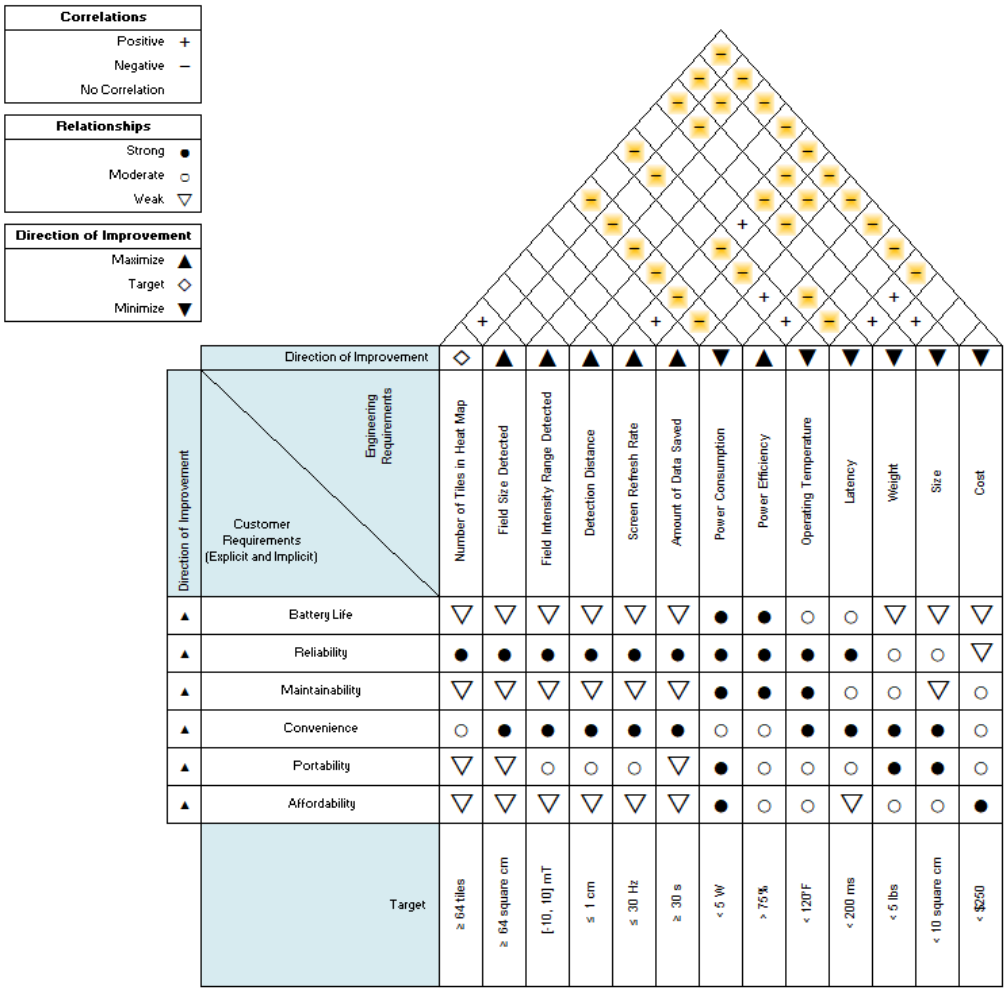



*Figure 2.6: House of Quality Diagram*

## 2.6 Block Diagram Representations

### 2.6.1 Hardware Block Diagram

The H.A.L.O. system consists of several custom PCBs, each dedicated to a specific subsystem. The Sensor Array PCB contains an 8x8 grid of DRV5053VA analog Hall-effect sensors used to detect magnetic field strength and polarity. These sensors are addressed sequentially using four CD74HC4067 analog multiplexers, and their outputs are digitized by a ADS7883SDBVR analog-to-digital converter. The digitized data is sent to the Microcontroller PCB, which houses the ESP32-WROOM microcontroller and a CP2102 USB-to-UART bridge for programming and debugging. The ESP32 handles all data acquisition and transmits the magnetic field data to the Raspberry Pi via UART.

The Battery Management PCB supplies power to the system using two single-cell lithium-ion batteries connected in series, delivering a nominal voltage of 7.4V. It incorporates the MP2615 battery management IC, which enables safe recharging via 12V DC input and provides over-voltage, over-current, and thermal protection to ensure reliable operation. To generate stable voltages for the system's components, the design includes two dedicated regulator boards. A 3.3V Regulator PCB was implemented using the LM2576-ADJ from Texas Instruments, followed by a 5V Regulator PCB based on the LM2679-ADJ. The 3.3V regulator powers the sensor array and ESP32, while the 5V regulator supplies power to the Raspberry Pi 4 and LCD. The Raspberry Pi runs a Python Graphical User Interface (GUI) that renders a real-time color-coded heat map and raw magnetic field data, depending on the view mode selected by the user. Data can also be saved to an external flash drive for further analysis. The system's modular design allows for significantly easier debugging and troubleshooting, reduced electromagnetic interference (EMI), and improved thermal regulation.

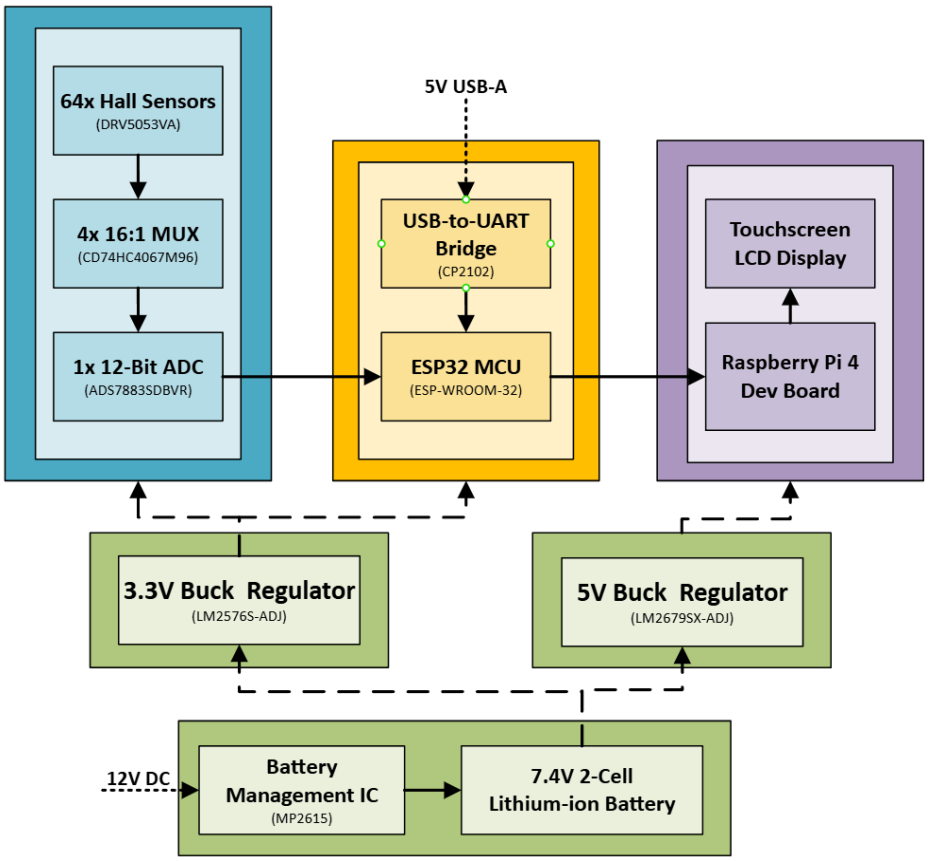



*Figure 2.6.1: Hardware Block Diagram*

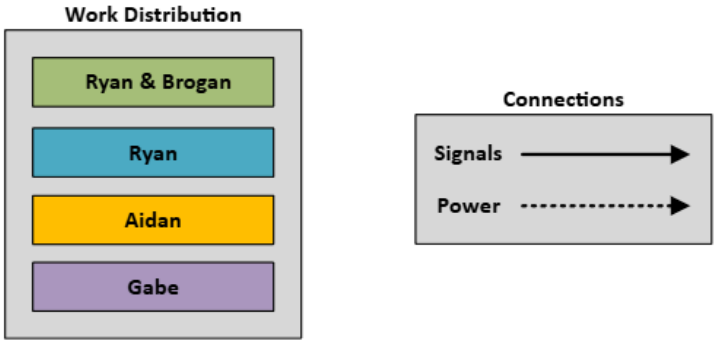



*Figure 2.6.2: Work Distribution & Connection Legends*

## 2.6.2 Software Block Diagram

The software for the H.A.L.O. system was built to handle data collection, processing, and visualization. A Raspberry Pi 4 Model B ran a Python-based

control program that received data from an ESP32 over a high-baud-rate UART link. The ESP32 handled all low-level sensor interfacing, including reading from the external ADCs and controlling the multiplexers on the sensor board. The Raspberry Pi focused on processing the incoming magnetic field measurements and updating the graphical interface on the touchscreen display. Data flowed through a real-time pipeline that ensured smooth communication between the ESP32, the signal-processing routines, and the Python-driven GUI. The software also supported data logging to a connected USB flash drive, allowing the collected data to be saved for later analysis.

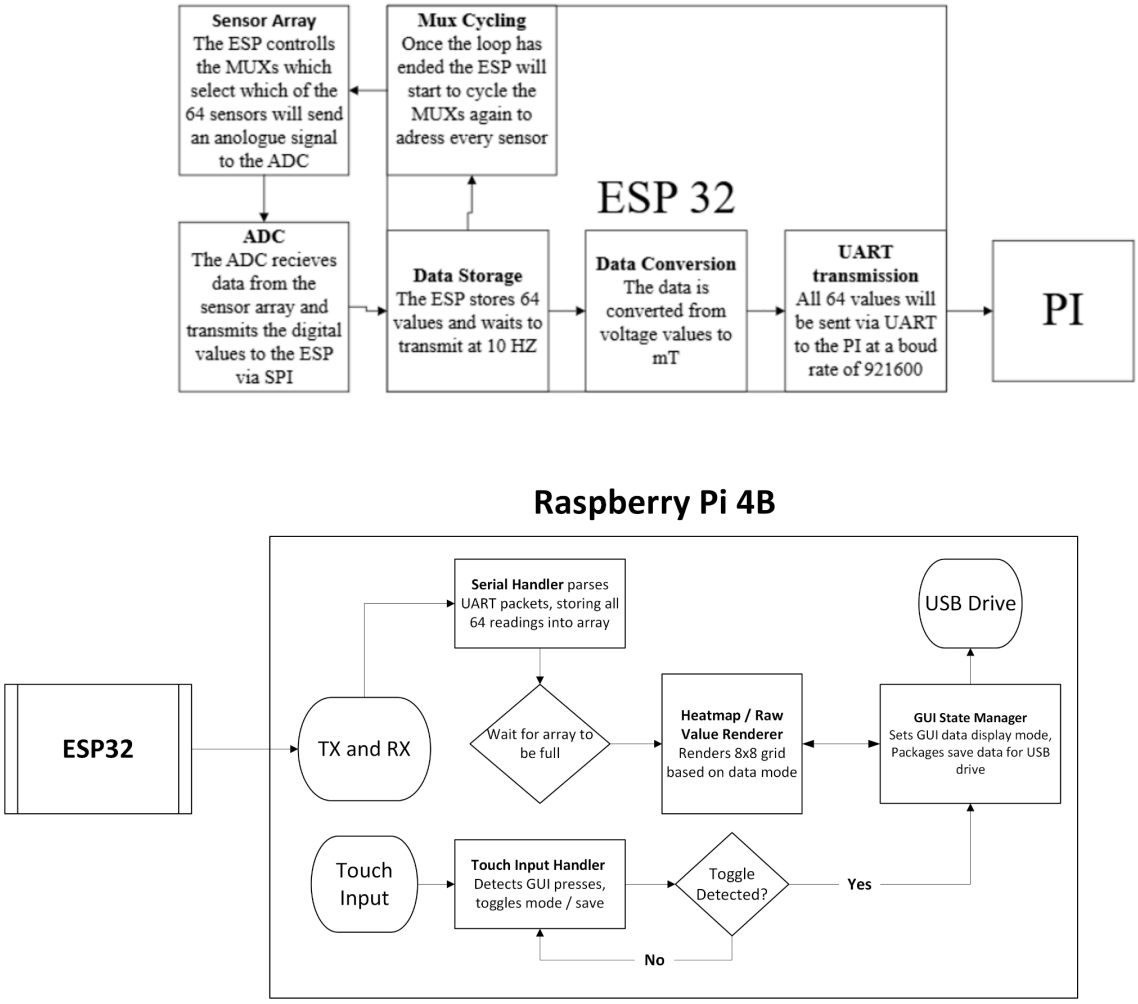



*Figure 2.6.3: Software Block Diagram*

# 3: Research & Investigation

## 3.1 Analysis of Similar Existing Products

### 3.1.1 Commercial & Research Systems

Magnetic field measurement technology was developed in the early 1800's and has seen many usages since then. Magnetic resonance imaging (MRI) is a mainstay in the medical field for its non-invasiveness and high-contrast images. [8] In electronics design and testing, magnetic field visualizers are used to image the fields around and generated by the test product. Magnetic field visualization can even be seen in its most simple form of iron filings for educational demonstrations. The point is, magnetic field visualization is a widely utilized and widely varied technology in the modern day.

### 3.1.2 Comparison and Justification

To better guide our project design, we analyzed two existing systems that align with our objectives and give insights into effective design strategies. These projects served as useful references, helping us understand other design approaches and areas for improvement.

The first project, "A Magnetic Imager Tile" [1], uses an 8x8 array of Hall-effect sensors to visualize magnetic fields. This system detects field polarity but does not measure the field's intensity. Data is transmitted using the I2C protocol to a laptop which displays the magnetic field. One of the key improvements we made was eliminating I2C from our system. Instead, we utilized a mixture of SPI, UART, and MIPI DSI. In our project, SPI was used for the transmission of data between the ADC and the MCU. SPI has higher bandwidth and lower latency, which proved to be crucial for providing our display with an adequate refresh rate. Also, this reference project relies on a development board and laptop to visualize the field. We were able to eliminate the need for external components by implementing a microcontroller chip and onboard LCD touchscreen to make the system function independently.

The second project, "A High-Resolution Magnetic Field Imaging System Based on the Unpackaged Hall Element Array" [2], has a different approach. It uses an array of 256 Hall-effect sensors attached to a motorized rail system to scan and render a high-resolution image of magnetic fields. The motorized rail system is great for high resolution magnetic field imaging, but it comes at the cost of real

time visualization and convenience. Our project differs in that we provide continuous real-time visualization rather than a static scan. Also, we incorporated data storage, allowing users to save up to 30s worth of values from every sensor into a spreadsheet that is saved to a flash drive.

By analyzing these existing projects, we were able to combine their strengths to create a system capable of high-speed magnetic field visualization.

### 3.1.3 Comparison and Justification

3.1.1 goes to show that there are numerous use cases for a magnetic field imager. The possible usages are so large that it's difficult to design a system for a specific purpose. In products like the HallinSight, Metrolab outlines the function and specifications of the device, but omits any text on how they recommend using it. [9] The purpose of this long preamble is to show that the applications for a field sensor are broad and trying to design a sensor for a niche purpose would take away from the applicability of the system. This applies to H.A.L.O. as well. The magnetic field visualizer can be used for a wide variety of purposes; however, H.A.L.O. shines best as an educational tool. Educational demonstrations of the magnetic field typically rely on simulations or iron filings. H.A.L.O. provides a new way of visualizing magnetic fields in an educational context. A student can see the magnetic field change around an object in real time as they manipulate the object across the sensors. This type of hands-on interactivity isn't available from demonstrations like a simulation, and the polarity and strength of the magnetic field being directly displayed on H.A.L.O. provides data that iron filings don't intuitively.

Another benefit of H.A.L.O. is the convenience of the system. The system is lightweight, small, easy to use, intuitive, and all of the components are completely contained in the system. All of these features don't exist together in a magnetic field visualizer currently on the market. This convenience and easily comprehensible visualization are among the reasons that H.A.L.O. shines so brightly as an educational aid. There is no complex start up process. Any user with a basic understanding of magnetic fields can turn on the system and use it for magnetic field visualization in a moment's notice. The portability of the system allows for easy transportation which only complements the previously discussed features.

## 3.2 Hall-Effect Sensors

Hall-Effect sensors are an essential part of H.A.L.O., as they are what allow for precise magnetic field measurements that are crucial for generating real-time maps of magnetic fields. These sensors operate on the Hall Effect principle, where the presence of a magnetic field will induce a voltage difference perpendicular to the current flow. Hall-Effect sensors are widely used in various fields and are available in two forms: digital and analog. Digital Hall-Effect sensors detect the presence of a magnetic field, while analog types provide a continuous output proportional to field strength. We are considering various factors for sensor selection within H.A.L.O., including sensitivity, resolution, noise characteristics, power consumption, and ease of integration.

This section examines the Hall-Effect sensors we considered for H.A.L.O., each having potential strengths in the context of our project's requirements. These options include the DRV5053VA, a linear analog sensor with a ratiometric voltage output; the DRV425RTJT, a high-precision fluxgate sensor ideal for high-resolution applications; the SS495A1, a robust sensor designed for industrial settings, and the A1324LLHLX-T, a low-power sensor with moderate sensitivity. In this section, we will analyze each sensor's specifications and determine which is the most suitable for H.A.L.O.'s performance needs.

### 3.2.1 Texas Instruments DRV5053VA

The DRV5053 VA from Texas Instruments is a chopper-stabilized Hall-Effect sensor that provides a linear analog output voltage in response to magnetic flux density. This sensor operates with a supply voltage ranging from 2.5V to 38V, meaning it is a versatile option that can be used in various applications. The sensor's chopper stabilized design modulates its signal at a high frequency to minimize noise and DC offset, meaning it is capable of maintaining high accuracy over a wide range of operating conditions.

In the absence of a magnetic field, the DRV5053VA outputs a voltage of 1V. When we expose the sensor to a magnetic field, the output voltage changes linearly, with the direction of the magnetic field influencing the voltage polarity. This sensor is available in models with varying sensitivities, including -11 mV/mT, -23 mV/mT, -45 mV/mT, -90 mV/mT, +23 mV/mT,and +45 mV/mT. For our application, the -90 mV/mT version will be the most suitable, as it provides a response to everyday magnetic fields such as desktop magnets or small electric motors.

One of the more notable features of the DRV5053VA is its capability to determine the polarity of an applied magnetic field. For this model, a south pole magnetic field decreases the output voltage below 1V, while a north pole magnetic field increases it above 1V. This polarity sensitivity is a great addition to this sensor, as we will also be able to observe the direction of magnetic fields.

This sensor also has excellent temperature stability, with its sensitivity only varying by ±10% over an operating temperature range of -40°C to +125°C. While it is not likely that our system will be exposed to extreme temperatures, it is helpful to note that this is a robust sensor that will not experience fluctuations when changing its surrounding environment.

The DRV5053VA is offered in a compact 3-pin surface-mount package, making it much easier to integrate into our custom PCB. Since our system involves integrating 64 of these sensors together, a compact packaging is crucial in minimizing the size of our PCB and simplifying routing.

It is worth considering, however, that this specific sensor is susceptible to errors caused by external magnetic fields unrelated to those we are trying to measure. If we do not properly manage this interference, we may receive inaccurate readings on our display. To mitigate this issue, we must be very careful when designing our PCB, ensuring that the sensor array is properly distanced from other components, such as the battery or voltage regulators. On the software side, incorporating algorithms to filter out noise would also help increase the overall reliability of our product.

In summary, the DRV5053VA Hall-Effect sensor offers a combination of high sensitivity, temperature stability, and ease of integration, making it an excellent choice for precisely measuring magnetic fields in our system. While it is important to mitigate interference when using this sensor, this limitation is minor compared to its significant advantages.

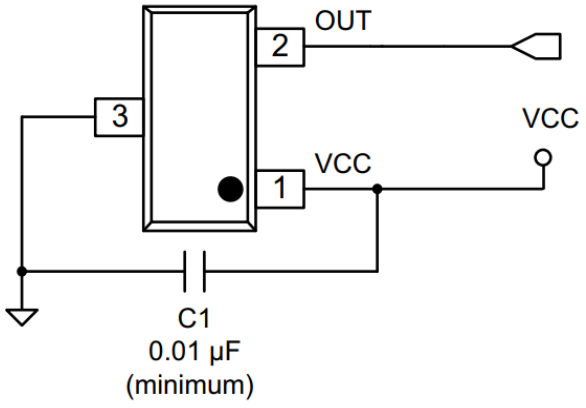


*Figure 3.2.1. Typical Application Schematic - No Filter*

## 3.2.2 DRV425RTJT

The DRV425RTJT, also developed by Texas Instruments, is a high-precision fluxgate Hall-Effect sensor that is used for applications requiring extremely accurate magnetic field measurements. Unlike traditional Hall-Effect Sensors, the DRV425RTJT uses a fluxgate design, which enables it to detect small magnetic fields with high resolution.

One of the more notable features of this sensor is its ability to overcome noise, due to its very low noise floor. This characteristic allows it to detect very small changes in magnetic field strength, making it an ideal option for applications where high-resolution mapping of these fields is essential. This sensor outputs a differential analog voltage, which is beneficial for integration with an ADC to achieve precise digital processing of data.

In terms of operation, the DRV425RTJT operates within a supply voltage range of 4.5V to 5.5V, which is more narrow than the DRV5053VA but still offers flexibility. This sensor also operates from -40°C to +125°C, which makes it suitable for the operating environment of our device.

Despite its excellent resolution, accuracy, and temperature stability, the DRV425RTJT presents a few integration challenges. The sensor is housed in a surface-mount package, which is compact but more complex than the simpler designs of sensors like the DRV5053VA. The fluxgate design requires additional signal conditioning circuitry, which complicates the PCB layout. This added complexity may result in a larger PCB and more difficult integration, particularly for a system like H.A.L.O., which involves arranging 64 sensors in a compact array.

Another concern is the DRV425RTJT's increased sensitivity to external magnetic interference. While this higher sensitivity makes it ideal for detecting weak magnetic fields, it also makes the sensor more vulnerable to interference from nearby magnetic sources. In a tightly packed system like H.A.L.O., where sensors are placed close together, this could lead to inaccuracies if not managed properly. Ensuring proper shielding and a clean PCB layout would be essential to reduce these errors, adding extra complexity to the design process.

In summary, the DRV425RTJT offers high precision and stability but comes with challenges in integration, packaging, and interference management. While it could potentially offer superior performance with careful design, these factors

make it a less ideal option for H.A.L.O., where a more straightforward and compact design is needed.

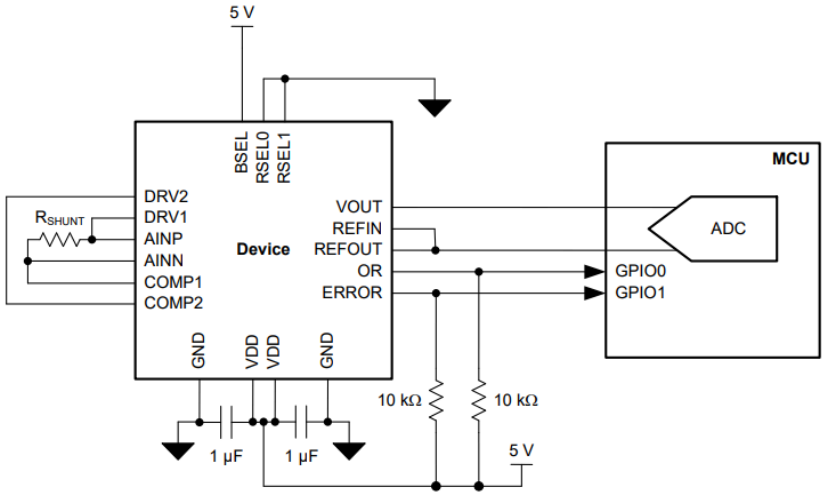



*Figure 3.2.2. Simplified Schematic of a DRV425-Based Linear-Position Sensing Application*

## 3.2.3 A1324LLHLX-T

The A1324LLHLX-T, developed by Allegro Microsystems, is a low-power, linear Hall-Effect sensor that provides an analog output in response to magnetic fields. Its main advantage is its low power consumption, which is essential for battery-operated systems like H.A.L.O. The A1324LLHLX-T operates with a supply voltage ranging from 4.5V to 5.5V, making it compatible with a wide range of power configurations.

One of the key features of this sensor is its ability to provide a ratiometric output, which varies linearly with the magnetic field strength. This ensures that the output voltage is proportionally related to the field strength, making it suitable for generating continuous magnetic field maps. With a typical sensitivity of 2.5 mV/G, it offers a moderate response to magnetic fields, which makes it ideal for general-purpose applications.

The A1324LLHLX-T's temperature coefficient is also quite favorable, with the sensitivity typically varying by ±10% over the temperature range of -40°C to +125°C, similar to the DRV5053VA. This makes the sensor quite stable in varying environmental conditions.

In terms of integration, the A1324LLHLX-T comes in a compact 3-pin SIP package, which simplifies its mounting and reduces PCB space usage. However, like the DRV5053VA, external magnetic interference could affect its accuracy. Adequate shielding and PCB layout considerations will be necessary to ensure precise measurements.

While this sensor offers low power consumption, moderate sensitivity, and easy integration, its lower sensitivity may not provide the resolution needed for higher precision magnetic field mapping. However, it still offers a balanced solution for applications where lower power consumption is prioritized over fine resolution.

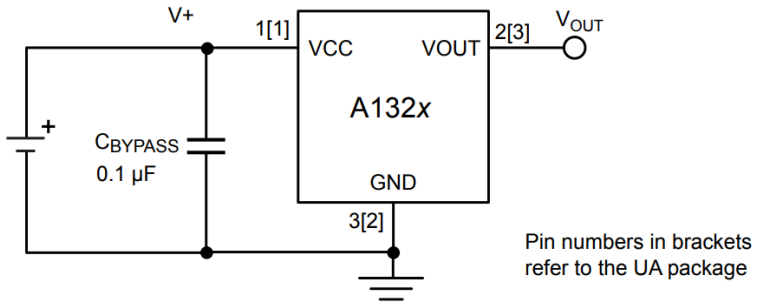



*Figure 3.2.3: Typical Application Circuit*

## 3.2.4 SS495A1

The SS495A1 Hall-Effect sensor, developed by Honeywell, is a radiometric linear sensor designed for magnetic field measurement applications. This sensor provides an analog output proportional to an applied magnetic field. This sensor has a sensitivity of 3.125 mV/G at 25°C, with its resting voltage at 2.5V when no magnetic field is present. The output voltage increases or decreases depending on the polarity and strength of the applied field, allowing for precise measurements. The operational magnetic range extends from -600 Gauss to +600 Gauss, ensuring that the sensor is reliable in environments with strong magnetic fields.

One of the standout features of this sensor is its high temperature stability, allowing it to function in harsh environments ranging from -40°C to +150°C. This range is broader than the DRV5053VA, DRV425RTJT, and A1324LLHLX-T, making it more robust for extreme conditions. However, while the SS495A1 is designed to withstand such extreme temperatures, this feature will not be necessary for H.A.L.O., as it is a desktop device that will operate in a controller indoor environment.

In terms of performance, the SS495A1 has a linearity specification of ±1.5% of span from -600 Gauss to +600 Gauss, ensuring that the sensor delivers accurate readings across its operational range. Its null voltage at 25°C is 2.5V, with a drift of ±0.04%/°C over the temperature range of 25°C to 125°C. This makes the sensor stable and reliable, with minimal variation in its output voltage due to temperature fluctuations.

Another advantage of this sensor is its ease of integration into a custom PCB. The SS495A1 has a compact 3-pin packaging, which is advantageous for H.A.L.O. where space is limited. Due to its simple design and ratiometric output, this sensor is a great option for H.A.L.O. to create maps of magnetic fields.

In all, the SS495A1 has good stability, sensitivity, and ease of integration, which makes it a solid candidate for H.A.L.O. While its temperature tolerance is impressive, this is not a key factor for our application.

### 3.2.5 Comparison & Selection

After comparing the Hall-Effect sensors for H.A.L.O., the DRV5053VA stands out as the most suitable option due to its balance of sensitivity, ease of integration, and stability. While the DRV425RTJT provides higher resolution and low noise for detecting weak magnetic fields, its added complexity in terms of packaging and increased risk of interference makes it less ideal for the design of our system. The A1324LLHLX-T offers low power consumption, but it lacks the precision needed to map magnetic fields from common objects. The SS495A1 has a wide range temperature tolerance, but this feature is unnecessary for H.A.L.O., as it is designed to be a desktop device and will not be exposed to extreme temperatures.

The DRV5053VA's combination of linear output, high sensitivity, and ease of integration make it the best choice for H.A.L.O.'s sensor array. Its small footprint and temperature stability allow it to function effectively within our design, and its ability to detect a wide range of magnetic fields ensures the device can create precise, real-time maps of applied fields. It is also worth noting that this sensor is the most affordable option, which is a crucial factor to consider as we must integrate 64 of them into one printed circuit board. Overall, the DRV5053VA will simplify both the hardware design and integration process, making it the most efficient option for H.A.L.O.

Table 3.2.5: Hall-effect Sensor Comparison

| Component | Sensitivity | Temperature Range | Power Consumption | Cost | Risk of EMI |
|---|---|---|---|---|---|
| DRV5053VA | ±90 mV/mT | -40°C to +125°C | Low | Moderate | Medium |
| DRV425RTJT | ±20 mT | -40°C to +125°C | Moderate | High | High |
| A1324LLHLX-T | 2.5 mV/G | -40°C to +125°C | Low | Low | Low |
| SS495A1 | 3.125 mV/G | -40°C to +150°C | Low | Low | Medium |

## 3.3 Data Storage and Handling

Efficient data storage and handling was a critical component of both the performance and functionality of the H.A.L.O. system. During development, we explored multiple approaches for real-time data acquisition from the Hall-effect sensor array, methods for writing to and reading from the display, and strategies for saving data to physical storage. We carefully evaluated storage solutions to balance speed, capacity, and efficiency. This section described the various storage mediums and options we considered, their implications on the project, and the final selections based on our requirements. Some key factors that influenced these decisions included the need for temporary data buffering, permanent storage of sensor readings and display output, display refresh rates, and power efficiency. Early in the design phase we accounted for the memory constraints of a microcontroller and the display's 480×272 resolution with a 60 Hz refresh rate, as well as the required sensor reading and logging frequency for accurate visualization and analysis. After comparing options, we moved away from an STM32-based design and adopted a Raspberry Pi 4 Model B paired with an ESP32, which eliminated the MCU memory limitations we had previously considered. The final system logged data to a USB flash drive instead of an SD card, providing the speed and flexibility needed for our real-time pipeline while meeting the project's performance goals.

### 3.3.1 RAM

Random Access Memory (RAM) was a crucial consideration for the H.A.L.O. system, serving as the primary medium for temporary data storage, real-time processing, and efficient data transfer between components. During the early stages of the project, we researched several approaches that relied on the STM32's internal memory, including careful allocation of its 564 KB of RAM for single or double-buffered framebuffers and data logging to an SD card. We analyzed memory requirements for a planned $480 \times 272$ display at 60 Hz in RGB565 format, which would have required approximately 261 KB of RAM per framebuffer and over 520 KB for double-buffering. At that time, we also considered using the STM32's Chrom-ART hardware acceleration to optimize graphical operations, and we evaluated adding external RAM modules to meet potential memory demands.

However, after extensive testing and evaluation, we shifted to our final design, which used an ESP32 for low-level sensor management and a Raspberry Pi 4 Model B for display rendering and overall control. This change allowed us to move beyond the MCU's tight RAM constraints. The ESP32's internal RAM was sufficient for buffering data from the Hall-effect sensor array before transmission, while the Raspberry Pi—equipped with gigabytes of system RAM—handled framebuffers, real-time processing, and graphical updates with significant headroom. Because the Raspberry Pi ran a Python-based GUI on a Linux environment, memory management and display buffering were handled through higher-level software libraries, eliminating the need for microcontroller-level RAM optimization or external SDRAM modules.

In the final system, the Raspberry Pi maintained a framebuffer for the 5-inch Hosyond capacitive touch LCD, which operated at $800 \times 480$ resolution, and handled all graphical rendering without the need for external memory expansion. The ESP32 continuously collected and temporarily stored sensor readings from the multiplexed array, transmitting them over a high-baud-rate UART link to the Raspberry Pi for visualization and analysis. RAM allocation on both platforms supported real-time data visualization and processing without introducing latency or performance bottlenecks. By carefully evaluating our options and ultimately leveraging the Raspberry Pi's abundant system memory in conjunction with the ESP32's efficient data handling, the final H.A.L.O. design met our responsiveness and performance targets without the complexity of external memory modules.

### 3.3.2 Flash

Flash memory was the primary non-volatile storage medium considered for the H.A.L.O. system, responsible for retaining critical data even when power was lost. During the early stages of development, we planned around the STM32 MCU's 1 MB of internal flash memory, which we expected to use for firmware storage, calibration data, system configurations, and limited data logging. Unlike RAM, flash memory retains information persistently, making it an attractive option for storing essential system parameters and ensuring that the device could resume normal operation after power cycles or resets. We accounted for flash memory's inherent limitations—such as slower write and read speeds and a finite number of program/erase cycles—during our design research.

Our initial design expected to rely on internal flash primarily for firmware, which would include all embedded software responsible for controlling the MCU, managing peripherals, and executing core system functions. Beyond firmware, we planned to use flash for calibration parameters and system settings, which define sensor thresholds, user preferences, and operational configurations. These parameters needed to persist between power cycles to ensure consistent performance without requiring recalibration on every boot. Because flash has limited write endurance (typically between 100,000 and 1,000,000 cycles per sector), we considered strategies such as wear leveling and buffering writes through RAM to reduce excessive wear.

However, as the design evolved, we moved away from relying on internal MCU flash for operational storage. The final implementation used an ESP32 to handle low-level sensor management and a Raspberry Pi 4 Model B to run the Python-based GUI and coordinate system control. Firmware and configuration data were handled within these platforms' own storage systems, with no need for explicit wear-leveling management. Most importantly, for data logging, we chose to use a USB flash drive connected to the Raspberry Pi rather than the microcontroller's internal flash or an SD card. This provided both the capacity and write endurance needed for continuous data acquisition without the constraints of internal flash.

In the final system, flash memory on the ESP32 and Raspberry Pi was used primarily for storing firmware, calibration data, and persistent configuration files. Real-time sensor data and GUI frame buffering were handled in RAM, while long-term logging of processed sensor data was directed to the removable USB flash drive. By carefully managing write operations and leveraging RAM as a staging area, we preserved flash memory for critical non-volatile storage and ensured that the H.A.L.O. system operated reliably over extended use.

### 3.3.3 SD Card and Flash Drive

The SD card was originally planned to serve as the primary long-term data storage solution for H.A.L.O., responsible for logging sensor data, system diagnostics, recordings of the display output, and any additional information that could be useful to the intended user of the system. Early in development, we considered SD cards attractive because, unlike RAM (which is volatile) and internal flash memory (which has limited write endurance), SD cards offer a high-capacity, removable storage medium capable of handling continuous data logging with minimal system constraints. These factors initially made an SD card appear to be the most practical choice, especially considering their low cost and ease of replacement.

When evaluating storage media, we analyzed write speed as a key parameter, since data had to be logged in real time without introducing system bottlenecks. At the time, we targeted operation at 60 Hz with high-frequency sensor sampling, so we considered class 10 or UHS-I SD cards to ensure sustained write speeds of at least 10 MB/s. We also designed buffering strategies that would use RAM to temporarily hold sensor readings before writing them in batches, reducing the risk of write latency affecting system performance. We planned to interface with the SD card using SDIO mode for higher throughput and lower CPU overhead compared to SPI.

However, as the design progressed and we moved from an STM32-centric architecture to our final implementation, we replaced the SD card with a USB flash drive connected to the Raspberry Pi 4 Model B. The Raspberry Pi, running our Python-based GUI, handled all data logging, and the USB flash drive provided the same removability and high-capacity storage we needed while offering better integration with the Pi's file system and easier access for users. The ESP32 focused solely on managing the sensor array and transmitting readings to the Raspberry Pi, while the Pi managed all long-term storage and retrieval operations.

In the final system, the USB flash drive played the crucial role originally intended for the SD card: it was the primary medium for recording sensor data and diagnostics, offering removability, ample capacity, and sufficient write speed for real-time operation. This solution gave us the same flexibility and durability we sought initially while better fitting the Raspberry Pi–based architecture we ultimately adopted.

### 3.3.4 External Hard Drive

External hard drives (HDDs) and solid-state drives (SSDs) offered substantial storage capabilities, ranging from gigabytes to terabytes, and we initially considered them for H.A.L.O. as potential solutions for extensive data storage. However, integrating such drives presented several challenges that ultimately outweighed their potential benefits.

The physical size and weight of external hard drives were a disadvantage in an embedded system where space was limited. Because we aimed to make H.A.L.O. a standalone and portable system, this factor alone posed a significant compromise. Interfacing and compatibility presented another obstacle. Early in the project, when we were considering an MCU-centric design, the available microcontrollers lacked native support for interfaces commonly used by external drives, such as USB Host or SATA. Implementing support for these interfaces would have required additional hardware components and complex software development, which conflicted with our project timeline. Power consumption was also a concern: mechanical drives, even compact 2.5-inch HDDs, typically exceeded the power budget of an embedded system and would have reduced battery life while demanding more robust power management.

Given these constraints, we determined that external hard drives were not a practical choice for H.A.L.O.'s long-term storage needs. After researching and comparing alternatives, we moved away from both the initial SD card plan and any consideration of HDDs or SSDs, and we ultimately implemented a USB flash drive as our primary storage medium. The flash drive provided the removability, capacity, and compatibility we required while aligning with the physical, power, and integration constraints of our final Raspberry Pi–based architecture.

### 3.3.5 Comparison & Selection

Since H.A.L.O. required RAM for buffering, flash for firmware storage, and a dedicated medium for real-time data logging, we conducted a selection process to identify the most suitable components within each storage category. During development, we compared a range of viable options, including the STM32's internal resources, external memory modules, SD cards, and even larger storage devices. Through this evaluation, we initially considered using an SD card for long-term logging, but after testing and refining the architecture, we ultimately implemented a USB flash drive on the Raspberry Pi for real-time data storage. The following comparisons reflected the options we researched and informed the choices we made for the final system.

Table 3.3.5.1: Ram Comparison

| Component | Type | Size | Speed | Features | Notes |
|---|---|---|---|---|---|
| Internal RAM (STM32) | Embedded SRAM | 564KB | High (tightly coupled) | Directly accessible by MCU | Best for real time operations |
| IS66WV51216EBLL | External SRAM | 8MB | 108MHz | Parallel Interface | Compatible with STM32, but adds complexity |
| AP Memory APS6404L-3SQR | PSRAM | 4MB | 133MHz | SPI / QSPI Interface | Lower power, useful for large buffers |
| Raspberry Pi 4 System RAM | LPDDR4 | 2–8 GB | High | Directly accessible by Pi's CPU/GPU | This was the final choice for display buffering, GUI processing, and high-level operations. |
| ESP32 Internal RAM | Embedded SRAM | 520 KB | High | Local buffering of sensor data | Used in the final design for low-level sensor management. |

During the early stages of development, we considered using the internal 564 KB SRAM of the STM32 MCU as the primary choice for low-latency tasks because of its ease of access and tight integration with the MCU. At that time, we planned for it to handle sensor reading, display buffering, and computational tasks. We also evaluated PSRAM as a viable secondary option due to its low power consumption and QSPI interface, which would have provided larger buffers if needed.

After further evaluation and prototyping, we moved away from the STM32-based approach. In the final design, sensor reading and buffering were handled by the ESP32's internal RAM, while all display buffering and computationally intensive tasks were offloaded to the Raspberry Pi 4's LPDDR4 system memory. This shift eliminated the need for external PSRAM entirely, as the Raspberry Pi provided

ample memory capacity and speed for the H.A.L.O. system's real-time data processing and visualization requirements.

Table 3.3.5.2: Flash Comparison

| Component | Type | Size | Write Speed | Endurance (cycles) |
|---|---|---|---|---|
| Internal Flash (STM32) | Embedded NOR Flash | 1MB | 1MB/s | 100,000 |
| Winbond W25Q128JV | External NOR Flash | 16MB | 50MB/s | 100,000 |
| Micron MT29F2G08A BAEAWP | NAND Flash | 2GB | 20MB/s | 10,000 |
| ESP32 Internal Flash | Embedded NOR Flash | 4 MB | 20 MB/s | 100,000 |

During the early stages of the project, we considered utilizing the internal flash available on the STM32 as the most practical choice for non-volatile storage. At that time, its low power consumption, high reliability, and direct accessibility made it an attractive option, and we believed that making efficient use of its available resources would ease development and reduce cost. The MCU itself had been selected in part because of its onboard flash capacity. We also evaluated external options such as Winbond NOR flash in case additional non-volatile storage became necessary.

As the design evolved, we transitioned away from the STM32-based architecture. In the final implementation, the ESP32's internal flash served as the primary non-volatile storage, holding firmware, calibration data, and configuration parameters. This approach retained the advantages we initially sought—low power, high reliability, and direct integration—while fitting seamlessly into the ESP32's role within the system. Because the ESP32's internal flash provided sufficient capacity and performance, no external flash devices were added in the final design.

Table 3.3.5.3: SD Card and Flash Drive Comparison

| Component | Type | Capacity | Speed | Interface |
|---|---|---|---|---|
| SanDisk Ultra 16GB | SDHC | 16GB | 80MB/s | SDIO |
| Samsung EVO Select 32GB | SDHC | 32GB | 90MB/s | SDIO |
| SanDisk Extreme PRO 64GB | SDXC | 64GB | 170MB/s | UHS-I |
| USB Flash Drive | USB-connected NAND Flash | 16–64 GB (user-swappable) | 20 MB/s | USB 2.0/3.0 |

We initially identified the SanDisk Ultra 16 GB SDHC card as an ideal storage option due to its sufficient capacity, reliable write speeds, and widespread availability. At that time, we determined that higher-capacity SDXC cards were unnecessary, as H.A.L.O.'s data logging requirements would not generate excessive storage demands. We prioritized Class 10 and UHS-I cards to ensure smooth, real-time writing and avoid buffer overruns or latency issues during continuous acquisition.

However, after transitioning to a Raspberry Pi–based architecture, we replaced the SD card with a USB flash drive for long-term data logging. This provided the same advantages in terms of performance and simplicity, while also offering easier user access, native support within the Pi's Linux environment, and better integration with our updated system. The USB flash drive met the same design criteria we initially targeted with SD cards—namely, a balance of performance, power efficiency, and ease of use—ensuring smooth operation without introducing unnecessary complexity.

# 3.4 Microcontroller

A microcontroller (MCU) is a compact integrated circuit designed to execute specific tasks within an embedded system. Unlike general-purpose CPUs, MCUs integrate a processor core, memory (RAM/Flash), and peripherals (I/O, timers, communication interfaces, etc.) into a single chip. Features such as low power consumption, real-time processing capabilities, and adaptability make

them ideal for applications requiring efficient, deterministic control over hardware components.

Some key characteristics shared by most microcontrollers include:

- Power Requirements: MCUs typically operate at low voltages (3.3 V or 5 V) and consume little power, making them suitable for both battery-operated and hardwired applications.
- Programmability: Most MCUs support programming in C, C++, or assembly language, with various development environments available for ease of use.
- Cost-Effectiveness: Compared to CPUs, MCUs are generally far more affordable, often ranging between $5 and $50 depending on the model.
- Adaptability: Integrated peripherals such as GPIOs, communication protocols, and timers allow MCUs to interface with external hardware like sensors, actuators, and displays.
- Analog-to-Digital Converters (ADCs): ADC functionality is critical for converting real-world analog signals (such as sensor data) into a digital format for processing. The selection of an MCU often depends on the resolution, sampling rate, and number of ADC channels it provides.

For H.A.L.O., we initially evaluated several microcontrollers, including STM32 variants, focusing on balancing power efficiency, ADC capabilities, processing power, and development support. After extensive testing and design iterations, we implemented an ESP32 as the system's low-level MCU. The ESP32 managed communication with the multiplexed sensor array, interfaced with external ADCs, and transmitted processed sensor readings to the Raspberry Pi for visualization. This final choice gave us the required adaptability and processing performance while simplifying development and ensuring reliable integration with the rest of the system.

### 3.4.1 MSP430FR6968

The MSP430FR6989, developed by Texas Instruments, was an ultra-low-power 16-bit microcontroller designed for embedded applications requiring efficient processing, extended battery life, and integrated analog peripherals. It belonged to the MSP430 family, well known for low power consumption and a flexible peripheral set.

Some key features of the MSP430FR6989 included a 16-bit RISC (Reduced Instruction Set Computer) CPU with a clock speed up to 16 MHz, 128 KB of FRAM (non-volatile, low-power memory with fast writes) and 2 KB SRAM, 83 general-purpose I/O pins, multiple 16-bit timers with PWM (Pulse-Width

Modulation) support, and an operating voltage of between 1.8 V to 3.6 V. It offered support for $I^2C$, UART, and SPI communication protocols to interface with peripheral devices, a 12-bit SAR ADC with up to 16 input channels and configurable sample-and-hold functionality, and various power modes to suit operational requirements.

In terms of form factor, the MSP430FR6989 was available in both QFP (Quad Flat Package) and VQFN (Very-Thin Quad Flat No-Lead Package), which were both surface-mounted packages, making it suitable for PCB design flexibility. For development and testing, the MSP-EXP430FR6989 LaunchPad was available, providing a USB interface for programming and debugging and breakout headers for easy prototyping. Texas Instruments developed Code Composer Studio (CCS) as the official development environment for the MCU, but there were also some alternative compatible environments available such as IAR Embedded Workbench and Energia (an Arduino-style development environment).

The MSP430FR6989 was a strong contender for H.A.L.O. due to its low power consumption, built-in ADC, flexible peripheral set, and FRAM memory (specifically advantageous for data logging). However, its 16-bit architecture and relatively low clock speed might have limited performance for the complex signal processing required for the project's real-time visualization tasks.

### 3.4.2 ESP32

The ESP32, developed by Espressif Systems, was a high-performance and low-power microcontroller with integrated Wi-Fi and Bluetooth connectivity. It featured a dual-core Tensilica Xtensa LX6 32-bit processor running up to 240 MHz with 520 KB of onboard SRAM. There were 34 general-purpose I/O pins with support for touch sensing and capacitive inputs, along with support for various communication protocols, including $I^2C$, SPI, UART, CAN, and PWM. It was also equipped with a 12-bit SAR ADC with up to 18 input channels. With an operating voltage of between 2.2 V and 3.6 V, the MCU was adaptable to a variety of applications.

The ESP32 was available in various form factors, most notably the surface-mount variant, the ESP-WROOM-32. There were various devkits available for the ESP32, most offering USB support and advanced peripherals, as well as a RISC-V-based power-optimized variant. The ESP32 was extremely well supported, with many of the most common development environments offering specific ESP32 IDEs, including Arduino IDE, Espressif IDE, MicroPython, PlatformIO, as well as extensive Microsoft Visual Studio packages.

In regard to H.A.L.O., the ESP32 was a very compelling option due to its high-performance capability, versatile I/O, and built-in wireless capability. Although our project scope did not call for wireless connectivity at that time, having the option available to our team was considered valuable. The built-in ADC was another advantage of this chip, with the only downside being negative reviews of its performance at higher frequencies. However, paired with the performance of the processor and our application likely not requiring usage of the ADC in the MHz range, we determined that this was not a significant concern.

### 3.4.3 STM32

The STM32 family, developed by STMicroelectronics, was a series of MCUs utilizing 32-bit ARM Cortex-M processors with a wide range of capabilities. The family consisted of chips with various performance levels, from low-power to high-performance real-time options. STM32 MCUs came in various flavors depending on performance needs, with the STM32F3, STM32F4, and STM32L4 being the most relevant for our application.

They featured ARM Cortex-M (M0, M3, M4, M7, or M33) processors running at clock frequencies between 72 MHz and 180 MHz, with 64 KB to 2 MB of flash storage and up to 512 KB of SRAM. The higher-end models had over 100 GPIO pins, with most pins supporting advanced peripheral functions. Multiple 16- and 32-bit timers were available, with support for $I^2C$, UART, SPI, USB, CAN, and Ethernet communication protocols. Depending on the series, the onboard ADCs (there were multiple) were either 12- or 16-bit, with most supporting up to 24 channels.

Despite the high-performance characteristics of the STM32, it operated at a voltage between 1.8 V and 3.6 V, making it suitable for a portable application. In terms of form factor, there were various package sizes available, from LQFP-32 (low pin count) to LQFP-144 (high-performance applications). A popular devkit was the STM32 Discovery Kit, a feature-rich board for rapid prototyping. Like the ESP32, the STM32 was widely supported in terms of development environments and was compatible with Arduino IDE, PlatformIO, Visual Studio Code, and STM32CubeIDE (the official IDE), among others.

The STM32 offered a strong balance between processing power, ADC performance, and efficiency. The high-resolution ADCs made it a strong candidate for use with precise sensor data, and the extensive peripheral set and power management capabilities provided flexibility. However, the STM32

required more effort to set up compared to similar MCUs and, despite being widely supported, lagged behind the popular ESP32 in terms of community resources and ease of development.

### 3.4.4 Comparison & Selection

Table 3.4.4: Microcontroller Comparison

| **Feature** | **MSP430FR6989** | **ESP32** | **STM32** |
|---|---|---|---|
| Architecture | 16-bit MSP430 | 32-bit Xtensa LX6 | 32-bit ARM Cortex-M7 |
| Clock Speed | 16 MHz | < 240 MHz | < 550 MHz |
| Flash Memory | 128 KB FRAM | 4 MB (external) | 64 KB - 2 MB |
| SRAM | 2 KB | 520 KB | 16 KB - 1024 KB |
| GPIOs | 40 | 34 | 100+ |
| ADC Resolution | 12-bit (16 channels) | 12-bit (18 channels) | 12 to 16-bit (24 channels) |
| Operating Voltage | 1.8V - 3.6V | 2.2V - 3.6V | 1.8V - 3.6V |
| Wireless Capability | None | Wi-Fi, Bluetooth | None |
| Interfaces | I2C, UART, SPI, ADC, Timer, PWM | I2C, UART, SPI, ADC, DAC, PWM, CAN | I2C, UART, SPI, USB, CAN, Ethernet, LTDC |

For H.A.L.O., we initially needed a microcontroller that was powerful enough to handle real-time data collection from the Hall-effect sensor array, efficiently process and visualize this data on a touchscreen display, and support additional functionality like data logging, all while maintaining a minimum refresh rate of 60 frames per second. Early in development, we selected both the STM32H735IGT6 from the STM32H7 family and the ESP32 by Espressif, with the expectation that the STM32 would handle display rendering and the ESP32 would manage sensor acquisition. This decision was driven by several factors, primarily related to anticipated bottlenecks and performance requirements.

At that stage, the STM32H735IGT6's 32-bit ARM Cortex-M7 core clocked at 550 MHz, 564 KB of SRAM, and 1 MB of embedded flash memory seemed ideal

for managing a framebuffer and rendering graphics without external memory. The integrated LTDC (LCD-TFT Display Controller) and MIPI DSI interface were also key features, providing a direct, hardware-accelerated path for driving a display and reducing CPU load for screen updates. These capabilities aligned well with our goal of sustaining 60 FPS on our chosen touchscreen.

The ESP32 was selected to handle sensor acquisition, using its high clock rate, GPIO flexibility, and SPI capabilities to address the multiplexers and sample data from external ADCs before sending the readings to the STM32 for processing.

However, as the project evolved, we shifted to a new architecture that better suited our needs. In the final design, the ESP32 retained its role as the low-level controller, managing the multiplexers and reading from the external ADCs. Instead of the STM32, we implemented a Raspberry Pi 4 Model B to handle all data processing and display tasks. The Pi, running a Python-based GUI, rendered magnetic field visualizations on a 5-inch Hosyond capacitive touch LCD with an 800×480 resolution, easily meeting our refresh rate targets. Data logging was redirected from an SD card to a USB flash drive, simplifying user access and improving integration with the Raspberry Pi's Linux environment.

Overall, the final combination of the ESP32 and Raspberry Pi met all computational, memory, and display requirements while providing a more flexible and powerful platform. This architecture allowed H.A.L.O. to deliver real-time, interactive visualization of magnetic field data at a smooth and responsive frame rate, while maintaining a straightforward development workflow and reliable long-term operation.

### 3.4.5 Development Board Selection

To assist in the early development and validation of H.A.L.O.'s firmware and interface, we initially chose to work with the STM32H735G-DK Discovery Kit. At that stage of the project, we were planning to base our custom PCB on the STM32H735IGT6 MCU, and the Discovery Kit offered a close match to those specifications. We also considered other development boards, such as the STM32H747I-DISCO and the STM32F746G-DISCO boards, but none aligned as closely with our target specifications and features as the STM32H735G-DK.

Our decision to use the STM32H735G-DK was motivated by its strong resemblance to the design we had envisioned for our custom hardware. It featured the same MCU we initially selected, along with a similar integrated display, allowing us to refine our UI design and functionality prior to transferring

to a final implementation. The board included native LTDC and MIPI DSI support, features we planned to leverage to drive a capacitive touchscreen at a 60 FPS refresh rate. The onboard ST-LINK V3 debugger further streamlined our testing workflow, eliminating the need for external hardware or breakout connectors. While the board included some components—such as external RAM and flash—that we did not plan to incorporate on our custom hardware, it remained the best option for early prototyping and demoing. It allowed us to validate ADC sampling, touchscreen response, and visualization under realistic conditions.

As development progressed, however, our architecture shifted away from the STM32 platform. The final system used an ESP32 for low-level sensor management and a Raspberry Pi 4 Model B to handle display rendering and data processing, paired with a 5-inch 800×480 Hosyond capacitive touch display and a USB flash drive for logging. Even so, the work we performed on the STM32H735G-DK Discovery Kit proved valuable for early firmware testing, interface exploration, and refining the overall system requirements before transitioning to our final hardware.

Table 3.4.5: Development Board Selection

| Feature | STM32H735G-DK | STM32H747I-DISCO | STM32F746G-DISCO |
|---|---|---|---|
| MCU | STM32H735IGK6 | STM32H747XIH6 (Dual-core: Cortex-M7 + M4) | STM32F746NGH6 |
| Core(s) | Cortex-M7 @ 550 MHz | Cortex-M7 @ 480 MHz + Cortex-M4 @ 240 MHz | Cortex-M7 @ 216 MHz |
| Flash / RAM | 1 MB Flash / 564 KB RAM | 2 MB Flash / 1 MB RAM | 1 MB Flash / 320 KB RAM |
| Display | 4.3" 480x272 TFT (with capacitive touch) | 4.3" 480x272 TFT (with capacitive touch) | 4.3" 480x272 TFT (with capacitive touch) |
| Connectivity | USB OTG HS (micro-B), Ethernet, microSD | USB OTG HS (micro-B), Ethernet, microSD, Wi-Fi expansion | USB OTG HS (micro-B), Ethernet, microSD |
| Expansion | Arduino + STMod+ | Arduino + STMod+ | Arduino |

| | | | |
|---|---|---|---|
| Debugger | On-board STLINK-V3E | On-board STLINK-V3E | On-board STLINK-V3E |
| Price Range | ~$65 | ~$95 | ~$55 |

## 3.5 Sensor Addressing

Sensor addressing will be an important part of how we will be collecting our data. Various addressing techniques exist, including direct GPIO connections, multiplexing, and digital protocols such as I2C and SPI. The choice of addressing method impacts system complexity, power consumption, and speed.

In this project, we are utilizing analog Hall effect sensors, which do not have built-in digital communication protocols such as I2C or SPI. This means that conventional digital addressing techniques cannot be applied, necessitating an alternative approach to efficiently read data from multiple sensors. Instead, we must implement an addressing scheme that allows the microcontroller to access and process data from each sensor individually without excessive wiring or hardware constraints.

To achieve this, we must convert the analog signals from the Hall effect sensors into a format that can be processed by the MCU. This requires an analog to digital converter to interpret the sensor outputs as meaningful data. Since the number of available ADC channels on the MCU is limited compared to the number of sensors in our system, we need a method to expand the MCU's ability to read multiple analog signals. This is where multiplexing becomes essential, as it enables the selection and reading of multiple sensors using a reduced number of ADC channels. Alternatively if enough GPIO pins are available we could utilize ADCs for every sensor output to have a direct connection between the sensors and the GPIO pins.

### 3.5.1 Direct GPIO and ADC

GPIO stands for General Purpose Input and Output. If a microcontroller has enough available GPIO pins, each sensor can be connected directly to its own dedicated pin. By default, GPIO pins are typically configured for digital input, reading high or low voltage levels. This makes them unsuitable for reading analog signals directly. However, if the microcontroller lacks sufficient built in

ADC channels, an external ADC can be used to convert analog signals into digital data that the MCU can process.

This approach offers the fastest data acquisition speed, assuming the selected MCU has enough GPIO pins but not enough ADC channels. Since each sensor has a dedicated connection, signals can be read in parallel without the delays associated with multiplexers. Additionally, this method simplifies circuit design by reducing the need for additional switching components.

However, the approach has some trade offs. It increases the number of required connections, which can impact PCB layout complexity, physical space, and cost. Power consumption may also rise due to the additional GPIO usage and external ADC requirements. In cases where space, cost, or power efficiency are critical factors, using multiplexers or shared ADC channels might be a more practical alternative.

### 3.5.2 MUX

A multiplexer (MUX) is a device that allows multiple input signals to be transmitted through a single output, which is particularly useful when working with a limited number of ADC channels on an MCU.

In this project, we used multiplexing to read the signals from multiple analog Hall effect sensors to a single ADC input. This enables the MCU to read data from multiple sensors without requiring a separate ADC channel for each hall effect sensor.

How we utilized MUX:

- Inputs: The input pins of the MUX are connected to the output signals of multiple Hall effect sensors. With our 16:1 MUXs a single MUX handles two rows of sensors
- Select Lines: These control which input is connected to the output.given that the ESP 32 has plenty of GPIO pins, it directly controls these lines. Otherwise, a binary counter would have been used to cycle through the inputs automatically.
- Enable (or Encoder) Pin: This pin is used to activate the MUX. The MCU controls this pin to enable data transfer when needed.
- Output: The selected sensor's signal is sent to the MUX’s output, which is then connected to an ADC for digital conversion.
  Handling the ADC Connection

Table 3.5.2: Multiplexer Comparison

| MUX | Num Channels | R on (ohms) | Voltage | Switch Time (Ton, Toff) (Max) |
|---|---|---|---|---|
| CD74HC4067M96 | 16:1 | 160 | 2V ~ 6V | 51ns, 49ns |
| ADG706BRUZ-REEL | 16:1 | 4.5 | 1.8V ~ 5.5V | 32ns, 16ns (Typ) |
| 74HC4051D,653 | 8:1 | 120 | 2V ~ 10V | 51ns, 42ns |

### 3.5.3 Row/Column Multiplexing

Multiplexer row/column addressing is a method that uses two multiplexers to read data from a sensor matrix. One MUX is used for selecting rows, and the other is used for selecting columns. For example, in an 8x8 matrix, this method reduces the number of multiplexers to just 2, and each multiplexer would only need to be an 8:1 MUX, which requires only 3 select lines. However, the select lines for the row and column MUXs must be independent of each other, meaning they cannot be synchronized but must be coordinated to select the correct sensor at the row-column intersection.

With this method, the number of lines required to control the matrix would be 6 select lines for the two MUXs and 1 enable pin, totaling 7 lines. While this reduces the hardware required, the speed of reading sensors would also decrease, as only one sensor can be read at a time. This is in contrast to other multiplexing methods where multiple sensors could be read simultaneously, resulting in a slower overall reading process.

A key challenge in this setup is properly controlling the output of the sensors. Since each sensor will be outputting a voltage, sensors not currently selected must have their outputs disabled to avoid interference. This can be accomplished by using tri-state buffers to place the unselected sensors into a high-impedance state, effectively disconnecting them from the data line and isolating the outputs of the active sensor.

### 3.5.4 ADC

ADCs work by sampling an analog signal (a continuous range of voltages) at regular intervals and then converting each sample into a corresponding digital

value. The resolution of the ADC is directly related to how precisely it can divide the analog signal's voltage range. Since this is done with bits, the number of discrete levels it can divide is $2^n$ where n is the number of bits. For instance, with an 8-bit ADC, the signal range is divided into 256 discrete levels , while a 10-bit ADC can divide the range into 1024 levels.

The higher the resolution, the more levels the ADC can use to represent the analog signal, which means that small variations in the input signal can be detected with greater precision.

The minimum detectable voltage change, known as the least significant bit, is determined by dividing the input range by the total number of levels.

$$\frac{input\ range}{2^n}$$

For example, with an 8-bit ADC and an input range of 1 volt, the voltage increment is 3.92 mV. This means that if the voltage of the input signal changes by less than 3.92 mV, the ADC will not be able to detect this change, and the value may stay the same. The table below represents the minimum voltage increments for different resolutions with an input range of 1 volt.

Table 3.5.4: Digitizer step size vs. resolution [6]

| **Resolution** | **Ideal Dynamic range** | **Minimum Voltage Increment** |
|---|---|---|
| 8 bit | 256:1 | 3.92 mV |
| 10 bit | 1024:1 | 0.98 mV |
| 12 bit | 4096:1 | 0.244 mV |
| 14 bit | 16384:1 | 61 µV |
| 16 bit | 65536:1 | 15 µV |

The input range of the ADC is determined by its reference voltage. Typically, ADCs are designed to convert an input voltage that falls within a fixed range. For example, from 0 to 3.3V or 0 to 5V. If the analog signal exceeds this range, it will either be clipped or not properly detected.

To ensure that the analog signal falls within the ADC's input range, you can amplify the signal using an operational amplifier (op-amp) or other types of signal conditioning. This can be particularly useful if the signal's voltage range is too small compared to the ADC's input range.

The sampling rate determines how frequently the ADC samples the analog signal. The faster the sampling rate, the more frequently the ADC will update the digital representation of the signal, allowing for a smoother and more accurate tracking of fast changing signals which is an outcome that we are hoping for with our project.

According to the Nyquist theorem, to capture a signal without aliasing , the sampling rate must be at least twice the highest frequency component of the signal. This ensures that each waveform is sampled sufficiently to capture its full shape. If you sample too slowly, the ADC may miss important variations in the signal, leading to distortion.

For example, if you are sampling a sine wave with a maximum frequency of 1 kHz, the sampling rate should be at least 2 kHz to avoid aliasing. For the purposes of measuring changing magnetic fields we will try to ensure that our ADCs can over compensate to avoid aliasing.

Many MCUs come with built in ADC channels, however if we do not have enough ADC channels or if the ADC channels are not a high enough resolution to provide meaningful information then we may need to add external ADCs.

### 3.5.4.1 SAR ADC

A SAR (Successive Approximation Register) ADC outperforms Δ-Σ devices in speed and delivers higher accuracy than flash architectures, making it a popular choice for multiplexed data acquisition systems.

Most front end signal conditioners scale their outputs to the converter’s standard 0–5 V input span. During each conversion the SAR’s sample and hold capacitor first captures this conditioned voltage. An on-chip DAC then generates a reference level that encodes the SAR’s provisional digital estimate. The input and reference are compared; the comparator’s output updates the register, and the DAC adjusts accordingly. This binary search runs n iterations—one per resolution bit—until the code that best represents the original signal is reached.

Because SAR cores lack built-in anti-aliasing filters, frequencies above half the selected sample rate will fold back as aliases if they reach the input. Once digitized those errors are permanent, so we must either sample above the Nyquist limit of every signal of interest or insert adequate low-pass filtering ahead of the converter. [7]

For the sake of this project sampling above Nyquist is the easier option.

### 3.5.4.2 Delta-sigma ADCs (ΔΣ)

Delta Sigma allows for more accurate resolutions when compared to SAR and Flash ADCs. This comes at the cost of speed. They work by capturing the input many times faster than the final output rate—often hundreds of oversamples for every delivered sample. Their on-chip DSP then decimates this torrent of data, distilling it into a single, high-resolution word (24 bits is typical). Heavy oversampling and a built in low pass filter enables the device to be resilient to noise and aliasing without any external modifications like implementing filters or sampling above Nyquist. The trade-off is speed: because so much time is spent oversampling and filtering, delta-sigma ADCs top out at only a few hundred thousand samples per second. Because they are so accurate they are typically used for high end audio or vibration data acquisition.

### 3.5.4.3 Flash ADC

A flash, or direct-conversion, ADC attains the highest sampling speeds of any architecture by digitizing the input in a single comparison step. A precision resistor ladder generates $2^N - 1$ evenly spaced reference voltages, each feeding its own comparator while the analog signal drives the opposite inputs. All comparators fire simultaneously, creating a thermometer code that is immediately encoded into an N-bit binary word. This simplicity yields multi-gigahertz sampling with sub-nanosecond latency, but practical resolution tops out at roughly 4–8 bits because comparator offsets and ladder mismatches rapidly erode linearity. The need for $2^N - 1$ comparators translates into large die area, high input capacitance that demands sample-and-hold buffer, and substantial static power as every comparator remains biased. Despite these costs, the architecture's unmatched speed makes it the default choice for oscilloscopes, RF digitizers, and other ultra-wide-bandwidth instruments.

### 3.5.4.4 ADC Type Selection

Table 3.5.4.4: ADC Comparison

| ADC Type | Max Resolution | Max Sample Rate |
|---|---|---|
| SAR | 18 bits | 10 MHz |
| Delta-sigma | 32 bits | 1 MHz |
| Flash | 12 bits | 10 GHz |

We have selected SAR as our preferred method of ADC because it provides a balance between speed and resolution. While our priority was speed, the resolution provided by the flash ADC as well as the footprint it would take up made it an unattractive choice.

## 3.5.5 ADC Selection

### 3.5.5.1 ADS7883SDBVR

The ADS7883SDBVR developed by texas instruments is a compact, low-power, capacitor-based SAR analog-to-digital converter that delivers 12-bit resolution at speeds up to 3 MSPS. It pairs an on chip sample and hold with a simple three wire SPI interface: the input is latched on the falling edge of CS, and each bit is shifted out with SCLK, enabling glueless connection to most microcontrollers and DSPs. Operating from a single 2.7 V – 5.5 V rail, the device draws only a few milliamps at full speed and offers a power down mode for duty cycled operation, making it a good fit for battery powered systems. Its digital input pins tolerate levels up to 5.5 V regardless of the actual supply voltage, which simplifies mixed-voltage designs and relaxes power-up sequencing requirements. Housed in a tiny six-pin SOT-23 package and specified over the –40 °C to +125 °C range, the ADS7883 slots easily into space-constrained, wide-temperature applications that need fast, high-resolution data acquisition without sacrificing power budget.

The speed that this ADC operates at makes it a good choice for reading data from sensors and providing a fast refresh rate. Its small form factor also makes a good choice for a small electronic device like the H.A.L.O. sensor array. With a 12 bit resolution it provides enough accuracy to acquire meaningful sensor data to be displayed.

The device is priced at $4.73 with a minimum order of 1 with 2,907 available to ship immediately.

## 3.5.5.2 AD7940BRMZ

The AD7940 is developed by Analog Devices Inc. and is a 14-bit SAR ADC that combines high resolution with low power draw in a compact SOT-23 package. It runs from a single 2.5 V – 5.5 V supply, and achieves throughput rates up to 100 kSPS while its low noise, wide bandwidth track and hold handles input frequencies beyond 7 MHz. Conversions are kicked off on the falling edge of CS—the same moment the input is sampled, which removes pipeline latency to obscure time-critical measurements. A simple three wire interface (CS, SCLK, SDO) lets any MCU or DSP dictate conversion timing; the SCLK frequency directly sets the sample rate, enabling dynamic power management. Because the reference is internally tied to VDD, the ADC accepts a full-scale input swing from 0 V to the supply rail without extra components. Advanced design techniques keep active-mode dissipation low, and a shutdown state slashes supply current to just 0.5 μA when the device is idle, making the AD7940 a strong fit for battery-powered or duty-cycled data-acquisition systems that need true 14-bit accuracy without the size or power penalty of larger converters.

This ADC would allow our sensors to have a higher resolution which would let H.A.L.O. be able to detect smaller fluctuations in magnetic field intensity. This would be useful because the fields we will be measuring will already be small and won't have too much fluctuation. This sensor does have a loper sampling speed at 100 kSPS which could hinder our frame count goal.

The device is priced at $16.90 with a minimum buy of 1 with 849 available to ship immediately.

## 3.5.5.2 ADC081C021CIMKX/NOPB

The ADC081C021 and ADC081C027 from Texas Instruments are low-power, 8-bit successive-approximation (SAR) analog-to-digital converters that run from a single +2.7 V to +5.5 V supply, which also serves as the reference voltage. They communicate over a 2-wire $I^2C$ interface that supports Standard (100 kHz), Fast (400 kHz), and High-Speed (3.4 MHz) modes, enabling throughput rates up to 188.9 kS/s with a typical conversion time of 1 μs. An internal track-and-hold handles input frequencies to roughly 11 MHz, while integral and differential

non-linearity are each held to ±0.2 LSB. Power consumption is just 0.26 mW at 3 V (0.78 mW at 5 V) during conversions and drops below 1 µW in the automatic power-down state when idle. Both devices include a programmable upper- and lower-limit “Alert” comparator that can continuously monitor the input and assert an interrupt if the measured voltage goes out of range, freeing the host MCU until attention is needed. Address flexibility differs by package: the VSSOP-8 version of the ADC081C021 offers nine pin-selectable addresses, while its SOT-23-6 counterpart provides one fixed address with an ALERT pin; the ADC081C027 (SOT-23-6) supplies three pin-selectable addresses. Each device is protected to ±8 kV HBM on the I²C pins, specified over –40 °C to +105 °C, and housed in tiny SOT-23-6 or VSSOP-8 footprints ideal for space- and battery-constrained designs such as system monitors, portable or medical instruments, peak detectors, and general test equipment.

The device is priced at $1.13 with a minimum buy order of 1 and 7,713 currently available to ship.

Table 3.5.5.2: ADC Comparison

| **ADC** | **Resolution** | **Sample Rate** | **Communication Protocol** | **Cost** |
|---|---|---|---|---|
| ADS7883SDBVR | 12 bit | 3 MSPS | SPI | $4.73 |
| AD7940BRMZ | 14 bit | 100 kSPS | SPI, DSP | $16.90 |
| ADC081C021CIMKX/NOPB | 8 bit | 188.9 kSPS | I2C | $1.13 |

The ADS7883SDBVR by texas instruments was chosen primarily because of its speed. It allows us to target a higher refresh rate than the other ADCs researched. It also has a lower cost and plenty available to ship immediately making ordering replacements in the event of issues easy.

## 3.5.9 Comparisons & Selection

Given the ESP32 MCU's 25 GPIO pins, we have more than enough resources to directly connect the sensor array without needing additional components like I/O expanders or binary counters. It is unlikely that expansion will be needed for

other parts of the project since the addressing will not utilize too many pins. This simplifies the design and avoids unnecessary hardware.

With 25 GPIO pins it is not possible to utilize the direct GPIO and external ADC method however we have decided against this for a number of reasons. Having a dedicated ADC for each sensor would take up far too much space on the board and more than double the size of the sensor array. Having to solder every sensor onto the board already presents a challenge where we could introduce error, but nearly doubling the total components that need to be soldered on creates potential for error that is undesirable. Each of the ADCs would also greatly increase the power consumption which would put us further away from our goal of a low power device. While it may improve the speed that we could collect data, the cost for that many ADCs would inflate our budget by more than what the team would like.

The ESP32 has built-in ADC channels that we could utilize, however we decided against using them because their performance was significantly lower than the external ADCs that we could use instead. The ADC we chose was ADS7883SDBVR from Texas Instruments which is a SAR ADC that has a 12 bit resolution and 3 million samples per second.

When comparing the two multiplexing methods, row/column addressing offers some advantages, but the drawbacks make it less ideal for our application. Row/column addressing would reduce the number of MUXs needed to read the sensor data, but it would also require more GPIO pins and would take longer to read all the sensors. Additionally, isolating the outputs of the sensors becomes more complicated, requiring tri-state buffers or other methods to prevent interference, ultimately adding complexity to the design.

Instead, the optimal choice is to use 4 16:1 MUXs, with each MUX reading two rows of the sensor matrix. This solution uses 1 external ADC, simplifies sensor selection (requiring only 4 GPIO pins for selection and 4 for enable), and offers faster sensor reads. Synchronizing the select pins is also easier, which streamlines the process of selecting sensors to read. Because we will be using 1 external ADC, the MUXs will need to be enabled and disabled.

## 3.6 User Interface

When determining a display and output solution for H.A.L.O., we narrowed down our selection to two primary options: an onboard RGB LED display and an onboard embedded touchscreen display with a graphical user interface (GUI). Each option presented unique advantages and disadvantages when considering

functionality, cost, complexity, ease of implementation, ease of use, and overall suitability for the project's needs. Throughout development, we evaluated how these factors aligned with both our current requirements and our anticipated future needs.

After analyzing these options, we concluded that the RGB LED approach, while simpler and less costly, lacked the flexibility and resolution necessary for real-time visualization of magnetic field data. The embedded touchscreen display with a GUI offered greater functionality, interactivity, and the ability to present more detailed visual output, despite being more complex to implement.

Based on this analysis, we ultimately selected the embedded touchscreen display solution. In the final system, we implemented a 5-inch Hosyond capacitive touchscreen operating at 800×480 resolution, driven by a Python-based GUI running on a Raspberry Pi 4 Model B. This choice provided the visual clarity, responsiveness, and user interface flexibility required for H.A.L.O., ensuring the system met both our immediate project goals and our anticipated future needs.

### 3.6.1 RGB LED Display

Utilizing an RGB LED display was a simple and cost-effective method that allowed us to visualize heat map data, acquired from the Hall-effect sensor array, by using an array of light-emitting diodes that used a combination of red, green, and blue light sources in a single package. These kinds of displays ranged from small, segmented indicators to large matrix panels capable of displaying simple text, symbols, and even low-resolution graphical interfaces. For H.A.L.O., an LED display could have been used to represent real-time magnetic field intensity using various color schemes or graphical patterns, providing a highly visible, low-power, and simple means of data representation.

One of the key advantages that made an LED display an attractive option was its cost-effectiveness. Basic models could be acquired for as little as $20 depending on size and resolution, with custom solutions being even cheaper and more customizable (although likely less dense). Power consumption would have been very manageable, especially if considering an OLED or low-brightness LED matrix. Unlike LCDs that required backlighting, an LED display did not require additional illumination, increasing its efficiency.

From an implementation perspective, RGB LED displays were generally easy to integrate with microcontrollers, as many models used serial communication protocols including SPI, $I^2C$, and UART, which simplified the wiring and

minimized the number of pins required. Some displays could also be used with direct GPIO control allowing for rapid updates, but they might have required some additional external hardware to ensure the LEDs were driven effectively. Popular models like the Adafruit 64×64 RGB LED Matrix provided well-supported libraries for microcontrollers like the STM32 and ESP32, reducing development complexity.

However, RGB LED displays had significant limitations in terms of resolution and interactivity. Even with a dense LED matrix, the level of detail that could be conveyed was very limited. This made it more difficult to display complex numerical values, intricate visualizations, or real-time graphs. If a high level of detail had been implemented on an LED matrix display, it would have had to be physically large, increasing both power consumption and cost. Another drawback was that LED displays lacked native interactivity support, meaning that any user input would have needed to be handled through external physical buttons, rotary encoders, voice commands, or some other kind of interface. This would have added additional complexity to the overall system design, as an independent input method would have had to be integrated alongside the display.

Another key factor that had to be considered was scalability (or adaptability). If future modifications to H.A.L.O. had required more detailed visualization of data, or different display modes, transitioning away from an LED matrix would have been necessary. While their simplicity was useful for initial implementation, LED displays did not provide a long-term scalable solution, especially if interactive data visualization or high-resolution output was required.

### 3.6.2 Touchscreen GUI

In contrast to an RGB LED display, an embedded touchscreen display offered a far more sophisticated and interactive means of visualizing data. These displays ranged from small TFT touchscreens to larger full-color capacitive screens, enabling real-time graphical representations, user interactions, and enhanced control capability. For H.A.L.O., an embedded touchscreen could be used to visualize the heatmaps of magnetic field intensities, provide real-time graphs, allow for the addition of various display modes, and even allow users to interact with the data in a dynamic fashion.

The biggest advantage of using a touchscreen over an LED matrix display was its native support for full interactivity. Unlike an LED display, which was strictly an output medium, a touchscreen allowed for direct user input, eliminating the need for development of external control interfaces and streamlining the

integration of hardware and software. This significantly simplified system design, as the user controls were built directly into the interface, and if needed could be completely overhauled to meet the needs of our design as it matured. Additionally, interactions like zooming, selecting data points, or switching between different display modes were made significantly more intuitive through the use of a touch-based interface, and allowed for the overall system to fit in a smaller footprint.

However, this added functionality came at the cost of increased expense and complexity, on top of greater processing requirements. Embedded touchscreens typically cost between $30–$200 depending on size, resolution, display technology, and touch technology. The power consumption of an LCD with backlighting was significantly greater than that of an LED matrix, which could have been a concern if H.A.L.O.'s final power budget had been highly constrained.

The implementation complexity of a touchscreen display was another important consideration, which presented a potential bottleneck in our development cycle. Unlike an LED display, which could be updated with simple framebuffer write operations, a touchscreen required a graphical user interface (GUI) framework to manage rendering, touch input, and real-time updates. Libraries such as LVGL (Light and Versatile Graphics Library), TouchGFX, or Nextion's HMI tools could significantly aid in UI development, but they required additional processing power as well as sufficient RAM to handle framebuffer storage. Microcontrollers working with limited processing resources could struggle or fail entirely in rendering high-resolution visuals while simultaneously managing sensor data acquisition and system control.

To address this challenge, we also evaluated touchscreen displays with integrated processing units that handled UI rendering independently. Displays such as the Nextion HMI Series or Waveshare TFT touchscreens offloaded rendering tasks, reducing the computational burden on the microcontroller. However, these solutions increased cost and required additional development to configure and customize the interface.

When selecting an embedded touchscreen, some of the key specifications we considered included:

- Screen size and resolution – Higher resolution allowed for more detailed visualization but at the cost of increasing data processing requirements significantly.

- Touchscreen type – Resistive screens worked with any input (stylus, gloves) but were less responsive, while capacitive screens provided a smoother experience but required direct skin contact.

- Communication interface – SPI-based displays were common but had limited refresh rates, whereas parallel RGB, LVDS, or HDMI connections were better suited for high-speed rendering.

- Development support – Some screens had pre-built firmware and GUI tools, making implementation much easier.

From a long-term perspective, using an embedded touchscreen provided far more flexibility than an LED display. It expanded our options to allow the development of interactive features, real-time data visualization, and future expansions without requiring significant hardware changes. In the final design, we implemented a 5-inch Hosyond capacitive touchscreen operating at 800×480 resolution, driven by a Raspberry Pi 4 Model B running a Python-based GUI. This solution delivered the interactivity, visualization quality, and flexibility we required, at the cost of higher computational demands, increased power consumption, and a more complex software development process—tradeoffs we accepted to meet the project's goals.

#### 3.6.2.1 Capacitive vs. Resistive Touchscreens

When considering touchscreen technology, two varieties were available in the form of capacitive and resistive displays. Capacitive touchscreens were more responsive and supported multi-touch gestures, making them the better choice when it came to smooth and intuitive interactions. They relied on the conductive properties of the human body, meaning that direct skin contact or a capacitive stylus was required for them to function. Their high transparency and lack of mechanical layers allowed for better color accuracy and brightness. Capacitive touchscreens also fared better over time, as they were more durable and did not suffer from mechanical wear due to repeated presses. However, they were more expensive than their counterparts and did not work well with non-conductive inputs like gloves.

On the other hand, resistive touchscreens relied only on pressure applied to the screen, making them compatible with any input method, including gloves, styluses, or any other firm object. They were generally more affordable and functioned in a wider range of environmental conditions. However, they were less responsive and did not support gestures requiring multiple inputs. The

mechanical layers situated above the display also reduced the clarity and brightness of the screen compared to capacitive touchscreens.

For our needs, since the touchscreen needed to function mostly as a data output with limited interactivity (primarily for navigation through menus and simple “button presses”), a capacitive touchscreen was the better option. This choice provided superior display quality and ensured that the limited touch functionality we needed to implement remained reliably responsive.

### 3.6.3 Comparison & Selection

The choice between an RGB LED display and an embedded touchscreen ultimately depended on the current and potential future requirements of H.A.L.O. If the primary need had been to display simple status indicators of basic magnetic field intensity changes, the LED display would have been the most cost-effective and efficient choice. Because it was relatively easy to implement, required minimal computational resources, and had abundant support for various microcontroller configurations and communication protocols, it would have been advantageous if its lack of interactivity and limited resolution had not been a hindrance to the goals of the project.

However, because we required real-time, high-resolution data visualization in tandem with the capability for user interaction, the embedded touchscreen display was the more suitable choice. It enabled intuitive controls, supported detailed graphical representations, and offered a scalable interface that could dynamically grow and adapt to fit the needs of H.A.L.O. While it was the more capable solution, it came at the cost of additional hardware and software complexity, demanded greater processing power, and increased the overall system cost.

After comparing several options, we ultimately selected a 5-inch Hosyond capacitive touchscreen operating at 800×480 resolution, driven by a Raspberry Pi 4 Model B running a Python-based GUI. This combination met our requirements for responsiveness, clarity, and interactivity while providing a platform that could support future feature expansions. Below is a comparison of some of the displays we considered, along with their features and our final selection.

Table 3.6.3: Screen Comparison

| Feature | Best LED Option | Best Touchscreen Option |
|---|---|---|

| | | |
|---|---|---|
| Cost Efficiency | Adafruit 32x32 RGB Matrix ($40) | Rocktech 4.3” RK043FN66HS-CTG ($45) |
| Best Resolution | Waveshare 64x64 RGB ($70) | Hosyond 5” Capacitive Touch MIPI DSI LCD Display |
| Fastest Refresh Rate | Pimoroni Interstate 75 (>120 FPS) | Nextion 5.0" Enhanced HMI (>80 FPS) |
| Easiest Development | Pimoroni (Prebuilt drivers) | Nextion (GUI pre-built tools) |
| Interactive UI | No interactivity | Nextion or Rocktech |
| Lowest MCU Load | Pimoroni (integrated driver) | Nextion or Rocktech (offloads rendering) |

Early in development, we considered using the Rocktech 4.3-inch display because, paired with the STM32H735IGT6 MCU we initially selected, it offered a visually pleasing, bright, and responsive interface that was capable of achieving our desired frame rate. Its highly customizable UI framework would have accommodated the various GUIs we needed to incorporate for data visualization. The Rocktech panel also featured an LTDC (LCD-TFT Display Controller) that could integrate with the same functionality on the STM32, offloading some of the graphics processing and allowing for more efficient CPU usage. At the time, we valued that the same Rocktech display was included on the STM32H735G-DK Discovery Kit, which would have allowed us to streamline hardware development and focus on integrating the display through software-based approaches.

However, as the project evolved, we transitioned away from the STM32-based architecture. In the final design, we selected a 5-inch Hosyond capacitive touchscreen operating at 800×480 resolution, driven by a Raspberry Pi 4 Model B running a Python-based GUI. This change gave us greater flexibility, higher resolution, and stronger community support while still meeting our performance and refresh-rate requirements. Although the Rocktech display was well suited for the STM32 platform we initially planned, the Hosyond display ultimately aligned better with our final system architecture and development goals.

## 3.7 Communication Protocols

Communication protocols are the rules for sending information between devices. In the case of our project, we needed to select a communication protocol that would be able to facilitate communication between our touchscreen and our Pi, our Pi and our MCU, and our MCU and ADC.

Protocols can be described as being simplex, half duplex or full duplex. A simplex protocol is one that only allows for a single direction of communication, meaning device A could send information to device B, but device B could not send it back to device A. Half duplex allows for communication in both directions but not at the same time. This means that device A could send information to device B and device B could send information to device A, but while one of them sends information ,the other must be on standby and hold the transmission of their information. Full duplex allows for simultaneous transmission of information. This means that device A and device B can both be sending and receiving at the same time without the need to hold.

There is also synchronous and asynchronous transmission. Synchronous transmission uses a clock signal to ensure that the devices are in sync. This requires an extra line for the clock. Asynchronous communication does not use a clock line but instead relies on start and stop bits in the transmission to signal the beginning or end of a transmission. It also requires that both the devices have the same baud rate. Synchronous communication is used for high speed continuous data transfer while asynchronous is used for low speed intermediate transfer.

### 3.7.1 I2C

I2C (Inter-Integrated Circuit) is a widely used communication protocol for low-speed communication with peripheral devices. It operates in half-duplex mode and utilizes synchronous communication with only two lines.

- SDA (Serial Data Line), which is used for bidirectional data transmission
- SCL (Serial Clock Line), which is used to synchronize the transmission of data using the clock

I2C typically uses a 7-bit address, though in rare cases, a 10-bit address may be used for applications that require more devices on the bus. The standard speed of I2C is 100 kbit/s (standard mode), but it can also operate at 400 kbit/s (fast

mode) for faster communication. In some cases, I2C can reach speeds of 3.4 Mbit/s in high-speed mode.

In I2C, data is transmitted one byte at a time. After each byte, the receiver sends an acknowledgment (ACK) bit. This acknowledgment confirms that the byte has been successfully received. If the sub device cannot receive the data, it sends a negative acknowledgement (NACK) to indicate an issue. If the main does not receive an ACK from the sub, it knows there was an issue with the transmission, and it may retry or take corrective action.

If a device does not respond with a NACK, the main device can decide whether to resend the data or handle the error in another way. Conversely, if the sub device receives a NACK from the controller, it knows that the main has stopped requesting more data, and it should halt further transmission.

I2C also allows for multiple masters and multiple slaves. Only one main will be able to use the bus at a time, and they will be ranked on priority so that the highest priority main will use the bus first.

### 3.7.2 SPI

SPI (Serial Peripheral Interface) is a widely used communication protocol for high-speed communication with peripheral devices. It operates in full-duplex mode and utilizes synchronous communication with four lines.

- CS (chip select) which allows the main to choose which sub to send the signal to
- SCLK (serial clock) carries the clock signal from the main
- MOSI (main out, sub in) data sent from the main to the sub
- MISO (main in, sub out) data sent from the sub to the main

The four lines enable full-duplex transmission, allowing data to be sent and received simultaneously on both MISO and MOSI. When transmitting data, the MOSI on the main connects to the MOSI on the sub, and the MISO on the main connects to the MISO on the sub. Data is transmitted with the most significant bit first. All subs share the same MISO and MOSI lines but each has its own CS line. If additional subs are needed, the IO must be expanded to accommodate the extra CS lines.

SPI does not have a maximum clock speed and it doesn't use addresses for sending information. Instead, it uses the CS line to choose the sub to send information to. It also does not have built-in error correction.

Because its signal degrades over long distances it is typically only utilized for short distance applications like connecting devices on the same board.

### 3.7.3 UART

UART (Universal Asynchronous Receiver-Transmitter) is a hardware communication protocol used for asynchronous serial communication. It can operate in three modes: simplex, half-duplex, and full-duplex. UART uses up to two lines for communication:

- TX (Transmit) is used to send data to a receiver
- RX (Receive) is used to send data back to the transmitter

In its simplex configuration, only the TX line is needed because communication occurs in one direction. In half-duplex and full-duplex configurations, both TX and RX lines are required. However, in a half-duplex, only one line can transmit information at a time. These lines only support a connection between a transmitter and a receiver meaning the protocol does not allow for the connection of multiple devices.

When UART data is transmitted, it is done using framing so that the receiving device knows when a transmission starts and stops. This is done by changing from a logical high (1) to a low (0) for the start bit. Following the start bit are the data bits. In some cases, a parity bit follows the data bits to check for errors in transmission. Finally, the stop bit switches back to logical high.

Because UART is asynchronous, it does not require a clock signal. However, this means other parameters must be synchronized between the transmitting and receiving devices, such as voltage levels, baud rate, parity bit, data bit size, and stop bit size.

Additionally, flow control mechanisms (either hardware-based with RTS/CTS or software-based with XON/XOFF) may be used to manage data flow, especially in more complex systems.

### 3.7.4 Comparison & Selection

Table 3.7.4: Communication Protocol Comparison

| Spec | I2C | SPI | UART | MIPI DSI |
|---|---|---|---|---|
| **Speed** | Slow | Fast | Slow | Fast |
| **Distance** | Long | Short | Short | Short |
| **Clock** | Synchronous | Synchronous | Asynchronous | Synchronous |
| **Configuration** | Half duplex | Full duplex | Simplex, Half Duplex, or Full Duplex | High-Speed Serial |
| **Connectable devices** | Multi | Multi | Single | Single |
| **Lines** | 2 | 4 | 2 | 4-6 |

For our project we decided to use SPI for communication between the ADC and MCU. This is because SPI is a fast protocol that would allow us to transmit our samples quickly for processing. We used SPI in a simplex configuration because we only needed information out of the ADC and didn't need to send any information to it from the MCU. We also used UART for communication between the MCU and the PI to transmit the values to be shown on the screen. Because of the screen we picked we also used MIPI DSI for communication between the Pi and the screen.

# 3.8 Languages, Tools, and Development Flow

## 3.8.1 Embedded Development

### 3.8.1.1 C

C was a widely used language for embedded systems and microcontroller programming because it was efficient, portable, and interacted with hardware at a low level. For our system, C was a strong candidate for the MCU because it provided direct memory access and deterministic execution, which were crucial for real-time applications like reading from an ADC and updating the touch screen display. It was well supported by STM32CubeIDE (our chosen development environment), ensuring that we had robust toolchain compatibility. Also, most STM32 HAL (Hardware Abstraction Layer) and LL (Low-Layer) libraries were written in C, making it the most compatible option. C also

benefited from an extensive community of developers, which eased the development process for us as we had a wealth of documentation and troubleshooting resources. While C was procedural and lacked object-oriented features, it still allowed us to develop highly optimized and well-structured code. Given our real-time constraints and reliance on embedded peripherals, C remained the most practical choice.

### 3.8.1.2 C++

C++ built on C with object-oriented programming features, which could provide modularity and scalability benefits. One of its main advantages was the ability to implement abstraction for our peripherals, which could result in a cleaner and more reusable codebase. The language also supported the Standard Template Library (STL), which was not the most ideal for embedded systems due to memory constraints, but could still improve development efficiency by providing some useful data structures.

However, C++ introduced additional complexity and runtime overhead, such as virtual function calls, which could negatively impact performance on an embedded system. While STM32CubeIDE did support C++, the available STM32 HAL and LL libraries were primarily written in C, necessitating a mix of C and C++ had we chosen to utilize embedded programming features. Given our needs, sticking with C for core firmware development was the most practical choice, though we used C++ selectively for higher-level abstractions like managing the display interface.

### 3.8.1.3 Comparison and Selection

Table 3.8.1.3: Programming Language Comparison

| **Feature** | **C** | **C++** |
|---|---|---|
| Performance | High, deterministic execution | Slightly lower due to OOP overhead |
| Memory Usage | Low, efficient memory management | Higher due to vtables, OOP features |
| Hardware Interaction | Direct access, low-level control | Can use abstraction, but requires mixing with C for HAL/LL compatibility |
| Toolchain Support | Fully supported in STM32CubeIDE | Supported, but HAL/LL libraries require C interop |

| | | |
|---|---|---|
| Code Modularity | Procedural, structured programming | Supports OOP, better for abstraction and reuse |
| Community Support | Large embedded systems community | Smaller but still viable for embedded systems |
| Real-time Suitability | Excellent, predictable execution | Potential overhead from dynamic features |
| Ease of Development | Simple but requires careful structuring | More structured but adds complexity |

C was the preferred language for our embedded development due to its efficiency, direct hardware access, and full compatibility with the toolchains and SDKs we used for the ESP32. It ensured deterministic execution, which was crucial for real-time tasks such as acquiring data from the external ADCs and streaming those readings to the Raspberry Pi for visualization. While C++ offered modularity through object-oriented programming, the additional runtime overhead and integration challenges made it less suitable for the core firmware running on the ESP32. However, we used C++ selectively for higher-level abstractions, such as managing the display interface on the Raspberry Pi side, where system resources and processing power were not as constrained.

## 3.8.2 Graphics Rendering

### 3.8.2.1 Python

Python was widely used for data visualization and analysis, which initially made it only a potential candidate for generating the heat map. Early in the project, we identified several challenges when considering Python for real-time embedded applications. It was an interpreted language that required significant processing power and memory, making it unsuitable for the resource-constrained environment provided by our originally planned MCU. While MicroPython existed, it was not optimized for real-time tasks such as ADC data acquisition and display rendering. At that stage, we expected that if Python was used at all, it would be limited to external post-processing of sensor data for debugging or analysis purposes, and we concluded that Python would not be a feasible option for real-time heat map rendering.

However, as the design evolved and we transitioned to a Raspberry Pi 4 Model B to handle data visualization, Python became an integral part of the final system. The Pi's significantly greater processing power and memory capacity made it well-suited for running a Python-based GUI capable of real-time heat map

rendering, display updates, and user interaction. In the final implementation, Python handled not just post-processing but the primary visualization pipeline, receiving data from the ESP32 and generating the interactive heat map on the 5-inch Hosyond capacitive touchscreen in real time.

### 3.8.2.2 Matlab

Matlab was a powerful tool for mathematical modeling, data visualization, and signal processing. It offered Simulink and embedded code generation tools, but several barriers existed when considering using Matlab for real-time embedded execution. While Matlab could generate C code using Embedded Coder, this process was often complex and not optimized for real-time performance. Also, Matlab was not designed for embedded systems where low-level hardware access and deterministic timing were required. However, Matlab could still be useful for offline simulation and algorithm development, particularly in testing heat map processing techniques before implementing them in C on the MCU. Despite the advantages Matlab offered, it was not an entirely viable option.

### 3.8.2.3 Open Source and Proprietary Graphics Libraries

Since Python and Matlab were initially unsuitable for real-time heat map rendering on the STM32 platform we first considered, we explored alternative solutions. One option we evaluated was LVGL (Light and Versatile Graphics Library), an open-source graphics library optimized for embedded systems that supported touchscreens and real-time visualization. We also considered the STM32 graphics framework TouchGFX, which was specifically designed for use with STM32 MCUs. TouchGFX offered efficient rendering capabilities and integrated directly with STM32CubeIDE, though it limited some visualization flexibility due to its modular, template-based approach rather than full-custom rendering.

As the project evolved and we transitioned away from an STM32-based architecture, these embedded graphics frameworks were ultimately set aside. In the final design, we implemented a Raspberry Pi 4 Model B paired with a 5-inch 800×480 Hosyond capacitive touchscreen. The Pi's processing power and memory allowed us to run a Python-based GUI that handled real-time heat map rendering and user interaction without the constraints of MCU-specific libraries. This approach gave us the flexibility and visualization quality we needed while simplifying development compared to integrating LVGL or TouchGFX on a resource-constrained microcontroller.

### 3.8.2.4 Comparison and Selection

Table 3.8.2.4: Programming Language Comparison

| Feature | Python | Matlab | LVGL | TouchGFX |
|---|---|---|---|---|
| Performance | Fair (interpreted, high overhead) | Moderate (code generation possible but inefficient) | High (optimized for embedded systems) | High (designed for STM32) |
| Memory Usage | High (requires significant RAM) | High (not optimized for MCUs) | Low (optimized for embedded) | Low (efficient STM32 integration) |
| Real-time Suitability | Moderate (interpreted, non-deterministic) | Poor (not designed for real-time execution) | Good (designed for embedded graphics) | Excellent (real-time STM32 graphics support) |
| Ease of Development | High (simple syntax, many tools) | Moderate (powerful but complex) | Moderate (requires embedded knowledge) | Moderate (STM32-specific learning curve) |
| Hardware Interaction | Limited (not MCU-friendly) | Limited (mainly for simulation) | Direct framebuffer access | Optimized for STM32 hardware |
| Best Use Case | Post-processing, simple real-time | Algorithm simulation, testing | Lightweight GUI on embedded devices | Primary GUI development for STM32 |

Python and Matlab were unsuitable for real-time heat map rendering on our embedded platform because of their high resource requirements and lack of native embedded support. While they still proved useful for offline analysis and simulation during development, they could not be directly used on the STM32 hardware we initially planned to deploy. At that stage, we identified LVGL and TouchGFX as the most viable options for embedded graphics. LVGL was originally our preferred choice due to its optimized rendering performance, direct hardware compatibility, and seamless integration with STM32CubeIDE. TouchGFX remained an alternative when a lighter or more template-driven approach was acceptable, but LVGL was expected to serve as the primary graphics framework for the project.

As our architecture evolved, we ultimately transitioned away from the STM32-based design in favor of a Raspberry Pi 4 Model B paired with a 5-inch 800×480 Hosyond capacitive touchscreen. This change eliminated the need for MCU-specific frameworks like LVGL or TouchGFX. Instead, we implemented a Python-based GUI on the Raspberry Pi, leveraging its higher processing power and memory to achieve real-time heat map rendering and interactive visualization without the constraints of an embedded graphics library. This approach provided greater flexibility and a more straightforward development process for our final system.

### 3.8.3 GitHub

GitHub was used as the version control system for our software development. The repository was structured to include firmware, visualization, and documentation. The firmware directory contained the C and/or C++ source code responsible for MCU operation. The visualization section housed code related to heat map rendering, and documentation included reports, design documents, and notes. The repository also contained binary files for precompiled firmware and relevant data. Using GitHub ensured that all team members could collaborate effectively, track changes, and resolve conflicts systematically. Version tracking allowed us to maintain a history of modifications and revert to previous versions if necessary.

## 3.9 Power Supply Unit

The power supply unit (PSU) of the system is comprised of the battery, battery connection, power regulators, and battery management IC. Every component in the system is dependent on the PSU, so a table is included below that outlines each component's supply voltage range and maximum current drawn.

Table 3.9: Circuit Components

| Component | Supply Voltage Range (V) | Maximum Current Drawn (mA) |
|---|---|---|
| Hall-effect Sensors | 2.5 - 38 | 230.4 |
| Multiplexers | 2.5 - 6 | 32 |
| Analog to Digital | 2.50 - 5.5 | 3 |

| | | |
|---|---|---|
| Converter | | |
| MCU | 3.0 - 3.6 | 240 |
| Raspberry Pi 4 w/ LCD | 5 | 3000 |

## 3.9.1 Batteries

For H.A.L.O., we were focused on maneuverability, weight, and size as some of the most important factors when it came to deciding on how our system would be powered. Since our system is stand alone, it can't rely on a wired connection to a wall outlet. There needed to be batteries that powered the system. To ensure the convenience of our system, high energy density was the chief priority. Battery life and cost were also factored into the decision making process.

### 3.9.1.1 Lead-acid Batteries

Lead-acid batteries were developed in the 19th century and have not undergone much change in design since. They are very dependable but aren't widely used in electronics because of their weight, size, and low energy densities. Lead-acid batteries have one of the lowest gravimetric energy densities as well as one of the lowest volumetric energy densities. Lead-acid batteries will often find utilization in systems that value dependability above all else or systems that have a large power storage requirement. Examples of such systems include emergency systems such as sirens and backup power. They're also utilized by renewable energy farms and automobiles. It's also worth noting that lead-acid batteries are quite affordable which lends to their continued use. [10]

While the affordability of lead-acid batteries were alluring, their low energy densities made them non-ideal for our project as convenience is of high importance.

### 3.9.1.2 Lithium Batteries

There are many types of lithium batteries. The ones that will be considered below are: lithium-polymer (Li-po), lithium-ion (Li-ion), and lithium-iron-phosphate (LFP). These batteries share lithium in their namesake because they all use lithium in some part in the composition of their cathode.

Each lithium battery type sports a high energy density resulting in less weight and size.

*Lithium-polymer*

Compositionally, the lithium-polymer battery mainly differs from its lithium battery counterparts in terms of its electrolyte. Where the typical lithium-ion and lithium-iron-phosphate battery utilizes an organic solvent, the lithium-polymer battery uses a polymer electrolyte from which it derives its name.

Lithium-polymers garner a wide variety of benefits such as the many varieties they are built in. A specific battery can be found for almost any need, and they are able to maintain lower temperatures under high rates of discharge; however, the biggest concern with lithium-polymers was safety. Fire and explosions are not uncommon due to mishandling. Overcharging and improper storage can more easily compromise the integrity of the lithium-polymer over other lithium battery types. In addition to these factors, the casing of the lithium-polymer is non-rigid. This leads to better affordability, but the batteries themselves become easier to damage. Their fragility combined with their volatility is a dangerous combination. The safety factor of lithium-polymer batteries was too prevalent for them to be used in this project.

*Li-ion*

The typical lithium-ion battery shares a lot of similarities with the lithium-polymer battery, so it's important to note the differences as an uneducated purchase can see a lithium-polymer battery acquired instead of the intended lithium-ion. This is furthered by the fact that a lithium-polymer is a type of lithium-ion battery, so many online listings will have lithium-ion in the title even if it is truly a lithium-polymer. Lithium-polymer batteries are usually flat while lithium-ion batteries are more often than not cylindrical. They both often have heat shrink over multiple cells, but the cells that make up a lithium-ion are rigid which is one of the factors that contributes to them having a better degree of safety. Lithium-ion batteries have a comparable gravimetric energy density to the lithium-polymer, but it has a better volumetric energy density.

*LFP*

The last battery type that will be examined in this section is the LFP battery. The LFP battery type has marked advantages over other lithium batteries such as: thermal regulation, safety, and cycle life. Like the Li-po, it is able to maintain lower temperatures under higher output. It's even safer than the lithium-ion as its casing is the most rigid of the lithium types discussed here and thus it is the most impervious to damage. These features however lead to a higher price point. [10]

If it weren't for the increased price, the LFP battery would have been the prime choice for this project; however, the improvements that the LFP provides over the lithium-ion weren't worth the price increase. In retrospect, this decision served our project well as we ended up purchasing multiple sets of Li-ion batteries, so the lower cost of Li-ion's was a continual benefit that allowed us to purchase the cells we needed for testing and demonstration. The lithium-ion is already plenty safe and provides adequate temperature regulation for the scope of this project. The cycle life of the LFP was an alluring quality, but our project was for demonstration purposes and wouldn't be used long enough to see this benefit pay dividends.

#### 3.9.1.3 Comparison & Selection

Since battery types are being compared here and not specific battery models. The categories and their rankings will have to deal in approximations and ranges.

Table 3.9.1.3.1: Battery Type Comparison

| | Lead-acid | Lithium-polymer | Lithium-ion | Lithium-iron-phosphate |
|---|---|---|---|---|
| Safety | great | terrible | good | great |
| Volumetric Energy Density (Wh/l) | 25-100 | 300-375 | 350-450 | 140-330 [11] |
| Gravimetric Energy Density (Wh/kg) | 10-40 | 125-190 | 125-190 | 90-120 [12] |
| Affordability | good | great | good | bad |
| Nominal Voltage (V) | 12 [13] | 3.7 | 3.7 [13] | 3.2 [14] |

Once the batter chemistry was decided. We moved on to choosing a specific battery.

Table 3.9.1.3.2: Battery Comparison

| | Option 1 | Option 2 | Option 3 |
|---|---|---|---|
| Minimum Order Quantity | 2 | 1 | 4 |
| Listed Price | $11.99 | $8.52 | $17.34 |
| Capacity (mAh) | 5000 | 2500 | 2600 |
| Nominal Voltage (V) | 3.7 | 3.7 | 3.7 |
| Brand | Generic | Qisuo | YDL |
| Connector | Button Top | XH2.54 | XH2.54 |

Option 1 has the best capacity by nearly double either other option. The price point lies in the middle of the other 2 options, but it comes with two cells, and a charger. Neither other option came with a charger, making the $11.99 price point extremely reasonable. The lack of a fixed connector allowed us to connect the cells to the system via a typical contact battery holder. This made taking the batteries in and out of the system for testing easy; however, the finished system doesn't require the user to change the cells as the user can charge the system without any disassembly.

## 3.9.2 Power Delivery

There are two instances in H.A.L.O. where power cables are used as they proved to be the best options. The Raspberry Pi Model 4 B receives power through its USB-C port which is supplied via a USB-A port on the 5 V regulator PCB, and the system itself is charged via a 12 V jack. We examined all these connections as well as Micro-USB before deciding on the usage of these cables in our system.

### 3.9.2.1 USB-A

The USB-A standard was created in 1996 [15], and you can continue to see it in common use today. It has undergone revisions and improvements, but those changes are centered around data speed. We were not concerned with the data

transfer of USB for this analysis as this cable was only used for power delivery. The USB-A has four wires, ground, $V_{cc}$, data+, and data-. [16] USB-A cables deliver 5 volts at a range of 0.5 - 2.4 A for a maximum wattage of 12 W [17].

### 3.9.2.2 Micro-USB

The Micro-USB was introduced in 2007 [15], 11 years after USB-A, and like its predecessor, it still sees widespread use to this day. Micro-USBs are rated for 5 V as well, but they're only rated for 2 A, so their maximum wattage is 10 W [18]. That's 2 W less than USB-A. It has the same four wires as USB-A, but there could be a potential fifth wire used for identification depending on if it's a Micro-A or Micro-B [16]. It does see the smallest size of the lot with dimensions of approximately 7.40 x 2.35 mm [19].

### 3.9.2.3 USB-C

USB-C is the newest addition of the bunch, seeing introduction in 2014. With its rotational symmetry, bidirectionality, and high data transfer speeds, it's poised to phase out the other USB types, a factor that shouldn't be disregarded as it affects the longevity of the product.

USB-C sees the same four-wire scheme as USB-A. Most pertinently, USB-C stands out in power delivery. It can support a voltage range of 5 - 20 V at an amperage range of 0.5 - 5 A [17]. This means the maximum wattage of the USB-C is 100 W. This dwarfs the maximum wattage of the USB-A and micro-USB standards. In terms of size, USB-C finds itself in between USB-A and micro-USB at approximately 8.34 x 2.65 mm [20].

### 3.9.2.4 12 V Power Supply

The 12 V Power Supply is rated for 12 V supplied at 2 A. There's a rectifier in the supply that converts the AC supply of the wall to a DC supply for the system. It is also the only power delivery option here that has complete rotational symmetry. It is also distinct in the fact that it contains no data wires. This cable is purely for power delivery.

### 3.9.2.5 Comparison & Selections

It is worth noting that the sizes compiled above and in the table below are only approximations, as the precise dimensions vary by manufacturer and the figures were compiled from specific examples around the internet. Also all power

delivery options see similar price ranges, so price will not be included in the table or analyses.

Table 3.9.2.5: USB Type Comparison

| | USB-A | Micro-USB | USB-C | 12 V Supply |
|---|---|---|---|---|
| Voltage (V) | 5 | 5 | 5 - 20 | 12 |
| Current (A) | 0.5 - 2.4 | 2 A | 0.5 - 5 | 2 |
| Maximum Wattage (W) | 12 | 10 | 100 | 24 |
| Dimensions mm | 12 x 4.5 [21] | 7.40 x 2.35 | 8.34 x 2.65 | 6.4 x 6.4 |

The 12 V power supply was selected to charge the system as we had a multitude of them on standby and the complete rotational symmetry made connecting and disconnecting the BMS PCB the easiest. A USB-A to USB-C cable was used to supply power from the 5 V regulator PCB to the Raspberry Pi respectively. This option proved to be the best of both worlds as we already had a cable for this purpose and this connection allowed us to use cheaper parts for the assembly of our PCB while still enjoying some of the benefits of the USB-C topology.

## 3.9.3 Power Regulation

Power regulation is essential to ensure a stable and consistent voltage for all components in the system. Even high-quality voltage sources may not maintain a constant output, and as a battery discharges, its voltage gradually drops below the nominal level. Without proper regulation, this decline can cause unstable operation or even system failure. A voltage regulation circuit maintains a steady output voltage to the system by compensating for variations in the supply voltage.

### 3.9.3.1 Linear Regulators

Linear regulators are the most common types of voltage regulators. Their simplicity makes them easy to implement and cheap to produce. Thus, their cost is very affordable. The linear regulator is purely a step-down mechanism while switching regulators allow for step up and step down voltage regulation.

With any voltage requirement, two options exist. We could've supplied a voltage higher than the requirement and stepped down or we could've supplied at a voltage lower than the requirement and stepped up.

For our system, the largest voltage rail was 5 V. We chose to go forth with the simpler option of supplying at a voltage above 5 V and stepping down as necessary throughout the circuit. This meant that linear and switching regulators were possible options for  our project.

*Standard*

When it comes to regulators, two of the most prevalent qualities to be concerned with are the dropout voltage and the ground pin current. The dropout voltage can be thought of as the excess voltage that is consumed by the regulator. If a regulator outputs at 3.3 V and has a dropout voltage of 1 V, the voltage source will need to be able to supply 4.3 V. H.A.L.O.'s smallest voltage difference was 2.4 V between the batteries and the 5 V regulator. With such a large gap, the dropout voltage wasn't a huge concern.

The standard linear regulator has the most dropout voltage at around 1.7 V to 2.5 V. However, its ground pin current is the lowest of the three types examined here at less than or equal to 10 mA. The numbers included here and for the two other types of regulators are for a typical regulator, hence why a general range is being used. Before a specific model was chosen, these ranges were examined to determine the proper regulator type for our project.

*Low Dropout (LDO)*

The LDO regulator varies from the standard regulator in that it only contains one PNP transistor in its feedback control loop. This results in less voltage being consumed by the regulator and results in the low dropout voltage the regulator receives its name from.

The LDO regulator has the lowest dropout voltage of the three types examined in this section, but as a tradeoff it has the most ground pin current. A higher ground pin current is less desirable as the ground pin current is analogous to wasted current.

*Quasi Low Dropout (Quasi LDO)*

The quasi LDO regulator is a modified version of the standard regulator. Its dropout voltage is higher than that of the LDO, but lower than the standard regulator that it is derived from. The ground pin current is comparable to the standard regulator, but the quasi LDO's ground pin current is larger. [22]

### 3.9.3.2 Switching Regulators

Linear regulators dissipate voltage through resistors. The excess electrical energy is converted to heat. This is why the linear regulator can only be used to step-down voltages. The switching regulator operates differently. Depending on the type of switching regulator, there's an amount of switches that cycle on and off to regulate the voltage. The excess power isn't dissipated through heat like in a linear regulator. It isn't delivered to the source in the first place. This results in better efficiency and a cooler system, but the switching mechanisms introduce more noise than linear regulators, and they're more expensive as well. [23]

The efficiency of switching regulators and linear regulators vary as the efficiency of a linear regulator is determined by $\frac{Vout}{Vs}$ and the efficiency for a buck regulator varies but is typically between 70-90% [24]. This means that the efficiency of linear regulators is directly proportional to the difference between $V_s$-$V_{out}$. Therefore, as the amount of voltage that needs to be stepped down increases, switching regulators become more attractive efficiency wise.

### 3.9.3.3 Comparisons & Selections

The dropout voltages and pin currents are compiled in the table below. As was the case in the text describing the options, the ranges look at a typical regulator from each type and are thus approximations.

Table 3.9.3.3: Voltage Regulator Type Comparison

| | Standard | Low Dropout | Quasi-Low Dropout | Switching |
|---|---|---|---|---|
| Dropout Voltage (V) | 1.7 - 2.5 | 0.1 - 0.7 | 0.9 - 1.5 | $\frac{V_{out}}{D}$ |
| Ground Pin Current (mA) | ≤ 10 | ≤ 20 - 40 | ≤ 10 | NA |

The efficiency of switching regulators with their reliability, many options, and endorsements by senior design advisors led us to choose them for inclusion in our project. The excess noise they introduced was mitigated by distance and ripple capacitors.

### 3.9.3.4 Analog Devices LTC3525 Switching Boost Converter

The LTC3525 from Analog Devices is a boost switching converter that provides a fixed 5V output, which makes it a strong candidate for power regulation within H.A.L.O. With an efficiency of up to 95%, this converter is optimized for applications where power loss must be minimized. This regulator is capable of delivering 5V from a 1-4.5V input, making it suitable for power loads such as single cell lithium-ion batteries.

A major advantage of the LTC3525 is its ultra-low quiescent current of 7μA, which significantly reduces power consumption and helps to maximize battery life. Also, it only requires three external components, which helps to simplify PCB design. Additionally, this converter features output disconnect and inrush current limiting. This feature ensures that there is no current flow during startup or shutdown, helping to prevent damage to other sensitive components.

As stated before, this converter can operate with input voltages ranging from 1V to 5V, which would work effectively with a 3.7V Lithium-Ion battery. Once power is supplied to the converter, it can maintain regulation even if the input voltage drops as low as 0.5V.

The LTC3525 is clearly an excellent choice for power regulation due to its high efficiency and low quiescent current. The only specification that could be considered a drawback is its size being significantly smaller than older LM26xx models. While this might make it more difficult to integrate into our final PCB, it would be a good option for a single-cell power source.

### 3.9.3.5 Texas Instruments LM2623MM Switching Boost Converter

The LM2623 from Texas Instruments is a general purpose boost DC-DC converter designed for efficient voltage conversion in battery powered systems. With an operating input voltage ranging from 0.8V to 14V, it is able to be used with many different power supplies. This converter has an efficiency of up to 90%, which is ideal for H.A.L.O. since we are looking to minimize power loss. It also uses Pulse Frequency Modulation (PFM) to effectively regulate output voltage while reducing output ripple. The LM2623 provides output voltages from 1.24V to 12V, covering a wide range suitable for many different applications

One of the major advantages of the LM2623 is its low start up voltage being 1.1V, meaning it can operate with minimal input. Its high switching frequency

also allows for the use of smaller passive components. This converter has a built in N-channel MOSFET with a low on-resistance, improving thermal performance and overall efficiency. While the LM2623 is not the most modern boost converter, its larger packaging makes it much easier to integrate onto a PCB, as its wider pin spacing simplifies routing and soldering. As long as we have the board space, this model may be a better option when compared to smaller modern regulators.

#### 3.9.3.6 Texas Instruments LM2596S Switching Buck Converter

The LM2596S from Texas Instruments is a step-down (buck) switching voltage regulator designed to efficiently convert higher input voltages to lower output voltages. With an input voltage range of up to 40V and the ability to provide output voltages of 1.2V to 37V, the LM2596S is suitable for a wide variety of applications, including the H.A.L.O. project. The device can deliver up to 3A of continuous output current, making it ideal for powering components that require a stable and reliable power source.

The LM2596S uses a fixed internal oscillator that operates at 150 kHz, allowing it to use smaller filter components compared to lower-frequency regulators. With an efficiency of up to 90%, the LM2596S helps minimize power loss and maximize battery life, a critical factor for portable applications like H.A.L.O. The regulator is available in fixed output versions of 3.3V, 5V, and 12V, with an adjustable version that can range from 1.2V to 37V.

A key benefit of the LM2596S is its simple design, requiring only four external components for operation. This simplicity reduces design complexity and simplifies PCB layout. The device also features line and load regulation, thermal shutdown, current-limit protection, and low quiescent current of around 80µA in standby mode, which ensures efficient operation even under low power conditions.

However, its larger size compared to newer, more compact converters may present challenges in highly constrained spaces. Either way, its robust design and ability to handle various input loads make it a solid option when space allows, especially in applications like H.A.L.O. that demand reliable voltage regulation for various subsystems.

#### 3.9.3.7 Texas Instruments LP2985 Linear 3.3V Regulator

The LP2985 from Texas Instruments is a low-noise, low-dropout (LDO) linear voltage regulator that provides a stable 3.3V output, making it a strong candidate for powering low-power components in H.A.L.O., such as the

microcontroller.The LP2985 can easily be powered with a 3.7V lithium-ion battery, as it can handle input voltages ranging from 2.5V to 16V. It offers a ±1% output voltage accuracy over load and temperature, which is essential for maintaining stable power delivery to sensitive subsystems.

LP2985 features a very low dropout voltage, typically around 40mV at a 150mA load. This ensures that the regulator maintains a consistent 3.3V output even when the input voltage is near the output, which is important for maximizing battery life in a system like H.A.L.O. The LP2985 also features low quiescent current and ultra-low shutdown current of just 1.12µA, which helps to extend operation time when in standby mode.

The regulator's low output noise (30µVRMS with a 10nF bypass capacitor) and high power supply rejection ratio (PSRR) of 70dB at 1kHz make it ideal for applications where noise-sensitive analog components, such as the Hall-Effect sensor array, are in use. This performance reduces the need for additional filtering, simplifying the design and enhancing reliability. Also, the LP2985 includes built-in protection features like current limiting and thermal shutdown, ensuring that the device remains safe under various operating conditions.

However, it is important to note that, like all LDO regulators, the LP2985 is less efficient than switching regulators when the input voltage is significantly higher than the output. For H.A.L.O., this may not be a concern as long as the device is used to power lower-power components where noise and size are primary considerations. With its compact SOT-23 package, the LP2985 can easily be fit into the H.A.L.O. design, making it a viable solution for specific low-power subsystems.

### 3.9.3.8 Texas Instruments LM2576 Switching Regulator

The LM2576 from Texas Instruments is a step-down switching voltage regulator designed to convert higher input voltages to a lower, controlled output. This regulator has an input voltage range of up to 40V and has an adjustable output model with voltages ranging from 1.2V to 37V, making it a great option for H.A.L.O. The IC can also deliver an output current of up to 3A, which is plenty for powering the ESP32 and Sensor Array PCBs.

The LM2576 uses a fixed internal oscillator that operates at 52 kHz. With an efficiency of up to 77%, the LM2576 helps to minimize power loss while also being simple to implement. Its simple design only requires a few external components, which helps to reduce the design complexity and simplifies PCB layout. This regulator also features line and load regulation, thermal shutdown,

current-limit protection, and low standby quiescent current. Although newer regulators may offer smaller footprints or higher efficiency, the LM2576's ease of use and available documentation make it a practical choice for H.A.L.O.'s 3.3V supply.

### 3.9.3.9 Texas Instruments LM2679 Switching Regulator

The LM2679 from Texas Instruments is another step-down switching voltage regulator that provides efficient voltage conversion with a higher switching frequency than the LM2576. This means that it can deliver a higher current output of 5A, which makes it a great option for powering components such as the Raspberry Pi and touchscreen LCD in H.A.L.O.

This regulator operates at 260 kHz, which allows it to use smaller inductors and capacitors than other low frequency options. Its efficiency of up to 90% helps to reduce heat and extend battery life, which are important factors for our portable system. The regulator also includes features like thermal shutdown, cycle-by-cycle current limiting, and under-voltage lockout, ensuring robust and reliable operation. While the LM2679 is an older and bulkier regulator, this actually makes it easier to integrate into our PCB due to its larger pin spacing and simplified layout requirements.

### 3.9.3.10 Final Component Selection

Table 3.9.3.8: Voltage Regulator Comparison

| Component | Input Voltage | Output Voltage | Size | Output Current |
|---|---|---|---|---|
| LTC3525 | 1V to 4.5V | Fixed 5V | Small | 0.5A |
| LM2623MM | 0.8V to 14V | 1.24V to 12V | Large | 2A |
| LM2596S | 4.5V to 40V | Fixed 3.3V | Large | 3A |
| LP2985 | 2.5V to 16V | Fixed 3.3V | Small | 150mA |
| LM2576-ADJ | 4V to 40V | Adjustable | Large | 3A |
| LM2679-ADJ | 8V to 40V | Adjustable | Large | 5A |

Our power system for H.A.L.O. has undergone many changes throughout senior design. At the start, we were designing around a single 3.7V lithium-ion cell, so we were looking at boost regulators to generate the necessary system voltages. Once we decided on using two cells in series, giving us 7.4V to 8.4V, buck

regulation became the obvious choice. Boost regulators no longer made sense when the input was already higher than our output.

We chose the LM2576-ADJ for the 3.3V supply since it can deliver up to 3A, which is more than enough for the ESP32 and sensor array. For the 5V line, we went with the LM2679-ADJ. It can output up to 5A and handles the Raspberry Pi and touchscreen without issue. Both of these regulators are older and come in larger, through-hole packages, which actually makes PCB integration easier. Their size gave us more flexibility with routing, and the physical footprint was helpful during soldering and debugging since we weren't limited by fine-pitch pads.

Other regulators considered were not suitable for the updated power requirements. The LTC3525 and LM2623MM are boost converters, incompatible with the higher battery voltage configuration. The LTC3525's maximum output current of 0.5A is insufficient for the system's needs. The LM2596S provides only fixed output voltages, limiting flexibility. The LP2985, while low noise and efficient for small loads, is limited to a maximum current of 150mA, restricting its use to low-power subsystems. Overall, the LM2576-ADJ and LM2679-ADJ offer the best combination of output current capability, efficiency, and ease of integration for the updated power system design.

## 3.9.4 Battery Management Chips

When working with rechargeable batteries such as Lithium-Ion, it is important to have an effective battery management system to ensure safe and efficient operation. Battery management chips, also known as power management integrated circuits, regulate the charge and discharge cycles of batteries, controlling the voltage and current to prevent overcharging or discharging. Many of these common chips offer features such as thermal management and overvoltage protection to increase safety and extend the lifespan of a battery.

For our application in H.A.L.O. where power reliability is a key factor, selecting the right battery management chip is crucial for creating a stable system. A well chosen power management integrated circuit will not only project the battery but also optimize charging for smooth operation.

### 3.9.4.1 Microchip Technology MCP73871

The MCP73871 from Microchip Technology is an integrated solution for both system load sharing and Li-Ion/Li-Polymer battery charge management. It is

designed to power a system and simultaneously charge a single-cell Li-Ion battery using either an AC-DC wall adapter or a USB port. One of the key features of the MCP73871 is its Voltage Proportional Current Control (VPCC), which ensures that the system load takes priority over the charge current to the battery, helping prevent overcharging.

This charger operates with a constant current/constant voltage (CC/CV) algorithm and offers selectable charge termination points, with voltage options of 4.10V, 4.20V, 4.35V, or 4.40V, all with a regulation tolerance of ±0.5%. The device also supports thermal regulation, adjusting charge current based on temperature conditions to protect the system and optimize the charge cycle time.

The MCP73871 can handle up to 1.8A of total input current, making it suitable for portable applications where power requirements can vary. Charge current control is resistor-programmable, ranging from 50 mA to 1A, providing flexibility for different battery configurations. Additionally, it includes essential safety features such as reverse discharge protection, undervoltage lockout (UVLO), and a safety timer with enable/disable control.

To keep users informed of the charging status, the MCP73871 provides several indicators, including a low battery status indicator (LBO), a power good status indicator (PG), and charge status/fault condition indicators. This makes it easy to monitor the charging process, and the device is designed to function within a wide temperature range of -40°C to +85°C, ensuring reliable operation in various environments.

Due to its small physical size and low number of external components, the MCP73871 is ideal for portable devices like GPS systems, PDAs, digital cameras, and Bluetooth headsets, where both system power and efficient battery charging are required.

### 3.9.4.2 Analog Devices LTC4054

The LTC4054 from Analog Devices is a complete constant-current constant-voltage linear charger specifically designed for single-cell Lithium-Ion batteries. Its compact ThinSOT package and low external component count make it a great choice for portable applications where space and simplicity are key. This charger is particularly well-suited for devices powered via USB ports, such as cellular phones, PDAs, MP3 players, and Bluetooth devices.

The LTC4054 is capable of charging Li-Ion batteries with a preset charge voltage of 4.2V, providing ±1% accuracy. It offers programmable charge current up to 800mA, which is set with a single external resistor, giving flexibility for various battery configurations. The charger operates with a constant-current/constant-voltage (CC/CV) charging algorithm, and it includes thermal regulation to prevent overheating during high-power operations or high ambient temperatures. This thermal regulation ensures that the charge rate is maximized without compromising the safety and longevity of the battery.

The charger also includes several useful features, such as automatic recharge and C/10 charge termination, which automatically ends the charge cycle when the current drops to 1/10th of the programmed charge value after the final float voltage is reached. Additionally, the LTC4054 includes a charge current monitor output for gas gauging, which provides real-time data on the battery's state of charge. It also has a status output pin that indicates the charge termination and the presence of an input voltage.

For low power consumption, the LTC4054 enters a low current state (less than 2μA) when the input supply is removed. It can also be placed in a shutdown mode that reduces the supply current to just 25μA, helping to conserve battery power when the charger is not in use.

This charger is ideal for compact, low-power devices that require efficient and reliable charging, such as charging docks, cradles, and Bluetooth applications.

### 3.9.4.3 Texas Instruments BQ24610RGER

The BQ24610RGER from Texas Instruments is a highly integrated, stand-alone battery charger controller designed for Li-Ion and Li-Polymer batteries. This device uses a synchronous buck converter to efficiently manage charging, supporting up to 6 cells in series with an operating input voltage range from 5V to 28V. With a charge current capability of up to 10A, it is suitable for a variety of applications, including mobile devices, industrial equipment, and ultramobile PCs.

The BQ24610RGER is known for its high-accuracy voltage and current regulation, with a charge voltage accuracy of ±0.5% and charge current accuracy of ±3%. It offers dynamic power management (DPM), which helps reduce battery charge current when the input power reaches its limit, preventing overloading of the AC adapter while simultaneously powering the system and charging the battery. This makes it an excellent choice for devices that require efficient power management and high-performance battery charging.

This device also includes several safety features to ensure the protection of both the battery and the system, including overvoltage protection, battery thermistor sensing for hot and cold charge suspends, battery detection, reverse input protection, and thermal shutdown. Additionally, it supports a programmable safety timer and includes charge overcurrent protection and battery short protection, which makes it a reliable choice for applications where safety is a priority.

The BQ24610RGER offers excellent status monitoring capabilities with three status output pins (STAT1, STAT2, and PG) to indicate the charging status and adapter presence. This allows for easy integration with a host processor or system for real-time monitoring of the charge cycle. It also features low quiescent current consumption, ensuring minimal drain on the battery during operation.

In addition, the BQ24610RGER is optimized for mobile and portable applications, such as netbooks, PDAs, handheld terminals, and medical equipment, providing a complete solution for Li-Ion/Li-Polymer charging with flexibility and safety.

#### 3.9.4.4 STMicroelectronics STC4054GR

The STC4054GR from STMicroelectronics is a standalone linear charger designed for single-cell Li-Ion batteries, offering constant-current/constant-voltage operation with thermal regulation to ensure optimal charging without risking overheating. This makes it an excellent choice for portable applications where efficient charging is required without the need for complex external components. The device charges Li-Ion batteries directly from a USB port and operates within USB power specifications, making it suitable for a wide range of consumer electronics.

The STC4054GR features programmable charge current of up to 800mA, which is set using a single external resistor. It also offers a fixed 4.2V charge voltage with 1% accuracy, ensuring precise charging for optimal battery life. The device automatically terminates the charge cycle once the current drops to 1/10th of the programmed value, and it supports automatic recharge once the battery voltage drops below a certain threshold.

Key features include an internal current regulation block that adjusts the charge current to prevent overheating during high-power or high-ambient temperature operation. The device also provides a charge current monitor output, which can

be used for gas gauging, allowing the system to monitor the battery's charge level accurately. Additionally, the low battery voltage detection triggers precharge mode, and the soft-start function limits inrush current during the charging process, ensuring a smooth start-up without damaging the device.

The STC4054GR is packaged in a compact TSOT23-5L package, making it ideal for space-constrained applications like cellular phones, PDAs, Bluetooth devices, and other battery-powered electronics. The device also includes a shutdown mode that reduces the supply current to just 25μA, further enhancing energy efficiency when not in use.

With its simple design, low external component count, and advanced features like thermal regulation and automatic recharge, the STC4054GR is an excellent choice for portable, energy-efficient applications.

### 3.9.4.5 Monolithic Power Systems MP2615

The MP2615 from Monolithic Power Systems is a switching charger that can be used in 1-cell or 2-cell lithium-ion battery applications. It can support charging currents up to 2A and allows the charge voltage to be set to 4.1V or 4.2V per cell. This was a key feature to consider, as H.A.L.O.'s updated power system utilizes two single cell lithium-ion batteries in series. The MP2615 also offers safety features such as overvoltage protection, thermal shutdown, and configurable limits for input and charge current.

A major advantage of this IC is its simple packaging, which makes it easier to integrate. While many other multi-cell chargers have complex pinouts and require intricate schematics, the MP2615 provides reliable charging with straightforward implementation. Its 16-lead QFN package is small enough for our system, but not so small that it cannot be integrated using the equipment available for senior design.

A major limitation of the MP2615 is its lack of power path management. This means it cannot simultaneously power the system while charging the battery. As a result, the system must be powered down in order for the battery to charge safely. While this tradeoff was acceptable for our application, it does restrict the device's ability to function as a true "plug and play" system during recharging, which may limit its use in scenarios requiring continuous uptime.

### 3.9.4.6 Final Comparison and Selection

In choosing the appropriate battery management chip for the H.A.L.O. project, several factors must be considered, including the power source, charging current, thermal management, and ease of integration with the system. The following table summarizes the key specifications of the four battery management chips evaluated:

Table 3.9.4.5: Battery Management Chip Comparison

| Component | Li-ion cells | Charge Current | Charge Voltage | Packaging | Additional Features |
|---|---|---|---|---|---|
| MCP73871 | 1 Cell | Up to 1.8A | 4.1V, 4.2V, 4.35V, 4.4V | 20-Lead QFN | System load sharing, automatic recharge |
| LTC4054 | 1 Cell | Up to 800mA | 4.2V | 5-Lead SOT-23 | Charge status output, soft-start |
| BQ24610RGER | 1-6 Cells | Up to 10A | Customizable based on the number of cells | 24-Lead VQFN | Dynamic power management, battery detection |
| STC4054GR | 1 Cell | Up to 800mA | 4.2V | TSOT23-5L | Charge current monitor, undervoltage lockout |
| MP2615 | 1-2 Cells | Up to 2A | 4.1V, 4.2V | 16-Lead QFN | Overcurrent/overvoltage protection, thermal protection |

Each of the battery management chips offers robust features for different applications. The MCP73871 provides high flexibility in terms of selectable charge voltage options (4.1V, 4.2V, 4.35V, and 4.4V), making it suitable for a wide variety of Li-Ion batteries. It also supports simultaneous system load sharing with charging, which could be advantageous for ensuring system

performance during battery charging. However, its larger 20-lead package may not be as space-efficient as some of the other options.

The LTC4054, with its compact SOT-23 package and low external component requirements, is another excellent choice for portable applications. Its charge current limit of 800mA makes it suitable for devices with lower power needs, and its thermal regulation and soft-start features provide protection in high-power or high-temperature environments.

For systems requiring higher charge currents, the BQ24610RGER stands out with a maximum charge current of 10A, making it ideal for devices with larger battery capacities or power demands. Its dynamic power management ensures that it can handle both the battery charge and system load efficiently without overloading the power source. However, its more complex features and larger package size may be overkill for simpler applications.

Lastly, the STC4054GR is a great option for applications that require a straightforward, efficient charging solution in a compact package. With a preset 4.2V charge voltage and programmable charge current up to 800mA, it is well-suited for battery-powered devices with moderate charging requirements. Its low quiescent current and shutdown mode also help to save power when the system is not in use.

After weighing all considerations, the MP2615 was selected for H.A.L.O. It provides an ideal balance of functionality, safety, and integration ease for our 2-cell (7.4V) lithium-ion battery configuration. With support for up to 2A of switching charge current, overvoltage and thermal protection, and selectable per-cell charge voltage, the MP2615 meets our power delivery needs without introducing unnecessary complexity. However, unlike the MCP73871, it does not include power path management, which requires the system to be powered down during charging. Despite this tradeoff, the MP2615 was ultimately the best fit for the H.A.L.O. project's requirements.

# 4: Standards & Design Constraints

# 4.1 Standards

## 4.1.1 IPC-A-610

IPC-A-610 is one of the most widely referenced standards for electronic assemblies. This of course includes PCBs. It's obvious now why this standard deserves to be discussed and looked into further in this report.

IPC-A-610 discusses component layout and orientation. The orientation of the components needs to be consistent with the documentation, and the layout of the components must ensure proper spacing. Proper spacing avoids interference between components and allows for proper testing to occur.

IPC-A-610 also includes requirements for the through-hole and surface-mount component types. For through-hole, the standard demands insurance of the condition of the component as well as proper protocol when attaching it to the assembly. The leads of the component can't be bent or damaged in any capacity. Looking for cracks in the leads of the component is essential in ensuring that the component complies with standard IPC-A-610. The leads of the component also need to be of the proper length to ensure that a sufficient and proper soldering job can be accomplished. To comply with IPC-A-610, the manufacturer needs to carefully examine the components before they are used, and they need to keep a diligent mind when securing the component.

IPC-A-610 also deals with surface-mount components. Just like with through-hole components, the condition of the surface-mount component needs to be evaluated before it can be used in the construction of the assembly. The standard once again requires cracks to be searched for. For the attachment of a surface-mount component, the manufacturer must ensure that the component is flat and securely pressed against the PCB. No tilting or misalignment is permitted under this standard. The standard also requires pressure be applied to the component throughout the entire soldering process. Only once the process is finalized, can the manufacturer release the pressure on the component. Whether it be through-hole or surface-mount, the component should be properly labelled so that identification is easy. This responsibility lies on the manufacturer of the component, but we as buyers and component selectors need to be aware of this when purchasing components so that we comply with IPC-A-610.
IPC-A-610 governs many facets of the soldering process as well. The shape of the solder is important. Excess solder resulting in a bubble is unacceptable. There should be a clear concave shape of the solder that goes from the borders

of the solder to its peak, and the leads of the component need to be trimmed to proper length once the soldering has been completed. The edges of the solder cannot touch the edges of other solder, pins, or leads as that would result in a short. Not only could that affect the functionality of the product, but it could affect the safety as well. Lastly, the condition of the solder itself must be inspected as well. The solder must be clean. No residue can be present. IPC-A-610 also provides rules for coating processes once the PCB is completed to ensure longevity of the product and protection of the board.

### 4.1.2 IPC-2221: Standards in Circuit Board Design

IPC-2221 as indicated by its title is about printed circuit board design. The actual construction is covered more in standards like the above IPC-A-610. IPC-2221 has a hierarchical structure with other relevant standards. It's important to discuss how they relate to each other to fully understand IPC-2221.

IPC-2221 covers the generic design of PCBs. There are four standards below it that each cover a specific type: IPC-2222 covers rigid circuit boards. This standard definitely applies to our project as we are most definitely going to be utilizing rigid circuit boards. IPC-2222 goes over spacing, material selection, board thickness, routing, etc. IPC-2223 covers flexible circuit boards. This one has possible bearing on our project as a flexible circuit board is a potential solution to deal with interference the battery may induce in the rest of the system. Like IPC-2222, IPC-2223 addresses material selection. It also covers minimum bend radius, filling of vias, and more. IPC-2224 deals with laminated multi-chip modules or MCM-L. A laminated multi-chip module uses laminated PCB as the substrate and has multiple chips on the PCB [25]. The last in the IPC 2220 series is IPC-2226. IPC-2226 covers high density interconnect PCBs, also known as HDI. These PCBs are characterized by their high wiring density. There are significantly more traces and connections than on your typical PCB. It is possible that our project will delve into this territory as our hall effect array will have 64 Hall Effect sensors alone, not counting any regulators, multiplexers, or other components. Although, our project will definitely dive into territory covered by IPC-2222 and possibly subject matter covered by IPC-2223 and IPC-2226, this paragraph was only included in this report to discuss the hierarchy of the 2220 Series so that IPC-2221's role as a standard could be better understand and the below discussion of IPC-2221 can be analyzed with its proper context in mind.

Two words that are seen often in IPC-2221 are clearance and creepage. Clearance is the distance between conductors measured through the air, and creepage is the distance between conductors measured along the surface of the

PCB. IPC-2221 has tables that provide minimum clearances based on a variety of voltage levels from 0 to 500 V. The maximum voltage that our system will see is 5 V, so only the 0 - 15 V range of these tables is relevant to H.A.L.O. The applicable data from these tables is compiled below.

Table 4.1.2: Board Clearances

| | Minimum Clearance for Bare Board | Minimum Clearance for Assembled Board |
|---|---|---|
| Internal Conductors (mm) | .05 | |
| External Conductors, uncoated (mm) | 0.1 | 0.13 |
| External Conductors, coated (mm) | 0.1 | 0.13 |

After seeing this table, it is worth defining the terms “bare board” and “assembled board.” A bare board is a PCB that does not have any electrical components or vias. It consists of the PCB material itself and traces [26]. The assembled board is the PCB, after all of the electrical components have been soldered on [27].

IPC-2221 provides the minimum creepage and clearance values, but also acknowledges that increasing distances can be challenging for designing PCBs where size is often a chief constraint of the project. So, it provides a few alternate solutions to achieve these values. One of the offered solutions is to physically mar the circuit board by carving specific patterns in between the components such as v-grooves or slots. Although this doesn’t increase the clearance between the conductors. It does increase the creepage.

Another route to achieve these values is by installing physical insulation on the board itself, and IPC-2221 provides guidance on selecting insulator material. Their recommended basis on making this decision is by evaluating different materials’ Comparative Tracking Index or CTI. The CTI is essentially a measurement of how much voltage it takes for an insulator to begin tracking which is the process of conducting pathways forming in the insulator [28].

Other solutions for achieving acceptable creepage and clearance values are implemented in the design portion of the PCB. If possible, components with

large voltage differences between them can be placed further away from one another or can even be placed on opposite sides of the board if the board type permits [29].

### 4.1.3 IPC-7351: Surface Mount Design & Land Pattern Standard

IPC-7351 is a critical standard that governs the design and placement of any surface mount components. This standard provides land pattern dimensions to make sure that components are places with proper spacing for soldering. Since our project integrates an 8x8 array of Hall-Effect sensors and other surface mount components, following IPC-7351 guidelines would help us optimize our PCB design for when we manufacture and assemble it.

IPC-7351 highlights the ned for correct pad sizes and tolerances so that one can avoid soldering issues such as misalignment. This is important for the design of H.A.L.O., where all sensors and peripheral components must be correctly integrated. Also, this standard provides guidelines on thermal relief patters, which help to maintain heat dissipation to preserve signal integrity and reduce thermal stress.

### 4.1.5 IEC 61000-4-2: Electromagnetic Compatibility (EMC)

Given that we are looking to measure magnetic fields in real time, it is critical that we ensure H.A.L.O. has electromagnetic compatibility. IEC 61000-4-2 outlines testing procedures and design strategies that protect systems from electrostatic discharge (ESD) events, which can damage sensitive components like Hall-Effect sensors, microcontrollers, and more.

To comply with this standard, our design for H.A.L.O.'s PCB should incorporate ESD protection mechanisms such as ground planes and shielding techniques. By having proper PCB layout with controlled impedance traces and isolation of high-speed signals, we will be able to mitigate the risk of signal degradation due to ESD events.

## 4.2 Constraints

Constraints have played a significant role in shaping our project, leading us to redefine its scope multiple times. At the start of the semester, our initial concept

was a photodiode-based radiation detector. However, after consulting with the Florida Space Institute (FSI) to assess its feasibility and applications, we secured sponsorship for radiation hardened components and a nuclear event detector on the condition that we shift our focus to something that the sponsors would be interested in. This was an emp protection system. Following our initial divide and conquer report and discussions with Dr. Chan and Dr. Weeks, the team ultimately decided to abandon the original idea and instead develop a system capable of imaging magnetic fields

## 4.2.1 Safety Concerns

### 4.2.1.1 Safety with the NED project

In our initial proposal, we planned to incorporate a nuclear event detector manufactured by Micross as the key component for detecting high doses of radiation. However, a critical limitation of this component was its minimum dose rate threshold of 50 krad per second, meaning that testing would need to be conducted in a high radiation environment that could pose significant health risks to individuals involved. Our team lacked direct experience in radiation testing, but we were working alongside Dr. Zhang and industry experts at Honeywell who had expertise in the field. With their guidance and proper safety procedures, we were confident in our safety.

When we initially presented our concept to Dr. Chan and Dr. Weeks, we had not yet developed a formal safety plan. As a result, they expressed strong concerns about the feasibility and risks associated with our proposed testing procedures. One of their primary objections was that all radiation testing would need to take place off UCF's campus, beyond their direct supervision. Given these concerns, Dr. Weeks suggested an alternative approach, using electrostatic discharge (ESD) as a means to trigger the device instead of radiation. This suggestion significantly alleviated their safety concerns and allowed us to explore a safer and more controlled method of testing.

### 4.2.1.2 Safety with the H.A.L.O. project

Although safety considerations were a major factor in prompting our team to reconsider the project's direction, they were not the sole reason for our decision to pivot. After thoroughly evaluating the technical complexities, logistical challenges, and long-term feasibility of our original design, the team decided to move away from radiation detection entirely. Instead, we chose to focus on measuring low-level magnetic fields—a direction that eliminated safety risks while still providing a technically challenging and meaningful project. The final

H.A.L.O. system was entirely centered around magnetic field imaging, ensuring that all testing was conducted safely and efficiently on the UCF campus without exposing team members or collaborators to hazardous conditions.

## 4.2.2 Feasibility Constraints

### 4.2.2.1 Feasibility with the NED project

Feasibility was a challenge that the team initially underestimated. When we were introduced to the idea of using the NED to trigger clamping in order to protect critical components from an EMP, we understood that it would be a difficult task. However, with the confidence expressed by our sponsors and the FSI, we believed it was an achievable goal. It wasn't until we spoke with Dr. Weeks that we fully understood that the project was beyond our abilities.

The NED operates with a response time in the nanosecond range, and for clamping to effectively mitigate the effects of an EMP, it would need to activate within a similarly short timeframe. The level of high speed circuitry required for this type of system was well beyond our team's current expertise. Designing and implementing such a system would have demanded knowledge of high speed electronics, specialized components, and testing environments that were simply outside our skill set.

### 4.2.2.2 Feasibility with the H.A.L.O. project

This lack of feasibility was the primary reason we ultimately decided to pivot to a magnetic field imager, a project that proved to be a much more attainable goal. With the imager, our initial target was to achieve a refresh rate of 60 frames per second. However, after conducting further research into the microcontroller we had selected at that time, we discovered that it might not have had enough processing power to sustain that rate. We explored potential solutions, including upgrading to a more powerful MCU or offloading some of the processing to a computer. After weighing the trade-offs, we concluded that the best course of action was to adjust our expectations and reduce the target frame rate. This compromise allowed us to maintain a functional and reliable design while staying within our technical limitations.

## 4.2.3 Time Constraints

### 4.2.3.1 Time with the NED project

Senior design places a strong emphasis on completing the project within the designated time frame. If we fail to do so, we are required to retake the course, we do not have the option to extend the project until it is finished. Because of this, it is crucial that our project remains feasible within the time allotted. For our team, this means completing our work within the spring and summer semesters. While some teams choose to spread their work across the spring and fall semesters, allowing them to utilize the summer as extra development time, our team's goal has always been to graduate by the end of summer.

At the start of our project, we devoted a significant amount of time to refining our idea. We scheduled meetings with professors, organizations like the FSI, and potential sponsors to explore different possibilities. When we were introduced to a project concept from Micross and Vorago, our team was eager to work with companies that could provide both technical expertise and critical hardware components. Our interactions with Vorago were smooth and efficient. They quickly answered our questions, confirmed their support, and even offered to provide a development kit along with a standalone MCU. However, our experience with Micross was much slower, and unfortunately, their hardware was a key part of the project. By the time we reached our D&C meeting, Micross had only just confirmed their interest in moving forward.

During our D&C meeting, Dr. Weeks and Dr. Chan raised concerns about working with these sponsors, many of which we had not previously considered. The components we were relying on were highly specialized, with limited publicly available information. This meant that if we encountered any technical issues, we would be entirely dependent on our sponsors for guidance and troubleshooting. Initially, both professors were completely opposed to us working with these sponsors, not only due to safety concerns regarding the NED but also because of the feasibility challenges associated with EMP clamping. However, once we determined that ESD could serve as a safer alternative to radiation, Dr. Weeks and Dr. Chan agreed to let us move forward, provided that we found a different application for the components beyond EMP protection.

After exploring alternative ways to integrate these parts into our project, we ultimately decided to abandon the idea of working with sponsors altogether. By this point, we had already spent a significant portion of the semester pitching ideas and searching for projects that sponsors might support. Although we had found a sponsor interested in a project similar to our original concept, moving forward with them would have required us to pitch a new idea and refine it until it aligned with their interests. Given that our initial D&C meeting had not been approved, we were already behind schedule, and we could not afford the

additional delays that would come with negotiating and troubleshooting a new sponsored project.

In the end, we chose to proceed without a sponsor to avoid being slowed down by dependence on an external company with its own priorities. While working with industry partners can provide valuable resources, the uncertainty surrounding sponsors' responsiveness and the potential time investment required to secure a viable project ultimately led us to take full control of our design. This decision has allowed us to move forward at a more efficient pace, ensuring that we remain on track to complete our project within the required timeframe.

#### 4.2.3.2 Time with the H.A.L.O. project

Scrapping our initial idea had significant consequences. Most notably, we had no completed pages for our final report and had to make rapid progress on our new project to stay on track. To ensure we met our deadlines, the team established a goal for each member to write a minimum of five pages per week while maintaining clear communication about which sections had been completed. This structured approach allowed us to make steady progress and avoid falling further behind.

When selecting components for our new project, we made a strategic decision to ignore lead times entirely. Most components had lead times of several weeks. On the time frame that we were working on, having to wait 7 weeks for a component vs having to wait 20 weeks didn't make that much of a difference. Both would likely result in project failure. To mitigate this risk, we chose to exclusively use widely available components with ample stock that could ship immediately. Additionally, for any relatively inexpensive components, we ordered spares to account for potential failures or mishaps during development.

Beyond that, our research also focused on identifying potential backup components. In the event that a critical part became unavailable, we ensured that we had alternative options that could be quickly sourced without significantly altering our design. By prioritizing availability and flexibility in our component selection process, we positioned ourselves to avoid unnecessary delays and maintain steady progress toward completing our project on time.

### 4.2.4 Cost Constraints

#### 4.2.4.1 Cost with the NED project

Cost was a significant concern when working with sponsors, particularly because the components they provided were extremely expensive, costing thousands of dollars each. If we accidentally damaged any of these parts, it was highly unlikely that we would have been able to replace them, adding another layer of risk to our project. This financial constraint played a role in our decision to move away from working with sponsored components and instead focus on a more cost effective, self managed design.

#### 4.2.4.2 Cost with the H.A.L.O. project

When selecting components for our revised project, we prioritized building a strong foundation with our microcontroller. The MCU was the most expensive single component, and its capabilities dictated many aspects of our overall design. Early on, we encountered a potential issue with the processing power of our selected MCU. We explored alternative options, including upgrading to a more powerful microcontroller that could comfortably handle our initial project requirements. However, after evaluating the trade offs, we determined that the increased cost of a high performance MCU wasn't justified for our specific needs. Instead, we made a strategic decision to add a raspberry Pi to aid with the graphics rendering to take some of the burden off our MCU.

Cost also played a major role in the selection of our sensors, as we needed to purchase a large quantity of them. Since our design relied on multiple sensors, we had to strike a balance between performance and affordability. Choosing an overly expensive sensor would have dramatically increased our budget, while opting for a cheaper, lower-quality sensor could have compromised the accuracy of our system. Additionally, we anticipated that some sensors might fail or get damaged during development, so we needed to account for the cost of ordering spares. Ultimately, our approach was to find a sensor that met our performance requirements while remaining cost-effective, ensuring that we could afford the necessary backups without exceeding our budget.

### 4.2.5 Layout Constraints

#### 4.2.5.1 Layout with the H.A.L.O. project

Since one of our primary design goals was to make the device as compact as possible, we focused on optimizing the layout by minimizing the number of components and ensuring that each component was utilized as efficiently as possible. Initially, we planned to use a 10×10 sensor array, mainly because the idea of having 100 sensors sounded appealing. However, once we began researching the implementation, we realized that using a power-of-two

arrangement would be more practical, especially since we planned to use multiplexers (MUXs) to manage the ADC channels on our MCU.

By structuring the array as an 8×8 grid with 64 sensors, we optimized our use of MUXs and simplified the routing of signals. This arrangement allowed us to dedicate one MUX to each row or pair of rows, improving efficiency. When selecting our MUX configuration, we weighed the trade-offs between speed and space. Using eight 8:1 MUXs would have provided faster switching but would have also required more board space. Instead, we opted for four 16:1 MUXs, which, while slightly slower, reduced the physical footprint of our circuit.

Another critical consideration in our layout design was the potential for interference between components and the sensors. Every component in our system generated its own magnetic field, which could have interfered with the readings from our magnetic sensors. To mitigate this issue, we carefully planned the placement of components, ensuring that any elements generating significant electromagnetic interference were positioned away from the sensor array. By balancing efficiency, compactness, and signal integrity, we created a design that was both practical and high-performing within our constraints.

## 4.2.6 Available Resources

It is important to keep in mind what is available to us for the duration of this project, both in terms of available equipment and available guidance. By default, we have the senior design lab and the instruction of Dr. Chan and Dr. Weeks. Additional equipment and guidance can be sourced through additional effort on our part, but that is the starting point.

The senior design lab comes equipped with Tektronix oscilloscopes, Tektronix dual arbitrary function generators, Tektronix DMM 4050 digital multimeters, Keithley 2230-30-1 triple-channel power supplies, Dell Precision 3420 computers, an SMD rework station, soldering and desoldering stations, a digital microscope inspection station, and many additional instruments not listed on the website [30]. If processes that require equipment not contained on this list, we must be prepared to acquire it ourselves.

### 4.2.6.1 NED

The NED project highlighted what was available to us as our initial scope was way beyond what we had access to; however, we were able to acquire access to additional facilities and equipment through the aid of sponsorships and connections: including access to the FSI laboratory and equipment from Micross

and Vorago. Gaining access to these resources wasn't the end of the dilemma as it shined a spotlight on the non-tangible resources we would need: guidance.

Dr. Weeks is dedicated to helping students succeed in senior design through hands-on help as well as providing the knowledge that he possesses. He made aware of these benefits by telling us we would not have access to them if we were to conduct most of our work in outside laboratories or with niche, specialized equipment.

We were taking a gamble on how much help the companies would be able to provide us and how much time they would be able to devote. This point was especially concerning since their proprietary equipment would be an integral part of our project and we would therefore be dependent on them for documentation and professional guidance. We decided that the dedicated guidance from an experienced professional like Dr. Weeks was too valuable a resource to neglect, and thus, we decided to proceed with an entirely different project.

#### 4.2.6.2 H.A.L.O.

There are two aspects to the resources at our disposal, the equipment and the guidance. By choosing H.A.L.O., we chose to forego equipment for guidance, and thus the constraints of the senior design lab are again prevalent. We corresponded with Dr. Weeks in the beginning of the project about some of these limitations that will shape our project. He informed us that UCF does not have the resources to solder ball grid array (BGA) parts. This constraint and constraints like it have shaped our selection of components.

# 5: ChatGPT vs. Similar Platforms

Artificial Intelligence (AI) has become increasingly popular in recent years, witha growing presence in educational and industrial fields. From natural language processing to machine learning, artificial intelligence tools have proven to be valuable assets for automation of tasks and solving complex problems. Among the tools available to the public, ChatGPT has grown in popularity due to its advanced language generation and user-friendly interface. Within this chapter, we will observe the benefits and drawbacks of ChatGPT in comparison to other AI platforms. Through this comparison, we aim to asses the effectiveness of artificial intelligence in applications like our senior design project, H.A.L.O.

The use of natural language processing (NLP) is what creates the overall functionality of ChatGPT, combining computer science and AI to help machines

understand the human language. NLP techniques enable tool like ChatGPT to comprehend user prompts and generate responses. Through the utilization of datasets, chatGPT is able to analyze text, recognize context, and provide coherent answers. This process is crucial for complex tasks. With NLP, ChatGPT can process and analyze entire documents or websites, offering insights based on the context the user provides. For our project, this feature is key to helping with research, documentation, and technical problem-solving. It allows the team to make the most of AI for smoother communication and more efficient task management as we develop H.A.L.O.

## 5.1 ChatGPT

ChatGPT is the juggernaut of the AI landscape. To measure a product's adoption, there exists a statistic that tracks how much time the product took to acquire 100 million users. ChatGPT snatched the record and claimed the title of the fastest growing consumer product after acquiring 100 million users in only 2 months. TikTok took 9. Instagram took 2.5 years [31]. Since then, it has been the name that people think of when AI is said. Being many people's first foray into the modern AI world, it has garnered a solid reputation amongst its contemporaries. This isn't without basis as its dedicated teams and years of development give it a leg up against its competition.

This reliability is one of ChatGPT's biggest strengths. It is solid across the various purposes you might need AI for, but that's also one of its downsides. ChatGPT is a jack of all trades and a master of none. It isn't weak in any areas, but it isn't the strongest either. It has strong reasoning capabilities, but it's outperformed by AIs like Grok and DeepSeek.

Being a jack of all trades comes with its own set of benefits as you can use ChatGPT for nearly any reason you would seek out an AI in the first place. It doesn't neglect the creative types as it offers image generation and has decent proficiency in long-text responses. On top of this, it has low AI detectability. In other words, its text output is less likely to be immediately identified as the product of an AI tool [32]. All of these are significant advantages of ChatGPT, but for our project, the creative side of ChatGPT is superfluous.

Cost is one of if not the most important factor in choosing products. ChatGPT does offer a free version, but it also offers a Plus and Pro plan at $20 a month and $200 a month respectively. The free version includes but is not limited to GPT-4o mini, (ChatGPT's cost optimized model), limited access to GTP-4o and o3-mini (flagship and reasoning models respectively), and limited access to file uploads, data analysis, image generations, and ChatGPT's voice mode.

The Plus plan lessens the limits imposed by the Free version, provides access to Sora as well as more reasoning models, and allows users to use experimental features and models. The Pro plan effectively allows access to all substantial models ChatGPT offers and pro mode, which uses more computational power for better answers [33].

## 5.2 DeepSeek

DeepSeek is one of the newest AI models, being released in early 2025, and quickly gained popularity in contention with Chat GPT. With all of ChatGPT's prominence and reliability, DeepSeek recently overtook ChatGPT in the Apple app store, showing that ChatGPT's place on the top isn't as certain as some would believe; however, it has an air of controversy due to its country of origin: China.

DeepSeek's ties to China has led to concerns not only about security but censorship as well. Since DeepSeek is located within China, the company has to comply with the CCP's censorship policies. This leads to certain prompts around forbidden topics in China yielding lackluster results. DeepSeek also doesn't have the language capabilities of models like ChatGPT or Google's Gemini. Less globally popular languages are prone to gramatical and translational errors.

Then there's the host of problems that come with the data storage being located in China. TikTok's current plight is indicative of how these concerns could prove problematic for DeepSeek. At best, international audiences may be reserved about fully placing their trust in DeepSeek, leading to lower use. At worst, governments may outright ban the platform in their respective nation. TikTok's place in the United States is still uncertain, a fact that does not bode positively for DeepSeek's longevity in foreign markets. Anecdotally, I wasn't able to access DeepSeek on UCF's internet due to "connection privacy" concerns onGoogle Chrome. Only once I was on my home network was I able to successfully access the website.

Even with these setbacks, there are multiple benefits that DeepSeek provides leading to its surpassing of ChatGPT on the Apple app store. The platform is the most open-source out of all its competitors. It has made its DeepSeek-R1 model open-source, an unprecedented move in the AI landscape that could force other companies to do the same. Grok has already hinted at allowing open-source access to older models.

Once again, the importance of cost cannot be overstated in consumers' decision making process. DeepSeek uses cheaper GPUs than its competitors, allowing for the service to be offered at a cheaper price point than Grok or ChatGPT's higher tiers. Yet, it doesn't follow the typical subscription based model. It instead uses "tokens" relegated on the basis of quantity inputted and outputted from the model. For reference, one English character equates to 0.3 token while one Chinese character equates to 0.6 tokens.

Like ChatGPT, DeepSeek offers two pricing plans, but its only free option is in the form of a demo. DeepSeek Chat is its more affordable model and DeepSeek Reasoner is the premier tier. Chat is priced at 7 cents per 1 million token inputs with cache hit, 27 cents for 1 million token input with cache miss, and $1.10 for 1 million token output. Cache hit occurs when the answer has already been compiled by DeepSeek and therefore less computing power has to be expended on the company's part to provide the user with their answer. Reasoner is priced at 14 cents for 1 million token input with cache hit, 55 cents for 1 million token input with cache miss, and $2.19 for 1 million token output [34]. The two tiers are mainly set apart by Reasoner's titular reasoning abilities.

DeepSeek's increased affordability has not only allowed it to surpass ChatGPT on the Apple store, but could see its widespread use in foreign markets, especially regions like Europe where the cost barrier to AI is more significant than in the United States. It's unclear whether its setbacks will outweigh its benefits as its future is uncertain. It's important to remember that it hasn't been around for long enough to truly cement itself in the industry. Unlike ChatGPT, which has undergone years of development in the public eye, DeepSeek has yet to fully attain a dependable reputation. The questionable longevity of DeepSeek in foreign markets may affect our adoption of it for our senior design project, and it may not. It's simply too early to tell if security concerns will impact our access to it during our senior design process.

## 5.3 Case Studies

Our report will feature three case studies that examine the three regions we have designated for our report: the power supply unit, the Hall-effect array, and the software.

### 5.3.1 Case Study 1

**Question**: I'm designing a circuit that consists of these components: 64 Hall-effect sensors, 4 multiplexers, 4 ADCs, 1 MCU, and 1 dev board. The

supply voltage ranges are as follows: 2.5 - 38 V for a Hall-effect sensor, 2.5 -6 V for a MUX, 2.50 - 5.5 V for an ADC, 3.0 - 3.6 V for the MCU, and 7 - 12 V for the dev board. The maximum currents drawn are as follows: 3.6 mA for a Hall-effect sensor, 25 mA for a MUX, 5.2 mA for an ADC, 500 mA for the MCU, and 800 mA for the dev board.

How should the circuit be laid out? Should components be placed in parallel, in series, or a combination of the two? Also, What voltage source should be used and how should it be regulated?

**Output:** To design a functional and efficient circuit incorporating 64 Hall-effect sensors, 4 multiplexers (MUXs), 4 analog-to-digital converters (ADCs), 1 microcontroller unit (MCU), and 1 development board, a parallel configuration is essential for nearly all components. Components such as sensors, MUXs, ADCs, and the MCU require independent, consistent voltage levels and must be connected in parallel to ensure each receives the correct supply voltage without voltage drops due to serial resistances. The sensors (64 total), each drawing 3.6 mA, should be connected in parallel and powered through a shared regulated line, as their combined current (230.4 mA) remains manageable. Similarly, the MUXs and ADCs should each have their own regulated voltage rails appropriate to their operating ranges (2.5–6 V for MUXs and 2.5–5.5 V for ADCs), supplied in parallel to prevent interference. The MCU, requiring 3.0–3.6 V and up to 500 mA, should be powered independently with a high-efficiency low-dropout (LDO) regulator or a buck converter capable of providing clean, stable power. The dev board, needing 7–12 V and up to 800 mA, should be supplied directly from the main voltage source if possible.

Given the various voltage requirements, the best approach is to use a single higher-voltage source—preferably a 12 V DC regulated power supply, which covers the upper threshold of the dev board. From this 12 V rail, step-down voltage regulators (buck converters or LDOs) should be used to derive the required voltages for the other components. A 5 V regulator can power the MUXs and ADCs, and a separate 3.3 V regulator can be used for the MCU. The Hall-effect sensors, which accept a broad range of supply voltages (2.5–38 V), can either be powered directly from the 12 V source or through the same 5 V rail used for the MUXs and ADCs for consistency and lower power dissipation. Voltage regulation must ensure each rail can supply at least the sum of maximum currents drawn by all connected components, with added headroom for stability and safety. This configuration ensures reliable performance, modular voltage control, and scalability for potential system expansions.

**Analysis**: This case study is geared towards the circuit layout and power supply unit design of this project.

This output from ChatGPT demonstrates significant promise and indicates benefits to our learning process without potential harm. To start off, ChatGPT summarizes the user's input in an easy to understand manner. For the purposes of this report and its formatting, ChatGPT was instructed to structure its outputs in paragraph form. Without this restriction, ChatGPT includes tables, bullet points, and other formatting nuances that help with the readability and intuitiveness of its response. Either way, the summary generated by ChatGPT allows the user to verify that ChatGPT not only received the correct information but interpreted it correctly as well.

The undeniable advantage of ChatGPT's answer to a prompt like this is its specificity. ChatGPT is able to provide advice for the exact circuit that we are designing. Looking for advice on this circuit through the traditional route of a search engine would provide similar circuits designed by other individuals, but we would have to apply the advice given about their similar circuit to our specific circuit. ChatGPT being able to provide insight directly to our circuit is the strongest benefit of using it or other services like it.

The content of the output itself is also respectable. Even if the conclusions that ChatGPT reached were incorrect, it provided its reasoning so that you can dissect its conclusions yourself and decide if they're accurate or not. Understandably, the response doesn't delve into all possibilities. ChatGPT suggests a step-down topology and doesn't examine a step-up topology. This is consistent with its response and reasoning used, but the user must keep in mind other options are available outside of what ChatGPT will end up suggesting.

ChatGPT provides its reasoning process so that the user can examine it for themselves and arrive at different decisions or come to the same conclusion as ChatGPT. ChatGPT doesn't just output the answer. Reading the response informs the user as well as provides a jumping off point for exploring other options. ChatGPT doesn't just take over and remove the responsibility of critical thinking from the user. These facts lead us to conclude that ChatGPT is a useful tool and not a hindrance to the learning process.

### 5.3.2 Case Study 2

**Question**: What type of ADC should be used for this project?

**Output**: For this project, a successive approximation register (SAR) ADC is the most suitable choice due to its balance of speed, resolution, and power efficiency. SAR ADCs are ideal for applications requiring moderate-to-high resolution and relatively fast conversion rates, such as reading analog signals

from multiple Hall-effect sensors through a multiplexing scheme. Given that the circuit includes 64 Hall-effect sensors distributed across 4 multiplexers and fed into 4 ADCs, a SAR ADC with at least 10- to 12-bit resolution would provide sufficient precision to accurately capture the analog signals without introducing significant quantization error. Additionally, SAR ADCs typically feature low power consumption, which aligns well with the power constraints of the system. They also offer short conversion times, which is critical when using multiplexers to scan multiple channels in rapid succession. Furthermore, many SAR ADCs come with integrated features such as sample-and-hold circuits and multiple input channels, enhancing their compatibility with multiplexed systems and simplifying the overall design. Therefore, SAR ADCs provide the optimal trade-off between performance, cost, and implementation complexity for this sensor-driven application.

**Analysis**: This case study focuses on the logic circuitry of our project. Specifically it examines the ADCs that will be used to convert the data from our Hall-effect sensors into a format that the MCU can understand.

Once again the specificity that ChatGPT provides can't be overlooked. Not only is ChatGPT able to provide ACD recommendations tailored to our specific circuit, but the parameters don't even have to be respecified. As long as the prompts are in the same conversation, ChatGPT will output relevant to the current prompt as well as past prompts in the conversation. This leads to ChatGPT gaining more information about the project throughout the conversation which results in better responses. This saves time during the research process.

ChatGPT's reasoning process is on display again, but this prompt is a proper example of other options not being explored. ChatGPT suggests SAR ADCs, but doesn't address any other type of ADC. The analysis of case study 1 applies here. The user needs to be aware that ChatGPT doesn't cover all options, and there are many possibilities that need to be looked into further; however, this can be performed by ChatGPT. With the context it possesses from the current conversation, a simple "What are other ADC options?" would provide the needed insight with minimal effort on the user's end to specify parameters and scan through irrelevant material.

### 5.3.2 Case Study 3

**Question**: What would be the best interface to connect an MCU and LCD display? Additionally, what parameters are taken into account? Include MCU specifications, driving, and the possibility of a touchscreen in your response.

**Output**: The best interface to connect a microcontroller unit (MCU) and an LCD display depends on the complexity and performance requirements of the display, but for most modern applications, a parallel RGB interface or Serial Peripheral Interface (SPI) is commonly used. If the LCD is a simple character or graphic display, SPI or $I^2C$ provides a compact, low-pin-count solution. However, for full-color graphical LCDs or touchscreen displays, a parallel RGB interface or Flexible Static Memory Controller (FSMC), if supported by the MCU, offers higher data throughput and better responsiveness. Key parameters to consider when choosing an interface include the MCU's available GPIO pins, clock speed, memory size, and whether it includes hardware support for specific protocols. For instance, if the MCU has limited GPIOs or slower processing capabilities, an SPI interface may be preferable, as it is easier to implement but may result in slower screen refresh rates. Conversely, if the MCU has a high-speed external bus interface or integrated graphics support, a parallel interface enables faster data transfer and smoother graphics.

Driving an LCD also depends on the voltage and current capabilities of the MCU, as well as the power requirements of the display. If the display requires higher current or backlight control, external transistor drivers or dedicated LCD controllers may be necessary. Additionally, if a touchscreen is to be integrated—particularly a capacitive or resistive touchscreen—a separate touchscreen controller may be required, communicating via $I^2C$ or SPI. Some LCD modules include built-in touch controllers that can simplify integration. It is also important to account for the resolution and refresh rate of the display, as higher resolutions demand more memory bandwidth and processing power from the MCU. For example, an MCU operating at 72 MHz with at least 128 KB of flash and 32 KB of RAM would be well-suited for driving a medium-resolution LCD with touch functionality using SPI or FSMC. In conclusion, selecting the best interface involves evaluating MCU hardware resources, display complexity, performance needs, and the inclusion of touchscreen capabilities.

**Analysis**: This final case study focuses on the last group designated by our group: software.

This is the first instance in all of the case studies of ChatGPT indicating the need for further information. To its credit, it clearly specifies what areas are vague and what specific information can be provided to improve the helpfulness of the response. For instance, "If the LCD is a simple character or graphic display" indicates to the user that they can specify what LCD they are using to remove unnecessary considerations from ChatGPT.

This case study is another instance where ChatGPT is fed a multi-part prompt consisting of multiple questions and requirements. The output doesn't neglect any part of the prompt and adequately addresses each section. The typical search engine approach would most likely require this prompt to be broken up into different sections to receive an adequate answer.

# 6: Hardware Design

The hardware design of the Hall-effect Array for Live Observation (H.A.L.O.) system centers around a modular architecture consisting of five custom PCBs that together support all major subsystems required for sensing, data acquisition, and power management. The Sensor Array PCB includes an 8×8 array of analog Hall-effect sensors, four 16:1 analog multiplexers, and one external 12-bit analog-to-digital converter (ADS7883SDBVR). An ESP32-WROOM microcontroller is used to manage sensor selection, data sampling, and communication. The power subsystem is distributed across three PCBs and includes two 3.7V lithium-ion batteries connected in series to supply 7.4V, a 12V DC input for charging, an MP2615 battery management IC for safe power source selection and charge control, and two buck converters that regulate system voltages down to 3.3V and 5V. The ESP32 interfaces with a Raspberry Pi 4, which is connected to a touchscreen LCD and Micro-SD card reader for visualizing and saving magnetic field data. Communication between the ESP32 and Raspberry Pi occurs over a UART serial connection, allowing for continuous data transfer without interrupting tasks.

The 64 analog Hall-effect sensors are the centerpiece of the system's sensing architecture, each producing a voltage proportional to the magnetic flux density perpendicular to its surface. These outputs are routed through the multiplexers to the ADC, which digitizes the analog voltages to be processed by the ESP32 microcontroller. The ESP32 cycles through each sensor channel, synchronizing multiplexer selection with ADC sampling to create a magnetic field map using all 64 sensors. This data is then transmitted to the Raspberry Pi, where it is reconstructed and rendered on the touchscreen as a 2D heat map. To ensure accurate sensor measurements, the PCB layouts were designed with signal integrity in mind. High frequency switching elements and high current traces are isolated from sensitive analog paths, and analog trace lengths are minimized where possible. These considerations help to reduce electromagnetic interference and noise within the system. Dividing responsibilities between the ESP32 and Raspberry Pi allows for modular software development, with the ESP32 handling data acquisition and the Raspberry Pi responsible for the display, storage, and user interface.

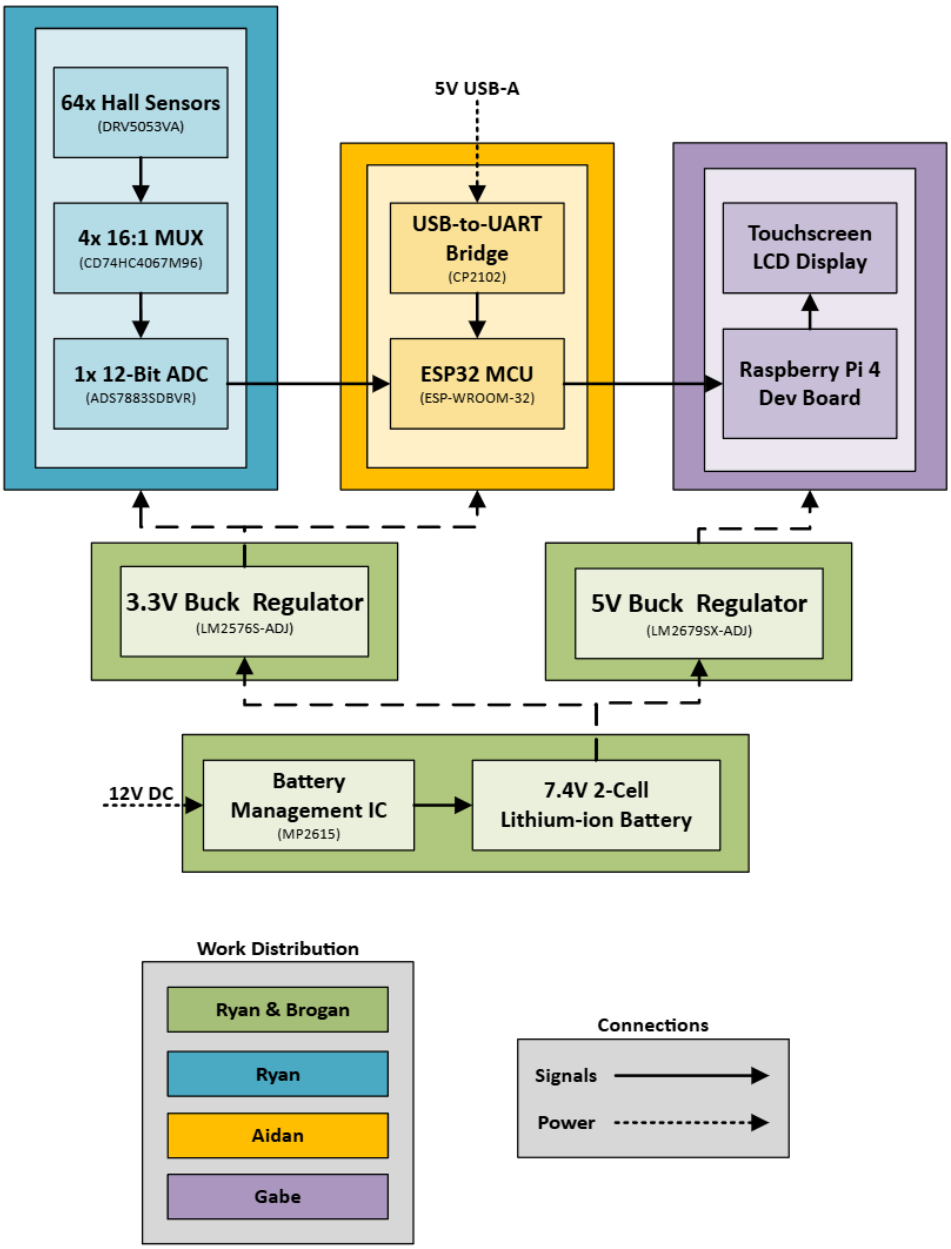



*Figure 6.0 Hardware Block Diagram*

# 6.1 Overall Schematics

## 6.1.1 Schematic Overview

As stated previously, H.A.L.O. has five separate PCBs that together form a modular system architecture. Each board was designed around a specific subsystem, allowing for easier testing and minimized noise between analog and

digital components. In this section, each schematic and corresponding PCB layout will be discussed in detail.

#### 6.1.1.1 Power Subsystem

The power subsystem consists of three separate PCBs: the Battery Management System, 3.3V Regulator, and 5V Regulator. These boards together generate and distribute the required voltages for the system to operate. The Battery Management System PCB uses two 3.7V lithium-ion batteries connected in series to provide a 7.4V supply. The main feature of this board is the MP2615 battery management IC, which enables charging using a 12V 2A DC input. This IC has integrated overcharge protection and temperature monitoring to ensure safe system operation and charging. A rocker switch is located on this PCB so that the system can be powered off during charging. Once the batteries have been fully charged, the 12V adapter must be unplugged before powering the system back on.

The schematic includes two status LEDs. The first LED, connected to the ACOK pin, indicates when the 12V input is present. The second, connected to CHGOK, turns on when charging begins and shuts off once the batteries reach full charge at 4.2V power cell. The board also features a 12V barrel jack for power input and test points to help verify charging and battery voltages. Careful PCB layout was essential when designing this board, as the MP2615 operates at high frequencies and is very sensitive to poor layout practices. To ensure proper operation, high-frequency paths were kept short, input/output capacitors were placed close to the IC, and grounding was carefully managed to minimize noise.

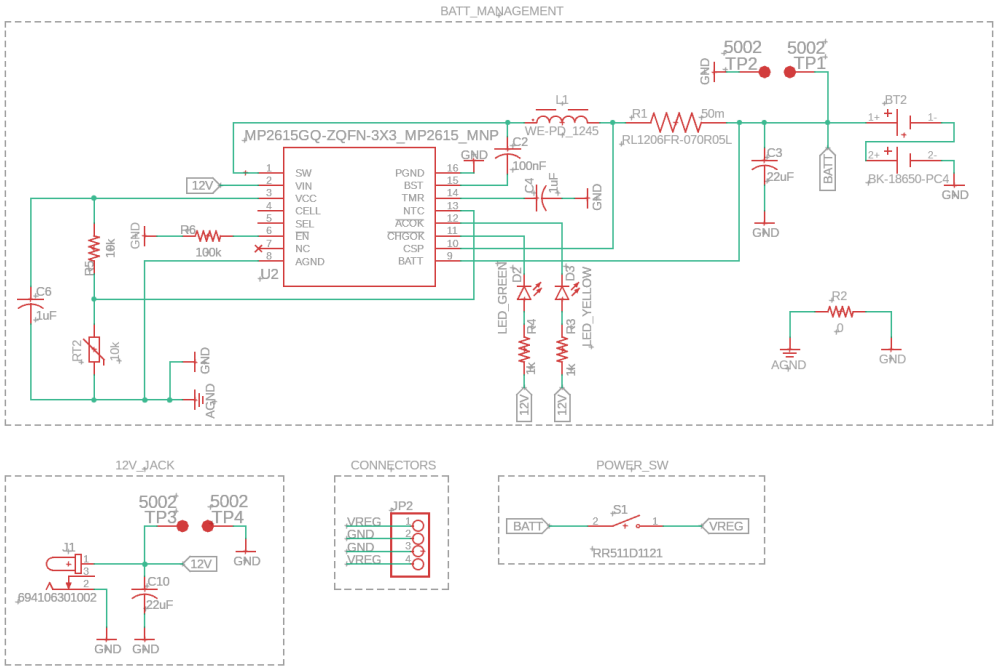



*Figure 6.1.1.0 Battery Management System Schematic*

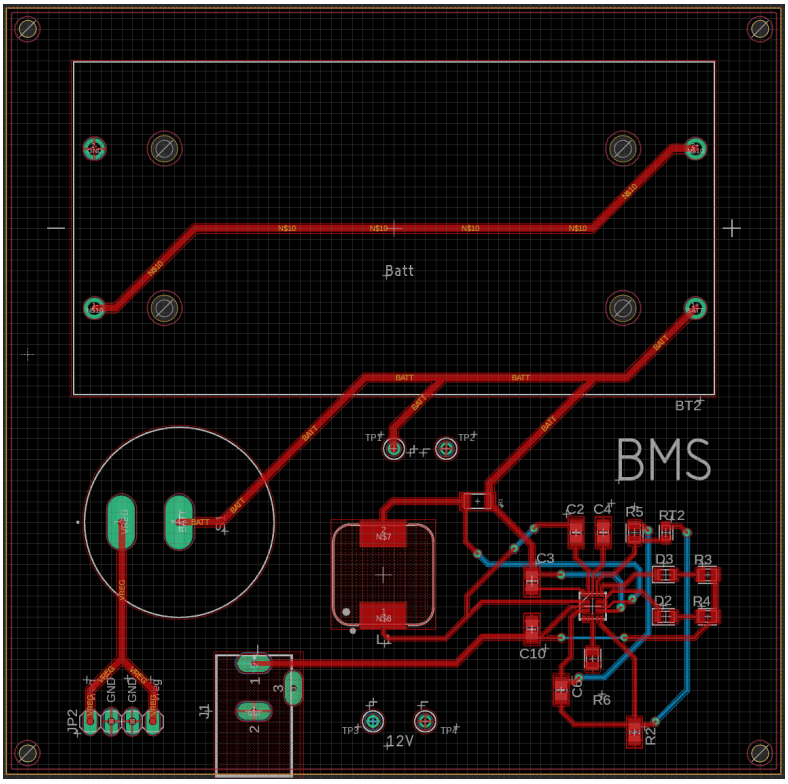



*Figure 6.1.1.1 Battery Management System PCB*

When the system is powered on, the battery output is fed into two voltage regulation PCBs. The 3.3V Regulator PCB steps down the voltage using a Texas Instruments LM2576-ADJ buck converter. This board powers the ESP32 and all components on the sensor array, including the Hall-effect sensors, multiplexers, and analog-to-digital converter. This design was created using TI's WEBENCH software to verify component selection and electrical behavior, such as overall efficiency. The PCB uses 32 mil traces for high current paths and components were spaced to allow for easier integration and troubleshooting. Test points are incorporated at both the input and output of the regulator schematic to verify proper voltage levels during development.

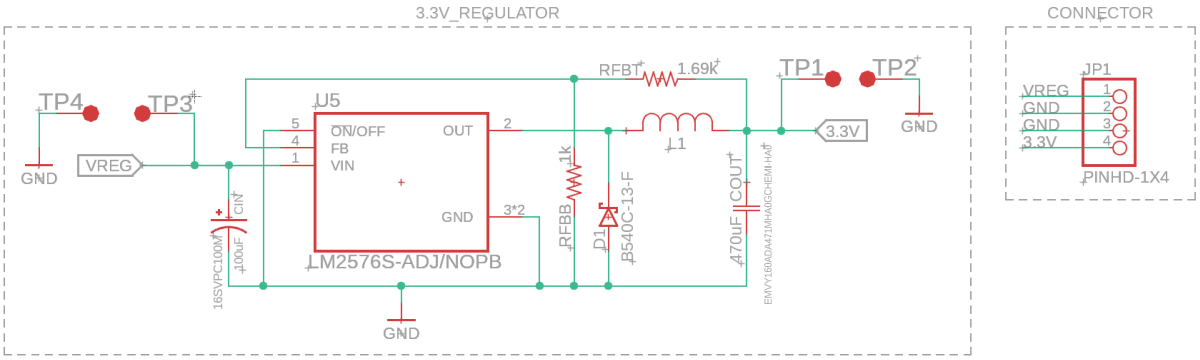



*Figure 6.1.1.2 3.3V Regulator Schematic*

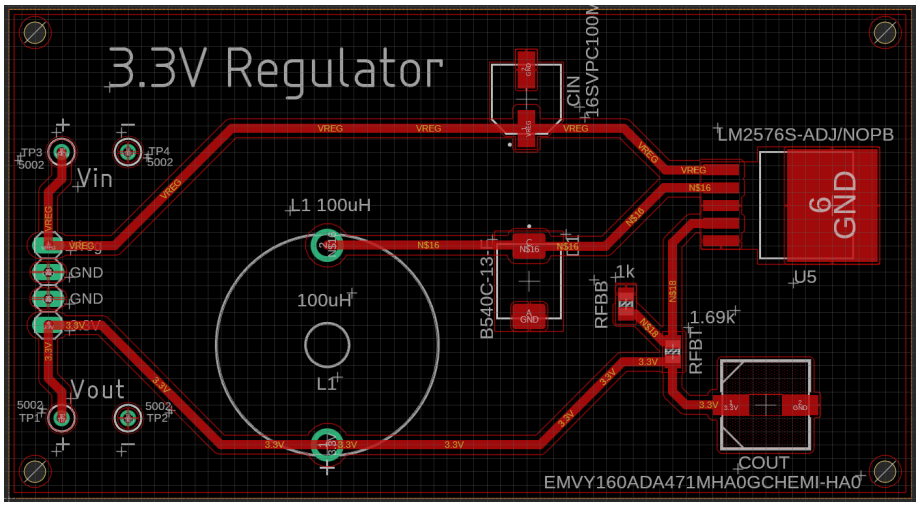



*Figure 6.1.1.3 3.3V Regulator PCB*

The 5V Regulator PCB uses the LM2679-ADJ to generate a 5V, 3A supply for the Raspberry Pi and its touchscreen LCD. Like the 3.3V regulator, this design was completed using TI's WEBENCH tool. The PCB layout follows best practices for

switching regulators, including careful placement of the inductor, diode, and capacitors to minimize electrical noise. It uses 32 mil traces for current flow and includes test points at the input and output to verify voltage levels. The output is routed to a USB-A port, which supplies power to the Raspberry Pi via a USB-A to USB-C cable. Both regulator boards are powered directly from the Battery Management PCB through header connections.

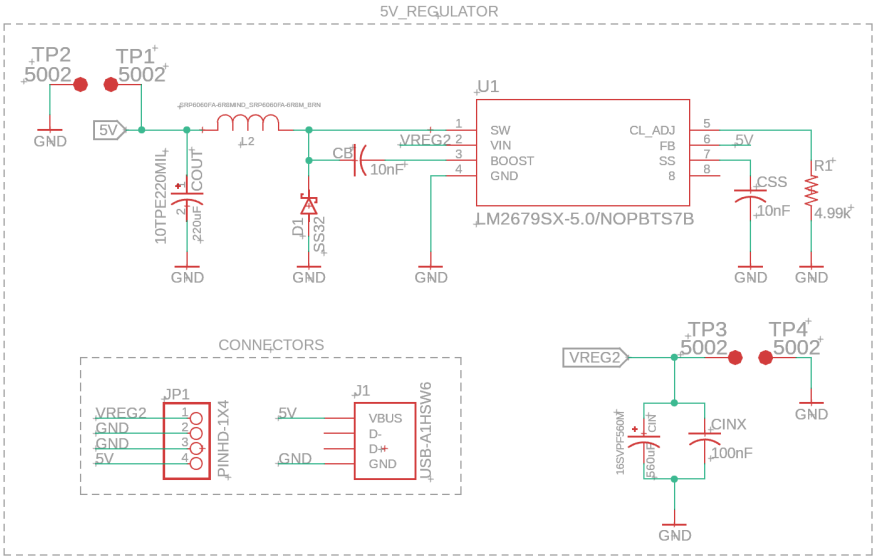



*Figure 6.1.1.4 5V Regulator Schematic*

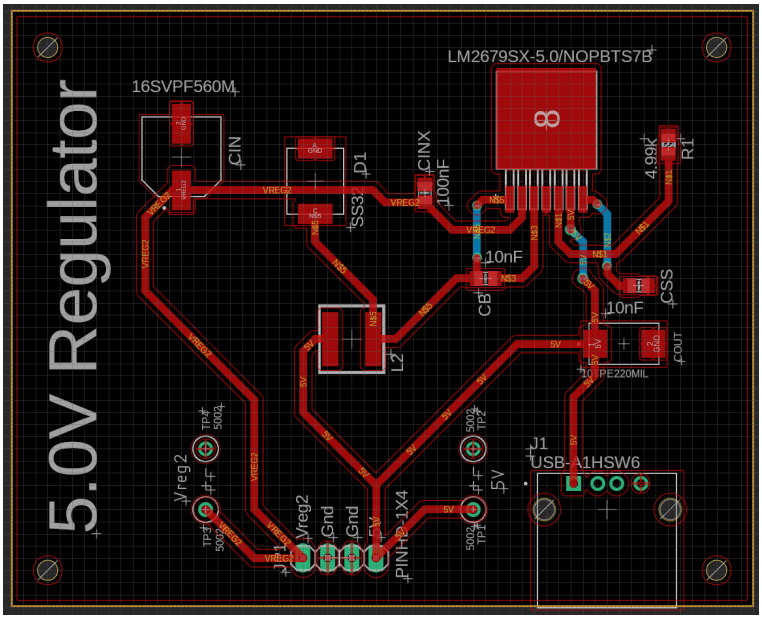
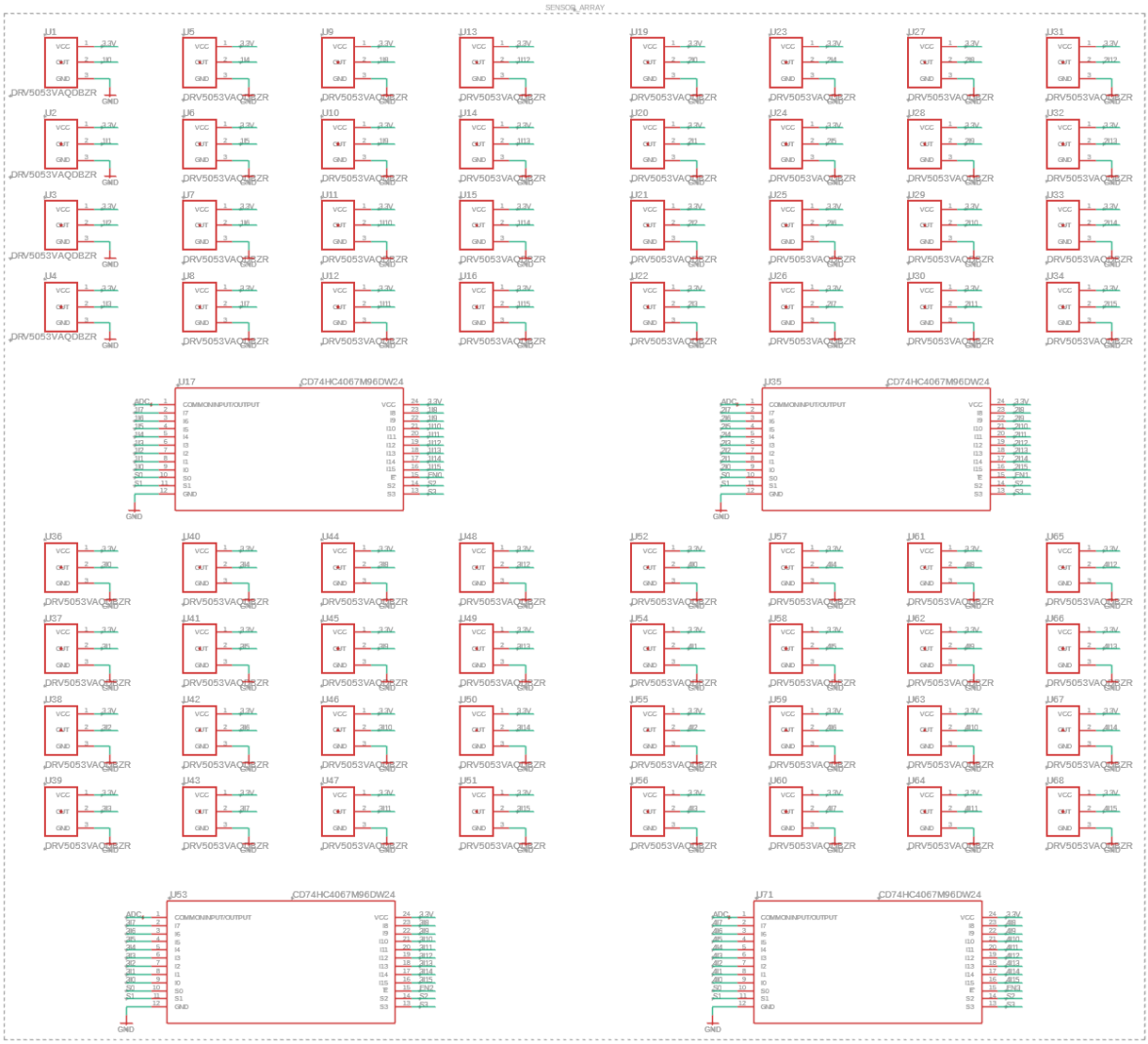



*Figure 6.1.1.5 5V Regulator PCB*

### 6.1.1.2 Sensor Array

The Sensor Array PCB was designed to capture magnetic field data using 64 Hall-effect sensors arranged in an 8 × 8 matrix. We selected the TI DRV5053VA, as it is a highly sensitive analog sensor that outputs a linear voltage proportional to magnetic field strength and polarity. The analog outputs of the sensors are routed into four CD74HC4067 analog multiplexers, each with 16 channels. Using four multiplexers reduces the number of ADC inputs necessary, as sensor outputs can be read sequentially. The selected output from each multiplexer is routed to a shared output line, which is then routed to the input of an ADS7883SDBVR 12-bit ADC. This ADC samples the signals and sends them to the ESP32 over SPI for further processing.



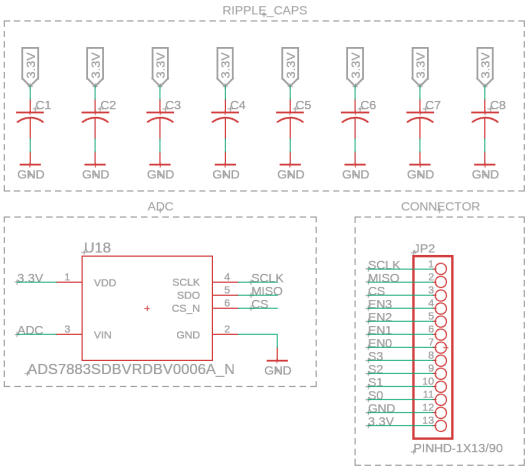



*Figure 6.1.1.6 Sensor Array Schematic*

Our PCB layout groups the 64 Hall-effect sensors into four 2x8 sections. This decision was made to minimize the analog trace lengths and overlap between the sensor outputs and their respective multiplexer channel, which was especially important for reducing signal degradation and noise. All four multiplexer select lines are connected using common traces to simplify routing, and their outputs are all routed into the ADC. A 13 pin header on this board has all of the required GPIO signals from the ESP32 PCB, including the multiplexer select lines, ADC lines, and 3.3V power input. The routing for this board was incredibly difficult at first, as many signals are being routed in close proximity to each other. After several iterations, we determined that the 2x8 grouping of sensors offered the best solution for signal integrity.

One of our major concerns when designing this PCB was the spacing of Hall-effect sensors. If the sensors were placed too close together, a present magnetic field might saturate too many sensors. On the other hand, spacing them too far apart could lead to poor resolution and gaps when visualizing a magnetic field. This concern led us to create two versions of the Sensor Array PCB: one with tightly spaced sensors and another with wider spacing. This allowed us to have a side-by-side comparison during testing to verify which board leads to a better visualization.

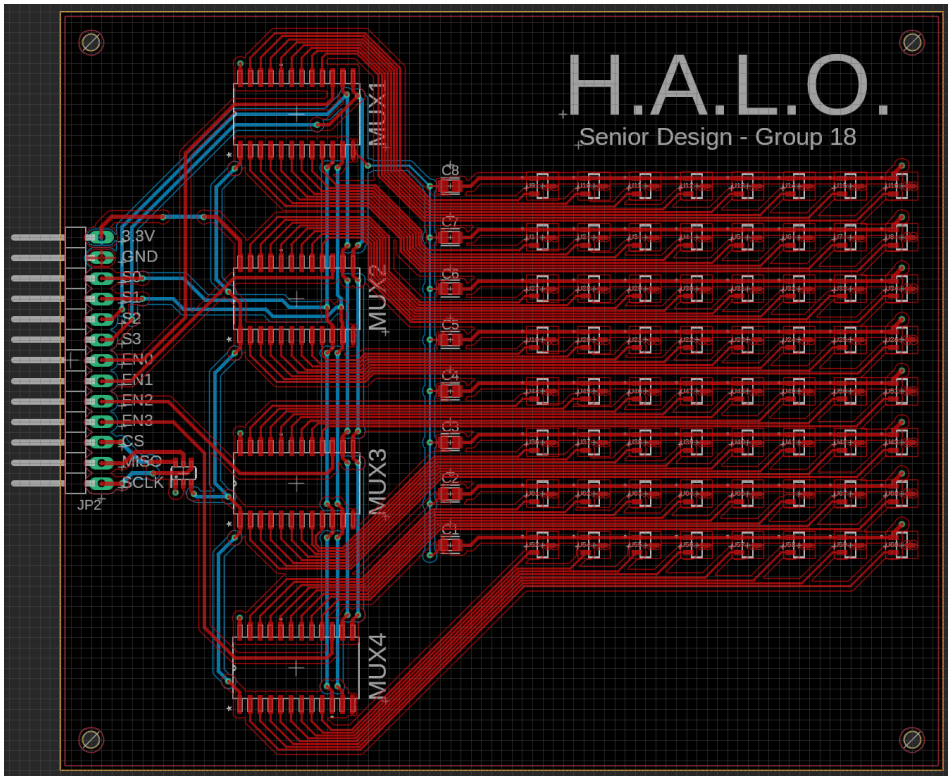



*Figure 6.1.1.7 Condensed Sensor Array PCB*

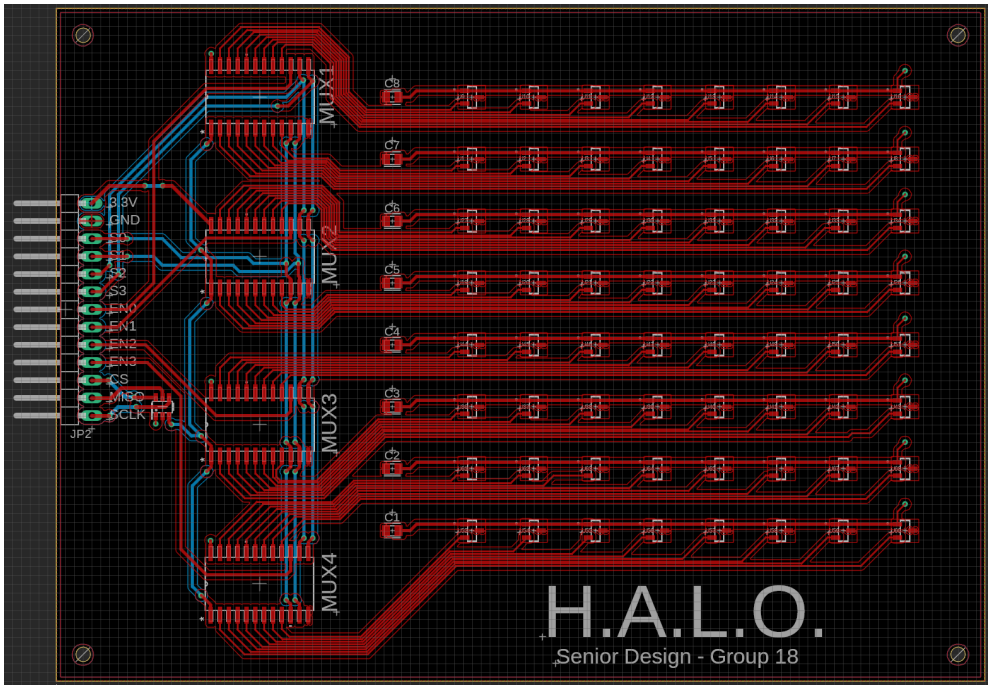



*Figure 6.1.1.8 Expanded Sensor Array PCB*

### 6.1.1.3 Data acquisition & Microcontroller

The Data Acquisition & Microcontroller PCB is built around the ESP32-WROOM module. Its primary function in this PCB is to control the timing of sensor readings, sample data from the ADC, and transmit the magnetic field data to the Raspberry Pi for visualization. The ESP32 receives sensor data from the ADS7883 12-bit ADC, which is transmitted over SPI. This communication protocol was chosen due to its high speed and reliability. The microcontroller also drives the four multiplexer select lines through GPIO. All of the necessary signals are routed from the ESP32 PCB to the Sensor Array PCB using a 13-pin header. To transfer data to the Raspberry Pi, the ESP32 communicates using UART. This connection uses a 3-pin header for the GND, TX, and RX lines that are required.

The schematic and PCB layout for this board were designed to ensure clean signal routing. All communication lines are kept short and grouped together to maintain signal integrity. The ESP32 is powered by the 3.3V Regulator discussed earlier and has a decoupling capacitor at the input to prevent fluctuations. To make this board truly standalone, a CP2102 USB-to-UART bridge is included for programming and debugging. The user can connect to the MCU using the USB-A port, implement new software, and reboot the system using the BOOT and RESET tactile switches on the PCB.

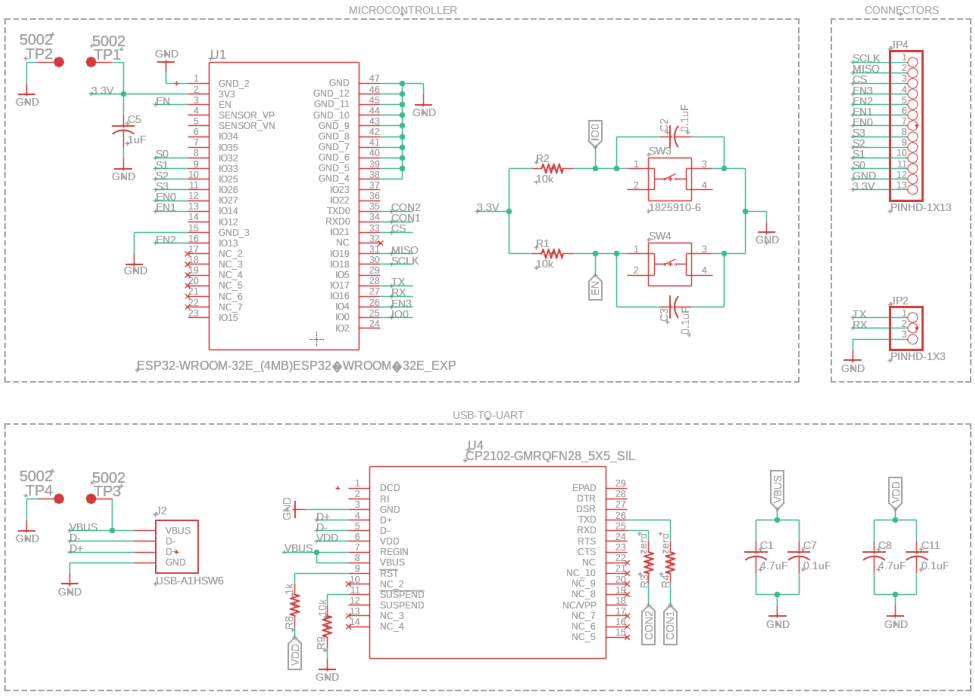



*Figure 6.1.1.9 Microcontroller Schematic*

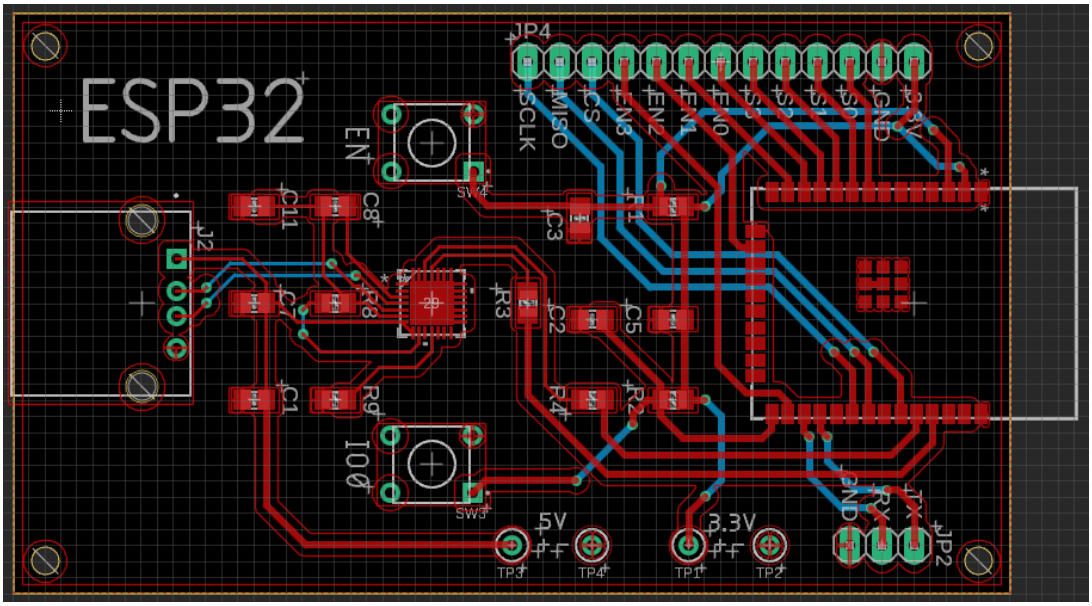



*Figure 6.1.1.10 Microcontroller PCB*

# 6.2 Schematic Justification

## 6.2.1 Power Supply Subsystem

The power supply subsystem is responsible for delivering regulated voltages to the various components of H.A.L.O., including the ESP32 microcontroller, sensor array, and Raspberry Pi. The main energy source for the system is a 7.4V lithium-ion battery pack composed of two 3.7V cells in series. To safely recharge these batteries, a Battery Management System (BMS) based on the MP2615 IC was selected. The MP2615 supports 2-cell lithium-ion charging and features overcharge protection, thermal protection, and automatic charge termination, allowing for safe and convenient recharging through a 12V, 2A DC barrel jack input. Before charging, the user must manually turn the power switch “off.” Once charging is complete, the 12V supply should be unplugged before powering the system back on. To provide feedback on power and charging status, two indicator LEDs are connected to the ACOK and CHGOK pins of the MP2615. The first LED indicates the presence of external power, while the second illuminates when charging is in progress. When the batteries reach full charge, the CHGOK LED turns off, signaling that the external supply can be safely disconnected.

To step down the 7.4V battery voltage to levels suitable for each subsystem, two dedicated voltage regulator PCBs were designed. For the 3.3V rail, the LM2576-ADJ was selected. This adjustable buck converter from Texas Instruments can supply up to 3A of output current, which exceeds the requirements of the ESP32 PCB and Sensor Array PCB. While the LM2576 operates at a relatively low switching frequency of 52 kHz and is less efficient than newer alternatives, it remains well-suited to this application due to its reliability, thermal performance, and ease of assembly thanks to its large footprint.

For the 5V supply required by the Raspberry Pi 4 and touchscreen LCD, the LM2679-ADJ was chosen. This regulator is similar in functionality to the LM2576 but operates at a higher switching frequency of 260 kHz and can deliver up to 5A of output current. These characteristics make it more appropriate for the Pi, which requires a stable 5V supply capable of sourcing at least 3A during peak load conditions.

In both regulator circuits, inductors were carefully selected to handle at least twice the nominal output current. This added current margin ensures the inductors will not saturate under transient conditions, improves voltage stability, and contributes to the overall robustness and reliability of the power subsystem.

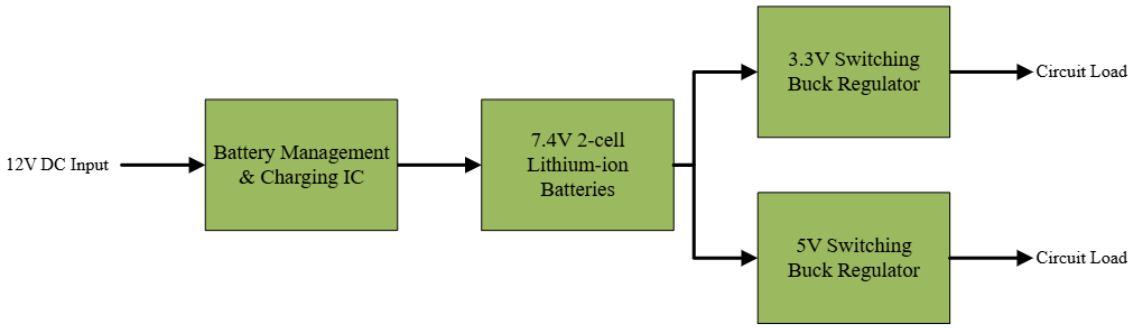



*Figure 6.2.1 Power Management Diagram*

## 6.2.2 Sensor Array & Multiplexing

At the front end of H.A.L.O. is an 8x8 array of analog Hall-effect sensors, which serve as the primary mechanism for detecting magnetic fields. This array

measures the strength and polarity of present magnetic fields, which the system will use to construct a two-dimensional heat map in real time. Each sensor generates a voltage output that varies linearly with the intensity and polarity of the magnetic field perpendicular to the surface of the sensor. The selected sensor for this application is the DRV5053VA from Texas Instruments. This sensor was chosen due to its linear analog output, low power consumption, compact footprint, and compatibility with our 3.3V voltage regulation. This sensor produces a ratiometric analog output centered around 1V when no magnetic field is present, with the voltage increasing or decreasing in proportion to the strength and polarity of the applied field.

The 64 sensors are powered by a common 3.3V rail and are individually routed to downstream circuitry. Due to the large number of sensors, it would be impractical to wire each output to an individual ADC channel. To solve this, H.A.L.O. uses a multiplexing scheme that incorporates four 16:1 analog multiplexers. Each multiplexer is responsible for a 4x4 quadrant of the array, with a common output pin routed to an external ADC input. This approach significantly reduces the number of ADC inputs required while still maintaining full access to the data from all 64 sensors. The corresponding block diagram for this subsystem can be found below.

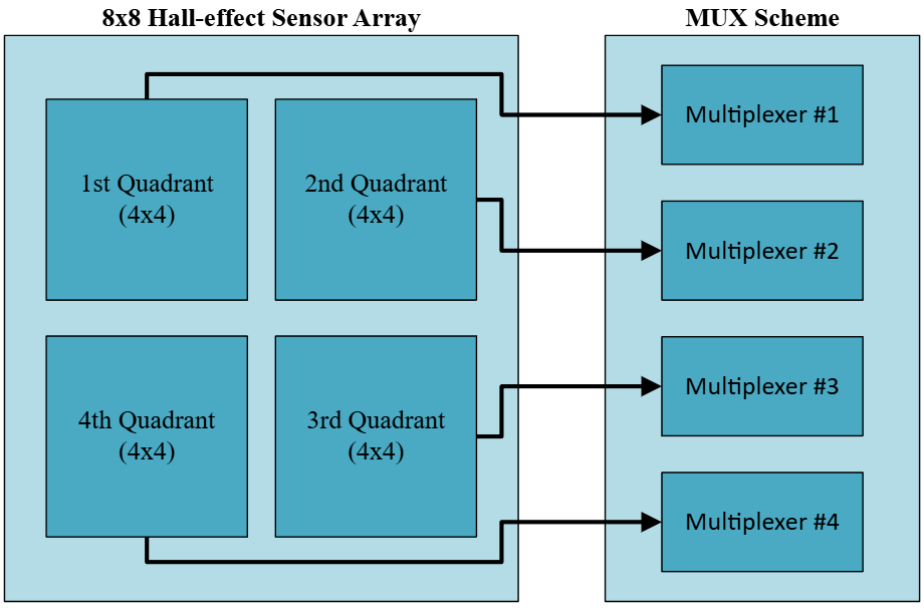



*Figure 6.2.2 Sensor Array Diagram*

The multiplexers are controlled by the ESP32 microcontroller using digital selection lines. By setting the appropriate control signals, the ESP32 can scan through each sensor sequentially in a defined order. This scanning occurs quickly enough that the full array can be read multiple times per second. This is important for maintaining a smooth, real-time magnetic field display. Overall, the combination of a high-density sensor array and an effective multiplexing scheme provides H.A.L.O. with a compact and scalable sensing architecture, allowing for dense spatial sampling while minimizing the number of ADCs required on the PCB.

### 6.2.3 Data Acquisition & MCU

The data acquisition subsystem in H.A.L.O. is responsible for converting the analog outputs of the Hall-effect sensors into digital data that can be processed, transmitted, and displayed. Once a sensor output is selected by a multiplexer, its analog voltage is routed to an external analog-to-digital converter (ADC) for digitization. H.A.L.O. uses a single external ADC, the Texas Instruments ADS7883SDBVR, a 12-bit SAR ADC with a 3 MSPS sampling rate and SPI output. It operates from a 3.3V supply and was selected for its fast throughput, low power consumption, and compact footprint. The output of each of the four CD74HC4067 multiplexers is routed to the input of this ADC. The ESP32 selects one multiplexer at a time, enabling sequential readout of all 64 sensor values.

The ESP32-WROOM microcontroller manages all data acquisition operations, including SPI communication with the ADC and controlling the multiplexer select lines. It cycles through all 64 channels by first selecting a multiplexer, then iterating through its 16 input lines, reading the voltage at each step. This process repeats for all four multiplexers, resulting in a complete scan of the 8×8 sensor array.

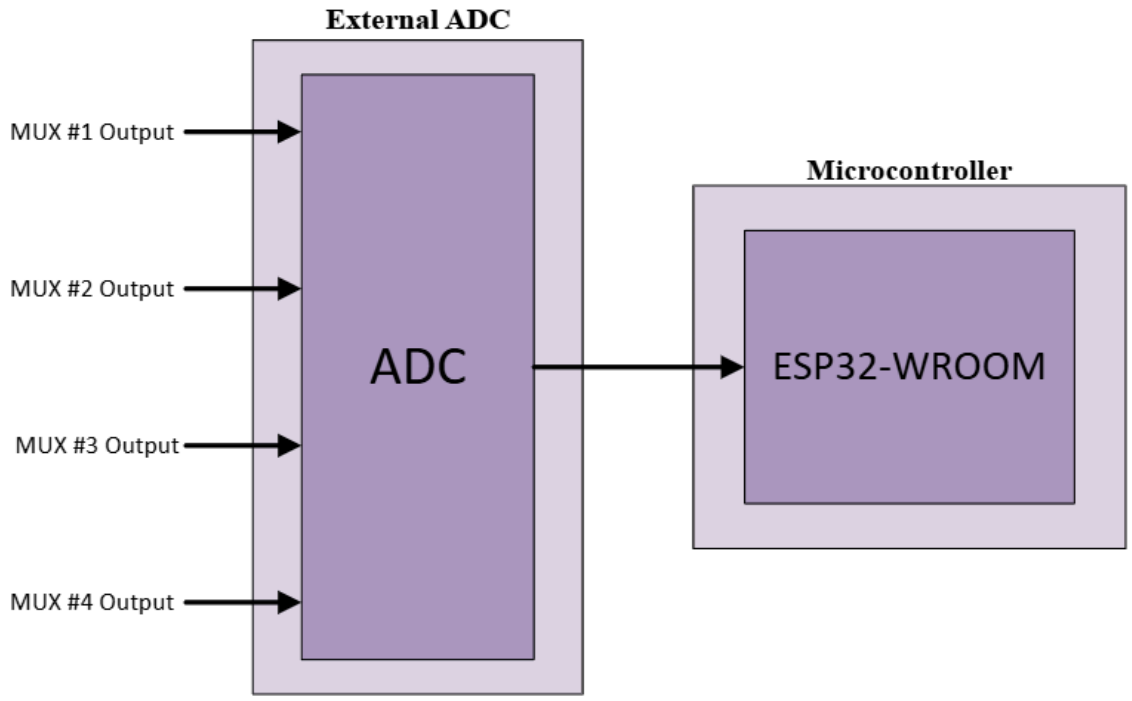



*Figure 6.2.3 ADC Diagram*

Once all 64 sensor values have been read, the ESP32 organizes them into an 8×8 matrix that matches the layout of the sensor array. This data is packed into a fixed-length format and sent over UART to the Raspberry Pi 4, which handles the display. The ESP32 and Pi communicate using a simple 3-wire serial connection (GND, TX, RX) through a 3-pin header. This setup allows for fast and consistent data transfer, which keeps the magnetic field display updating in real time.

Handling data acquisition on the ESP32 and keeping the Raspberry Pi focused on visualization helps split up the workload. Each processor does exactly what it's best suited for, which keeps the software clean and separates the key functions.

## 6.2.4 User Interface

The user interface of H.A.L.O. is responsible for visualizing the data collected by the sensor array in a format that is intuitive and responsive. This functionality is

handled by a Raspberry Pi 4 development board equipped with a touchscreen LCD, which receives processed sensor data from the ESP32 and renders it in real time. The Raspberry Pi is used exclusively for display purposes, allowing it to dedicate its resources to rendering updates and managing user interaction without interfering with sensor polling or data acquisition tasks.

Communication between the ESP32 and Raspberry Pi is performed over UART. The ESP32 transmits a complete dataset representing the 64 sensor values, formatted as a fixed-length data packet. Upon receiving this data, the Raspberry Pi reconstructs the 8x8 matrix corresponding to the physical layout of the Hall-effect sensor array. This matrix is then used to generate a 2D heat map on the touchscreen, where each cell represents the magnetic field strength at a specific location.

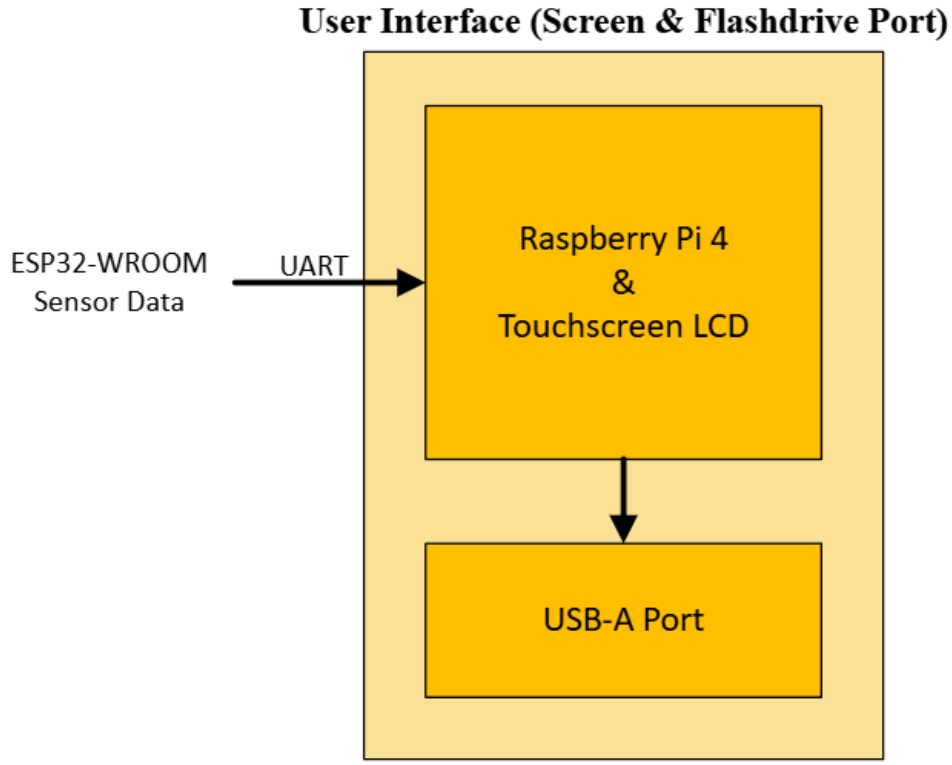



*Figure 6.2.4 UI Diagram*

The touchscreen interface provides a visual representation of the magnetic field in real time. Each sensor value is mapped to a color based on field strength, allowing the user to quickly identify magnetic sources and changes in the

environment. The display is updated multiple times per second, ensuring that the system remains responsive to new data and that changes in the field are reflected immediately.

Using the Raspberry Pi for the interface allows for a more flexible and powerful GUI environment while keeping the ESP32 focused on time-critical data acquisition. By isolating the user interface on a separate device, the design ensures that the ESP32 can handle time-sensitive data acquisition tasks without interruption. This separation of roles improves overall system stability and performance, and allows the display interface to be developed and modified independently from the sensor hardware.

# 7: Software Design

## 7.1 Overview of Software

The software architecture of H.A.L.O. was designed to transform raw data from a grid of Hall-effect sensors into a smooth, interactive visualization of magnetic fields in real time. While the initial concept used a bare-metal firmware architecture on an STM32H735IGT6 for display and an ESP32-WROOM-32 for sensor acquisition, extensive feasibility testing led to a significant redesign. In the final implementation, a custom PCB with an ESP32-WROOM-32 microcontroller handled all low-level sensor acquisition and preprocessing, while a Raspberry Pi 4 Model B ran a high-level Python application for data visualization and user interaction. This division of labor provided both deterministic low-level control and the flexibility of a powerful GUI platform.

The software followed a modular architecture, with each module focused on a distinct function—sensor acquisition, data processing, visualization, storage, and system control.

- On the ESP32, we implemented a bare-metal super-loop firmware with interrupt service routines (ISRs) for critical timing events such as multiplexer switching, ADC conversions, and UART transmission.
- On the Raspberry Pi, we implemented a multithreaded Python application that managed real-time rendering, touch input, and data logging.

This two-tier approach minimized overhead on the microcontroller while offloading computationally heavy tasks to the Pi, ensuring real-time responsiveness even under high data rates.

The ESP32 controlled four 16:1 multiplexers via GPIO select lines to scan an 8×8 Hall-effect sensor array.

- Each sensor output was digitized by external ADCs interfaced over SPI.
- A super-loop continuously cycled through all 64 sensors, gathering a complete frame of data.
- Interrupts were used to precisely time ADC read completions and to trigger rapid transmission of data packets.

The firmware normalized the raw ADC readings and packed them into fixed-length UART frames with sequence numbers and checksums for integrity.

The ESP32 performed basic signal conditioning and scaling to reduce the workload on the Raspberry Pi. Once a frame was ready, it was transmitted over a high-baud-rate UART link (921,600 bps) to the Raspberry Pi. This ensured a continuous, low-latency stream of sensor data.

On the Raspberry Pi, a Python application handled all graphical and interactive components:

- A dedicated receiver thread parsed incoming UART packets, validated checksums, and rebuilt the 8×8 sensor matrix in memory.
- A rendering module mapped sensor values to a 16-bin color gradient and drew the heat map.
- The interface updated multiple times per second, maintaining real-time responsiveness.

Touch input was handled through the Pi's built-in drivers, allowing for features like:

- Toggle View: switch between raw sensor values and the heat map.
- Save: write current frames to a USB flash drive for offline analysis.
- The Python GUI's modular design made it easy to add new visualization modes or processing steps without altering the ESP32 firmware.

In the original design, we planned to log data to an onboard SD card managed by FATFS middleware on the MCU. In the final system, logging was handled on the Raspberry Pi for simplicity and flexibility:

- Sensor data and snapshots were saved directly to a USB flash drive attached to the Pi.
- This approach eliminated the constraints of MCU-level file systems and allowed users to remove the drive for immediate access to recorded data.

System-wide behaviors such as startup initialization, mode transitions, and error handling were divided between the two platforms:

- The ESP32 initialized the sensor hardware and managed the acquisition cycle.
- The Raspberry Pi initialized the GUI, handled configuration files for scaling or calibration, and managed user-driven state changes (e.g., switching display modes).

Shifting from an STM32-centric design to a hybrid ESP32 and Raspberry Pi architecture addressed the challenges of real-time visualization on a resource-constrained MCU. The ESP32 provided deterministic, low-level control for precise sensor acquisition, while the Raspberry Pi delivered the computational horsepower and development flexibility needed for high-resolution graphics, rapid prototyping, and feature expansion. This separation of concerns resulted in a stable, responsive system that met our performance goals and simplified both hardware and software development.

## 7.2 Development Environment

Software development for H.A.L.O. ultimately utilized the Arduino IDE for programming and debugging the firmware on the custom ESP32 board. The Arduino environment was chosen because of its simplicity, rapid iteration cycle, and extensive community support. It provided straightforward access to libraries for SPI, UART, and GPIO control—features that were critical for managing the multiplexed sensor array and transmitting data in real time.

In the early stages of the project, we also used STM32CubeIDE when prototyping with the STM32H735G-DK development kit. This environment offered robust peripheral libraries and integrated debugging tools, which were

valuable during initial experimentation with display concepts. However, after the project pivoted away from STM32-based rendering, all user-interface functionality was moved to the Raspberry Pi.

On the Raspberry Pi 4 Model B, we implemented the high-level user interface in Python, taking advantage of its rich ecosystem for GUI development and rapid prototyping. Python libraries were used to handle UART communication with the ESP32, process incoming sensor frames, and render the real-time heat map on the touchscreen. This toolchain combination allowed the team to rapidly iterate on firmware while still building a robust, high-performance visualization interface on the Raspberry Pi.

## 7.3 User Interface

The user interface (UI) of H.A.L.O. was responsible for visualizing the data collected by the sensor array in a format that was both intuitive and responsive. In the final system, this functionality was handled by a Raspberry Pi 4 Model B connected to a 5-inch 800×480 Hosyond capacitive touchscreen LCD. The Raspberry Pi received processed sensor data from the ESP32 in real time and rendered it into a graphical display that allowed users to observe magnetic field changes instantly and interact with the system directly through the touchscreen.

Early in development, we initially planned to handle the display with an STM32 development board, leveraging its built-in display controller and MIPI DSI interface. The ESP32 would send processed sensor data to the STM32 over SPI, and the STM32 would reconstruct the 8×8 sensor matrix and generate a heat map. This approach seemed attractive because it isolated the user interface on a dedicated microcontroller, allowing the ESP32 to focus on time-sensitive data acquisition tasks.

However, as we progressed, we found that the STM32 solution could not meet our performance goals. The complexity of rendering a high-resolution heat map with a responsive interface at our target frame rate (60 FPS) exceeded what the STM32 could reliably sustain while remaining easy to develop for. After weighing alternatives, we pivoted to using a Raspberry Pi as the dedicated UI processor. The Pi's much higher processing power, memory capacity, and mature development ecosystem (with Python libraries readily available) allowed us to

implement richer visualization features while maintaining real-time responsiveness.

In the final design, the ESP32 handled all low-level tasks:

- It continuously cycled through the 8×8 Hall-effect sensor array, controlled four 16:1 multiplexers, and read the analog outputs through external ADCs.
- After sampling all 64 sensors, the ESP32 packaged the data into fixed-length UART packets, including frame counters and checksums for data integrity.
- The ESP32 transmitted these packets over a high-baud-rate UART connection (921,600 bps) to the Raspberry Pi.

On the Raspberry Pi, a dedicated receiver thread decoded these packets, validated the checksums, and reconstructed the 8×8 matrix representing the magnetic field measurements.

Once a complete frame of data was received, the Raspberry Pi's Python application translated the sensor readings into a visual heat map:

- Each sensor value was mapped to a color based on magnetic field strength, using a 16-bin gradient that transitioned smoothly from bright blue (low field) through less intense blues and reds to bright red (high field).
- The sensor matrix was rendered onto an 800×480 canvas, scaled to fill the touchscreen display.
- Updates occurred multiple times per second, providing real-time feedback and allowing users to observe dynamic changes in the environment.

The processing pipeline ensured that latency between sensor acquisition and display update was minimal, preserving the system's responsiveness.

The capacitive touchscreen was not just a display but also the primary input device. The interface included:

- View Toggle: A button to switch between a numerical data grid and the heat map visualization.

- Save: A button that stored the frame data and visualization onto an attached USB flash drive for offline analysis.
- Extensibility: The Python-based GUI was structured to allow easy addition of future features, such as alternate color maps, filtered data views, or diagnostic overlays.

Because the Raspberry Pi handled all UI tasks, user interactions did not interfere with the ESP32's data acquisition loop. This separation improved system stability and allowed the display interface to be developed and iterated independently.

By isolating the user interface on the Raspberry Pi, we achieved a responsive and visually rich interface without overloading the low-level hardware. The Pi's abundant processing resources allowed us to implement higher-resolution graphics, smooth transitions, and interactive controls, all while maintaining reliable communication with the ESP32. This architecture ensured that time-sensitive sensor acquisition tasks were never interrupted and that the interface could continue evolving without impacting the core acquisition firmware.

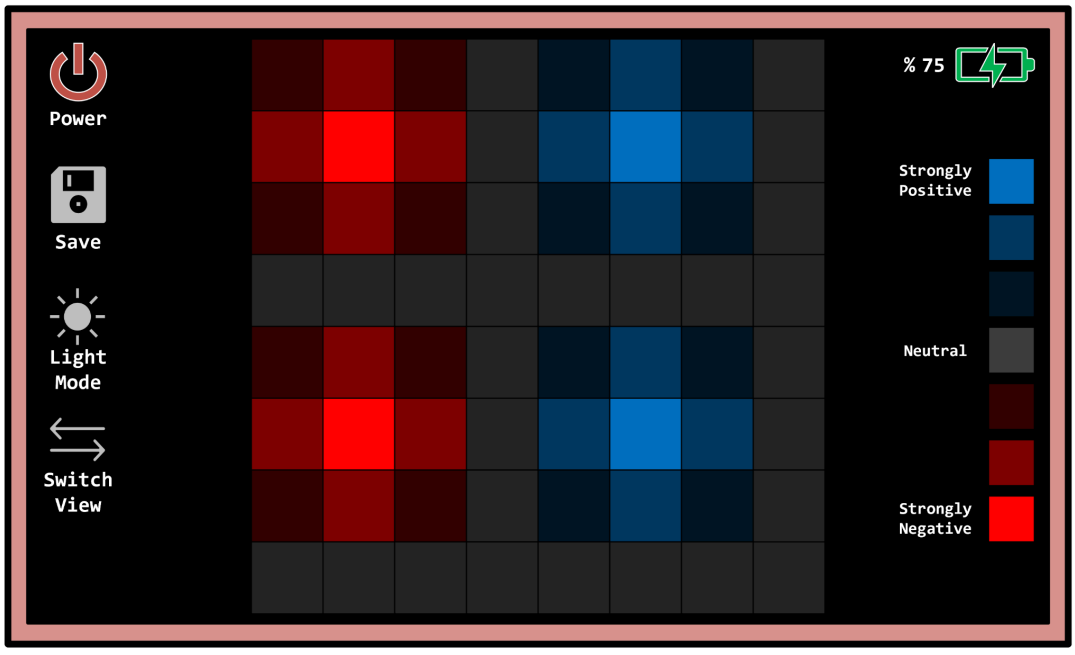



*Figure 7.3 H.A.L.O. Example Graphical User Interface*

## 7.4 Peripheral and Driver Integration

H.A.L.O. leveraged several key peripherals and external components to achieve its core functionality, and integrating these into the software required careful coordination between hardware and firmware. At the heart of the sensing subsystem were 64 Hall-effect sensors, each outputting an analog voltage proportional to the local magnetic field strength. These sensors were arranged in an 8×8 grid to provide a dense spatial sampling area. Reading 64 independent analog signals directly would have required a microcontroller with an unusually high number of ADC inputs, which would have been impractical. To solve this, we implemented a multiplexing scheme using four 16:1 multiplexers. Each multiplexer could route one of sixteen sensor signals to a single ADC channel, and by driving the select lines of these multiplexers with GPIO pins from the ESP32, the firmware was able to scan through all 64 sensors in a systematic and repeatable way. This setup drastically reduced pin count and board complexity while still giving full coverage of the sensor array.

The ESP32 firmware was written in C and deployed using the Arduino IDE. It operated as a bare-metal application structured around a super-loop architecture with interrupt service routines for time-critical events. The super-loop iterated continuously, stepping through each multiplexer state, triggering ADC conversions, and collecting data. Interrupts ensured that sampling, data formatting, and transmission remained tightly synchronized and did not drift over time. External high-speed ADCs connected over SPI provided digitized readings from each sensor with low latency. The ESP32 configured and clocked these ADCs to achieve a sampling rate high enough to cycle through all 64 sensors multiple times per second. Once a complete frame of sensor data was captured, the firmware applied basic normalization to map the raw ADC values to a consistent intensity range suitable for visualization. This minimal preprocessing ensured that the data was ready to be used without adding unnecessary overhead or computation on the ESP32, which needed to maintain a tight timing loop.

Transmission of sensor data from the ESP32 to the visualization platform was accomplished through a high-baud-rate UART link running at 921,600 bits per second. Each frame of data was serialized into a fixed-length packet containing all 64 readings as 16-bit values, along with a frame counter and a checksum for

error detection. This design ensured data integrity even at high transmission rates. On the receiving end, the Raspberry Pi 4 Model B continuously listened to the UART stream, using a dedicated background thread in Python to parse the incoming bytes. The thread validated the checksum, confirmed packet order with the frame counter, and reconstructed the 8×8 matrix of sensor values. Because this process was fully decoupled from the rendering loop, it could keep up with the ESP32's output even under heavy load.

Visualization was handled entirely on the Raspberry Pi, which proved to be far more capable than the microcontroller-based STM32 solution initially considered. The Raspberry Pi ran Raspberry Pi OS and hosted a Python application built around libraries such as Tkinter for GUI elements and Pillow for image processing. The graphical output was rendered to a 5-inch 800×480 Hosyond capacitive touchscreen LCD connected through the Pi's DSI interface. Each of the 64 sensor readings was mapped to a color representing field intensity, using a 16-step gradient from purple through blue, green, and yellow to red. The Python program updated the visualization several times per second, creating a smooth and responsive heat map. To make the system interactive, the capacitive touchscreen's native drivers were used to capture touch events. The GUI included onscreen buttons that allowed the user to toggle between a raw numerical display and the heat map view, as well as a button to save snapshots of the current frame. Pressing the snapshot button triggered Python routines that wrote the sensor data and corresponding image to a mounted USB flash drive, making retrieval of logged data as simple as unplugging the drive.

Unlike the original STM32 design, which would have required LVGL, LTDC, and careful memory management to render graphics, the Raspberry Pi allowed us to take advantage of its built-in GPU acceleration, larger RAM pool, and operating system support. This removed many of the limitations that constrained development early on. The GUI could be expanded or modified purely in software without touching the sensor firmware. Debugging and iteration were much faster because changes to the Python code could be deployed directly on the Pi without reflashing an MCU.

The GPIO integration on the ESP32 also required attention. Each multiplexer's select lines were driven through dedicated pins, and careful routing ensured minimal crosstalk or latency. Interrupts were configured to handle critical timing,

allowing the ESP32 to complete ADC reads and prepare the next channel selection without falling out of sync. This synchronization was crucial, as any drift or missed timing would have caused inconsistent frames and visible artifacts in the heat map display.

By shifting the display workload from an STM32 to a Raspberry Pi, we eliminated the complexity of embedded display controller integration and avoided the limitations of microcontroller-level graphics libraries. The ESP32 focused on real-time acquisition with deterministic timing, while the Raspberry Pi provided the computational headroom to produce smooth, detailed, and interactive visualizations. This division of responsibilities resulted in a system that reliably acquired, processed, and displayed magnetic field data in real time while remaining flexible for future enhancements.

## 7.6 Task Scheduling and Concurrency

Real-time management on the ESP32 was implemented using a straightforward, bare-metal approach rather than a full RTOS. Because the system's requirements were well-defined and tightly constrained, we opted for a super-loop architecture augmented with carefully placed interrupt service routines to guarantee timing consistency and responsiveness. The ESP32 continuously cycled through sensor acquisition steps in its main loop, controlling the multiplexer select lines, triggering ADC conversions over SPI, and collecting results. Interrupts were configured to fire on ADC completion events and SPI transfer completions, ensuring that no sampling windows were missed and that data was captured at predictable intervals. Once a complete frame of 64 readings was ready, the firmware serialized the values into a fixed-length packet and pushed it to the UART transmitter, again using interrupts to avoid blocking the main loop during data transmission. This design allowed the ESP32 to maintain a stable acquisition cadence without introducing the overhead or latency of preemptive multitasking frameworks.

On the Raspberry Pi side, the architecture was inherently different because the Pi ran a full Linux environment. Task scheduling and concurrency were handled at the application level using Python's threading model. A dedicated background thread handled continuous UART communication with the ESP32, reading and validating incoming packets as soon as they arrived. Meanwhile, the main thread

managed the graphical rendering pipeline, updating the heat map on the 5-inch capacitive touchscreen display, responding to touch events, and handling user commands such as toggling views or saving snapshots. Because these threads ran concurrently, incoming data could be received and buffered while the GUI continued to render without stalling. The Pi's operating system also handled low-level concurrency, ensuring that touch input drivers, file system writes to the USB flash drive, and display updates all operated smoothly in parallel with the sensor data stream.

This combination of a tightly controlled super-loop with interrupts on the ESP32 and a multithreaded Python application on the Raspberry Pi provided a stable and efficient concurrency model. The ESP32's deterministic timing ensured that every sensor was sampled in sequence without jitter, while the Raspberry Pi's higher-level concurrency allowed for fluid rendering and responsive user interaction, even as data was continuously received and logged. By splitting these responsibilities between platforms and tailoring the scheduling approach to each, the overall system achieved real-time performance without overcomplicating the design.

## 7.7 Software Flow and Structure

### 7.7.1 ESP32 Main PCB

The software running on the ESP32 was built around a straightforward super-loop architecture, chosen to keep execution predictable and debugging simple. At startup, the firmware initialized all necessary hardware peripherals, including the GPIOs that controlled the multiplexer select lines, the SPI interfaces used to communicate with the external ADCs, and the UART interface used to transmit data to the Raspberry Pi. Once initialization was complete, the ESP32 entered its main execution loop. Within this loop, the firmware cycled sequentially through the sensor inputs by stepping through the multiplexer select combinations. For each selection, the firmware triggered ADC conversions on the external converters, waited for conversion completion, and collected the resulting sensor data.

After all 64 sensors in the 8×8 grid had been read, the ESP32 formatted the collected values into a fixed-length data packet that included a frame counter and checksum to ensure data integrity. This packet was then transmitted over a high-baud-rate UART connection directly to the Raspberry Pi, which handled all subsequent processing and visualization. The loop then iterated again, maintaining a consistent cadence so that new frames of data were produced several times per second. By keeping the architecture simple and deterministic, the ESP32 was able to reliably handle real-time sensor acquisition without introducing latency or jitter, providing the Raspberry Pi with a steady stream of accurate sensor data for visualization.

### 7.7.2 Raspberry Pi 4

The software running on the Raspberry Pi 4 was structured as a multithreaded Python application designed to manage data reception, visualization, user interaction, and data logging concurrently. At startup, the program initialized the serial communication interface to establish a high-baud-rate UART link with the ESP32, configured the 5-inch 800×480 Hosyond capacitive touchscreen, and prepared the graphical environment using Python libraries such as Tkinter and Pillow. Once initialization was complete, the application launched a dedicated background thread to continuously read incoming data packets from the ESP32. This thread validated each packet using a checksum, extracted the 64 sensor values, and reconstructed them into an 8×8 matrix that mirrored the physical sensor layout.

While the data reception thread handled incoming frames, the main thread focused on the graphical user interface. It rendered the sensor matrix as a real-time heat map, applying a 16-bin color gradient ranging from purple to red to represent increasing magnetic field intensity. The GUI refreshed multiple times per second, ensuring that changes in the environment were reflected almost immediately on the display. The touchscreen also supported interactive controls: a button allowed the user to toggle between a raw numerical grid view and the heat map visualization, and another button allowed the user to save the current frame and its associated visualization to a USB flash drive for later analysis. Because these tasks ran concurrently, the GUI remained responsive even while data was continuously being streamed from the ESP32.

This software structure on the Raspberry Pi provided a clear separation between data acquisition and user interface management. By isolating data reception in its own thread, the system avoided blocking operations and ensured that the display could update without interruptions. This design allowed the Raspberry Pi to fully leverage its operating system's multitasking capabilities, enabling smooth visual output, responsive controls, and reliable data logging while maintaining real-time synchronization with the ESP32's data stream.

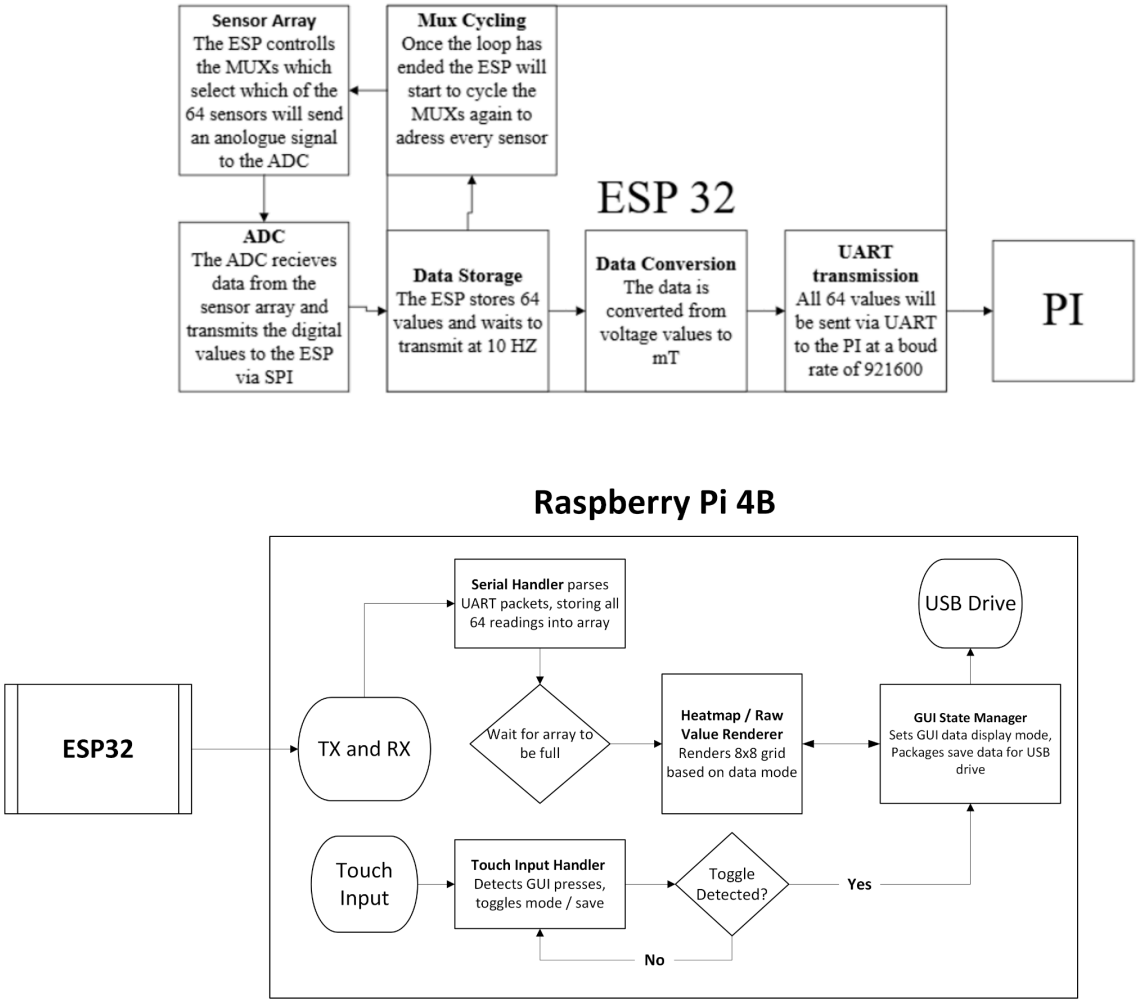



*Figure 7.7 Software Flow and Structure*

# 7.8 Software Challenges and Solutions

Developing the H.A.L.O. software involved a highly iterative process, with many challenges encountered and solved as we refined both the firmware on the

ESP32 and the GUI on the Raspberry Pi. One of the earliest challenges stemmed from our initial plan to run both sensor acquisition and display rendering on a single STM32. While we were able to prototype basic functionality, the system could not sustain our desired frame rate once we scaled up to a full 8×8 sensor array and high-resolution graphics. This limitation prompted us to pivot to a split architecture, where the ESP32 focused solely on sensor acquisition and the Raspberry Pi handled visualization and interaction. That decision became the foundation for further iterations, giving us the flexibility to continually expand functionality on the Pi without destabilizing the acquisition firmware.

On the ESP32 side, integrating the ADCs through SPI required extensive troubleshooting to achieve clean, stable readings. At one point, we attempted to read each 16-bit value from the ADC as two separate 8-bit transfers, but this method introduced significant noise into the readings and created inconsistencies across the sensor array. After experimenting and reviewing the ADC's communication protocol, we switched to using the SPI.transfer16() function to read the full 16-bit value in a single transaction. This change eliminated the noise artifacts and produced reliable, repeatable sensor data for every channel. Another subtle but critical issue was an "overflow" effect we observed in the heat map output. Some values were spiking far beyond expected ranges, which created bright artifacts on the display. After reviewing our bit-manipulation logic on the ESP32, we discovered that we had been shifting out too many bits when processing the raw ADC readings. The ADC output contained two null bits at the start, so only two right shifts were necessary. Correcting this eliminated the overflow effect entirely and resulted in clean, stable data reaching the Raspberry Pi.

On the Raspberry Pi, the GUI and its functionality evolved iteratively over the course of development. We began with a simple proof of concept that displayed an 8×8 heat map with static test data to validate rendering on the 5-inch 800×480 capacitive touchscreen. Once that baseline worked reliably, we integrated live data reception from the ESP32, validated packet handling, and ensured that frame updates remained smooth. From there, we built out additional features step by step, such as a toggle to switch between a raw numerical view and the heat map, as well as the ability to save snapshots of the data to a USB flash drive. Each new feature was added on top of a known-good

foundation, which reduced regression bugs and allowed us to steadily expand the interface without breaking existing functionality.

Another area where iteration was critical was concurrency. Early in development, the Python program handled data reception and rendering in the same thread, which occasionally caused input lag or dropped frames when the system was under load. We solved this by restructuring the program into a multithreaded design, with a dedicated thread managing UART reception and buffering, while the main thread handled rendering and touch events. This separation ensured that the GUI remained responsive even during continuous data flow. Likewise, data logging initially caused stutters when performed directly in the rendering loop. By offloading file operations and buffering writes to the USB flash drive, we eliminated these interruptions and maintained smooth visual updates.

Overall, the development of H.A.L.O.'s software was characterized by this iterative process of testing, identifying issues, and refining both low-level firmware and high-level GUI features. Solving problems like noisy ADC readings, incorrect bit shifting, and GUI concurrency issues not only improved performance but also deepened our understanding of the system's behavior. By building incrementally on each validated step, we created a robust and responsive platform that reliably transforms raw sensor data into an intuitive, real-time visualization.

# 8: Fabrication & Prototyping

## 8.1 PCB Design Software

All hardware design for H.A.L.O. was completed using autodesk Fusion 360, which has its own integrated schematic capture and PCB layout environment. This feature alone provides a seamless workflow between design stages. The software's large component library and the ability to import custom footprints and 3D models helped to accommodate all necessary parts. Since we have had experience using this software in previous courses at UCF, it seemed to be the best available option. A key feature that simplified routing, especially for the Sensor Array PCB, was the push and shove routing tool. This feature made manual routing much easier by automatically adjusting nearby traces to avoid

overlapping. Fusion 360 also has 2D and 3D visualization tools, helping to verify component placement and mechanical fit in our enclosure. While Fusion 360 is a paid software, educational licenses giving UCF students free access makes it a practical choice for our project.

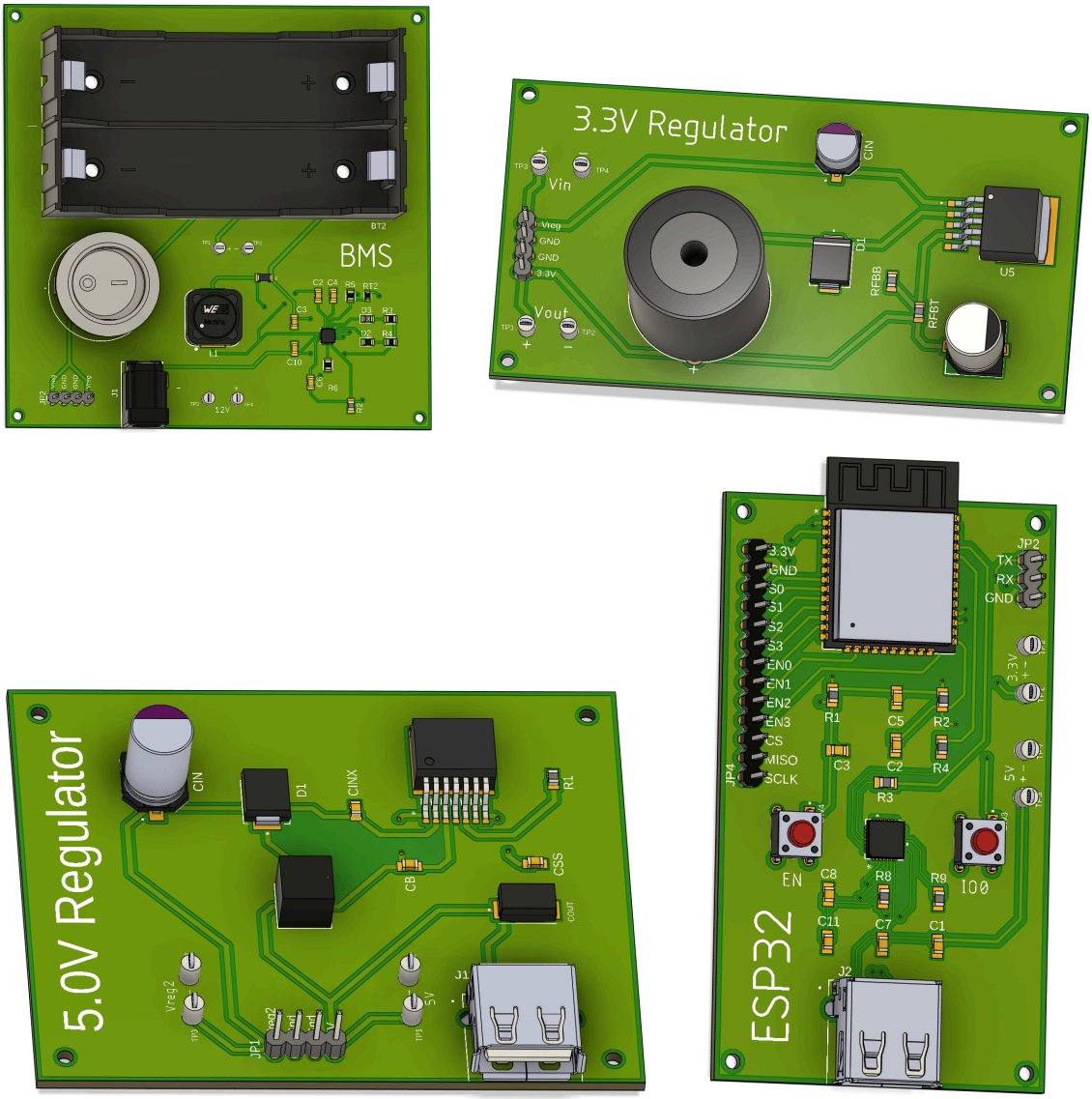



*Figure 8.1.0 Fusion 360 PCB 3D Models*

## 8.2 PCB Vendor Selection

When fabricating our PCBs, we used JLCPCB as the primary vendor. This manufacturer was chosen because of its low cost, fast turnaround, and solid

reputation among electrical engineers. Their interactive website made the ordering process straightforward, as we only needed to upload our gerber files to the website. Design rule checks and pricing estimates are conducted immediately, allowing us to make quick decisions if there were any changes that needed to be made. The Sensor Array PCB, for example, required fine traces and precise alignment. JLCPCB's standard capabilities were more than sufficient for our layout, and we didn't need to pay extra for tighter tolerances. Turnaround time was less than one week, which gave us plenty of time to create and test our prototypes.

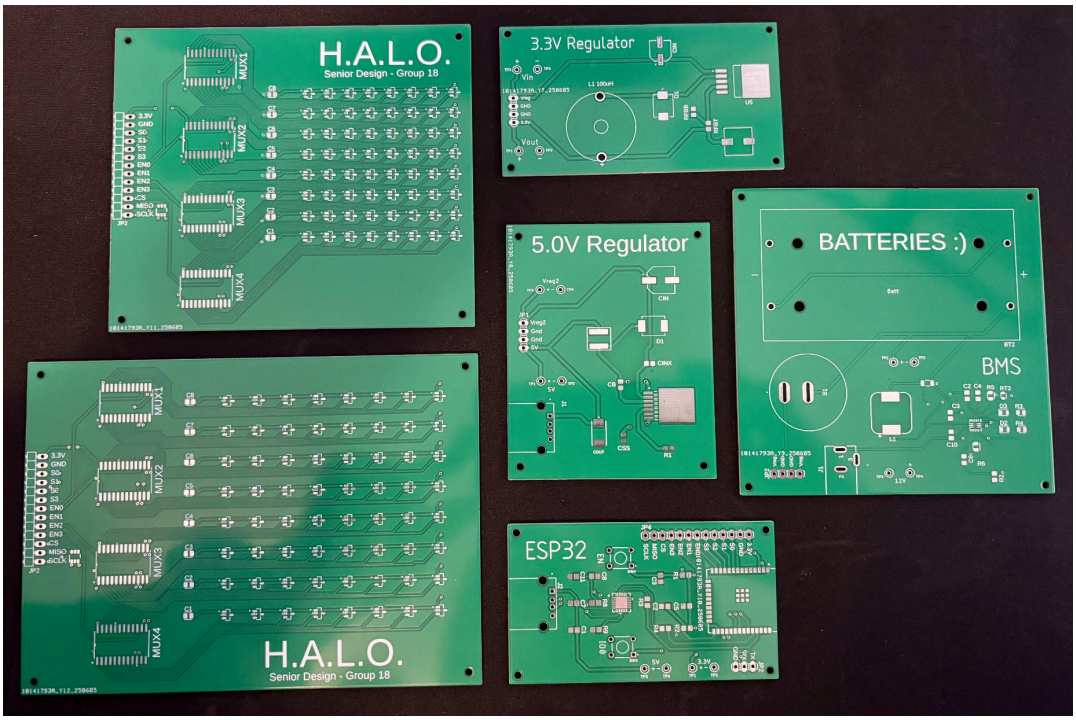



*Figure 8.1.2 Fabricated PCBs*

## 8.3 PCB Prototyping

After receiving the fabricated PCBs from JLCPCB and obtaining all necessary components from DigiKey, we began the process of physically assembling the boards. This step was focused entirely on fabrication, ensuring that each board matched its intended schematic. We made use of the tools available in the senior design lab, including soldering irons, solder paste, heat guns, and magnification equipment.

To ensure accurate PCB assembly, we closely referenced the schematics to verify component values, orientations, and pinouts before placing each part. Cross-checking with the schematic helped prevent errors like incorrect values or mirrored parts, which could have compromised functionality and made debugging more difficult later on.

We used different soldering techniques depending on the package type and density of the components. Surface-mount devices were typically placed using solder paste and then heated with a heat gun to reflow the joints. This method was very helpful for the Sensor Array PCB, which includes 64 Hall-effect sensors arranged in a dense 8×8 matrix. One major benefit of using solder paste is that any excess away from the metal pads turns into flux during heating, which helps to prevent short circuits and improves joint quality. Through-hole and larger surface-mount components were soldered manually using a fine-tip iron and solder wire. This was the preferred method for connectors, power jacks, and switches.

By assembling each board individually and closely referencing the schematics, we maintained a high level of control during the build process. This modular approach allowed us to identify and isolate any fabrication issues specific to a single board before full system integration.

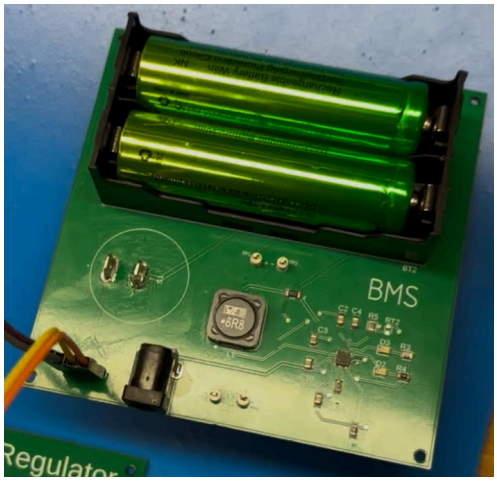



*Figure 8.3.0 Battery Management System Prototype*

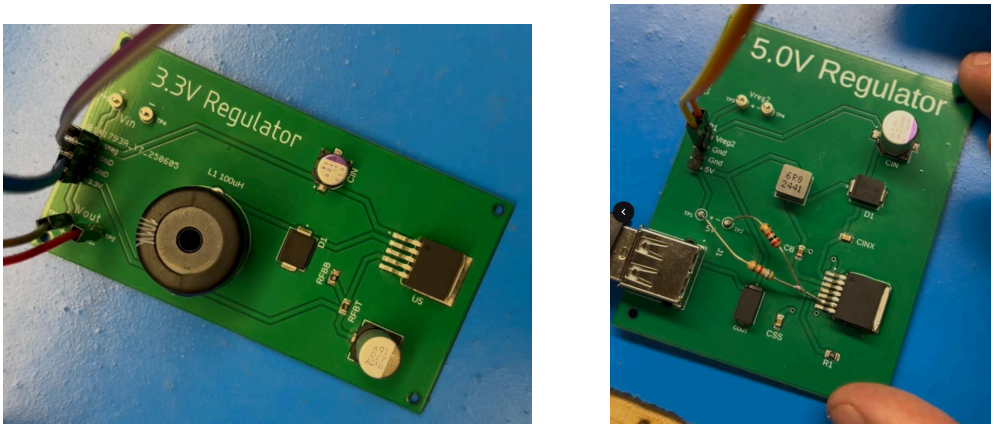



*Figure 8.3.1 3.3V & 5V Regulator Prototypes*


*Figure 8.3.3 ESP32 Module Prototype*


*Figure 8.3.4 Sensor Array Prototype*

## 8.4 Prototype Enclosure Design

The enclosure for this system needed to house the 5 PCBs, Raspberry Pi, display, and all the wiring connections that make up the system while still allowing access to the screen of the display, the USB ports of the Pi, the power switch of the BMS PCB, and the 12 V jack of the BMS PCB.

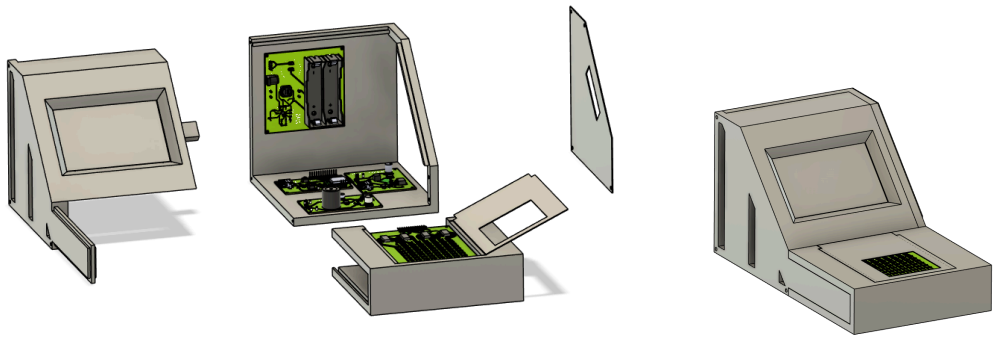


*Figure 8.4.1 H.A.L.O. Deconstructed*

Figure 8.4.1 shows all of the components of the H.A.L.O. enclosure in a deconstructed format. In total, there are 5 parts that assemble to fully house the system: the main holder, display holder, array holder, array cover, and wall cover.

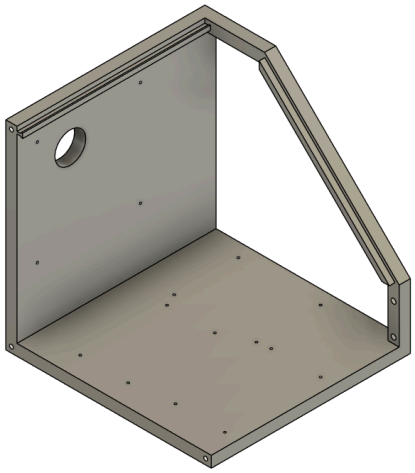


*Figure 8.4.2 Main Holder*

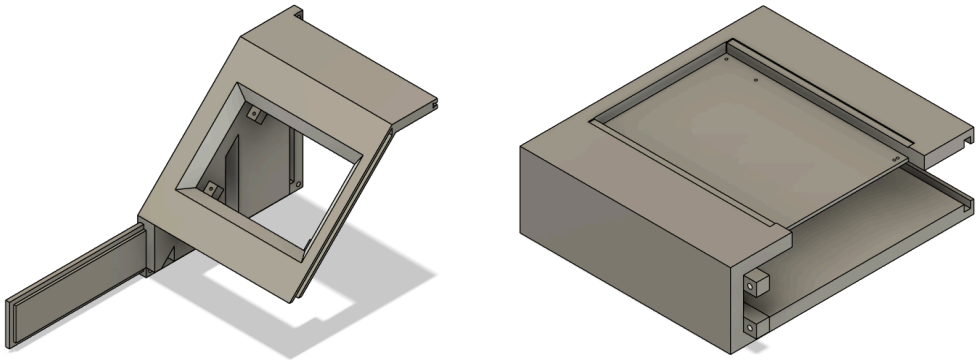


*Figure 8.4.3 Display holder (left) and Array Holder (right)*

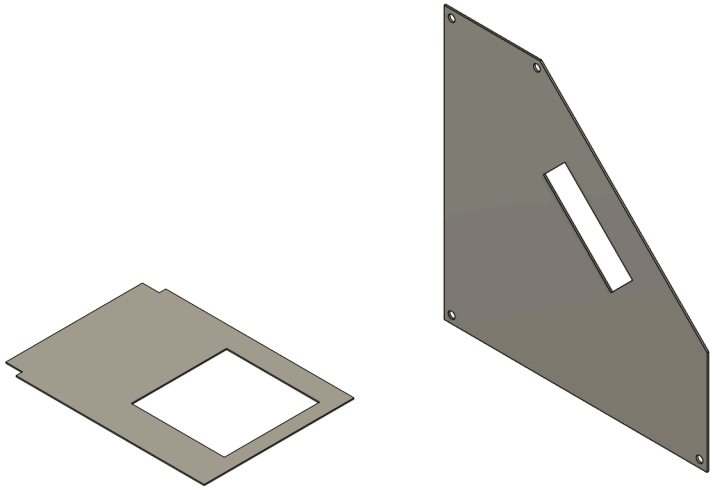


*Figure 8.4.4 Array Cover (left) and Wall Cover (right)*

Figure 8.4.2 shows the main component of the housing. It holds the BMS, regulator, and MCU PCBs. There are holes modelled into the holder that allow for the PCBs to be secured by bolts. The large hole on the back wall aligns with the power switch on the BMS PCB. The absence of a wall on the right allows the user access to the system if need arises. This access point can be useful for removing/inserting the batteries, testing the system, checking the routing, etc. This is covered by the wall cover.

There are holes through the wall cover and in the main holder that align which allows for the components to be screwed together. This is also the case for the

display holder and array holder. Both of these components can be secured to the main holder with bolts. The top and slanted extrusions on the main holder align with cutouts in the display holder, resulting in the two components sliding into position for assembly.

The display holder holds the display and the Raspberry Pi. The Pi and display connect via a ribbon cable. This cable is very sensitive, so it's best to just leave the two connected how the display manufacturer intended, with the Pi mounted to the back of the display. There are arms behind the display window on the holder that allow the display to be secured with bolts. There's a large protruding arm on the front of the display holder that fits into the gap in the array holder. This results in the two components sliding together for assembly. The slots on the side of the display holder are for ventilation, and the largest one doubles as an access point to the 12 V jack.

The array holder has two depressions. The lowest houses the sensory array PCB. The PCB is secured using the same method described for the main holder. The higher depression fits the array cover and allows it to sit flush with the array housing. It can simply be taken on and off, but we secured one edge with electrical tape. This causes the cover to act like a hatch that opens and closes on top of the sensor array.

The array cover has a square window that aligns with the sensor array so as not to obstruct the sensors. The rectangular window in the wall cover is for USB access. The hole aligns with the USB ports of the Raspberry Pi.

# 9: Testing and Evaluation

To validate both our firmware and hardware integration we set up a test on a bread board before committing to our PCB design. For this initial test we simply used 2 MUXs being controlled by our ESP 32 with their outputs feeding into the ADC. Each MUX had two sensors. This test allowed us to demonstrate that the core components of our system would work without the need to build the entire array of 64 sensors.

## 9.1 DRV5053VA

To test our Hall sensors we first soldered them to a break out board that would allow us to use them on the breadboard. The package for all of our components was made for PCB mounting making the breakout boards a necessity for early testing. We started by setting Vcc to 3.3 volts and grounding the sensor. Then we put an LED at the output of the sensor so that we would be able to observe the light changing as we moved a magnet around the sensor. This first attempt was unsuccessful because the current produced by the sensor was much lower than what was necessary to turn the LED on. We solved the issue by using a transistor with the base connected to the output of the sensor. This allowed the sensor to toggle the transistor and allow the LED to turn on. In later testing we utilized a tesla meter provided by Dr. Weeks in the senior design lab to calibrate the sensors to get proper readings. Through our testing we were only able to identify only one dead sensor on our array board.

## 9.2 STM32H735G-DK

After programming the functionality for controlling the MUX, we tested the output of the GPIOs using LEDs to demonstrate that the 4 GPIO pins which were being used for controlling the select lines of the MUX light up each successive LED with twice the delay of the previous one. This functions as a binary counter which confirmed that our GPIO lines would function properly as select lines. Later testing would lead us to abandon the STM32 as it proved too difficult to make a graphics driver.

## 9.3 CD74HC4067M96

To use our MUX on the breadboard we utilized another breakout board. We connected the select lines to their respective pins and grounded the enable pin to keep the MUX on. We used LEDs in parallel with the select lines and slowed down the select line switching so that we could visualize where the MUX was in its switching. We placed an LED at the output of the MUX so we could see when outputs were coming through. Then we connected our power rail to two of the MUX channels so that they would turn on the LED when they were cycled through. With this we were able to confirm that the MUX was being controlled by the MCU and was able to cycle through its channels.

## 9.4 Display

The LCD screen was the most crucial part of this demonstration because it allowed us to view all of the components working together in one place. We set up the screen so that we would be able to display all the different channels from the MUX at once and have them update at 10 frames per second. The output of the MUX was fed into one of the ADC channels on the MCU, storing its outputs in an array which would update the screen once the MUX had finished cycling its channels. The two Hall sensors we had on breakout boards were connected to two of the channels on the MUX. We were able to observe the values from the hall sensors changing on the screen as we moved a magnet around the board.

## 9.5 ADC Troubleshooting

When we ran our first test of the screen and full board integration we ran into an issue with our ADC values. When we held a magnet with a south polarity too close to a sensor then the ADC value would hit its max of 4095 and roll back over to 0 and continue up from there. We initially tried to fix the issue with software but didn't have much luck. Next we tried a hardware fix, the rationale being that we would hit the max ADC value at around 1.5V being outputted from the sensor. The ADC had a reference of about 3.3V meaning we were getting about twice the expected ADC value because the ADC was outputting as if that 1.5 was its maximum reference voltage. This would mean the maximum output of the sensors, 2V, would read around 4V. To solve this we decided to try raising the reference voltage by changing our 3.3V input to a 5V input to our board. The problem with this is that a 5V reference would also mean a 5V MISO which would damage the ESP. Our solution was using a voltage divider which we first tested on a breadboard to change our 5V MISO to a 3.3V MISO. Our attempts to do this on a breadboard proved successful and we were able to resolve the issue but when we attempted to solder resistors we ran into issues and couldn't get the system to function the same way it was when on the breadboard. At this point we turned to Dr. Weeks for help who troubleshooted our system and concluded that the issue was definitely on the software side. After this we were able to identify that we had not set up SPI properly which was leading to the improper values. After we fixed the set up the system functioned as expected.

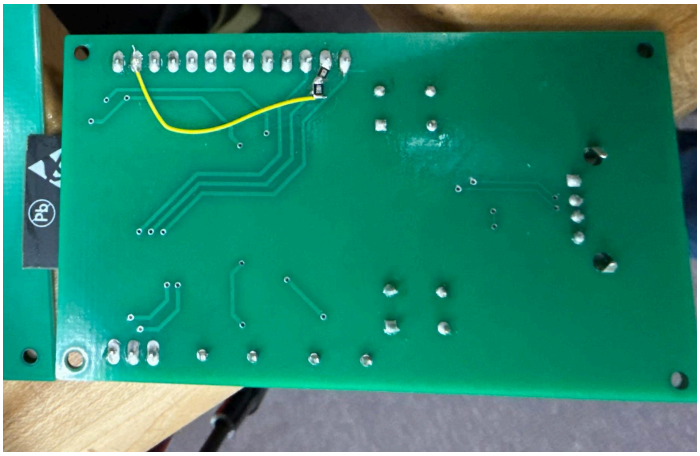


*Figure 9.5.1 voltage divider soldered onto board*

## 9.6 Battery Management System Troubleshooting

During initial testing of the Battery Management PCB, we observed unexpected behavior when supplying 12V through the DC jack. Instead of charging, we found that the batteries began rapidly discharging. Also, the status LEDs on the board remained illuminated even after external power was disconnected. This indicated a critical error somewhere in the circuit. In trying to solve the issue, each passive component on the board was probed individually to verify correct resistance, inductance, and capacitance values. During this process, the capacitor connected to the BATT pin of the MP2615 was found to be reading as a short. This strongly suggested that the MP2615 had been internally damaged and could no longer regulate power correctly. The shorted BATT pin explained the reverse current flow and the battery depletion under charging conditions.

As a result, we fabricated a new Battery Management PCB, replacing the MP2615 and carefully validating the supporting components. Upon powering the new board, the batteries began to charge as expected, confirming that the original IC had failed. However, we ended up discovering a new issue. When the 12V input was disconnected, the voltage at the input terminal lingered at 18V. This behavior indicated that the input capacitor remained fully charged, with no discharge path to ground. This caused the batteries to backfeed through the IC, leading to the unexpected voltage. To address this, we added a pull-down resistor to the input side, providing a path for the capacitor to safely discharge once external power was removed. This resolved the voltage hold-up problem

and allowed the system to function reliably during and after charging. From that point on, the Battery Management PCB successfully operated as a standalone charging solution for the 2-cell lithium-ion battery configuration.


*Figure 9.6 Battery Management PCB with Pulldown Resistor*

## 9.7 5V Regulator Adjustments

During testing of the 5V Regulator PCB, we initially used the fixed 5V output version of the LM2679 switching regulator to power the Raspberry Pi. However, we quickly discovered that the Pi is highly sensitive to input voltage and displays under-voltage warnings if the supply drops even slightly below 5V. These warnings can lead to performance issues or force the Pi to shut down unexpectedly. To prevent this, we determined that the regulator needed to supply a voltage slightly above 5V to maintain stable operation under all load conditions.

Rather than redesigning and manufacturing a new PCB, we came up with a practical solution using our existing PCB. We replaced the fixed-output LM2679 with the adjustable version and added an external voltage divider at the output using through-hole resistors. This modification allowed us to raise the output voltage to approximately 5.3V, ensuring that the Raspberry Pi received a stable input and eliminating all under-voltage warnings. The adjusted regulator performed reliably during all subsequent tests, and no further power-related

issues were observed with the Pi. This solution proved effective while preserving the existing board layout and minimizing cost and turnaround time.


*Figure 9.7 LM2679-ADJ Buck Converter with Voltage Divider*

# 10: Administrative Content

## 10.1 Bill of Materials

Table 10.1: Bill of Materials

| Quantity | Component | Cost |
|---|---|---|
| 64 | Hall-effect sensor | $24.44 |
| 1 | ADC | $4.73 |
| 4 | MUX | $2.32 |
| 2 | Raspberry Pi | $180 |
| 1 | 3.3 V regulator | $4.32 |
| 1 | 5 V regulator | $6.60 |

| | | |
|---|---|---|
| 1 | Battery management chip | $3.19 |
| | Miscellaneous components | $90.94 |
| | PLA for Housing | $20.23 |
| | Obsolete Parts | $129.38 |
| | Extras | $44.41 |
| | Shipping, taxes, and tariffs | $38.20 |
| Total | | $548.76 |

We exceeded the estimated budget for this project, but there are costs that could be eliminated upon the construction of another system. One of these is the spare parts we ordered for safety. Additionally, throughout the creation of our project, multiple components that we thought we would use ended up getting discarded for one reason or another. On top of all that, parts were ordered during the height of the Trump administration tariffs. While all of these costs were necessary for the development and continued design process of our system, excluding them from the overall budget calculation leaves us with a figure of $336.77.

## 10.2 Project Milestones

Table 10.2.1: Senior Design I Milestones

| Item | Start Date | End date | Duration |
|---|---|---|---|
| Brainstorming | 1/10/25 | 1/16/25 | 1 week |
| Divide and Conquer (D&C) Report | 1/16/25 | 1/23/25 | 1 week |
| Divide and Conquer (D&C) Group Meeting | 1/27/25 | 1/27/25 | 1 day |
| Update and upload D&C report onto group website | 2/1/25 | 2/6/25 | < 1 week |

| | | | |
|---|---|---|---|
| Parts selection | 2/7/25 | 2/15/25 | 1 week |
| Acquire parts | 2/7/25 | 3/10/25 | Around 1 month* |
| Initial PCB design | 2/16/25 | 3/2/25 | 2 weeks |
| Breadboard prototyping | 3/11/25 | 4/14/25 | 1 month |
| Midterm Report | 3/16/25 | 3/23/25 | 1 week |
| Midterm Report Group Meeting | 3/26/25 | 3/28/25 | 1 day |
| Update and upload midterm report onto group website | 3/28/25 | 4/3/25 | 1 week |
| SD1 Final Report & Mini Demo Video | 4/15/25 | 4/23/25 | 1 week |

Table 10.2.2: Senior Design II Milestones

| **Item** | **Start Date** | **End date** | **Anticipated Duration** |
|---|---|---|---|
| Finalize Design and Parts Inventory | 05/12/2025 | 05/14/2025 | 2 days |
| PCB Assembly | 05/14/2025 | 05/28/2025 | 2 weeks |
| Finalize Firmware Development | 05/28/2025 | 06/25/2025 | 4 weeks |
| Integration Testing | 06/25/2025 | 07/02/2025 | 1 week |
| System Optimization | 07/02/2025 | 07/09/2025 | 1 week |
| Final Testing and Validation | 07/09/2025 | 07/16/2025 | 1 week |
| Final Documentation and Presentation | 07/23/2025 | 07/30/2025 | 1 week |

# 11: Conclusion

The goal of H.A.L.O. was to create a system capable of visualizing magnetic fields. We are pleased to say that this goal was not only achieved but surpassed. Magnetic field strength values were captured from 64 Hall-effect sensors arranged in an 8x8 grid. These values were combined via 4 MUXs and converted to digital values by an ADC. These values were then transmitted to a MCU that interpreted the values. The MCU then communicated with the display driver to control the LCD display which the user ultimately interacted with. The user could visualize the magnetic field in two different modes that both served a niche purpose and complemented each other to provide an intuitive and convenient way of visualizing magnetic fields.

All of us are pleased with how our project turned out and that we didn't have to compromise on any of the features or specifications we worked on; however, the greatest takeaway from this project has been the knowledge and experience gained during the dual semester endeavor. The technical knowledge that goes into designing and ordering the components that make up the system; the experience of testing, troubleshooting, and integrating a system; and the experience of working in a team, compromising, standing your ground, and holding yourself and others accountable are all invaluable.

None of this would have been possible without certain individuals that we would like to take the following space to thank. Group 18 would like to thank Dr. ChungYong Chan for his continual guidance and advice as our group's advisor; Dr. Arthur Weeks for the countless hours and indispensable knowledge he generously provided to our group; Oli Marquez for the gracious lending of his 3D printer and time; and our review committee: Dr. Piotr Kulik, Dr. Jaesung Lee, and Dr. Enxia Zhang who we would like to extend an extra thanks towards for her role in shaping our project with her valuable input and time.

# 12: References


[1] Jansen, Peter. "A Magnetic Imager Tile." Hackaday.io, 29 July 2017, https://hackaday.io/project/18518-iteration-8/log/64376-a-magnetic-imager-tile.

[2] Cai, Jiangwei, et al. "A High-Resolution Magnetic Field Imaging System Based on the Unpackaged Hall Element Array." *Applied Sciences*, vol. 14, no. 13, 2024, p. 5788. https://doi.org/10.3390/app14135788.

[3] S. Akka and M. Missous, "Design of a portable, two dimensional magnetometer, using 2 dimensional quantum Hall effect sensor array, optimised for low magnetic field applications," The 10th IEEE International Symposium on Electron Devices for Microwave and Optoelectronic Applications, Manchester, UK, 2002, pp. 177-182, doi: 10.1109/EDMO.2002.1174951. keywords: {Sensor arrays;Magnetometers;Hall effect devices;Design optimization;Magnetic sensors;Voltage;Data acquisition;Spinning;Magnetic fields;Magnetic switching},

[4] N. Fischer, J. Kriechbaum, D. Berwanger and F. Mathis-Ullrich, "Compliant Hall-Effect Sensor Array for Passive Magnetic Instrument Tracking," in IEEE Sensors Letters, vol. 7, no. 3, pp. 1-4, March 2023, Art no. 2500404, doi: 10.1109/LSENS.2023.3250971.
keywords: {Sensor arrays;Sensors;Magnetic sensors;Bending;Target tracking;Magnetic flux;Shape;Magnetic sensors;passive magnetic sensing;flexible sensors;Hall-effect sensors;magnetic tracking;self-sensing;wearable sensors},

[5] McGill University, "Analog to Digital Conversion - Virtual Lab on Biomedical Signals," McGill University. [Online]. Available: https://www.medicine.mcgill.ca/physio/vlab/biomed_signals/atodvlab.htm. [Accessed: 22-March-2025].

[6] Spectrum Instrumentation, "ADC and Resolution," [Online]. Available: https://spectrum-instrumentation.com/support/knowledgebase/hardware_features/ADC_and_Resolution.php.
[Accessed: 22-March-2025].

[7] G. M. Smith, "*Types of ADC Converters [Updated 2024]*," *Dewesoft Blog*, 18 Feb. 2025. [Online]. Available: https://dewesoft.com/blog/types-of-adc-converters.

[8] “Magnetic Resonance Imaging.” *Wikipedia*, en.wikipedia.org/wiki/Magnetic_resonance_imaging. Accessed 22 Apr. 2025.

[9] “HallinSight® 3D Hall Magnetic Field Camera.” *Metrolab Technology SA*, www.metrolab.com/hallinsight. Accessed 22 Apr. 2025.

[10] “Types of Batteries.” *Monolithic Power Systems*, www.monolithicpower.com/en/learning/mpscholar/battery-management-systems/introduction-to-battery-technology/types-of-batteries. Accessed 22 Apr. 2025.

[11] “LFP Battery Energy Density.” *Ecotree Lithium*, ecotreelithium.co.uk/news/lifepo4-battery-energy-density. Accessed 22 Apr. 2025.

[12] “What Is the Energy Density of a Lithium-Ion Battery?” *Valley Industrial Trucks*, valleyindustrialtrucks.com/what-is-the-energy-density-of-a-lithium-ion-battery/#:~:text=LFP%20batteries%20have%20a%20high,metallic%20backing%20for%20the%20anode. Accessed 22 Apr. 2025.

[13] “An Introduction to Batteries: Components, Parameters, Types, and Chargers.” *Monolithic Power Systems*, www.monolithicpower.com/learning/resources/an-introduction-to-batteries-components-parameters-types-and-chargers. Accessed 22 Apr. 2025.

[14] “LiFePO4 Cell Voltage – Importance and Impact.” *EVLithium*, www.evlithium.com/Blog/lifepo4-cell-voltage-importance-and-impact.html. Accessed 22 Apr. 2025.

[15] “USB History & Timeline.” *USB.org*, www.usb.org/history. Accessed 22 Apr. 2025.

[16] “USB Wiring Diagram.” *PinoutGuide.com*, pinoutguide.com/Wiring/usb_wiring.shtml. Accessed 22 Apr. 2025.

[17] “USB-A vs USB-C: What's the Difference?” *Cable Matters*, www.cablematters.com/Blog/USB/usb-a-vs-usb-c. Accessed 22 Apr. 2025.

[18] “Micro-USB vs USB-C Connectors.” *TechSpot*, www.techspot.com/article/2135-usb-c-vs-micro-usb/. Accessed 22 Apr. 2025.

[19] “Micro USB Connector, B Female, 5 Pin SMD.” *Amphenol ICC*, www.amphenol-icc.com/micro-usb-b-female-connector.html. Accessed 22 Apr. 2025.

[20] “USB Type-C Explained.” *How-To Geek*, www.howtogeek.com/211843/usb-type-c-explained/. Accessed 22 Apr. 2025.

[21] “USB 2.0 Type-A (Male) Connector.” *Mouser Electronics*, www.mouser.com/ProductDetail/xxx. [Note: Please provide a specific URL.]

[22] Texas Instruments. *PCB Layout Guidelines for Li-Ion and Li-Polymer Battery Charger Applications.* 2010, www.ti.com/lit/an/snva558/snva558.pdf. Accessed 22 Apr. 2025.

[23] Texas Instruments. *Understanding and Applying Current-Mode Control Theory.* 2010, www.ti.com/lit/an/snva559c/snva559c.pdf. Accessed 22 Apr. 2025.

[24] *Efficiency and Power Characteristics of Switching Regulators.* Analog Devices, www.analog.com/media/en/technical-documentation. Accessed 22 Apr. 2025.

[25] “What Is a Multi-Chip Module Assembly?” *Tempo Automation*, www.tempoautomation.com/blog/what-is-a-multi-chip-module/. Accessed 22 Apr. 2025.

[26] “When Should I Use Bare Board Testing?” *Advanced Circuits*, www.4pcb.com/bare-board-testing.html. Accessed 22 Apr. 2025.

[27] “What Is Printed Circuit Board Assembly (PCBA)?” *Bittele Electronics Inc.*, www.bittele.com/pcba.html. Accessed 22 Apr. 2025.

[28] “Applying IPC-2221 Standards in Circuit Board Design.” *Tempo Automation*, www.tempoautomation.com/blog/ipc-2221-standards/. Accessed 22 Apr. 2025.

[29] *CTI: The Comparative Tracking Index Test.* 3M, multimedia.3m.com/mws/media/xxx. [Please provide the exact link or document title.]

[30] “Senior Design Lab Equipment.” [Institution/Website Name], [Provide link if applicable].

[31] “ChatGPT Sets Record for Fastest-Growing User Base.” *Analyst Note*, [Publisher], [Link if available]. Accessed 22 Apr. 2025.

[32] “Grok 3 vs ChatGPT vs DeepSeek vs Claude vs Gemini.” *Medium*, medium.com/@xxx/grok3-vs-chatgpt-vs-claude-vs-gemini. [Provide exact URL.]

[33] “ChatGPT Pricing.” *OpenAI*, openai.com/pricing. Accessed 22 Apr. 2025.

[34] “DeepSeek Pricing: How Much Does It Cost & Is It Worth It in 2025?” *AI Price Watch*, www.aipricewatch.com/deepseek-pricing-2025. Accessed 22 Apr. 2025.