Suction Sense Project

ECE 445 Design Document- Fall 2025

Project #19

Jeremy Lee, Suleymaan Ahmad, Hugh Palin

Professor: Arne Fiflet

TA: Lukas Dumasius

Contents

Introduction:

1. Problem

Currently, suction systems in hospital operating rooms are left running unnecessarily for nearly 35% of their total runtime, including periods such as overnight when no surgeries are taking place. This results in wasted energy, wear overtime on expensive vacuum equipment, and higher maintenance demands. Without any system to detect or alert staff when suction is left on, hospitals face unnecessary electricity consumption and shortened equipment lifespan. This creates a huge inefficiency that scales across entire healthcare systems.

The financial and environmental impact of this waste is significant. Leaving suction on overnight alone contributes to approximately 8 billion kilograms of CO_2 emissions globally every year. This efficiency can cause hospitals to incur significant additional costs including: replacement vacuum systems that range from $100,000 to $750,000, filters that cost $2,500 to $10,000, and annual oil changes that add another $8,000. On top of that, hospitals spend an estimated $30,835 each year just on electricity for their vacuum systems. Together, these demonstrate the urgent need for a solution that minimizes unnecessary suction runtime, reduces costs, and lessens environmental impact.

2. Solution

To tackle this problem, we propose a combined hardware and software solution designed to monitor and reduce unnecessary suction usage in operating rooms. At a high level, the system consists of two parts: pressure sensors installed on vacuum systems in each operating room and a software interface that collects real-time suction data and compares it with the operating room schedule.

To implement this system, we will design a custom PCB that integrates a microcontroller and supporting components to capture suction pressure data and transmit it over Wi-Fi. A flow sensor coupled with a BMS will accomplish this. A Raspberry Pi module will receive and store the incoming data, serving as the central hub for processing. This module will also host the software component, which connects to the hospital's internal network, via Epic, to access operating room schedules. By cross-referencing suction activity with scheduled procedures, the software can automatically identify where there is unnecessary suction use. A user interface displayed on a raspberry pi touch screen will then present this information in a visual format, displaying the status of each operating room and highlighting rooms with unnecessary suction in red. The display will be placed in a central location, ensuring that medical staff can easily monitor system status and respond promptly.

3. Visual Aid

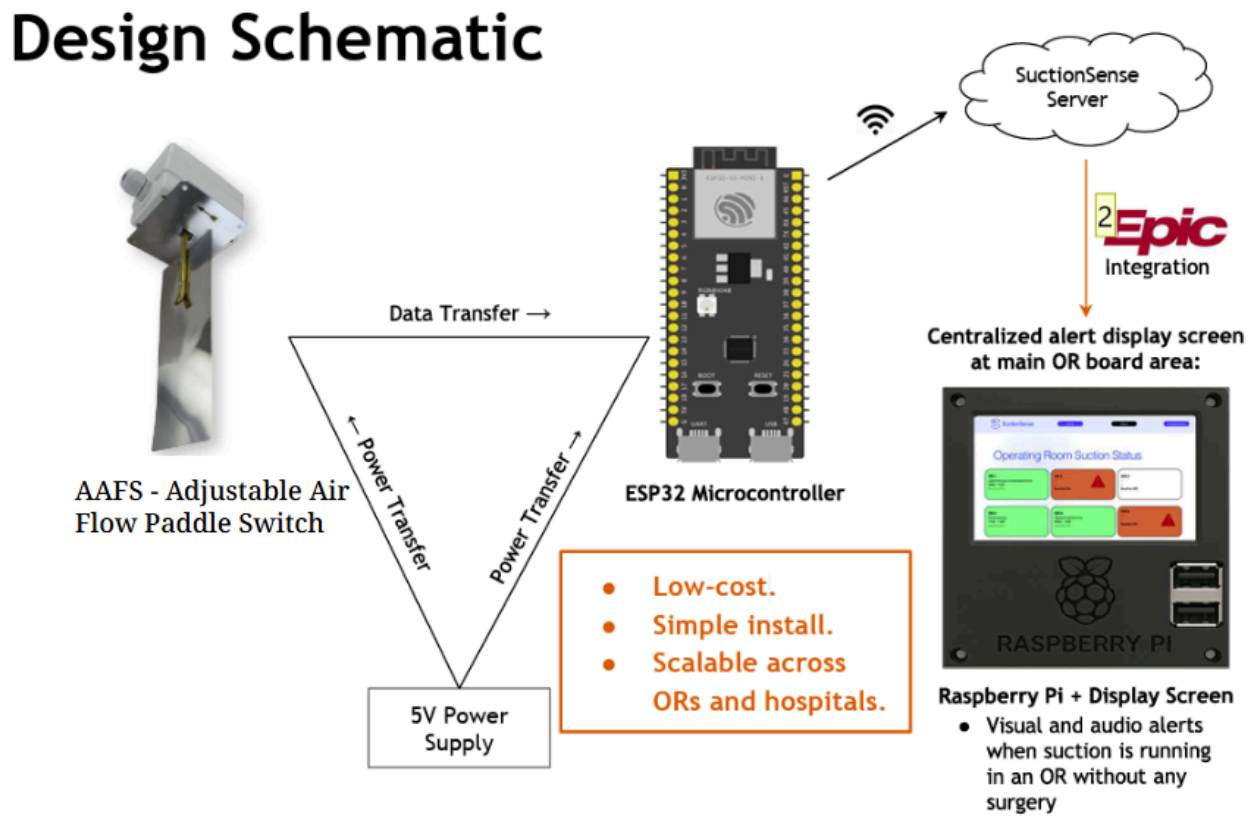


Figure 1 Addendum: This figure is from the Suction Sense class presentation [2]. Note we have moved to using a flow sensor, not a pressure transducer.

4. High-level requirements

- The system must be able to handle input from at least 8 operating rooms simultaneously without data loss or low performance
- The user interface must refresh visual indicators (e.g., red highlighting, OR status) within 10 seconds of identifying unnecessary suction usage
- The hardware sensors and transmission module must support continuous operation for 24 hours without data transmission failures.

Design:

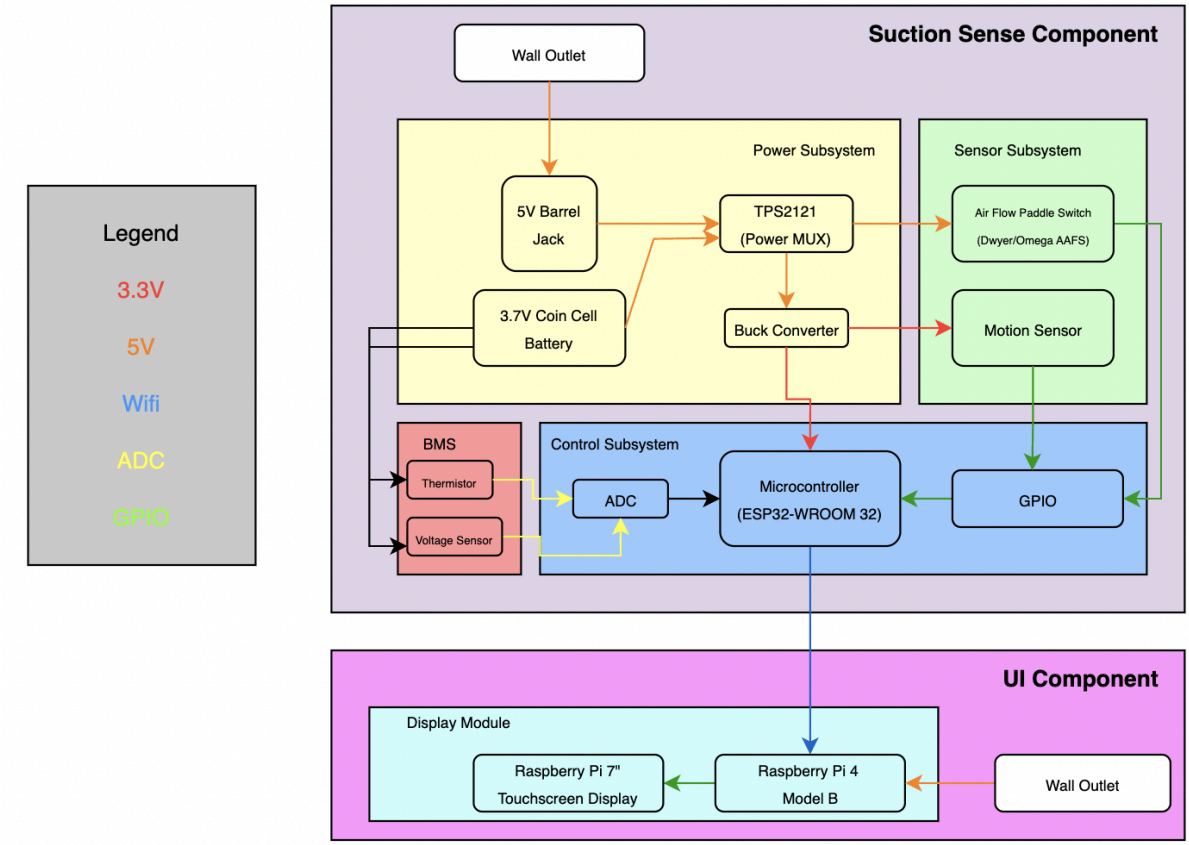


Figure 2: System Block Diagram

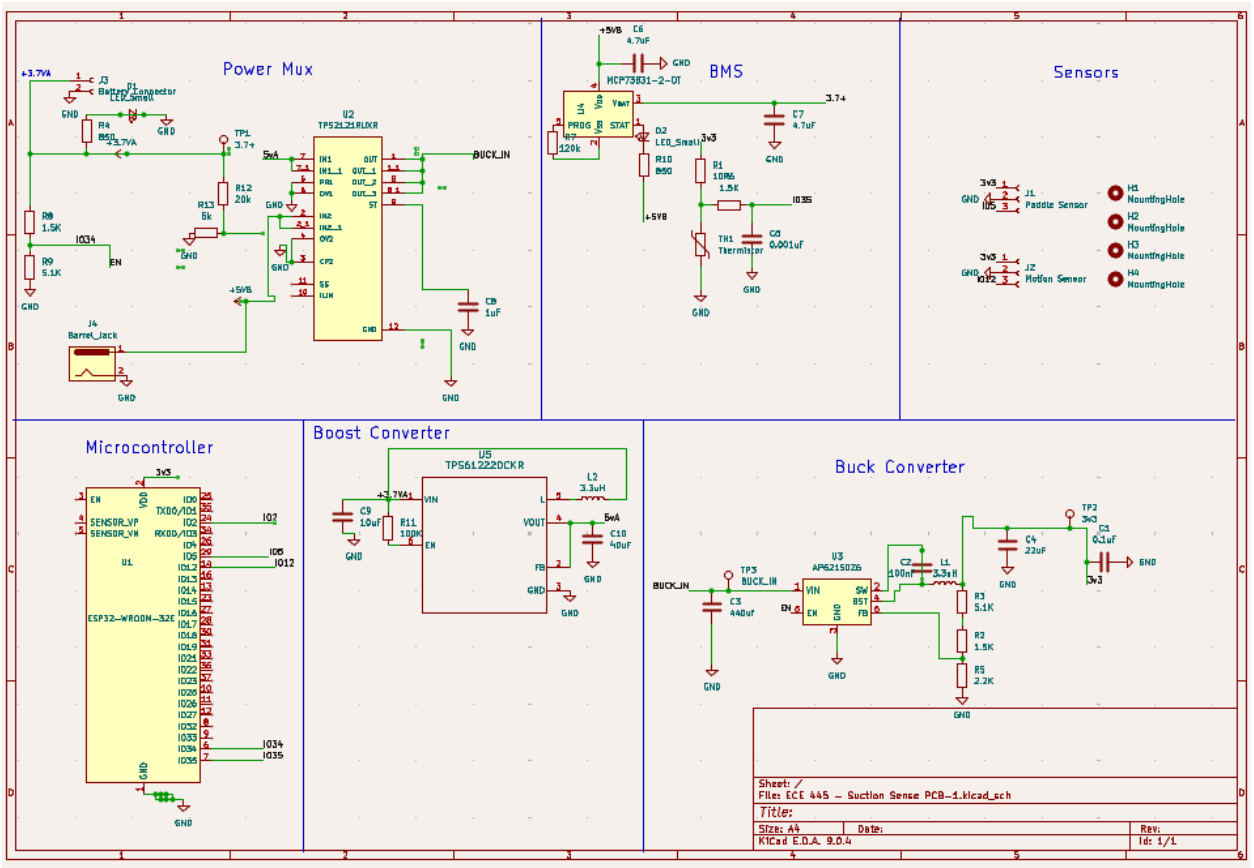


Figure 3: Suction Sense Schematic

Subsystem Overview:

Sensors Subsystem:

Our first subsystem is responsible for measuring the pressure of the vacuum which will be used to monitor real-time suction. It works by converting vacuum flow rate into an electrical signal readable. We will be using the AAFS ADJ Air Flow Paddle Switch for its compatibility with medical suction ranges, compact design for easy integration, and reliability in continuous-use environments. The sensor's analog output provides a simple and accurate way to track suction status with minimal additional circuitry. The output of our flow rate sensor will then be stepped down to a safe operating voltage and imputed into our MCU. Another sensor we will be including is an Adafruit MINI PIR MOT motion sensor to monitor if anyone is in the room during OR operation. This is necessary in case we do not gain access to Operating Room data due to HIPAA or data compliance issues, so we can still determine if an operation is being conducted. We chose the Adafruit motion sensor as it has the output voltages for our MCU as well as the input voltage as our MCU.

The AAFS air flow paddle switch functions as a simple on/off indicator of air movement within the vacuum line[2]. As shown in the diagram, when sufficient air flow is present, the switch makes continuity between terminals 1 and 2. When no air flow is detected, continuity shifts to terminals 1 and 3. By monitoring which terminals are connected, our system can determine whether suction is active or lost. This binary signal can then be read by the MCU to verify real-time vacuum operation and trigger alerts if air flow drops below the required threshold. By supplying a standard 3.3 V signal to terminal 1, the presence of air flow can be detected by monitoring terminal 2 to our data pin. When the data pin reads a high voltage it indicates that air flow is present.

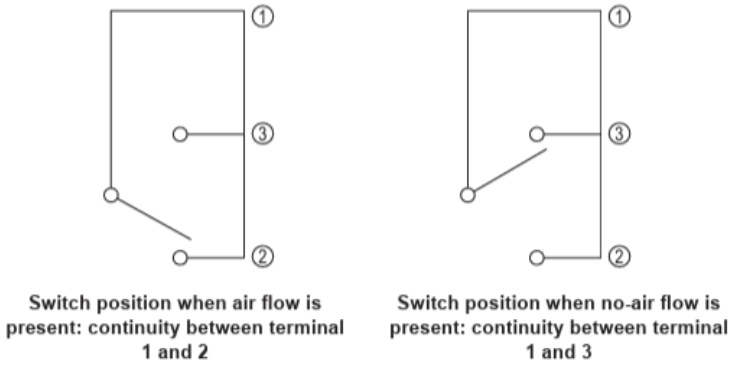


Figure 4: AAFS Manual Switch Diagram

To use the Adafruit MINI PIR motion sensor, the three pins are connected to power, ground, and a digital input on the MCU[1]. The sensor outputs a logic HIGH of 3.3 V when motion is detected. The onboard jumper allows switching between retriggering modes, while small potentiometers can adjust sensitivity and timeout, helping reduce false triggers and tune detection range. This will help us tune the motion sensor.

Requirements	Verification
The flow rate sensor must return the correct state of flow with 99% accuracy.	When producing a vacuum on our testing stand, connect a DMM between terminal 2 (output) and terminal 1 (ground) of the AAFS paddle switch to measure its logic state. Compare the measured output voltage to the known on/off state of the vacuum pump over repeated cycles, recording results in a data table. Verification is achieved if the switch output matches the pump state in at least 99% of trials.

The flow rate sensor requires a 3.3 V supply, so we will supply it with a stable 3.3 V rail while conditioning its output for the MCU.	We will use our DMM to measure input voltage being a stable 3.3 V from our rail. Then measure the output voltage of our switch on the data terminal, ensuring it is still a stable 3.3 V when the flow rate is changing. We will change flow rate from none to maximum hospital suction.
The motion sensor shall be supplied with 5 V and will present a 3.0 V output signal compatible with the MCU input.	Check the power rail the motion sensor is supplied with is 5 V, and its output pin shall be monitored with a DMM connected between signal and ground. The measured output voltage will be compared to observed room motion (person entering/leaving) over repeated trials, and results will be recorded in a data table. Verification is achieved if the output consistently switches to ~3.0 V during motion.

Power Subsystem:

The power subsystem is responsible for delivering continuous and reliable power to our board so that our electronic components such as our MCU and sensors can operate as intended. To enhance the reliability of this system, we decided to create a power system that made use of two power inputs, A 3.7v signal from a rechargeable 2032 Lithium Ion as well as 5v input form a DC barrel jack connector that will connect to a wall outlet. In order to make use of both inputs, we connect them to a Power Mux, allowing us to effectively operate in all of 3 different scenarios.

Scenario 1: the 5v barrel jack is connected and the 3.7v coin cell is discharged, where the output is 5v.

Scenario 2: The 3.7v coin cell is charged and the barrel jack is not connected where the output is the 3.7v signal.

Scenario 3: both the barrel jack is connected and the 3.7v coin cell is charged, where the output is the 5v signal since it is a higher voltage.

The only scenario where our Subsystem has no power is when the coin cell is discharged and the barrel jack is not connected.

To ensure seamless operation, we choose our Power Mux with the following requirement: at most a 10us switching time to avoid a transient that would brownout our MCU. This is why we choose the TPS2121[10], which is a Power Mux that is manufactured by Texas Instruments that makes use of an Ideal Diode O-ring mechanism to seamlessly transition between two power sources. Figure 5 below contains the specification information from the datasheet of this device, demonstrating that it more than fits our requirements.

Table 10-3. Automatic Switchover Design Requirements

DESIGN PARAMETER	SPECIFICATION	DETAILS
IN1 Voltage	V_{IN1}	12 V
IN2 Voltage	V_{IN1}	5 V
Load Current	I_{OUT}	2 A
Load Capacitance	C_L	200 µF
Maximum Inrush Current	I_{INRUSH}	100 mA
Switchover Time	t_{SW}	TPS2120: 5 µs
Mode of Operation	Automatic Switchover	TPS2121: XCOMP

Figure 5: Power Mux Voltage Specifications[10]

In order to operate our MCU and Sensors we require a 3.3v signal, the signal that our Power Mux emits is 5V, which does not currently meet our needs. To resolve this we will make use of a regulator that can step our voltage down to a safe operable voltage. To do this we have decided to use an AP62150 synchronous buck converter that will take the inputted signal from the Power Mux and step it down using this device by configuring the Vout using the correct component values according to the table below.

9 Setting the Output Voltage

The AP62150 has adjustable output voltages, starting from 0.8V, using an external resistive divider. The resistor values of the feedback network are selected based on a design trade-off between efficiency and output voltage accuracy. There is less current consumption in the feedback network for high resistor values, which improves efficiency at light loads. However, values too high cause the device to be more susceptible to noise affecting its output voltage accuracy. R1 can be determined by the following equation:

$$\mathbf{R1 = R2 \cdot \left(\frac{VOUT}{0.8V} - 1\right)} \qquad \textbf{Eq. 8}$$

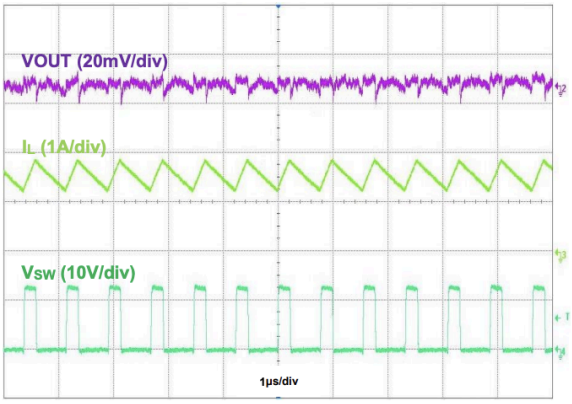
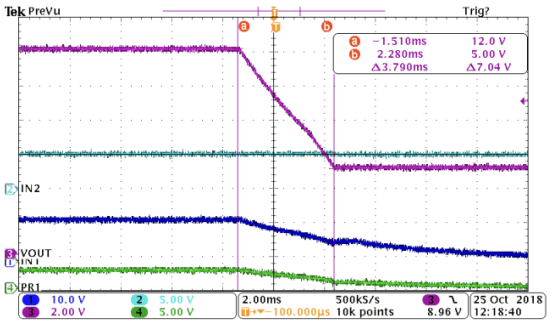
Table 1 shows a list of recommended component selections for common AP62150 output voltages referencing Figure 1. Consult Diodes Incorporated for other output voltage requirements.

Table 1. Recommended Component Selections

AP62150						
Output Voltage (V)	**R1 (kΩ)**	**R2 (kΩ)**	**L (µH)**	**C1 (µF)**	**C2 (µF)**	**C3 (nF)**
1.2	4.99	10	1.2	10	22	100
1.5	8.66	10	1.5	10	22	100
1.8	12.4	10	1.8	10	22	100
2.5	21.5	10	2.2	10	22	100
3.3	31.6	10	3.3	10	22	100
5.0	52.3	10	3.3	10	22	100

Figure 6: Buck Converter Voltage Specifications

Requirements	Verification

• The 3.3 V rail must remain within ±5% regulation to ensure stable MCU operation.	Operate board in standard operation with each power source and verify with waveform from data sheet  **Figure 16. Output Voltage Ripple, VOUT = 3.3V, IOUT = 1.5A**
• The power subsystem must seamlessly switch to battery power via the BMS when the external 5 V supply is removed, with no loss of operation.	Operate board with single power source and then disconnect power source to force power mux. Probe signals for Vin to Buck converter and Vout, and compare with datasheet to observe the transient by zooming in really close on time scale. Ensure operation is as expected per datasheet  **Figure 10-10. Automatic Switchover from IN1 to IN2**
• Buck converter must be able to step down 5 to 3.3v with load current <200ma	Measure current through Vout using a rogowski coil or current probe or through shunt resistor. Should be similar to iL in this graph from datasheet:

F

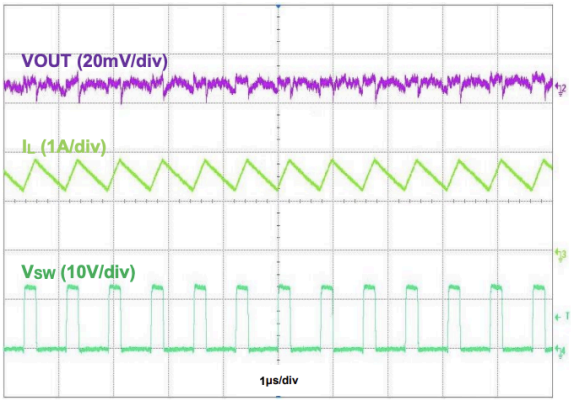


Figure 16. Output Voltage Ripple, VOUT = 3.3V, IOUT = 1.5A

BMS Subsystem:

One of our two power sources is a 3.7 V lithium-ion battery, which serves as a compact and energy-dense backup supply. To ensure safe and reliable operation under all conditions, a dedicated Battery Management System (BMS) has been integrated to monitor and protect the battery against unsafe states such as overcharging, over-discharging, and excessive temperature.

The BMS consists of three primary components: the MCP73831-2-OT Li-ion charging circuit, a 10 kΩ NTC thermistor, and a voltage sensing circuit. The MCP73831-2-OT manages the charging process when the 5 V barrel jack is connected, using a constant-current/constant-voltage (CC/CV) profile to safely charge the cell while providing built-in thermal regulation and charge termination. To ensure thermal safety, a 10 kΩ NTC thermistor is placed near the battery holder and connected to an MCU analog input, allowing continuous temperature monitoring and enabling the system to halt charging or discharging if the temperature falls outside safe limits. Additionally, a voltage divider circuit connected to the battery terminals feeds into another MCU ADC input to measure the battery voltage and estimate its state of charge, ensuring that the system can make intelligent power management decisions such as switching sources, issuing low-battery warnings, or preventing deep discharge. Together, these components provide a comprehensive and reliable BMS solution that ensures the lithium-ion battery operates safely, efficiently, and within optimal performance parameters.

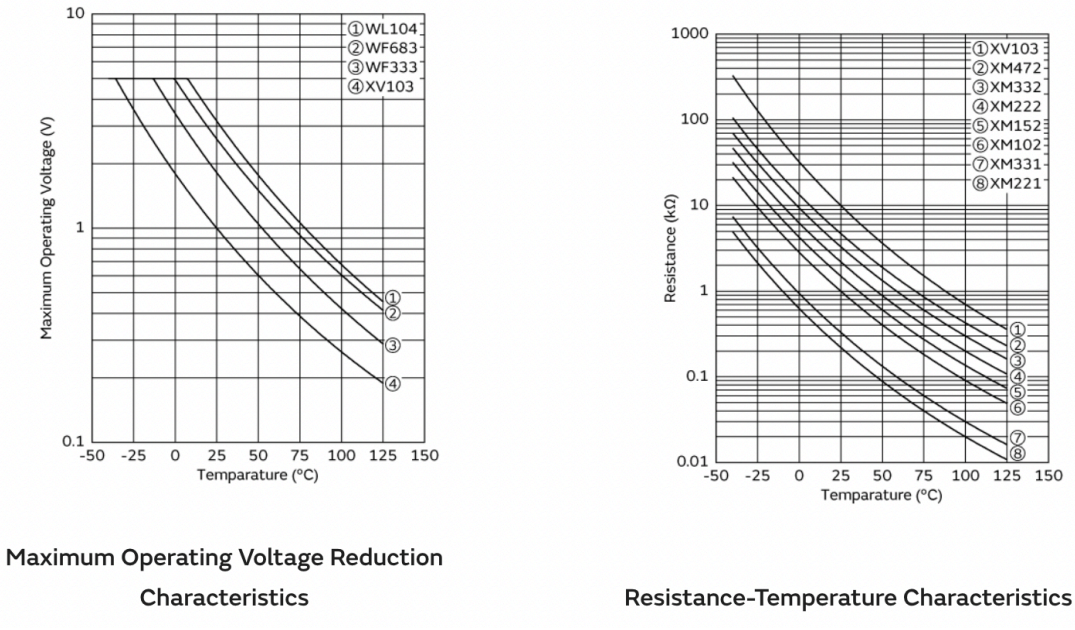
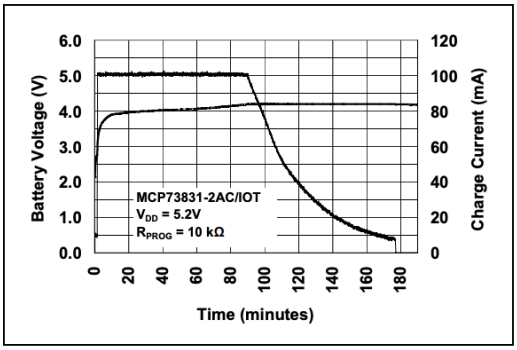


Figure 7: NTC Thermistor Voltage Characteristics[11]

Requirements	Verifications
The thermistor must be NTC 10 kohm to reduce power draw, but accurately provide temperature values.	Probe thermistor voltage on an oscilloscope and do a verification test where we artificially heat up the board and see if the voltage across is reflected.
Charging circuit should be able to accurately charge 3.7v li-ion battery.	Probe battery as it is in the charging state, and verify with expected charge profile form datasheet shown below. 
Voltage Divider should be able to be stepped down to voltage that can safely read by MCU and data can be streamed with less than 5% error.	Probe output of voltage divider to ensure voltage is within acceptable threshold to be read by GPIO pin.

MCU Subsystem:

The MCU subsystem provides system control and wireless connectivity for communication with the external software subsystem. It reads output data from the BMS to manage power switching between wall and battery sources for optimal energy efficiency, while monitoring sensor signals through its GPIO pins. We selected the ESP32-WROOM-32E-N4 for its powerful microcontroller core with integrated Wi-Fi, compact form factor, low cost, and proven reliability. Its built-in Wi-Fi enables direct communication with the Raspberry Pi and hospital networks without the need for an external BLE module, simplifying both hardware design and system integration[5].

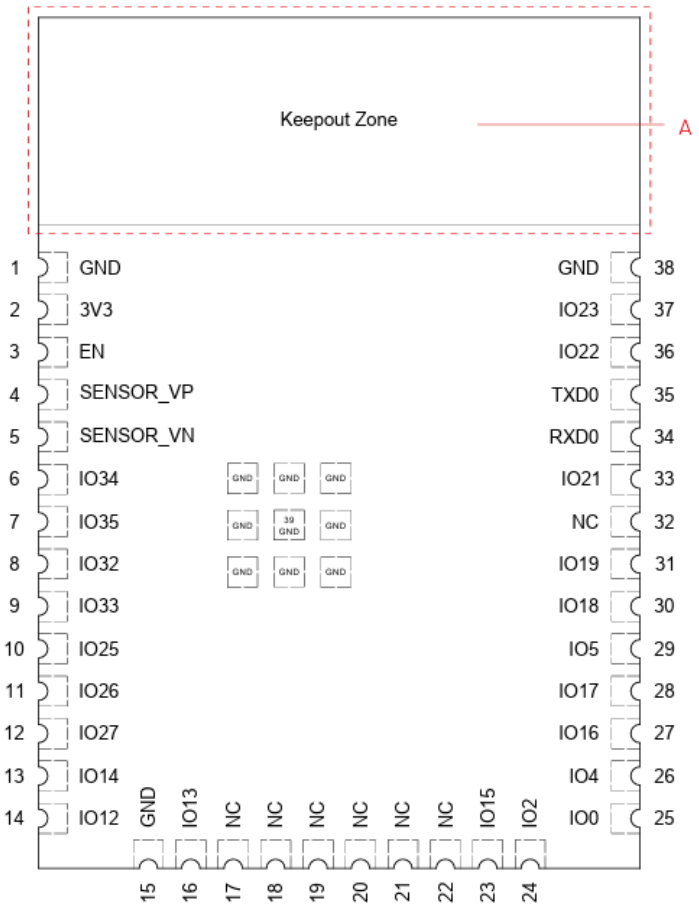


Figure 8: ESP32-WROOM-32E Pin Layout [5]

The first function of the MCU subsystem is to monitor and record key values from the Battery Management System and power system. The MCU will read inputs from the thermistor and voltage divider connected to the BMS to ensure the battery is operating within safe limits for temperature and charge. In addition, the MCU will track which power source is currently active by monitoring the output of the power mux, allowing it to log transitions between wall power and the battery supply for reliability and energy efficiency.

The MCU also monitors the AAFS flow sensor and the Adafruit motion sensor to detect instances of unnecessary suction within an operating room. While continuously polling and collecting this data, it streams the readings over Wi-Fi to the external software subsystem hosted on a Raspberry Pi, where the results are visualized through a custom user interface. To support this communication, an MQTT client will be implemented on the ESP32 to serve as a local message bus, enabling the MCU to periodically publish telemetry data that can be received and processed by the Raspberry Pi module.

Requirements	Verification
The MCU's wifi streaming must ensure that ≥99.9% of 1 Hz telemetry messages reach the Raspberry Pi over a 24-hour period. This ensures system reliability and minimal data loss during continuous operation.	Create a custom Python MQTT subscriber with timestamps and run the MQTT client for 24 hrs; count total vs. received messages on the Pi. Verify ≤86 missed of 86,400 total.
The MCU must be capable of sampling the flow rate sensor output with a correct reading within a +-1-2% margin of error.	Provide suction to the flow rate sensor and have the MCU sample the output and compare the recorded readings to the known state. Verification is achieved if the MCU's readings stay within ±2% of the known flow state in repeated trials.
The MCU must reconnect automatically if the WiFi link to the Raspberry Pi is lost.	Disable the WiFi link between the MCU and Raspberry Pi, then restore it. Verify that the MCU automatically reconnects without user input and resumes data transmission. Verification is achieved if reconnection occurs reliably in repeated trials of various times such as, 1 second, 5 seconds, and 10 seconds.

Raspberry Pi + LCD Display, and Software Subsystem:

The Raspberry Pi 4 Model B paired with the Raspberry Pi 7″ Touchscreen Display will serve as the central monitoring and alert system. The Raspberry Pi was chosen for its quad-core processing power, I/O support, and strong software ecosystem, which will allow us to easily integrate with the Epic scheduling system[6]. The 7″ touchscreen will allow the module to be mounted in the hallway, providing an interface that allows staff to quickly view operating room suction status, with clear color-coded indicators and alerts. This combination also enables both visual and audio notifications when suction is unnecessarily left on, ensuring staff can respond promptly.

The application will run on the Raspberry Pi and serve as the central hub for data processing and visualization. It will collect suction pressure readings from the ESP32 via its WiFi transceiver and compare this data against the hospital's operating room schedule retrieved through the Epic system. If integration with the Epic system is not available, the operating room schedules will be entered manually. A color-coded interface on the Raspberry Pi touchscreen will clearly show

which operating rooms are in use, whether suction is active, and show where suction has been unnecessarily left on.

To accurately receive and process data being streamed out from the ESP32, we will first run an MQTT broker service on the Pi to collect sensor telemetry data[3]. The ESP32 firmware will connect directly to the broker, sending flow and motion sensor data for processing. Next, the data will be inserted into a locally hosted SQLite database, that will additionally store operating room schedule data.

A lightweight Crow web application written in C++ serves as the interface between the SQLite database and the user interface[4]. It periodically queries the database and exposes an HTTP endpoint that returns each operating room's latest suction status in JSON format, while hosting a static dashboard on the Raspberry Pi's 7″ touchscreen. The dashboard polls this endpoint every few seconds and displays color-coded tiles(green for normal use, red for unnecessary suction, and gray for no suction) allowing staff to quickly identify rooms with suction left on. This setup provides an automated, real-time monitoring system that updates continuously with no manual input required.

Raspberry Pi 4 Model B features a high-performance 64-bit quad-core processor, dual-display support at resolutions up to 4K via a pair of micro HDMI ports, hardware video decode at up to 4Kp60, up to 8GB of RAM, dual-band 2.4/5.0 GHz wireless LAN, Bluetooth 5.0, Gigabit Ethernet, USB 3.0, and PoE capability (via a separate PoE HAT add-on). For the end user, Raspberry Pi 4 Model B provides desktop performance comparable to entry-level x86 PC systems.

This product retains backwards compatibility with the prior-generation Raspberry Pi 3 Model B+ and has similar power consumption, while offering substantial increases in processor speed, multimedia performance, memory, and connectivity.

The dual-band wireless LAN and Bluetooth have modular compliance certification, allowing the board to be designed into end products with significantly reduced compliance testing, improving both cost and time to market.

Figure 9: Raspberry Pi Description and Capabilities from Datasheet

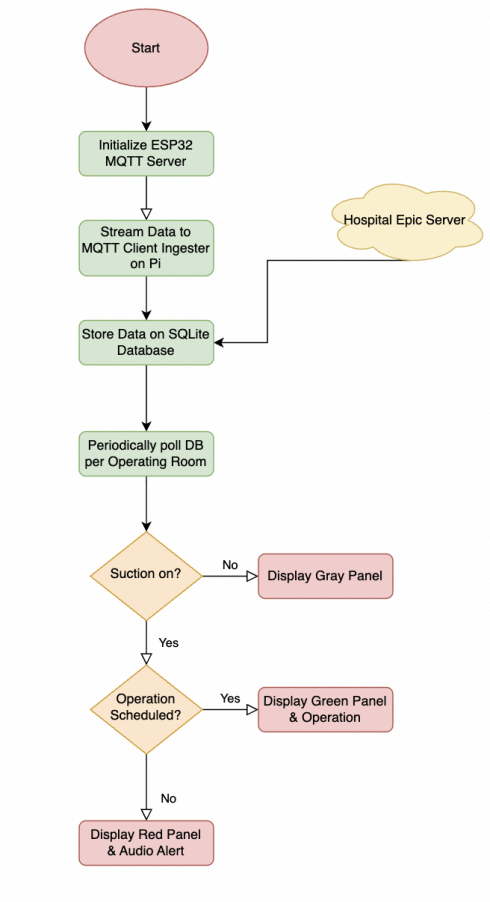


Figure 10 : Chart displaying the software application flow from our Suction Sense Module to Raspberry Pi

Requirements	Verification
• The Suction Sense UI should update room tiles every ≤ 2 s. This will ensure our system provides clinically useful "near-real-time" feedback.	• Publish via a change for an operating room to turn suction on from an off state. • Start stopwatch. • Observe tile color/text change from gray to green on the kiosk • Must update ≤ 2 s
• The system should continue to serve the UI even if the MQTT broker is down. This ensures that the UI won't crash even if one of the telemetry systems is down	• Stop the MQTT service • Ensure that the software application still returns last known statuses (HTTP 200)

Requirements	Verification
● Audible alert from Pi plays on transition to red and can be acknowledged	● Trigger red ● System plays alert sound once ● Tap “Acknowledge” (or button) to silence the alert for the configured window.

Cost and Analysis:

Tolerance Analysis

The most critical part of any system is what powers it, especially in this system with its intended applications in life saving environments. We believe the most susceptible part of design is the power system switch (power mux) that controls where we source the power for the entire board from. The intended purpose of this is to provide a dynamic response to when our primary power source (the 12v Lithium-ion battery) fails. The biggest issue that could arise from this is the potential transient voltage drop that would be seen across our power module as the LTC4412 switches from the battery to the power source, which could brown out our MCU.

The battery we are using is a 4000mAh 18650 lithium ion DC 12.6V this will feed into the LTC4412. When we enable the GPIO pin connected to the LTC4412 the primary FET will switch off, this will take approximately 13-22us, during this interval of switching the load will be supplied by Bulk capacitors the ESP32-wroom which operates at voltage 3.0-3.6, will brownout at voltages 2.9v-3.0v, to avoid this we will use 470uf Capacitances at the output(per data sheet recommendations) of the LTC4412 and the input of the Buck Converter(TPS629210), the TPS629210 is rated at a maximum load output of 1A, this means that during the switching time (22us worst case) we should not see an inrush current greater than this from the 470uf cap to account for our load this would cause a voltage drop across output of the buck converter. The buck converter operates at 85% efficiency (per the data sheet) so ideally input current to the MCU is 0.5A (Esp32 max load) x 3.3v(buck converter output)/(12v*0.85), which is equal to 0.15A. $0.15A < 1A$. In addition, using the capacitor differential model $C (dv/dt) = I$ we can solve for the voltage dip across the output of the buck converter since $C = 470uf$, $dt(worst) = 22us$, and $I = 0.15$. Rearranging our equation gives us $dv = I*dt / C$, so $dv = 27mv$, which is within tolerance because $3.3v - 27mv = 3.2v$ will not put us in the 2.9v -3.0v range on our MCU to brownout.

Cost Analysis

According to the UIUC ECE Department statistics, the average EE graduate makes close to $90,000 annually, which comes out to about $43.27/hour. Assuming that each of us works 6 hours a week to complete this project, we can estimate the cost of labor to be $43.27/hr x 2.5 x 6hrs/week x 14 weeks, which comes out to $9086.7 per group member or $27260.10 total. For the cost of materials, we estimate it will come out to $264.03 to build our entire system. Thus, combining both the cost of labor and cost for parts, the grand total for our project will be $27,525.13.

Parts List:

Description	*Manufacturer*	*Part Number*	*Quantity*	*Unit Cost*	*Total Cost*
Boost Converter	Texas Instruments	TPS61222	1	$1.22	$1.22
LCD Display	Raspberry Pi	SC1635	1	$81.25	$81.25
Raspberry Pi	Raspberry Pi	Raspberry Pi 4 Model B 2019	1	$63.88	$63.88
Airflow Sensor	Dwyer	AAFS	1	$106.05	$106.05
Motion Sensor	Adafruit	HC-SR312	1	$3.95	$3.95
MCU	Espressif	ESP32-WROOM-32E-N4	1	$4.84	$4.84
Thermistor	Murata	NCP21XV103J03RA	1	$0.16	$0.16
Buck Converter	Diodes Inc.	AP62150Z6-7	1	$0.31	$0.31
Power Mux	Texas Instruments	TPS2121RUXR	1	$2.37	$2.37
					Grand Total: $264.03

Figure 11: The Parts List and Cost for Suction Sense

Schedule

Week	Task	Person
October 12th-18th	Order parts for prototyping	Jeremy
	Software Planning & Environment Setup, Basic App	Jeremy, Everyone
	PCB Revision	Suley
	Machine Shop	Hugh
October 19th-25th	Power Subsystem bring-up and testing	Suley
	BMS Subsystem bring-up and testing	Suley
	MCU Subsystem & Firmware	Jeremy
	Sensor & Peripheral Subsystem	Hugh
	Test Wifi Message from ESP32 -> Pi	Jeremy, Hugh
October 26th-November 1st	Assemble PCB	Everyone
	2nd Breadboard Demo	Everyone
	Design and implement SQLite DB	Jeremy, Hugh
	PCB Testing	Suley
	Design Enclosure for PCB	Hugh

November 2nd-8th	3rd PCB Order	Everyone
	PCB Revisions	Suley
	Design UI for display, connect to Crow app	Jeremy, Hugh
	Connect backend Crow app to DB	Jeremy, Hugh
November 9th-15th	4th PCB Order	Everyone
	PCB Revisions	Suley
	Software Integration testing	Jeremy
	3D Print Encloser	Hugh
November 16th-22nd	Integration Testing	Everyone
	Final Software Testing	Everyone
November 23rd-29th	Fix Minor Bugs	Everyone
	Fall Break	Everyone
November 30th-December 6th	Final Demos	Everyone

Figure 12: Schedule for Suction Sense Group Members

Ethics and Safety:

Our project raises both ethical and safety considerations that we must address responsibly. Following the IEEE and ACM Codes of Ethics, we will avoid harm, be honest about our system's limitations, and protect privacy[8]. The system is not meant to be a real-time controller, meaning it will not be in charge of turning off the suction. Its purpose is purely advisory, and we will make sure users understand this through clear labeling and timestamp updates on the UI. We will also minimize privacy risks by only using room numbers and schedules, not any personal health information. Because our system connects to hospital scheduling software, we will also treat the project as subject to HIPAA rules. That means we will limit access to the minimum necessary information, only displaying the operating room numbers and type operation without any individual patient information in order to maintain privacy. We will also secure our connections with either encryption or an authorization layer to ensure only authorized staff can view or interact with the data.

On the safety side, we must ensure our hardware does not interfere with existing medical gas systems, so we will design it to attach non-invasively and comply with hospital facility rules. We will also follow basic lab safety practices during development, such as using PPE while soldering and keeping prototypes separate from live medical systems until properly reviewed. By keeping these ethical and safety principles outlined by IEEE and ACM in mind, we can deliver a system that helps hospitals save energy and reduce emissions without creating new risks for patients or staff.

References

[1] PIR Motion Sensor Created by Lady Ada, ADA Fruit, cdn-learn.adafruit.com/downloads/pdf/pir-passive-infrared-proximity-motion-sensor.pdf. Accessed 13 Oct. 2025.

[2] Inc., Alpha Controls & Instrumentation. “AAFS - Adjustable Air Flow Paddle Switch.” Alpha Controls & Instrumentation Inc., Dwyer Instruments, www.alphacontrols.com/AAFS-Adjustable-Air-Flow-Paddle-Switch/model/7411. Accessed 12 Oct. 2025.

[3] “1 Introduction.” MQTT Version 5.0, docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html. Accessed 12 Oct. 2025.

[4] “Crowcpp.” Crow, crowcpp.org/master/. Accessed 12 Oct. 2025.

[5] ESP32-WROOM-32E ESP32-WROOM-32UE Datasheet Version 1.9, www.espressif.com/sites/default/files/documentation/esp32-wroom-32e_esp32-wroom-32ue_datasheet_en.pdf. Accessed 13 Oct. 2025.

[6] Raspberry Pi 4 Model B Published February 2025 Raspberry Pi Ltd, datasheets.raspberrypi.com/rpi4/raspberry-pi-4-product-brief.pdf. Accessed 13 Oct. 2025.

[7] MCP14628 Data Sheet, ww1.microchip.com/downloads/aemDocuments/documents/APID/ProductDocuments/DataSheets/MCP14628-Family-Data-Sheet-DS20002083.pdf. Accessed 13 Oct. 2025.

[8] IEEE, "IEEE Policies - Section 7-8 - IEEE Code of Ethics," [Online]. Available: https://www.ieee.org/about/corporate/governance/p7-8.html. [Accessed: 18-Sep-2025].

[9] Chao, Sharon. “Suction Sense Lecture Proposal.” Carle Illinois College of Medicine, courses.grainger.illinois.edu/ece445/lectures/Fall_2025_Lectures/Lecture2/lecture_2_suction.pdf

[10] TPS212x Data Sheet
https://www.ti.com/lit/ds/symlink/tps2120.pdf?ts=1760390503520

[11] NCP21XV103J03RA Data Sheet
https://pim.murata.com/en-us/pim/details/?partNum=NCP21XV103J03RA