Noctisense

Group 1 Authors:
Colby Toner (EE), Edward Perez (PSE), Jose Ojeda Bermudez (CpE), Mason Miles (CpE)

Advisors:
Dr. Suboh Suboh, Dr. Paul Leisher

Reviewers:
Dr. Zakhia Abichar, Dr. Sazadur Rahman, Dr. Darren Hudson

Table of Contents

List of Figures

List of Tables

List of Equations

Chapter 1: Executive Summary

Noctisense was developed with one key insight in mind, that there exists a gap within the landscape of accurate, consumer-grade biometric sleep-tracking devices. Even more so at an affordable price point with a privacy-forward architecture. The demand for these devices has already been comprehensively established by the given market of smart watches, rings, and mattress covers amongst other solutions. The key advantage that makes Noctisense stand out amongst competitors is two-fold; non-intrusive design, and privacy-centric data practices. This system was designed and assembled entirely by a group of engineering students at the University of Central Florida using low-cost, readily available, or custom-fabricated components. The design of the Noctisense system consists of four primary technologies unified into one cohesive system.

24 GHz mmWave Radar: The primary means by which Noctisense tracks both restless movement during sleep as well as estimated respiratory rate. Using a discrete, purpose-build module to gather and process doppler shift data accurately and reliably this enables tracking of numerous health data points. The use of this discrete module also supports our team's philosophy of unobtrusive design given its ability to function through the housing of the system unseen by the end-user.

Infrared Thermal-Imaging: A key feature of the premier sleep-tracking devices is body temperature variation. With this data our system is able to calculate the time it takes for a user's body temperature to cool to a point indicative of each stage of sleep. Along with drawing insights to the quality of sleep throughout the night. Additionally, using the average ambient temperature of the room attained from the same sensor we are able to give recommendations on best practices to improve sleep quality by adjusting ambient temperature.

Audio Analysis: There are a plethora of valuable insights that can be given to a user of Noctisense through the tracking of auditory anomalies throughout the night. Including (but not limited to); snoring, sleep-apnea, sleep-talking, catathrenia, and many more. Insight of which is communicated through a simple and intuitive system for recording above-baseline level audio clips served to the user via our companion application.

Near-Infrared Optical Presence Detection: With any cohesive system, there is a reliable method for determining whether a user is present before initializing sleep monitoring. Noctisense accomplishes this using a continuous-beam near-infrared (NIR) optical subsystem that detects reflected infrared light from the user's upper torso. Rather than measuring distance, the system continuously monitors changes in the intensity of the reflected signal to determine whether the bed is occupied.

This project report in its entirety serves as a record of the development of the Noctisense system. Including the foundational principles and technologies, component selection, implementation, and limitations of the system. Motivations for our design as well as justifications are discussed first, followed by the systems hardware & software architectures, then design methodologies. Following this is the standards, constraints, and schematics/diagrams of the project's design in detail. Finally, following the conclusion of this report, there are appendices which outline administrative and reference material for the project.

Chapter 2: Project Description

2.1 Project Description

Developed by engineering students at the University of Central Florida, Noctisense is an affordable, privacy-forward, and completely non-intrusive consumer-grade biometric sleep-tracking system designed to bridge the gap in the current smart-wearable market. The system unifies four primary, low-cost technologies into a single cohesive architecture: a 24 GHz mmWave Radar to track restless movement and respiratory/heart rate via doppler shift data, an Infrared Thermal-Imaging sensor to monitor body and ambient temperature variations for sleep-stage insights, an Audio Analysis subsystem to detect and record nocturnal auditory anomalies like snoring or sleep apnea, and a Pulsed Direct Time-of-Flight sensor for reliable user presence detection. By utilizing a hidden, discrete hardware design and local data processing, Noctisense delivers robust, actionable health insights through a companion application while strictly prioritizing user privacy and comfort without the need for wearable devices.

2.2 Motivation & Background

Since the early 1970s with the use of actigraphy devices [1], both health professionals and watchful people have sought to gain insight into their health through the monitoring of sleep. While the market for such devices was initially restricted to medical practices and researchers. The early 2000s to the mid-2010s brought forth a boom of all consumer wearable devices thanks to the coinciding rise of the semiconductor industry [2]. Alongside the rise of wearable devices, the demand for individual health data tracking continues to grow year over year [3].

The most common variant of consumer devices used to measure this data is by far the “smart watch” (e.g. Apple Watch, Fitbit, Whoop). With some market share in recent years shifting towards a new “smart ring” category of devices (e.g. Oura, UltraHuman). Amongst these devices, there is a common barrier to continued use in both having to wear a physical sensor whilst sleeping, along with maintaining the device [4]. One recent product by the name of “Eight Sleep” aimed to avoid this pitfall by implementing their sensors via a mattress cover. Which based on consumer feedback, affirms the belief that the most ideal solution to sleep tracking is a non-wearable methodology.

However, even with Eight Sleep solving the first barrier to lasting consumer adoption, it shares an arguably even greater blocking factor with many of its predecessors, a subscription model. Software as a Service (SaaS) models have taken over the entire digital marketplace over the past decade. This has led to a growing fatigue and anger amongst consumers with even simple applications that could operate on one compact disc alone asking users to subscribe to unlock basic functionalities. As well as taking the capability of local saving user data away completely [5]. Conjunctionally, with the boom of “AI,” companies have exploited the virality of this term to add erroneous features and unnecessary complexity to every application experience. Consumers in response to both of these trends of owning nothing and paying more each month while being bombarded with bloated features have begun to protest against many products [6]. Hence, Noctisense aims to eliminate all of the previously stated barriers. Through non-intrusive sleep data measurement, locally-stored data with no cloud or subscription services, and no erroneous features.

Goals & Objectives

2.2.1 Base Goals

- Utilize thermal imaging sensor to record body temperature throughout the night.
- Implement restless movement tracking via 24 GHz mmWave radar tracking.
- Record audible disruptions during sleep period.
- Binary person detection using NIR optical subsystem

2.2.2 Advanced Goals

- Develop a user-friendly mobile application for displaying sleep data and audio clips, along with system configuration controls such as bedtime system automation.
- Measure respiratory rate using 24 GHz mmWave radar tracking.
- Design and build an aesthetically pleasing housing for the sensors that is consumer-friendly.
- Compare NIR optical system with commercial products and have lower cost with high quality

2.2.3 Stretch Goals

- Incorporate additional charging functionalities using Inductive/USB power delivery (USB-PD).
- Track respiratory rate using a NIR optical system to detect chest cavity movement.
- Add an additional measurement metric of heart rate, posture, and ambient room temperature.
- Develop automated recommendations within the mobile application to help the end-user improve their sleep hygiene.
- Integrate a programmable alarm clock functionality with display.

2.2.4 Base Objectives

- Average Body Temperature: Using a thermal image capture every sampling period, normalize pixel values that represent the heat map to account for environmental conditions. Then apply noise filtering using Gaussian blurring and detect hotspots by identifying pixels above a certain threshold and track their changes over time.
- Restless Movement: Using a 24GHz mmWave radar signal, specifically the MR24BSD1 radar module by Seeed Studios. This sensor has built in DSP processing to detect periods of movement within the target area. Then, apply a 2nd order Butterworth low-pass filter to smooth out high-frequency noise. We will use this data along with the modules included software library tools to record each period by the start and end points of detected movement and use the total duration of movement to determine period of restlessness.
- Sound Tracking: During the system's active period, monitor audio data with the signal normalized to the environmental conditions into the on-board DMA buffer within the ESP32-S3-N16R8. Then, when a triggering event over the set dB threshold value occurs, feed data into a ADPCM encoder to be stored until the event is detected to be over by the dB level dropping below the threshold for over a set period. Lastly, store this clip in its compressed format on the larger SD-flash storage to be synchronized to the application.

- Determine whether a person is physically present on the bed by detecting reflected infrared light from the user's upper torso. The system emits short pulses of near-infrared (NIR) light toward the expected location of the subject and monitors whether any portion of that light returns within a specific time interval. The received signal will be converted into a voltage to determine whether the reflected signal exceeds a predefined detection threshold within a nominal round-trip propagation delay of light at approximately 1.5 meters or greater.

2.2.5 Advanced Objectives

- Develop a mobile application utilizing the Flutter framework in conjunction with a REST API backend connecting to the database via HTTP. Then, synchronizing the sleep metric and audio data to the application. This connection will be achieved by utilizing the WI-FI capabilities of the ESP32-S3-N16R8 to host an HTTP server for TCP/IP communications, and connecting to the access point created with the end-user application. Lastly, we will develop system configuration control functionality within the mobile application to set system active time through remote changes to a configuration variable within the application.
- Respiratory Rate: Similar to the method used to measure restless movement, using a 24GHz mmWave radar signal, specifically the MR24BSD1 radar module by Seeed Studios. Which has built in DSP processing to accurately measure target metrics within the desired measurement area. We will use this data along with the modules included software library tools to record sleep data throughout the systems active time within the on-board database.
- Use 3D CAD software to develop a custom housing to accommodate all sensor components that will be 3D-printed into an aesthetic housing that is end-user friendly. Beginning with the base of the housing containing the power and optical systems, this will have to accommodate for the visibility of the beam while avoiding unwanted interference. Along with accounting for thermal considerations within the housing. If we encounter thermal hurdles we will utilize passive cooling primarily followed by active if absolutely necessary. The writing for the connections of the power and optical systems will be routed through the rear shaft of the lamp-style housing to the upper enclosure. The upper section of the housing will contain the PCB for the MCU, thermal imaging sensor, microphone sensor, and the radar sensor. Both the thermal imaging sensor and microphone sensor will require dedicated perforations to allow visibility and signal reception. While the radar sensor can be contained within the housing itself obscured from the end-user.
- Track over time to classify the subject as stationary with a modulated signal of 1 kHz using an infrared detector with a 0.8 responsivity. During each active monitoring period, the burst-averaged return signal from the photodiode will be recorded as a time-varying signal proportional to reflected optical intensity. This signal will be normalized to account for environmental variations such as ambient lighting and surface reflectivity.

2.2.6 Stretch Objectives

- For extended monitoring periods, the filtered micro-motion signal will be analyzed to extract periodic oscillations corresponding to respiratory motion. A bandpass filter within the expected human breathing frequency range (~0.1–0.5 Hz) will be applied to the signal prior to performing frequency-domain analysis using Fast Fourier Transform (FFT) or peak detection techniques. The dominant frequency component will be identified and converted

into breaths per minute (BPM) by calculating the number of oscillatory cycles within a defined observation window. The calculated respiratory rate will then be logged and stored for synchronization with the system application interface using the NIR system.

- Upgrade measurement collection by using similar techniques as in respiratory rate detection utilizing the existing feature set within the 24 GHz mmWave radar and thermal imaging monitors. This would also involve modifying the code to the application, and possibly adjusting the scalar function to calculate the overall sleep score.

2.3 Features & Comparable Existing Products

2.3.1 Smart Watches (Apple, Samsung, Google, Whoop, etc.)

Noctisense is a contactless sleep monitoring system designed to track sleep patterns and generate sleep quality scores without the need for wearable devices. It utilizes a combination of thermal imaging and radar technology to monitor the user from a bedside position. The system is integrated into a lamp design, allowing it to function as a standard light source while unobtrusively capturing biometric data. Noctisense draws power directly from a standard wall outlet, eliminating the battery constraints associated with wearable technology. By utilizing various sensors, Noctisense can monitor body temperature, movement, and acoustic disturbances to build a comprehensive sleep profile.

The Noctisense system relies on a suite of high-precision sensors controlled by an ESP32 processor. To monitor body temperature, it uses the MLX90640 thermal imaging module, which offers a 55°x35° field of view to adequately capture the user. The ESP32 processes this thermal data using a Gaussian blur to reduce image noise and improve definition.

Noctisense shares similarities with commercially available smart watches, as both technologies seek to track sleep patterns and assign quality scores. Both utilize sensors to detect physical changes during sleep, such as movement and breathing. The biggest differences lie within the form factor and the sensor methodology. Smart watches, like the Apple Watch, rely on accelerometers and blood oxygen sensors worn on the wrist to detect motion and determine sleep stages. In contrast, Noctisense is entirely contactless, using thermal and radar sensors to achieve similar results without physical contact.

Another significant difference is the power management and reliability of data collection. Smart watches are limited by battery life; if a user forgets to charge the device, sleep data is lost, disrupting long-term health tracking. Noctisense avoids this issue entirely by being a stationary, wall-powered device. While smart watches require the user to actively manage the device's battery, Noctisense operates continuously in the background.

2.3.2 Smart Rings (Oura, Ultra Human, Samsung, etc.)

Smart rings are a commercially available alternative to watches that utilize photoplethysmography (PPG) and accelerometers to measure sleep quality. The main difference is that smart rings condense this technology into a smaller form factor worn on the finger. This design solves the problem of the bulkiness associated with watches, though it introduces new logistical challenges such as specific sizing requirements. A significant characteristic of products like the Oura Ring is

the reliance on a software subscription, costing approximately $5.99 a month. These devices generally depend on external servers to process and display the data collected by the hardware.

Noctisense shares similarities with smart rings in its objective to track sleep hygiene using sensor data. Both projects utilize electronic sensors to monitor the user's physiological state. The biggest differences lie within the cost structure and data architecture. For smart rings, the device is often tethered to a subscription model and cloud computing. Noctisense keeps everything locally hosted, making it so the user no longer needs to communicate with an offsite server. This eliminates the recurring costs associated with smart rings, keeping the system free for users. This is achieved through the use of a REST API running directly on the system's MCU, ensuring total autonomy from external networks.

2.3.3 Smart Mattress Cover (Eight Sleep)

Devices like Eight Sleep utilize a grid of piezoelectric sensors embedded within the mattress topper to detect movements and read heart rate. This is due to the vibrations created by the user when they breathe in and out or when their heart beats. Using this technology allows eight sleep to achieve over 90% accuracy for Heart Rate, Heart Rate Variability, and breath rate. The data is interpreted by looking at the sharp peaks, indicating heartbeat, while the overall wavelength will determine breath patterns.

This differs from Noctisense as it's standalone and does not require anything on the bed in order to work. Data is also stored locally removing the reliance on cloud processing and subscriptions. Another similarity between Eight Sleep and Noctisense is how it achieves being a non-intrusive sleep measurement system. Amongst previously mentioned technologies that operate in a similar manner, key metrics of which include; restless movement, and respiratory rate.

Noctisense tackles this using the MR24BSD1 24 GHz mmWave sensor specifically designed for sleep applications. It operates with a detection range of up to 1.5 meters and has a variable sampling frequency we can adjust to our specific needs. Due to the intentional design of this radar module, the measurement data from the radar arrives at the MCU with the necessary DSP processing already done. This serial data is then used in conjunction with predefined source code libraries to feed our computations and metrics.

2.3.4 Audio Analyzers (Sleep Cycle, etc.)

Audio analyzers like Sleep Cycle rely on the users' phone's hardware to track sleep patterns. They use the phone's microphone to analyze the cadence of breathing as well as distinct sounds like snores. While this method is non-invasive, it suffers from significant accuracy issues, and the software will have to account for the environmental noise.

While Noctisense will also use a sound tracking microphone, we will implement lightweight encoding to only store audio clips over a certain threshold dB which will help us eliminate a large portion of the noise and make our design more accurate than the sleep cycle app. This accuracy improvement will greatly help us in differentiating different sounds and determining what they mean for sleep quality.

2.3.5 Optical Advantages

For this system, another example of our non-intrusive approach is in our body presence detection (on-bed vs. off-bed). A continuous binary detection subsystem is used to detect a reflective target at a working distance of approximately 1.5 m, as illustrated in Fig. 2.3.6.1. A near-infrared (NIR) laser source (940 nm) emits modulated optical signal towards the user's upper torso. The transmitted signal propagates through free space, reflects from the subject, and is collected by a receiving photodiode.

The system employs two aspheric condenser lenses to maximize optical coupling between the transmitter, target, and receiver. A 940 nm near-infrared (NIR) LED is positioned at the focal point of the transmitting lens to produce a concentrated beam with reduced divergence, increasing the optical irradiance incident on the target. A second aspheric condenser lens collects the reflected light and focuses it onto a silicon photodiode, significantly increasing the received optical power compared to an unassisted detector. Both theoretical modeling and experimental measurements demonstrated an increase in received optical power from the nanowatt range to the microwatt range, enabling reliable signal acquisition at the required operating distance.

Using radiometry one can derive the expected power values.
The arc length (l) of a circle is $l = \theta \cdot r_{radius}$ representing a 2D view of how the optics propagates respect to the diverging angle of our source. However, since we are in a 3-dimensional world will use the solid angle (Ω) to represent θ to determine the area profile. Below we have visuals of how to see this

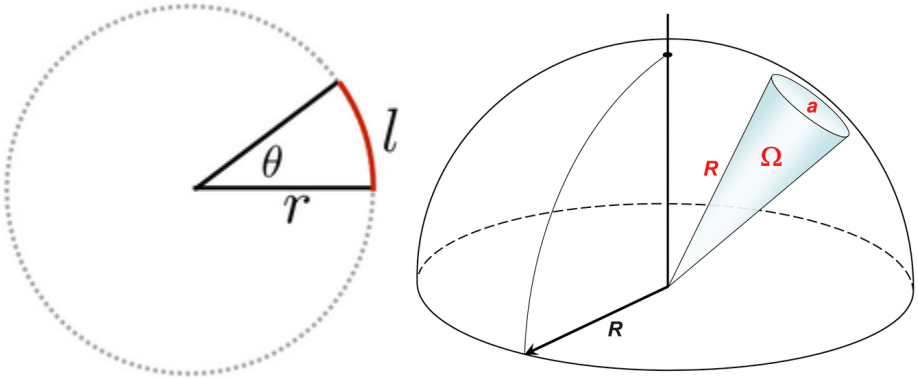


Figure 2.0: *Left representing arc length and on the right representing arc area.*

We can use these formulas to approximate within a magnitude to be true, as the detector is small compared to the distance r and the detector faces the source directly.
The spec sheet provides mW/sr (I_e) so determining power is $P = I_e\Omega$ and define our solid angle as $\frac{Area}{r^2}$. For spot size one can use x(distance)tan$(Divergence\ of\ LED)$
For step 1, how much power are we getting from the wall using a lens. Figure 2.1 displaying a visual.

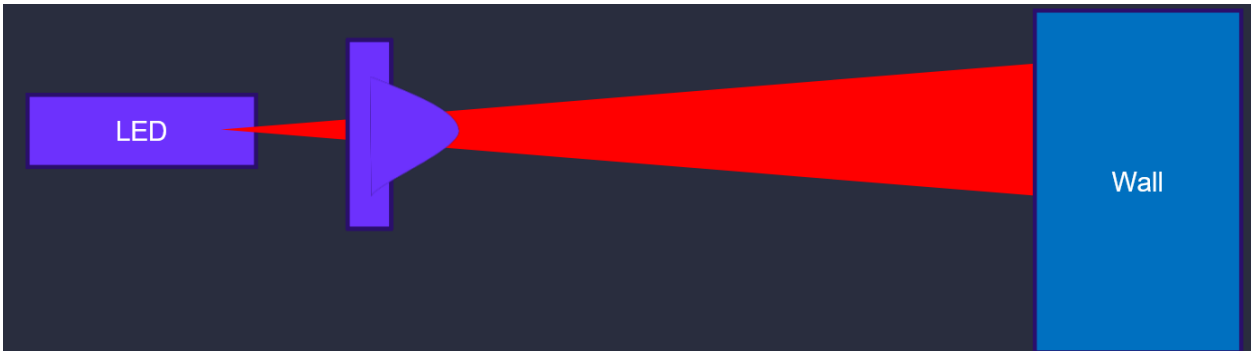


Figure 2.1: *demonstration of just LED detector*

The beam profile on the lens is 9.37 mm using the divergence of the LED and the position from the lens (the focal length) 20.1 mm, $radius = 20.1 \cdot \tan(25°) \approx 9.37\, mm$. With the addition of divergence of the lens we get another series radius $\approx 180\, mm$, originating from $\theta = \arctan(\frac{Diameter}{2f})$ (Diameter = 5 mm) ~7° $\Rightarrow 1500mm \cdot \tan(7°) \approx 180\, mm$

The power at the wall is described by the ratio of the initial power over the spot size

$$20mW \frac{1}{\pi(700)^2} = 0.0013\mu W$$

$$20mW \frac{1}{\pi 200^2} = 0.16\mu W$$

Putting a lens infront of the detector, one can control the concentration of light into a smaller area. Pseudo increasing the power by at least ten times.

Step 2

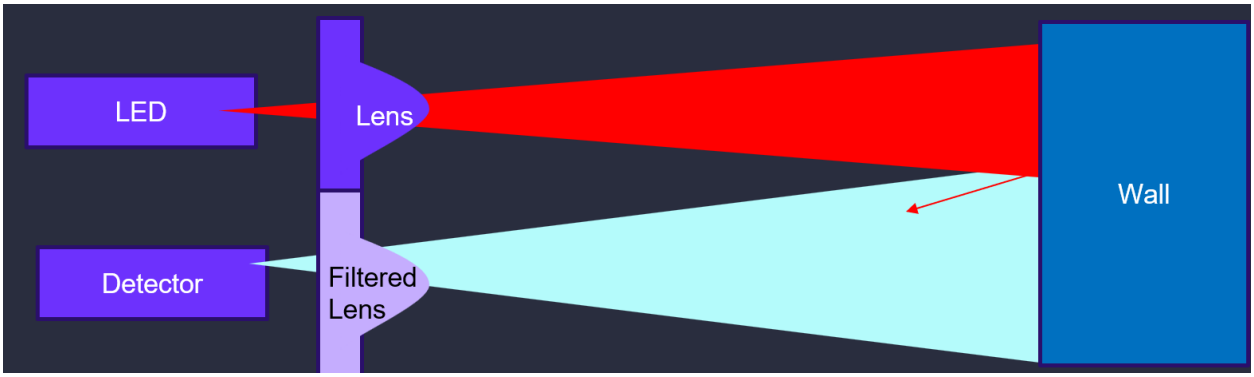


Figure 2.2: *Detector with lens*

We check the magnification ($M = \frac{z'}{z} = \frac{f}{z}$)would support enough concentration on the detector. Since our numerical aperture (NA) is 0.6, the entire spot size can be reflected back into the lens., From the height of the spot size times the magnification, we get our new spot size at the photodiode.

$$h' = h \cdot \frac{f}{z}$$

Since we have a 200 mm spot on the wall, focal length (f) of 20.1 mm (distance between image plane and lens, z'), and 1.5 m displacement of the detector (distance between the lens and the

object, z) we can apply the formula to obtain a new spot size of 2.68 mm ($or\ 200 \cdot \frac{20.1}{1500}$) which covers the detectors' active area of ⌀5 mm. Therefore, minimizing loss from overspreading. Additionally, this is a parameter that can be optimized for higher focal lengths to have new spot sizes to cover more area of the detector (making the system more robust) or lower for more concentration.

To obtain power at the detector:

$$P \cdot \frac{\pi\left(Diameter\ of\ \frac{lens(mm)}{2}\right)^2}{2\pi\big(Displacement(mm) - f(mm)\big)^2} = 40mW\,\frac{\pi\left(\frac{25.4}{2}\right)^2}{2\pi(1500 - 20.1)^2} \approx 1.47\mu W$$

The equation above describes how much power goes into the lens from the wall to the photodetector, as the focal length being our 20.1 mm lens. When increasing the Diameter of the lens we begin to pick up more power, same could be applied to the focal length (curvature of the lens).

With no lens the power hits nW. Assuming the light continues to propagate as a Lambertian, the wall reflects back another 1.5 meters

$$P = 0.04 * \frac{\pi 2.5^2}{3000^2} = 87nW$$

The introduction of the lens has increased the magnitude of power ~20 times.

For how the system would generally work: The system measures the relative signal energy arriving within an expected temporal detection window corresponding to the nominal round-trip path length. When a subject is present, a detectable return signal is observed within this window (Case 1). In the absence of a subject, the return signal either falls below detection threshold or exhibits temporal dispersion due to multi-path reflections from background objects. (Case 2)

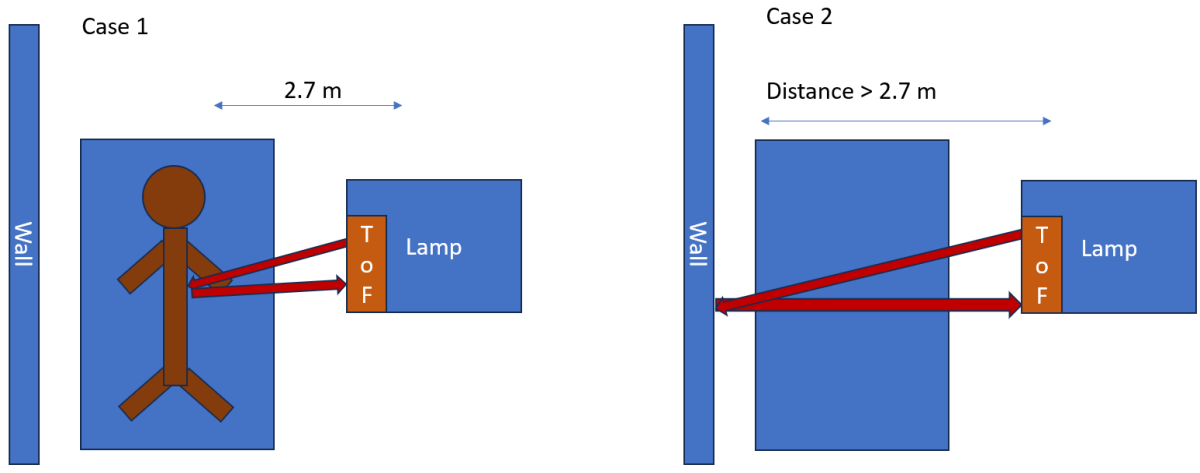
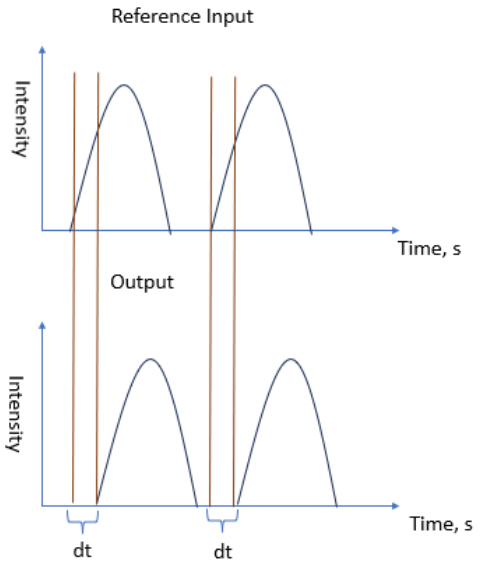


Figure 2.2 *A top view of two cases with and without a person.*

A Time of Flight sensor (Tof) will be used to compare with our person detection. In the papers [23 24 25 26] they focused on ultra fast pulses however, the concepts below will still apply.

In the case the group has enough time, the Time of Flight (ToF)will generally worked as the following and similarly to the purchased ToFour own ToF system. A laser diode supports a maximum duty cycle of 0.1 % with a 100 ns full-width half-maximum (FWHM) pulse width. To maintain conservative operation and account for driver variability, the system operates at a reduced 50 ns pulse width and 0.05 % duty cycle. A burst of 200 pulses is transmitted within a 20 ms integration period, which remains below the system monitoring interval of 50 ms (20 Hz). This burst-based acquisition improves detection robustness while maintaining compatibility with the processing bandwidth of the MCU.
The receiving photodiode has a high responsivity (~0.8 A/W) and 5 ns rise/fall time, enabling accurate detection of nanosecond-scale optical pulses. A narrowband 940 nm optical filter is coupled to the receiver to reduce ambient illumination noise. Collimating and focusing optics are used to improve optical coupling efficiency between the reflected signal and the detector. In Figure 2.2 Left has imagery of two pulses from the input and the output and there being a time delay (dt). The table would be the interpretation of the signal where we quantize the starting and ending frame, and how the MCU will detect.



Time (s)	Input	Output	MCU
1	0	0	0
2	1	0	0
3	0	0	0
4	0	1	1
5	1	0	0
6	0	0	0
7	0	1	1
⋮	⋮	⋮	⋮

Figure 2.3 *Input and Output with time delay (dt). The table is a result of the MCU detection.*

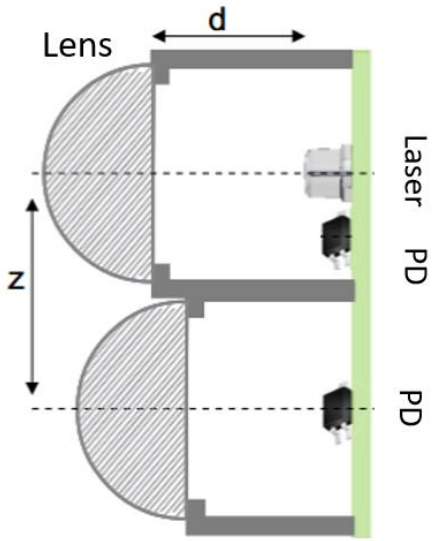


Figure 2.4 *General schematic of optical design.*

The optical design can be seen in Figure 2.3, displaying the reference input and output will be placed. Lens are shaded in gray, 3D print encapsulation is the grayed filling. A photodiode (PD) in each container and one laser in one of the containers. "d" will be around 25 mm. Assuming a distance z between laser and photodiode of 30 mm

In practice, ToF uses multiple pulses (bursts) to gather information on the distance it traveled [27]. Obtaining a histogram to average the data to obtain a realistic value and reducing stochastic noise.
The following equations define the relationships between duty cycle, pulse repetition frequency, burst duration, and sampling window used to determine these operating parameters. These will define the sacrifices of our system, like increase or decrease of sampling and what laser diodes we should be looking for.
Pulse Repetition Frequency (PRF): $f_{PRF} = \frac{Duty\ Cycle}{FWHM}$; Burst Time is time you need to keep firing pulses to collect enough data for one measurement $T_{Burst} = \frac{\#\ Samples}{f_{PRF}}$, also known as integration time; MCU updates without mixing data $T_{rep} = \frac{1}{MCU\ [Hz]}$; $T_{Burst} \leq T_{rep}$
Further noise will be reduced by using a white blanket as the target surface, which provides higher diffuse reflectivity in the near-infrared compared to darker fabrics. Lenses are used for collimating the laser and focusing the reflected signal to the photodiode.

In similar testing, the color of the material will not be a great significant factor in detecting a person. With the help of calibrating our system, having the system take in the first three seconds of reflection and using the power return as our reference point. Our return signal afterwords is normalized to the wall and so a percentage change (9%) we can see if a person enters or leaves the detectable area.

2.4 Specifications

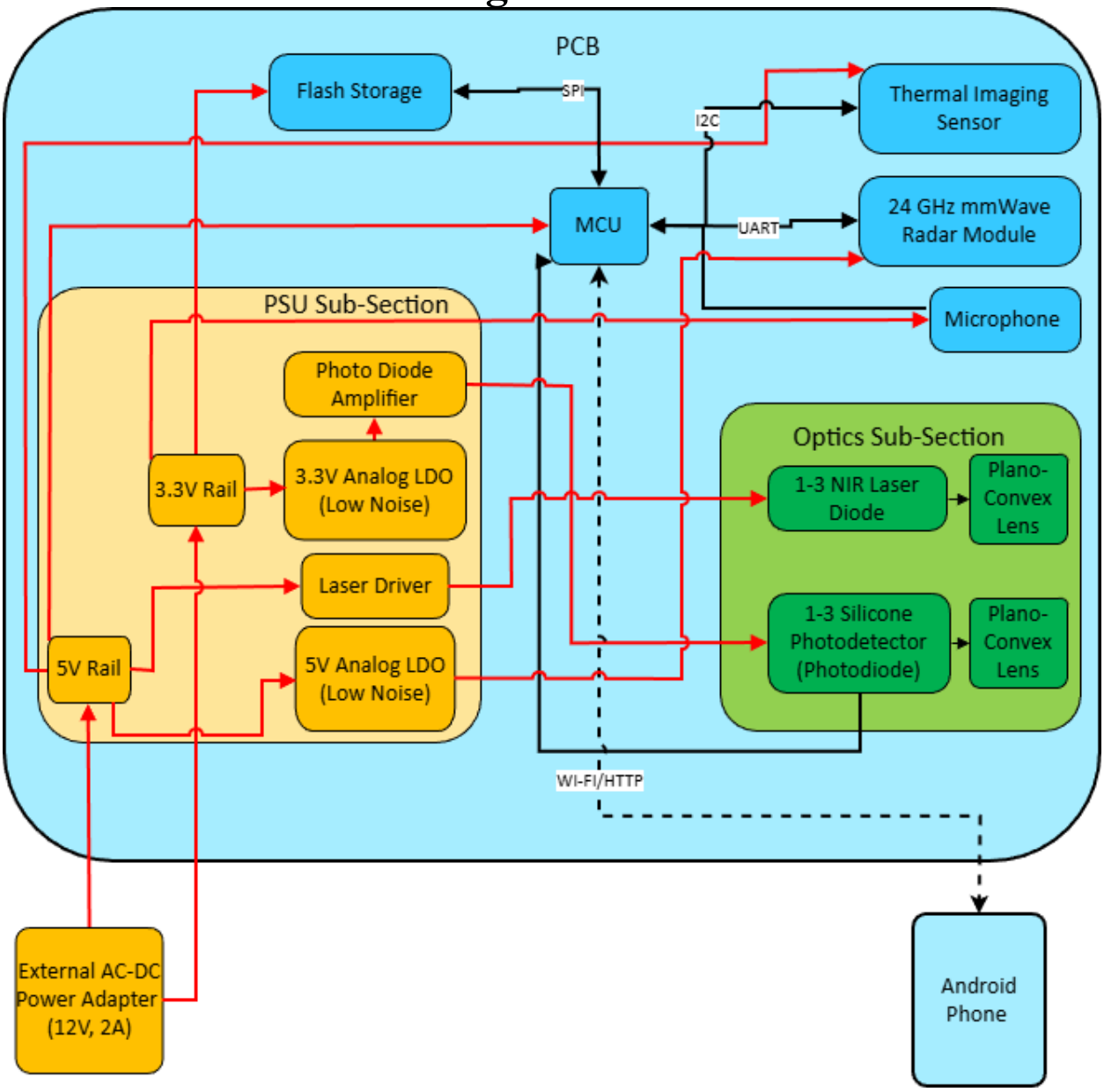
Table 2.1 System Specifications

Parameter	Description	Target Value
Monitoring Area	The maximum distance that the device will be able to measure effectively to track sleep based on our target zone of interest within the combined field of view of the sensor suite	2850 in^2
Lamp Housing Footprint	Overall dimensions of the lamp housing	19.7 in W x 23.6 in H
Operation/Setup	The system must be able to process and store data within the database, which is then used to update a mobile application. This is based on the estimated total execution time to update the database, combined with the latency of the applications connection and synchronization	Synchronization of updated data is populated on the mobile application within $\leq 1\ minute$ from the time of connection
Operating Voltage	Device voltage under which it will operate nominally	5V
Usability	Low friction between the amount of time and effort it takes to initially set up the system including the application. Based on hardware setup time which corresponds to the combined execution time and connection latency	10 minutes
Optical Detection Distance	Provides higher irradiance producing limited noise and higher SNR	1.5 m ≥

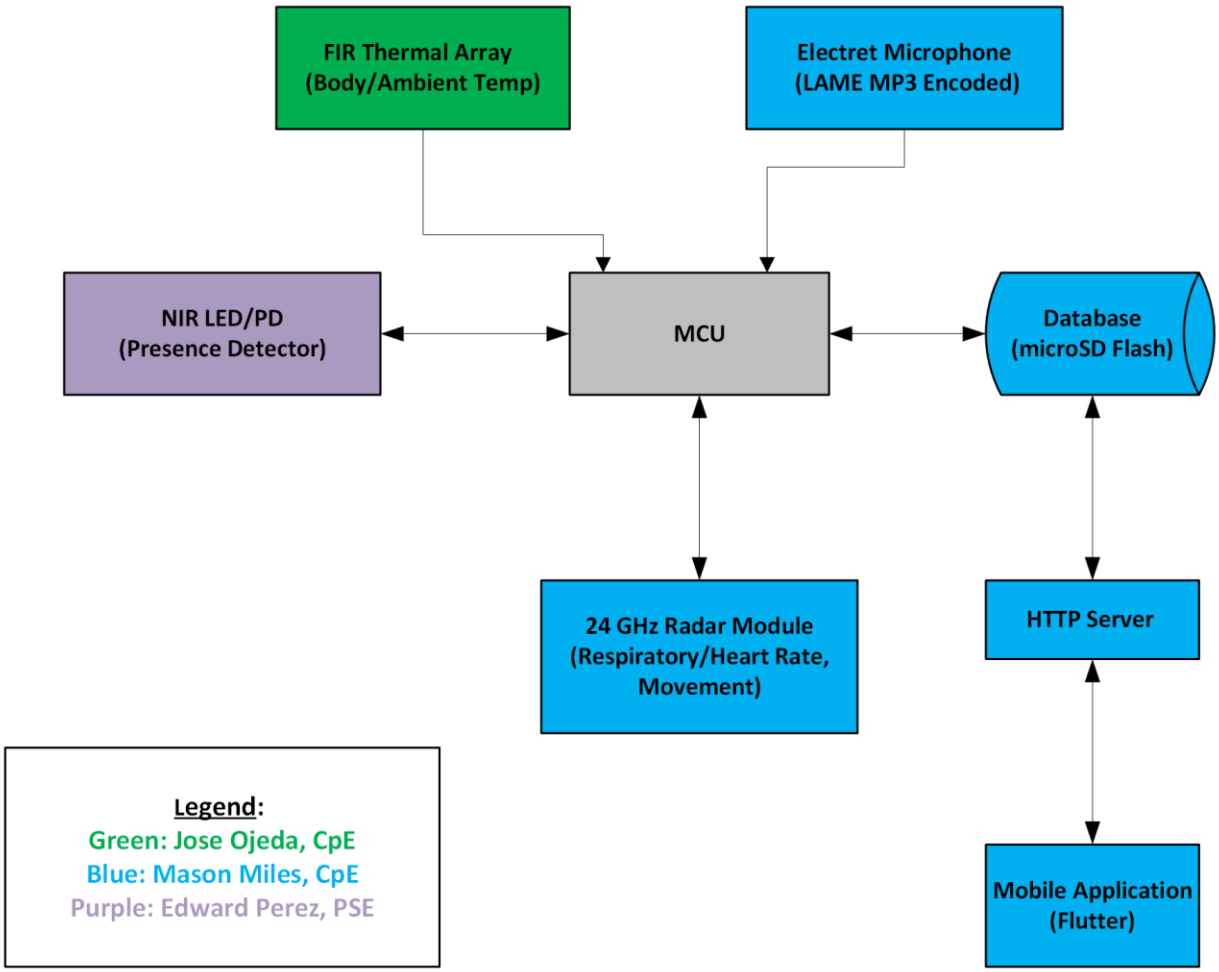
Table 2.2 Component Specifications

Component	Description	Target Value
Thermal Imaging Sensor	The thermal sensor will be used to calculate the user's body temperature.	Field of view of 55°x35°
Silicon Photodiode	Photodiodes will have a wide active area for easier photo detection.	Responsivity 0.65 A / W, rise/fall time 5ns, half angle 60 degrees
Laser Diode	Emits pulses for photodiodes to detect.	Wavelength: 940 nm Power: ≤ 5 mW
Plano-Convex lenses	Collimating and reducing spot size for sharper detection	BK7, Ø1/2", f = 25 mm
MmWave Radar Sensor	Detects motor movements without the need for physical contact using 24 GHz mmWave Doppler radar	Within a 1.5M range and a 40°x 40° detection angle
Microphone	Will capture ambient noise and determine environmental disruptions	20 Hz - 20 kHz
HTTP/LAN Connection (Micro-Controller)	Will transmit data to mobile application	0.3 - 0.7 Mbit/s

2.5 Hardware Block Diagram



2.6 Software Block Diagram



2.7 Prototype Illustration

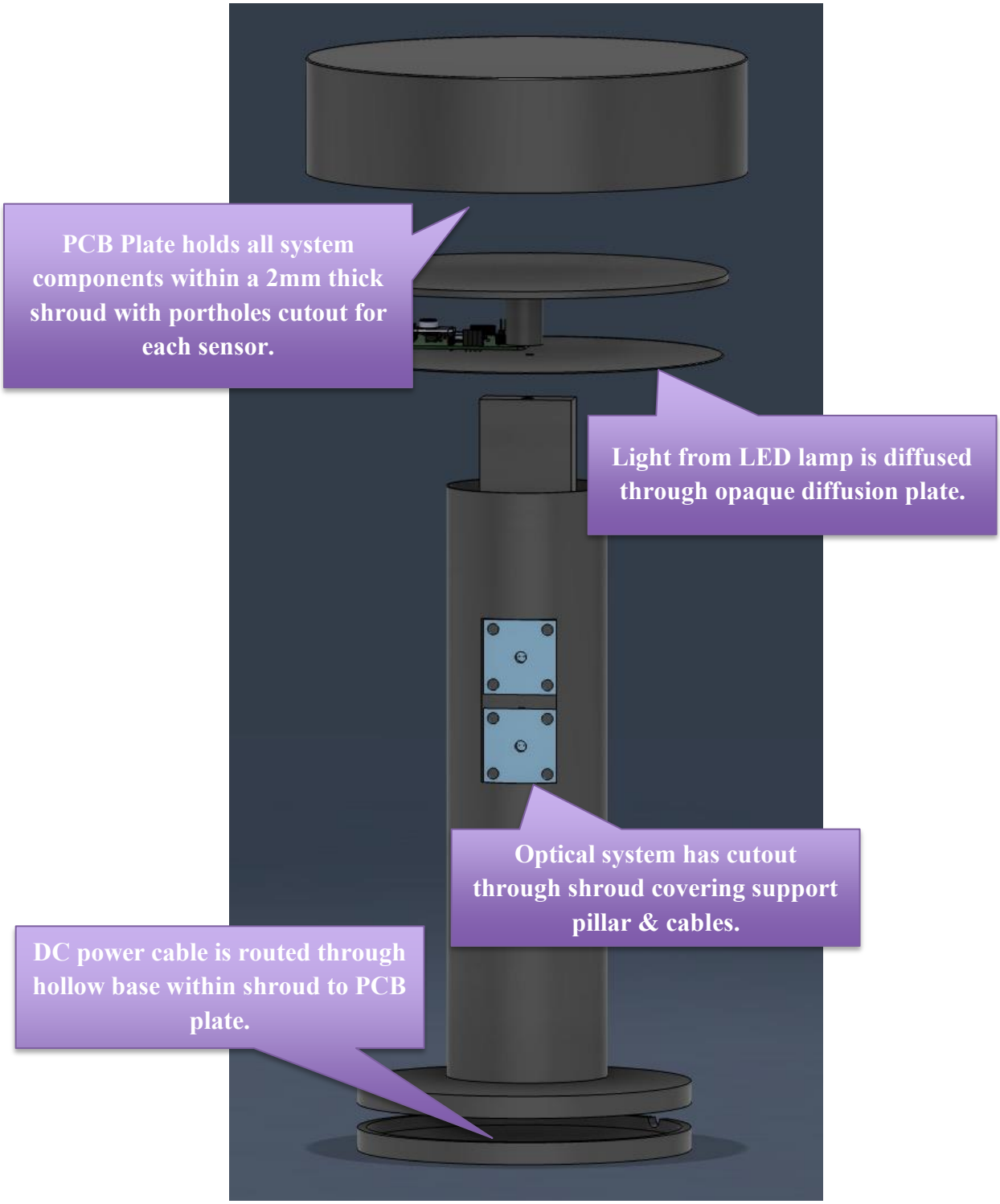
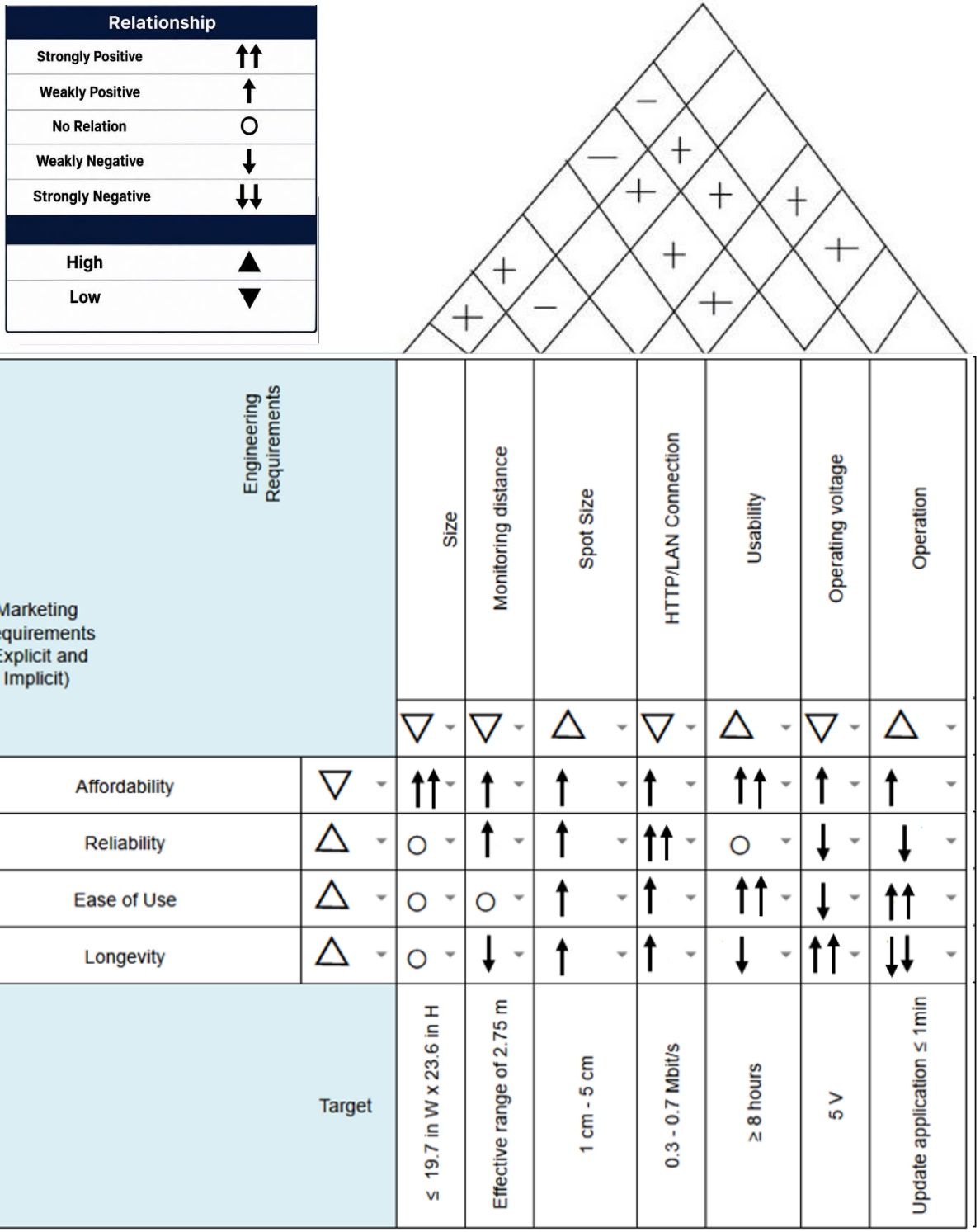


Figure 2.1: *Overall view of design*

2.8 House of Quality

Table 2.3 House of Quality Table



Chapter 3: Research and Selection of Technologies, Hardware, and Parts

3.1 Technologies

3.1.1 Main Processor Considerations

3.1.1.1 Real-Time Processing and Determinism

Within our system, one of the major requirements is real-time processing of various types of data. All of which must be controlled with tight timing and latency bounds. Microcontrollers executing bare-metal firmware or real-time operating systems (RTOS) provide the deterministic control over sampling intervals, interrupt handling, and task scheduling that we require for our system. All of which can be managed at our desired up-to MHz data rates across multiple sensors. FPGAs can deliver even stronger determinism and parallelism given their HDL-based design, but at the cost of significantly greater design effort. Which is not justified given the relatively modest bandwidth requirements of our projects sensors and data requirements compared against the increased complexity. Single-board computers (SBCs) on the other hand, run general-purpose operating systems where scheduling jitter, background processes, and network stacks introduce external latency to our software timing requirements that require implementation of strict real-time frameworks. Although many SBCs do have robust support for such frameworks, they also introduce considerable complexities which will be outlined in later sections.

3.1.1.2 Complexity of Sensor Integration

Our system incorporates several forms of data collection, including thermal imaging, radar, acoustic microphones, and NIR ToF data which demand a mix of digital interfaces and moderate data throughput. Modern MCUs like the ESP32 class integrate the required peripheral sets (I2C, SPI, I2S/PDM, UART, ADC, PWM, and GPIO) that natively support common thermal imaging sensors, mm-wave radar modules, digital microphones, and photodiode data as demonstrated in existing ESP32-based designs that combine radar motion sensing and audio capture. This peripheral richness, coupled with direct memory access engines and flexible pin multiplexing, gives us straightforward interfacing to multiple sensors without external software or logic, whereas FPGA designs require the explicit construction of each interface in hardware description language and SBCs often rely on kernel-level driver support or USB bridges for less common sensors, increasing integration risk and complexity to a degree.

3.1.1.3 Software Ecosystem and ESP-IDF Library Support

For our project software, the primary concern is existing library integration. As this will allow us to incorporate a wider variety of sensors and computations for the system in our constraint deadline. Thus, the software ecosystem substantially favors a microcontroller approach centered on ESP32-class devices. Specifically, the ESP-IDF framework, which provides mature libraries and examples for Wi-Fi connectivity, HTTP servers, TLS, time synchronization, OTA updates, and common sensor interfaces such as I2C, SPI, and I2S audio, as well as community projects that integrate radar modules, cameras, and microphones for security and monitoring applications. This ecosystem enables rapid reuse of reference designs and proven middleware, whereas FPGA development would require building or licensing comparable IP cores and protocol stacks, and

SBCs would pivot the software base to Linux-centric frameworks that do not directly exploit ESP-IDF and may fragment the development effort across the various stacks.

3.1.1.4 Software Development Language and Dependencies

When considering a primary language that is suitable for this project in terms of speed, simplicity, and broad library support. There are many suitable options for our system requirements, primarily; C/C++, Python, Rust, and Java. With the fastest of these primary candidates being C/C++ or Rust based on the premise of being compiled languages. Wherein they are translated to machine code before being executed, unlike Python being a scripting language that must be interpreted before turning into machine code. Along with low-level features in areas like memory management and optimizations for runtime making them faster than Java. Thus, deciding between C/C++ and Rust comes down to integration complexity and feature set. C/C++ being the much more widely used and older language, along with it being used widely within our chosen ESP-IDF framework makes it a more ideal candidate for our use case. There is an argument to use Rust for its security improvements over C/C++, but for our system existing within a consumer's household with no connection to external networks integration takes priority between these.

3.1.1.5 Development Effort and Rapid Prototyping

Given the timeline of this project, the development effort associated with each technology is a decisive factor. Microcontrollers offer a comparatively gentle learning curve, with C/C++ development, straightforward debugging, and abundant examples for sensor drivers and web servers, leading to faster bring-up and iteration cycles for a multi-sensor prototype. FPGAs demand proficiency with hardware description languages and digital design methodologies, significantly increasing the time and expertise required to implement and verify sensor interfaces, control logic, and streaming pipelines; SBC solutions reduce low-level work but introduce complexities around OS configuration, dependency management, and system hardening that are non-trivial when reliability is essential for long-term overnight monitoring.

3.1.1.6 HTTP Server Hosting and Network Stack

While our requirement to host a local HTTP server for an accompanying application can be satisfied by all three platforms, modern microcontrollers now routinely support embedded web servers with acceptable throughput for configuration dashboards and low-rate data visualization. ESP32-based designs use lightweight HTTP stacks within the ESP-IDF framework to implement REST APIs and web interfaces over Wi-Fi, while still allowing the device to enter low-power modes between transactions when the access pattern is sparse. SBCs can leverage full web servers and application frameworks, but their always-on network stacks and background services increase baseline power draw, and their complexity is unnecessary for the limited concurrency and bandwidth expected in a single-user, local-only sleep monitoring interface; FPGAs generally require an embedded soft core or external processor to host an HTTP server at the required abstraction level, which undermines their advantage and reintroduces MCU-like software layers.

3.1.1.7 Power Consumption and Energy Budget

For the specific application of our system monitoring sleep throughout the entire night, the power envelope strongly favors a microcontroller implementation. Mainstream SBCs such as Raspberry

Pi 3/4 exhibit idle power consumptions on the order of 1.4–2.7 W, with HTTP serving and moderate CPU activity pushing consumption into the 2.4–5.1 W range, which is substantial for a bedside appliance. By contrast, microcontrollers are designed for low-power IoT operation, offering active power in the hundreds of milliwatts realm with integrated Wi-Fi and very low-power modes that can reduce consumption by orders of magnitude when sensors and the local HTTP server are duty-cycled. FPGAs, though capable of fine-grained clock gating, generally consume more power than MCUs for equivalent control workloads because of their large configurable logic fabrics, making them less attractive for ultra-low-power, always-on sensing systems.

3.1.1.8 Thermal Behavior and System Integration

Thermal management is a critical concern for our platform that must operate in close proximity to the user and also utilizing an on-board thermal imaging sensor; elevated device temperatures can both affect measurement accuracy and user comfort. SBCs with multi-watt dissipation often require heat sinks or airflow, leading to local hot spots and potential thermal coupling into nearby thermal sensors, complicating calibration and mechanical design. Microcontrollers, operating at significantly lower power, generate far less heat and can typically be integrated into compact enclosures without active cooling, simplifying packaging and minimizing thermal interference with the thermal imaging sensor. FPGAs, while potentially configurable for lower clock rates, still tend to have higher power densities than MCUs and may require more careful PCB layout and thermal mitigation than a microcontroller-based solution for the same application scope.

3.1.1.9 Hardware Complexity and PCB Design

From a hardware design perspective, a microcontroller-based solution simplifies the printed circuit board and reduces risk in fabrication and assembly. MCUs such as the ESP32 integrate RF front ends, memory, and power management support, allowing relatively simple two- or four-layer PCBs with modest routing constraints for the sensor interfaces and power distribution. In contrast, FPGAs often require careful layout for multiple supply rails, high-pin-count packages, configuration memories, and external interfaces, while SBC-based designs frequently rely on off-the-shelf modules but then need daughterboards or HATs for sensor breakouts, increasing mechanical complexity and potentially degrading signal integrity for high-impedance analog or high-speed digital links.

3.1.1.10 Scalability, Upgradability, and Future Extensions

While FPGAs excel in applications that demand significant reconfigurability and parallel process scaling, the anticipated evolution of this sleep-monitoring platform is more likely to involve incremental additions of sensors, refinement of signal processing algorithms, and enhancements to the user interface. Microcontrollers provide ample headroom for such evolution by supporting firmware updates, modular sensor drivers, and protocol extensions within the existing hardware, leveraging over-the-air update mechanisms and flexible software architectures supplied by platforms such as ESP-IDF. SBCs can scale up in software capability but at the cost of increased resource usage and energy, and FPGAs, while reprogrammable, still require extensive verification for each new hardware configuration, making software-centric extensibility on an MCU a more practical path for iterative system enhancement in this domain.

3.1.1.11 Reliability, Maintainability, and System Robustness

Reliability as a pillar of importance within our system, encompasses both continuous overnight operation and long-term maintainability with minimal user intervention. Microcontrollers run minimal firmware stacks with limited moving parts, reducing susceptibility to software crashes, file system corruption, or dependency breakage relative to OS-based SBCs that rely on complex storage hierarchies and update mechanisms. Additionally, the deterministic and resource-bounded nature of MCU firmware simplifies worst-case analysis and watchdog integration, whereas FPGA-heavy designs must carefully validate timing closure and corner cases in hardware, and SBCs require comprehensive OS-level hardening and monitoring to achieve comparable robustness under the same environmental and usage conditions.

3.1.1.12 Cost Efficiency

When comparing a microcontroller, (SBC), and an FPGA for our system, the microcontroller provides the most favorable cost profile at both unit and development levels for the required feature set. Readily available 32-bit MCUs such as the ESP32 family and similar devices are typically available in the sub-$5–$10 range in modest volume and fast shipping times, whereas even low-end FPGAs and complete SBC modules frequently exceed this cost once configuration memory, power management, and PCB overheads are included. In addition, FPGA environments often incur higher non-recurring engineering costs through proprietary toolchains and IP licensing, and SBC solutions require more expensive supporting components (high-current regulators, non-volatile storage, connectors), while microcontroller toolchains are commonly free and the bill-of-materials overhead is minimal.

Table 3.1 Comparison of Processing Technologies

	MCU (ESP32/Arduino)	Single-Board Computer (Raspberry Pi)	FPGA (Digilent Basys)
Core Architecture	32-bit Xtensa dual-core MCU; in-order, embedded focus	64-bit ARM application processor; out-of-order, general-purpose CPU	Array of configurable logic blocks, embedded DSPs, and optional soft-cores
Clock Speed	80–240 MHz typical; configurable DVFS	1–2 GHz typical; multi-core, OS-driven frequency	50–500 MHz common for logic; interface clocks up to GHz domains
Memory Organization	On-chip SRAM and ROM; external flash via SPI; limited, but tightly coupled	DDR SDRAM on-board; layered caches plus swap/storage; large working set	Block RAM and distributed RAM primitives; external DRAM when needed via memory

			controllers
Power Management	3–4 main rails; integrated LDOs, low-power sleep states (light/deep sleep)	Multiple high-current rails; complex PMIC; dynamic voltage–frequency scaling	Many domain-specific rails (core, I/O, PLL); external PMIC required
I/O Interfaces	Multiple I2C, SPI, I2S/PDM, UART, ADC, DAC, PWM, RTC	GPIO, I2C, SPI, UART, USB, HDMI, Ethernet; often via SoC or external bridges	Flexible I/O banks (LVCMOS, LVDS, etc.); each interface must be instantiated in HDL
On-Chip RF/Connectivity	Integrated 2.4 GHz Wi-Fi + BLE transceiver and baseband	2.4/5 GHz Wi-Fi + Bluetooth via discrete module or SoC	No native RF; requires external transceiver or soft-core connectivity stack
Timing Controls	Windowed timers, RTOS, and deterministic interrupt scheduling for sensor sampling	OS scheduler with jitter; real-time extensions possible but not default	Cycle-accurate pipelines and parallel state machines; strongest timing control
Programming Model	C/C++ firmware; direct register access and HALs; deterministic execution	Linux userspace + kernel drivers; C/C++/Python; event-driven and multi-threaded	HDL (VHDL/Verilog) or HLS-based; concurrent hardware description
Development Environment	Free IDF/IDE; fast compile–flash cycles; simple debugging	GCC/Clang, Python tools, build systems; OS image build and deployment overhead	Proprietary HDL tools (synthesis, place-and-route, timing analysis); long compile times

3.1.1.13 MCU Selection

As described in prior sections, a microcontroller (MCU) presents itself as the clear option in terms of performance, reliability, cost, and complexity overhead. This will allow us to transition from development board to custom fabricated components smoothly over the course of our project. Below is a table comparing the performance of various popularly used MCUs based on data from the manufacturer(s), from which we can derive the most ideal for our use case.

Table 3.2 Comparison of MCU

	ESP32	ESP32-S3-N16-R8	ESP32-P4	Arduino Nano R4
CPU Cores/Clock Speed	Xtensa single-core LX6 CPU, frequency up to 160MHz	Xtensa LX7 Dual-Core 240MHz CPU	RISC-V Dual-Core 400MHz HP CPU + Single-Core 40MHz LP CPU	Renesas RA4M1 (Arm Cortex-M4) up to 48MHz
WI-FI	Yes	Built-in 2.4 GHz Wi-Fi 4	None	None
UART	Yes	Yes	Yes	Yes
I2C	Yes	Yes	Yes	Yes
SPI	Yes	Yes	Yes	Yes
I2S	Yes	Yes	Yes	NO
PWM	Yes	Yes	Yes	Yes
ADC	Yes	2X (12-bit, 20 Channel)	Yes	14-bit, 8 Channel
DAC	Yes	Yes	Yes	Yes
Flash Memory	4MB	16MB	16MB	256kB
RAM	520kB	384kB ROM, 8MB SRAM	768kB on-chip SRAM + 8kB SPM	32kB
DMA	Yes	Yes	Yes	Yes
Operating Voltage (V)	3 ~ 3.6	3 ~ 3.6	3 ~ 3.6	4.8 ~ 5.5
Current Draw (A)	0.379	0.379	0.380	0.500
Power Consumption (W)	1.3644	1.2	Up to 10	2.75

Cost (USD)	8.99	12.99	17.28	15.47

Based on the data comparison above, the best option based on our requirements, cost, and availability are the ESP32-S3-N16-R8. As this SoC provides ample support for all of our various sensor data as well as capacity for future growth for later development.

3.1.2 Movement/Respiratory Tracking

3.1.2.1 Primary Methodology of Measurement

While passive infrared motion detection (PIR) works by detecting broad movements by sensing IR head changes. Millimeter-wave radar works on Doppler principles: it emits continuous or FMCW radio waves and measures frequency shifts from any moving surface. This lets the system directly detect velocity or displacement of a person (even small chest movements). The Seeed MR24BSD1 outputs raw radar frames which are decoded by its library into motion states, for example, its Bodysign_judgment() function classifies “no one/stationary” vs. “movement” based on thresholds. Whereas, a PIR sensor measures changes in infrared radiation and simply signals binary motion/no-motion. PIRs are very inexpensive (~$10) and easy to use, but cannot detect very slow or subtle movements. Similarly, a video or thermal camera system used at a high sampling frequency needed for movement or respiratory tracking would require high data rates and heavy image processing constantly. In practice, the 24 GHz radar uniquely captures minute body motions (including breathing) with fine resolution, all without physical contact or image capture.

3.1.2.2 Integration Complexity

Integration of the mmWave sensor is straightforward. The radar module is a small PCB with 5 V (VCC) and GND pins, plus UART TX/RX and two GPIO outputs. One simply wires 5 V/GND to power it and connects TX/RX to the ESP32’s serial port (or uses the I/O pins for a simple presence switch). By comparison, a PIR sensor also only needs 5 V, GND, and its output tied to a microcontroller input (with a pull-up resistor). Complex devices like cameras or LIDARs would require high-speed buses and extensive setup. In our multi-sensor system all devices speak ESP32-compatible interfaces (UART, SPI, I²C, GPIO), so the radar fits in without special adapters or drivers. In short, hooking up the 24 GHz module is almost as easy as wiring a digital sensor, and no specialized calibration is needed beyond uploading firmware

Table 3.3 Movement/Respiratory Technology Comparison

	24 GHz mmWave Radar	PIR (Passive IR)	Ultrasonic (40 kHz)	IR Depth / ToF	RGB / Thermal Camera
Operating Principle	Doppler shift + FMCW phase detection	Detects IR radiation changes	Time-of-flight of acoustic waves	Light pulse ToF / structured IR	Image capture (visible or IR spectrum)
Effective Range	0.2 m – 5 m (optimal ≤3 m)	2 m – 7 m (coarse)	0.02 m – 4 m	0.1 m – 4 m	0.5 m – 10 m+
Motion Sensitivity	Very high (micro-motion, breathing detectable)	Low (only large motion)	Medium (cm-scale motion)	Medium–high (depends on resolution)	Very high (pixel-level motion)
Static Presence Detection	Yes (via phase/amplitude variation)	No	Limited	Yes	Yes
Through-Obstacle Capability	Partial (blankets, clothing)	No	No	No	No

Lighting Dependency	None	None	None	Low (active IR)	High (RGB), Medium (thermal)
Privacy Impact	Very low (no images/audio)	Very low	Very low	Medium (depth map)	High (visual data)
Data Output Type	Processed motion states / raw RF frames	Digital HIGH/LOW	Distance values (cm)	Depth map / point cloud	Video frames / thermal map
Data Rate	~10–100 B/s	~1 bit event	~10–100 B/s	~KB/s–MB/s	MB/s
Processing Load (ESP32)	Low–Moderate (library-assisted)	Negligible	Low	Moderate–High	Very high (requires external processing)
Interface	UART / GPIO	GPIO	GPIO (trigger/echo)	I²C / SPI / UART	USB / CSI / SPI
Power Consumption	~0.4–0.6 W	~0.05 W	~0.1–0.3 W	~0.5–1.5 W	1–5 W+
Environmental Robustness	High (dust, temp, darkness)	Medium (heat sensitive)	Medium (airflow sensitive)	Medium	Low–Medium

Integration Complexity	Low (prebuilt modules + libs)	Very low	Low	Medium	High
Cost (consumer BOM)	~$20–$40	~$5–$10	~$5–$15	~$20–$80	$50–$200+
Suitability for Sleep Tracking	Excellent (best overall)	Poor	Limited	Moderate	High (but impractical)

Table 3.4 Component Selection

Parameter	Seeed Studio MR24BSD1	DFRobot SEN0395	Hi-Link HLK-LD2410	Acconeer A111
Frequency Band	24 GHz ISM	24 GHz ISM	24 GHz ISM	60 GHz (pulsed radar)
Detection Range	~0.2–5 m	~0.2–9 m	~0.2–6 m	~0.06–2 m
Optimized Use Case	Sleep monitoring (bio-motion)	General presence detection	High-resolution presence tracking	Short-range precision sensing
Micro-Motion Detection	Yes (breathing, sleep states)	Limited	Good (configurable zones)	Excellent (high resolution)

Output Data Type	Processed states + raw frames	ASCII motion data	Distance + energy per gate	Raw radar envelope data
Interface	UART + GPIO	UART + GPIO	UART	SPI / I²C
Baud Rate	115200	115200	256000 (configurable)	High-speed SPI
Onboard Processing	High (built-in sleep algorithms)	Medium	Medium–High (DSP filtering)	Low (host-side processing required)
ESP32 Integration	Very easy (Arduino libs)	Easy	Easy (popular in ESP32 projects)	Difficult (requires DSP + SDK)
Data Rate	Low (~10–100 B/s)	Low	Low–Moderate	Moderate–High
Power Supply	5 V @ ~100 mA	5 V @ ~90 mA	5 V @ ~70–100 mA	3.3 V @ ~50–100 mA
Power Consumption	~0.5 W	~0.45 W	~0.4–0.5 W	~0.3–0.5 W
Field of View (FOV)	~120°	~100°	~120°	Narrow (~40–80° depending config)
Firmware Configurability	Limited (high-level API)	Limited	High (zones, thresholds)	Very high (raw radar tuning)

Library / SDK Support	Strong (Seeed Arduino libs)	Moderate	Strong (community + tools)	Strong (but complex SDK)
Cost (Unit)	~$25–$35	~$20–$30	~$10–$20	~$50–$100

3.1.2.3 Software Support

The MR24BSD1 comes with an Arduino-compatible library and example code. This library provides high-level functions: for instance, recvRadarBytes() reads the latest frame and Bodysign_judgment() parses the motion data, while Sleep_inf() decodes breathing rate, sleep state, and quality metrics from the radar output. As a result, most signal-processing work is encapsulated in these routines, and the ESP32 firmware simply needs to call the library and read its outputs. The ESP32 platform natively supports these Arduino libraries (or equivalent ESP-IDF code), and can easily drive a Wi-Fi/Web server to publish the results. This supporting the idea that existing software libraries and community examples make integration very practical. In implementation, a developer writes a small loop that fetches the radar frame each second, applies the provided parsing functions, and then serves the interpreted data (e.g. motion or breathing status) via HTTP or MQTT.

3.1.2.4 Throughput/Computational and Data Rate Considerations

The mmWave radar produces very little data. By default the sensor outputs a short ASCII string (about 12 bytes) once per second. Documentation[48] confirms the data refresh rate is variable and customizeable, and the interface runs at 9600baud. In practice this amounts to on the order of 10–20 bytes per second of actual data, which is effectively negligible. The ESP32 can parse this stream and execute the simple parsing routines with almost zero load. Any additional computation (e.g. threshold checks or small algorithms) is trivial compared to the MCU's capacity. Even if we increased the radar output rate, the data volume remains tiny relative to e.g. video. By comparison, a camera might produce megabytes per second. Here, the radar's low data rate and on-board preprocessing mean the processing and networking overhead is very low. The HTTP server only needs to send a few bytes of JSON (e.g. "movement detected" or breathing rate), which is negligible over Wi-Fi. In summary, the radar's throughput and computational demands are minimal and well within the ESP32's real-time capabilities.

3.1.2.5 Energy Budget Considerations

Power consumption for this sensor is low, the selected radar module operates at 5 V and draws about 90–100 mA (≈0.5 W) under normal conditions. The ESP32 (with Wi-Fi on) may draw a couple of hundred milliamps, but together the entire system stays under ~2 W (<0.4 A at 5 V). This meets the requirement of staying below 5 V with low current draw. Since this is a mains-powered bedside device, continuous operation is acceptable. (For very low-power designs, we could duty-cycle the sensor; for example, an ultrasonic sleep monitor paper reduced its current from 321.75 mA to ~6.9 mA by waking the sensor only periodically[49]. However, our radar runs continuously for real-time monitoring.) In practice, ~1–2 W total is quite low for an always-on

home sensor. Key considerations are simply using an efficient 5 V regulator and ensuring stable power (the radar's datasheet allows 4.5–6 V). Overall, the energy budget is very manageable for a consumer product.

3.1.2.6 Thermal Considerations

Thermal effects are minimal with this design. The radar board and ESP32 together dissipate only ~1–2 W, so they will warm only slightly. The MR24BSD1 is rated for ambient use up to +60 °C, and with only ~0.5 W dissipation it should remain well below that in normal room conditions. Likewise, the ESP32 board may run warm but well under its spec limits. At most we may require a small vent on the enclosure to avoid any heat trap, or at most a passive heat-sink, but no active cooling is needed. In comparison, high-power transmitters or 77 GHz automotive radars might require heatsinking, but a 24 GHz sensor at low output power generates negligible heat. In short, as long as the module is not enclosed in an airtight box, thermal management is trivial. Ambient bedroom conditions and the module's temperature ratings comfortably cover continuous operation.

3.1.2.7 Reliability and Maintainability Considerations

The MR24BSD1 is a solid-state module with no moving parts, which yields high reliability. The Seeed sensor is specified for continuous operation, and even advertises "high stability" unaffected by temperature, humidity, airflow, dust, or light[48]. A similar DFRobot radar module touts strong anti-interference performance and stable sensing in adverse conditions[50]. In practice this means the sensor should operate night after night without drift or recalibration. Maintenance is minimal, essentially just occasional resets or firmware updates via the ESP32 if new features are added. The only practical care would be ensuring the radar's antenna opening isn't blocked or excessively dirty. If a problem does occur, the module supports a factory reset and provides diagnostic output on its serial interface. Overall, the radar-based system is as maintainable as any other IoT sensor; it compares favorably to camera systems (which can suffer lens/sensor issues) or ultrasonic modules (which may wear). A failed module can simply be replaced as a drop-in unit, keeping long-term maintenance straightforward.

3.1.2.8 Cost Efficiency

The chosen 24 GHz radar strikes a good balance of cost and capability. On Seeed's website the MR24BSD1 lists for about $28 each (with volume pricing near $24)[48]. An entire sensing node (radar + ESP32 + ancillary parts) can be built for only a few tens of dollars. For comparison, a simple PIR motion sensor costs around $5–$10, and a high-end thermal camera can cost hundreds, while there are some more budget options. Even among radar options, the 24 GHz module is cheaper than most 60 GHz or UWB sensor kits. Importantly, there is no recurring subscription or backend cloud fee. For a consumer sleep-monitoring device, these component costs are quite low. In summary, the mmWave solution provides advanced sensing and privacy at a price point compatible with a mass-market product.

3.1.3 In-Bed Detection

3.1.3.1 Comparing Optical Technologies

The specific implementation of the optics is to find someone laying on a bed 1.5 meters away. At the core this will be done with direct Time of Flight technique due to its affordability and reliability.

The exact info that is extracted from the system, such as the diodes, will produce a Voltage that identifies if a person (or thing)is on or off the bed. We send a pattern signal from the emitter and if the signal comes back we take the time differential to verify a person is on the bed (which is 1.5 m). The Near Infrared Camera will be combined to recognize if an actual person is on the bed, from imaging processing. The optical subsystem is to verify a person is on the bed, as one camera can't tell if a person is some distance away. Using only the camera would need heavy computation and or a second camera.
Additionally, the camera would assist in cases where the laser can't be aligned or if theirs is a pillow interfering. The optical system can generally tell if a person or thing is on the bed. But the combination of the thermal camera and this optical subsystem would accomplish if a person is on or off the bed.

Detection systems come in many shapes and sizes. Overall it is a light source, optic modulation (such as lenses, prism, and etc.), detection, and post processing. However, the system best fit has to consider the constraints such as cost, size, complexity, range, noise, and light conditions. I will compare Structured light Camera, LiDAR camera, Michelson Interferometer, and our direct Time of Flight System (dToF).
Structured light depth sensing technology or system fundamental uses a camera and light projector to calculate depth and presence. The technology projects a structured illumination pattern onto the surface and detects surface distortion via camera-based optical flow OpenCV program. For example a matrix, sine, or other patterns will be used to produce a high resolution and calibrate the mapping, in addition, often backed up with a Radio Stereo for correction. The optical subsystem would use a Digital Micromirror Device (DMD , consists of many microscopic mirrors that can tilt to create pixels for images.) to imprint a pattern to build a profile of the system (quantizes what the camera sees as actual unit values). The system takes in a few frames of the object and adjusts for any distortion, such as the picture below[20 21 22].

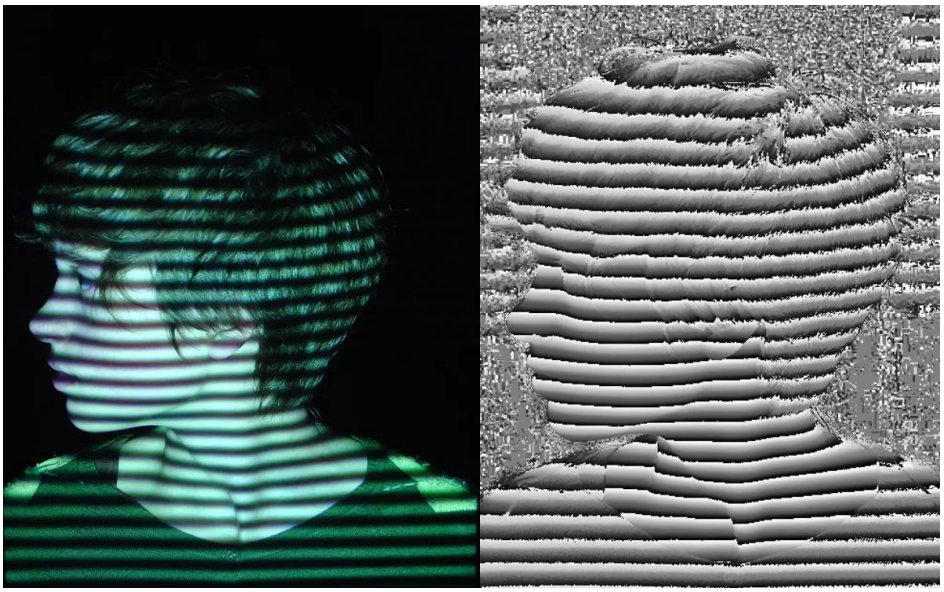


Figure 3.1: *Person has a projected pattern on their face. The Right picture is the post processed*

The light source would be a high power LED's in the Near Infrared spectrum, specifically 940 nm and would illuminate on the DMD. More importantly between those steps, the lenses would focus the emitted light on to the DMD for optimal concentration and minimal loss. The optical system would appear just like the picture below.

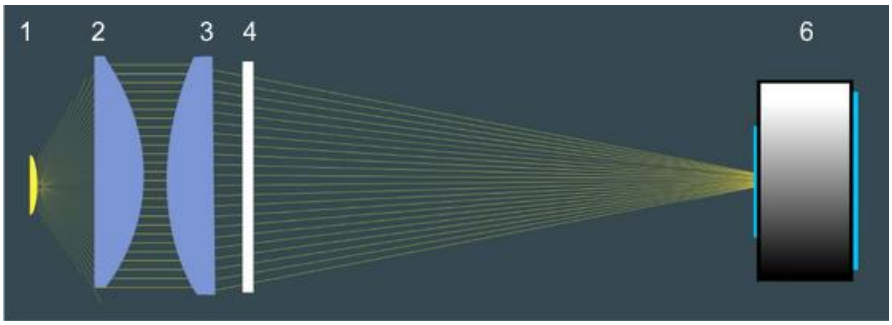


Figure 3.2: *Projection set up*

A projection lens in the 48 mm focal length range is selected to achieve approximately 0.5 m coverage at 2 m based on first-order geometric magnification relationships between the DMD active width and projection distance. Figure 3.2 contains a visualization of the setup. The ESP32 will couple to the camera and visualize our data [20 21 22]. Structured light can operate reliably over a working distance of ~ 0.5 – 2 meters. The light source follows UCFs Laser Safety Manual and is not laser-classified.

However, the major drawback is the Projection Lens cost, computational heavy, and size of the system. For our lamp system, having just the projector would require a more significant amount of space, causing great heat damage to other surroundings because of our application of ~ 5 - 8 hours. Having a cooling system would add additional cost.

The Michelson interferometer technology is another method with the greatest accuracy and has been proven to work to measure depth[14,15,16], it would also require a much larger physical footprint to make usable, increasing the total cost and complexity of making a housing. Additionally, the further cost and complexity required to realize a fully functional Michelson Interferometer to measure depth could lead to the actual measurements we end up taking being virtually indistinguishable from noise.

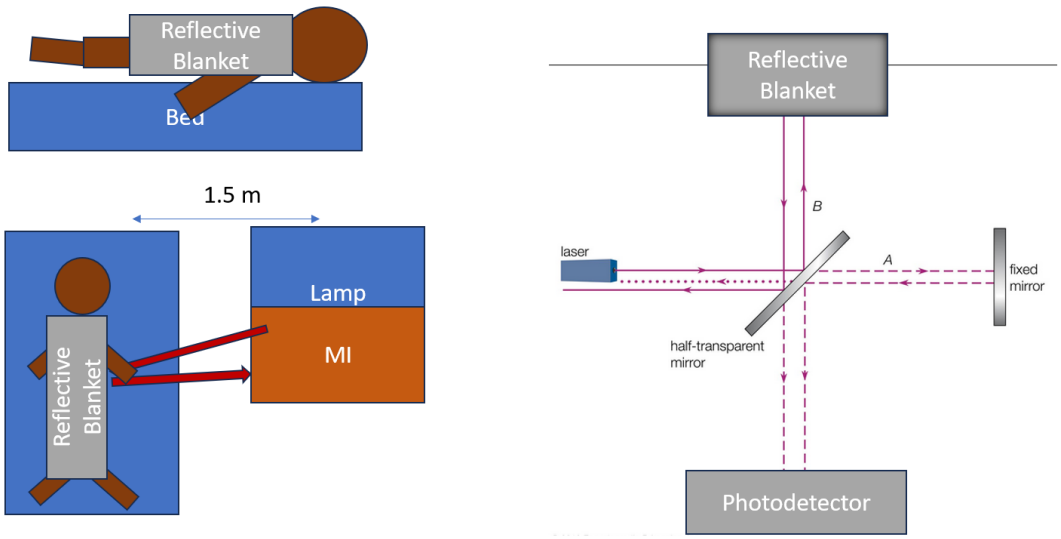


Figure 3.3: *Michelson Interferometer (MI) setup.*

In the Michelson interferometer, the 980 nm laser beam is split into a reference arm and a measurement arm, visualized in Figure 3.3. The reference mirror remains stationary while the measurement mirror moves in response to person detection. When the beams recombine, any change in the measurement mirror position alters the relative optical phase between the two arms. The interferometer converts this phase change into an intensity modulation at the output port. When the person is absent from the bed no signal returns or the phase changes drastically, causing the interference intensity to oscillate rapidly between constructive and destructive states. This oscillating optical power is directed onto a silicon photodiode, which converts the intensity variation into a time-varying current. A transimpedance amplifier converts that current into a measurable voltage, which is then digitized. If the mirror is stationary, the phase difference is constant and the photodiode output becomes essentially a DC level (with only noise). When the mirror moves, the output contains a measurable oscillatory component whose instantaneous frequency is proportional to the mirror's velocity. By sampling the photodiode voltage and analyzing it in short time windows (e.g., 1 ms segments), the presence or absence of this oscillatory component can be identified in the frequency domain. Windows corresponding to absence of a person would lose signal entirely, due to loss in coherency, making the system a great person detection, however, at great cost.

Table 3.5 Depth Measuring System Technology Comparison Table

Category	Structured Light Camera	LiDAR Camera	Michelson Interferometer	Binary Detection (Our System)

Depth Measuring Technology	Triangulation (pattern projection)	Time-of-flight (laser pulse + TDC)	Phase difference (interference)	Time-of-flight (pulse + gated detection)
Depth Accuracy / Resolution	~1–10 mm (at ~1–2 m) Degrades with distance	~5–50 mm (low-cost systems) High-end: <1 cm	< 1 μm (relative) *Not absolute distance*	~5–50 cm (practical) (binary detection focus)
Range	~0.5 – 3 m	~1 – 100 m+	~mm–cm displacement only	~0.5 – 3 m (our design)
Environmental Constraints	Fails in sunlight (>10^4 lux)	Sensitive to reflectivity (10–90%)	Requires vibration stability < λ/10 (~100 nm)	Requires sufficient return power (~μW)
Reliability	Moderate (calibration drift ~1–2%)	High (industrial-grade systems)	Low (alignment tolerance < 10 μm)	Moderate (depends on SNR + electronics stability)
Features	Dense depth map (~10^5–10^6 pixels)	Long range, per-pixel depth	Ultra-sensitive displacement detection	Fast binary detection (<1 ms response)
Cost	$150 – $500 (e.g., Intel RealSense)	$100 – $1000+ (low-cost modules to LiDAR units)	$1000+ (precision optics + mounts)	$20 – $150 (LED/laser + PD+Optics)

The dToF is the ultimate choice for the goals of this project. The subsystem gets rid of a need for a second camera which costs around $70. Additionally, the subsystem would be around the same price as getting a second camera or significantly cheaper depending where I get my lenses. However, the best part of my subsystem is the ability to evolve and eventually become used to reading Respiratory Rate, we probably don't have time this semester to do this. However, if another person picks up this project they would be able to obtain that goal in a significantly short amount of time, since we did the base work. Also less memory would be used from not imaging processing for two cameras. The Michelson Interferometer will not be used at all due to its price being over

+$1000 just for the optics and its complexity. Structured Light is a concern as it would require specially designed lenses, cost and overall size. It is incompatible with a small lamp. Commercially available LiDAR cameras are incredibly expensive, around a thousand dollars, complex to make from scratch, and of limited use. Where the dToF has the ability to evolve.

3.1.3.2 Design

For the measurements to start an emitter like an LED will be needed. However, the design will be constraint by how much power the detector needs and in which manner, as well as simplicity. For example, using an LED (powered with a 2 V forward bias) to hit a wall 1.5 meters away. How much power is received from the reflected surface. With an initial power of 50 mW/sr, and 50% absorption, we'll get 1.11 uW/cm^2 reflected off the wall and 0.28 uW/cm^2 will reach the photodetector.

The photodetector with a minimum power is 100 nW from the linear conversion of the responsivity, as dark current of 100 nA. Being above dark current is great, however, for more sturdier results (avoiding Noise interference) there needs to be a magnitude higher to receive significant results, so approximately ten times.

To obtain higher optical power, a lens system would greatly help. The lens does not increase the emitted power but redistribute it spatially (Concentrates the power into a small area). For example, because my LED has a 25-degree divergence (causing the beam profile to spread over larger distances). With our target goal of 1.5 m the power distribution would significantly decrease (from the distance traveled times the full divergence).
With collimation, distance becomes significantly less relevant and dependent on the focal length. For example, instead of having a factor of 1.5 m over 20.1 mm, the spot size is 3 magnetude smaller. That's a lot of power saved.
Having a small spot size at the end of a 1.5 meter distance, we can concentrate the beam onto a person's torso or main body parts (possibly leading to future endeavors for movement tracking).
Additionally, that is why having a focusing lens on the receiver is greatly helpful as it increases the power density significantly.

Originally the optical subsystem was obtaining direct time-of-flight (dToF) using the TDC7200 to measure signal propagation delay. Using an 8 MHz clock (Think of time resolution) to receive nanosecond resolution. However, through initial testing the TDC-based approach was not suitable for this application. Measured values were inconsistent and heavily influenced by signal noise, limited pulse resolution, and synchronization challenges between the emitter and receiver. These limitations made it difficult to reliably distinguish small changes in distance at the target operating range of 1.5 meters.

Due to these constraints, the optical engineer pivoted toward an intensity-based detection method, with a premaid dToF sensor. The MCU monitors changes in reflected optical power, proving significantly more stability, as it does not require sub-nanosecond timing resolution and is less sensitive to jitter and synchronization errors. Detection is then performed by comparing the measured signal against a baseline reference, allowing the system to determine the presence of a subject based on relative changes in intensity.

The dedicated distance sensor (VL53L1X) was integrated into the system. The sensor provides direct distance measurements to confirm that detected changes in optical intensity correspond to objects within the intended range. By combining intensity-based detection with independent distance verification, the system improves overall reliability and reduces the likelihood of false detections caused by environmental variations.

As the new sensor gates the recorded information, such as when to start sampling. As in, if someone where to put their hand super close to the system, a notification will occur, displaying unable to detect.

Overall, this design pivot reflects a shift while maintaining precision timing-based solution for more robust and practical sensing.

3.1.3.3 Choosing emitter and Detector

The emitter used will be TSAL6400. As it is used in Infrared Remote controllers, reaching high brightness from longer distances while bringing minimal cost, less than 50 cents. The great advantage of low cost is the ability to be affordable in the testing stages. When an LED is burnt it is an easy 50 cent replacement, making it an easy option for buying redundancies.

Compared to lasers such as SPL TL90AT08, VCSEL SPL S1L90H_3 A01. They have a much greater cost and require a more complicated driver, as they need to be current dependent. Despite their heavy optical peak powers ranging from 25 W to 125 W, this will be a greater concern for eye safety and long exposures. Additionally, even though these lasers were meant for greater distances of LIDAR, there are limited duty cycles on these devices making them irritating to work with for slower pulse testing for this project. The divergence is not a limiting factor as the lens would collimate major divergence from the diodes.

Below are comparisons of TSAL6400 and SPL TL90AT08. I focused on these two sources, as they have enough contrast. The other SPL XX are similar enough to be represented as SPL TL90AT08.

Table 3.6 Illumination Technology and Part Selection

Metric	TSAL6400 IR LED	SPL TL90AT03 Laser Diode
Peak optical power	~0.04 W	~65 W
Pulse capability	~15 ns edgesth	1–100 ns optimized
Beam divergence	Wide (±25°)	Narrow (10° × 25°)
Drive current	0.1–1 A	~20 A
Cost	~$0.48	~$17+
Safety	Easy	Requires laser precautions
dToF performance	Low–moderate	High

3.1.3.4 Optical Filters

To decrease as much noise as possible, the optical system will be using an Optical filter for our photodetector. Despite most rooms having lower amounts of infrared light and if a person decides to sleep with some sunlight entering the room, for the worst case scenario, the optical filter would decrease the signal to noise ratio. Additionally, having a room with no windows and having the lights turned on or off does not greatly impact the receiving end as most lights do not produce high contents of Near Infrared Light.

Two types of optical filters are considered: dielectric bandpass filter and an absorptive color bandpass filter. They both have their unique uses, however, we are prioritizing affordability then Quality.

Absorptive color filters are affordable as it's dependent on the material's structure causing absorption of different wavelengths, therefore passing desired wavelength into the system. Due to the rudimentary function of the filter, it has a broader passband. Additionally, absorptive color filters are not greatly impacted by angle of incidence and limit the Field of View of the transmitted signal.

Dielectric bandpass filters are multiple layered thin film on a reflective surface. They behave similar to a Fabry-Perot cavity, allowing central wavelengths to pass through and diminish the non central wavelengths through constructive and destructive interference from the material of the coating.

Table 3.7 Optical Filter Table

Component	Dielectric Bandpass Filter	Absorptive Color Filter
Passband Width (FWHM)	10–50 nm (narrowband)	50–200 nm (broadband)
Peak Transmission	>90–95%	60–85%
Out-of-Band Rejection (OD)	OD 3–6 (10^{-3} to 10^{-6})	OD 1–2 (10^{-1} to 10^{-2})
AOI Sensitivity	High (center wavelength shifts ~1–5 nm per 10°)	Low (minimal spectral shift)
Angular Performance (>20°)	Degrades significantly (band shifts, leakage increases)	Stable performance
Temperature Sensitivity	Moderate (thin-film effects)	Low

Cost (typical)	$20 – $150	$2 – $20
Suitability for 940 nm dToF	High (preferred) – improves SNR	Moderate – cheaper but weaker rejection

While there might be great advantages with Broad Passband. Absorptive color filters are a better choice for our design overall as they do not limit the FOV of our device.

3.1.3.5 Lens Selection

These N-BK7 Plano-Convex lenses are available uncoated or with one of five antireflection coatings (-A, -AB, -B, -C, or -D), which reduces the amount of light reflected from each surface of the lens (see Table 1.1 for quick links to each coating option). Since approximately 4% of the incident light is reflected at each surface of an uncoated substrate, the application of an AR coating improves transmission, which is important in low-light applications, and prevents the undesirable effects (e.g., ghost images) associated with multiple reflections. Having optics that are AR coated on both surfaces is particularly desirable for applications utilizing multiple optical elements. Please see the Graphs tab for additional coating information [60].

The B coating (650–1050 nm) is selected as the optimal choice due to its spectral alignment with the system wavelength and its availability in commercially manufactured aspheric condenser lenses.

Aspheric lenses are designed nob-spherical shaped to reduce aberrations from high diverging light. For lasers, they have a fast angle, horizontal or vertical side of the laser diverges faster, leading to an elliptical or spherically aberrated deviation (The reason for having a fast angle is due semiconductors are normally rectangular, therefore light will spread faster in one axis then another). Aspheric lenses are great for laser diode collimation and fiber coupling, which typically requires a small f-number and large numerical aperture (NA). However, aspheric lenses are made from a single material and suffer from chromatic aberration. As such, they are typically used for monochromatic applications [60].

The NA of the emitter can be approximated as the sine of its half-angle divergence, and a lens with a larger NA is selected to ensure efficient light collection beyond the full-width half-maximum (FWHM) region. While larger aperture lenses can introduce additional aberrations, aspheric condenser lenses mitigate this through their optimized surface profiles, enabling both high collection efficiency and reduced spherical aberration [62]. This will be in great use when shopping for lenses.

Since the TSAL6400 is a wide-divergence infrared LED rather than a laser diode, the optical design prioritizes high light collection rather than diffraction-limited collimation. Therefore, an aspheric condenser (Plano-Convex style of lens) will be used for this project.

The Thorlabs ACL25416U aspheric condenser lens (Ø1", f = 16 mm, NA = 0.79) is used as the reference optical element due to its high numerical aperture and suitability for efficient light collection in near-infrared systems for Table 3.5. The other lenses fall slightly short, either for lack of coating or for being a bit more expensive. For this project, we are minimizing cost while also recognizing the significant value of the characteristics. For this project, it is not urgent to obtain a precision-polished aspheric lens, as it would not improve the quality of our results. However, it is not practical to spend half our budget on a singular lens.

For how we decided on a lens was starting with the limits we have. For example, we require a NA greater than 0.42. Derived from NA ~$nsin(\theta)$, where n is index of refraction of air ($n \approx 1$) and θ = 25, from the spec sheet of the TSAL6400 (NA = $nsin(\theta)$ = (1)(sin(25))). Anything shorter than the 0.42 would pick up significantly less light as the lens diameter would cut off more than 30% of the light.

There is moderate consideration in terms of focal length. For the figure below, we want the laser and photodiode to overlap so that the detector can scan. The focal length does diverge the light overall. As we increase the focal length to reach further distances, the overlap would increase and have a shorter distance of focusing the receiver and transmitter.

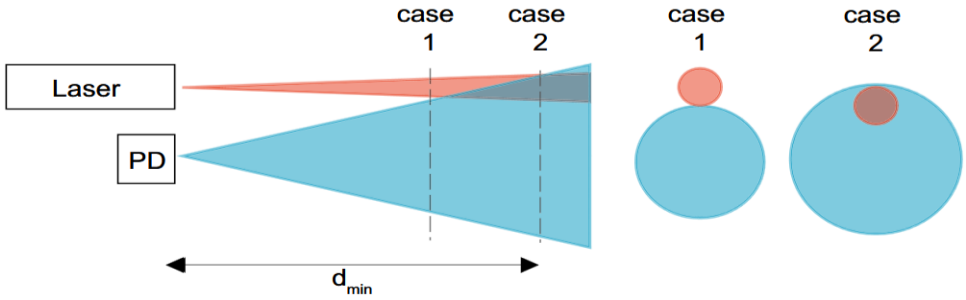


Figure 3.4: *Laser and Photodiode System.*

Table 3.8 Lens selection

Attribute	Edmund Optics 25mm Aspheric Condenser	Newport KPA18AR.16 Aspheric Lens	Generic Aspheric Condenser Lens	ACL25416U-B
Diameter	25 mm	~25 mm	~25 mm	~25.4
Focal Length	16 mm (variant)	16–25 mm	~20 mm	16
Numerical Aperture	~0.79	High (not always specified,	~0.5–0.7	~0.79

		typically lower than molded NA=0.79)		
Coating	Optional / varies	AR (600–1050 nm)	Usually none	ARC: 650-1050 nm
Surface Type	Molded aspheric	Precision polished aspheric	Molded aspheric	Molded aspheric
Optical Quality	High collection efficiency	Excellent (low aberration)	Moderate	High collection efficiency
Cost Range	$43	$455	$22.99	$34.33

3.1.4 Body Temperature Tracking

Body temperature is a critical physiological metric for determining sleep quality and establishing an accurate sleep profile, as natural circadian rhythm fluctuations correlate directly with distinct sleep stages (e.g., a drop in core temperature during NREM sleep)[1]. In a contactless sleep monitoring system, measuring body temperature presents a unique challenge, as traditional wearable or contact-based thermometers are not viable[8]. To achieve this, the Noctisense system relies on a non-intrusive, radiometric thermal imaging approach to securely track the user's physiological state.

Table 3.9 Comparison of Thermal Imaging Technologies

Technology	Operational Principle	Pros	Cons	Selection / Justification
RGB Camera	Captures visible light spectrum.	High resolution, inexpensive.	Requires ambient light; severe privacy concerns. Cannot measure temperature.	Rejected: Violates bedroom privacy requirements and cannot track physiological thermal data.
Near-Infrared (NIR) Camera	Captures near-infrared light (often with an IR LED emitter).	Works in total darkness; moderate cost.	Only captures shapes/movement, not absolute temperature. Identifiable features are still visible.	Rejected: Does not provide radiometric temperature data for sleep stage analysis.
Far-Infrared (FIR) Thermal	Passively detects heat emissions (8-14 µm) from the body.	Captures accurate temperature data; works in absolute darkness; inherent privacy (no facial features).	Lower resolution compared to optical cameras; computationally heavier.	Selected: Ideal for capturing DPG and core temperature safely.

3.1.4.1 Thermal Imaging Module

Below is a comparison of different types of thermal imaging cameras found on the market along with their justification.

Table 3.10 Comparison of FIR Cameras

Component	Resolution	Interface / Processing	Field of View (FOV)	Selection / Justification
Panasonic AMG8833	8x8 (64 pixels)	I2C / Very Low	60° x 60°	Rejected: Resolution is too low to accurately isolate exposed extremities (hotspots) from the ambient bed temperature.
FLIR Lepton (2.5/3.5)	80x60 to 160x120	SPI / High	50° to 160°	Rejected: High cost and excessive resolution.
Melexis MLX90640	32x24 (768 pixels)	I2C / Moderate	Available in 55°x35° or 110°x75° variants.	Selected: The 110°x75° variant captures the whole bed from a bedside lamp enclosure. I2C (Fast Mode) allows easy ESP32 integration.

The MLX90640 Far-Infrared (FIR) thermal sensor array was selected for the Noctisense system to monitor body temperature changes. Specifically, the Distal-to-Proximal Gradient (DPG) and core temperature without requiring physical contact. Unlike standard single-point infrared thermometers, the MLX90640 uses a 32x24 pixel array to function as a low-resolution thermal camera. This 768-point thermal map continuously tracks the temperature of the sleeper's exposed extremities alongside the room's ambient temperature. By using a wide-angle lens (110° x 75°), the sensor captures the entire bed from the lamp enclosure. This approach provides the necessary temperature data to analyze sleep onset without the discomfort of wearable sensors.

For hardware integration, the module uses the I2C protocol. Because the sensor transmits raw data for all 768 pixels, the ESP32's I2C bus is set to Fast Mode (400 kHz) or Fast Mode Plus (1 MHz). This higher clock speed transfers the thermal data fast enough to keep the main program loop running smoothly. By managing these tasks within FreeRTOS, the I2C traffic is kept separate from the UART communication used by the mmWave radar. This separation prevents data collisions and allows both sensors to operate reliably at the same time.

Privacy is a primary concern for a bedroom sensor, and the MLX90640 addresses this physically. With a low resolution of 32x24 pixels, the sensor cannot capture identifiable facial features, clothing, or standard video; it only records heat signatures. This ensures that no sensitive visual information is ever stored on the device. Furthermore, the module is a passive receiver of infrared light and emits no radiation, making it safe for continuous overnight use beside the bed.

Finally, the sensor is well-supported by existing software libraries for the ESP32. Converting the raw MLX90640 data into accurate Celsius readings requires complex math, including reading calibration constants and performing matrix multiplications to adjust for ambient temperature changes. The software processes the full 768-point frame locally, extracting just three key values maximum (user), minimum (ambient), and average temperature. Only these three data points are saved to the SD card, which significantly reduces the file size and keeps the data logging efficient.

collecting information from the sensor. It captures biometric data at a low sampling rate (e.g., 1 to 2 Hz) suitable for slow-changing thermal metrics without obtruding on the user's environment.

3.1.4.2 Image Processing and Noise Filtering

The raw thermal data collected by the MLX90640 must be rigorously processed to extract accurate, body temperature readings. An ESP32 microcontroller is utilized to manage the I2C data acquisition and handle the floating-point computational requirements of this image processing pipeline [42].

First, the system normalizes the absolute pixel values (object temperature) by compensating for varying environmental conditions, using the ambient room temperature measured by the sensor's internal reference. The ESP32 also applies an emissivity correction factor calibrated to 0.98, the standard thermal emissivity of human skin.

Because raw, low-resolution thermal images often contain visual artifacts and hardware-induced noise (such as checkerboard patterns), the ESP32 applies noise filtering using a combination of temporal averaging and a spatial Gaussian blur algorithm. This blurring technique reduces transient image noise and improves the overall definition of the thermal map through interpolation. Following the noise filtration, the system applies a dynamic region-of-interest algorithm. It detects localized physiological hotspots by identifying clustered pixels that exceed a predefined temperature threshold (typically 34°C to 37°C for exposed skin). The peak temperature is extracted and tracked over time to monitor continuous fluctuations in the body temperature throughout the night.

Computationally, this filtering and normalization pipeline requires approximately 7,000 operations per image, translating to roughly 7,000 clock cycles per sample on the ESP32 processor [42]. In

terms of spatial complexity, capturing 768 pixels at a 16-bit depth (2 bytes per pixel) yields a baseline memory footprint of approximately 1.5 KB per thermal frame.

To effectively monitor continuous temperature fluctuations and track physiological hotspots over time, the system employs a 100-frame circular ring buffer. Retaining this historical data increases the total working memory allocation for the thermal tracking module to approximately 150 KB, which remains well within the available SRAM capabilities of the ESP32 [42].

3.1.5 Audio Tracking

3.1.5.1 Primary Methodologies of Measurement

For our long-period overnight audio tracking at a distance of roughly 1.5 m, the primary methodology is to use a small microphone front-end feeding the ESP32 either via its internal SAR ADC (analog microphone module) or via I2S (digital MEMS mic or external audio ADC). In the simplest analog path, an electret or analog MEMS microphone with preamplifier and anti-alias filter drives an ESP32 ADC channel, and the firmware continually samples in short buffers (for example 8–20 ms) and computes a short-term energy or RMS value to compare against a configurable noise-floor threshold; once the signal exceeds this threshold, the firmware switches from low-rate envelope monitoring to continuous PCM capture until the level stays below threshold for a hold-off period. A refinement is to use an I2S digital microphone or I2S ADC module, which offloads analog gain and filtering and streams 16–32-bit samples into ESP32 DMA buffers; this can improve SNR and linearity, and it is commonly operated around 16 kHz mono for voice/noise events while still being light enough for an ESP32 to process in real time. Alternatives that still meet the small, mains-powered constraint include using an external low-power codec with built-in AGC and programmable threshold detection (letting the codec assert an interrupt to the ESP32 only when level crosses a threshold), or using a dedicated sound-detection module that exposes both an analog envelope output and a digital comparator output; in the latter case, the ESP32 uses the digital line purely as a trigger to start ADC or I2S recording, reducing the need for continuous DSP while maintaining event capture.

3.1.5.2 Integration Complexity Considerations

An analog microphone plus preamp into the ESP32's internal ADC has the lowest BOM and PCB complexity but requires careful analog design—biasing, gain setting, and layout to avoid injection from Wi-Fi and switching supplies—and it is more susceptible to nonlinearity and temperature drift in the on-chip ADC. In contrast, a digital I2S MEMS microphone or I2S ADC module simplifies the analog front-end to power and clock routing and moves gain, filtering, and digitization into a well-characterized component, but it adds a few more pins (BCLK, LRCLK/WS, data) and requires I2S driver configuration and DMA buffer management in firmware. Using a full codec (e.g., those supported by Espressif's esp_audio_codec component) pushes complexity into configuration of I2C control registers, multiple I2S data lines, and potentially external crystals, which raises integration effort but brings benefits like integrated microphone bias, AGC, hardware high-pass filters, and multi-rate support that may reduce firmware complexity over time. Finally, adding a separate sound-detection breakout (analog envelope + digital trigger) is mechanically simple but increases wiring and board real estate; it can, however, keep the ESP32 firmware simpler for thresholding, since the external module pre-processes audio into a digital trigger signal that the MCU can treat like a GPIO interrupt.

3.1.5.3 Software Support

On the firmware side, the ESP32 ecosystem offers mature support for both ADC-based sampling and I2S-based audio capture, including Arduino core examples, Espressif IDF drivers, and community code showing how to configure mono I2S microphones at 16 kHz with DMA for continuous recording. For encoding, Espressif's esp_audio_codec framework exposes a unified interface for multiple encoders, allowing an application to register encoders and stream captured samples into a chosen codec with relative ease. LAME MP3 is particularly attractive for balancing high-quality compression with lightweight embedded implementations. Outside the official SDKs, there are optimized ports of the LAME library that map effectively onto microcontroller-friendly integer arithmetic, demonstrating how to compress 16-bit PCM into variable-bitrate MP3 frames while keeping encoder logic compact enough for real-time ESP32 use. For HTTP serving of captured clips, both Arduino (ESPAsyncWebServer, WebServer) and ESP-IDF (esp_http_server) stacks support static file and chunked responses, so serving small, encoded audio clips stored in SPI flash, SPIFFS, LittleFS, or external flash is straightforward once the project defines a storage layout and naming scheme.

3.1.5.4 Throughput/Computational Data Rate & Encoding Considerations

At a typical configuration of 16 kHz mono, 16-bit PCM, the raw data rate is about 256 kbit/s (32 kB/s), which is easily within the ESP32's I2S and memory bandwidth for single-channel overnight monitoring, though storing many hours of uncompressed data would consume substantial flash; with threshold-triggered recording, only short segments are persisted, keeping storage usage manageable. Lightweight compression such as IMA ADPCM reduces 16-bit samples to 4-bit codes (a 4:1 ratio), bringing the data rate down to roughly 64 kbit/s (8 kB/s); reference tables and algorithms show that such encoders run in real time on small MCUs using a predictor, a step-size index, and inexpensive integer operations. For primary threshold detection, computing a simple RMS or average absolute value over blocks of 256–1024 samples per channel is computationally trivial compared with Wi-Fi operations, so the ESP32 can continuously monitor the noise floor while occasionally running the encoder and writing to flash without saturating the CPU, especially when Wi-Fi traffic is mostly idle overnight. If higher-level features are added. E.g., spectral analysis to distinguish coughs, snoring, or environmental noises, then short FFTs (256–1024 points at 16 kHz) are still tractable on an ESP32 core, but they might require careful task prioritization and use of PSRAM for buffering to avoid contention with other sensors and the HTTP stack that serves clips.

3.1.5.5 Energy Budget Considerations

Although this system is mains-powered, the energy budget still affects thermal behavior, component sizing, and EMC; fortunately, typical digital microphones or small audio ADCs draw only a few milliamps, and even when the ESP32 runs its I2S and encoder tasks, overall power consumption remains in the hundreds of milliwatts range, which is modest for an always-on appliance. Using threshold-based event detection rather than continuous storage means the flash memory is written only during meaningful events, reducing write energy and extending flash endurance while allowing the ESP32 to idle or enter lighter sleep states between events if system-level power optimization is later desired. Light compression such as ADPCM or G.711 trades a small increase in CPU activity for a reduction in the number of bytes written over SPI flash or sent

over Wi-Fi; given that flash writes and radio use are relatively energy-expensive, this trade is favorable even on mains power by lowering average current draw and limiting peak radio usage. More aggressive codecs like AAC provide smaller files but require substantially more cycles and often dedicated libraries with higher RAM footprints, which are unnecessary given that overnight clips are short and local storage plus periodic HTTP retrieval impose relatively loose bandwidth constraints in this application.

3.1.5.6 Thermal Management Considerations

Because the total power dissipation of an ESP32 plus a single microphone and associated front-end remains low (typically under 1 W in active Wi-Fi scenarios and significantly lower during idle periods), thermal management within a compact enclosure is more about preventing hot spots near RF and analog sections than about absolute temperature rise. Digital microphones and small audio ADCs themselves dissipate only tens of milliwatts, so they contribute minimally to heating; however, placing them too close to the main ESP32 or to switching regulators may slightly increase self-noise, making careful PCB placement and, if needed, small thermal isolation strategies (spacing, ground pours) more relevant than heatsinking. Since the device will operate overnight in a bedroom environment, using mains power without active cooling, the chosen approach should avoid unnecessary continuous high-CPU audio processing or heavy Wi-Fi streaming; instead, short bursts of encoding and brief HTTP transfers per event help keep average die temperature modest, which in turn supports ADC stability and component longevity.

3.1.5.7 Reliability and Maintainability Considerations

Reliability for overnight monitoring hinges on robust long-duration sampling, encoder stability, and flash wear management; using the ESP32's DMA-based I2S or ADC drivers with ring buffers avoids sample drops, and limiting writes to threshold-triggered clips reduces cumulative erase/write cycles on internal or external flash, supporting multi-year operation. The use of standard codecs like PCM, ADPCM, or G.711, all supported by Espressif's audio codec component, improves maintainability because clips can be decoded easily on a wide range of tools and platforms, and firmware upgrades can evolve the thresholding logic or metadata formats without changing the underlying audio representation. From a hardware perspective, digital MEMS microphones and I2S ADC modules avoid the drift and component aging associated with discrete analog gain stages, providing more stable performance over temperature and time, while the modular nature of the ESP32 audio stack and HTTP server makes it straightforward to log diagnostic metrics (e.g., clipped samples, buffer overruns, flash error counts) for field monitoring and future debugging. Incorporating watchdog timers and simple self-test routines, such as periodically checking that audio envelopes respond to a built-in test tone or noise, which can further enhance reliability without substantial overhead, particularly given the steady availability of mains power.

3.1.5.8 Cost Efficiency

For this use case, an analog microphone plus a handful of passive components and perhaps a low-cost op-amp is the lowest-cost path in terms of BOM, leveraging the ESP32's internal ADC to avoid external converters, and is attractive when the design team is comfortable with analog layout and calibration. Low-cost I2S MEMS microphones and basic I2S ADC boards are slightly more expensive per unit but simplify design and tuning and can reduce development time and risk, which is often more valuable than saving a few cents when integrating into a complex multi-sensor product. In terms of compute cost, simple encoders such as IMA ADPCM or G.711 require no

licensing and fit within the existing ESP32, whereas using more advanced codecs like AAC may involve higher code size, potential licensing considerations, and more RAM—costs that are unjustified for short, event-triggered sleep-monitoring clips served over a local HTTP interface. Overall, a threshold-triggered, lightly compressed mono audio path using an ESP32 plus a small I2S MEMS microphone offers a strong balance of cost, integration effort, and performance for a mains-powered, compact overnight monitoring device intended to coexist with other sensors in a shared housing.

Table 3.11 Audio Recording Technology Comparison

	Analog Mic + ESP32 ADC	**I2S Digital MEMS (e.g., INMP441, SPH0645)**	**External Audio Codec (e.g., MAX9814-style or I2S ADC)**	**Dedicated Sound-Detector Module (Envelope + Digital Trigger)**
Physical interface to ESP32	Single analog pin, bias/amp, RC anti-alias	I2S (BCLK, LRCLK, DOUT) + 3.3 V / GND	I2S (ADC stream) + I2C for control	Analog envelope + 1–2 digital trigger pins
Sample rate / resolution typical	8–16 kHz, 12-bit, software-limited by ADC noise	16–48 kHz, 16–24-bit, factory-characterized SNR	16–48 kHz, up to 24-bit, better THD/linearity	Low-rate envelope (few Hz–100 Hz) + event flag
ADC location	On-chip ESP32 SAR ADC	Internal to MEMS module	On-chip ESP32 SAR ADC	Internal comparator or simple ADC
Power consumption (mic front-end)	Low (few mA if using small analog stage)	Moderate (MEMS + internal ADC/amp, ~1–2 mA)	Moderate–high (full codec, more complex)	Very low (mostly passive + digital logic)

BOM complexity on PCB	Simplest (few passives, 1 channel)	Low (power, decoupling, I2S routing)	Low (just the integrated IC to the ESP32 ADC)	Medium (extra IC, but reduced analog design)
Firmware complexity (sampling)	Must configure ADC, DMA-like reads, handle noise	Use I2S driver, DMA buffers, fixed format	Must manage I2S + I2C registers, possibly multiple modes	Use digital pin as trigger; minimal DSP
Encoder suitability (ADPCM, G.711)	Works; format is 16-bit PCM before encoding	Naturally aligned to 16-bit PCM, easy to encode	Same; PCM from codec to encoder	Cannot encode audio directly; only trigger events
Storage footprint (16 kHz mono)	32 kB/s raw; 8 kB/s with 4:1 ADPCM	Same as analog, but cleaner input to encoder	Same; potential for higher fidelity if needed	Clip length only; no baseline audio stream

Table 3.12 Audio Recording Technology Comparison

	INMP441-based module	**SPH0645-LE-TR-RP-S module**	**ESP32 internal ADC + analog mic (MAX4466)**
Interface	Digital I2S (BCLK, LRCLK, DOUT)	Digital I2S (same interface)	Analog RMS or AC-coupled output to ESP32 ADC

Typical supply voltage	1.8–3.3 V	1.5–3.3 V	3.3–5 V for biased analog mic
Typical sensitivity (dB SPL → digital)	–26 dBFS @ 1 kHz, 94 dB SPL	–26 dBFS @ 1 kHz, 94 dB SPL	Configurable per op-amp gain; often ~1 Vpp at 94 dB SPL
Power consumption	~1.5–2 mA at 3.3 V	~1–1.5 mA at 3.3 V	A few mA (mic + small amp)
Output data format	16-bit, 16–48 kHz on I2S, left-justified	16-bit, 16–48 kHz on I2S, left-justified	Continuous 12-bit ADC values, software-controlled rate
Hardware DSP / filtering	Built-in low-pass filter, no extra computation	Similar built-in filter	None; anti-alias must be analog RC
Matching with lightweight encoder	Output is 16-bit PCM, ideal for ESP-Audio-Codec ADPCM/G.711	Same 16-bit PCM; identical encoder compatibility	PCM-like after ADC; same encoder options

3.1.6 Data Storage

3.1.6.1 Primary Methodology

The Noctisense system relies on a continuous, high-fidelity stream of physiological data generated by the onboard sensor suite, particularly the mmWave radar. Due to the high frequency of data acquisition required for accurate sleep staging, a robust and reliable method of local data storage is imperative. The ESP32 microcontroller, while powerful, possesses limited internal SRAM and flash memory, rendering it insufficient for buffering an entire eight-hour sleep session [42].
To resolve this, a discrete MicroSD card module utilizing non-volatile memory serves as the primary local data repository [20]. This architecture acts as a crucial fail-safe; in the event of a

localized power outage or a disruption in Wi-Fi connectivity, the biometric data remains secure and uncorrupted locally.
Utilizing established hardware modules and communication protocols, the system will capture raw data from the sensors, process it, and structure it into a standardized format. The ESP32 will package this biometric telemetry into JavaScript Object Notation (JSON) format. JSON was selected as the serialization standard due to its lightweight nature and its direct compatibility with the downstream web application and mobile interface. Once packaged, the payload is transmitted from the ESP32 to the SD card module via the Serial Peripheral Interface (SPI) bus for appended storage.

3.1.6.2 Integration Complexity Considerations

The synchronization and asynchronous handling of data streams represent a significant integration challenge. The ESP32 must simultaneously manage incoming interrupts from the radar sensor while executing SPI write commands to the SD card.
A lack of precise synchronization could result in fatal data misalignment, leading to an inaccurate interpretation of sleep architecture. For example, if a respiratory rate of 12 breaths per minute is captured but delayed in storage or overwritten in the buffer by a subsequent reading of 20 breaths per minute, the sleep analysis algorithm will misinterpret the user's physiological state.
Furthermore, every data packet must be tagged with a precise UNIX epoch timestamp at the exact moment of acquisition, rather than at the moment of storage, ensuring chronological integrity regardless of write delays.

3.1.6.3 Software Support

The implementation relies on robust, extensively tested libraries within the ESP32 development ecosystem. The native SD.h and SPI.h libraries will manage the low-level hardware communication, initialization of the FAT32 file system, and the mounting of the drive.
Specifically, the system will utilize a custom implementation of an appendFile(const char* path, const char* message) function. Standard write operations overwrite existing files; therefore, appending is strictly required to continuously build the nightly sleep log without erasing previous hours of data.
For data serialization, the highly efficient ArduinoJson library will be implemented. The system will define a StaticJsonDocument allocated in the ESP32's stack memory to prevent heap fragmentation. The raw sensor variables (e.g., respiratory rate, movement index) will be written to this document, serialized into a formatted string, and passed to the appendFile function. Because JSON is inherently supported by modern web infrastructure, the mobile app can eventually parse this exact file with zero intermediate formatting required.

3.1.6.4 Throughput Data Rate

Ensuring the SPI bus can handle the data throughput without bottlenecking the main execution loop is critical. The default clock rate for SPI communication on the ESP32 is 4MHz
The theoretical maximum throughput can be calculated as follows:

$$\textbf{Throughput=(4,000,000 bits/second)/(8 bits/byte)=500 kB/s}$$

Equation 3.4: *Theoretical Throughput*

Factoring in file system overhead, command latency, and SPI bus transaction times, the practical sustained write speed is roughly 400 kB/s
A single JSON packet containing a timestamp, respiratory rate, and status flags is estimated to be approximately 100 bytes. Even if the system samples and stores data at a highly aggressive rate of 10 Hz, the required data rate is:

100 bytes/packet*10 packets/second=1kB/s

Equation 3.5: *Data Rate*

Since 1kB/s is much smaller than 400 kB/s, the SPI throughput provides a massive overhead margin. The data packets are significantly small enough that the bus speed will not pose a bottleneck, allowing the ESP32 to spend most of its clock cycles in a low-power idle state or handling Wi-Fi tasks.

3.1.6.5 Energy Budget Considerations

While the system is powered via a wall outlet (as a bedside lamp), energy efficiency remains a design parameter to reduce component stress and adhere to green engineering principles. MicroSD cards can consume transient current spikes of up to 100 mA during active write cycles [20].

To optimize the energy budget, the system will avoid writing data byte-by-byte. Instead, the ESP32 will aggregate multiple JSON packets in its internal SRAM buffer. Once the buffer reaches a predefined threshold (e.g., 512 bytes, matching the physical sector size of standard SD cards), a single burst-write operation will be triggered. This reduces the time the SD card spends in its active, high-power state.

3.1.6.6 Thermal Management Considerations

The combination of an ESP32 handling Wi-Fi transmissions, an active mmWave radar, and continuous SD card writes will generate localized heat. Because these components will be housed within the confined enclosure of the bedside lamp, thermal management is necessary to prevent hardware degradation and sensor drift.
Prolonged high temperatures can cause the SD card controller to throttle its write speeds or, in extreme cases, corrupt data. The PCB layout will require adequate thermal vias around the ESP32 and voltage regulators to dissipate heat into the ground plane. Additionally, the physical enclosure design must incorporate passive ventilation slots to allow convective airflow over the MCU and storage modules.

3.1.6.7 Reliability and Maintainability Considerations

Data reliability is paramount for an overnight medical-adjacent device. SD cards using the FAT32 file system are highly susceptible to file corruption if power is removed abruptly during a write cycle.
To ensure reliability, the software will periodically call the file.flush() command. This forces the SD card to update its File Allocation Table (FAT) and physically commit the buffered data to the flash memory, ensuring that even if the lamp is unplugged unexpectedly, only the last few seconds of data are lost, rather than the entire night's file.
For maintainability, the SD card module will be positioned on the PCB such that the card slot aligns with an external access port on the lamp's housing. This allows the user or developer to easily remove the SD card for physical data retrieval, troubleshooting, or replacing the card if it reaches the end of its read/write lifespan.

3.1.6.8 Cost Efficiency

Integrating a physical SD card module is a highly cost-effective solution for local storage. Compared to integrating large-capacity onboard Flash or EEPROM chips directly onto the custom PCB while a standard SPI MicroSD card adapter module costs less than $2.00 in low-volume quantities.

Furthermore, memory cards are ubiquitous commodities. Allowing the user to supply their own standard MicroSD card, or including a low-capacity (e.g., 8GB) card in the initial Bill of Materials (BOM), keeps the overall production cost of the system well within the project's budget constraints while providing vastly more storage than strictly necessary.

3.1.7 Application Server

3.1.7.1 Primary Methodology

Table 3.13 Comparison of Various Hosting Methods

Technology	Operational Principle	Pros	Cons	Selection / Justification
Cloud Server (MQTT / HTTP)	ESP32 streams data to AWS/Google Cloud; App retrieves it from the cloud.	Infinite storage; accessible from anywhere.	High ongoing server costs; severe privacy concerns with cloud health data; fails if home Wi-Fi drops.	**Rejected:** Violates the localized, privacy-first mandate and introduces recurring infrastructure costs.
Bluetooth Low Energy (BLE)	Direct device-to-phone pairing.	Very low power consumption; no router required.	Very slow data transfer rates (~100-250 kbps practical); struggles with transferring large JSON logs.	**Rejected:** Insufficient throughput. Transferring a multi-megabyte nightly sleep log would take too long, frustrating the user.

Localized SoftAP (Wi-Fi)	ESP32 broadcasts its own 2.4 GHz Wi-Fi network directly to the phone.	No internet router required; fast transfer speeds (1.5 - 2.0 MB/s); entirely private local area network.	Higher power draw during active transmission.	**Selected:** Provides the necessary throughput to rapidly sync large JSON files while maintaining total data privacy.

The Noctisense system is designed with a localized, privacy-first data architecture. Rather than continuously streaming sensitive biometric data to a remote cloud server during the night, the system utilizes the onboard ESP32 microcontroller as an independent Application Server [42]. This is achieved by configuring the ESP32 to operate in Software Access Point (SoftAP) mode. In this configuration, the device broadcasts its own localized 2.4 GHz Wi-Fi network, establishing an isolated local area network (LAN) strictly between the physical bedside lamp and the user's smartphone.

The communication framework is built upon the REST (Representational State Transfer) architectural style. Upon awakening, the user initiates a synchronization process via the companion mobile application. The smartphone connects to the ESP32's SoftAP network, and the mobile app acts as the HTTP client. The app transmits an API request to the ESP32's embedded web server, requesting the sleep telemetry compiled during the night.

The ESP32 retrieves the aggregated JSON files stored on the local SD card module (detailed in Section 3.1.6) and serves them as the HTTP response payload. Once the smartphone successfully receives and validates the data payload, the mobile application will display the user's final "Sleep Score" for that session.

3.1.7.2 Integration Complexity Considerations

Deploying a concurrent web server and localized access point on a resource-constrained microcontroller introduces significant integration complexities, primarily regarding memory management and thread execution.

1. Concurrency and Blocking: The ESP32 must handle asynchronous HTTP requests without blocking the primary loop. If the user attempts to sync data while the radar sensor is still actively appending a final JSON packet to the SD card, a race condition may occur. To resolve this, strict Mutex (Mutual Exclusion) locks must be implemented around the SD card read/write functions.
2. Buffer Limitations: The sleep data generated over an eight-hour period will likely result in a file size that exceeds the ESP32's available free heap memory [42] Therefore, the application server cannot load the entire JSON file into RAM before sending it over Wi-

Fi. The integration requires a chunked transfer encoding strategy, where the server reads from the SD card in discrete blocks and streams them sequentially to the client.

3. State Machine Management: The system requires a deterministic finite state machine (FSM) to handle the transition between the "Monitoring State" (Wi-Fi disabled, sensors active, logging to SD) and the "Syncing State" (sensors disabled, Wi-Fi active, serving HTTP requests).

3.1.7.3 Software Support and Implementation

The software architecture relies on a highly efficient embedded tech stack to manage networking and data serialization. To establish the isolated local network, the native WiFi.h library will be utilized to initialize the access point. Standard configurations, such as WiFi.softAP(ssid, password), will be invoked alongside WPA2-PSK encryption to ensure the network remains secure against unauthorized proximity access. Operating on this localized network, the core of the Application Server will be built using the WebServer.h library, with the asynchronous ESPAsyncWebServer under consideration to provide enhanced performance during concurrent loads.

This embedded server will expose specific, stateless REST API endpoints to facilitate seamless communication with the mobile client. For instance, a GET /api/v1/status request will return the current system state, detailing metrics like battery levels and sensor health. Alternatively, a GET request will trigger the system's data retrieval pipeline. Upon receiving this request, the server executes a file pointer operation on the SD card. By leveraging the ArduinoJson library, the system ensures the requested file is properly formatted, attaches the required Content-Type: application/json HTTP header, and streams the payload directly to the client.

On the client side, the user interface will be developed using the Flutter framework, utilizing the standard http package to execute these REST calls. The mobile application will parse the incoming JSON stream, efficiently decoding the transmitted arrays of timestamps and respiratory rates for further analysis.

3.1.7.4 Throughput Data Rate

The IEEE 802.11 b/g/n physical layer on the ESP32 supports theoretical data rates up to 150 Mbps. However, practical throughput in an embedded application server is ultimately bottlenecked by the SPI read speed of the SD card and the processing overhead required by the TCP/IP stack. While theoretical models factor in both hardware read limits and protocol latency, empirical testing demonstrates that an ESP32 serving files directly from an SD card typically achieves a maximum sustained HTTP transfer rate of approximately 1.5 MB/s to 2.0 MB/s.

To evaluate the system's performance, the total anticipated data volume must be established. Assuming the Noctisense system monitors an eight-hour sleep session and generates data at a frequency of 1 Hz, with an estimated payload of 100 bytes per packet, the total accumulated file size for a single night is approximately 2.88 MB. Operating at the lower-end effective throughput of 1.5 MB/s, this entire data payload will successfully transfer to the user's smartphone in less than 2.0 seconds. This rapid data synchronization ensures a seamless and highly responsive user experience immediately upon waking.

3.1.7.5 Energy Budget Considerations

The ESP32's internal Wi-Fi radio is the most power-intensive component of the system. During active transmission, the RF power amplifier can draw transient current spikes up to 240 mA

To maintain strict energy efficiency and minimize thermal output during the sleep cycle, the Application Server will not run continuously. The system employs a "Radio Silence" protocol during the monitoring phase. The WiFi.mode(WIFI_OFF) command will be executed when the user initiates a sleep session.

The SoftAP will only be initialized when the user physically interacts with the device in the morning (e.g., pressing a capacitive wake button) or when the internal Real-Time Clock (RTC) detects the scheduled wake-up time. By constraining the web server's uptime to only the brief synchronization window, the overall power footprint of the device is drastically reduced.

3.1.7.6 Thermal Management Considerations

When the ESP32 operates in SoftAP mode and handles heavy HTTP traffic, the SoC generates considerable localized heat. While the active transmission window is brief (under 5 seconds for a full data sync), repeated connection attempts or an extended connection session could elevate the internal enclosure temperature.

The PCB layout must strategically place the ESP32 module away from the sensitive mmWave radar and any ambient temperature sensors to prevent the application server's thermal dissipation from skewing the sleep environment telemetry.

3.1.7.7 Reliability and Maintainability Considerations

Because the primary interaction within the Noctisense system occurs over a wireless protocol, the architecture must robustly handle connection instability to ensure data integrity. In the event that the Wi-Fi connection drops mid-transfer, the Flutter application is programmed to recognize a truncated JSON payload, detecting faults via a mismatched content-length header or a malformed JSON closure. To further guarantee reliability, the REST API incorporates a strict data verification acknowledgment sequence. Once the smartphone successfully downloads and validates the complete data array, the mobile application transmits a POST /api/v1/sync_confirm request back to the ESP32. Regarding storage maintenance, the microcontroller will only mark the specific file on the SD card as "Synced" upon receiving this HTTP 200 OK confirmation. This fail-safe mechanism ensures that no physiological data is permanently deleted or overwritten until it has been safely and entirely duplicated on the user's mobile device. Should the connection fail at any point during this handshake, the data remains securely stored on the local SD card, ready for the next synchronization attempt.

3.1.7.8 Cost Efficiency

Hosting the application server directly on the ESP32 provides massive cost efficiencies for the final product lifecycle of the Noctisense system. Primarily, this approach eliminates ongoing cloud infrastructure expenses. By utilizing a localized REST API and SoftAP, the project entirely circumvents the need for dedicated remote web hosting platforms, such as AWS EC2 instances,

Google Cloud, or MongoDB Atlas databases. This strategic design choice removes the recurring subscription costs and server maintenance overhead that typically burden connected IoT devices. Furthermore, the architecture benefits from significant hardware consolidation. Because the ESP32 serves dual roles as both the primary microcontroller for sensor data acquisition and the network application server, it eliminates the need to populate the custom PCB with a secondary, dedicated networking coprocessor. This dual functionality effectively reduces the overall Bill of Materials (BOM) cost and simplifies the manufacturing and assembly.

3.1.8 User Application

3.1.8.1 Primary Methodology

The Noctisense mobile application serves as the primary and exclusive interface between the user and the bedside hardware. Because the physical lamp unit is designed to be unobtrusive and lacks a built-in graphical display, the mobile application acts as the central hub for data visualization, system configuration, and physiological tracking. Below is a decision table with possible frameworks that could have been used to develop the User Application.

Table 3.14 Comparison of Various Application Frameworks

Technology	Architecture	Pros	Cons	Selection / Justification
Native (iOS / Android)	Swift for iOS Kotlin for Android.	Maximum performance and deep hardware access.	Requires two entirely separate codebases doubling development and debugging time.	Development overhead is too high for the project timeline.
React Native	JavaScript bridge rendering native UI components.	Single codebase and massive web-developer community.	The JS bridge can introduce performance bottlenecks when parsing large JSON data arrays.	Parsing efficiency is critical for handling the ESP32's dense sleep telemetry.

Flutter	Dart codebase compiled directly to native ARM code via custom rendering engine.	Fast execution, strictly consistent UI across platforms, excellent background parsing.	Slightly larger compiled app size.	The cross-platform approach halves development time while maintaining the native performance required to smoothly render complex data visualizations. Previously used by members on the team

The application is developed using the Flutter framework, an open-source UI software development kit created by Google. The primary engineering justification for selecting Flutter is its ability to utilize a single Dart codebase to compile natively for both Android and iOS devices. This cross-platform approach significantly reduces software development time, streamlines the debugging process, and ensures a perfectly consistent user experience regardless of the user's mobile operating system.

A foundational pillar of the Noctisense application is its offline-first, privacy-centric architecture. The app is designed to operate entirely independent of the internet. Instead of communicating with external cloud servers or third-party databases, the smartphone connects directly to the bedside lamp's localized Wi-Fi network. This closed-loop system reinforces the project's strict privacy constraints, ensuring that highly sensitive biometric data such as core body temperature and movement profiles never leaves the physical confines of the user's bedroom.

3.1.8.2 User Interface and Experience (UI/UX) Design

The user interface is structured around a "morning-review" paradigm. The software architecture operates on the assumption that users will primarily interact with the application immediately upon waking up, a time when cognitive load should be minimized, and visual sensitivity is high [4].

The main dashboard prominently displays a daily "Sleep Score," a single composite metric calculated by the hardware that grades the overall quality of the night's rest. Below this primary score, the interface utilizes standard Flutter charting packages (such as FL Chart) to render detailed time-series graphs. These visualizations map the user's core body temperature estimates and physical movement events across the entire sleep session. To accommodate the bedroom environment, the application utilizes a dark-mode color palette. This design choice prevents the harsh glare of a bright smartphone screen from disrupting the user's visual adjustment or their circadian rhythm if they check the application during the night or early in the morning.

To ensure the application is universally usable, strict considerations have been integrated into the UI design to meet modern accessibility standards. The following ten accessibility elements form the core design philosophy of the Noctisense interface:

1. Provide sufficient contrast between foreground and background: In the dark-mode environment, text and critical data points utilize high-luminosity colors (such as pure white or high-visibility yellow) against deep gray backgrounds, ensuring a contrast ratio that exceeds standard Web Content Accessibility Guidelines (WCAG) AA requirements [21] .
2. Do not use color alone to convey information: When displaying the sleep graphs, the difference between "Awake" and "Deep Sleep" is not merely represented by red and green colors. The interface also employs different line textures, icon markers, and explicit text labels so that users with red-green color vision deficiencies can easily interpret their data.
3. Ensure that interactive elements are easy to identify: All buttons, sliders, and synchronization triggers feature distinct borders, subtle elevation shadows, and minimum touch-target sizes of 48x48 logical pixels. This is particularly crucial for users navigating the app while groggy in the morning.
4. Provide clear and consistent navigation options: The application utilizes a persistent bottom navigation bar. Whether the user is viewing their daily score, accessing historical trends, or configuring the lamp's settings, the navigation hierarchy remains predictable, preventing the user from getting lost in nested menus.
5. Ensure that form elements include clearly associated labels: Any configuration menus, such as setting the internal clock of the lamp or adjusting sensor sensitivity, feature explicitly labeled text fields rather than relying solely on disappearing placeholder text.
6. Provide easily identifiable feedback: When the user initiates a data synchronization with the lamp, the application provides immediate visual feedback via a loading indicator, followed by localized haptic feedback (a physical vibration) from the phone to confirm the data transfer was successful.
7. Use headings and spacing to group related content: The dashboard utilizes strong typographic hierarchy. Large headers separate distinct data sets (e.g., "Thermal Data" versus "Movement Data"), and generous padding is used to create breathable zones, reducing visual clutter.
8. Create designs for different viewport sizes: Utilizing Flutter’s MediaQuery classes, the application features a responsive layout. The charts and text elements automatically scale and restructure themselves to maintain legibility whether the app is running on a compact smartphone or a large-format tablet.
9. Include image and media alternatives in your design: All graphical icons and visual indicators include underlying "semantic labels." This ensures that if a visually impaired user utilizes a screen reader (like Android TalkBack or iOS VoiceOver), the application will verbally describe the sleep charts and scores accurately.
10. Provide controls for content that starts automatically: To respect the user's battery life and sensory load, the application features no auto-playing animations or automatically scrolling carousels. All deep-dive interactions with the data graphs require deliberate user input.

3.1.8.3 Data Synchronization and API Integration

Because the Noctisense system does not use Bluetooth or cloud servers, the application must manage a localized networking sequence to retrieve the night's data. When the user taps the "Sync" button, the application executes a series of network commands. First, the app prompts the smartphone to verify its connection to the ESP32’s specific SoftAP network.

Once the local area connection is verified, the application utilizes Dart's standard HTTP packages to execute a REST API GET request directed at the lamp's IP address. The incoming payload from the lamp is a structured JSON file containing the time-stamped biometric arrays generated by the thermal and movement sensors over the course of the night. Because JSON parsing can be computationally expensive on a mobile device, the Flutter application utilizes asynchronous execution (specifically, Dart Isolates) to parse the JSON stream into local Dart objects on a background thread. This ensures the user interface remains completely smooth and responsive during the download. To guarantee data integrity, the app immediately sends an HTTP 200 OK acknowledgment back to the ESP32 upon successful parsing. This precise handshake tells the microcontroller that the data was received intact, and it is safe to mark the local SD card files as synced.

3.1.8.4 Data Display and Interface Binding

By design, the data processing and Sleep Scoring Algorithm are executed onboard the ESP32 microcontroller. Therefore, the mobile application functions exclusively as a dedicated data presentation layer rather than a primary data processor. When the smartphone synchronizes with the bedside lamp, the received JSON payload already contains the finalized calculations. This includes the composite 1-to-100 Sleep Score, the calculated durations of specific sleep stages, and the aggregated temperature averages. The Flutter application simply parses these pre-calculated values and maps them directly to the user interface variables for visual rendering.

This specific architectural decision provides several distinct engineering advantages. Primarily, it significantly lowers the computational overhead on the user's smartphone, reducing the application's active memory footprint and battery consumption during use.

3.1.8.5 Remote Database Retrieval and Historical Tracking

To provide users with meaningful insights into their long-term sleep habits, the Noctisense system supports historical data tracking. However, to adhere to the strict local-storage constraints, the historical database is maintained entirely on the ESP32’s SD card rather than on the mobile device's internal storage.

When a user navigates to the "History" or "Trends" tab within the Flutter application, the app does not pull from a local SQLite database on the phone. Instead, it generates a parameterized API request to the lamp (e.g., GET /api/history?range=7days). The ESP32 accesses the SD card, compiles a summary of the requested date range, and transmits it back to the phone. The application temporarily holds this data in the device's volatile RAM to populate the weekly or monthly trend charts. Once the user closes the application, this historical data is cleared from the phone's memory.

This remote-retrieval methodology guarantees that the mobile app maintains a near-zero storage footprint on the user's smartphone. Furthermore, it ensures that the physical bedside lamp remains the sole source for all physiological data, creating a highly secure, decentralized approach to personal health tracking.

3.1.8.6 Android Application

The Noctisense system requires a robust, intuitive mobile interface to translate raw physiological data into actionable sleep metrics. To achieve this, the software architecture utilizes a mobile application built entirely on the Flutter framework. Flutter is an open-source UI software development kit that uses the Dart programming language. The primary engineering justification for selecting this specific framework lies in its declarative, widget-based architecture, which allows for the rapid construction of complex user interfaces. Because the Noctisense system relies heavily on rendering detailed time-series graphs (such as respiratory rates and thermal fluctuations) and dynamic daily scores, Flutter's highly optimized Skia rendering engine ensures that these visual elements operate smoothly at native performance speeds.

Beyond the technical merits of the framework itself, the selection of Flutter was heavily influenced by the team's prior development experience. In the context of a time-constrained senior design project, minimizing the learning curve for new software tools is critical to maintaining development velocity. The team's familiarity with Dart's syntax allowed the software phase to immediately enter active prototyping rather than stalling in a prolonged research and tutorials phase. Furthermore, Flutter's "hot reload" feature proved invaluable. This capability allows developers to inject updated source code directly into the running application, instantly reflecting UI changes and bug fixes without requiring a full application restart or recompilation. This drastically reduces the time required to fine-tune the visual layout and test the REST API synchronization logic with the ESP32.

While Flutter is inherently a cross-platform framework capable of compiling from a single codebase into native applications for both Android and iOS the deployment strategy for the Noctisense prototype is strictly "Android-First." The decision to target the Android operating system exclusively during the development and testing phases was driven by the necessity to bypass the severe logistical, cryptographic, and hardware barriers associated with the Apple iOS ecosystem.

Deploying an application to an iOS device, even strictly for local testing, introduces significant administrative overhead. Apple's ecosystem requires strict cryptographic signing for all application packages (IPA files). To compile and push an app to a physical iPhone, the developer must navigate Xcode provisioning profiles, generate specific development certificates, and register the physical device's Unique Device Identifier (UDID) with an active Apple Developer account. Furthermore, this entire compilation process is hardware-locked; it requires a macOS environment running the Xcode integrated development environment (IDE).

Conversely, the Android operating system offers a highly accessible, open ecosystem that is uniquely suited for rapid hardware-software integration testing. Android allows developers to easily compile the Flutter codebase into an Android Package Kit (APK). This APK can be transferred to any standard Android device and sideloaded instantly, entirely bypassing the Google Play Store and the need for paid developer accounts or complex cryptographic certificates. By enabling "Developer Options" and utilizing the Android Debug Bridge (ADB) via a simple USB connection, we are able to push updates, monitor system logs, and debug API network errors in real time.

Furthermore, developing primarily on Android simplifies the localized networking requirements of the Noctisense system. Because the application must disconnect from the user's standard home Wi-Fi and connect directly to the ESP32's SoftAP network to retrieve the nightly JSON files, managing these network state changes is generally more permissive on Android. Apple iOS imposes highly restrictive sandboxing and background execution limits on local network scanning, which often requires complex workarounds and special user permissions just to establish the local handshake. By testing on Android first, the team can validate the core data synchronization logic, ensure the SD card parsing works correctly, and confirm the sleep scoring algorithms are accurate without fighting OS-level network restrictions.

Ultimately, this Android-First strategy protects the project timeline while preserving future scalability. Because the underlying codebase is written in Flutter, the application is not permanently locked into the Android ecosystem. Once the core functionality is completely validated, the UI is finalized, and the hardware communication is proven stable, the exact same Dart codebase can eventually be compiled for iOS. This ensures that the immediate prototyping phase remains agile and unhindered by Apple's deployment logistics, while the final Noctisense product retains the capability to serve a multi-platform consumer base

3.1.9 AC-DC Power Supply Considerations

Noctisense will have its power system in two main parts and this is the first. An external AC-DC conversion stage will generate a stable and regulated DC supply voltage that will allow noctisense to be much simpler and safer than needing to integrate high voltage AC circuits that are far less stable. Also, based on the fact that the system will need to comfortably handle at least 10W, a higher DC voltage after the conversion means much less current draw to provide the same power [28].

3.1.9.1 Power Integrity and Noise Considerations

Power integrity is a vital part of the Noctisense system due to the presence of both high frequency switching regulators and noise sensitive analog and RF components. Switching regulators are very efficient but introduce ripple voltage and high-frequency noise from rapidly switching MOSFETs. This noise can propagate through the power distribution system and negatively affect the performance of sensitive components such as the mmWave radar module, thermal imaging sensor, and the optical detection circuitry.

To mitigate these effects, careful consideration is given to both the magnitude and frequency characteristics of power supply voltage ripple. The output ripple from switching regulators is minimized by the use of output capacitors and inductors, as well as through the use of additional filtering stages. However, even with all of this, the inherent limitations of switching regulators in achieving very low noise performance linear regulators had to also be used as post regulation stages for the sensitive analog and RF components. The use of low dropout linear regulators after switching buck converters provides significant reduction of high frequency noise due to their high power supply rejection ratio. This hybrid regulation approach ensures that high efficiency conversion is achieved to reduce heat while simultaneously providing clean and stable voltage rails for sensitive measurement components. Localized decoupling capacitors further suppress voltage

fluctuations and maintain supply stability during the dynamic conditions of real world application [38], [47].

3.1.9.2 Grounding Considerations

Due to the mixed signal nature of the Noctisense system including digital processing, RF sensing, and analog signal conditioning, a shared grounding plane could cause significant degradation of system performance over time. In order to address this, the system implements a partitioned grounding scheme that separates analog and digital ground domains. The analog ground plane is dedicated to the sensitive components such as the photodiode transimpedance amplifier, thermal sensor, and the mmWave radar module, while the digital ground plane supports the ESP32 microcontroller, SD card, and other digital circuitry. This separation further prevents switching noise and digital return currents from interfering with the sensitive components. A third ground plan is used to connect the analog and digital ground planes and serves as a common reference ground for the system, minimizing ground loops and reducing the potential for noise generation between different subsystems. [42]

3.1.9.3 Efficiency and Thermal Considerations

Thermal management and power conversion efficiency are very closely related as inefficient power conversion results in excess heat generation, which can negatively impact both system reliability and sensor accuracy. This is why switching regulators are selected as the primary means of voltage conversion due to their high efficiency, typically around 85%. This significantly reduces power dissipation compared to only linear regulation as linear regulators shed their excess voltage as heat which is especially important given the enclosed lamp style housing of the system. Linear regulators, while less efficient, are still needed]for noise sensitive applications where low output ripple is required but the power dissipated by these regulators is minimized by ensuring a very small voltage differential between input and output voltage thereby reducing thermal stress. Component placement within the enclosure is also very important as the heat generating switching regulators need to be positioned away from the thermal sensor to prevent measurement bias [32].

3.1.9.4 Reliability and Safety Considerations

The Noctisense power system is designed with reliability and safety as its primary considerations ensuring consistent and reliable operation over long periods of time and protection against common electrical faults. The use of an external AC-DC adapter provides inherent electrical isolation from the AC mains significantly reducing the risk of voltage hazards within or near the device. The protection circuitry is implemented at the system input to safeguard against abnormal operating conditions and human error. These protective measures prevent damage to critical components and enhance the overall longevity of the system [45].

3.1.9.5 Design Tradeoffs and Justification

The design of the Noctisense power system involves several tradeoffs between efficiency, complexity, cost, and performance. One of the primary decisions is the use of an external AC–DC adapter instead of an internal AC–DC conversion stage or a USB-C-PD solution. While an internal solution offers better integration, it introduces too many significant safety risks and unnecessarily increases design complexity while the external adapter mitigates these risks and provides a reliable

and regulated DC power source. This source then provides power to the multi-rail power distribution system which serves to further enhance system stability by isolating the noise sensitive components from high current digital components. This approach does increase design complexity and cost, but it provides improved measurement accuracy, reliability, and thermal regulation which are all essential for Noctisense's intended application.

Table 3.15 Power Budget

Component	Supply Voltage (V)	Maximum Current (mA)	Maximum Power (W)	Description
ESP32-S3 MCU	3.3	500	1.65	WiFi transmission + processing peak
24 GHz mmWave Radar	5.0	120	0.60	Active motion/respiration tracking
Thermal Sensor (MLX90640)	3.3	40	0.13	High refresh mode operation
Microphone + ADC	3.3	60	0.20	Continuous audio monitoring
SD Card	3.3	100	0.33	Write-cycle current spikes
NIR Laser Driver	5.0	150	0.75	Pulsed operation (worst-case duty)
Photodiode + TIA	3.3	30	0.10	Analog front-end circuitry
Indicator LEDs / Misc.	3.3	20	0.07	Status indicators and logic
Total (Load Only)	—	—	3.83 W	—
Total	—	—	5 W	Heavily overestimated at +30% for loss

Supply Capability needed for robustness	—	—	10W	Leave two times the total for headroom and longevity.

3.1.9.6 Linear Power Supplies

Linear power supplies utilize a step-down transformer followed by rectification, filtering, and voltage regulation to produce a stable DC output voltage. These systems have extremely low output noise and minimal electromagnetic interference, which makes them well-suited for the sensitive mmWave radar module and optical components. However, the disadvantages of linear power supplies include poor efficiency due to dissipation as heat within the regulator, large size due to low-frequency transformers, and higher weight. For a compact, mixed-signal embedded system like Noctisense, these limitations make linear supplies not the most effective choice for this project despite its low noise attributes [28].

3.1.9.7 Switch Mode Power Supplies

Switch mode power supplies (SMPS) convert oscillating AC voltage to stable DC using high-frequency switching techniques, achieving significantly higher efficiency enabling smaller and lighter designs because of a lack of thermal issues to contend with. This method allows for efficient voltage conversion but SMPS introduce high-frequency noise and electromagnetic interference, which can interfere with the sensitive components in Noctisense. Proper design and filtering are required to minimize the noise generated by the supply. While SMPS provide a compact and efficient solution, implementing a custom SMPS adds complexity and possible safety issues in working with high voltage AC mains [28].

3.1.9.8 External AC-DC Adapters

External AC–DC adapters use a complete SMPS but put it within a certified, isolated enclosure, providing a regulated DC output without being exposed to the high voltage AC mains. This significantly reduces the design complexity and greatly increases the safety of Noctisense while providing a stable DC voltage source for the system to use. The main drawbacks of these adapters are a limited customization of the output and a lack of being able to aesthetically integrate it into the housing. In spite of these drawbacks for a project such as Noctisense, external adapters offer a low-risk, reliable solution that is simple to implement and robust [45].

3.1.9.9 Embedded AC-DC Modules

Embedded AC–DC converter modules offer a more integrated solution by putting SMPS functionality on a PCB-mountable package. They provide a compact solution and can be placed directly within the device enclosure. Despite this the implementation of such a power supply would require very careful PCB layout to maintain proper creepage and clearance distances and to ensure proper thermal management of supply to keep temperatures tolerable. These challenges increase design complexity and would likely require additional shielding components to maintain stable operation [28].

3.1.9.10 USB-C Power Delivery

USB-C-PD is a modern AC–DC conversion tactic that changes both voltage and current in a dynamic way between the power source and device. It enables operation at multiple voltage levels like 5 V, 9 V, and 12 V, and can deliver high power through minimal wiring. While this flexibility is nice to have, USB-C-PD introduces additional challenges like needing to implement a communication protocol and hardware controller that, for systems like Noctisense, is not justified or even necessary to realize Noctisense [28].

3.1.9.11 Power Supply Selection

For Noctisense the best option of power supply is an external regulated AC–DC adapter. They offer the best balanced combination of high efficiency, safety, and implementation simplicity while avoiding any high voltage exposure or concerns inside the device enclosure. By supplying a stable low to moderate DC voltage the adapter enables efficient voltage regulation for multiple rails using switching and linear regulators. This ensures the power systems reliability and ability to provide a steady clean source of DC power without needing to generate and deal with a large amount of heat inside the enclosure. The external adapter established the power system on a very stable foundation on which the Multi-Rail Distribution system will rely on to function properly. The final Part choice for Noctisense is the Jameco ReliaPro 12V, 2A wall adapter, with a more expensive backup option being the CUI SDI24-12-U 12 V, 2 A. This will provide electrical isolation from the AC mains and reduce the input current requirement needed because of the higher voltage. This configuration also leaves plenty of headroom for power and current draws so this supply will not be stressed providing the needed power.

Table 3.16 AC-DC Power Supply Technology Comparison

Technology	Description	Pros	Cons	Selection / Justification
Linear Power Supply	Transformer-based AC-DC conversion with linear regulation	Very low noise, minimal EMI, simple design	Large size, heavy, low efficiency	Not selected due to efficiency, size, and thermal concerns
Switch Mode Power Supply (SMPS)	High-frequency switching conversion of AC to DC	High efficiency, compact, lightweight	High noise, complex, safety concerns with AC mains	Not selected due to complexity and safety risks
External AC-DC Adapter	Prebuilt isolated SMPS providing regulated DC output	Safe, low complexity, reliable, no internal AC handling	Limited customization, external form factor	Selected for safety, simplicity, and reliable regulated DC output
Embedded AC-DC Module	PCB-mountable AC-DC converter	Compact, integrated	Requires careful PCB layout, thermal and safety concerns	Not selected due to complexity and safety concerns
USB-C Power Delivery	DC power via USB-C protocol	Flexible voltage levels, modern interface	Requires controller /communication protocol	Not selected due to complexity

Table 3.17 AC-DC Power Supply Component Comparison

Part Number	Type	Output Voltage (V)	Output Current (A)	Power Rating (W)	Efficiency (%)	Selection / Justification
Jameco ReliaPro 12V Adapter	AC-DC Wall Adapter	12	2.0	24	85	Selected for optimal voltage output, current draw, and efficient buck conversion
CUI SDI24-12-U	AC-DC Wall Adapter	12	2.0	24	88	Backup option with higher reliability and efficiency, but higher cost
Mean Well GST18A05-P1J	AC-DC Wall Adapter	5	3.6	18	88	Not selected because of increased current demand and reduced system efficiency of using lower voltage.
Mean Well GST36A15-P1J	AC-DC Wall Adapter	15	2.4	36	89	Not selected because the increased voltage drop across the regulators would generate more heat

3.1.10 Switching Regulation Technologies

Switching regulators are power conversion systems that regulate voltage by rapidly switching a transistor or MOSFET on and off, transferring energy through inductors, capacitors, and sometimes transformers. Unlike a linear regulator, they are known for high efficiency and minimize energy dissipation by instead storing and transferring energy in reactive components. This makes them the general case solution in modern electronics, especially when efficiency, and heat generation are very important.

3.1.10.1 Non-Isolated DC-DC Converters

Non isolated converters do not provide isolation between input and output and are widely used in applications that are low voltage and low risk.

Buck (Step Down) Converters:
A Buck regulator reduces input voltage to a lower output voltage and includes asynchronous diode rectification and synchronous MOSFET rectification.

Boost (Step Up) Converters:
Boost converters increase the input voltage to a higher output voltage and include synchronous boost converters and interleaved configurations for improved efficiency and reduced ripple.

Buck Boost Converters:
This type can produce output voltages either higher or lower than the input. Variants include inverting and non inverting designs as well as much more complex designs that are far outside of the scope of this project.

Charge Pumps (Switched Capacitor Converters):
These converters use capacitors instead of inductors for energy transfer and are typically used in low power or low space applications and include voltage doublers, inverters, and fractional conversion ratios.

3.1.10.2 Isolated DC-DC Converters

Isolated converters use transformers to provide input and output isolation and voltage scaling. These are essential in high voltage applications

Flyback Converters:
They are commonly used in low power applications and have a low component count making them very cheap, small, and still containing an isolated input and output. However, with all these advantages they do still come with high noise and electromagnetic interference as well as lower efficiency as their output power increases [29].

Forward Converters:
These are more efficient than flyback in higher-power applications, and can contain multiple transformer windings allowing them to provide both higher and lower voltage outputs at the same time.

3.1.10.3 Resonant and Soft Switching Converters

Resonant converters are converters that reduce the losses and electromagnetic interference that switching generates by ensuring switching occurs under the most favorable voltage or current conditions possible. These conditions being either to switch when the voltage is zero or the current is zero. [28]

Table 3.18 Switching Regulation Technology Comparison

Technology	Description	Pros	Cons	Selection / Justification
Buck Converter	Steps down DC voltage to a lower regulated output	High efficiency, simple design	Cannot increase voltage	Selected as primary method for generating 7V and 5V rails from 12V input
Boost Converter	Steps up DC voltage to a higher level	Useful for low input voltage systems	Higher ripple, reduced stability, unnecessary for system	Not selected since input voltage needs no step up
Buck Boost Converter	Can step voltage up or down	Flexible output voltage	Increased complexity, higher noise	Not selected due to the fixed voltage needs of the system and increased complexity
Charge Pump	Uses capacitors for voltage conversion	Compact	Limited current capability, and bad efficiency	Not suitable for the system's power requirement.
Flyback Converter	Isolated converter using transformer	Provides isolation, simple topology	High noise, lower efficiency at higher loads	Not required due to existing isolation
Forward Converter	Transformer based isolated converter	Higher efficiency than flyback, supports multiple outputs	More complex design, larger footprint	Overly complex for the system

Resonant Converter	Soft-switching converter	Very high efficiency, low noise	Very complex design	Not selected due to unnecessary complexity

3.1.11 Linear Regulation Technologies

Linear regulators maintain a constant output voltage by operating a transistor in its active (linear) region, which effectively dissipates excess voltage as heat. Although these regulators are far less efficient than switching regulators, they offer low noise and low interference which is desperately needed by the mmWave radar module for proper use and accurate readings.

3.1.11.1 Series Linear Regulators

Series regulators place a controllable element between the input and output and use this element to control the voltage output steadily and reliably.

Classic Series Regulators:

These regulators use bipolar junction transistors or MOSFETs to regulate voltage using a feedback control loop. These regulators are very stable and are good at filtering out input noise but shed a lot of energy as heat and often require heat sinks [28].

Low-Dropout Regulator (LDO):

LDOs are Linear regulators that operate with minimal voltage difference between input and output voltage to reduce heat waste and to increase the stability of the output voltage. These regulators are made using PMOS, NMOS, and BJT designs [30], [31].

3.1.11.2 Linear Post Regulation

Linear regulators are frequently used after switching converters to improve output quality by reducing output noise. This technique will need to be applied in Noctisense to provide the stable voltage source with very low ripple voltage.

3.1.11.3 Hybrid Power Architecture

Noctisense will need both very efficient regulators for low thermal generation and low losses to heat but it will also need a very low noise regulator to supply the doppler sensor with power. This means that Noctisense will need a hybrid mix of both types of regulators and cannot get away with just one type. To implement this functionality the switching regulators will handle the bulk of the conversions as efficiently as possible and after that an analog low noise rail will be used to clean up the output of the switching regulators so it is usable by all noise sensitive components.

Table 3.19 Linear Regulation Technology Comparison

Technology	Description	Pros	Cons	Selection / Justification
Classic Series Regulator	Uses a transistor in series with a load to maintain output voltage	Stable, low noise, good input ripple rejection	Low efficiency and high heat production	Not selected for bulk conversion; too inefficient
Low Dropout Regulator (LDO)	regulator that works with minimal input/ output voltage difference	Low noise, low output ripple, stable	Still less efficient than switching, limited current capability	Selected as supplies for sensitive analog and RF systems
Linear Post Regulation	Uses LDO after a switching regulator to reduce noise	Combines high efficiency of switching with low noise of LDO	Adds small heat and cost	Selected for hybrid architecture on sensitive rails

3.1.11.4 Multi-Rail Power Distribution and Regulator Selection

In order to fully utilize the array of sensors and features that Noctisense will have we will have to implement the previously mentioned Hybrid Power Architecture and use a multi-rail power distribution system. The different components on the PCB will need different input voltages or have different tolerances to input voltage fluctuation and as such will require different regulators to supply them the power they need. The first Rail is a 7V switching regulator rail that is supplied its power by the AC-DC power supply and will provide power to the the 5V Linear Regulator rail. This Linear 5V rail needs to be a low noise low drop out linear regulator with a maximum ripple voltage of 100 mV. This will supply the mmWave radar module with a steady very low noise power source it needs to take accurate readings and measurements as well as powering the NIR laser driver. The third rail is a 5V switching regulator that will be supplied also by the AC-DC supply and will provide the power to both of the 3.3V Linear Regulators. These Regulators will be responsible for powering the ESP32, Thermal Sensor, Microphone, SD Card, miscellaneous LED's, optical photodiode, and its transimpedance amplifier. [30].

The part selection of the 5 regulators is as follows. For the 7V switching regulator rail, the part choice is an AP63201 adjustable Buck Converter. This will generate not only the 7V rail using a

high efficiency buck switching regulator but is fully adjustable, allowing for fine tuning of voltage output to anything under or equaling 31V. The selected implementation can deliver up to 2 A of output current with efficiency exceeding 90%. This regulator converts the 12 V input to a stable 7V output used to power the Linear Regulator that will power the subsystems of Noctisense [32].

In choosing a low noise 5V regulator to achieve the precise voltage regulation and reduce switching noise for the sensitive mmWave Radar subsystem, a low dropout (LDO) linear regulator was chosen to post-regulate the 7V rail. The selected component is the Texas Instruments fully adjustable TL3080, which offers an output voltage accuracy of ±0.05 V. This satisfies the system requirement of maintaining voltage deviation within ±0.1 V and comes in as another fine tunable regulator that can be fully adjusted to the exact output needed anywhere from 0-36V.

The second switching voltage rail is a 5 V rail, which powers both of the following 3.3V LDO regulators the power the rest of the systems components. Similar to the 7 V rail, an AP63201 fully adjustable buck converter is used and configured to provide an efficient and reliable 5V output. This regulator also supports current levels up to 2A, ensuring reliable operation during full load. Other devices such as the TPS62133 and MP2145 were looked at for improved efficiency and integration. However, these components introduce too much additional design requirements without providing any significant benefits for the system's operating range.

Finally, the last regulator is a second and third low-noise rails that operate at a 3.3V output voltage using a linear regulator to supply sensitive subsystems, including the photodiode-based optical circuitry as well as having enough current handling capability to power the additional 3.3V subsystems of Noctisence such as the ESP32 microcontroller, and SD card. The selected component is also the fully adjustable LT3080, which provides an output accuracy of ±1% and high power supply rejection ratio while also being a very robust and adjustable component [31]. This Third Low noise rail, while seeming redundant, is in place to power just the system microphone to help reduce the electrical noise picked up by the microphone.

Table 3.20 Switching Regulator Component Comparison

Part Number	Input Voltage Range (V)	Output Voltage (V)	Max Output Current (A)	Switching Frequency	Efficiency (%)	Selection / Justification
AP63201	3.8 – 32	Adjustable (0.8V / 31V)	2.0	500 KHz	90	Selected for, high current handling, simplicity, adjustab-ility and low switching requency

TPS62133	3 – 17	Adjustable	3.0	2.5 MHz	95	Not selected due to high switching Frequency and lack of adjustability
MP2145	4.5 – 16	Adjustable	4.0	1.2 MHz	90	Not selected due to unnecessary complexity and lack of adjustability
LM2596	4 – 40	Adjustable	3.0	150 kHz	75–80	Not selected due to lower efficiency and component requirements

Table 3.21 Linear Regulator Component Comparison

Part Number	Input Voltage (V)	Output Voltage (V)	Max Output Current (A)	Output Accuracy	Selection / Justification
LT3080	1.2 – 36	0 - 34V	1.1A	±1%	Selected for low noise fully adjustable rails, easy of use, and suitable for mmWave radar and optical circuitry accuracy requirement
LM2937	4.75 – 26	3.3 / 5.0	0.5	±2%	Not selected due worse accuracy and lack of adjustability
TPS7A49	4 – 36	5.0	0.5	±0.5%	Not selected due to higher cost despite the excellent accuracy

LM317	3 – 40	Adjustable	1.5	±1%	Not selected due to larger footprint and higher component count

3.1.12 Protection Circuitry

To ensure that the Noctisense system is fully safe and reliable, some protection circuitry is implemented at multiple stages of the power delivery network. These protections are designed to limit risks associated with overcurrent, overvoltage, reverse polarity, and electrical noise. Given the use of an external 12 V DC adapter in tandem with the sensitive subsystems including the analog optical sensors and the mmWave radar module, proper protection is necessary to prevent component damage and ensure that Noctisense meets its function comfortably and will stand a much better test against time.

3.1.12.1 Input Protection

In order to properly protect Noctisense we first must protect the input of the system. To accomplish this goal, we will need to protect the system from overcurrent, overvoltage, and reverse polarity. Overcurrent protection can be handled by a series of fuses on the 12 V input line. In the event of a short circuit or component failure, the fuse will blow and interrupt the current flow, preventing any current from damaging the sensitive Notisense components. Reverse polarity protection can be realized by using either a Schottky diode or a P-channel MOSFET. While a diode provides the simplest solution, it introduces a voltage drop and another power loss as heat. A MOSFET-based configuration is more complex, but it is much more efficient than the diode configuration because it minimizes the voltage drop present while still preventing damage from an incorrect power connection. Finally, to protect the system from overvoltage, a transient voltage suppression diode is placed across the input rails. This component clamps high-voltage spikes to a safe level and protects the regulators and any other components from voltage spikes that could potentially damage them or make their readings invalid [33], [34], [35].

3.1.12.2 Buck Converter Protection

While the 7V and 5V switching regulators used already include internal protection features such as overcurrent protection, thermal shutdown, and under-voltage lockout from the manufacturer, external filtering and stabilization components are still required to ensure reliable operation of the downstream sensitive components. Input and output capacitors are used to stabilize the switching regulators and reduce the ripple voltage they produce and make their output even more stable. Furthermore, if there is high frequency noise a ferrite bead can be added at the output to further suppress high frequency noise before it reaches sensitive components [38], [39], [40]

3.1.12.3 Low Dropout Linear Regulator Protection

The selected low dropout regulators, like the buck regulators, include internal current limiting and thermal protection mechanisms that prevent damage due to excessive load current or overheating. However, in order to maintain the required ±0.1 V output stability, the input and output must have capacitors to help prevent oscillations and to improve the output response of the regulators [31].

3.1.12.4 Load Level Protection

At the load level, additional protection and stabilization measures are implemented to filter high-frequency noise and stabilize voltage levels. In order to achieve these decoupling capacitors are placed near each integrated circuit and bulk capacitors are used on major rails to handle transient current demands especially for the ESP32 during wireless transmission bursts. The Sensitive analog components, including the photodiode transimpedance amplifier, thermal sensor, and mmWave radar module are isolated from digital noise through the use of dedicated low noise power rails and stabilization capacitors. This completely minimizes the risk of measurement errors in the sensitive components caused by any power supply fluctuations [29].

3.1.12.5 Electrostatic Discharge Protection

Electrostatic discharge presents a significant reliability concern for embedded systems like Noctisense, particularly where user interaction or other factors may introduce high voltage events. These discharges can result in component failure or damage that degrades system performance over time and wreaks havoc on the longevity of the system. As such, appropriate ESD protection measures are incorporated into Noctisense to protect it. ESD protection devices are placed at all external or accessible electrical interfaces and are designed to provide a low impedance path, effectively diverting high voltage spikes away from internal components. Under normal operating conditions, the protection devices remain electrically inactive and do not interfere with the signal or power being supplied.

Low capacitance ESD protection diodes are selected to minimize signal distortion and are capable of responding rapidly to voltage spikes by clamping voltage levels to safe operating limits. The selected devices are rated to handle a typical human body discharge ensuring robustness against the general everyday electrostatic events encountered in consumer environments. Proper placement of these ESD protection components is also very important to their effectiveness. The devices they protect are positioned as close as possible to the system protection, minimizing the length of unprotected conductive paths [24], [29].

3.1.12.6 Justification of Protection Circuitry

The implemented protections are layered and address potential failure points at the input, regulation, and load levels of Noctisense. Input protection ensures that faults from the external power source do not reach the system or any of its sensitive optical and radar components. Regulator level protections help to maintain stable voltage conversion under varying conditions, while local decoupling and filtering preserve signal integrity for sensitive Noctisense subsystems. The selected protection mechanisms are also consistent with industry best practices for embedded systems like Noctisense and will add a layer of robustness to the design of Noctisense so that it will last the test of time.

3.1.13 Housing

3.1.13.1 Design Methodology Considerations

For the design of the housing of our system, the primary methods to consider are; modifying an existing consumer lamp housing for our needs, or designing a custom housing using 3D Computer Aided Design (CAD) software such as Fusion 360, Onshape, Shapr3D, etc. While it may be simpler in some respects to modify an existing product to meet the needs of our system, it lacks the ability to accommodate the tight tolerances of our sensors. Specifically, the needed bracket for the spacing and orientation requirements of the radar module. As well as accommodation for the lens/passthrough for our thermal imaging and infrared sensors. This also due to being a less modular design unintended to be opened for prototyping purposes, which makes modifications and rapid development more complex. Therefore, designing a simple and functional housing in 3D-CAD software presents itself as the best option for the development of this project. Amongst the primary options for design software for our project, Fusion 360 has vast community support and documentation, as well as native PCB integration for modeling. As well as an educational license fitting the scope of our project.

3.1.13.2 Fabrication Complexity Considerations

For the fabrication of our custom design, there are various manufacturing technologies that are able to create this housing within our requirements, with the primary deciding factor in this area being rapid-revisioning and cost. Amongst widely used technologies our primary options are; Fused Deposition Modeling (FDM) 3D-Printing, Laser-cut acrylic paneling, and Computer Numerical Control (CNC) cut components. While the latter two methodologies can offer similar delivery times from design-to-prototype parts, the cost of both when compared to FDM 3D-Printing is exponentially higher. Further solidifying 3D-Printing as a primary option for our system housing is the accessibility to these technologies, with the university having vast access to numerous 3D-Printers throughout the campus.

Table 3.22 Housing Fabrication Technology Comparison

	FDM 3D-Printing	Laser-Cut Acrylic	CNC
Time-to-Prototype	Moderate, with modern consumer-class printers for our design < 12hr	Fast, for our design < 6hr	Slow, for our design > 24hr (due to machine complexity)
Material Selection	Highest, many specialty variants readily available	Lowest, only a few primary materials available	Moderate, specialty materials cost significantly higher with much lower availability
Programmability	Easiest, well documented toolset	Moderate, more niche toolset with limited toolset	Highest barrier to entry, requires a subject matter expert to operate

Cost	Low	Low	High
Availability for Use	High	Moderate	Low
Reliability	Moderate-High	Moderate	High

3.1.13.3 Housing Material Considerations

There are a wide array of housing materials to choose from within FDM 3D-Printing, with our primary requirements being; durability, thermal stability, electrostatic discharge safe (ESDS), and cost efficiency. This narrows our focus to three primary choices; Polyethylene Terephthalate Glycol (PETG), Acrylonitrile Butadiene Styrene (ABS), Acrylonitrile Styrene Acrylate (ASA), and some Polylactic Acid (PLA) variants. There are ESDS variants widely available for all of these materials, as well as they all share very similar thermal characteristics for the purpose of housings. Therefore the deciding factor comes down to mechanical reliability, and cost. Which brings the conclusion to use either an ABS or PLA based filament for cost, and ABS having higher mechanical strength compared to PLA leads that to be the primary option. Studies show that materials such as ABS-ESD7 and other FDM filaments have been used in many critical and RF-oriented projects with success[59].

3.1.13.4 Thermal Considerations

Thermal performance is critical in the design of the housing of our system, as the heat generated by our components may be low, this can lead to degradation over time. The commonly used approach we will follow for this will be to begin with a closed housing and measure thermal performance. Followed by the use of internal heat-sinks added as needed, then moving onto ventilation cut-outs, and possibly active cooling fans if required.

Table 3.23 Housing Material Comparison

	PETG	ABS	ASA	PLA
Reliability	High	High	Moderate, requires	High
Rigidity	Moderate	Moderate	High	Low
ESDS Available	Yes	Yes	No	Yes
Cost	Low	Low	Moderate	Low
Availability	High	High	Moderate	High

Chapter 4: Standards and Design Constraints

4.1 Regulatory Standards and Design constraints

4.1.1 Power System Standards

4.1.1.1 Electrical Safety Requirements

The Noctisense is designed to meet international and regional electrical safety standards IEC 60950-1 and IEC 62368-1 to ensure safe and predictable operation. These standards specify that the maximum allowable touch voltages of 60 V DC or 30 V AC RMS for accessible parts, minimum creepage distances at 3 mm for low voltage circuits and 8 mm for AC main interfaces, and a maximum component surface temperatures of 70 °C to 85 °C to prevent thermal hazards. Compliance also requires protection against overcurrent and short circuits. In order to meet these requirements, Noctisense uses a 12 V, 2 A regulated wall adapter that isolates main voltage from the internal circuitry of Noctisense. Overcurrent protection is handled by a Bourns MF-R200 2 A resettable fuse, while reverse polarity protection is provided by Alpha & Omega AO3407A P-channel MOSFET rated for −30 V. Surge suppression is handled by Littelfuse SMBJ15A TVS diodes and is rated at 15 V clamp voltage. Furthermore, high voltage areas are separated from low voltage logic with a minimum of 8 mm PCB clearance [22], [23].

4.1.1.2 Electromagnetic Compatibility (EMC)

Noctisense complies with EMC standards, including CISPR 32/EN 55032 and FCC Part 15 for emissions, and IEC 61000-4-3 for immunity. These standards dictate that radiated emissions must be controlled below 30 dBµV/m from 30 MHz to 1 GHz, and conducted emissions below 79 dBµV/m from 150 kHz to 30 MHz. Immunity testing requires that the device tolerates 10 V/m radiated RF fields between 80 MHz and 2 GHz without functional degradation.To achieve these requirements the use of the Murata BLM21PG101SN1D ferrite beads, 0.1 µF, and 10 µF ceramic decoupling capacitors are all used to suppress any high frequency emissions, and grounded metal shields around high frequency circuits such as the mmWave radar module. Furthermore, sensitive analog circuits are isolated on a dedicated ground plane to reduce electromagnetic interference [25], [26], [27].

4.1.1.3 Electrostatic discharge Immunity (ESD)

IEC 61000-4-2 requires ESD immunity to 4 kV direct contact and 8 kV air discharge for human body model scenarios. To comply with this standard Noctisense integrates a Nexperia PESD5V0S1UL diode array on all user accessible input or output lines and power entry points. Reinforced grounding ensures currents are safely diverted to the ground plane, preventing any functional degradation or permanent component damage [24].

4.1.2 Optical Safety Standard

4.1.2.1 Laser Classification Standards and MPE Analysis

The optical source selected for this system is a 940 nm near-infrared (NIR) diode emitter, operated in a pulsed mode. The system is designed to meet Class 1 laser safety requirements in accordance with University of Central Florida (UCF) Laser Safety guidelines. According to the most current laser safety manual published by UCF, “Class 1 lasers are considered to be incapable of producing damaging radiation levels and are exempt from most control measures or other forms of surveillance.” Additionally, “This Laser Safety Manual is a reflection of some guidelines and standards set forth by federal and state regulators, including: the American National Standards Institute (ANSI), Committee Z-136; the State of Florida’s Laser Safety Program regulations,

administered by the Florida Department of Health (FDOH), found in the Florida Administrative Code (FAC), Chapter 64E-4, Control of Non-ionizing Radiation Hazards; Food and Drug Administration (FDA), Code of Federal Regulations, Title 21, Part 1040.10; and the Occupational Health and Safety Administration (OSHA) standards."

For safe emission to be considered safe is if the Accessible Emission Limit (AEL) is above the radiation level of a laser and people are limited from exposure. Exposure includes eye and skin for a set amount of time period. That's where Maximum Permissible Exposure (MPE) comes into play. "MPE depends on wavelength, exposure duration, pulse structure, and viewing geometry. ANSI Z136.1 uses MPE as the core biological limit, and FDA currently recognizes conformance paths based on IEC 60825-1 Ed. 3 for classification of laser products in the U.S." [73].

Pulsed-source methods require checking at least three things: the single-pulse limit, the average-power limit over the relevant exposure time, and the pulse-train rule for many pulses [73]. That stepwise structure is explicitly shown in the UNL laser safety guide's pulsed-laser example

For this design, the source wavelength was 940 nm, the radiant intensity is assumed to be 50 mW/sr (I_e), the beam half-angle being 25 degrees, the repetition frequency being 10 kHz, and the minimum eye distance is 6.6 cm. The beam radius at the eye plane is estimated as

$$\mathbf{r = z \cdot tan(\theta) 6.6 \cdot tan(25^{\circ}) \approx 3.08\ cm}$$

Equation 4.1: *Beam Radius at the Eye Plane*

giving a beam area of approximately

$$A_{beam} = \pi r^2 = \pi(3.08)^2 = 29.8\ cm^2$$

Equation 4.2: *Beam Area*

Using the conical solid-angle relation $\Omega=2\pi(1-\cos\theta)$, the beam solid angle was estimated as 0.589 sr, resulting in an average optical power of

$$\boldsymbol{P_{avg} = I_e \Omega \approx 29.5\ mW}$$

Equation 4.3: *Average Radiant Power*

For a pulse width of 10 μs, the duty cycle is

$$\boldsymbol{(D = \tau f = (10 \cdot 10^{-6})(10 * 10^{3})) = 10\%}$$

Equation 4.4: *Duty Cycle for a Pulsed Signal*

leading to a peak power of 0.295 W. In the case the project starts using nanosecond pulses (Let's say 100 ns) the average power would be 29.5 W.

The average irradiance at the eye plane is

$$E_{avg} = \frac{P_{avg}}{A_{beam}} \approx 9.99 \cdot 10^{-4}\, W/cm^2$$

Equation 4.5: *Average Irradiance at the Eye Plane*

The pulse energy remains the same as it is not dependent on pulse width. The radiant exposure per pulse is

$$H_{pulse} = Q/A_{beam} = \frac{2.95\mu J}{29.8\ cm^2} \approx 9.9 \cdot 10^{-8}\, J/cm^2$$

Equation 4.6: *Radiant Exposure per Pulse*

For repetitive pulse operation, the number of pulses within a typical evaluation window is

$$N = ft$$

Equation 4.7: *Total Number of Pulses*

So in our example 20-micro second interval we will have $N = 200$. Classification requires verifying that the system satisfies all three exposure conditions (approximate limits were taken for λ = 940 nm (retinal hazard region) and nanosecond pulses, the standards (ANSI Z136.1 / IEC 60825-1)):

The single-pulse exposure must satisfy: $H_{pulse} \leq H_{MPE,\ single}$;

$$H_{pulse} < H_{MPE,single} \approx 5 \cdot 10^{-7}\, J/cm^2$$

Equation 4.8: *Maximum Permissible Exposure*

The time-averaged exposure must satisfy: $E_{avg} \leq E_{MPE,\ avg}$;

$$E_{avg} < E_{MPE,avg} \approx 10^{-2}\, W/cm^2$$

Equation 4.9: *Time Averaged Exposure*

The repetitive-pulse condition must satisfy: $H_{MPE,rep} \leq \frac{H_{MPE,single}}{N^{.25}}$

$$H_{MPE,rep} \leq \frac{H_{MPE,single}}{N^{.25}} \approx \frac{5 \cdot 10^{-7}}{200^{.25}} \approx 1.33 \cdot 10^{-7}$$

Equation 4.10: *Repetitive Pulse Condition*

The optical setup satisfies all three conditions of exposure. Overall, safety improves by reducing how much energy reaches the eye over time. This can be achieved by lowering the pulse repetition rate (which reduces cumulative exposure), increasing the beam divergence (which spreads the energy over a larger area), or reducing the energy per pulse (which lowers both instantaneous and accumulated exposure).

4.1.3 Radar System Safety Standards

4.1.3.1 Radio Frequency Safety and Security Standards

Safety is a critical design factor for any consumer electronics device intended to operate continuously in a bedroom environment. Our sleep-monitoring system uses a low-power 24 GHz millimeter-wave radar sensor, a microcontroller, and low-voltage power electronics (amongst other sensors) operating below 5 V DC, which collectively fall within well-established consumer safety regulatory frameworks. Because the device remains powered continuously on a bedside table near a sleeping person, regulatory compliance must address radio-frequency exposure limits, electromagnetic compatibility (EMC), electrical safety, and product certification requirements. The radar sensor operates within the 24 GHz Industrial, Scientific, and Medical (ISM) band, which is internationally allocated for short-range sensing technologies such as presence detection and motion monitoring[51]. RF exposure safety in the United States is regulated by the Federal Communications Commission (FCC), whose guidelines incorporate scientific research and exposure models from the International Commission on Non-Ionizing Radiation Protection (ICNIRP) and the Institute of Electrical and Electronics Engineers (IEEE). These organizations define Maximum Permissible Exposure (MPE) limits for RF radiation, and for frequencies between approximately 10 GHz and 300 GHz the ICNIRP guideline for general public exposure is roughly 10 W/m² averaged over a 30-minute period[51]. Low-power radar modules such as the MR24BSD1 transmit only a few milliwatts of RF power, which results in exposure levels several orders of magnitude below these limits even at close distances. Consequently, similar radar sensors are widely used in consumer smart-home products, automotive driver-monitoring systems, and occupancy detectors without documented health risks. Additionally, the radar transmitter must comply with intentional radiator regulations defined by FCC Title 47, Chapter 1, Sub-Chapter A, Part 15, Sub-Part C, ensuring the device operates within permitted emission limits and does not cause harmful interference to other electronic systems[51]. Our system demonstrates a field effect strength (V/M) orders of magnitude lower than that of the regulatory limit of 2500 V/M, as seen in equations (1) & (2). Equivalent international requirements exist for global markets, including certification under standards developed by the European Telecommunications Standards Institute (ETSI) as part of the CE Radio Equipment Directive (RED) framework, and similar compliance regimes such as Industry Canada (ISED) regulations[52].

$$P_{Watts} = 10^{(P_{dBi}/10)} = 10^{\frac{6}{10}} = 3.98\,W$$

Equation 4.11: *Antenna Power*

$$Field\ Effect\ Strength\ (\frac{V}{M}) = \frac{\sqrt{30 * Watts * 10^{(Gain_{dBi})/10)}}}{Distance}$$
$$= \frac{\sqrt{30 * 3.98 * 10^{(6/10)}}}{1.5} = 14.54\,V/M$$

Equation 4.12:*Field Effect Strength*

Electrical and product safety considerations are equally important because the device will operate continuously in a home environment. The system is powered entirely from low-voltage 5 V DC, which places it within the category of Safety Extra Low Voltage (SELV) systems and significantly

reduces the risk of electrical shock compared to devices connected directly to mains voltage. Compliance with product safety standards such as IEC 62368-1 ensures that electrical insulation, component selection, and system design prevent hazards such as electric shock, overheating, or fire[53]. In practical terms, this requires using a certified external AC-DC adapter, implementing appropriate PCB trace spacing and grounding, and incorporating basic protection mechanisms including reverse-polarity protection, current limiting, and regulated power rails. Since the ESP32 includes a Wi-Fi radio and the radar sensor operates in the microwave spectrum, the complete device must also satisfy electromagnetic compatibility (EMC) standards to ensure that it neither emits excessive interference nor becomes vulnerable to nearby electronic noise. Standards such as CISPR 32[54] define limits for radiated and conducted emissions, while immunity requirements specified in IEC 61000-4 cover disturbances including electrostatic discharge, electrical fast transients, and radiated RF interference[55]. Thermal safety is also considered during enclosure design because the radar module and ESP32 together typically dissipate less than 2 W of power, producing minimal heat but still requiring verification that external surface temperatures remain below safe touch limits under standards such as UL 62368-1[56]. Appropriate enclosure ventilation and flame-retardant materials such as UL94-V0 rated plastics could further reduce fire risk. Finally, the radar-based sensing approach inherently improves privacy and security compared with camera-based monitoring solutions because it collects only motion and physiological movement data rather than identifiable visual or audio information. This design approach aligns with secure IoT development guidelines promoted by organizations such as the National Institute of Standards and Technology (NIST), which emphasize minimizing unnecessary data collection while maintaining robust system security[57].

4.2 Design Constraints

4.2.1 Power System Design Constraints

4.2.1.1 Voltage Regulation and Ripple Constraints

The Noctisense power system employs a hybrid regulation approach to supply stable and low noise 5 V and 3.3 V rails. Switching regulators such as the fully adjustable AP63201 buck converters rated for 2 A output, provide the bulk power to the rails of Noctisense while low dropout linear LT3080’s supply the clean low noise power to the sensitive components that need very stable input voltages to make reading and take measurements as intended, whle also having the current handling capabilities to provide all of the systems power, This further reduces residual noise and voltage ripple by maintaining a voltage deviation of no more than ±0.1 V at the voltage output of the linear regulators. These clean non-noisy power supplies will allow the readings gathered by the system sensors to be accurate and reliable, satisfying the design constraints.

4.2.2 Optical Design Constraints

Performance of the optical time-of-flight (dToF) subsystem is governed by a combination of geometric, temporal, and signal-to-noise constraints that directly influence emitter selection, receiver design, and system architecture.

4.2.2.1 Geometric Constraints

The system is designed to detect a person at a working distance of around 1.5 m. At this range, optical power is significantly lowered due to spreading and diffuse reflection from an object or thing.

The selected emitter, the TSAL6400 infrared LED, exhibits a wide angular divergence (±25°). This introduces two key constraints: low optical intensity because of natural dispersion of light and low coupling from the receiver because a significantly small portion of light is detected.

To increase optical power detection beam shaping and collection optics would improve this factor significantly. Adding collimating optical elements to the system would increase optical power density at the target and improve return signal. This will be done with a large numerical aperture (NA). Therefore aspheric condenser lenses are used for their high collection efficiency for wide-angle sources and reduction in spherical aberrations at high angles.

4.2.2.2 Signal-to-Noise Ratio (SNR) Constraints

With a great distance of 1.5 meters or any long propagation there is a concern of attenuation. Before attenuation there's concern of reflectivity. For example, we plan on using white fabrics for high diffuse reflectivity, and overall stronger return signal. While avoiding dark fabrics as they absorb more NIR light reduces signal strength. However, as optimization continues, we will start to cover darker clothes as hopefully the strength of the signal will not be an issue.

Now there's attenuation to consider, as it is intrinsic loss to the system. For example, I can not change the inverse square law, diffusion of reflection, or noise from lighting. However, we can compensate for these parameters with the technique of comparing the input and output signals and seeing the time difference between the pulses.

This improves detection robustness by averaging stochastic noise.

However, we plan on using the MCU 280 MHz to time differentiate between two signals, which restricts the optical parameters to become dependent on the MCU and overall electronics.

Additionally, a narrowband 940 nm filter is incorporated to suppress out-of-band radiation. Absorptive filters are selected over dielectric filters due to their angle-of-incidence tolerance and wider field-of-view compatibility.

4.2.2.3 Cost Constraints

The choice of emitter introduces a fundamental trade-off between performance, cost, and safety.

Compared to high-power laser diodes (e.g., SPL TL90AT03), the TSAL6400 LED provides lower optical power (~0.04 W vs. tens of watts), wider divergence, simpler drive requirements, and significantly lower cost (<$1). The reflectivity of the target surface plays a critical role in system performance. Human clothing introduces variability in reflectivity, particularly in the near-infrared spectrum. Additionally, if the system were to try to obtain the best optical filters or a bandpass filter, the price could reach a couple of thousand dollars.

4.2.3 Radar System Design Constraints

4.2.3.1 Radio Frequency Emissions Constraints

With respect to the radar module, due to it being an electromagnetic wave sensor, interference within that spectrum is a major concerning factor to the reliability of data collection. Materials such as metals, conductive liquids, electronic devices, amongst other objects can obstruct the return energy[72]. This is primarily mitigated with control over the field of view of the radar module to limit its visible scope to only that of the person being measured. While this method is majorly effective for our systems use case, we may still encounter shifts in something like a blanket or a pet walking through the field of view causing miscalculations. To which we will have to accommodate for these anomalies by limiting the resulting delta from other sampling results within a range of acceptability before recording it. Along with creating a method for reporting an error if there is continued mismeasurement over a non-ideal period of time. Along with RF constraints on the performance of our radar sensor, we must also consider the safety constraints which were covered within the standards section of this report in section 4.1.3.

Chapter 5: Use of Large Language Models

5.1 Limitations, Pros and Cons

Large Language Models, such as ChatGPT, Claude, etc., are a category of deep learning models trained on human text (often by predicting the next word it's going to say based on positive or negative reinforcement). In other words, Large Language Models fall under the umbrella term Artificial Intelligence because they rely on neural networks, a subset of machine learning. It tries to imitate human-like intelligence when accommodating or completing a task.

Cloudflare uses a great example of how it is able to "learn". Using a high probability of predicting the next word. An example would be the ability to detect a pattern of commonly appearing letters in a partially completed phrase, then, through immense analysis of trillions of sentences, it has a higher probability of finishing the phrase and even creating its own sentences [59].

One of the great skills of Large Language Models (LLMs) is the ability to answer niche and broad questions in a matter of moments. Communicating with a computer always involves specific phrases, commands, syntax, verbage, case sensitivity, or designated inputs. However, with these LLMs, they can respond in a human-like tone and context, providing perspective (with some reasoning) on one's inquiry. For example, when asked the best way to eat a sweet potato, it can respond with a solution from its training, whether it be from entire Reddit pages or websites. It can collect data and provide a statistically significant response [59].

However, even in circumstances where the AI can respond to your question, it doesn't mean the dataset it is learning from is necessarily reliable. In circumstances where the AI has a dataset of all Reddit pages, there could be unexpected responses, like false narratives, irrelevant information, and even hallucinations. Hallucinations are when the AI makes something up that never existed, like citations, theories, people, or information. This is a major drawback because the AI doesn't know whether the information is true.

Using AI comes with significant risks and limitations when asking questions. As above, it could make up information or contradict itself within the same response. Or when correcting the AI, it does not standardly update the model to adjust for its mistake, as it will continue with the same misinformation and pattern. There has been some progress in reducing hallucinations and improving computation and reasoning across most of the AIs. However, the AIs are made by humans and cause biases and imperfections.

Additionally, there is a security risk, as malicious inputs may be manipulated, and their confidentiality is limited. One can host their own LLM, but it requires significant knowledge of computers (and computer science), and the cost of RAM has risen significantly in the past two years due to the AI-driven RAM surge.

When applying to our project, there is a concern of hallucination. If we ask how a device works and receive an incorrect answer, it will set the group back from writing and moving the project forward. However, in a similar way, it can harm us; it can also help us find new methods we haven't thought of before, speeding up the project. Additionally, as students on this project, we have the responsibility to verify the validity of any new information we come across. Sourcing

articles or papers to verify the statement that the AI said is true. Additionally, there are now new AI's that provide sources for the statements they use by looking online and speaking with other AI's to improve validity, such as Perplexity.

For verifiability, ask the LLM for further explanation, and validate the information from your notes or previous classes. For example, ChatGPT isn't very good at circuit analysis. I tried asking a few trivial questions, such as which voltage source I should use to light an LED without damaging it. Sometimes it is just easier to look up your questions online, as there are more visuals and books that provide richer lessons. Overall, even with these LLMs' flaws, it's a great time to be suspicious, methodical, and to use your resources. Especially as LLMs become more prevalent in our daily lives. It would be a great thing if we learned how to utilize different LLMs in different scenarios and how they can be experts in specific fields. As this assignment and grinding through to find answers for this and other classes, using an LLM is not a bad choice to start, and move on from there. Even Perplexity, another AI, provides links (evidence) to the sources of its information. It is good to use LLMs as tools, but not crutches.

5.2 Case Studies

5.2.1 Case Study 1: Generation of ideas

With the warnings and concerns noted above, Perplexity was used to develop a response to this section. LLMs can assist in brainstorming alternative design approaches. For example, when deciding to use Direct Time-of-Flight (dToF), Indirect Time-of-Flight (iToF), or even Structured light methods. It can search the web to find old and new ideas that most people haven't heard about in a while, or to help you act on the plan. Just asking "what are good questions to ask you, the LLM?" As it can give you more insight and has for many of us how to ask more questions critically and what to consider.

LLMs bring in a new perspective. Even when brainstorming how to complete one of these sections, they have provided templates and/or outlines to our members. There might be a fear of getting the wrong concept down. However, that is why it is important to always question what the LLM's provide. For example, comparing previously done Senior Design Reports and assessing what is required. Using this comparison reduces the risk of going on a tangent or including unnecessary items, such as suggesting AR coating tables and so forth.

The LLM's can oversimplify complex systems in both good and bad ways. The perspective is that it lacked the understanding the user was hoping to learn. LLMs may forget to mention a vital part or contradiction in different conversations and won't adjust themselves. An example of this is giving false hope about the system parameters. For example, the LLM mentioned that a Michelson interferometer can achieve> 2 meters of detection. However, very impractical, and forgot to mention that it requires several mirrors to ensure equal optical path difference (ODP) for both arms. Which would have been vital information at the beginning of the semester. Another example would be the Radio Module. ChatGPT said the module was capable of detecting> 2 meters. However, it was soon corrected by a professor and a spec sheet revision. Overall, it is critical to verify LLM information, as it may set you back a couple of days or a week on the report if it forgets to mention or fails to correct the information it provides.

On the same note, without oversimplifying the system, it would be very difficult to learn the basic concept without simplification.

And when you run out of ideas, you can ask them again to help build you another example or outline. Additionally, one should use any LLM's text with a grain of salt as its verbiage lacks nuance, depth, and meaning. It will use terms broadly, and it is significant to follow up on questions. Therefore, it is not the best to copy and paste whatever the LLMs say, as their words don't have much meaning.

5.2.2 Case Study 2: Part Selection

LLMs such as ChatGPT has and can be significantly helpful for finding components and assistance in finding cheaper products. In recent updates, ChatGPT displays the product and links to the website of the product and compares items wanted, as in the picture below, *Figure 5.2.2.1*. There is a concern for possible fake websites or ChatGPT tracking your info. For tracking the links, usually there is a fragment tracing back to chatGPT on the website link. For fake websites, one can click the website or search the key terms to verify the existence of the product. Additionally, one can try to look for the desired product beforehand to see if there are any biased prices, oftentimes chatGPT can find cheaper products than using duckduckgo search engines. The following prompt was used, "Help me find a cheap absorption filter for my central wavelength of 940 nm" The figure below displays affordable Glass filters for Near Infrared Light and compares commercially available products

	ELP 940nm IR Bandpass Filter Glass $3.04 • AliExpress	**940nm Narrow Bandpass Optical Filter** $5.00 • eBay - tangsinuo	**HWB940 Long-Pass IR Black Glass Filter** $9.00 • eBay - tangsinuo
Price	~$3	~$5–10	~$9–15
Type	Bandpass (loose)	Bandpass (moderate)	Absorption / long-pass
Selectivity @ 940 nm	Medium	Better	Low
Blocks visible light	Yes	Yes	Yes
Blocks other IR (850 nm, etc.)	Partial	Better	No
Best use	Cheapest demo	Budget system	Noise reduction only

Figure 5.1. *Example Prompt to LLM*

5.2.3 Case Study 3: Accelerating Early Research

LLMs can significantly accelerate background research. For example, exploring different optical methods for person detection, such as dToF, and summarizing how the photodiode, pulse emitter, and connection to the MCU work. Overall, helping with finding key components. By asking for affordable products available, ChatGPT will build a list and compare the different uses of the products, and/or compare products amongst themselves. Additionally and more importantly. Making starting points for a deeper understanding of general systems and interactions between subsystems. For example, providing information such as the significance of a transimpedance amplifier (TIA) for boosting the photodiode's output signal. For optical engineers, they become

more of an opto-electronic engineer, as they need to communicate with other members who require parameters, while also learning the limitations of electronics. Without a strong foundation in each other's fields, it will be difficult to communicate or exchange with members. Overall, LLMs can reduce the language or technical barrier in a shorter amount of time. However, it is key to communicate with group members, as ChatGPT may be incorrect in some cases. Sometimes it is easier to discuss a troubling concept with your group partners. Do not solely rely on ChatGPT or any LLMs in that matter, just as you don't want to solely get information from one source.

5.2.4 Case Study 4: Interpreting Datasheets

There have been many times where LLMs have helped individuals with understanding terms and jargon barriers. For example, when an individual misinterprets or generalizes specs incorrectly, LLMs can give a second opinion when there might be a lack of assistance or provide a nuance that a teammate might miss. For example, a datasheet may list a photodiode's rise time, it may not explain how that rise time interacts with the transimpedance amplifier bandwidth or how it limits the system's ability to resolve short optical pulses. An LLM can help bridge this gap by connecting the parameters into a fundamental understanding for everyone. Additionally, if an individual has forgotten what a component means entirely, the LLM can step in and provide resources like links to said topics and gives you the potential to keep asking questions.

Chapter 6: Hardware Design

6.1 Power Delivery / Electrical Power System

The power delivery system draws power from the AC wall outlet, where a standard barrel plug is used to provide power to the full PCB. This requires only one power input into the PCB. One consideration our team made when designing the power distribution circuit was reverse polarity protection. This is because barrel jacks can come with positive or negative polarity. Having reverse polarity protection ensures that the sensitive components don't get damaged in the case where a user utilizes the wrong cable.

To achieve the necessary voltage levels for the project, a buck converter is used to step down from the 12V supplied from the barrel jack to voltages suitable for our project. The buck converter allows us to step down larger voltages in a much more efficient way than a linear regulator. This comes at a cost; however, this cost being switching noise. To resolve this, the output of the buck converter is fed to a Low Drop Out regulator. The Low Drop Out regulator is in place to mitigate the switching noise and provide a clean voltage output to alleviate the risk of bad power messing up sensor readings. In terms of packaging, the regulators were designed on a separate board to help reduce risk and localize the error. However, the voltage protection circuit was still kept on the main MCU board.

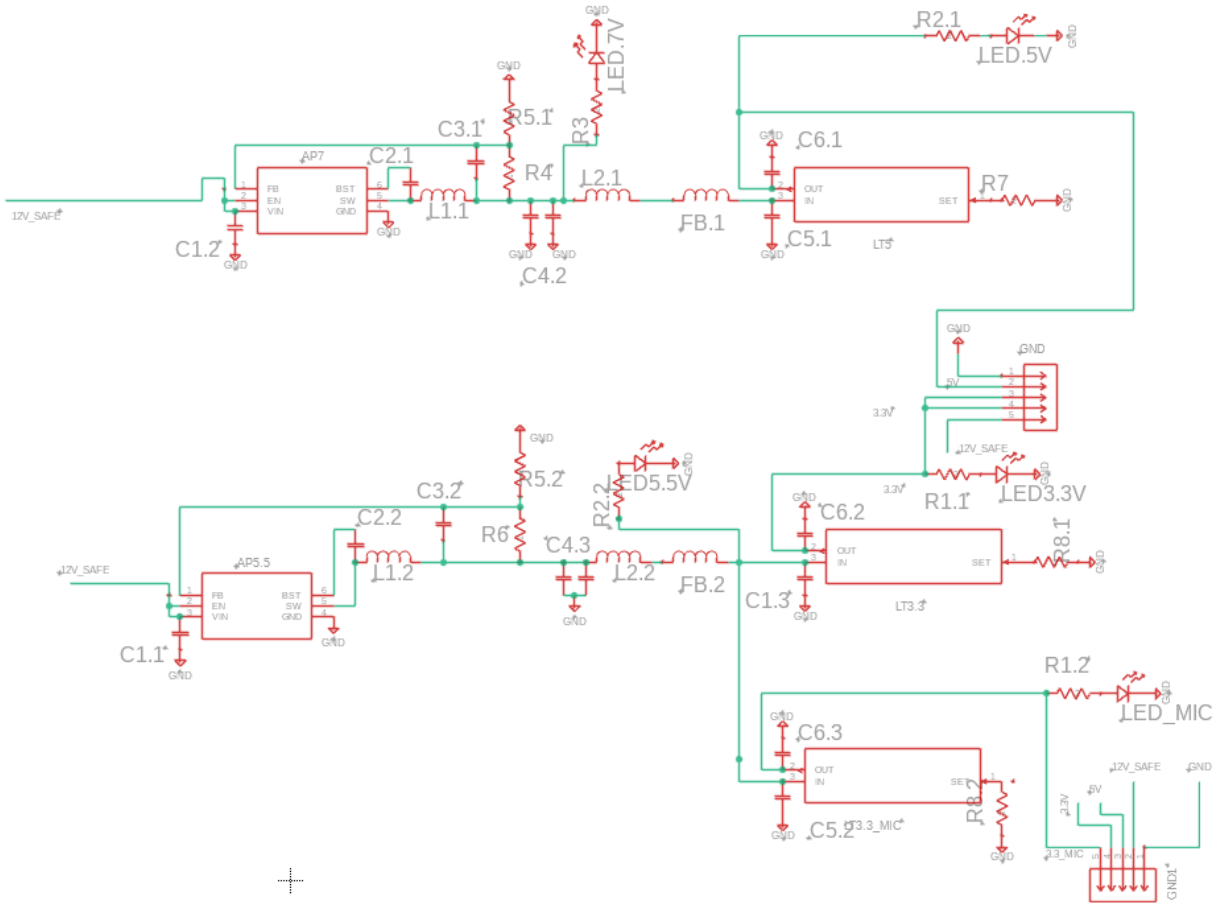


Figure 6.1: *Regulator Schematic*

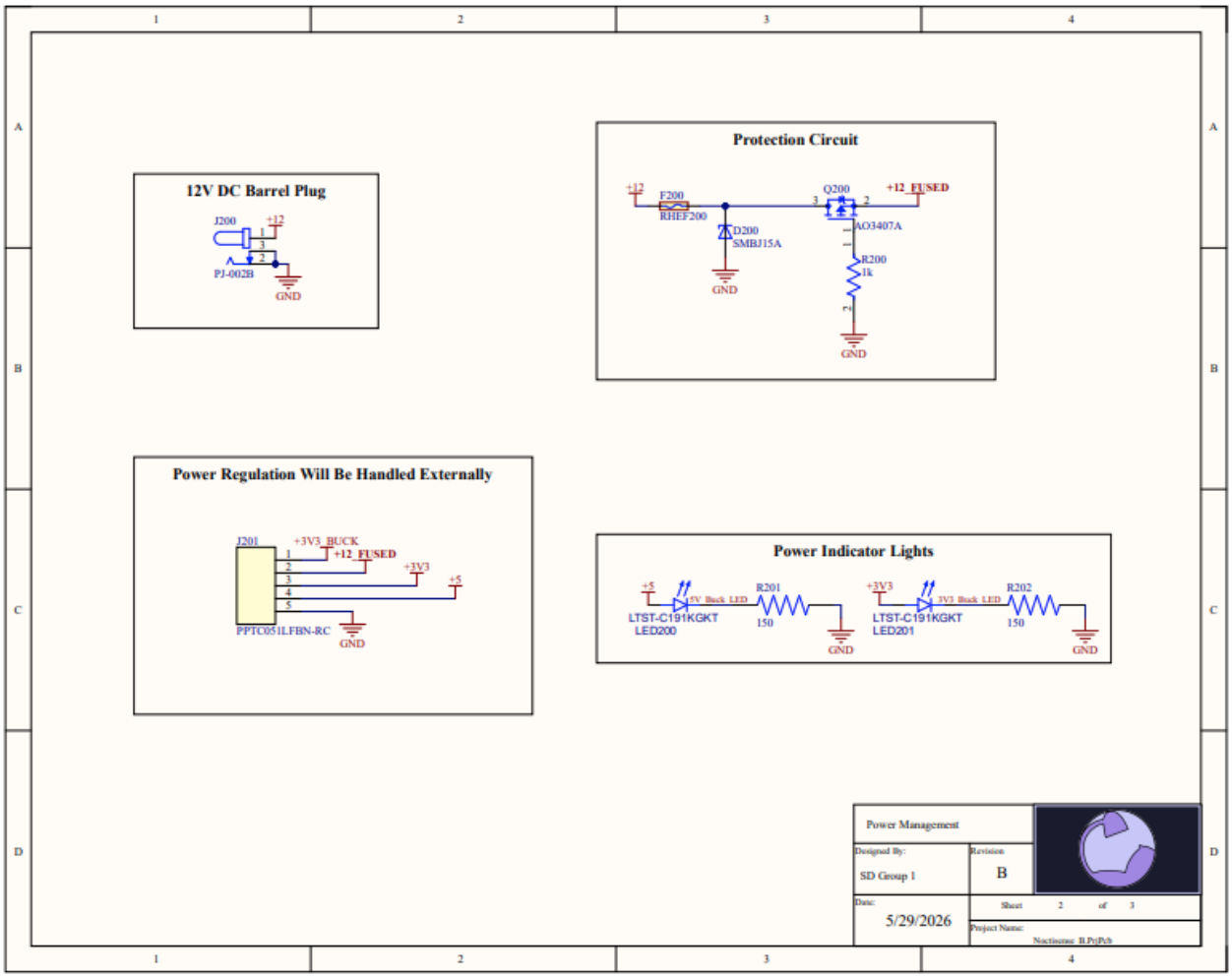


Figure 6.2: *Voltage Protection Circuit Schematic*

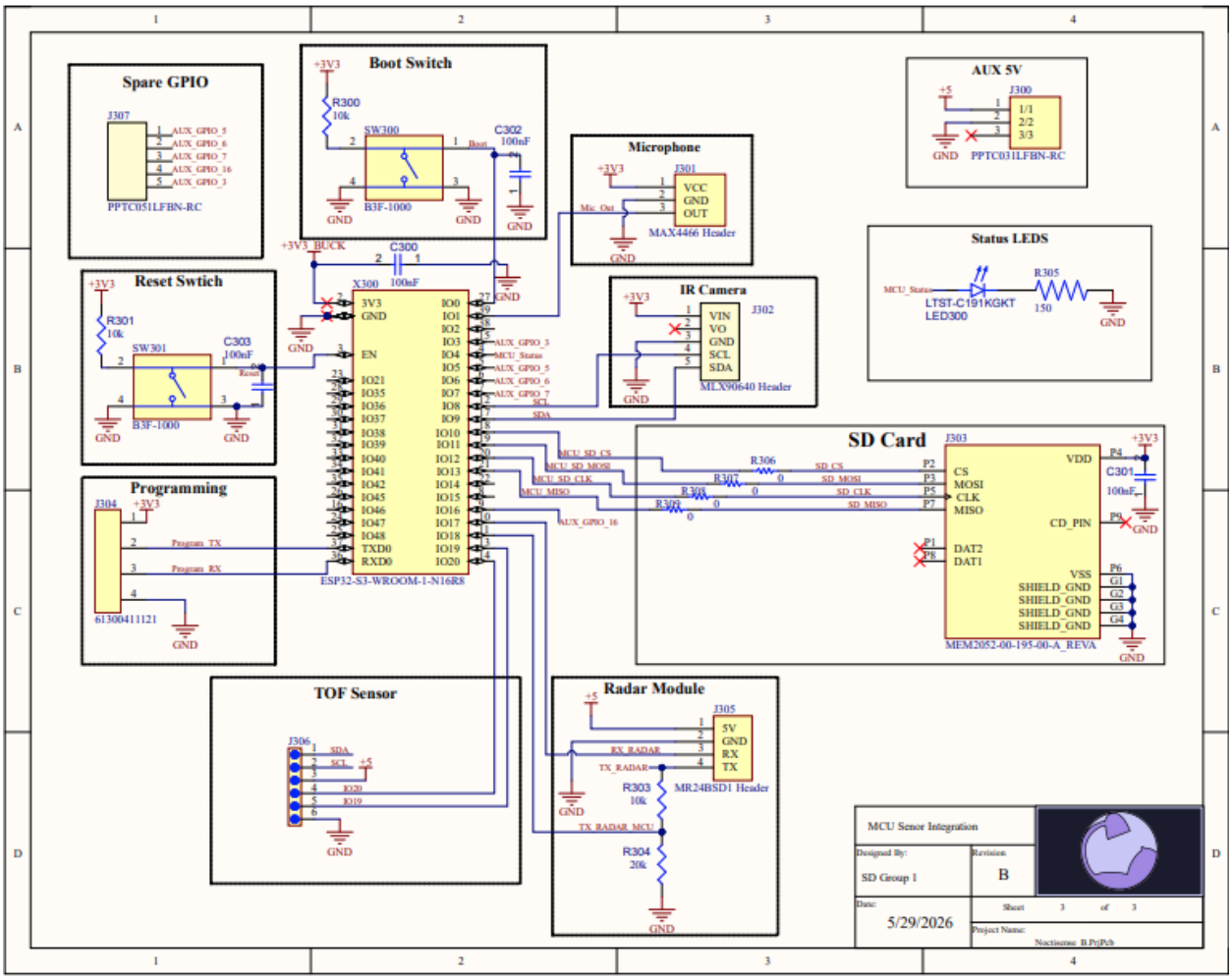


Figure 6.3: *MCU Sensor Schematic*

Figure 6.3 shows the connections between all the sensors in the Noctisense sleep analyzer. The voltage from the LDO found in Figure 6.1 will provide the power for the whole system and were chosen such that their maximum power ratings far exceed the theoretical current draw of the device.

To fulfill the requirements for a significant PCB design, the board must feature at least one MCU. The MCU is of 41-SMD Module package type. This is neither QFN nor BGA, so it will be suitable for use on the project, satisfying the MCU requirements. To meet the stringent physical design guidelines, the final prototype cannot contain any solder-less prototyping board. Additionally, relying on a final product with multiple development boards integrated is not sufficient; these development boards must be replaced by a customized board with significant layout design. The Noctisense system adheres strictly to this by routing the bare ESP32 module and its required peripherals directly onto a unified custom board.

The system integration heavily involves several peripherals connected to the MCU. The SD card is physically attached to the PCB to ensure that there is a good connection between it and the MCU. The other sensors are more so like modules and therefore contain headers to connect to

them. The use of header pins allows the system to be modular. This makes things like switching out components if there's a fault, as simple as plugging another sensor back in.

By combining these specifically chosen modules such as the sensors and customized filters the design focuses strictly on approved project scopes. Ultimately, the integration of the central MCU, the custom power supply unit, and the multiple hardware peripherals is enough to satisfy all the PCB requirements as listed in the guidelines.

6.2 Optical Schematic

Below (Figure 6.4) we have an image of the schematic of the optical subsystem. The TSAL6400 LED, two TSOP58238 photodiodes, ACL25416U-B lenses, and 3D printed containment. The distance between the lens and LED is the focal length (d), which is 16 mm. The lens had many constraints. First being the right size to encompass both the LED and photodetector, accumulating a width of ~ 10 mm. So we would need a lens that can encompass both, while being an aspheric condenser lens. The distance separating the emitter and (second)detector is 25.4 mm. Where the separation of the two lenses manages the minimal distance in which both the TSAL6400 and photodetector (PD) overlap or first point in space where the signals overlap at a maximum.

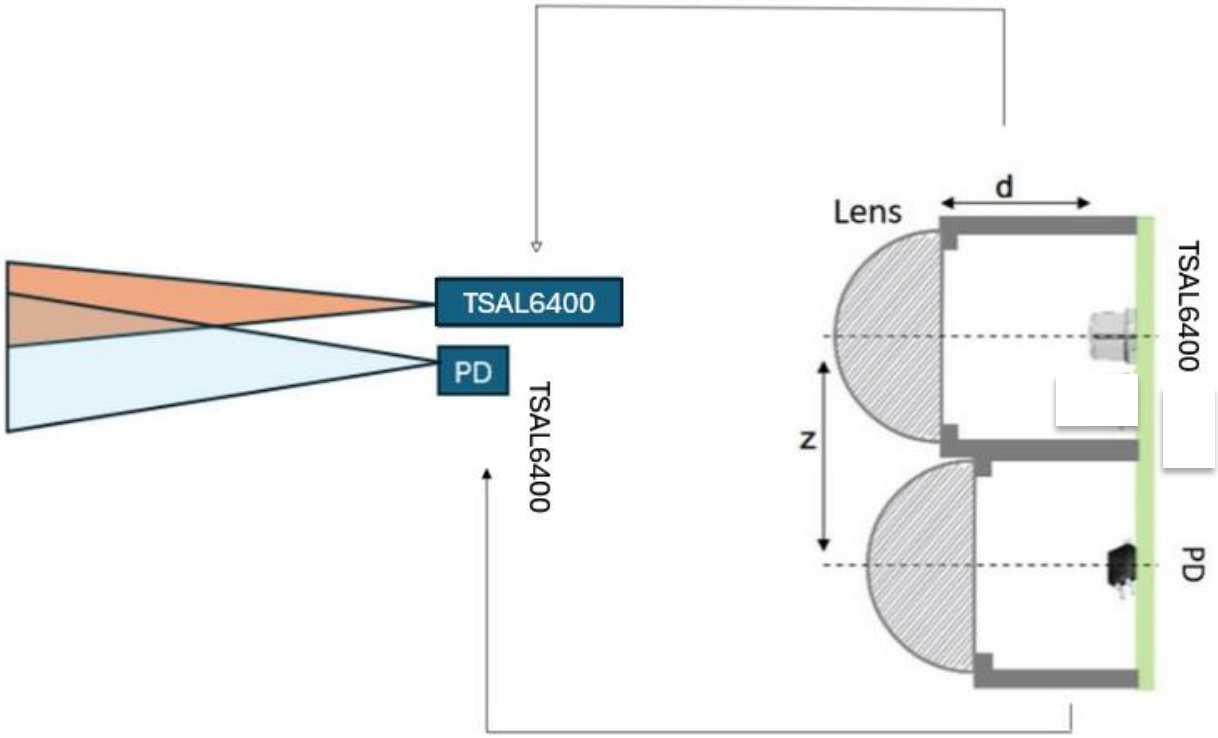


Figure 6.4: *Schematic of optical design*

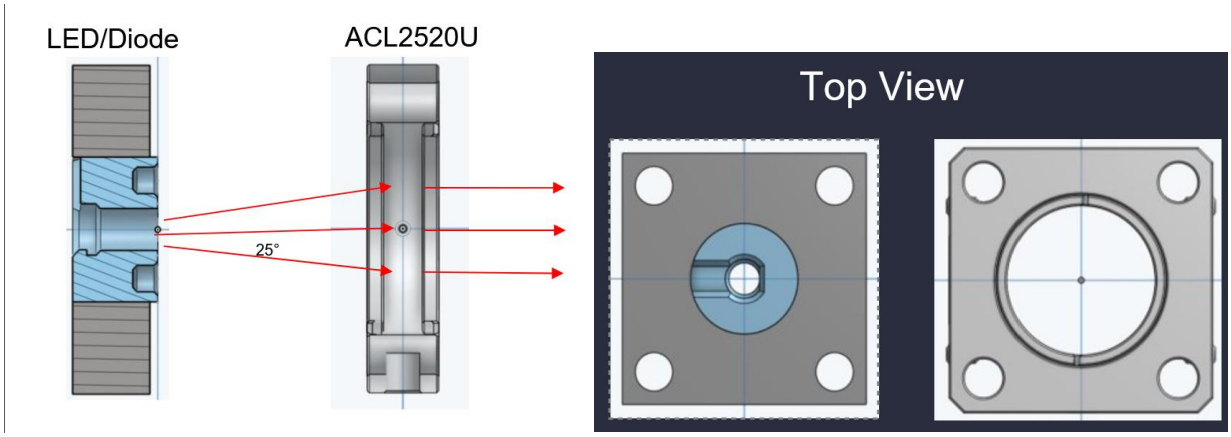


***Figure 6.5**: Side View of actual holders and Top View*

For lens mounts thorlabs were used due to the high rigidity and limited movement. The 3D printed mounts lacked the tight fit and control that was present in the thorlabs holder.

The LED can be held with a holder which prevents it from moving it back. However, it is not required as in the demo's we could just use a plate with holes big enough to fit the LED probes. The significance of this optomechanical design is to hold the LED and lens directly together as any imperfection in x, y, or z axis causes great loss in power. Since the LED has such a narrow field, a 2 degree off the optical axis can make reading the signal difficult to read.

For the electro-optics side we have a circuit schematic with voltage and resistances.

The following equation was used to find how much the voltage source(V_S) would be to turn on the LED, where V_F is the forward voltage required, I is current (how much is required of the LED), and the resistance is assumed to be 1Ω.

$$\mathbf{V_S = V_F + IR}$$

***Equation 6.1:** Voltage Source Equation*

VS =1.4 V + (100 mA) *(1 Ω) = 1.5 V. The Photodiodes are reverse biased for the depletion region to increase, leading to photoelectric effect. Therefore, leading to a larger photocurrent when the optical signal is received.

A transimpedance amplifier (TIA) is used to convert the photodiode current into a measurable voltage. Assuming a maximum photodiode current of 50 µA and a desired output swing from 0 V to 3 V, the feedback resistor R_f .We can then calculate the feedback resistor from the difference of maximum and minimum volt desired (0 V to 3 V) divided by the current from the photodiode.

$$\boldsymbol{R_f = \frac{V_m - V_o}{I_{PD}} = \frac{3\ V}{50\mu A} = 60k\Omega}$$

***Equation 6.2:** Resistor R_f*

This leads to a resistance of 60kΩ for the feedback resistor.

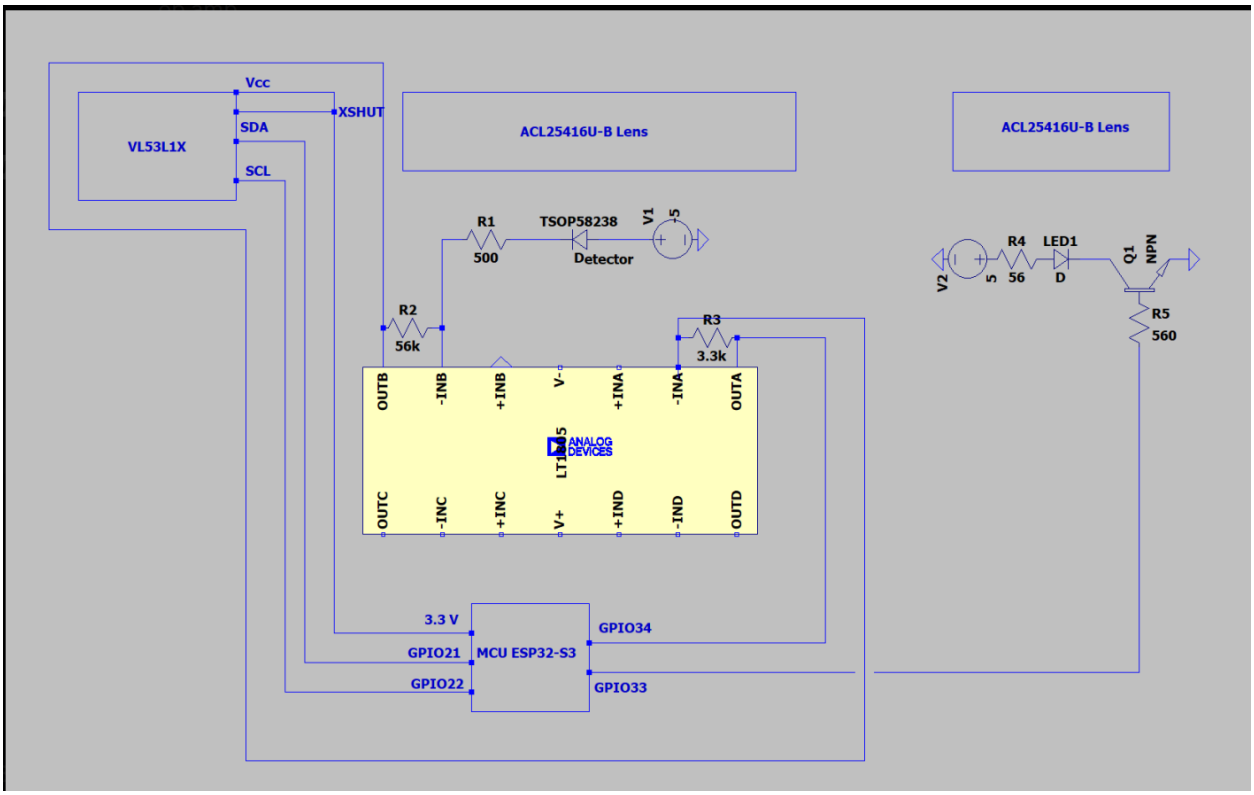


Figure 6.5: *Circuit Schematic of the electro-optics*

6.3 Microphone Schematic:

To fulfill the PCB's peripheral requirements, our group developed a custom microphone schematic. While the design was primarily based on an Adafruit reference schematic, we incorporated one significant modification: the addition of a Low-Dropout (LDO) regulator for the microphone's input voltage. During testing, we discovered that without a clean power supply, random artifacts would occur during the recording process. The modified microphone schematic featuring the LDO is shown below.

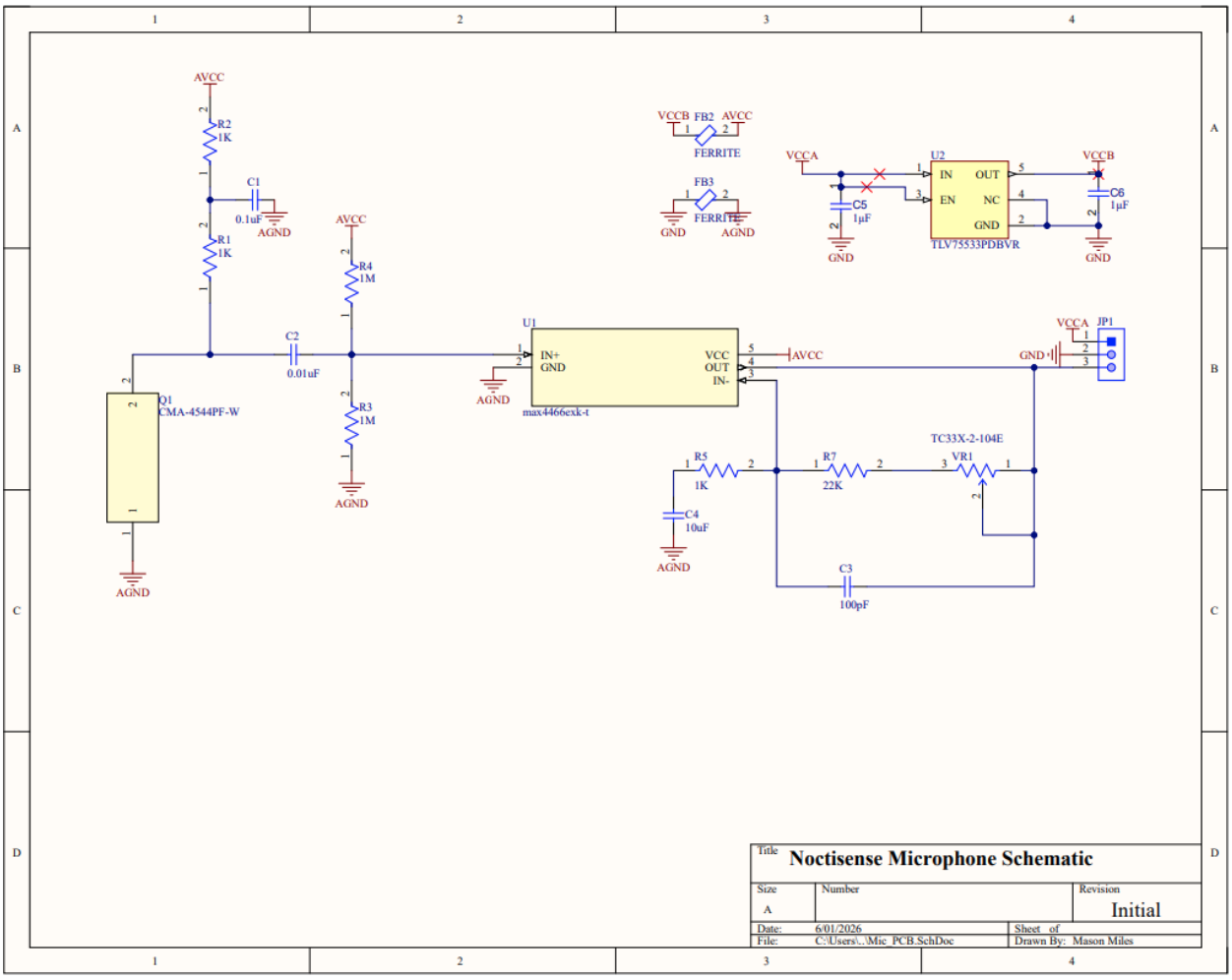


Figure 6.6 *Microphone Schematic*

Chapter 7: Software Design
7.1 Application Use Case Diagram

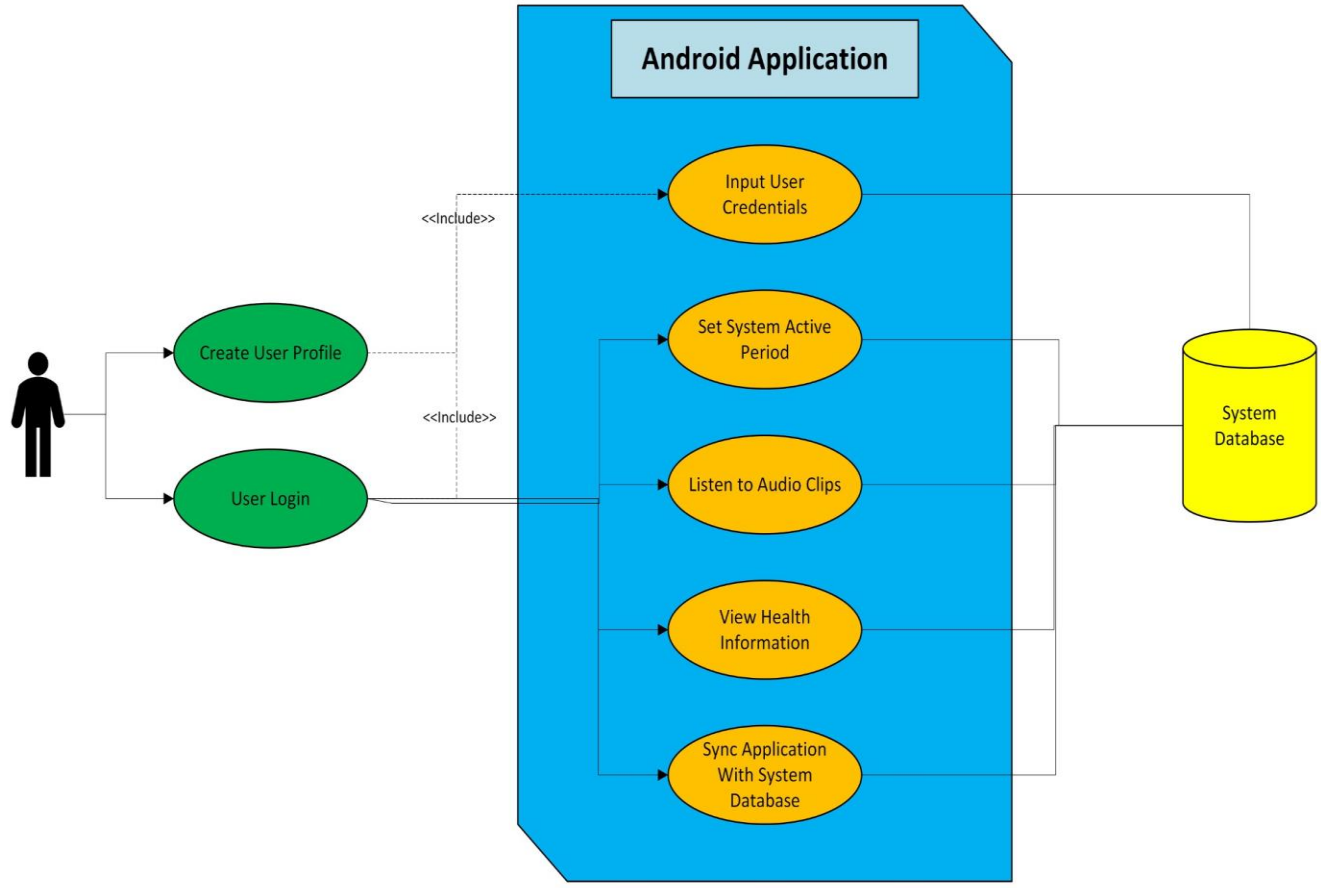


Figure 7.1: *Application Software Use Case Diagram*

7.2 Application-to-HTTP Server Activity Diagram

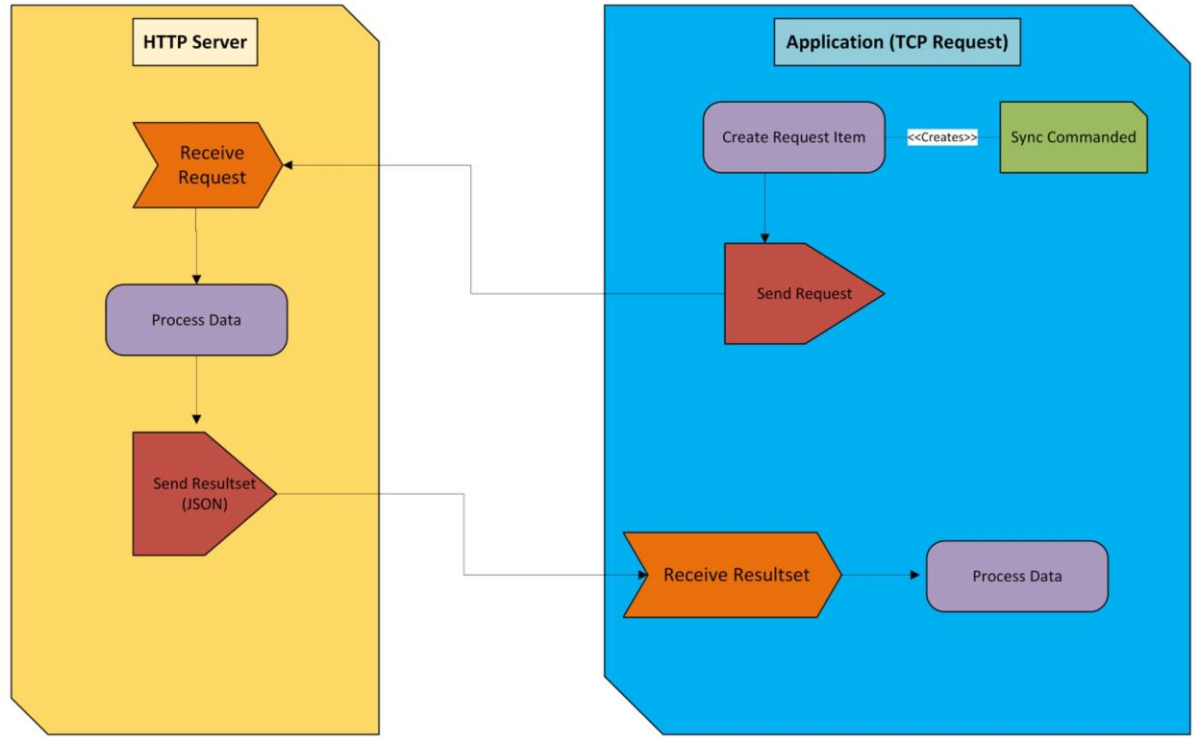


Figure 7.2: *Application-to-HTTP Server Activity Diagram*

7.3 Bed Detection: Time of Flight Sensor

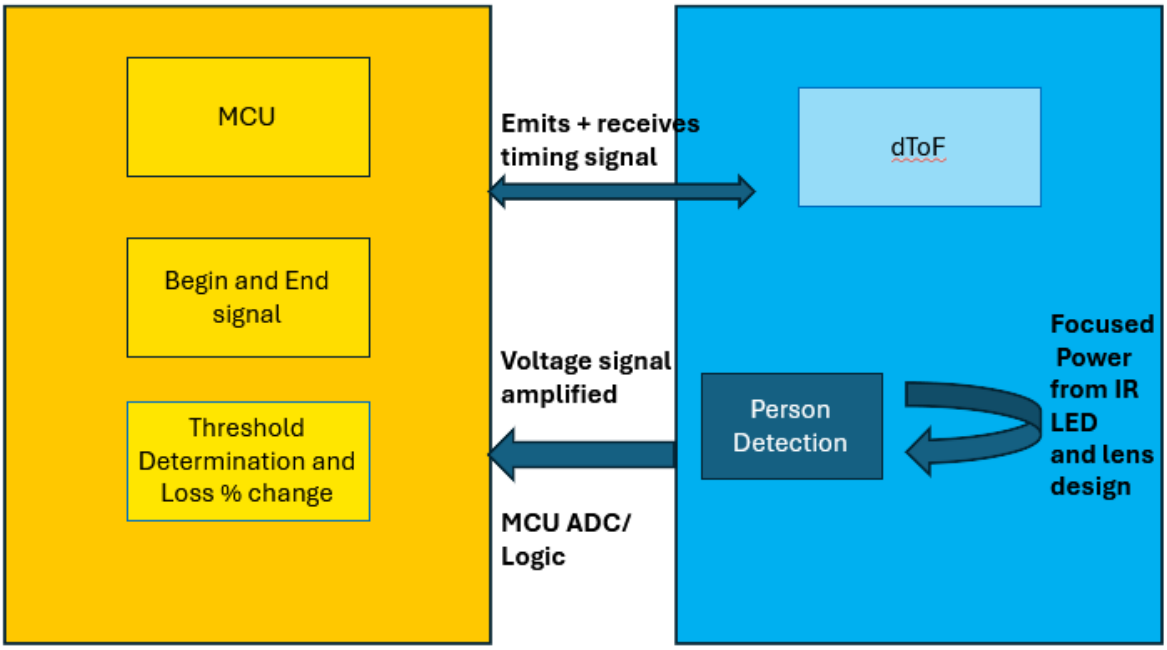


Figure 7.3: *Bed Detection: Time of Flight Sensor*

The optical detection system converted the analog voltage output from the photodiode/TIA circuit into a simple person-detection decision. The ESP32 first reads the TIA output through GPIO 34, which is used as an analog input. Since the ESP32 ADC is set to 12-bit resolution, each reading is converted from a digital value between 0 and 4095 into a voltage using the 3.3 V reference. To reduce noise from individual ADC samples, the code samples 500 ADC samples, averages them, and then converting them to an average ADC value into a voltage. This gives more stablility.

Before detection begins, the system goes through a calibration phase for 3 seconds. During this time, the code assumes the bed or reference region is in its normal empty condition. It keeps adding the measured voltage values and then calculates an average baseline voltage. This baseline becomes the reference signal level. After calibration, every new voltage measurement is compared against this baseline. The main quantitative value used for detection is the percent voltage drop (Vbaseline−V)

This value describes how much the received signal has changed compared to the original reference condition. The detection logic was written as a state machine with two possible states: EMPTY and OCCUPIED. When the system is in the EMPTY state, it checks whether the measured voltage drop is greater than 12%. If this happens for 1 consecutive measurement frames, the code switches the state to OCCUPIED. This prevents one noisy reading from falsely triggering detection. When the system is already in the OCCUPIED state, it waits until the voltage drop falls below 6%. If this happens for 5 consecutive frames, the system switches back to EMPTY. The 9% detection threshold and 6% release threshold create hysteresis, meaning the system does not rapidly jump between empty and occupied when the signal is close to the threshold.
“The output of the code is printed in CSV for plotting in Matlab. Each line contains the time in milliseconds, the measured voltage, the calibrated baseline voltage, the percent voltage drop, and the detection state. The state is printed as 1 for occupied and 0 for empty. This allows the optical signal and detection decision to be compared directly over time. In this way, the code connects the physical optical measurement to a quantitative detection result: the photodiode voltage gives the received light level, the baseline gives the reference condition, the percent drop gives the change caused by a subject, and the state machine determines whether that change is stable enough to count as person presence,” ChatGPT summarized my code and gave description here.

7.4 Audio Tracking

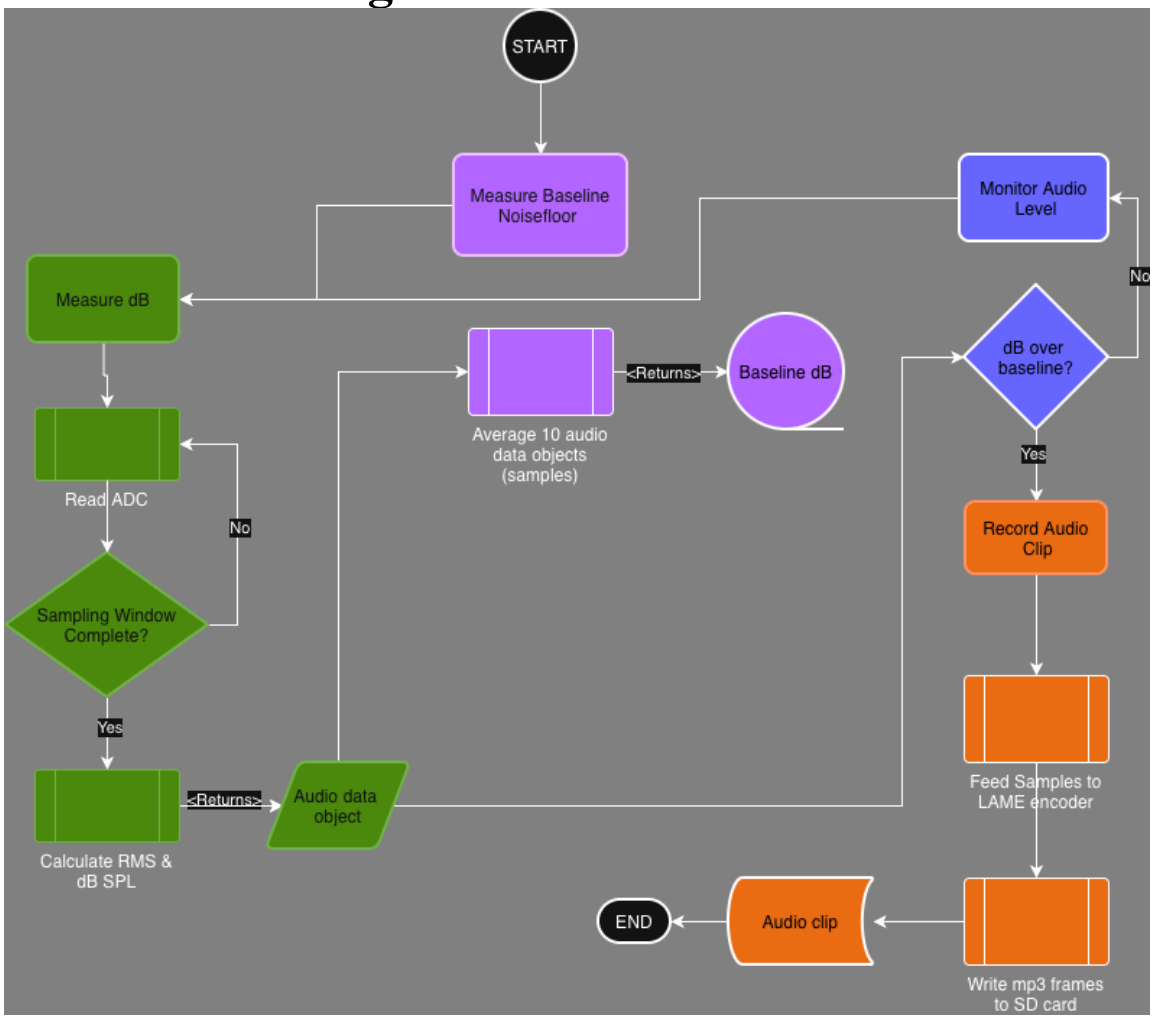


Figure 7.4: *Audio Tracking*

7.5 Movement Tracking Activity Diagram

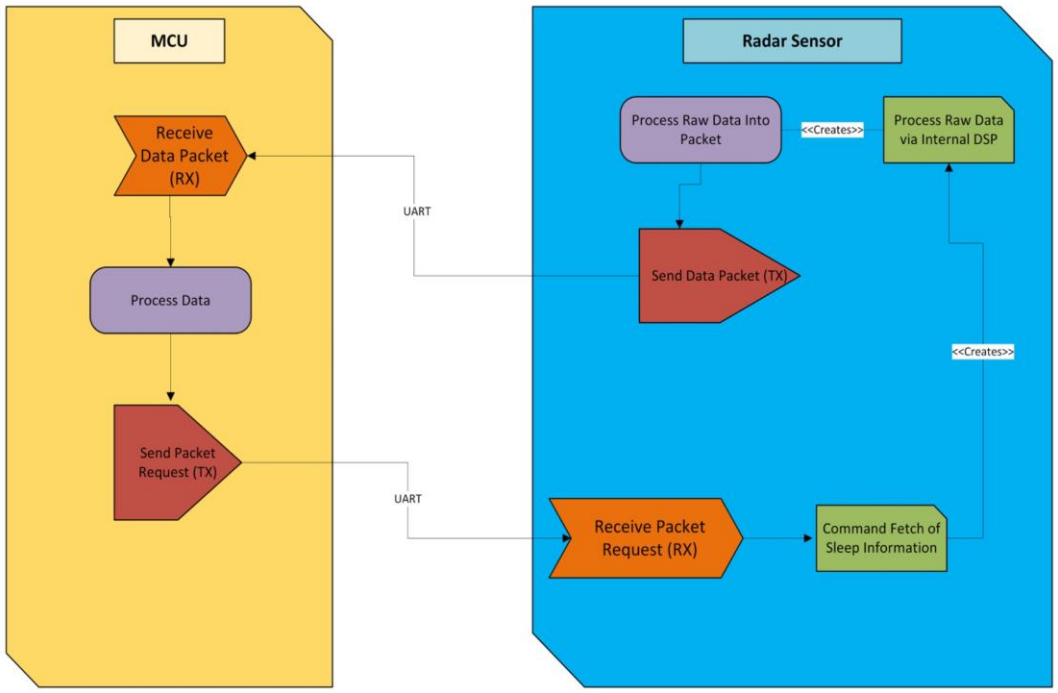


Figure 7.5: *Movement Tracking Activity Diagram*

7.6 Thermal Imaging Activity Diagram

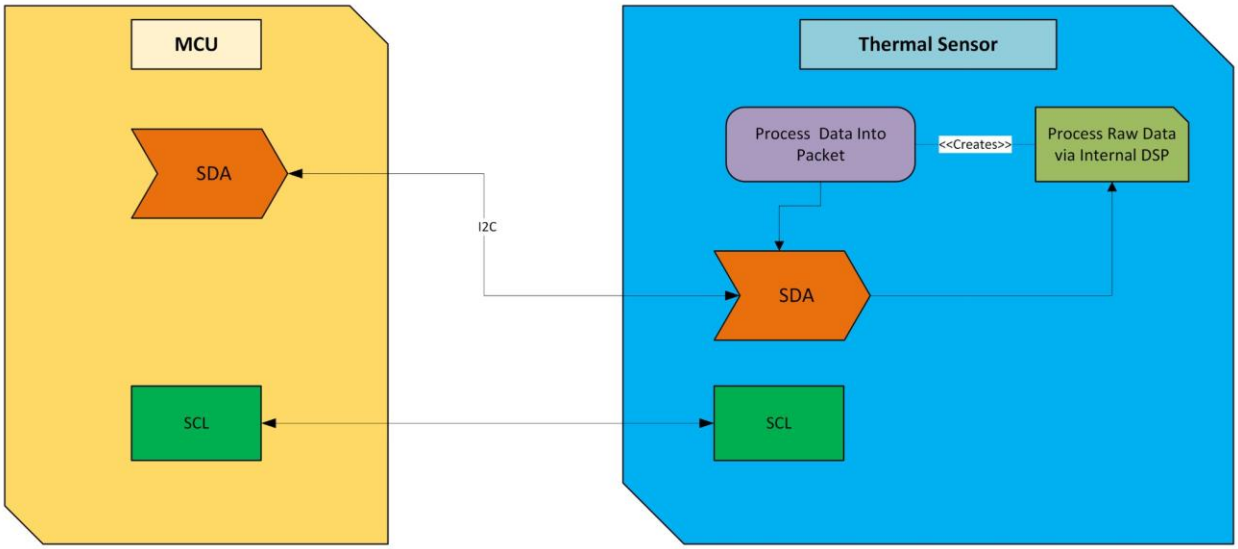


Figure 7.6: *Thermal Imaging Activity Diagram*

Chapter 8: PCB Design

The printed circuit board (PCB) serves as the central hardware backbone for the Noctisense contactless sleep monitoring system, routing power, digital signals, and sensor data between the core processing unit and its peripheral components. The board was engineered utilizing Altium Designer, an industry-standard electronic design automation (EDA) software suite. To guarantee

high yield and reliability during the fabrication process, the design strictly adheres to Design for Manufacturability (DFM) principles. A customized set of design rules was imported into Altium, specifically tailored to the manufacturing capabilities and tolerances of JLCPCB. This preemptive design rule check configuration ensured that trace spacing, via drill sizes, and annular rings fell within standard manufacturing limits, eliminating the need for costly board revisions.

The PCB stack-up utilizes a standard two-layer FR-4 configuration with a 1 oz (35 μm) copper weight. While four-layer boards offer dedicated power and ground planes, a two-layer approach was consciously selected to maintain strict budget constraints and keep manufacturing prices low. The 1 oz copper thickness is highly optimal for this specific application. Under maximum operational load, including the ESP32-S3 transmitting over Wi-Fi, the mmWave radar actively pulsing, and the thermal imager scanning the total continuous current draw is projected to remain beneath one amp. For loads in this range, 1 oz copper provides excellent conductivity without the increased cost or fabrication challenges of heavier 2 oz copper boards.

All trace widths were mathematically determined rather than arbitrarily selected, ensuring full compliance with the IPC-2152 standard for determining current carrying capacity in printed board design. The Saturn PCB Toolkit was leveraged to calculate precise widths based on thermal constraints and current limits. Power delivery network traces were routed with a 0.5 mm track width. According to the calculations for 1 oz copper, this width comfortably supports a one-amp continuous load while restricting the localized thermal rise of the trace to a maximum of 10°C. This thermal headroom is critical for ensuring the longevity of the board, particularly when operating continuously in a consumer-style lamp enclosure. Digital communication lines, such as I2C, SPI, and UART, were routed using a 0.2 mm track width. Utilizing these two distinct track widths establishes a clear visual hierarchy on the board. During the prototyping and validation phases, this visible contrast allows engineers to instantly distinguish between a power rail and a data line, significantly streamlining the manual debugging and probing process.

Managing electromagnetic interference and switching noise is paramount in a system utilizing both sensitive radar sensors and microcontrollers. The power regulation circuitry utilizes Ap63201 adjustable buck converters to step down the input voltage. Because these buck converters generate high frequency switching noise, all inductive noise eliminating elements, including inductors and ferrite beads, were strategically isolated in the far-left corner of the board near the barrel jack input. This geographic isolation prevents magnetic field coupling and switching transients from interfering with the delicate digital signals of the sensors on the opposite side of the board. Furthermore, a cascaded system of regulators was used to increasingly reduce the noise seen by the system from the switching regulators. This means having the switching regulators output being fed to the LT3080 adjustable linear regulators to clean up the very noisy 3.3V and 5V signals before being sent to the rest of the components on the board. Additionally, the regulator board alone was manufactured utilizing a 2oz copper pour to more efficiently handle and dissipate the heat being generated by the five regulators. To further manage transient loads, such as the sudden current spikes drawn by the ESP32-S3 during Wi-Fi transmission, decoupling capacitors were also utilized. These capacitors are placed as physically close to the input pins of their respective integrated circuits as possible. This minimizes the loop inductance between the capacitor and the IC, ensuring immediate current delivery and stabilizing the logic levels.

The physical layout of the board was prioritized to maintain a pristine, low impedance ground return path while ensuring physical durability. Component placement was heavily optimized to keep almost all signal routing on the top layer. By minimizing the number of traces that cross into the bottom layer, the bottom copper acts as a largely uninterrupted, continuous ground pour. This continuous plane is essential for minimizing EMI, providing a solid reference voltage for the sensors, and ensuring proper operation of the microcontroller's onboard antenna. Components were grouped by functional blocks to keep signal traces as short as physically possible, reducing parasitic capacitance and inductance vital for high-speed data transfer. Because this iteration of the board requires manual assembly and testing, the minimum spacing clearance between all components was set to 1mm. This generous spacing provides ample room for the surface tension of solder paste during reflow, prevents solder bridging, and allows sufficient physical clearance for a soldering iron and tweezers.

To facilitate secure installation within the final Noctisense lamp enclosure, four standard M3 mounting holes have been strategically placed at the outer corners of the PCB. Sized to accommodate standard 3mm metric machine screws or standoffs, these mounting points ensure the board remains rigidly anchored. To further aid in the development and diagnostic process, visual indicators and modular connectors were integrated into the top layer. A row of surface-mount LEDs is situated on the top right quadrant of the board to serve as immediate visual indicators for the power delivery network, confirming that both the 5V and 3.3V power rails are active and stable. Additional status LEDs are tied to the MCU to indicate system states like booting, sensor initialization, or active data streaming. The board also features heavily labeled silkscreen headers to securely interface with the external mmWave radar and thermal imaging modules, ensuring the system remains modular and easy to upgrade during iterative testing.

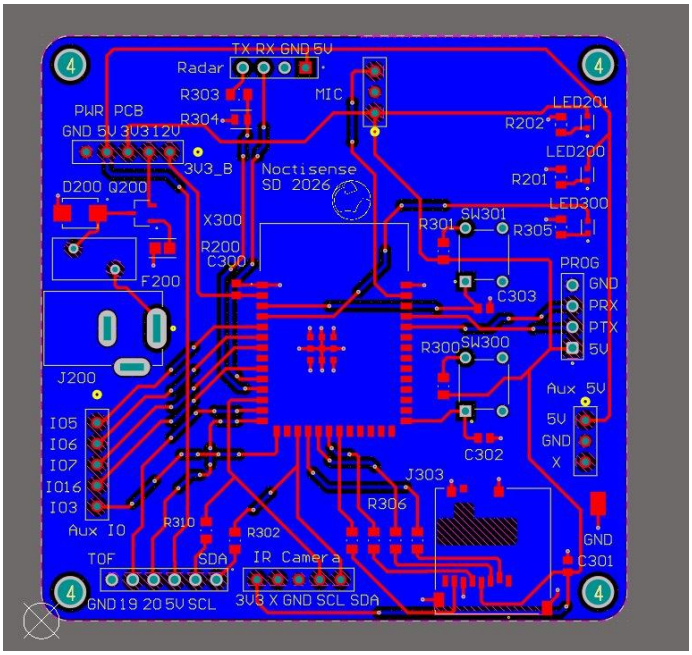


Figure 8.1: *Noctisense PCB*

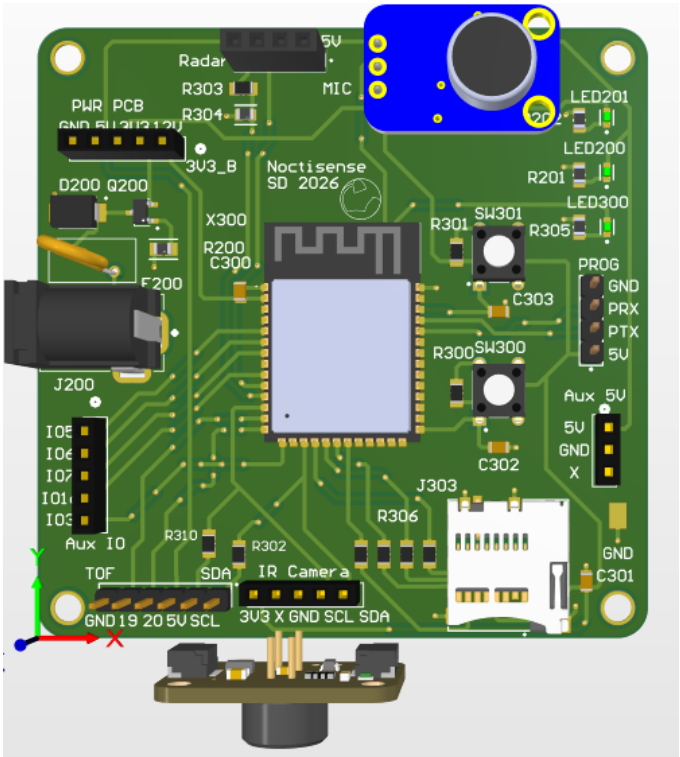


Figure 8.2: *Noctisense PCB 3D View*

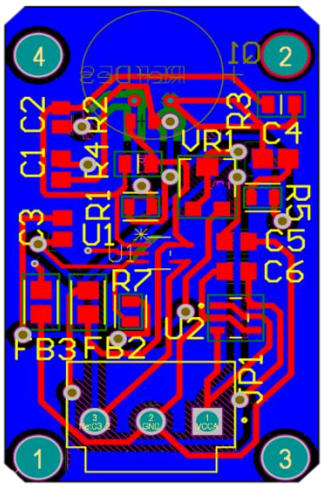


Figure 8.3:*Microphone PCB*

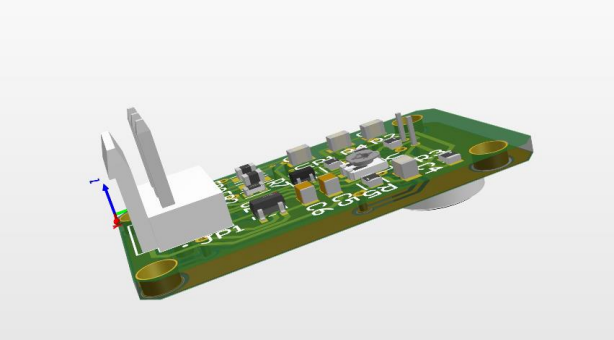


Figure 8.4: *Microphone PCB 3D View*

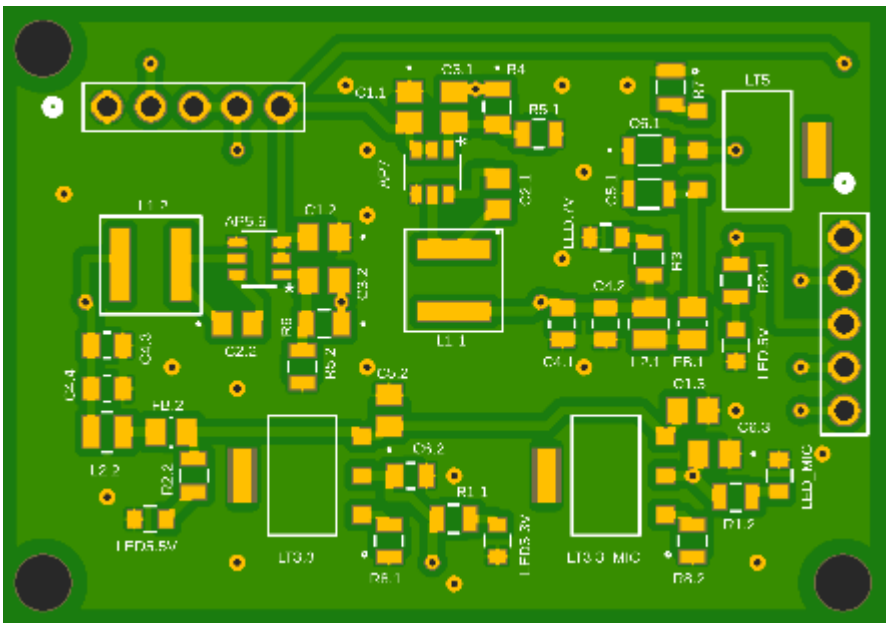


Figure 8.5: *Regulator PCB*

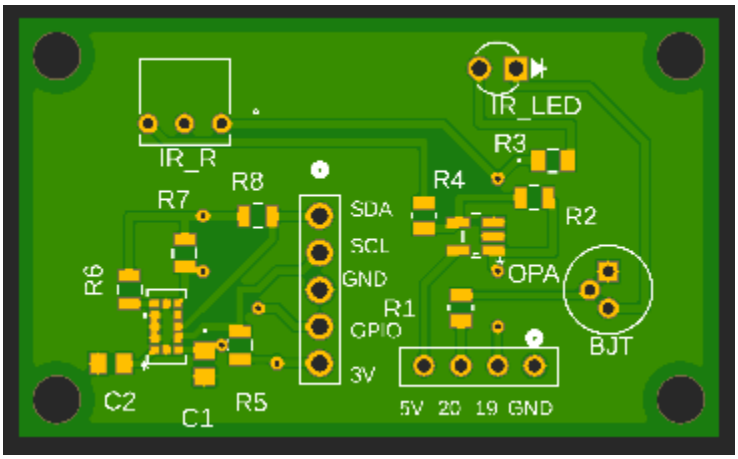


Figure 8.6: Optical PCB

Chapter 9: System Testing and Evaluation

9.1 Optoelectronics feasibility study and testing

The validity of the proposed optical dToF system was evaluated through an optical demonstration designed to prove signal detectability and system feasibility. The primary objective of this experiment was to determine whether a near-infrared (NIR) emitter and photodetector pair could produce a measurable reflected signal at a short working distance, thereby establishing a baseline for future long-range detection.

A 940 nm near-infrared LED was used as the emitter. The LED was drive by a voltage around 1.7 V forward voltage with a series resistor of 1 Ω and corresponding operating current of 100 mA within the expected range for emission. To confirm that the LED was actively emitting radiation in the infrared spectrum (which is not visible to the human eye), a Thorlabs VRC2 IR viewing card was used. This allowed visualization of the emitted beam profile and ensured proper operation of the source.

The optical setup consisted of a white reflective panel placed at a distance of 0.5 meters from the emitter. This reduced distance—approximately one-third of the intended system range—was intentionally chosen to represent a worst-case proof-of-concept scenario, where signal attenuation due to geometric spreading and reflection losses is minimized. By validating detection under these conditions, the experiment establishes a lower-bound feasibility before scaling to larger distances.

A plano-convex lens was introduced into the system to both collimate the emitted light and improve the collection efficiency of the reflected signal. The lens was sourced from the CREOL undergraduate laboratory; however, its material properties (e.g., whether it was N-BK7) and exact optical specifications were not fully characterized. This introduces uncertainty in transmission efficiency at 940 nm, as well as in the effective numerical aperture and focal behavior of the system.

To evaluate temporal response, the LED was modulated using a 50 Hz square wave input. The reflected optical signal was received by a photodiode and observed on an oscilloscope. The presence of a detectable periodic signal corresponding to the input modulation served as a key validation step, confirming that: Optical power was successfully transmitted, reflected, and received; The photodetection system was capable of converting incident optical power into an electrical signal; The system could distinguish signal from background noise under controlled conditions.

In our next revision we will be using a commercial IR LED, used for TV remotes and produces high power. As the LED (borrowed from the same lab) used in the midterm demo might have been defective and not produce as much light, as the 3 other LEDs of the same kind were not working prior. The photodiode, also for the same commercial use, will be used and apply a voltage bias to

increase our output. During the demo the photodiode's lacked a DC bias, it pumped very little current as we needed a high gain transimpedance amplifier.
The optical filter was attempted to use. However, there was a great dense smudge on the filter and or the filter was damaged, it blocked all Near Infrared Light to the photodetector. So the filter was taken out. However, in the actual project we plan using an undamaged filter to maximize the permitted light, while eliminating noise.

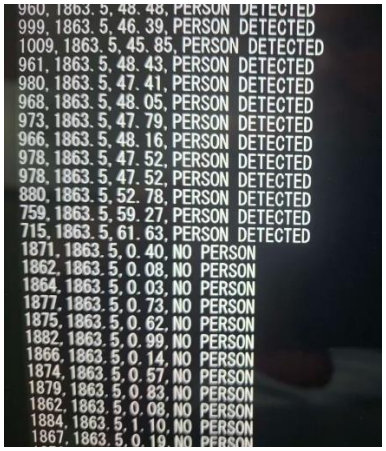
Receiving a signal back, we had success. However, there were limitations as we were setting up for the midterm demo:

The photodiode was operated without a DC reverse bias. As a result, the device operated in photovoltaic mode, producing very low current output. This significantly reduced signal amplitude and required the use of a high-gain transimpedance amplifier, which can introduce bandwidth limitations and noise amplification.
The LED used in the experiment was inconsistent. Three of the same LED type were not producing any light with the regular forward voltage, possibly because of degradation or prior damage. Therefore this brings concern to the validity of the power of the LED we used.
An optical filter was originally supposed to be used to suppress ambient light and improve signal-to-noise ratio. However, the filter had great surface damage (dense smudging or coating damage), which resulted in significant loss in power of the desired 940 nm signal, overall losing the signal. Therefore, the filter was removed from the setup.
The plano-convex lens introduces uncertainty in beam divergence, focusing efficiency, and transmission losses as there was no engraving to tell the model or info of the lens. Which affected the emitter collimation and receiver field-of-view, which are critical parameters for long-distance detection.

The following improvements will be practiced for the next use:

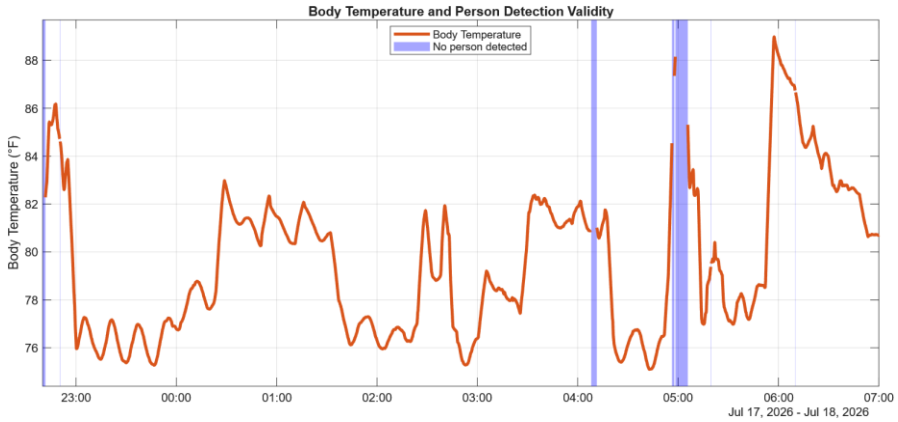
A commercially available infrared LED, commonly used in remote control applications, will be used to ensure higher and more consistent optical output power.
The photodiode will be operated under reverse bias to increase carrier sweep-out speed and improve responsivity, thereby increasing signal amplitude and bandwidth.
A properly characterized and undamaged NIR bandpass filter centered at 940 nm will be used to reduce ambient noise while maximizing transmission of the desired signal.
The project will use lenses with known material properties (e.g., N-BK7 or IR-grade glass) and specified numerical aperture to improve optical efficiency and reproducibility.

A ToF was used to determine the validity of the optical subsystem. Using the same code, however, with distance instead of intensity we can use the same principles and achieve a binary person detection system.
The below figure display the commercial ToF on the left and the optical subsystem on the right. During the general time intervals, the optical subsystem receives a 1 to 1 person detection with the commercial product. When incorporating all subsystems together there is a few second delay, due to the way the json(data) is uploaded to the app. However, maintaining the principle of a person detection the optical subsystem works.



9.1 Figure: *Person Detection Validity test*

Additionally, comparing the optical detection with the IR camera there is also a one-to-one person detection value. This can be seen with the binary detection system displays “No Person” the values of the body temperature of the individual have a NA value at the same time. In other words, when the plot of the Body Temperature breaks or there is a gap in data, there is a blue fill representing at that time there was no one on the bed. This also applies to other testing such as the breathing rate. Below is the graph with the data from one of the sleep testings.



9.2 Figure: *Body Temperature and Person Detection Validity*

9.2 System Initialization Built-In Self-Test (IBIT)

Upon initialization of the system, there are several integrated system configuration commands with integrated tests that verify each component is functioning properly. Each includes their own error handling exceptions in the case of a failure with relevant debug messages visible from the console. First of the checks is the SD-card storage presence and filesystem verification ensuring there is writeable storage with the appropriate directories. This is then followed by configuration of the radar module wherein communication is established, and three configuration data frames are sent and queried in sequence to put it in the correct operational state. Next is the configuration of the WIFI connection enabling tandem connection of the HTTP/TCP servers. Here are confirmation messages seen in the console giving information that our configured settings were deployed successfully. Lastly, the custom audio analyzer is initialized and confirms

this by taking its baseline measurement for room-noise normalization. Appendix A[75] demonstrates an example of a successful boot sequence of the Noctisense system.

9.3 Audio Detection Testing

Testing of the audio detection and recording subsystem for the Noctisense monitoring device was conducted to verify the system's ability to establish an ambient acoustic baseline, detect a sudden audio event, and trigger a localized recording to the onboard SD card. To ensure accurate environmental calibration, an independent acoustic measurement application was used alongside the ESP32-S3 processing unit with the microphone connected. In a closed room simulating a standard monitoring environment, the ambient noise floor was profiled first. The reference meter registered a dynamic range with a current reading hovering around 43.2 dBA but established a more reliable average ambient noise baseline of 54.9 dBA to prevent false positives. Spectral analysis of this ambient noise indicated minor low-frequency resonances with notable peaks concentrated between 280 Hz and 284 Hz.

Figure 9.3: *Baseline App Decibel Readings*

With the 54.9 dBA baseline established and the threshold parameters locked in logic, a high-amplitude audio stimulus specifically a music track by Fetty Wap was introduced to simulate a sudden, loud environmental anomaly. The reference meter successfully captured this spike, registering a maximum peak of 70.7 dBA during the test window. Simultaneously, the ESP32-S3 successfully recognized the threshold breach, proving that the event-triggering logic and interrupts were highly responsive. Upon detecting the anomaly, the system immediately began

the file generation and data logging process. The serial console output confirmed that the firmware correctly processed the 5000 ms recording window requirement, generating sequential filenames such as clip_1.wav and clip_2.wav, and mapping them to the correct /audio/ directory on the SD card. (Note: although we use the same functionality in the final system, it is now in MP3 file format.)

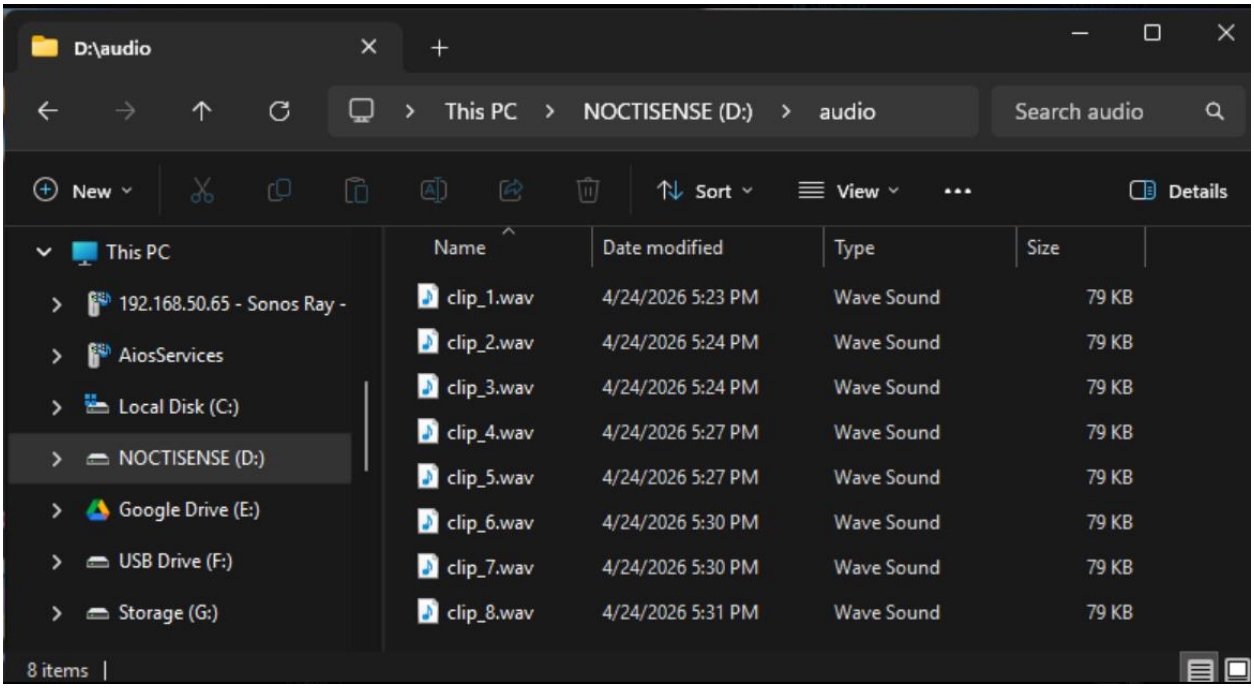


Figure 9.4: *Validation of Audio Being Stored on SD Card*

The successful generation and storage of these audio clips demonstrate that the I2S audio data stream and the FAT32 file writing process are functioning exactly as intended. The acoustic threshold detection algorithm proved to be highly responsive and accurate, reliably transitioning the system from an idle listening state to an active recording state upon detecting anomalous environmental noise. Overall, the test confirms that the audio subsystem is fully operational and effectively capable of capturing and logging localized audio events for the Noctisense system.

9.4 Temperature Reading Validation

Testing of the Noctisense thermal monitoring subsystem was conducted to evaluate the accuracy of the MLX90640 sensor in capturing both ambient room conditions and contactless body temperature over a one-hour continuous period. For the ambient temperature baseline, the system was cross-referenced against a dedicated external environmental sensor. Throughout the hour, the system demonstrated strong stability with a very minimal delta, consistently staying within a 2°F margin of the reference device. As shown in the captured data, the external sensor recorded **75.3°F**, while the system's mobile dashboard logged an average ambient temperature of **77.35°F** over 162 samples. This confirms that the stationary lamp configuration provides a reliable baseline for tracking general room climate.

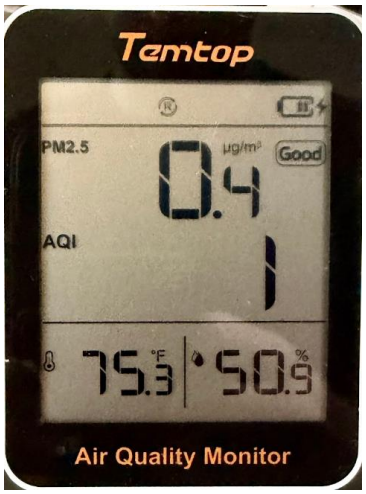


Figure 9.5: *Ambient Temperature Captured by External Thermometer*

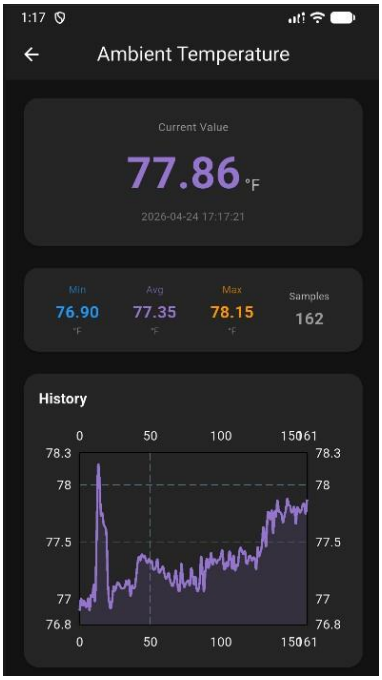


Figure 9.6: *MLX 90640 Temperature Reading*

The second phase of the test evaluated the contactless body temperature readings against a direct-contact external reference (a digital cooking thermometer) over the same one-hour window. This comparison revealed a significant delta between the two instruments. The mobile application recorded an average body temperature of **92.37°F** across 200 samples, which is notably lower than standard human surface temperatures. This discrepancy is primarily attributed to the testing environment; the subject was positioned directly under an active ceiling fan. The forced circulation of cold air, combined with the physical air gap between the subject and the wall-powered device, introduced a pronounced convective cooling effect that skewed the thermal imaging data. It is important to account for this variable moving forward, as running a ceiling fan

or cooling device is a highly common sleep configuration in most bedrooms and perfectly reflects realistic operational conditions.

Figure 9.7: *MLX 90640 Body Temperature Reading*

To immediately address this environmental interference, a rapid software-level compensation algorithm was devised to mathematically reintroduce the cooling ratio back into the data pool. The experimental function applied was

$$T = T_{max} + \left(\frac{T_{ambient}}{T_{min}}\right)$$

Equation 9.1: *Attempted Solution to Body Temperature Reading*

Unfortunately, integrating this algorithm yielded negligible improvements, only raising the final calculated body temperature by approximately **1°F**. Because this adjustment is not significant enough to close the delta caused by ambient airflow, further refinement of the thermal compensation logic will be necessary to ensure high-accuracy biometric logging in drafty sleeping environments.

9.5 Radar System Validation

The 24 GHz mmWave radar subsystem was evaluated with respect to three primary performance metrics: (1) respiratory rate estimation, (2) heart rate detection, (3) user movement, restlessness detection, and threshold based event classification. These metrics collectively assess the subsystem's capability to monitor both periodic physiological signals and non periodic motion within a non-contact sensing framework.

9.5.1 Respiratory Rate Estimation

Respiratory rate measurements were obtained through analysis of periodic chest displacement detected by the radar sensor. The subsystem utilizes variations in the reflected signal to extract breathing patterns, which are subsequently processed to estimate breaths per minute (BPM). Validation was performed through direct comparison with manual breath counting over fixed-duration intervals.

Across multiple trials conducted under controlled indoor conditions, manually measured respiratory rates ranged from approximately 15 to 16 BPM, while radar derived estimates ranged from approximately 14 to 17 BPM. The resulting deviation was generally within ±1 BPM, indicating close agreement between radar based measurements and the reference method. This level of accuracy is consistent with expected performance for non contact radar sensing systems operating under stable conditions.

Testing was conducted with the subject positioned at a fixed distance and orientation relative to the radar module to minimize geometric variability. The subject remained in a resting state to isolate respiration induced motion from other sources of movement. Under these conditions, the radar subsystem consistently detected periodic motion and produced stable respiratory rate estimates.

It is noted that measurement accuracy may be influenced by environmental and operational factors, including subject orientation, distance from the sensor, and the presence of intervening materials such as clothing or bedding. External motion sources within the radar field of view may also introduce interference. Despite these considerations, the subsystem demonstrated reliable respiration detection under representative operating conditions.

9.5.2 Heart Rate Detection

In addition to respiration monitoring, the radar subsystem was evaluated for its ability to detect heart rate through the identification of very small, high-frequency chest micro-movements associated with cardiac activity. Unlike respiration, which produces relatively large amplitude periodic motion, heart rate signals are significantly lower in amplitude and more susceptible to noise and interference.

During testing, the radar subsystem demonstrated the capability to detect periodic signals consistent with expected heart rate ranges under controlled conditions. However, due to the significantly lower signal amplitude and increased sensitivity to environmental noise, heart rate measurements exhibited greater variability compared to respiratory rate estimation. Reliable detection was most successful when the subject remained stationary and external disturbances were minimized.

Several factors were identified as limiting influences on heart rate detection performance. These include sensor-to-subject distance, subject orientation, and the presence of clothing or bedding that may attenuate micro movement signals. Additionally, any concurrent body motion can obscure the relatively small signal associated with cardiac activity, making separation of heart rate from other motion components more challenging.

As a result, while the radar subsystem demonstrates the feasibility of non-contact heart rate detection, the current implementation is more suitable for qualitative observation rather than precise quantitative measurement. Nevertheless, the ability to detect heart rate related signals represents a significant extension of system capability and provides a foundation for further refinement through improved signal processing and filtering techniques.

9.5.3 Movement and Restlessness Detection

In addition to periodic physiological monitoring, the radar subsystem was evaluated for its ability to detect non periodic body motion associated with user movement and restlessness. Such movements produce larger amplitude variations in the reflected radar signal compared to respiration and heart rate signals, allowing for effective differentiation.

Experimental observations confirmed that the subsystem is capable of detecting and distinguishing movement events such as repositioning, shifting, and other gross body motions. These events were observable both in processed outputs and within the mobile application interface, where periods of increased activity corresponded to known instances of subject motion.

Environmental considerations play a significant role in motion detection performance. The radar sensor operates within a defined field of view, and any moving object within this region may contribute to the received signal. During testing, external motion sources such as nearby individuals or pets were occasionally detected as movement events. While this demonstrates the sensitivity of the system, it also highlights the importance of proper sensor placement and environmental awareness in real world deployment.

When operated under controlled conditions with minimal external interference, the subsystem reliably detected user-specific movement events. This capability enables monitoring of restlessness and activity levels, which are important indicators of sleep quality.

Motion and event detection within the radar subsystem is governed by a configurable floating-point threshold parameter implemented within the signal processing algorithm. This threshold defines the minimum signal amplitude required for classification as a significant event and serves as a key mechanism for tuning system sensitivity.

The use of a continuous floating-point parameter allows for fine-grained adjustment of detection behavior. Lower threshold values increase sensitivity, enabling detection of subtle movements and low-amplitude signals, including minor restlessness. Higher threshold values reduce sensitivity to smaller variations, effectively filtering noise and isolating more significant motion events.

During testing, variation of the threshold parameter produced a direct and observable effect on system performance. Lower thresholds resulted in increased event detection frequency, including detection of minor and transient movements. Higher thresholds reduced the number of recorded events, prioritizing only larger, more pronounced motion. This tunability enables adaptation of the system to different operating environments and user conditions.

Event detection was validated through both the mobile application interface and real-time console output. The application displayed recorded motion events and periods of activity, providing a visual representation of system behavior over time. Simultaneously, console output logs generated timestamped messages whenever the detected signal exceeded the defined threshold, confirming that events were being processed in real time by the embedded system.

9.6 HTTP/TCP Server & Application Testing

To test the HTTP/TCP connection functionality along with the companion application, the continuous updating and synchronization accuracy of data was checked against live serial console output. This showed persistent accuracy across all data metrics measured within the system. As well as successful transmission of configuration change data being processed by the system when sent from the application. Below are images outlining the layout of the user interface (UI), as well as demonstrating a successful transmission of configuration change data being processed by the system[74] respectively.

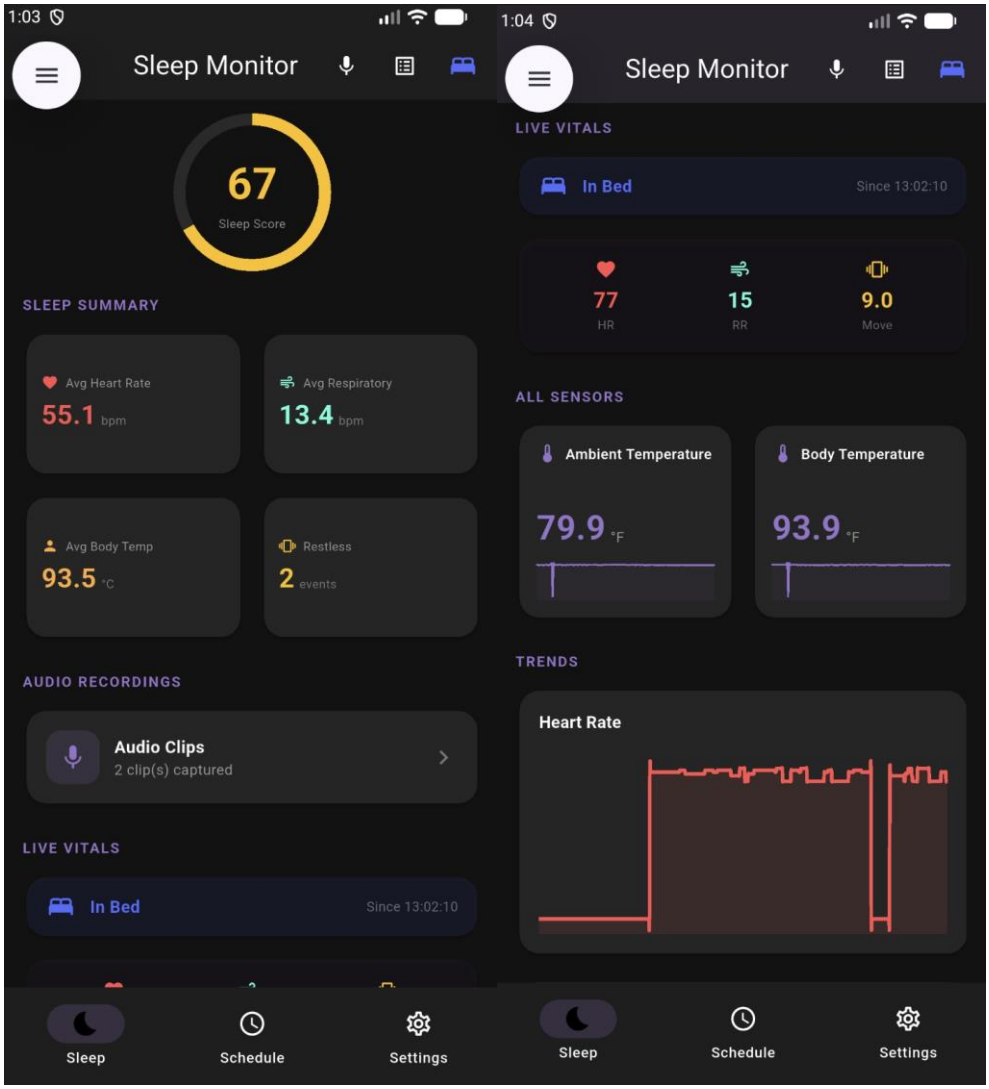


Figure 9.8: *Application User Dashboard View*

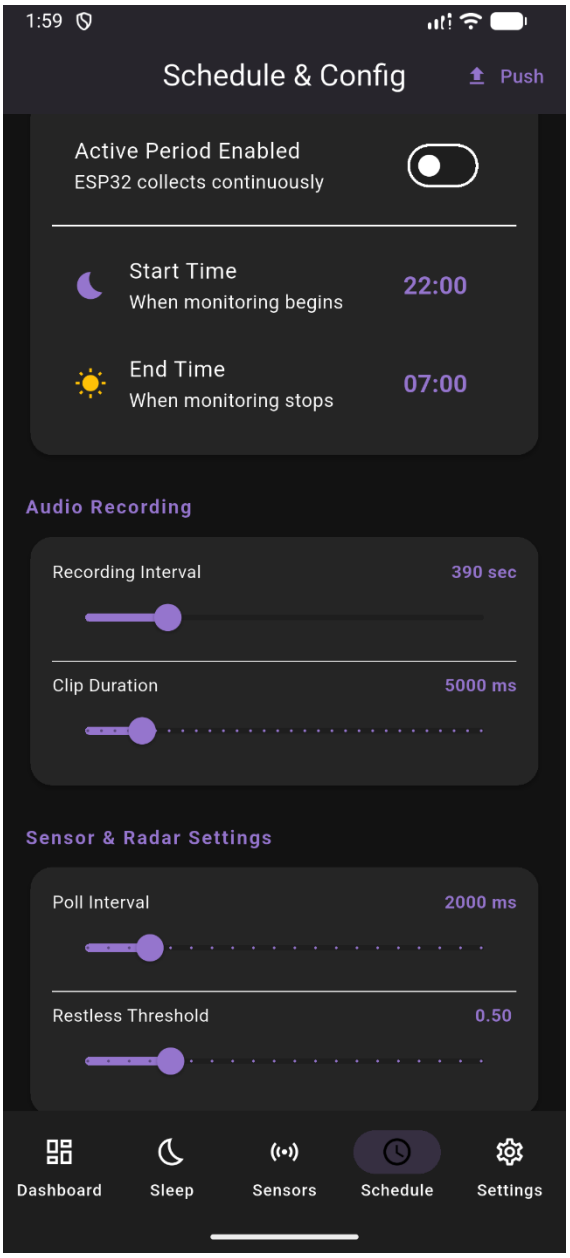


Figure 9.9: *Application Configuration Control View*

9.7 Optical Detection Validation

Testing of the optical detection subsystem was conducted to evaluate the system's ability to reliably detect the presence of a subject using reflected infrared (IR) light at a working distance of 1.5 meters. The system uses the TSAL6400 940 nm LED with a photodiode, transimpedance amplifier (TIA) to convert received optimal voltage readings, and a lens system to optimize as much light to be converted. Detection is performed by comparing real-time signal levels to a dynamically updated baseline reference.

To establish a controlled baseline, the system was first operated in an empty environment with no subject in the target detection distance (1.5 meters). The photodiode output was sampled continuously using the ESP32 ADC (GPIO 34), and an average baseline voltage was computed using an exponential smoothing factor. This baseline generates the base level.

After 20 ms of observational data with 100 samples, a obstacle was placed in the detection area. The obstacle determined a change in optical power in the MCU reading (from Voltage) due to the difference in reflectivity and geometry compared to the background. This produced a consistent deviation in the TIA output voltage, which was captured across multiple sampling windows. The system applies a percentage-based threshold detection method, where the relative drop in signal compared to baseline is used to determine occupancy state.

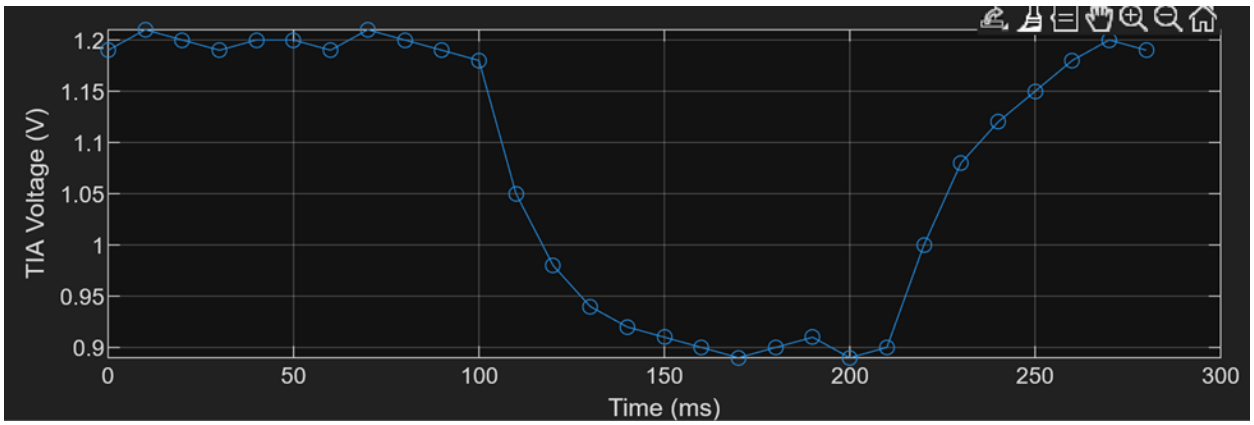


Figure 9.11: *Voltage response from ESP32*

Figure 9.11 shows the voltage response of the system during a detection event. A clear transition is observed as the subject enters the detection region, characterized by a rapid decrease in measured voltage. This corresponds to the obstruction and redistribution of reflected IR light. The system state machine successfully transitions from an EMPTY state to an OCCUPIED state after meeting the required detection frame count, demonstrating stability against transient noise.

Overall, the optical subsystem demonstrated reliable detection capability at the target range, with consistent signal separation between empty and occupied states. However, the system remains sensitive to environmental factors such as ambient light, surface reflectivity, and optical alignment. Future improvements will focus on increasing signal-to-noise ratio through optical filtering, improved collimation, and higher responsivity photodiodes to enhance detection of robustness under varying conditions.

9.8 Regulation Verification

To fully ensure that the regulator board is functioning correctly, testing the power system under load is essential before trusting is to power the sensitive Noctisense system. Referring to the Power Budget table and considering that the mic has its own dedicated power regulator, the maximum possible current load that can be drawn from one regulator is approximately 600ma. As such, all load tests were carried out under a 600ma constant load. During this load testing, a B&K Precision 8500B electronic load generator was used to lead tests in conjunction with banana leads to reach

the leads. The banana leads are rather lossy in this case, losing half of one volt across their length. As such to measure the voltage being produced by the regulators, an external multimeter was used to measure voltage at the pin outs. The output of the 5V LT380 under 0.6A of current load was measured to be 5.084V on the Fluke 106 Multimeter at the output header pin as shown in figure 9.12 below. The output of the 3.3V LT3080 that is powering all 3.3V needs except for the mic measures at 3.167V under the same load conditions as shown in figure 9.13 Below. Finally, the 3.3V LT3080 that supplies power to only the mic measures in at 3.231V at the output pin as shown in figure 9.14 below.

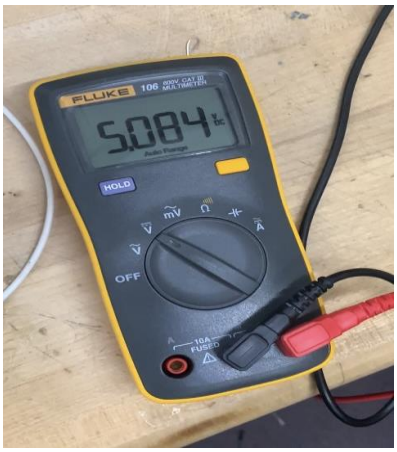


Figure 9.12: 5V Regulator Output Under Load

Figure 9.13: 3.3V General Regulator Output Under Load

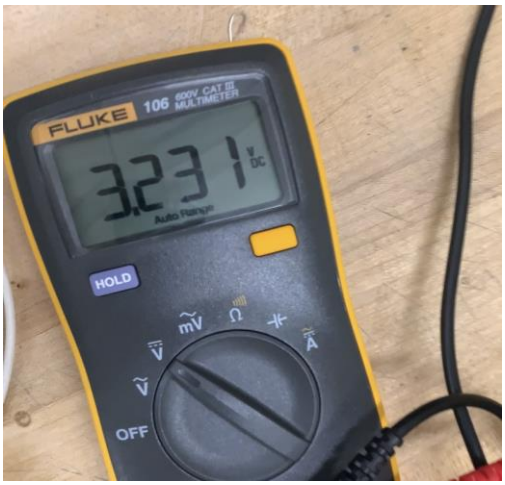


Figure 9.14: 3.3V Mic Regulator Output Under Load

Chapter 10: Administrative Content

10.1 Budget & Financing

Our group, composed of two Computer Engineers (CpE), one electrical engineer (EE), and a Photonics Science Engineer (PSE) plan on having a budget of $1000 total. The budget is not only composed of core system components but also also for redundancy, prototyping materials, fluctuating market prices, and unforeseen failures. Anticipated risks include diode burnout due to misuse or defects, shipping damage or loss, PCB fabrication errors, and component incompatibility during system iteration. Incorporating redundancy reduces the likelihood of schedule delays and ensures project continuity.

Photonics components, including the 940 nm laser diode, photodiode, and wavelength-appropriate optical elements, represent a significant portion of the budget. Operating in the near-infrared improves eye safety and user comfort but increases cost relative to visible-wavelength alternatives due to reduced component availability and tighter performance tolerances. These components are necessary to achieve reliable optical subsystem respiratory detection.

Electrical engineering costs cover analog signal conditioning, power regulation, AC-DC conversion, and PCB fabrication. These elements ensure stable, low-noise operation of sensitive analog and mixed-signal circuitry. Redundant power components and allowance for PCB re-fabrication are included to mitigate risks associated with component failure, manufacturing errors, and design iteration.

Computer engineering costs support embedded processing, storage, wireless communication, and sensor interfacing. The microcontroller, storage media, radar module, microphone, and thermal imaging array enable local data processing, event detection, and synchronization with a mobile application over an HTTP/LAN connection. Costs reflect the need for sufficient processing capability, memory, and interface compatibility across all sensors.

Several optical mounts and mechanical components will be loaned from CREOL's Undergraduate and Senior Design laboratories with approval, reducing overall expenditure while maintaining access to precision hardware. The remaining budget margin accounts for inflation, tariffs, shipping costs, and unforeseen integration challenges.

A detailed cost breakdown is provided below, reflecting current price estimates as of February 2026. The majority of funding will be supplied through CREOL's Senior Design Scholarship, which will support component procurement and system development.

Table 10.1 Budget

Component	Part Number	Quantity	Unit Price	Total Price
PSE				
TSAL6400, High Power Infrared Emitting Diode, 940 nm, GaAlAs, MQW	Digi-Key TSAL6400	10	$0.48	$5.14
MTD8600N4-T (Phototransistor)	DigiKey 475-SFH2400FA-ZCT-ND	5	$6.64	$33.20
ACL25416U-BAspheric Condenser Lens, Ø1", f=16 mm, NA=0.79, ARC: 650-1050 nm	Thorlabs ACL25416U-B	4	$34.33	$137.32
Miscellaneous	NA	1	NA	$150
EE				
Transimpedence Amplifier	AliExpress	1	$6	$6
12 V 2 A AC-DC Adapter	Jameco Part no: 2350087	1	$15.00	$15.00
Barrel Jack (5.5 mm × 2.5 mm)	Digi-key: CP-002B-ND	1	$1.50	$1.50
Buck Regulators	AP63201	2	$2	$3
Low Noise Linear Regulators	LT3080	3	$6	$18

Capacitors (Filtering and decoupling)	—	16	$1	$16
Resistors	—	14	$0.30	$4.20
Inductors	—	4	$2.00	$8
Protection Circuitry (Including Backups)	See table below for full part breakdown	—	—	$13.50
LED / Misc.	—	—	$10	$10
PCB Fabrication	—	1	$80	$80
CPE				
Storage	Amazon	1	$25	$25
MCU	Amazon	1	$10	$10
Radar Module	DigiKey 1891-XM132-ND	1	$30	$30
Microphone	Own already	1	$7	$7
MLX90640 IR Array Thermal Imaging Camera 32×24 Pixels 55° Field of View I2C Interface 3.3V/5V Compatible with Raspberry Pi (ESP32) STM32	MLX90640-D55 Thermal Camera	1	$66.29	$70
SD-Card Port	Own already	1	$7	$7
PCB Fabrication		1	$80	$80
Miscellaneous				
Safety Net		1	$100	$109.43
Total Estimate				$1000

Table 10.2 Protection Circuity BOM Including Backups

Component Description	Function	Manufacturer	Part Number	Quantity	Unit Cost (USD)	Total Cost (USD)
Resettable Fuse (PTC), 2 A Hold	Overcurrent protection	Bourns	MF-R200	2	$0.75	$1.50
P-Channel MOSFET, −30 V, Low Rds(on)	Reverse polarity protection	Alpha & Omega	AO3407A	2	$0.50	$1.00
TVS Diode, 15 V Clamp	Surge/transient suppression	Littelfuse	SMBJ15A	2	$0.60	$1.20
Ferrite Bead, 100 Ω @ 100 MHz	EMI filtering (power rails)	Murata	BLM21PG101SN1D	8	$0.20	$1.60
Electrolytic Capacitor, 220 µF, 25 V	Input bulk filtering	Panasonic	EEU-FR1E221	2	$0.50	$1.00
Electrolytic Capacitor, 470 µF, 10 V	Load transient buffering	Nichicon	UPM1A471MPD	4	$0.50	$2.00
Ceramic Capacitor, 0.1 µF, 50 V (X7R)	High-frequency decoupling	Murata	GRM188R71H104KA93D	12	$0.10	$1.20

Ceramic Capacitor, 10 µF, 10 V (X5R/X7R)	Rail stabilization	Murata	GRM21BR61A106KE51L	8	$0.15	$1.20
Ceramic Capacitor, 1 µF, 10 V	LDO stability (input/output)	Murata	GRM188R61A105KA61D	8	$0.10	$0.80
ESD Protection Diode Array, 5 V	ESD protection for I/O lines	Nexperia	PESD5V0S1UL	4	$0.40	$1.60
Gate Resistor (MOSFET), 100 kΩ	Gate biasing	Yageo	RC0402FR-07100KL	2	$0.05	$0.10
Series Resistor (RC filter), 10 Ω	Analog rail filtering	Yageo	RC0402FR-0710RL	4	$0.05	$0.20
TOTAL	—	—	—	—	—	$13.50

10.2 Project Milestones (SD1 - SD2)

Table 10.3 General Milestones SD1

Week Number(s) | End Date(s)	Description
1 | 1/16/2026	Familiarization with course content and begin group formation.
2 | 1/23/2026	Form groups, finalize project ideas, research project ideas, and work on D & C proposal documents.

3 | 1/30/2026	Finish and turn in the D & C project proposal document.
4 | 2/6/2026	Revise the original D & C document and turn in the revision.
5 - 6 | 2/20/2026	Finalize initial parts, design, and application research.
7 | 2/27/2026	30 page milestone
7 - 8 | 3/6/2026	Initial isolated prototyping and testing phase
9 - 10 | 3/20/2026	60 page milestone
11 | 3/27/2026	Turn in the midterm report
11 - 13 | 3/21/2026 - 4/10/2026	Prototype, test, and integration phase and 90 page milestone.
14 | 4/17/2026	Continue prototyping, testing, integration, and implementation including initial PCB Design.
15 | 4/24/2026	120 page milestone and put finishing touches on the final document
16 | 4/28/2026	Turn in Final Documentation

Table 10.4 General Milestones SD2

Week Number(s) | End Date(s)	Description
1 | 5/23/2026	Append Final SD1 as needed
2 | 5/30/2026	Complete any final research and continue development of PCB/software design.
3 - 4 | 6/13/2026	Finish initial PCB design/software design and order all parts needed to realize the PCB design
5 | 6/20/2026	Wait for parts to arrive and continue prototyping and testing of physical hardware as well as mobile application.
6 - 8 | 6/21/2026 - 7/11/2026	Assemble PCB, test and troubleshoot initial PCB, and order any final revision,

	replacement, or backup parts.
9 | 7/18/2026	Wait for parts to arrive and continue prototyping and testing.
10 - 11 | 7/29/2026 - 8/1/2026	Assemble the final PCB, realize final design, and work on the final presentation and its script.
12 | 8/3/2026 - 8/5/2026	Final Presentation

Table 10.5 Detailed Electrical Milestones SD1

Week Number(s) | End Date(s)	Description
1 - 4 | 2/6/2026	General SD Milestone Table
5 - 6 | 2/20/2026	Research all power delivery needs of all the parts in the project, and begin to design the power delivery stage.
7 - 8 | 3/6/2026	Design Regulator(s) PCB(s) and order the parts needed to build them
9 - 10 | 3/20/2026	Put together the regulators and order any additional, replacement, or backup parts needed for prototyping.
11 - 14 | 3/21/2026 - 4/17/2026	Learn the micro controller and work between all members to prototype, test, and integrate each sensor and the information it provides together with the software.
15 - 16 | 4/28/2026	General SD Milestone Table

Table 10.6 Detailed Electrical Milestones SD2

Week Number(s) | End Date(s)	Description
1 - 2 | 5/30/2026	General SD Milestone Table
3 - 4 | 6/13/2026	Finish Initial PCB design and order all of the parts needed to realize the initial PCB
5 | 6/20/2026	Continue to prototype and test while waiting

	on parts
6 - 8 | 6/21/2026 - 7/11/2026	Assemble PCB, test/ troubleshoot functionality and power delivery, and order any final revision, replacement, or backup parts.
9 | 7/18/2026	Continue to test and troubleshoot while waiting on parts to arrive.
10 - 12 | 7/29/2026 - 8/5/2026	General SD Milestone Table

Table 10.7 Detailed Computer Milestones SD1

Week Number(s) | End Date(s)	Description
1 - 4 | 2/6/2026	General SD Milestone Table
5 - 6 | 2/20/2026	Research System design, Architecture of and application/UI, and the thermal imaging/laser modules.
7 - 8 | 3/6/2026	Begin testing sensor algorithm test cases, testing software prototypes, and work on algorithm sampling/data flow.
9 - 10 | 3/20/2026	Continue prototyping, testing, and refining algorithms.
11 - 14 | 3/21/2026 - 4/17/2026	Begin mobile application testing and integration and General SD Milestone Table.
15 - 16 | 4/28/2026	General SD Milestone Table

Table 10.8 Detailed Computer Milestones SD2

Week Number(s) | End Date(s)	Description
1 - 2 | 5/30/2026	General SD Milestone Table
3 - 4 | 6/13/2026	Continue application development/user-testing, finalize overall system features, finish MCU PCB design and order parts needed to realize the MCU PCB.
5 | 6/20/2026	Finalize application features and ensure core functionality while waiting on parts.
6 - 8 | 6/21/2026 - 7/11/2026	Assemble PCB, test and troubleshoot initial PCB, and order any final revision, replacement, or backup parts.
9 | 7/18/2026	Finalize the application
10 - 12 | 7/29/2026 - 8/5/2026	General SD Milestone Table

Table 10.9 Detailed Photonics Milestones SD1

Week Number(s) | End Date(s)	Description
1 - 4 | 2/6/2026	General SD Milestone Table
5 - 6 | 2/20/2026	Define optical and system-level requirements (displacement sensitivity, bandwidth, SNR, alignment tolerances).
7 - 8 | 3/6/2026	Prepare for Optical progress demo (3/3) and imitate similar optical design in undergraduate lab for potential concerns. Finalize parameters and adjustments to the component list and order all optical parts needed to prototype.
9 - 10 | 3/20/2026	Continue research on the ordered optical components and begin aligning .
11 - 14 | 3/21/2026 - 4/17/2026	Integrate photodetection, signal conditioning, and MCU. Perform displacement validation using external device
15 - 16 | 4/28/2026	General SD Milestone Table

Table 10.10 Detailed Photonics Milestones SD2

Week Number(s) | End Date(s)	Description
1 - 2 | 5/30/2026	General SD Milestone Table
3 - 4 | 6/13/2026	Improve stability & robustness, such as drift characterization and temperature sensitivity.
5 | 6/20/2026	Develop and optimize signal processing algorithms for real-time displacement and respiratory rate extraction.
6 - 8 | 6/21/2026 - 7/11/2026	Enclosed housing and produce repeatable breathing rate
9 | 7/18/2026	Finalize testing and work on presentation
10 - 12 | 7/29/2026 - 8/5/2026	General SD Milestone Table

Chapter 11: Conclusion

In conclusion, Noctisense as a project embodies our team's goal of implementing a complete cohesive system for tracking biometric sleep data. All with a focus on affordability, security, and unintrusive design. This project represents a development filling a major gap within the consumer and healthcare markets. As outlined previously in our report, Noctisense utilizes a suite of sensors to collect relevant data for end-users. Spanning 24 GHz mmWave radar for movement, respiratory rate, and heart rate detection, infrared thermal imaging for body and ambient temperature monitoring, audio analysis for auditory anomaly detection, and pulsed direct time-of-flight measurements for presence detection. Each of these subsystems was carefully selected and integrated to work in concert, delivering a comprehensive sleep profile without requiring any physical contact with the user.

Beyond the hardware, the system's software architecture is built around a locally-hosted HTTP/TCP local network server driving a companion mobile application. This reinforces our commitment to user privacy and data ownership. In an era where subscription fatigue and cloud dependency have become the norm, Noctisense demonstrates that a high-quality, consumer-grade biometric device can operate entirely on-device without sacrificing functionality or user experience.

Developed entirely by a team of engineering students at the University of Central Florida using low-cost and custom-fabricated components, Noctisense proves that the gap between clinical-grade sleep monitoring and accessible consumer technology is one that can be meaningfully closed. The limitations encountered throughout development have been documented and serve as a foundation for future iterations of the system. It is our hope that this work serves not only as a record of what was built, but as a roadmap for what this platform can ultimately become.

Appendix A: Citations

1. M. R. Lujan, I. Perez-Pozuelo, and M. A. Grandner, "Past, Present, and Future of Multisensory Wearable Technology to Monitor Sleep and Circadian Rhythms," *Front. Digit. Health*, [Online]. Available: https://pmc.ncbi.nlm.nih.gov/articles/PMC8521807/. [Accessed: Jan. 27, 2026].
2. P. Dhunoo, "The Digital Evolution of Sleep Tracking," *The Medical Futurist*, [Online]. Available: https://medicalfuturist.com/the-digital-evolution-of-sleep-tracking. [Accessed: Jan. 27, 2026].
3. Global Market Insights, "Personal Health Record Software Market Size," [Online]. Available: https://www.gminsights.com/industry-analysis/personal-health-record-software-market. [Accessed: Jan. 27, 2026].
4. A. Lazar, C. Koehler, T. Tanenbaum, and D. Nguyen, "Why we use and abandon smart devices," in *Proc. ACM Int. Joint Conf. Pervasive Ubiquitous Comput. (UbiComp)*, 2015, [Online]. Available: https://dl.acm.org/doi/abs/10.1145/2750858.2804288. [Accessed: Jan. 27, 2026].
5. D. Levi, "SaaS or Scam as a Service? The growing backlash against subscriptions," *Tech Startups*, May 17, 2025. [Online]. Available: https://techstartups.com/2025/05/17/saas-or-scam-as-a-service-the-growing-backlash-against-subscriptions/. [Accessed: Jan. 27, 2026].
6. V. Tangermann, "Dell Admits That Customers Are Disgusted by PCs Stuffed With AI Features," *Futurism*, [Online]. Available: https://futurism.com/artificial-intelligence/dell-admits-customers-disgusted-pcs-ai. [Accessed: Jan. 27, 2026].
7. *Apple Inc., "Estimating Sleep Stages from Apple Watch," Oct. 2025. [Online]. Available: https://www.apple.com/health/pdf/Estimating_Sleep_Stages_from_Apple_Watch_Oct_2025.pdf.*
8. Eight Sleep, "How the Pod Detects Your Breathing and Heartbeats Without a Wearable," *Eight Sleep Blog*, [Online]. Available: http://www.eightsleep.com/blog/how-the-pod-detects-your-breathing-and-heartbeats-without-a-wearable/. [Accessed: Jan. 30, 2026].
9. *Sleep Cycle, "Phone Placement with the Sleep Cycle App," Sleep Cycle Support, 2016. [Online]. Available: https://support.sleepcycle.com/hc/en-us/articles/207388465-Phone-Placement-with-the-Sleep-Cycle-App.*
10. G. Sun, T. Negishi, T. Kirimoto, T. Matsui, and S. Abe, "Noncontact Monitoring of Vital Signs with RGB and Infrared Camera and Its Application to Screening of Potential Infection," *IntechOpen*, [Online]. Available: https://www.intechopen.com/chapters/63842.
11.
12. J. Sauer, F. Schmälzle, M. Beck, C. Geiger, and H. Steger, "Using Laser Doppler Vibrometry," Polytec, White Paper. [Online]. Available: https://www.polytec-cn.com/uploads/soft/20200418/Polytec-white-paper.pdf.
13. C. Rembe and L. Mignanel, "Introduction to Laser-Doppler Vibrometry," in *Laser Doppler Vibrometry for Non-Contact Diagnostics*, Springer, 2021, ch. 2. [Online]. Available: https://link.springer.com/content/pdf/10.1007/978-3-030-46691-6_2.pdf.

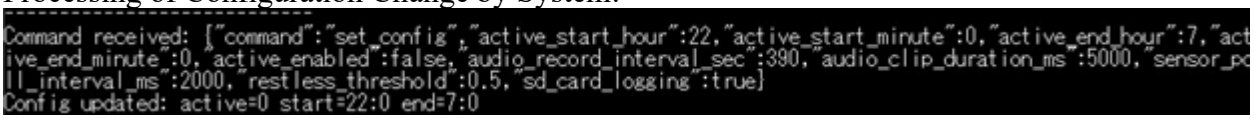
14. P. O'Donoughue et al., "Comparison of three full-field optical measurement techniques applied to vibration analysis," Scientific Reports, vol. 13, no. 3261, 2023, doi: 10.1038/s41598-023-30053-9.41598_2023_Article_30053
15. S. Que et al., "Speckle Vibrometry for Contactless Instantaneous Heart Rate and Respiration Rate Monitoring on Mechanically Ventilated Patients," Sensors, vol. 24, no. 19, p. 6374, 2024, doi: 10.3390/s24196374.
16. G. Sun et al., "Noncontact Monitoring of Vital Signs with RGB and Infrared Camera and Its Application to Screening of Potential Infection," in Non-Invasive Diagnostic Methods – Image Processing, IntechOpen, 2016, doi: 10.5772/intechopen.80652.
17. W. Bachir, A. F. K. Khamaysa, and O. Iwasińska-Kowalska, "Noncontact Measurement of Respiratory Rate by an Optical Time of Flight Sensor," IEEE Access, vol. 12, pp. 156442–156455, 2024, doi: 10.1109/ACCESS.2024.3484992.
18. J. A. Bigalke, E. L. Cleveland, E. Barkstrom, J. E. Gonzalez, and J. R. Carter, "Core body temperature changes before sleep are associated with nocturnal heart rate variability," *Journal of Applied Physiology*, vol. 135, no. 1, pp. 136–145, Jul. 2023, doi: https://doi.org/10.1152/japplphysiol.00020.2023.
19. "24GHz mmWave Sleep Breathing Monitoring | Seeed Studio Wiki," *Seeedstudio.com*, Jan. 12, 2023. https://wiki.seeedstudio.com/Radar_MR24BSD1/ (accessed Mar. 06, 2026).
20. "ESP32: Guide for MicroSD Card Module Arduino | Random Nerd Tutorials," Mar. 12, 2021. https://randomnerdtutorials.com/esp32-microsd-card-arduino/
21. W3C, "Understanding Success Criterion 1.4.3: Contrast (Minimum)," *www.w3.org*, Jun. 20, 2023. https://www.w3.org/WAI/WCAG21/Understanding/contrast-minimum.html
22. International Electrotechnical Commission, *"IEC 60950-1: Information Technology Equipment—Safety—Part 1: General Requirements,"* 2005.
23. International Electrotechnical Commission, *"IEC 62368-1: Audio/Video, Information and Communication Technology Equipment—Part 1: Safety Requirements,"* 2018.
24. International Electrotechnical Commission, *"IEC 61000-4-2: Electromagnetic Compatibility (EMC)—Part 4-2: Testing and Measurement Techniques—Electrostatic Discharge Immunity Test,"* 2008.
25. International Electrotechnical Commission, *"IEC 61000-4-3: Electromagnetic Compatibility (EMC)—Part 4-3: Testing and Measurement Techniques—Radiated, Radio-Frequency, Electromagnetic Field Immunity Test."*
26. CISPR, *"CISPR 32: Electromagnetic Compatibility of Multimedia Equipment—Emission Requirements."*
27. Federal Communications Commission, *"FCC Part 15: Radio Frequency Devices,"* Code of Federal Regulations.
28. P. T. Krein, *Elements of Power Electronics.* New York, NY, USA: Oxford University Press, 1998.
29. H. W. Ott, *Electromagnetic Compatibility Engineering.* Hoboken, NJ, USA: John Wiley & Sons, 2009.
30. Texas Instruments, *"LP2985 Low-Noise, Low-Dropout Regulator Datasheet,"* 2019.
31. Texas Instruments, *"TLV755xx/TLV755xxA Low Dropout (LDO) Regulator Datasheet,"* 2023.
32. Monolithic Power Systems, *"MP1584EN Step-Down Converter Datasheet,"* 2021.
33. Bourns, *"MF-R200 Resettable Fuse Datasheet."*

34. Alpha & Omega Semiconductor, *"AO3407A P-Channel MOSFET Datasheet."*
35. Littelfuse, *"SMBJ15A TVS Diode Datasheet."*
36. Nexperia, *"PESD5V0S1UL ESD Protection Diode Datasheet,"* 2022.
37. Murata Manufacturing, *"BLM21PG101SN1D Ferrite Bead Datasheet."*
38. Murata Manufacturing, *"GRM Series Ceramic Capacitor Datasheet."*
39. Panasonic, *"EEU-FR Series Aluminum Electrolytic Capacitors Datasheet,"* 2020.
40. Nichicon Corporation, *"UPM Series Aluminum Electrolytic Capacitors Datasheet."*
41. Yageo Corporation, *"RC Series Chip Resistors Datasheet."*
42. Espressif Systems, *"ESP32-S3 Series Microcontroller Datasheet."*
43. Melexis, *"MLX90640 Infrared Thermal Sensor Datasheet."*
44. Texas Instruments (or applicable manufacturer), *"24 GHz mmWave Radar Sensor Datasheet."*
45. Jameco Electronics, *"ReliaPro 12 V, 2 A AC-DC Wall Adapter Datasheet/Product Page."*
46. CUI Inc., *"SDI24-12-U 12 V, 2 A AC-DC Power Adapter Datasheet."*
47. R. W. Erickson and D. Maksimović, *Fundamentals of Power Electronics,* 2nd ed. Norwell, MA, USA: Springer, 2001.
48. Seeed Studio, "24GHz mmWave Sensor - Sleep Breathing Monitoring (MR24BSD1)," *Seeed Studio Wiki*, [Online]. Available: https://wiki.seeedstudio.com/Radar_MR24BSD1/. [Accessed: Mar. 12, 2026].
49. A. Al-Naji, A. J. Al-Askery, S. K. Gharghan, and J. Chahl, "A System for Monitoring Breathing Activity Using an Ultrasonic Radar Detection with Low Power Consumption," *J. Sens. Actuator Netw.*, vol. 8, no. 2, p. 32, 2019. [Online]. Available: https://doi.org/10.3390/jsan8020032.
50. DFRobot, "24GHz mmWave Human Presence Detection Sensor," *DFRobot Wiki*, [Online]. Available: https://wiki.dfrobot.com/sen0395. [Accessed: Mar. 12, 2026].
51. U.S. Government Publishing Office, "Title 47, Chapter I, Subchapter A, Part 15, Subpart C — Intentional Radiators," *Code of Federal Regulations*, [Online]. Available: https://www.ecfr.gov/current/title-47/chapter-I/subchapter-A/part-15/subpart-C. [Accessed: Mar. 12, 2026].
52. European Telecommunications Standards Institute (ETSI), "CE Radio Equipment Directive (RED)," [Online]. Available: https://www.etsi.org/technologies/radio. [Accessed: Mar. 12, 2026].
53. International Electrotechnical Commission, "IEC 62368-1: Audio/Video, Information and Communication Technology Equipment — Safety Requirements," IEC, Geneva, Switzerland, Standard IEC 62368-1, [Online]. Available: https://webstore.iec.ch/en/publication/69308. [Accessed: Mar. 12, 2026].
54. International Electrotechnical Commission, "CISPR 32: Electromagnetic Compatibility of Multimedia Equipment — Emission Requirements," IEC, Geneva, Switzerland, Standard CISPR 32, [Online]. Available: https://webstore.iec.ch/en/publication/22046. [Accessed: Mar. 12, 2026].
55. International Electrotechnical Commission, "IEC 61000-4-2: Electromagnetic Compatibility (EMC) — Testing and Measurement Techniques — Electrostatic Discharge Immunity Test," IEC, Geneva, Switzerland, Standard IEC 61000-4-2, [Online]. Available: https://webstore.iec.ch/en/publication/68954. [Accessed: Mar. 12, 2026].

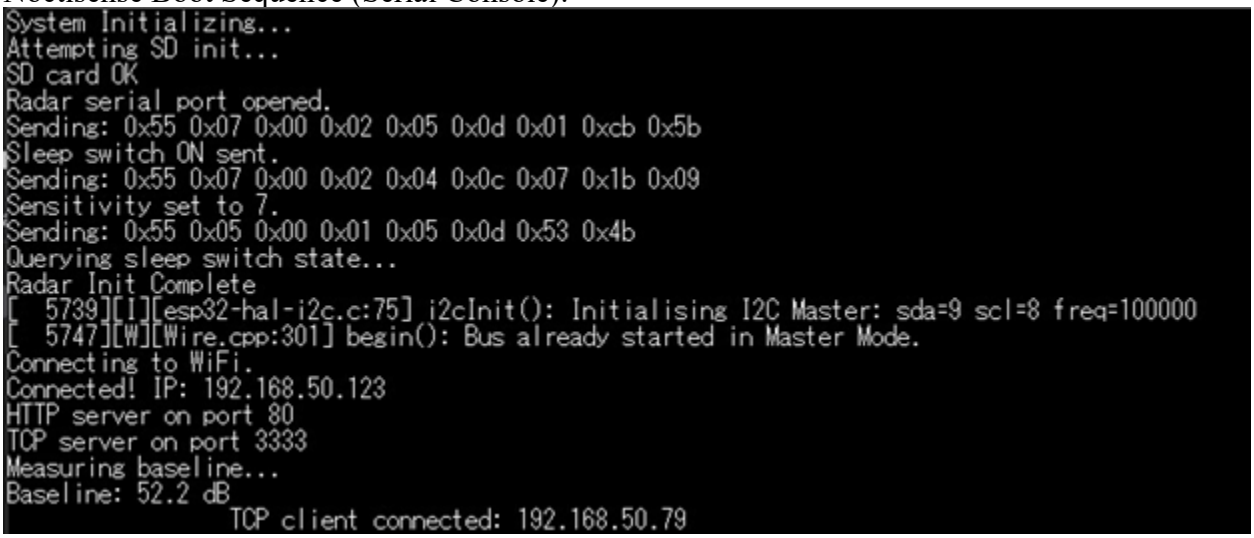
56. Underwriters Laboratories, "UL 62368-1: Audio/Video, Information and Communication Technology Equipment — Safety Requirements," ANSI/UL, Standard UL 62368-1, [Online]. Available: https://webstore.ansi.org/standards/ul/ul62368ed2025. [Accessed: Mar. 12, 2026].
57. National Institute of Standards and Technology, "Security and Privacy Controls for Information Systems and Organizations," U.S. Dept. Commerce, Washington, DC, NIST Special Publication 800-53 Rev. 5 (upd. 1), [Online]. Available: https://csrc.nist.gov/pubs/sp/800/53/r5/upd1/final. [Accessed: Mar. 12, 2026].
58. Rajan, K., Samykano, M., Kadirgama, K. *et al.* "Fused deposition modeling: process, materials, parameters, properties, and applications." Accessed 22 March 2026 [Online]. Available: https://doi.org/10.1007/s00170-022-08860-7
59. Cloudflare, "What is a large language model (LLM)?," Accessed 20 March 2026 [Online]. Available: *Cloudflare.com*. https://www.cloudflare.com/learning/ai/what-is-large-language-model/
60. "Thorlabs," *Thorlabs.com*, 2026. https://www.thorlabs.com/n-bk7-plano-convex-lenses-ar-coating-400---1100-nm?tabName=Overview (accessed Mar. 26, 2026).
61. "Thorlabs," *Thorlabs.com*, 2026. https://www.thorlabs.com/lens-tutorial?tabName=Aspheric%20Lenses (accessed Mar. 26, 2026).
62. "Thorlabs," *Thorlabs.com*, 2026. https://www.thorlabs.com/molded-glass-aspheric-lenses-600---1050-or-650---1050-nm-ar-coating?tabName=Collimation%20Tutorial (accessed Mar. 26, 2026).
63. UCF Department of Environmental Health and Safety. "Laser Safety Manual." Accessed: Feb 12, 2026. [Online]. Available: https://ehs.ucf.edu/wp-content/uploads/sites/3/2023/02/Laser-Safety-Manual.pdf
64. newtonn2, "How to Make a Small DIY LED Projector," Instructables, Circuits section. [Online].Available:https://www.instructables.com/How-to-make-a-small-Diy-LED-Projector/. [Accessed: Feb. 12, 2026].
65. Hesamh, "DIY 3D Scanner Based on Structured Light and Stereo Vision in Python Language," Instructables, Design section. [Online]. Available: https://www.instructables.com/DIY-3D-scanner-based-on-structured-light-and-stere/. [Accessed: Feb. 12, 2026].
66. F. Benetazzo, A. Freddi, A. Monteriù, and S. Longhi, "Respiratory rate detection algorithm based on RGB-D camera: theoretical background and experimental results," BMC Research Notes, vol. 8, no.1,p.252,2015.[Online].Available:https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4611185/ [Accessed: Feb. 12, 2026].
67. Texas Instruments, LIDAR-Pulsed Time-of-Flight Reference Design Using High-Speed Data Converters (TIDA-01187), Design Guide TIDUC73B, Rev. Aug. 2017.
68. ams-OSRAM AG, Time-of-Flight (ToF) Measurement Using Pulse Lasers, Application Note AN106, Oct. 22, 2024.
69. Texas Instruments, Introduction to Time-of-Flight Long-Range Proximity and Distance Sensor System Design, TI Designs: TIDA-01187, Mar. 2018, rev. Aug. 2019.
70. OSRAM Opto Semiconductors, Range Finding Using Pulse Lasers – Application Note, Sep. 10, 2004

71. C. Guang, “Understanding Time of Flight Systems: ToF, iToF, and dToF,” Lumimetric, [Online].Available:https://www.lumimetric.com/en/new/TOF-time-of-flight-definition-and-principle.html [Accessed: Feb. 21, 2026].
72. Seeed Studio, “MR24BSD1 Respiratory sleep detection datasheet.” [Online]. Available: https://files.seeedstudio.com/wiki/mmWave-radar/MR24BSD1_Datasheet.pdf
73. University of Nebraska-Lincoln, “Safe Operating Procedure.” Accessed: Mar. 27, 2026. [Online]. Available: https://ehs.unl.edu/sites/unl.edu.business-and-finance.university-operations.ehs/files/media/file/s-laser-safety_calculation_guide.pdf

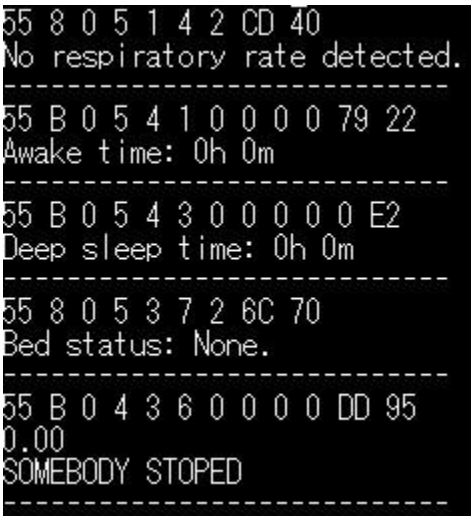
74. Processing of Configuration Change by System:



75. Noctisense Boot Sequence (Serial Console):



76. Noctisense Sleep Console 1



77. Noctisense Sleep Console 2

78. Noctisense Sleep Console 3

```
55 8 0 5 1 1 14 4F DE
Respiratory rate: 20 breaths/min
----------------------------
55 B 0 4 3 6 0 0 80 3F FC 45
1.00
SOMEBODY STOPED
----------------------------
55 8 0 5 1 1 14 4F DE
Respiratory rate: 20 breaths/min
----------------------------
55 B 0 4 3 6 0 0 80 3F FC 45
1.00
SOMEBODY STOPED
----------------------------
55 B 0 4 3 6 0 0 80 3F FC 45
1.00
SOMEBODY STOPED
----------------------------
55 B 0 4 3 6 0 0 80 3F FC 45
1.00
SOMEBODY STOPED
----------------------------
55 B 0 4 3 6 0 0 80 3F FC 45
1.00
SOMEBODY STOPED
----------------------------
55 8 0 5 1 1 16 CE 1F
Respiratory rate: 22 breaths/min
----------------------------
```

Appendix B: Software Source Code

“main.cpp” full integration of all components firmware/software running on system:

```
##include <Arduino.h>
#include <HardwareSerial.h>
#include "sleepbreathingradar.h"
#include "audioAnalyzer.h"
#include "ThermalCamera.h"
#include "InBedDetector.h"
#include <WiFi.h>
#include <WebServer.h>
#include <ArduinoJson.h>
#include <SD.h>
#include <SPI.h>
```

```
#include <time.h>
#include "esp_task_wdt.h"

// ==================== Prototypes ========================
void backgroundStorageTask(void * pvParameters);

// ==================== SYSTEM CONFIGURATION ====================
const char* WIFI_SSID     = "iPhone (56)";
const char* WIFI_PASSWORD = "Miles71198!!";

const char* DEVICE_ID    = "esp32-s3-n16-r8";
const char* DEVICE_NAME  = "Noctisense";
const char* FIRMWARE_VER = "1.0.0";

const int HTTP_PORT = 80;
const int TCP_PORT  = 3333;


// ==================== GLOBALS ====================
WebServer* httpServer = nullptr;
WiFiServer* tcpServer = nullptr;
bool sdCardPresent = false;
SleepBreathingRadar radar;
ThermalCamera thermal(9, 8);
char buff[30];
bool radarReady = false;
WiFiClient tcpClient;
unsigned long lastTcpSend = 0;

#define ESP_RX 3
#define ESP_TX 16

// --- FreeRTOS Task Handles ---
TaskHandle_t LoggingTaskHandle = NULL;

// --- Shared Thread-Safe Flags/Variables ---
volatile bool triggerAudioRecord = false;
volatile bool triggerSdLog = false;
String jsonLogBuffer = "";
portMUX_TYPE logMux = portMUX_INITIALIZER_UNLOCKED;

// --- Global Timers ---
unsigned long lastAudioSample = 0;
unsigned long lastStamp = 0;
unsigned long lastWifiCheck = 0;
unsigned long lastOptical = 0;

// ---- Active period config (set from Flutter app) ----
```

```
bool     activeEnabled       = true;
int      activeStartHour     = 22;
int      activeStartMinute   = 0;
int      activeEndHour       = 7;
int      activeEndMinute     = 0;
int      audioRecordInterval = 300;   // seconds between recordings
int      audioClipDuration   = 5000; // ms per clip
int      sensorPollInterval  = 2000;  // ms between sensor reads
float    restlessThreshold   = 0.5;
bool     sdCardLogging       = true;

// ---- In-bed state (can be set remotely via app trigger) ----
bool     inBedState          = false;

// ---- Audio recording state ----
unsigned long lastAudioRecordTime  = 0;
int           audioClipCounter     = 0;
String        lastAudioFilename    = "";
int           lastAudioDurationMs  = 0;
int           lastAudioSizeBytes   = 0;
bool          hasNewAudioClip      = false;
float         baselineDb           = 40.0f;
bool          baselineCaptured     = false;
volatile bool isRecording          = false;
volatile bool isStreamingAudio     = false;

// ---- Sensor poll timing ----
unsigned long lastSensorPoll = 0;

// ==== Build and send a complete frame with CRC ============================
static void sendFrame(const unsigned char* payload, size_t len) {
    size_t total = len + 2;
    unsigned char frame[total];          // fixed size — radar frames are never > 30 bytes
    memcpy(frame, payload, len);

    unsigned short crc = radar.us_CalculateCrc16((unsigned char*)payload, len);
    // Protocol: CRC low byte first, then high byte
    frame[len]     = (crc & 0xff00) >> 8;
    frame[len + 1] =  crc & 0x00ff;

    Serial2.print("Sending: ");
    for (size_t i = 0; i < total; i++) {
        char b[6]; sprintf(b, "0x%02x ", frame[i]);
        Serial2.print(b);
    }
    Serial2.println();
    Serial1.write(frame, total);
}
```

```
// ====================== SD CARD ====================
void initSDCard() {
    Serial2.println("Attempting SD init...");
    delay(250);

    // Manually assert CS high before SPI init to prevent SD card confusion
    pinMode(10, OUTPUT);
    digitalWrite(10, HIGH);
    delay(100);

    SPI.begin(12, 13, 11, 10); // CLK, MISO, MOSI, CS
    delay(100);

    // Only try ONCE — the SD library retry loop is what corrupts the heap
    if (SD.begin(10, SPI, 400000)) {    // ← drop to 400kHz, much safer for init
        sdCardPresent = true;
        Serial2.println("SD card OK");
        if (!SD.exists("/audio")) SD.mkdir("/audio");
        if (!SD.exists("/data"))  SD.mkdir("/data");
    } else {
        sdCardPresent = false;
        Serial2.println("SD not present — continuing without it");
        SPI.end();
    }
}

// ---- Log sensor data to SD as JSON lines ----
void logToSD(const String& jsonLine) {
  if (!sdCardPresent || !sdCardLogging) return;
  File f = SD.open("/data/sensor_log.jsonl", FILE_APPEND);
  if (f) {
    f.println(jsonLine);
    f.close();
  }
}

// ====================== AUDIO RECORDING ====================
// Audio encoding goal: use I2S DMA to record from the MEMS mic,
// encode (PCM 16-bit 16kHz), and write to SD card.

void recordAudioClip() {
    if (!sdCardPresent || isRecording) return;
    isRecording = true;

    audioClipCounter++;
    String filename = "/audio/clip_" + String(audioClipCounter) + ".mp3";
```

```
  File f = SD.open(filename, FILE_WRITE);
  if (!f) {
    Serial2.println("Failed to open file: " + filename);
    audioClipCounter--;
    return;
  }

  int recordDur = audioClipDuration;
  const int SAFE_MAX_RECORD_MS = 5000; // maximum supported clip length
  if (recordDur > SAFE_MAX_RECORD_MS) recordDur = SAFE_MAX_RECORD_MS;

  Serial2.println("Recording " + String(recordDur) + " ms → " + filename);

  // Register this task with the task watchdog while performing long I/O
  esp_err_t wdt_err = esp_task_wdt_add(NULL);
  if (wdt_err != ESP_OK) {
    Serial2.println("Warning: failed to add task to WDT");
  }

  bool ok = (audio_record_clip(f, recordDur) == AUDIO_OK);

  // Remove this task from the WDT (best-effort)
  esp_task_wdt_delete(NULL);
  uint32_t size = f.size();
  f.close();

  if (ok) {
    lastAudioFilename   = filename;
    lastAudioDurationMs = recordDur;
    lastAudioSizeBytes  = size;
    hasNewAudioClip     = true;
    Serial2.println("Saved: " + filename + " (" + String(size) + " bytes)");
  } else {
    Serial2.println("Recording completed.");
  }
  isRecording = false;
}


// ====================== ACTIVE PERIOD CHECK ====================
bool isInActivePeriod() {
 if (!activeEnabled) return true; // Always active if scheduling disabled

 struct tm timeinfo;
 if (!getLocalTime(&timeinfo)) return true; // If no time, stay active

 int nowMinutes   = timeinfo.tm_hour * 60 + timeinfo.tm_min;
```

```
  int startMinutes = activeStartHour * 60 + activeStartMinute;
  int endMinutes   = activeEndHour * 60 + activeEndMinute;

  if (startMinutes <= endMinutes) {
    // Same-day window (e.g., 08:00 - 18:00)
    return nowMinutes >= startMinutes && nowMinutes < endMinutes;
  } else {
    // Overnight window (e.g., 22:00 - 07:00)
    return nowMinutes >= startMinutes || nowMinutes < endMinutes;
  }
}

// ====================== TIMESTAMP ======================
String getTimestamp() {
  struct tm timeinfo;
  if (getLocalTime(&timeinfo)) {
    char buf[30];
    strftime(buf, sizeof(buf), "%Y-%m-%dT%H:%M:%SZ", &timeinfo);
    return String(buf);
  }
  // Fallback
  return "1970-01-01T00:00:" + String(millis() / 1000 % 60) + "Z";
}

//====================== SENSOR STUBS ======================
//Replace each with real hardware reads when ready.

float readAmbientTemperature() {
    return thermal.getAmbientTemp();
}

float readBodyTemperature() {
    return thermal.getHotSpot();
}

float readRadarHeartRate() {
    return radar.HR;        // populated by Sleep_inf()
}

float readRadarRespiratoryRate() {
    return radar.RR;         // populated by Sleep_inf()
}

float readRadarRestlessMovement() {
    return radar.RM;       // populated by Sleep_inf(); 0.0–~2.0
}

bool readRadarPresence() {
```

```
  return radar.inB;               // driven by app trigger + radar.inBed if available
}

// ====================== BUILD JSON ======================
String buildSensorJson() {
  JsonDocument doc;
  bool activeNow = isInActivePeriod();

  doc["device_id"]   = DEVICE_ID;
  doc["device_name"] = DEVICE_NAME;
  doc["firmware"]    = FIRMWARE_VER;
  doc["uptime_ms"]   = millis();
  doc["free_heap"]   = ESP.getFreeHeap();
  doc["sd_card"]     = sdCardPresent;
  doc["active_now"]  = activeNow;

  String ts = getTimestamp();

  // ---- Sensor readings ----
  JsonArray sensors = doc["sensors"].to<JsonArray>();

  // Ambient temperature
  JsonObject ambTemp = sensors.add<JsonObject>();
  ambTemp["sensor_id"]  = "ambient_temp";
  ambTemp["name"]       = "Ambient Temperature";
  ambTemp["unit"]       = "°F";
  ambTemp["value"]      = readAmbientTemperature();
  ambTemp["timestamp"]  = ts;

  // Body temperature
  JsonObject bodyTemp = sensors.add<JsonObject>();
  bodyTemp["sensor_id"]  = "body_temp";
  bodyTemp["name"]       = "Body Temperature";
  bodyTemp["unit"]       = "°F";
  bodyTemp["value"]      = readBodyTemperature();
  bodyTemp["timestamp"]  = ts;

  // ---- Radar/Optical sleep data ----
  JsonObject sleep = doc["sleep"].to<JsonObject>();
  sleep["respiratory_rate"]   = readRadarRespiratoryRate();
  sleep["heart_rate"]         = readRadarHeartRate();
  sleep["in_bed"]             = inBedState; // Added bool flag -> JSON for optical detection
  sleep["restless_movement"]  = readRadarRestlessMovement();
  sleep["timestamp"]          = ts;

  // ---- Audio clips (only include new ones since last poll) ----
   if (hasNewAudioClip && audioClipCounter > 0 && !isRecording) {
      JsonArray audioClips = doc["audio_clips"].to<JsonArray>();
```

```
      JsonObject clip = audioClips.add<JsonObject>();
      clip["clip_id"]      = "clip_" + String(audioClipCounter);
      clip["filename"]     = lastAudioFilename;
      clip["encoding"]     = "mp3";
      clip["duration_ms"]  = lastAudioDurationMs;
      clip["size_bytes"]   = lastAudioSizeBytes;
      clip["timestamp"]    = ts;
      hasNewAudioClip = false;
    }

  String output;
  serializeJson(doc, output);
  return output;
}

// ====================== HTTP HANDLERS ======================
void handleSensors() {
  String json = buildSensorJson();
  httpServer->send(200, "application/json", json);
}

void handleAudioDownload(){
    if(!sdCardPresent){
     httpServer->send(500, "application/json", "{\"error\":\"SD card not present\"}");
    return;
    }

    if(!httpServer -> hasArg("file")){
        httpServer->send(400, "application/json", "{\"error\":\"Missing 'file' parameter\"}");
        return;
    }

    String path = httpServer->arg("file");

    if(!SD.exists(path)){
        httpServer->send(404, "application/json", "{\"error\":\"File not found\"}");
        return;
    }

    // Open and read file
    File downloadFile = SD.open(path, FILE_READ);
    if(!downloadFile){
        httpServer->send(500, "application/json", "{\"error\":\"Failed to open file\"}");
        return;
    }

    isStreamingAudio = true;
```

```
    //Drop the TCP timeout to 1 second so it unfreezes when paused ---
    httpServer->client().setTimeout(1000);

    httpServer->setContentLength(downloadFile.size());
    httpServer->sendHeader("Content-Disposition", "inline; filename=\"" + path + "\"");
    httpServer->sendHeader("Connection", "close");
    httpServer->send(200, "audio/mpeg", "");

    WiFiClient client = httpServer->client();
    client.setTimeout(500);
    uint8_t buffer[1024];

    while (downloadFile.available() && client.connected()) {
        size_t bytesRead = downloadFile.read(buffer, sizeof(buffer));
        if (bytesRead > 0) {
            if (client.write(buffer, bytesRead) != bytesRead) {
                break;
            }
        }
        delay(1);
    }

    downloadFile.close();
    isStreamingAudio = false;
}

// Added to give filelist to flutter app
void handleListClips() {
  if (!SD.exists("/audio")) {
    httpServer->send(200, "application/json", "{\"clips\":[]}");
    return;
  }

  // 1. Create a DynamicJsonDocument.
  // Adjust the size (e.g., 2048) based on how many files you expect.
  JsonDocument doc;
  JsonArray clips = doc["clips"].to<JsonArray>();

  File root = SD.open("/audio");
  if (!root) {
    httpServer->send(500, "text/plain", "Failed to open directory");
    return;
  }

  File file = root.openNextFile();
  while (file) {
    if (!file.isDirectory()) {
      JsonObject obj = clips.add<JsonObject>();
```

```
      String fileName = String(file.name());

      // logic to remove .mp3 extension
      String clipId = fileName;
      if (clipId.endsWith(".mp3")) {
        clipId = clipId.substring(0, clipId.length() - 4);
      }

      obj["clip_id"] = clipId;
      obj["filename"] = "/audio/" + fileName;
      obj["size_bytes"] = file.size();
    }
    file.close(); // Crucial: prevents memory leaks/file handle exhaustion
    file = root.openNextFile();
  }
  root.close();

  // 2. Serialize to a String for the server
  String response;
  serializeJson(doc, response);
  httpServer->send(200, "application/json", response);
}


void handleCommand() {
  if (!httpServer->hasArg("plain")) {
    httpServer->send(400, "application/json", "{\"error\":\"no body\"}");
    return;
  }

  String body = httpServer->arg("plain");
  Serial2.println("Command received: " + body);

  JsonDocument cmdDoc;
  DeserializationError err = deserializeJson(cmdDoc, body);
  if (err) {
    httpServer->send(400, "application/json", "{\"error\":\"invalid json\"}");
    return;
  }

  String command = cmdDoc["command"] | "";

  if (command == "set_config") {
    // Active period
    activeStartHour     = cmdDoc["active_start_hour"]     | activeStartHour;
    activeStartMinute   = cmdDoc["active_start_minute"]   | activeStartMinute;
    activeEndHour       = cmdDoc["active_end_hour"]       | activeEndHour;
```

```
    activeEndMinute     = cmdDoc["active_end_minute"]     | activeEndMinute;
    activeEnabled       = cmdDoc["active_enabled"]        | activeEnabled;
    // Audio
    audioRecordInterval = cmdDoc["audio_record_interval_sec"] | audioRecordInterval;
    audioClipDuration   = cmdDoc["audio_clip_duration_ms"]    | audioClipDuration;
    // Sensor
    sensorPollInterval  = cmdDoc["sensor_poll_interval_ms"]   | sensorPollInterval;
    // Radar
    float rt = cmdDoc["restless_threshold"] | -1.0;
    if (rt > 0) restlessThreshold = rt;
    // SD
    sdCardLogging = cmdDoc["sd_card_logging"] | sdCardLogging;

    Serial2.println("Config updated: active=" + String(activeEnabled) +
                    " start=" + String(activeStartHour) + ":" + String(activeStartMinute) +
                    " end=" + String(activeEndHour) + ":" + String(activeEndMinute));

    httpServer->send(200, "application/json", "{\"status\":\"config_updated\"}");

  } else if (command == "set_in_bed") {
    inBedState = cmdDoc["in_bed"] | false;
    Serial2.println("In-bed state set to: " + String(inBedState));
    httpServer->send(200, "application/json", "{\"status\":\"in_bed_updated\",\"in_bed\":" + String(inBedState ? "true" :
"false") + "}");

  } else {
    httpServer->send(200, "application/json", "{\"status\":\"unknown_command\"}");
  }
}

void handleNotFound() {
  httpServer->send(404, "application/json", "{\"error\":\"not found\"}");
}

// ========== RADAR INIT ==============================
void initRadar() {
    // Initialize radar
    Serial1.begin(9600, SERIAL_8N1, 17, 18);
    delay(500);
    if(!Serial1) {
      Serial2.println("Radar Serial connection failed");
      return;
    }
    Serial2.println("Radar Serial2 port opened.");
    delay(1500);

    // ------------------------------------------------------------
    // Step 1: Enable the sleep detection switch (off by default per DP8)
```

```
    // Write command (0x02), addr1=0x05 (Other functions), addr2=0x0D (sleep switch), data=0x01 (On)
    // Frame: 0x55  0x07  0x00  0x02  0x05  0x0D  0x01  crc_L  crc_H
    // ------------------------------------------------------------------
    {
        unsigned char enableSleep[] = {0x55, 0x07, 0x00, 0x02, 0x05, 0x0D, 0x01};
        sendFrame(enableSleep, sizeof(enableSleep));
        Serial2.println("Sleep switch ON sent.");
        delay(200);
    }

    // ------------------------------------------------------------------
    // Step 2: (Optional) Set sensitivity — default is 7, range 1-10
    // Write command (0x02), addr1=0x04 (System params), addr2=0x0C (threshold gear), data=0x07 UPDATED TO 9
TO CHECK WITH HOUSING
    // ------------------------------------------------------------------
    {
        unsigned char setSensitivity[] = {0x55, 0x07, 0x00, 0x02, 0x04, 0x0C, 0x09};
        sendFrame(setSensitivity, sizeof(setSensitivity));
        Serial2.println("Sensitivity set to 9.");
        delay(200);
    }

    // ------------------------------------------------------------------
    // Step 3: (Optional) Query current sleep switch state to confirm
    // Read command (0x01), addr1=0x05, addr2=0x0D
    // ------------------------------------------------------------------
    {
        unsigned char querySleep[] = {0x55, 0x05, 0x00, 0x01, 0x05, 0x0D};
        sendFrame(querySleep, sizeof(querySleep));
        Serial2.println("Querying sleep switch state...");
        delay(500);
    }

    Serial2.println("Radar Init Complete");
    delay(100);
    while (Serial1.available()) Serial1.read(); // flush startup frames
    radarReady = true;
}

void setup() {
    Serial2.begin(115200, SERIAL_8N1, ESP_RX, ESP_TX);
    delay(2000);
    Serial2.println("ESP32-S3 System Initializing...");

    httpServer = new WebServer(HTTP_PORT);
    tcpServer  = new WiFiServer(TCP_PORT);

    initSDCard();
```

```
initRadar();
thermal.begin();
audio_analyzer_init();
optical_system_init();

// Non-blocking WiFi connection on boot (10s timeout)
WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
Serial2.print("Connecting to WiFi");
unsigned long wifiStart = millis();
while (WiFi.status() != WL_CONNECTED && (millis() - wifiStart < 10000)) {
    delay(500);
    Serial2.print(".");
}

if (WiFi.status() == WL_CONNECTED) {
    Serial2.println("\nConnected! IP: ");
    Serial2.println(WiFi.localIP());
} else {
    Serial2.println("\nWiFi Timeout. Continuing boot, will retry in background.");
}

configTime(-18000, 3600, "pool.ntp.org", "time.nist.gov");

httpServer->on("/sensors", HTTP_GET, handleSensors);
httpServer->on("/command", HTTP_POST, handleCommand);
httpServer->on("/download",HTTP_GET,handleAudioDownload);
httpServer->onNotFound(handleNotFound);
httpServer->begin();
tcpServer->begin();

// Pre-reserve JSON buffer to reduce heap fragmentation from repeated allocations
jsonLogBuffer.reserve(2048);

// Added to send file list to flutter app to catch up on past recordings on SD.
httpServer->on("/audio/clips", HTTP_GET, handleListClips);

if (!sdCardPresent) {
    Serial2.println("SD card not found");
}

Serial2.println("Measuring baseline...");
baselineDb = audio_get_baseline_db();
Serial2.printf("Baseline: %.1f dB\n", baselineDb);
baselineCaptured = true;

// CREATE BACKGROUND TASK ON CORE 0 FOR HEAVY SD CARD / AUDIO I/O
xTaskCreatePinnedToCore(
    backgroundStorageTask,    // Function to implement the task
```

```
      "StorageTask",           // Name of the task
      8192,                    // Stack size in words (Plenty of room on S3)
      NULL,                    // Task input parameter
      1,                       // Priority of the task
      &LoggingTaskHandle,      // Task handle
      0                        // Core 0 (Wi-Fi core)
    );

    // --- MOCK DATA FOR APP TESTING ---
  if (sdCardPresent && SD.exists("/audio/test.mp3")) {
      File testFile = SD.open("/audio/test.mp3", FILE_READ);
      if (testFile) {
          lastAudioFilename = "/audio/test.mp3";
          lastAudioSizeBytes = testFile.size();
          lastAudioDurationMs = 5000;
          audioClipCounter = 1;
          testFile.close();
          Serial2.println("Found test.mp3 — Mocking data for Flutter app");
      }
  }
}

// --- CORE 1: High-Speed / Real-Time Loop ---
void loop() {
    unsigned long currentMillis = millis();
    bool activeNow = isInActivePeriod();

    //Handle network clients instantly
    httpServer->handleClient();

    // Lock: Skips sensor polling during audio streaming
    if (isStreamingAudio) {
        if (radarReady) {
            radar.recvRadarBytes();
            if (radar.newData) radar.newData = false;
        }
        vTaskDelay(pdMS_TO_TICKS(2));
        return;
    }

    //Poll Radar Serial (Must happen constantly to prevent buffer overflow)
    if (radarReady) {
        radar.recvRadarBytes();
        if (radar.newData) {
            byte dataMsg[14];
            if (radar.dataLen <= 12) {
                dataMsg[0] = 0x55;
                for (byte n = 0; n < radar.dataLen; n++) {
```

```
                dataMsg[n + 1] = radar.Msg[n];
            }
            radar.Sleep_inf(dataMsg);
            radar.Bodysign_judgment(dataMsg, 2.0, 5.0);
        }
        radar.newData = false;
    }
}

// 3. Thermal Update
thermal.update();

// 4. Optical Presence Detection (Every 5 seconds)
if (currentMillis - lastOptical >= 5000) {
    lastOptical = currentMillis; // <-- FIXED: Resets the timer so it doesn't loop-spam
    inBedState = optical_read_presence();
}

// 5. Audio Sampling (Every 500ms)
if (baselineCaptured && (currentMillis - lastAudioSample >= 500)) {
    lastAudioSample = currentMillis;
    AudioReading rdata = audio_measure_db();

    // If triggered, signal Core 0 to handle the heavy recording
    if (rdata.db_spl > (baselineDb + DB_TRIGGER_OFFSET) && !triggerAudioRecord) {
        triggerAudioRecord = true;
    }
}

// 6. Sensor Polling & Preparing Data for SD
if (activeNow && (currentMillis - lastSensorPoll >= (unsigned long)sensorPollInterval)) {
    lastSensorPoll = currentMillis;
    String json = buildSensorJson();

    // Push string safely to background task if it isn't already busy processing one
    if (!triggerSdLog) {
        portENTER_CRITICAL(&logMux);
        jsonLogBuffer = json;
        triggerSdLog = true;
        portEXIT_CRITICAL(&logMux);
    }
}

// 7. TCP Server Server Handshaking & Data Sending
if (!tcpClient || !tcpClient.connected()) {
    if (tcpClient) tcpClient.stop();
    WiFiClient newClient = tcpServer->available();
    if (newClient && newClient.connected()) {
```

```
            tcpClient = newClient;
            lastTcpSend = currentMillis;
        }
    }

    if (tcpClient && tcpClient.connected()) {
        // ---- Send sensor JSON periodically ----
        if (currentMillis - lastTcpSend >= (unsigned long)sensorPollInterval) {
            lastTcpSend = currentMillis;
            tcpClient.println(buildSensorJson());
        }
    }

    // 8. Non-blocking WiFi watchdog
    if (WiFi.status() != WL_CONNECTED && (currentMillis - lastWifiCheck >= 10000)) {
        lastWifiCheck = currentMillis;
        WiFi.reconnect();
    }

    // 9. Printable Timestamp (Every 1 min)
    if (currentMillis - lastStamp >= 60000) {
        lastStamp = currentMillis;
        time_t timeNow = time(nullptr);
        struct tm* timeinfo = localtime(&timeNow);
        char buffer[40];
        strftime(buffer, sizeof(buffer), "%Y-%m-%d %H:%M:%S %Z", timeinfo);
        Serial2.println(buffer);
    }

    // Let Core 1 breathe
    vTaskDelay(pdMS_TO_TICKS(2));
}

// --- CORE 0: Background Storage Task (Handles slow SD card writes) ---
void backgroundStorageTask(void * pvParameters) {
    while(true) {

        if(isStreamingAudio){
            vTaskDelay(pdMS_TO_TICKS(50));
            continue;
        }
        // Handle heavy Audio Recording
        if (triggerAudioRecord) {
            // This blocks Core 0, but Core 1 keeps processing radar data perfectly!
            recordAudioClip();
            triggerAudioRecord = false;
        }
```

```
    // Handle slow SD Card Logging
    if (triggerSdLog) {
      portENTER_CRITICAL(&logMux);
      String txtToLog = jsonLogBuffer;
      portEXIT_CRITICAL(&logMux);

      logToSD(txtToLog);

      triggerSdLog = false;
    }

    // Mandatory small delay to let FreeRTOS manage Core 0 Wi-Fi background tasks
    // Reset the task watchdog timer in this background loop
    esp_task_wdt_reset();
    vTaskDelay(pdMS_TO_TICKS(10));
  }
}
```

“sleepbreathingradar.cpp” customized library for MR24BSD1 radar module:

```
#include "Arduino.h"
#include "sleepbreathingradar.h"

#ifdef __AVR__
    #include <SoftwareSerial.h>
    SoftwareSerial SSerial(2, 3); // RX, TX
    #define Serial1 SSerial
#endif

void SleepBreathingRadar::SerialInit(){
  Serial1.begin(9600);
}

// Receive data and process
// Modded version
void SleepBreathingRadar::recvRadarBytes(){
    static boolean recvInProgress = false;
    static byte ndx = 0;
    byte startMarker = MESSAGE_HEAD;
    byte rb;

    while (Serial1.available() > 0 && newData == false) {
        rb = Serial1.read();
        if (recvInProgress == true) {
            if (dataLen > ndx) {
                Msg[ndx] = rb;
```

```
                if (ndx == 0) {
                    // Clamp to buffer size, reject obviously corrupt length
bytes
                    if (rb < 4 || rb > MsgLen) {
                        recvInProgress = false;
                        ndx = 0;
                        continue;
                    }
                    dataLen = rb;
                }
                ndx++;
            } else {
                recvInProgress = false;
                ndx = 0;
                newData = true;
            }
        } else if (rb == startMarker) {
            recvInProgress = true;
            dataLen = MsgLen;  // reset to max until length byte arrives
            ndx = 0;
        }
    }
}


// Unpacking of physical parameters
// Modded Version
void SleepBreathingRadar::Bodysign_judgment(byte inf[], float Move_min, float
Move_max){
    typedef union {
        unsigned char Byte[4];
        float Float;
    } Float_Byte;

    if (inf[3] == ACTIVE_REPORT) {
        if (inf[5] == BODYSIGN) {
            Float_Byte fb;
            // memcpy avoids LoadStoreAlignment crash on ESP32
            memcpy(fb.Byte, &inf[6], 4);
            ShowData(inf);
            Serial.println(fb.Float);
            if (fb.Float >= Move_min && fb.Float < Move_max) {
                Serial.println("SOMEBODY STOPED");
            } else if (fb.Float < Move_min) {
                Serial.println("NO PRESENCE DETECTED");
```

```
            } else if (fb.Float >= Move_max) {
                Serial.println("SOMEBODY MOVED");
                RM = fb.Float;
            }
            Serial.println("----------------------------");
        }
    }
}

// Judgment of occupied and unoccupied, approach and distance
void SleepBreathingRadar::Situation_judgment(byte inf[]){
  switch(inf[3]){
    case ACTIVE_REPORT:
      switch(inf[4]){
        case REPORT_RADAR:
          switch(inf[5]){
            case ENVIRONMENT:
              ShowData(inf);
              switch(inf[6]){
                case NOBODY:
                  Serial.println("Radar detects no one.");
                  Serial.println("----------------------------");
                  break;
                case SOMEBODY_BE:
                  switch(inf[7]){
                    case SOMEBODY_MOVE:
                      Serial.println("Radar detects somebody is moving.");
                      Serial.println("----------------------------");
                      break;
                    case SOMEBODY_STOP:
                      Serial.println("Radar detects somebody is stopping.");
                      Serial.println("----------------------------");
                      break;
                  }
                  break;
              }
              break;
            case HEARTBEAT:
              ShowData(inf);
              switch(inf[6]){
                case NOBODY:
                  Serial.println("Radar detects no one.");
                  Serial.println("----------------------------");
                  break;
                case SOMEBODY_BE:
```

```
                switch(inf[7]){
                  case SOMEBODY_MOVE:
                    Serial.println("Radar detects somebody is moving.");
                    Serial.println("----------------------------");
                    break;
                  case SOMEBODY_STOP:
                    Serial.println("Radar detects somebody is stopping.");
                    Serial.println("----------------------------");
                    break;
                }
                break;
            }
            break;
          case CLOSE_AWAY:
            ShowData(inf);
            switch(inf[6]){
              case CA_BE:
                switch(inf[7]){
                  case CA_BE:
                    switch(inf[8]){
                      case CA_BE:
                        Serial.println("Radar detects no move.");
                        Serial.println("----------------------------");
                        break;
                      case CA_CLOSE:
                        Serial.println("Radar detects somebody moving
closer.");
                        Serial.println("----------------------------");
                        break;
                      case CA_AWAY:
                        Serial.println("Radar detects somebody moving away.");
                        Serial.println("----------------------------");
                        break;
                    }
                    break;
                }
                break;
            }
            break;
          case ABNOEMAL:
            ShowData(inf);
            Serial.println("Exception thrown for abnormal packet type.");
            Serial.println("----------------------------");
            break;
        }
```

```
        break;
      case REPORT_OTHER:
        switch(inf[5]){
          case ENVIRONMENT || HEARTBEAT:
            ShowData(inf);
            switch(inf[6]){
              case NOBODY:
                Serial.println("Radar detects no one.");
                Serial.println("----------------------------");
                break;
              case SOMEBODY_BE:
                switch(inf[7]){
                  case SOMEBODY_MOVE:
                    Serial.println("Radar detects somebody is moving.");
                    Serial.println("----------------------------");

                    break;
                  case SOMEBODY_STOP:
                    Serial.println("Radar detects somebody is stopping.");
                    Serial.println("----------------------------");
                    break;
                }
                break;
            }
            break;
          case CLOSE_AWAY:
            ShowData(inf);
            switch(inf[6]){
              case CA_BE:
                switch(inf[7]){
                  case CA_BE:
                    switch(inf[8]){
                      case CA_BE:
                        Serial.println("Radar detects no movement.");
                        Serial.println("----------------------------");
                        break;
                      case CA_CLOSE:
                        Serial.println("Radar detects a presence closer.");
                        Serial.println("----------------------------");
                        break;
                      case CA_AWAY:
                        Serial.println("Radar detects presence further away.");
                        Serial.println("----------------------------");
                        break;
                    }
```

```
                    break;
                }
                break;
            }
            break;
          case ABNOEMAL:
            ShowData(inf);
            Serial.println("Exception thrown for abnormal packet type.");
            Serial.println("----------------------------");
            break;
        }
        break;
    }
    break;
  }
}

//Respiratory sleep data frame decoding
// Modded version
void SleepBreathingRadar::Sleep_inf(byte inf[]){
    // Only process frames flagged as sleep radar data
    if (inf[3] != SLEEP_INF) return;

    ShowData(inf);

    switch(inf[4]){
        case BREATH:
            switch(inf[5]){
                case BREATH_RATE:
                    Serial.print("Respiratory rate: ");
                    Serial.print(inf[6]);
                    Serial.println(" breaths/min");
                    RR = inf[6];
                    break;
                case CHECK_SIGN:
                    switch(inf[6]){
                        case BREATH_HOLD:
                            Serial.println("Abnormal: breath-holding detected.");
                            break;
                        case BREATH_NULL:
                            Serial.println("No respiratory rate detected.");
                            break;
                        case BREATH_NORMAL:
                            Serial.println("Breathing signal: Nominal.");
                            break;
```

```
                        case BREATH_MOVE:
                            Serial.println("Abnormal: movement interference
detected.");
                            break;
                        case BREATH_RAPID:
                            Serial.println("Abnormal: shortness of breath
detected.");
                            break;
                        default:
                            Serial.print("Unknown breath signal: 0x");
                            Serial.println(inf[6], HEX);
                            break;
                    }
                    break;
                default:
                    Serial.print("Unknown BREATH sub-addr: 0x");
                    Serial.println(inf[5], HEX);
                    break;
            }
            break;

        case SCENARIO:
            switch(inf[5]){
                case CLOSE_AWAY_BED:        // 0x07 — bed entry/exit
                    switch(inf[6]){
                        case AWAY_BED:
                            Serial.println("Bed status: Left the bed.");
                            inB = 0.0f;
                            break;
                        case CLOSE_BED:
                            Serial.println("Bed status: In bed.");
                            inB = 1.0f;
                            break;
                        default:
                            Serial.println("Bed status: None.");
                            break;
                    }
                    break;

                case SLEEP_STATE:           // 0x08 — was completely missing
                    switch(inf[6]){
                        case AWAKE:
                            Serial.println("Sleep state: Awake.");
                            break;
                        case LIGHT_SLEEP:
```

```
                            Serial.println("Sleep state: Light sleep.");
                            break;
                        case DEEP_SLEEP:
                            Serial.println("Sleep state: Deep sleep.");
                            break;
                        case SLEEP_NULL:
                            Serial.println("Sleep state: None (switch off or no
data).");
                            break;
                        default:
                            Serial.print("Unknown sleep state: 0x");
                            Serial.println(inf[6], HEX);
                            break;
                    }
                    break;

                default:
                    Serial.print("Unknown SCENARIO sub-addr: 0x");
                    Serial.println(inf[5], HEX);
                    break;
            }
            break;

        case SLEEP_TIME:
            switch(inf[5]){
                case AWAKE_TIME:
                    Serial.print("Awake time: ");
                    break;
                case LIGHT_SLEEP_TIME:
                    Serial.print("Light sleep time: ");
                    break;
                case DEEP_SLEEP_TIME:
                    Serial.print("Deep sleep time: ");
                    break;
                default:
                    Serial.print("Unknown time type 0x");
                    Serial.print(inf[5], HEX);
                    Serial.print(": ");
                    break;
            }
            SleepTimeCalculate(inf[6], inf[7], inf[8], inf[9]);
            break;

        case SLEEP_QUALITY:
            switch(inf[5]){
```

```
                case SLEEP_SCORE:
                    Serial.print("Sleep quality score: ");
                    Serial.println(inf[6]);
                    break;
                default:
                    Serial.print("Unknown quality sub-addr: 0x");
                    Serial.println(inf[5], HEX);
                    break;
            }
            break;

        case 0x06:  // Heart rate parameter (added to library, packet was being
sent from radar firmware.)
            switch(inf[5]){
                case 0x01:
                    Serial.print("Heart rate: ");
                    Serial.print(inf[6]);
                    Serial.println(" bpm");
                    HR = inf[6];
                    break;
                default:
                    Serial.print("Unknown heart rate sub-addr: 0x");
                    Serial.println(inf[5], HEX);
                    break;
            }
            break;
        default:
            Serial.print("Unknown sleep frame addr1: 0x");
            Serial.println(inf[4], HEX);
            break;
    }
    Serial.println("----------------------------");
}

//Sleep time decoding
void SleepBreathingRadar::SleepTimeCalculate(unsigned char inf1, unsigned char
inf2,
                                             unsigned char inf3, unsigned char
inf4){
    unsigned long minutes = ((unsigned long)inf1 << 24)
                          | ((unsigned long)inf2 << 16)
                          | ((unsigned long)inf3 << 8)
                          |  (unsigned long)inf4;
    unsigned long hours = minutes / 60;
    unsigned long mins  = minutes % 60;
```

```
    Serial.print(hours);
    Serial.print("h ");
    Serial.print(mins);
    Serial.println("m");
}

//Radar transmits data frames for display via serial port
void SleepBreathingRadar::ShowData(byte inf[]){
  for (byte n = 0; n < dataLen+1; n++) {
    Serial.print(inf[n], HEX);
    Serial.print(' ');
  }
    Serial.println();
}

const unsigned char cuc_CRCHi[256]= {
  0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41,
  0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
  0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41,
  0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
  0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41,
  0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
  0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
  0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
  0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41,
  0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
  0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41,
  0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
  0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41,
  0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
  0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
  0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
  0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41,
  0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
  0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41,
  0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
  0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41,
  0x00, 0xC1, 0x81, 0x40
};

const unsigned char  cuc_CRCLo[256]= {
  0x00, 0xC0, 0xC1, 0x01, 0xC3, 0x03, 0x02, 0xC2, 0xC6, 0x06, 0x07, 0xC7,
  0x05, 0xC5, 0xC4, 0x04, 0xCC, 0x0C, 0x0D, 0xCD, 0x0F, 0xCF, 0xCE, 0x0E,
  0x0A, 0xCA, 0xCB, 0x0B, 0xC9, 0x09, 0x08, 0xC8, 0xD8, 0x18, 0x19, 0xD9,
  0x1B, 0xDB, 0xDA, 0x1A, 0x1E, 0xDE, 0xDF, 0x1F, 0xDD, 0x1D, 0x1C, 0xDC,
```

```
  0x14, 0xD4, 0xD5, 0x15, 0xD7, 0x17, 0x16, 0xD6, 0xD2, 0x12, 0x13, 0xD3,
  0x11, 0xD1, 0xD0, 0x10, 0xF0, 0x30, 0x31, 0xF1, 0x33, 0xF3, 0xF2, 0x32,
  0x36, 0xF6, 0xF7, 0x37, 0xF5, 0x35, 0x34, 0xF4, 0x3C, 0xFC, 0xFD, 0x3D,
  0xFF, 0x3F, 0x3E, 0xFE, 0xFA, 0x3A, 0x3B, 0xFB, 0x39, 0xF9, 0xF8, 0x38,
  0x28, 0xE8, 0xE9, 0x29, 0xEB, 0x2B, 0x2A, 0xEA, 0xEE, 0x2E, 0x2F, 0xEF,
  0x2D, 0xED, 0xEC, 0x2C, 0xE4, 0x24, 0x25, 0xE5, 0x27, 0xE7, 0xE6, 0x26,
  0x22, 0xE2, 0xE3, 0x23, 0xE1, 0x21, 0x20, 0xE0, 0xA0, 0x60, 0x61, 0xA1,
  0x63, 0xA3, 0xA2, 0x62, 0x66, 0xA6, 0xA7, 0x67, 0xA5, 0x65, 0x64, 0xA4,
  0x6C, 0xAC, 0xAD, 0x6D, 0xAF, 0x6F, 0x6E, 0xAE, 0xAA, 0x6A, 0x6B, 0xAB,
  0x69, 0xA9, 0xA8, 0x68, 0x78, 0xB8, 0xB9, 0x79, 0xBB, 0x7B, 0x7A, 0xBA,
  0xBE, 0x7E, 0x7F, 0xBF, 0x7D, 0xBD, 0xBC, 0x7C, 0xB4, 0x74, 0x75, 0xB5,
  0x77, 0xB7, 0xB6, 0x76, 0x72, 0xB2, 0xB3, 0x73, 0xB1, 0x71, 0x70, 0xB0,
  0x50, 0x90, 0x91, 0x51, 0x93, 0x53, 0x52, 0x92, 0x96, 0x56, 0x57, 0x97,
  0x55, 0x95, 0x94, 0x54, 0x9C, 0x5C, 0x5D, 0x9D, 0x5F, 0x9F, 0x9E, 0x5E,
  0x5A, 0x9A, 0x9B, 0x5B, 0x99, 0x59, 0x58, 0x98, 0x88, 0x48, 0x49, 0x89,
  0x4B, 0x8B, 0x8A, 0x4A, 0x4E, 0x8E, 0x8F, 0x4F, 0x8D, 0x4D, 0x4C, 0x8C,
  0x44, 0x84, 0x85, 0x45, 0x87, 0x47, 0x46, 0x86, 0x82, 0x42, 0x43, 0x83,
  0x41, 0x81, 0x80, 0x40
};

unsigned short int SleepBreathingRadar::us_CalculateCrc16(unsigned char
*lpuc_Frame, unsigned short int lus_Len){
  unsigned char luc_CRCHi = 0xFF;
  unsigned char luc_CRCLo = 0xFF;
  int li_Index=0;
  while(lus_Len--){
    li_Index = luc_CRCLo ^ *( lpuc_Frame++);
    luc_CRCLo = (unsigned char)( luc_CRCHi ^ cuc_CRCHi[li_Index]);
    luc_CRCHi = cuc_CRCLo[li_Index];
  }
  return (unsigned short int )(luc_CRCLo << 8 | luc_CRCHi);
}
```

“audioAnalyzer.cpp” custom library for utilizing the microphone to analyze/record audio clips:

```
#include "audioanalyzer.h"
#include <math.h>
#include "AudioTools.h"
#include "AudioTools/AudioCodecs/CodecMP3LAME.h"
#include "esp_task_wdt.h"

// Removes a narrow band around NOTCH_FREQ_HZ from recorded clips.
#ifndef NOTCH_FREQ_HZ
#define NOTCH_FREQ_HZ 1090.0f
#endif
```

```c
#ifndef NOTCH_Q
#define NOTCH_Q 10.0f   // higher Q = narrower notch
#endif

typedef struct {
    float b0, b1, b2, a1, a2;   // normalized transfer function coefficients
    float x1, x2, y1, y2;       // filter state (previous in/out samples)
} IirNotchFilter;

static IirNotchFilter g_notch;

static void notch_filter_init(IirNotchFilter *nf, float freq_hz, float q, float sample_rate_hz) {
    float w0    = 2.0f * (float)M_PI * freq_hz / sample_rate_hz;
    float alpha = sinf(w0) / (2.0f * q);
    float cosw0 = cosf(w0);

    float b0 = 1.0f;
    float b1 = -2.0f * cosw0;
    float b2 = 1.0f;
    float a0 = 1.0f + alpha;
    float a1 = -2.0f * cosw0;
    float a2 = 1.0f - alpha;

    nf->b0 = b0 / a0;
    nf->b1 = b1 / a0;
    nf->b2 = b2 / a0;
    nf->a1 = a1 / a0;
    nf->a2 = a2 / a0;
    nf->x1 = nf->x2 = nf->y1 = nf->y2 = 0.0f;
}

static inline void notch_filter_reset(IirNotchFilter *nf) {
    nf->x1 = nf->x2 = nf->y1 = nf->y2 = 0.0f;
}

static inline int16_t notch_filter_process(IirNotchFilter *nf, int16_t in_sample) {
    float x0 = (float)in_sample;
    float y0 = nf->b0 * x0 + nf->b1 * nf->x1 + nf->b2 * nf->x2
                           - nf->a1 * nf->y1 - nf->a2 * nf->y2;
    nf->x2 = nf->x1;
    nf->x1 = x0;
    nf->y2 = nf->y1;
    nf->y1 = y0;

    if (y0 >  32767.0f) y0 =  32767.0f;
    if (y0 < -32768.0f) y0 = -32768.0f;
    return (int16_t)y0;
}
```

```
static inline void notch_filter_apply_buffer(IirNotchFilter *nf, int16_t *buf, uint32_t n) {
    for (uint32_t i = 0; i < n; i++) {
        buf[i] = notch_filter_process(nf, buf[i]);
    }
}


// ── Internal helpers ──────────────────────────────────────────────────────
static float calculate_db_from_rms(float rms_adc) {
    if (rms_adc < MIN_VALID_RMS_ADC) return 0.0f;
    float db = 20.0f * log10f(rms_adc / REF_RMS_ADC) + REF_DB_SPL;
    if (db < 0.0f) db = 0.0f;
    if (db > 120.0f) db = 120.0f;
    return db;
}

// Spin-wait until `next_us` arrives, then advance it by `interval_us`.
// Returns the signed ADC sample (centred on 0, scaled to 16-bit).
static inline int16_t _sample_and_advance(uint32_t &next_us,
                                          uint32_t  interval_us) {
    // Wait until the next sample slot; yield so the scheduler can run
    while ((long)(micros() - next_us) < 0) {
        yield();
        esp_task_wdt_reset();
    }
    next_us += interval_us;
    int raw = analogRead(AMP_PIN);
    // Map 12-bit unsigned (0–4095) → signed 16-bit PCM.
    // Subtract mid-point first, then scale to fill ~16-bit range.
    return (int16_t)((raw - 2048) * 16);
}

// ── MP3 recording ─────────────────────────────────────────────────────────
static AudioResult _record_mp3(File &f, uint32_t duration_ms) {
    // ── Set up encoder chain: PCM write()s → LAME → File ─────
    MP3EncoderLAME      lame;
    EncodedAudioStream  encoder(&f, &lame);

    // Provide format metadata so LAME knows what PCM it is receiving.
    AudioInfo info;
    info.sample_rate    = SAMPLE_RATE_HZ;
    info.channels       = 1;
    info.bits_per_sample = 16;

    if (!encoder.begin(info)) {
        return AUDIO_ERR_ENCODER;
    }
```

```
    // ── Sample loop ──────────────────
    const uint32_t total_samples = ((uint32_t)SAMPLE_RATE_HZ * duration_ms) / 1000UL;
    const uint32_t interval_us   = 1000000UL / SAMPLE_RATE_HZ;

    int16_t  buf[RECORD_BUF_SAMPLES];
    uint32_t collected = 0;
    uint32_t next_us   = micros();

    notch_filter_reset(&g_notch);   // ADDED: avoid transient bleed from a previous clip

    while (collected < total_samples) {
        uint32_t chunk = min((uint32_t)RECORD_BUF_SAMPLES,
                             total_samples - collected);
        for (uint32_t i = 0; i < chunk; i++) {
            buf[i] = _sample_and_advance(next_us, interval_us);
        }
        // Feed PCM into the encoder; LAME buffers internally and emits
        // MP3 frames to the file as they fill.
        notch_filter_apply_buffer(&g_notch, buf, chunk);
        encoder.write((uint8_t *)buf, chunk * sizeof(int16_t));
        // Feed the task watchdog periodically
        esp_task_wdt_reset();
        collected += chunk;
    }

    // Flush any partially-filled LAME frame — essential for a valid MP3 file.
    encoder.end();
    return AUDIO_OK;
}

// ═══════════════════════════════════════════════════════════════════════
//  Public API
// ═══════════════════════════════════════════════════════════════════════

void audio_analyzer_init(void) {
    delay(500);   // let PSRAM + boot transient settle
    pinMode(AMP_PIN, INPUT);
    analogSetPinAttenuation(AMP_PIN, ADC_11db);
    notch_filter_init(&g_notch, NOTCH_FREQ_HZ, NOTCH_Q, (float)SAMPLE_RATE_HZ);   // ADDED
}

AudioReading audio_measure_db(void) {
    double   sum_sq  = 0.0;
    uint32_t samples = 0;
    uint32_t start   = millis();
    uint32_t min_raw = 4095;
    uint32_t max_raw = 0;
```

```
    while ((millis() - start) < SAMPLE_WINDOW_MS) {
        uint32_t raw = analogRead(AMP_PIN);
        if (raw < min_raw) min_raw = raw;
        if (raw > max_raw) max_raw = raw;

        float centred = (float)raw - ADC_MID;
        sum_sq += centred * centred;
        samples++;

        // yield occasionally so other tasks (WiFi, background) can run
        if ((samples & 0x3F) == 0) { yield(); }
    }

    AudioReading result;
    result.rms_adc = (samples > 0) ? sqrtf((float)(sum_sq / samples)) : 0.0f;
    result.db_spl  = calculate_db_from_rms(result.rms_adc);
    result.peak_to_peak = (uint16_t)(max_raw - min_raw);
    return result;
}

float audio_get_baseline_db(void) {
    float sum = 0.0f;
    int count = 0;

    for (int i = 0; i < 50; i++) {
        AudioReading reading = audio_measure_db();
        if (reading.rms_adc < MIN_VALID_RMS_ADC) {
            continue;
        }
        if (reading.peak_to_peak > MAX_BASELINE_P2P) {
            continue;
        }
        sum += reading.db_spl;
        count++;
    }

    return (count > 0) ? (sum / count) : 0.0f;
}

AudioResult audio_record_clip(File &f,
                              uint32_t duration_ms) {
    if (!f)             return AUDIO_ERR_FILE;
    if (duration_ms == 0) return AUDIO_ERR_PARAMS;
    if (SAMPLE_RATE_HZ == 0) return AUDIO_ERR_PARAMS;   // guard against div-by-zero

    return _record_mp3(f, duration_ms);
}
```

“ThermalCamera.cpp” custom library for reading body/ambient temperatures:

```
#include "ThermalCamera.h"

// Constructor: sets up the pins
ThermalCamera::ThermalCamera(int sdaPin, int sclPin) {
  _sdaPin = sdaPin; // Using 9
  _sclPin = sclPin; // Using 8
}

// Initialization function
bool ThermalCamera::begin() {
  // If custom pins are provided, use them
  if (_sdaPin != -1 && _sclPin != -1) {
    Wire.begin(_sdaPin, _sclPin);
  } else {
    Wire.begin();
  }

  if (!mlx.begin(MLX90640_I2CADDR_DEFAULT, &Wire)) {
    return false; // Return false if camera isn't found
  }

  mlx.setMode(MLX90640_CHESS);
  mlx.setResolution(MLX90640_ADC_18BIT);
  mlx.setRefreshRate(MLX90640_1_HZ);
  return true;
}

// Fetch a new frame
bool ThermalCamera::update() {
  if (mlx.getFrame(frame) != 0) {
    return false;
  }
  return true;
}

float ThermalCamera::convertToFahrenheit(float currValue) {
  return ((1.8 * currValue) + 32);
}

float ThermalCamera::getAmbientTemp() {
  int n = 768;
```

```
  std::sort(frame, frame + n);

  float min = frame[0];
  float max = frame[767];

  for(int i = 0; i < 10; i++) {
    float mean = (min + max) / 2.0f;
    int coolCount = 0;
    float coolSum = 0;
    int hotCount = 0;
    float hotSum = 0;

    for(int j = 0; j < 768; j++) {
      if(frame[j] < mean) {
        coolCount++;
        coolSum += frame[j];
      }
      else if(frame[j] > mean) {
        hotCount++;
        hotSum += frame[j];
      }
    }
    if(coolCount != 0 && hotCount != 0) {
      min = coolSum / coolCount;
      max = hotSum / hotCount;
    }
  }
  return convertToFahrenheit(min);
}

// Note: frame is already sorted by getAmbientTemp() before this is called
float ThermalCamera::getHotSpot() {
  float hotSpot = frame[767];
  return convertToFahrenheit(hotSpot);
}

Temperature ThermalCamera::getTemp() {
  Temperature currTemp;
  currTemp.ambientTemp = getAmbientTemp();
  currTemp.hotSpot = getHotSpot();
  return currTemp;
}

void ThermalCamera::printTemp(Temperature currTemp) {
  Serial.print("Hot Spot: ");
```

```
  Serial.println(currTemp.hotSpot);
  Serial.print("My Ambient Temp: ");
  Serial.println(currTemp.ambientTemp);
}
```