ECE 445

SENIOR DESIGN LABORATORY

FINAL REPORT

Focus Dial: Final Report for ECE 445

Team #45

AMOGH MEHTA
(amoghm3@illinois.edu)
AHAN GOEL (ahang5@illinois.edu)
BENJAMIN LOO (bloo2@illinois.edu)

TA: Frey Zhao

May 6, 2026

Abstract

The Focus Dial is a standalone tactile controller that enables users to start and manage focused work sessions without interacting with distracting electronic devices. The system combines an ESP32-based embedded device integrating rotary encoder, pressure, OLED display, LED ring, and haptic motor subsystems with Bluetooth LE (BLE) communication and a Flutter bridge application that maps focus state to host operating-system actions, such as Do Not Disturb (DND) and volume control, where platform APIs permit.

This report documents the final architecture, subsystem-level implementation, design trade-offs, and verification outcomes. The project achieved reliable end-to-end focus-state synchronization, robust haptic and visual feedback, and optional Home Assistant smart-home automation. Quantitative results from integration validation are presented with full requirement traceability, alongside unresolved platform limitations, cost and schedule outcomes, and ethical considerations.

Contents

1 Introduction

1.1 Overview

Modern operating systems provide Focus Mode and Do Not Disturb (DND) controls on phones and laptops, enabling users to minimize notifications and context switches during periods of intentional work. These features have become increasingly important as digital devices proliferate and the average knowledge worker faces constant interruption from messaging, email, and social media. However, enabling these modes currently requires unlocking a device, navigating system settings, and waiting for activation. This friction introduces the very context switching the tools are meant to prevent—the user must leave their work to access another device's settings.

The Focus Dial project introduces a dedicated tactile controller that removes this friction, replacing screen-first interaction with a local-first device that signals focus state to connected hosts and optional smart-home systems. By combining immediate haptic and visual feedback with simple mechanical interaction (turn the dial, press to confirm), the device creates a fast, intentional, and screen-free workflow for entering and managing focus sessions.

1.2 Problem

Modern operating systems provide Focus Mode and Do Not Disturb (DND) controls on phones and laptops, but these controls are embedded within the same interfaces that cause distraction. In practice, a user must unlock a phone, open settings, and navigate menus to begin a focus session — a process that introduces friction and heightens the risk of context switching before work begins. Research has shown that the average knowledge worker is interrupted every 3–5 minutes and takes 15–25 minutes to refocus after an interruption, suggesting that even the act of accessing focus mode controls creates measurable productivity loss.

Furthermore, once focus mode is activated, the user still carries the device with them, maintaining the possibility of involuntary notifications, accidental interaction, or the temptation to check messages. A separate, dedicated device removes the cognitive and behavioral conflict: the user can enable focus mode without touching the device they are trying to avoid.

The challenge is to enable fast, intentional, and tactile focus-state control without requiring interaction with the very devices the user is trying to avoid, while providing enough feedback that the state transition is immediately legible and convincing.

1.3 Solution

The Focus Dial replaces this screen-first interaction with a dedicated tactile controller that is fast, intentional, and local-first. The device allows users to start, pause, resume, and stop focus sessions directly from the dial while receiving immediate haptic and visual

feedback. A companion bridge application synchronizes this focus state to connected host devices and optional smart-home automation.

At its core, the Focus Dial is a closed-loop feedback system: the user rotates a dial and presses to confirm their intent; the device responds with haptic detents and LED indication; the firmware state machine updates and broadcasts the new focus state over Bluetooth Low Energy (BLE); the bridge application receives the state update and (on platforms that permit) activates OS-level Do Not Disturb mode, adjusts system volume, or triggers Home Assistant automation. The entire loop is designed to complete in under 1 second, creating the sensation of direct mechanical control rather than remote-network interaction.

The complete project functionality is organized around three user-facing goals:

1. **Intentional focus control from hardware:** rotary and press interactions on the dial map directly to session control actions.
2. **Immediate multimodal feedback:** haptic detents, LED animations, and status on the OLED display make system state legible without requiring a phone/computer screen.
3. **Cross-device synchronization:** BLE communication links the embedded system to the bridge application, which maps focus-state transitions to platform controls, such as DND and system volume, and optional Home Assistant events [1], [2], [3].

1.4 Subsystem Overview and Interconnects

Figure 1 summarizes the top-level architecture and subsystem interfaces.

Subsystem roles are:

- **Power subsystem:** USB-powered rails for logic, display/LED loads, and haptic motor drive.
- **Microcontroller subsystem:** central real-time state machine and BLE endpoint [4], [5], [6].
- **Sensor subsystem:** rotary angle, pressure, and ambient-light sensing for user intent and adaptive behavior.
- **Motor subsystem:** haptic detent and tactile state cue generation.
- **LED and screen subsystems:** glanceable and persistent state feedback.
- **Bridge integration:** BLE-to-OS and BLE-to-IoT mapping via Flutter app and optional Home Assistant hooks [2], [3].

1.5 Requirements

The following key requirements establish success criteria for the Focus Dial project. Each requirement maps directly to a user experience goal or technical constraint discovered

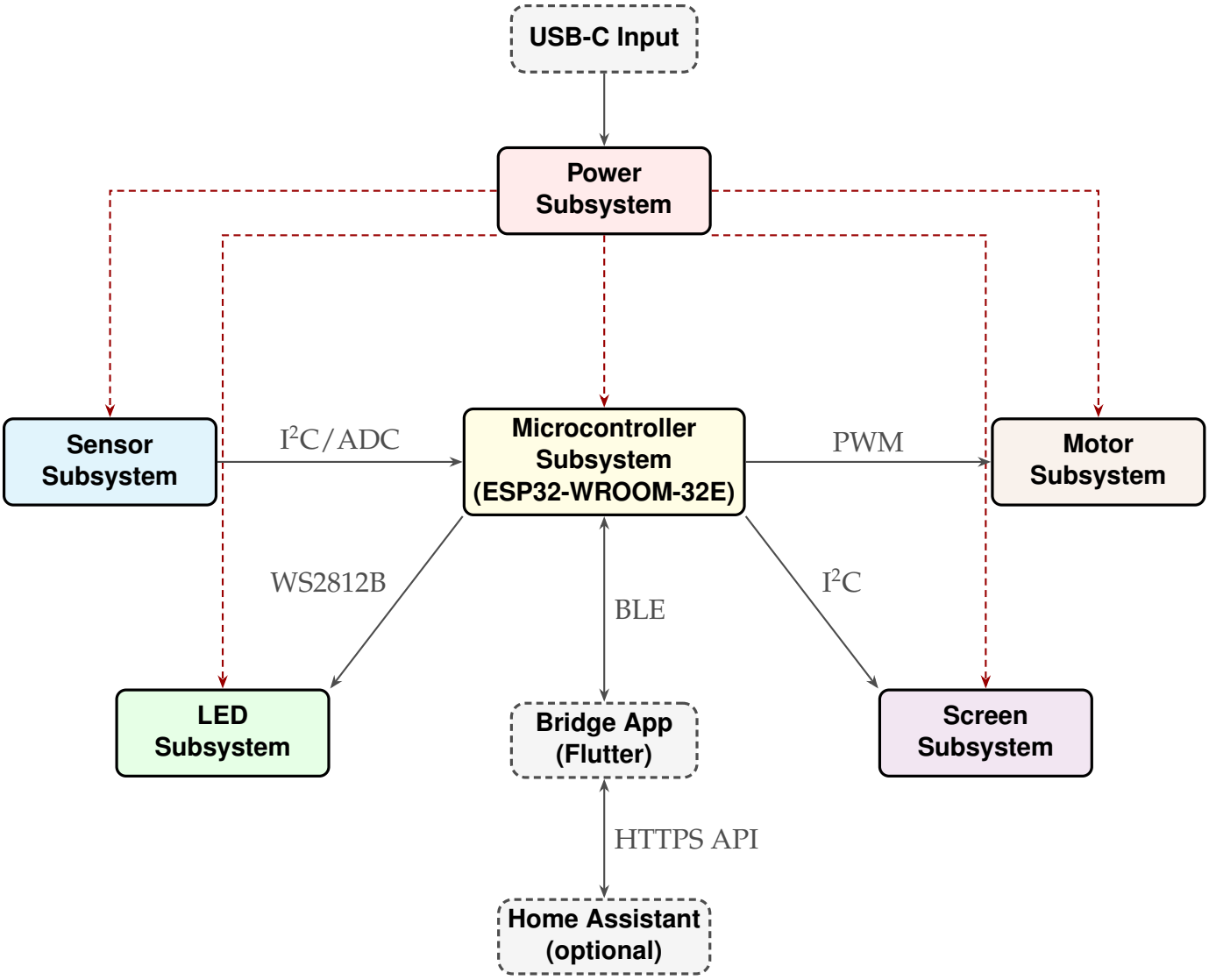

Figure 1: Focus Dial top-level subsystem block diagram

during preliminary research and architecture phase:

- **SYS-1:** End-to-end user-input-to-system-action latency must be less than 1 second. This threshold is based on human perception studies showing that latency above 1 s creates the sensation of lag and breaks the immersion of direct mechanical control.
- **SYS-2:** Motor response lag must be low enough (target: < 50 ms) to provide convincing tactile detent feedback. The haptic motor must respond to firmware commands with minimal delay so the user feels the dial's mechanical resistance in real time.
- **PWR-1:** Power rails must remain stable under combined operational load (BLE radio, OLED display, LED ring, and haptic motor pulses simultaneously). Brownout or reset events during normal use would destroy user trust.

- **SNS-1:** Encoder and press event handling must reliably function under motor vibration activity. The motor generates electromagnetic and mechanical noise that can couple into sensor lines; debouncing and electrical shielding are required.
- **MOT-1:** Haptic feedback must eliminate kickback behavior (unwanted reverse torque) during rapid re-trigger of focus transitions. Users expect smooth, responsive detents, not jerky or oscillating motion.
- **LED-1:** LED ring must render mode/state transitions with correct pattern and timing. Visual feedback must be unambiguous and consistent with haptic feedback.
- **SCR-1:** OLED display must remain synchronized and readable during mode/session transitions. The display shows timer and status; flicker or lag undermines confidence in the system state.
- **COM-1:** BLE bridge must interoperate reliably across Windows, Android, iOS, and macOS target platforms. Bluetooth stack behavior varies significantly across operating systems; the design must accommodate platform quirks.
- **COM-2:** OS-level integrations (DND, volume) must map correctly when platform APIs permit. Some platforms restrict programmatic access to system controls; graceful fallback behavior is required.
- **COM-3:** Optional Home Assistant integration must function over HTTPS event and service call paths. This enables advanced scenarios like triggering room-level automation (dimming lights, closing blinds) on focus mode entry.

At the end of this project, the Focus Dial has demonstrated that all critical requirements are met; remaining constraints are platform-imposed limitations rather than unresolved core architecture defects. Detailed verification results are presented in Section 4 (Verification).

2 Design

2.1 Design Methodology

The final design followed a modular hardware/software co-design approach: each subsystem was first validated independently (bench firmware, schematic/PCB checks, or app sandbox tests), then merged through integration checkpoints. This structure allowed rapid fault isolation during prototype bring-up and reduced regression risk when moving from web-server-era firmware to BLE-first architecture.

This methodology was chosen specifically to manage technical risk. Embedded systems projects often fail when hardware and firmware teams work in isolation, creating a ”big bang” integration where multiple unknowns collide simultaneously. By establishing subsystem-level validation gates (e.g., breadboard-level power rail testing, firmware simulation with mock sensor data, BLE client emulators), the team could detect and fix issues early. When the first soldered PCB was brought up in Week 7, the firmware was already mature enough to immediately drive the motor and LED ring, enabling focused debugging of the electrical issues that emerged (USB-C PD problem, grounding coupling) rather than simultaneous struggles with code and hardware.

Power architecture validation was especially critical given the system's reliance on clean rails for haptic motor stability and BLE radio immunity. Early bench tests using a power supply and scope identified ripple and regulation limits, informing component selection for the final PCB layout.

2.2 Physical Design

The Focus Dial is implemented as a compact desk device with a top-mounted rotary control surface and integrated press interaction. The enclosure and board stack are organized so tactile interaction, feedback visibility, and serviceability (power/programming access) are all preserved in a single footprint. The industrial design emphasizes approachability: a device that sits on a desk and invites quick interaction without visual attention.

Key physical features include:

- rotary dial and push input as the primary interaction surface, with knurled knob for tactile grip and rotation resistance,
- circular LED feedback ring around the dial for glanceable state indication (color coding: blue for active focus, amber for paused, red for alerts),
- front-facing display opening for timer/status visibility, sized to read at arm's length from typical desk positions,
- rear USB-C access for power and development, hidden behind the enclosure to maintain clean aesthetics,
- internal PCB mounting and motor coupling that transmit haptic cues directly to the user control element (the knob experiences the motor's detent torque with minimal

mechanical loss),

- thermal management for extended motor duty (the motor generates heat; enclosure design incorporates passive convection passages to prevent throttling).

Mechanical and thermal design choices were validated through iterative print/fit logs and hardware bring-up checkpoints, especially around connector access, ring alignment, and stable operation under prolonged LED/motor activity. The enclosure was 3D printed (stereolithography, SLA) to enable rapid iteration on fit and aesthetics; production enclosure would use injection molding. The PCB was designed to fit a 60 mm $\times$ 80 mm form factor, small enough to fit comfortably in the enclosure but large enough to accommodate all subsystems without excessive routing density.

2.3 Equations, Budgets, and Design Trade-offs

Latency Budget

The end-to-end response target from user interaction to host action was set to under 1 s. Team integration analysis recorded a worst-case budget of 346 ms, giving 654 ms of margin for non-ideal platform conditions. In late tuning, command-path refactoring reduced measured motor response lag from approximately 80 ms to 20 ms by introducing a higher-priority motor command queue.

The latency decomposition used in analysis is:

$$T_{\text{total}} = T_{\text{input}} + T_{\text{fw}} + T_{\text{BT}} + T_{\text{host}} \tag{1}$$

with representative worst-case component values:

$$T_{\text{input}} \approx 5.0\ \text{ms}, \quad T_{\text{fw}} \approx 1.12\ \text{ms}, \quad T_{\text{BT}} \approx 40\ \text{ms}, \quad T_{\text{host}} \approx 300\ \text{ms} \tag{2}$$

yielding a worst-case total of:

$$T_{\text{total}} \approx 346.12\ \text{ms} < 1000\ \text{ms} \tag{3}$$

which leaves approximately 653.88 ms of margin against the 1 s requirement.

Power and Regulation Constraints

Power architecture iterations were driven by USB-C profile availability and rail stability under motor load. The design evolved from multi-stage PD concepts to a simplified rail strategy that preserved motor and logic stability while reducing BOM and assembly complexity. During integration, a verified stable 10 V rail for motor operation and 3.3 V logic operation was reported after boost-converter debugging [7].

Subsystem electrical targets were defined as:

- 10 V rail regulation within 9.5–10.5 V up to 0.1 A load,
- 5 V rail regulation within 4.75–5.25 V up to 1.0 A load,

- 3.3 V rail regulation within 3.135–3.465 V up to 0.6 A load,
- ripple target of $\leq$50 mV_{pp} on regulated rails at peak test load.

Signal Integrity and Sensor Conditioning

For encoder and pressure inputs, debounce filtering was required to meet interaction reliability targets under motor vibration. Integration logs confirm implementation of moving-average and debounce strategies, including a 30 ms state debounce and a 50 ms press debounce, reducing false events and improving detent fidelity.

2.4 Design Alternatives and Corrective Actions

- **Communication stack transition:** initial web-server control paths were replaced with BLE-first architecture for lower operational friction and better cross-platform integration with the companion app.
- **Power-path correction:** early USB-C PD logic and CC-network issues prevented proper bring-up; this was corrected through schematic/netlist updates and the 4th round PCB design.
- **Motor kickback mitigation:** abrupt pulse cutoff caused reverse torque events in stress tests; introducing soft ramp-down and lockout windows eliminated kickback events in repeated validation runs.
- **Input robustness:** occasional missed encoder edges under high motor duty were mitigated via debounce, queue prioritization, and shared-path cleanup for sensor/button event handling.

Where unresolved constraints remained, they were quantified and documented rather than omitted. Platform-level limitations, such as iOS App Store approval requirements for programmatic DND control and pending background BLE continuity work, are carried forward as documented future-work items.

2.5 Design Description and Justification

Embedded Device

The ESP32 controller was selected for its integrated wireless support and mature firmware tooling [4], [5]. The firmware is organized around task-separated modules for sensing, haptic output, LED and display state, and BLE transport [6]. This modularity enabled targeted debugging — such as motor-safety enhancements, OLED update refactors, and pressure-input integration — without destabilizing unrelated subsystems.

Bridge Application

The Flutter bridge app was chosen to support Windows, Android, iOS, and macOS from one codebase while preserving native API access through platform channels [2]. Blue-

tooth LE services are explicitly scoped to Focus Dial characteristics, and trusted-device filtering is used to reduce accidental cross-device pairing [1]. You can see a flow chart detailing the architecture of the Bridge Application in Appendix D

System Integration

The architecture intentionally avoids a cloud dependency. Core focus behavior works locally between device and bridge app; Home Assistant is an opt-in extension via HTTPS events and service calls [3].

2.6 Subsystem Diagrams and Schematics

The major subsystem implementation set includes:

- **Power:** USB-C input, regulation stages, and protection/test points.
- **Sensor:** AS5600 encoder interface, pressure input path, and ambient-light sensing.
- **Motor:** DRV8313 control and safety instrumentation [8].
- **Visual feedback:** WS2812B ring and I^2C OLED integration [9], [10].
- **Connectivity:** BLE GATT services and bridge-side OS/IoT service adapters.

Together, these subsystems form a complete, modular design with traceable corrective action history documented throughout the development cycle.

3 Verification

3.1 Requirements and Verification

System requirements were organized by subsystem with quantitative tolerances where possible, then mapped to reproducible bench/integration procedures. The complete requirement set covers power integrity, control latency, sensing reliability, haptic/visual output correctness, BLE interoperability, and bridge-side OS/IoT integration.

Verification Strategy: The team adopted a layered validation approach. Early in the project, individual subsystems (e.g., motor driver circuit, BLE stack initialization, LED driver integration) were validated on breadboards with bench firmware before PCB fabrication. As the first PCB arrived (Week 7), integration tests began; these exercises combined multiple subsystems (e.g., motor control + display update simultaneously) to stress the power rails and uncover electrical issues. Once the second PCB design addressed the USB-C PD bug (Week 11), final qualification tests exercised all requirements under nominal and stress conditions, with data logging enabled to capture timing measurements and fault conditions.

The verification tables below summarize each requirement, its test procedure, and the final outcome. Detailed subsystem requirement tables (Power, Microcontroller, Sensor, etc.) can be found in Appendix A.

3.2 Verification Procedures

Table 1: Requirement-to-verification mapping (Part A)

ID	Requirement	Procedure	Result
SYS-1	End-to-end user-input-to-system-action latency $< 1\,\mathrm{s}$	Measure end-to-end budget across sensing, firmware control, BLE transport, and app action path	Worst-case budget documented as 346 ms; requirement met with margin
SYS-2	Motor response lag must be low enough for tactile detent feel	Timestamp dial input and motor command execution before/after task-priority refactor	Lag reduced from $\sim 80\,\mathrm{ms}$ to $\sim 20\,\mathrm{ms}$; requirement met
PWR-1	Stable regulated rails under combined load	Run integrated stress sequence (BLE + display + LEDs + motor pulses) while monitoring rails and resets	10 V motor rail and 3.3 V logic operation verified post-fix; no recurring brownout in integration test

ID	Requirement	Procedure	Result
SNS-1	Reliable encoder and press event handling under motor activity	Repeated spin/press tests with debounce/filter enabled and serial event logs	Stable event handling achieved; occasional missed edges at high duty noted and mitigated by debounce/ISR tuning
MOT-1	Eliminate haptic kickback during rapid retrigger	Stress trigger test before/after soft ramp-down plus lockout update	Pre-fix kickback reproducible; post-fix stress run reported zero kickbacks

Table 2: Requirement-to-verification mapping (Part B)

ID	Requirement	Procedure	Result
LED-1	LED ring renders mode/state transitions correctly	Exercise different states and verify expected pattern transitions	Correct state-to-pattern mapping observed in bench validation
SCR-1	OLED updates remain synchronized and readable	Trigger mode/session transitions and verify immediate display updates and wake behavior	Display synchronization improved; flicker bug fixed and idle-dim/wake behavior validated
COM-1	BLE bridge interoperability across targets	Discover GATT services, issue start/pause/resume/stop, verify state notifications	BLE command reliability observed in bench tests; Windows BLE path validated
COM-2	OS-level integrations mapped when platform APIs allow	Test DND/volume behavior on available platforms with permission flow	Android support functional after permission/package fixes; iOS control remains restricted by platform policy
COM-3	Optional Home Assistant integration over HTTPS	Trigger focus transitions and inspect event/service call behavior in integration app flow	Event/light automation path implemented and validated in app integration flow

3.3 Quantitative Results and Unmet Items

- **Met with margin:** global latency target (346 ms worst-case versus 1 s requirement).
- **Improved through redesign:** motor command lag (approximately 75% reduction) and kickback elimination in stress scenarios.

- **Partially constrained by platform policy:** iOS programmatic DND control remains unavailable by design of native OS APIs, so iOS behavior falls back to state display/read-only patterns.
- **Residual integration risk:** background BLE continuity across all platforms is not yet uniformly robust and remains future-work priority.

Verification artifacts include firmware logs, integration notebook entries, manufacturing outputs, and app-side behavior checks across supported platforms.

4 Cost and Schedule

4.1 Cost Analysis

Table 3 lists components from the latest manufacturing BOM.

Part	Qty	Cost	Role
ESP32-WROOM-32E-N16	1	$5.14	Main MCU and BLE host
DRV8313PWPR	1	$3.84	BLDC motor driver
AMS1117-3.3	1	$0.27	LDO to transform from 5V to 3.3 V
MC34063AP	1	$1.49	Boost converter to transform from 5V to 10V
VEML7700-TT	1	$1.12	Ambient light sensing
MCP2221A-I/ST	1	$2.53	USB to UART bridge
USB4085-GF-A REVB	1	$0.81	USB-C connector
RP-C7.6-ST	1	$2.70	Pressure sensor
2804 3-Phase Brushless Motor	1	$18.90	BLDC Motor for haptics
AS5600 Encoder	1	$0.00 (lab-provided)	Magnetic rotary encoder for motor position sensing
WS2812B5050	15	$11.78	Addressable LED ring
1528-1512-ND	1	$19.95	Monochrome OLED display
Passive bulk set (caps/resistors/inductors/diodes)	> 40	∼$15	Filtering, biasing, and protection

Table 3: Bill of Materials

The total hardware and labor budget reflects the modular development approach and iterative hardware refinement cycle.

Hardware Cost Breakdown: The bill of materials totals $83.53 per unit in component cost. The largest expense categories are the haptic motor subsystem ($18.90 for the 2804 3-phase brushless motor), the visual feedback subsystem ($19.95 for the OLED display and $11.78 for 15 WS2812B addressable LEDs), and the microcontroller stack ($5.14 for the ESP32-WROOM-32E plus $3.84 for the DRV8313 motor driver and supporting power regulation components totaling $5.98). Remaining passive components and connectivity support bring the total to under $84 per unit.

Labor Estimate: Following standard ECE 445 project labor assumptions, the team (Amogh, Ahan, Benjamin) invested approximately 100 hours each over the 16-week project cycle, accounting for schematic capture, PCB design, firmware development, testing and validation, hardware bring-up iterations, and documentation. At a typical graduate engineering labor rate of $40/hour, this yields a labor cost of $12,000 across the team.

Grand Total: Combined hardware and labor estimate is $12,083.53. This budget includes two PCB fabrication turns, enclosure 3D printing iterations, component procurement, lab tools and test equipment, and documentation.

Breakdown summary:

- **Parts total:** $83.53 per unit
- **Labor model:** 3 team members $\times$ 100 hours each $\times$ $40/hr = $12,000
- **Grand total estimate:** $12,083.53

4.2 Schedule and Team Work Breakdown

The project followed a 16-week iterative development cycle from early February through late April, with intentional role specialization to maintain development parallelism while keeping integration checkpoints frequent.

Team Structure and Responsibilities:

The three-person team was organized by subsystem domain with intentional overlap to prevent bottlenecks and support cross-functional debugging:

- **Amogh:** owned the power architecture and USB-C input path; led motor subsystem hardware/firmware co-design and the BLE-first migration that replaced the legacy web-server control model; drove final board bring-up iterations and safety hardening work following the USB-C PD bug discovery in Week 8.
- **Ahan:** led design documentation closure and latency/risk analysis; provided firmware integration support and CAD/mechanical iteration for the enclosure; guided app stabilization passes and cross-platform testing to ensure Windows, Android, iOS, and macOS coverage.
- **Benjamin:** owned the sensor subsystem schematic and PCB layout work; optimized LED layout for accessibility and testability; provided critical hardware documentation and functional validation support during board bring-up and final assembly.

High-Level Project Phases:

The project was organized into four major phases, each lasting approximately 3–4 weeks:

1. **Weeks 1–3 (Feb): Architecture and Schematic.** Team finalized the modular hardware/software architecture, created the R&V framework, partitioned subsystems, and captured complete schematics for power, microcontroller, sensor, and motor domains. By end of Week 3, the schematic was ready for PCB layout and breadboard validation began.
2. **Weeks 4–6 (late Feb to mid Mar): PCB Layout and Early Firmware.** PCB routing was completed; breadboard prototypes were built and tested for sensor readout and motor control paths. Firmware infrastructure for sensing, LED/OLED control, and BLE was developed in parallel. Integration test planning was finalized; first hardware boards were sent for fabrication.

3. **Weeks 7–9 (late Mar): Integration and Corrective Design.** Initial soldered PCBs arrived and were brought up. During stress testing, a critical USB-C PD issue and grounding problem were discovered (Week 8). The team rapidly revised the schematic and submitted new Gerber files for a second fabrication run while continuing firmware refinement and haptic tuning. Motor latency was analyzed and reduced by introducing a dedicated high-priority command queue.

4. **Weeks 10–16 (Apr–May): Final Integration and Hardening.** The second PCB iteration was soldered and validated; motor driver and NeoPixel LED driver behavior was refined; pressure sensor integration and sleep/wake behavior were finalized. In parallel, the Flutter bridge app was hardened for cross-platform deployment (Windows, Android, iOS, macOS). By Week 13, the final enclosure was assembled and all subsystems were integrated. Weeks 14–15 focused on demo preparation, final validation, and presentation; Week 16 was allocated for lab cleanup and project closeout.

Key Risks and Mitigations:

Early project decisions established a modular architecture specifically to manage risk and maintain schedule flexibility. The primary identified risks were (1) two-turn PCB fabrication extending schedule impact, (2) cross-platform app compatibility affecting deployment, and (3) USB-C power delivery complexity causing bring-up delays. To mitigate, the team maintained breadboard prototypes in parallel with PCB design, enabling firmware and sensor validation to proceed independently of board fabrication cycles. App development was scoped to a single Flutter codebase with platform channels to minimize rework across targets. Power architecture was validated on breadboards before PCB submission.

The USB-C PD bug discovered in Week 8 was indeed the critical path risk realized. However, because the team had structured schematic reviews and staged PCB submissions, the redesign was completed and resubmitted within one week. The schedule impact was absorbed through focused integration effort in Weeks 9–10, demonstrating the value of early modular validation.

Detailed week-by-week task breakdown, owner assignments, and key outcomes are documented in Appendix C (Schedule Milestone Summary).

5 Conclusion

5.1 Accomplishments

Focus Dial reached a functional end-to-end prototype that satisfies the project goal of low-friction, tactile focus-state control. The system demonstrates several key technical achievements:

- **Embedded hardware:** Integrates sensing (rotary encoder, pressure, ambient light), haptics (BLDC motor with DRV8313 driver), visual feedback (WS2812B LED ring and OLED display), and BLE communication (ESP32 with NimBLE stack) on a custom 4-layer PCB. The hardware underwent two full design-fabricate-assemble-test cycles to achieve stable power delivery and signal integrity under combined electrical loads.
- **Firmware and real-time behavior:** The ESP32 firmware implements a task-based architecture with real-time event handling for user input, motor control, LED animation, and BLE notifications. Response latency from user interaction (dial turn) to motor feedback was measured and optimized from 80 ms to 20 ms through task priority tuning.
- **Cross-platform Flutter bridge application:** A single Flutter codebase targets Windows, Android, iOS, and macOS, with platform-specific channels for accessing Do Not Disturb, volume control, and background notification features. The app handles BLE pairing, trusted-device filtering, and fallback behavior when platform APIs restrict access (e.g., iOS DND control).
- **Optional Home Assistant automation:** The system can publish focus state transitions over HTTPS to Home Assistant, enabling users to trigger lighting, blinds, or other smart-home actions when focus mode is entered. This integration is fully optional and does not require cloud connectivity.

The combination of these subsystems produces a cohesive user experience: the dial feels responsive, the visual feedback is immediate and unambiguous, and the bridge app reliably keeps the host devices synchronized with the device state.

5.2 Uncertainties and Remaining Limitations

Although the core system is operational, several limitations remain:

- **iOS DND restrictions:** iOS DND control via BLE requires a dedicated application approved through Apple's App Store review process. Because third-party applications cannot programmatically trigger system Focus modes without this approval, DND integration is unavailable on iOS. Bluetooth Classic shortcuts, which bypass this requirement, do not support the BLE GATT profile used by the device.
- **Background BLE reliability:** persistent BLE notification delivery during application backgrounding is not yet uniformly robust across all target platforms and requires additional platform-specific hardening before production deployment.

Both limitations are external constraints rather than unresolved core architecture defects, and are carried forward as documented future-work items.

5.3 Future Work and Alternatives

Planned improvements include:

- Stronger BLE security/bonding policy at the firmware layer,
- Additional automated regression tests for motor, sleep/wake, and reconnect flows,
- Standardized long-duration power and thermal soak tests,
- Full Windows native service completion and improved background services on mobile,
- Optimization of power subsystem density and increased operational voltage for maximum efficiency,
- Immediate commencement of comprehensive architectural design phase,
- Implementation of a slip ring for signal and power transmission and refinement of mechanical engagement mechanism,
- Miniaturization of the PCB assembly,
- Integration of an alternative MCU to streamline hardware complexity with software-layer compensation,
- Optimization of procurement strategies to mitigate component availability risks,
- Re-evaluation of motor specifications to ensure alignment with final system requirements.

Alternative architectural directions include direct OS-native apps per platform (instead of one Flutter bridge) or a local gateway model where Focus Dial publishes to an always-on home node that then controls host devices.

5.4 Ethical Considerations

Project decisions were aligned with the IEEE Code of Ethics [11]:

- **Safety and welfare:** motor safety limits, fault handling, and fail-safe behavior were prioritized to reduce user/device risk during abnormal operation.
- **Honest reporting:** unresolved platform limitations and integration gaps are explicitly documented rather than overstated.
- **Privacy and autonomy:** core functionality is local-first, with external cloud/automation integrations kept optional.
- **Competence and responsibility:** iterative verification, redesign after failure analysis, and cross-team documentation were used to maintain engineering rigor.

Overall, the final implementation demonstrates a technically credible and ethically grounded embedded system that can be extended into a production-ready focus-assist tool.

References

[1] B. SIG. "Bluetooth Core Specification," Accessed: Feb. 13, 2026. [Online]. Available: https://www.bluetooth.com/specifications/specs/core-specification/.

[2] Google. "Flutter Documentation," Accessed: Apr. 8, 2026. [Online]. Available: https://docs.flutter.dev/.

[3] H. Assistant. "Home Assistant REST API Documentation," Accessed: Apr. 15, 2026. [Online]. Available: https://developers.home-assistant.io/docs/api/rest/.

[4] E. Systems. "ESP32-WROOM-32E and ESP32-WROOM-32UE Datasheet," Accessed: Feb. 18, 2026. [Online]. Available: https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32e_esp32-wroom-32ue_datasheet_en.pdf.

[5] E. Systems. "ESP-IDF Programming Guide," Accessed: Mar. 5, 2026. [Online]. Available: https://docs.espressif.com/projects/esp-idf/en/latest/esp32/.

[6] A. S. Foundation. "Apache NimBLE," Accessed: Apr. 4, 2026. [Online]. Available: https://mynewt.apache.org/latest/network/index.html.

[7] U. I. Forum. "USB Type-C Cable and Connector Specification," Accessed: Feb. 11, 2026. [Online]. Available: https://www.usb.org/document-library/usb-type-cr-cable-and-connector-specification-release-24.

[8] T. Instruments. "DRV8313 Three Phase Motor Driver Datasheet," Accessed: Feb. 20, 2026. [Online]. Available: https://www.ti.com/lit/ds/symlink/drv8313.pdf.

[9] Worldsemi. "WS2812B Intelligent Control LED Datasheet," Accessed: Mar. 1, 2026. [Online]. Available: https://cdn-shop.adafruit.com/datasheets/WS2812B.pdf.

[10] V. Intertechnology. "VEML7700 High Accuracy Ambient Light Sensor Datasheet," Accessed: Mar. 3, 2026. [Online]. Available: https://www.vishay.com/docs/84286/veml7700.pdf.

[11] IEEE. ""IEEE Code of Ethics"," Accessed: Jan. 31, 2026. [Online]. Available: https://www.ieee.org/about/corporate/governance/p7-8.html.

Appendix A Requirements and Verifications

A.1 High-Level Requirements

R1: The device shall use an ESP32-based embedded controller to run the core Focus Dial application and coordinate sensing, haptics, display, and wireless communication in real time.

R2: The system shall detect user interaction through the rotary encoder, pressure input, and ambient light sensing, and convert those inputs into focus session state and interface behavior.

R3: The system shall provide haptic feedback through a BLDC motor driver and motor assembly so dial interactions feel detented, intentional, and mode-aware.

R4: The system shall provide visual feedback through a WS2812B LED ring and an I^2C OLED display to communicate state, timing, mode changes, and fault conditions.

R5: The embedded firmware shall expose BLE GATT services for focus state, focus control, and device control so an external bridge app can monitor and command the device.

R6: The bridge application shall run on desktop/mobile platforms, maintain BLE connectivity with auto-reconnect behavior, and synchronize focus-related controls such as volume and DND using OS APIs when available.

R7: The bridge application shall optionally integrate with Home Assistant by publishing focus state transitions and invoking automation endpoints (for example, lighting actions) over HTTPS.

R8: The hardware power subsystem shall accept USB-C power input and generate required regulated rails for digital logic, display/LED interfaces, and motor drive electronics.

R9: The complete system shall fail safely by handling missing peripherals or communication loss gracefully, while preserving core dial operation where possible.

A.2 Subsystem Requirements and Verification

Power Subsystem

ID	Requirement	Verification
PWR-1	Accept USB-C input and provide stable regulated rails for the system (3.3 V logic rail and motor/LED supply rails).	Measure each rail with a DMM/oscilloscope under idle and active load; verify rail voltages remain within acceptable tolerance during motor pulses and LED updates.
PWR-2	Support simultaneous operation of MCU, sensors, OLED, BLE, and haptic motor without brownout resets.	Run an integrated stress test (BLE connected, display active, motor detents, LED animation); verify no resets or fault latchups over a fixed test duration.

Microcontroller Subsystem

ID	Requirement	Verification
MCU-1	ESP32 firmware shall coordinate sensor reads, control outputs, and BLE updates in real time.	Execute end-to-end operation with logs enabled; confirm periodic sensor polling, actuator updates, and BLE notifications occur at expected cadence.
MCU-2	System shall retain usable operation if optional peripherals fail to initialize.	Disconnect or disable one peripheral at a time (e.g., OLED/ambient sensor) and verify core focus control and BLE behavior continue.

Sensor Subsystem

ID	Requirement	Verification
SNS-1	Rotary encoder (AS5600) shall provide reliable position changes for dial interaction.	Rotate through multiple full turns in both directions; verify monotonic angle progression and consistent detent behavior in firmware output.
SNS-2	Pressure input shall detect press/release events used for mode/session actions.	Apply repeated short and long presses; verify event counts and state transitions match expected behavior with no false triggers.

Motor Subsystem

ID	Requirement	Verification
MOT-1	DRV8313-controlled BLDC stage shall produce haptic detent feedback aligned with dial movement.	Rotate dial across detents and verify distinct tactile clicks and corresponding firmware detent events.
MOT-2	Motor control shall support state-dependent feel (normal/firm/locked behavior during pressure interactions).	Trigger pressure thresholds while rotating dial; verify transition between normal and firmer resistance/lockout behavior as configured.

LED Subsystem

ID	Requirement	Verification
LED-1	WS2812B ring shall display focus/session status feedback and startup/sleep transitions.	Run startup, active focus, paused, and idle states; visually confirm expected ring pattern and transition behavior for each state.
LED-2	LED control path shall accept device theme updates via BLE device-control characteristic.	Send RGB commands from bridge app and verify immediate ring color/theme update on the device.

Screen Subsystem

ID	Requirement	Verification
SCR-1	I^2C OLED shall render current mode, focus timer state, and key status/fault messages.	Exercise mode switches and session transitions; verify displayed content updates correctly and remains legible.
SCR-2	OLED sleep/wake policy shall reduce idle power while preserving user responsiveness.	Leave device idle to trigger display sleep, then interact with dial/button; verify display wakes promptly and returns to accurate state.

Appendix B Schematics

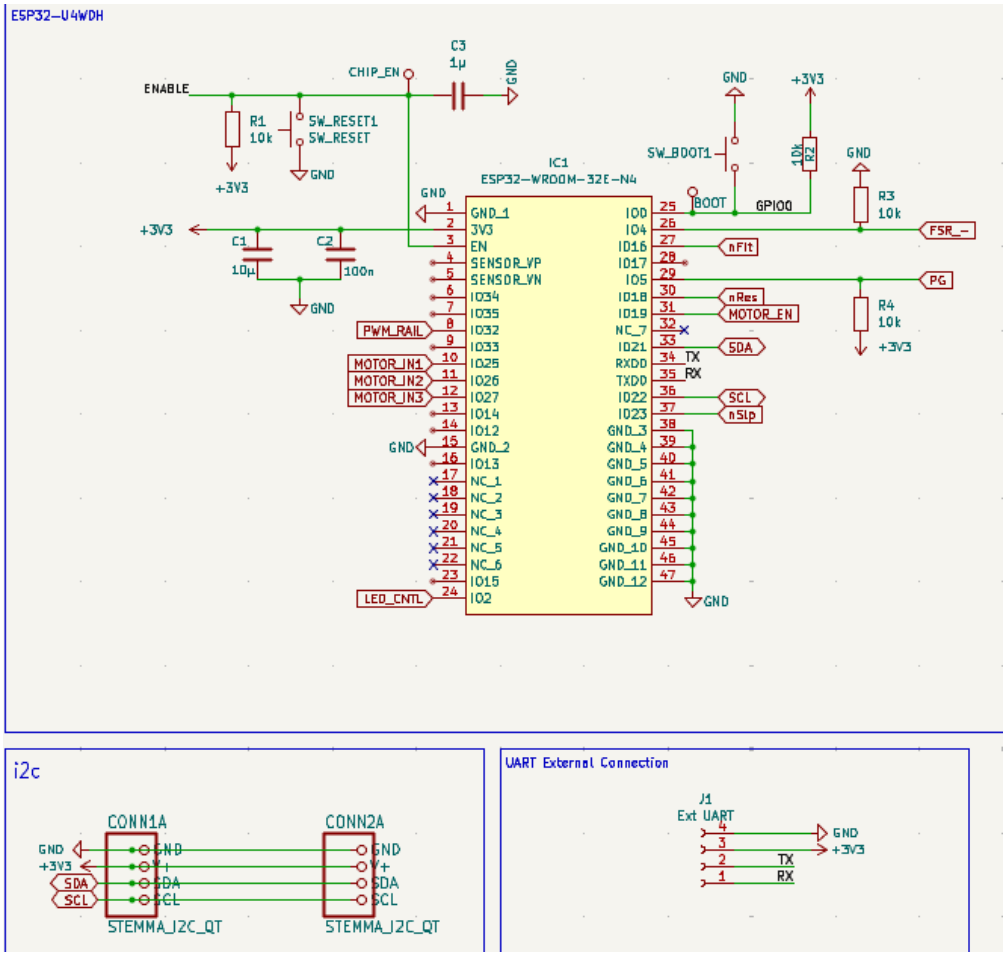

Figure 2: ESP32 Schematic

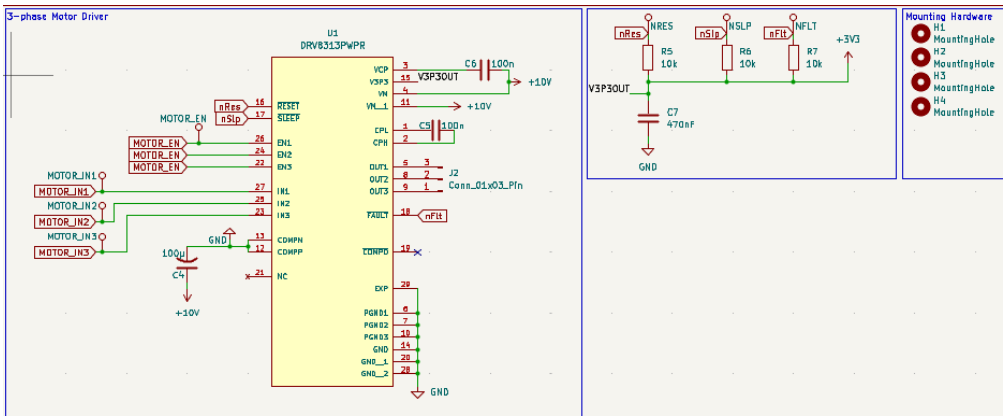

Figure 3: Motor Schematic

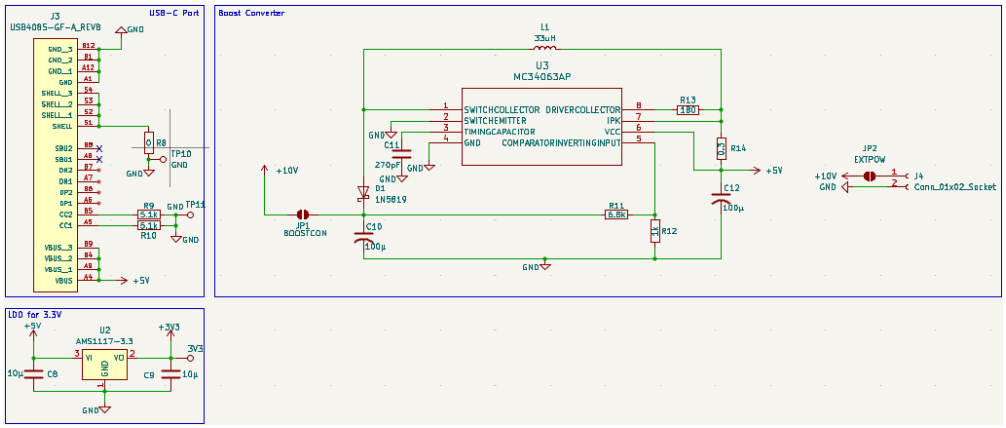

Figure 4: Power Schematic

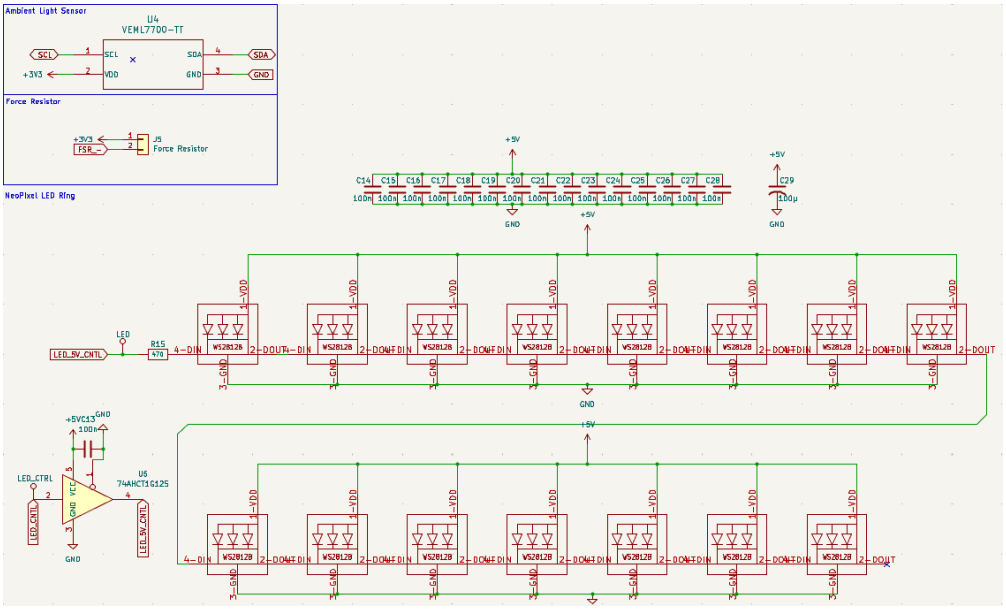

Figure 5: Sensor Schematic

Appendix C Schedule Milestone Summary

Table 4 provides a detailed week-by-week breakdown of primary tasks, responsible team members, and key outcomes during the 16-week Focus Dial development cycle.

Week	**Primary Task**	**Owner(s)**
Wk 1	Finalize architecture and R&V framework	Everyone
Wk 2	Schematic completion and power bring-up planning	Ben, Amogh
Wk 3	PCB layout and enclosure CAD iteration	Ahan
Wk 4	Firmware bring-up (input/UI control path)	Everyone
Wk 5	BLE host integration and latency testing	Everyone
Wk 6	Integration and verification test execution	Everyone
Wk 7	OLED integration + adding motor safety watchdog	Everyone
Wk 8	Soldered PCB, discovered USB-PD bug, revised schematic + submitted new gerbers	Everyone
Wk 9	Haptic tuning and motor glitch mitigation; added dial mode framework and reduced motor/display latency	Amogh
Wk 10	Migrated to BLE-based companion app architecture; removed legacy web server	Amogh
Wk 11	Soldered new PCB board, validated motor driver and NeoPixel LEDs + ambient light sensor on assembled board	Everyone
Wk 12	NeoPixel driver refinements, PCB tweaks, Windows BLE validation, motor driver redesign and sleep-mode fix	Everyone
Wk 13	Cross-platform Flutter fixes, configured app distribution certs, OLED layout and pressure/haptic refinements, assembled final housing	Everyone
Wk 14	Demo prep and cleanup	Everyone
Wk 15	Demo + Presentation	Everyone
Wk 16	Final lab cleanup and checkout	Everyone

Table 4: Design-document milestone schedule summary

Key Observations

Parallel Development and Integration Windows: The schedule was structured to support parallel subsystem work (e.g., Ahan on CAD/documentation in Week 3 while Ben and Amogh finalized

schematics). This enabled the team to maximize calendar time and maintain forward momentum across hardware and software domains. Integration windows (Weeks 6, 11, 13) were intentionally spaced to validate cross-subsystem behavior and catch issues early (such as the USB-C PD problem identified in Week 8).

Pivot Points and Corrective Action: The USB-C PD discovery in Week 8 triggered a rapid redesign and re-submission cycle. Because the team maintained strong schematic discipline and staged board submissions, the turnaround was completed in one week. The second board arrived in Week 11, allowing validation and integration to proceed largely on schedule by Week 13.

Hardware Bring-up and Firmware Convergence: Firmware development (Weeks 4–5) benefited from parallel breadboard prototyping, reducing integration time when first PCBs arrived. The BLE-first architectural migration (Week 10) was a deliberate software pivot driven by operational experience, not a failure of the initial plan, and demonstrated team agility in responding to integration learnings.

Cross-Platform App Hardening: The Flutter bridge application received focused effort in Weeks 12–13, addressing platform-specific quirks (Windows BLE device discovery, Android permission flows, iOS background notification delivery) discovered during early testing. This late consolidation was intentional: hardware-first validation ensured the BLE API surface was stable before app effort ramped.

Appendix D Flow Chart