ECE 445 Senior Design Laboratory

Final Report

StepWise: A Smart Insole System for Gait Analysis and Muscle Rehabilitation

---

Team 24

Kerui Xie (keruix2)
Nuo Pang (nuopang2)
Zhichao Chen (zc64)
Xiaorui Zhang (xz104)

May 2026

Abstract

StepWise is a portable smart-insole prototype for short walking tests and post-session gait screening. It is intended as an engineering and rehabilitation-support tool, not as a clinical diagnostic device. The final prototype uses four force-sensitive resistor pressure channels on a right-foot insole, together with an MPU6050 inertial measurement unit, to record plantar loading and foot orientation during a walking session.

The sensing unit is controlled by an ESP32. It samples the four pressure channels and inertial signals at a target rate of 100 Hz, then sends each sample to a host computer through WiFi UDP. Each packet contains 13 little-endian floating-point values, including four pressure readings, three acceleration values, three gyroscope values, and three orientation estimates. On the host side, a Tkinter-based graphical interface receives the stream, displays real-time curves, and saves the session as a structured text file. A separate offline analysis script processes the saved data, extracts stance-phase features, generates plots, and reports non-diagnostic loading-pattern findings.

A representative recorded session contained 1682 samples over 16.822 s, corresponding to an estimated sampling rate of 99.93 Hz. The analysis pipeline detected 12 single-foot stance phases and produced a high-quality, balanced-like screening result. The session also showed repeated PC timestamps, which is treated as a logging limitation of the current host-side timestamp method rather than as evidence of sensor failure. Overall, StepWise demonstrates a low-cost path from embedded pressure and motion sensing to interpretable gait-session feedback, while leaving final hardware calibration, larger validation trials, and clinical interpretation outside the scope of this prototype.

Contents

Chapter 1

Introduction

1.1 Motivation

Walking is an everyday motion, but the details of a person's gait are not easy to observe by eye. Small changes in plantar loading, foot strike, or foot orientation can appear long before they become obvious in a casual visual check. Clinical gait laboratories can measure these changes with high accuracy, but they are expensive, space-intensive, and not designed for frequent informal screening. At the other end of the spectrum, common fitness wearables usually report steps, speed, or general motion. They do not show how the foot is loaded during each stance phase.

StepWise was developed to sit between those two options. The goal is not to replace a gait lab or provide a medical diagnosis. Instead, the project builds a low-cost smart insole that can record useful pressure and motion signals during a short walking test, then turn the recorded session into clear engineering feedback. This makes the system more suitable for education, prototype rehabilitation support, and repeated testing during development.

1.2 Project Function and Scope

The final StepWise prototype is a right-foot smart insole with four pressure sensing regions and one inertial measurement unit (IMU). The four pressure channels are labeled P1–P4. In the current analysis layout, P1 corresponds to the heel, P2 to the arch or midfoot, P3 to the medial forefoot, and P4 to the lateral forefoot. These four regions were chosen because they preserve the main loading information needed by the present prototype while keeping wiring, mounting, and signal handling manageable.

During a walking session, the embedded system samples the pressure sensors and IMU at a target rate of 100 Hz. The ESP32 sends the data to a host computer through WiFi UDP on port 5005. The host program receives the packet stream, displays pressure and orientation curves, and saves the session for offline analysis. After the walk is complete, the analysis script smooths the pressure data, estimates stance phases, computes step-level features, and generates reports with plots and non-diagnostic screening findings.

The scope of the project is intentionally limited. StepWise can support engineering screening of loading patterns, such as a tendency toward balanced loading, medial or lateral bias, rearfoot-heavy loading, or weak push-off candidates. It should not be presented as a system that diagnoses flat foot, pronation, supination, toe-in, toe-out, or any medical condition. Those claims would require clinical validation and, in some cases, additional measurements such as video-based foot progression angle.

1.3 Final Prototype Overview

Figure 1.1 shows the implemented data path of the final prototype. The insole collects four pressure readings and IMU measurements. The ESP32-S3 firmware packages each sample as 13 little-endian floating-point values: four pressure channels, three acceleration values, three gyroscope values, and three estimated orientation angles. Since each float is 4 bytes, the transmitted payload is 52 bytes per sample. At 100 Hz, this gives a raw payload rate of approximately 41.6 kbps before WiFi and UDP overhead, which is well within the practical capability of the wireless link.

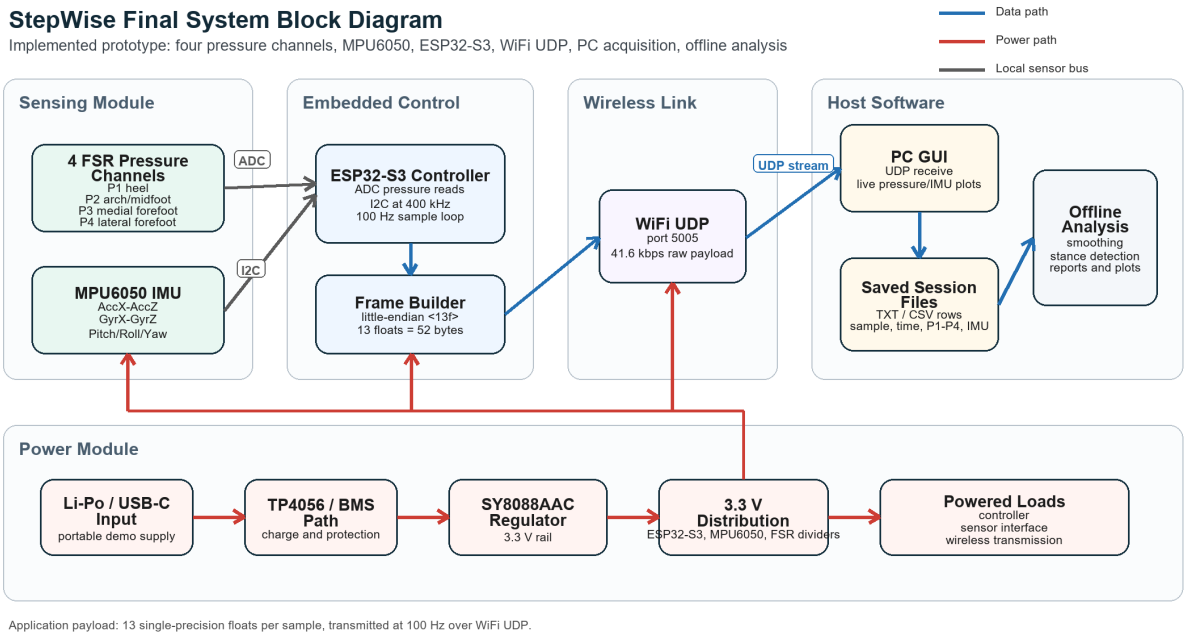

Figure 1.1: Final system block diagram for the StepWise smart-insole prototype. The implemented system uses four FSR pressure channels, an MPU6050 IMU, ESP32-S3 acquisition, WiFi UDP transmission, PC session saving, and offline analysis.

The host-side software is an important part of the system rather than only a display tool. It is responsible for receiving UDP packets, parsing the 13-float frame format, plotting incoming data, and saving each session as a text file with sample index, PC receive time, pressure channels, acceleration, gyroscope readings, and orientation estimates. This saved file is then used by the offline analysis pipeline. Keeping the screening step offline

makes the system easier to test and avoids giving distracting real-time feedback while the user is walking.

Figure 1.2 gives a physical overview of the insole and data-processing workflow. The final prototype uses four pressure sensors, not the five-point pressure layout described in the early design document. The original five-sensor plan remains useful historical context, but the final report treats the four-channel layout as the implemented design.

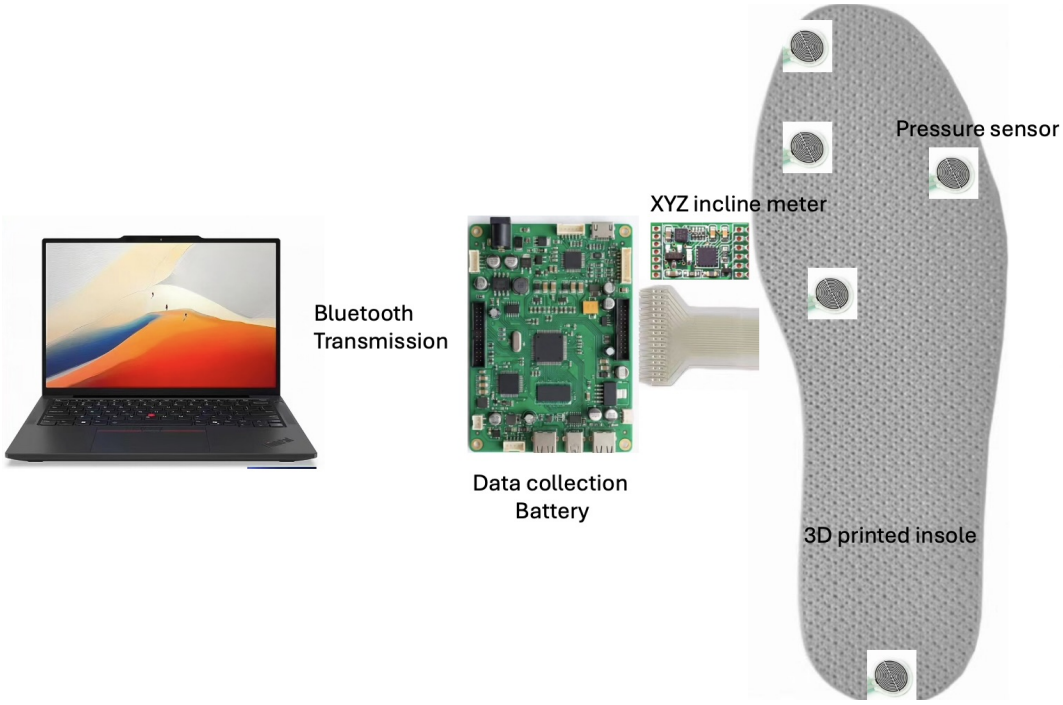

Figure 1.2: Physical overview of the StepWise insole, embedded sensing hardware, and host-side analysis path.

1.4 Requirements

The project requirements were written to check the complete sensing path, not just isolated components. The first requirement is end-to-end data integrity. During a standardized walking session, the system should collect and transmit synchronized data from all four pressure channels and the IMU, with no channel-level dropout longer than 1 s and no more than 5% session-level packet loss.

The second requirement concerns sensor behavior. The pressure subsystem should be repeatable under static and repeated loading, with target pressure error no greater than 5% for static loading and 8% for repeated loading. The IMU-based orientation subsystem should keep static angle error within 2 degrees and slow-tilt error within 3 degrees. These limits are meant to verify that the prototype is stable enough for engineering feedback, not that it meets clinical measurement standards.

The final requirement is software reliability. The PC program should parse at least 95% of valid frames and save complete session files. It should also report low-quality or

corrupted data instead of hiding it. For selected labeled or simulated screening tasks, the offline pipeline targets at least 90% accuracy. Appendix A lists the detailed requirement and verification plan.

1.5 Major Design Changes

The largest change from the original proposal was the move from five pressure points to four pressure channels. This change was made after considering mechanical space, wiring complexity, and the analysis features that were most important for the prototype. The final four-channel layout still separates heel, arch/midfoot, medial forefoot, and lateral forefoot loading, which is enough for the current rule-based screening pipeline.

The communication and analysis workflow also became simpler and more robust. Instead of building a complex real-time feedback loop, the final prototype sends a compact UDP stream to the PC and performs most analysis after the session is saved. UDP was chosen because it is lightweight and easy to handle at the required data rate. The tradeoff is that packet loss must be checked during verification, and the current text-file timestamp is based on PC receive time rather than an embedded sample timestamp. In the representative dataset, this caused repeated timestamps even though the estimated session-level sample rate was close to the 100 Hz target.

These changes narrowed the project into a more testable system. StepWise now focuses on reliable data collection, readable visualization, and interpretable offline feedback. That focus matches the final prototype better than a broader claim of real-time correction or clinical diagnosis.

Chapter 2

Design

2.1 Design Procedure

StepWise was divided into mechanical, power, pressure sensing, inertial measurement unit (IMU), embedded control, wireless communication, PC acquisition, and offline analysis blocks. Each block was designed around a practical constraint: the system had to be wearable enough for a short walking test, simple enough to build within the course schedule, and quantitative enough to produce useful engineering feedback. Table 2.1 summarizes the final prototype parameters used throughout the report.

Table 2.1: Key final-prototype implementation parameters.

Parameter	**Final prototype value**
Pressure channels	Four channels, P1–P4
Pressure mapping	P1 = heel, P2 = arch/midfoot, P3 = medial forefoot, P4 = lateral forefoot
IMU	MPU6050 over I2C
Embedded controller	ESP32-S3-based controller
Sampling target	100 Hz, 10 ms sample period
Wireless link	WiFi UDP, PC receive port 5005
UDP payload	Little-endian `<13f`, 13 floats, 52 bytes per sample
Analysis mode	Offline, rule-based engineering screening

2.2 Mechanical Insole Structure

The insole structure went through several material and layout choices. PLA and PETG were attractive early options because they are easy to print and hold their shape, but testing showed that they were too rigid for sensing: under body weight, they did not

transfer enough local deformation to the force-sensitive resistors (FSRs). The final design therefore uses a flexible thermoplastic polyurethane (TPU) lower layer and a cotton fabric upper layer. TPU lets the insole bend under the foot, while the cotton layer improves short-term comfort and keeps the user's foot away from printed edges and wiring.

The final prototype uses four sensing regions rather than the five-point target in the early design document. The analysis mapping is P1 for heel, P2 for arch/midfoot, P3 for medial forefoot, and P4 for lateral forefoot. Mechanically, the forefoot sensors are placed near the medial and lateral forefoot/toe-base areas so that push-off and side-to-side loading differences remain visible. This wording keeps the mechanical placement consistent with the firmware and analysis labels.

Electronics were kept out of the plantar contact area. A side-mounted container holds the PCB and battery, and hollow wire channels route the sensor leads toward the controller. This choice is less compact than a fully embedded commercial insole, but it avoids placing hard electronics under the foot and makes the prototype easier to inspect after a walking test. Figure 2.1 shows the CAD layout used for the final mechanical iteration.

Figure 2.1: Mechanical insole CAD layout with side electronics container and routed sensor paths.

2.3 Power and Pressure Sensing

The power subsystem could have used USB-only power, direct battery supply, or a regulated battery-powered design. USB-only power is convenient during debugging but does not support the wearable goal. Direct battery supply would reduce parts count, but it would expose the ESP32, IMU, and pressure-divider circuits to battery-voltage variation. The selected design uses a regulated 3.3 V rail so that the microcontroller and sensor interface circuits operate within their expected electrical range. Figure 2.2 shows the regulator block and SY8088AAC implementation.

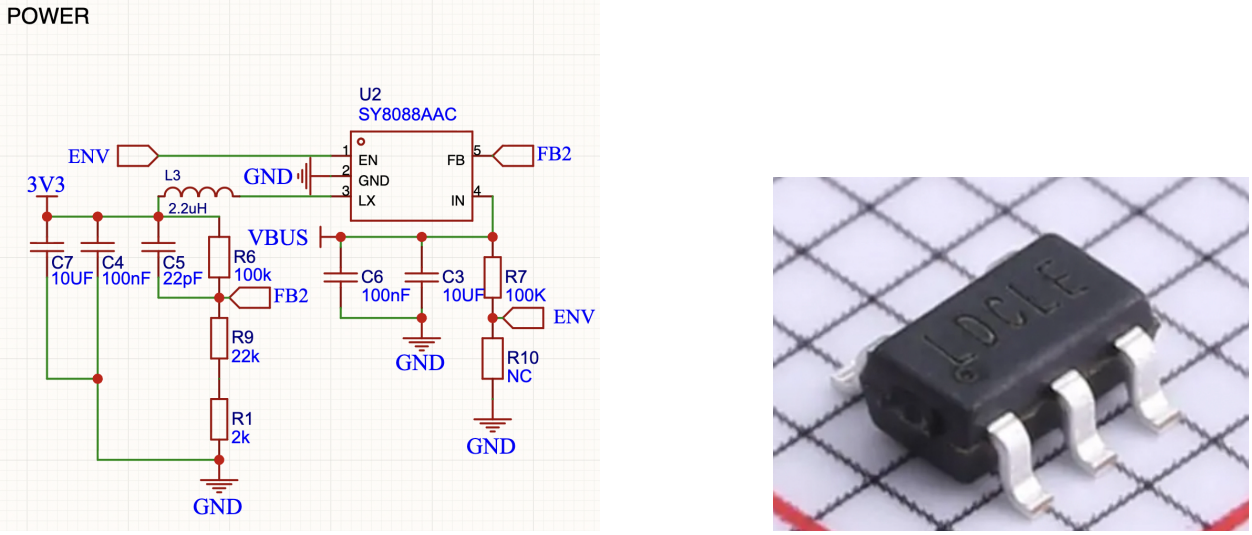

Figure 2.2: Power regulation subsystem and SY8088AAC buck-converter implementation.

Pressure was measured with FSRs because they are thin, inexpensive, and compatible with analog-to-digital converter (ADC) measurement. Load cells and commercial pressure insoles would provide stronger calibration, but they are bulkier or outside the budget and fabrication scope. In the final prototype, each FSR channel is read by the ESP32 ADC. The firmware records a no-pressure baseline at startup and maps the ADC drop into an approximate 0–500 N engineering scale. The scale is useful for visualization and relative loading patterns, but it should not be treated as clinical force measurement without a completed calibration test.

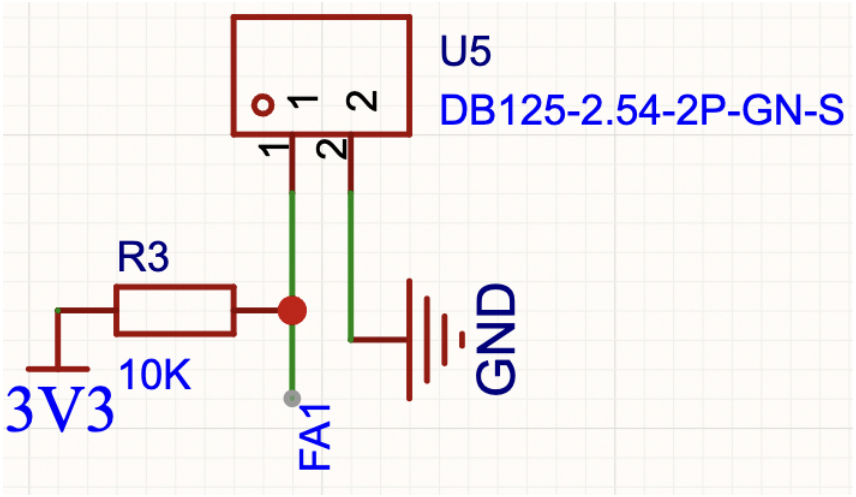

Figure 2.3: Pressure sensing interface used for plantar-loading measurement.

2.4 IMU and Embedded Firmware

The orientation subsystem uses an MPU6050 six-axis IMU. Pressure-only sensing cannot measure foot orientation, while a simple tilt switch would be too coarse. The MPU6050 provides three-axis acceleration and gyroscope signals, is supported by Arduino libraries, and can operate above the 50 Hz project requirement [1]. It is connected to the ESP32 through I2C with SDA on GPIO 4, SCL on GPIO 5, and a 400 kHz I2C clock.

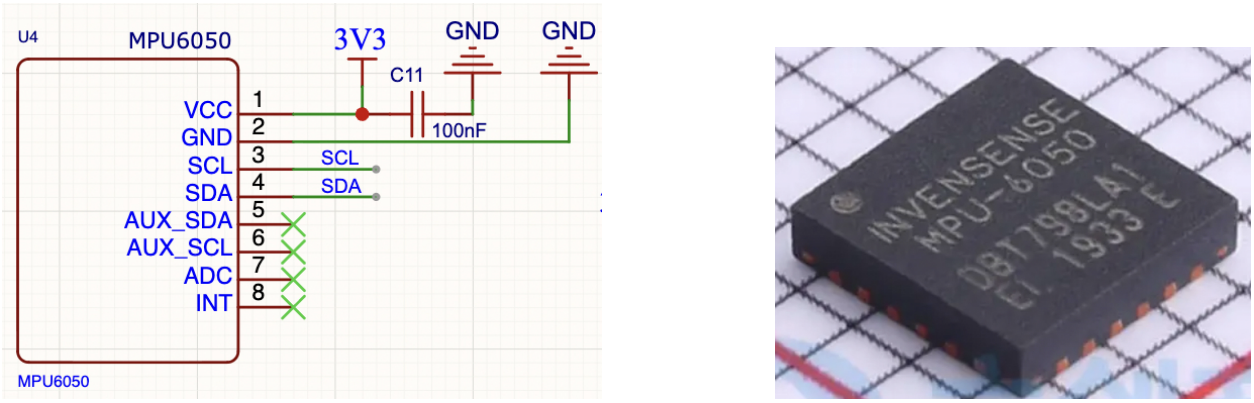

Figure 2.4: MPU6050 orientation-sensing subsystem and interface details.

Pitch and roll are estimated with a complementary filter that combines gyroscope integration with an accelerometer gravity reference. The final firmware uses $\alpha = 0.96$:

$$\theta[k] = \alpha\left(\theta[k-1] + \omega_y[k]\Delta t\right) + (1-\alpha)\tan^{-1}\left(\frac{-a_x}{\sqrt{a_y^2 + a_z^2}}\right), \tag{2.1}$$

$$\phi[k] = \alpha\left(\phi[k-1] + \omega_x[k]\Delta t\right) + (1-\alpha)\tan^{-1}\left(\frac{a_y}{a_z}\right). \tag{2.2}$$

Yaw is obtained by integrating the z-axis gyroscope, so it is treated as a relative trend rather than an absolute heading. Roll-based inversion or eversion feedback also depends on sensor mounting and requires static calibration before it can be interpreted confidently.

The ESP32-S3 was selected because it provides ADC inputs, I2C, integrated WiFi, and enough timing margin for 100 Hz acquisition [2]. The firmware runs a 10 ms sample frame:

$$T_s = \frac{1}{f_s} = \frac{1}{100\ \text{Hz}} = 10\ \text{ms}. \tag{2.3}$$

During each frame, the firmware reads four pressure channels, reads the IMU, updates pitch/roll/yaw, and sends one UDP packet. The final sensor pin assignment is shown in Table 2.2.

Table 2.2: Embedded sensor pin assignment in the final prototype.

Signal	ESP32-S3 pin	Function
P1	GPIO 12	Heel pressure channel
P2	GPIO 13	Arch/midfoot pressure channel
P3	GPIO 14	Medial forefoot channel
P4	GPIO 19	Lateral forefoot channel
SDA	GPIO 4	MPU6050 I2C data
SCL	GPIO 5	MPU6050 I2C clock

2.5 Wireless Communication and PC Acquisition

Wireless options included Bluetooth Low Energy, TCP over WiFi, UDP over WiFi, and USB serial. UDP over WiFi was selected because it is simple, has low overhead, and makes packet reception behavior visible during testing. Since UDP does not guarantee delivery [3], the PC side treats the stream as best effort and relies on saved-session checks before offline analysis.

Each sample contains 13 little-endian single-precision floating-point values:

$$\mathbf{x}[k] = [P_1, P_2, P_3, P_4, a_x, a_y, a_z, \omega_x, \omega_y, \omega_z, \theta, \phi, \psi]^T . \tag{2.4}$$

The payload is $13 \times 4 = 52$ bytes per sample. At 100 Hz, the raw application payload rate is 41.6 kbps, which is well below practical ESP32-S3 WiFi throughput [2]. The main communication risks are therefore packet loss and timestamp uncertainty, not bandwidth.

The host software is a Python Tkinter graphical user interface (GUI). It binds a UDP socket to port 5005, unpacks each complete 52-byte frame using the `<13f` format, displays pressure and IMU curves, and saves rows containing Sample, SystemTime, P1–P4, AccX–AccZ, GyrX–GyrZ, Pitch, Roll, and Yaw. Because SystemTime is assigned on the PC side, repeated millisecond timestamps can occur; the offline analysis therefore estimates sampling rate from total sample count and session duration.

Figure 2.5: PC GUI for UDP reception, live signal visualization, and session saving.

2.6 Offline Analysis Software

The offline analysis script parses the saved TXT file, smooths pressure channels, computes total and regional pressure ratios, detects stance intervals with adaptive hysteresis, and generates engineering screening findings. A rule-based method was selected over a machine-learning classifier because the prototype has limited labeled data and the user-facing output must remain explainable. Prior studies support the use of pressure insoles and wearable IMUs for gait-event detection and feature extraction [4, 5].

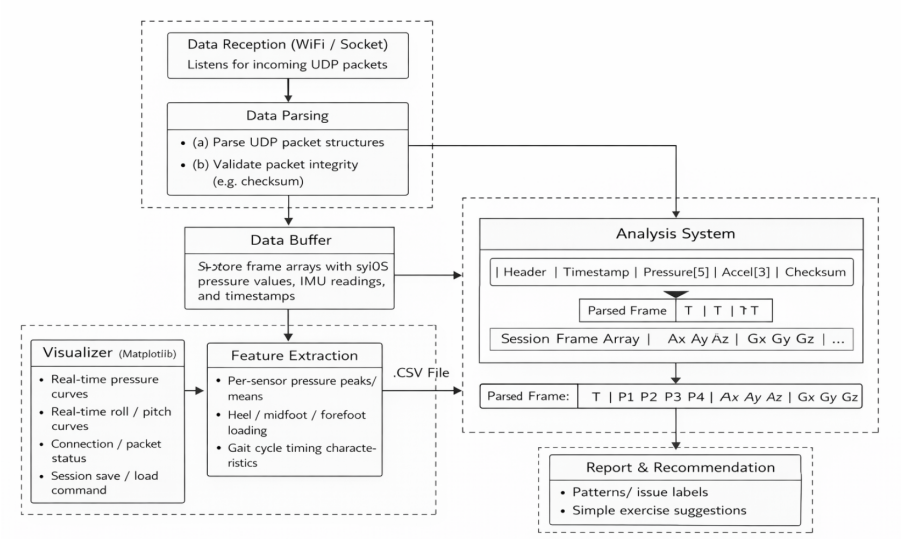

Figure 2.6: Host-side software architecture for offline data processing and gait screening.

The main pressure feature is total pressure,

$$P_{\text{tot}}[k] = \sum_{i=1}^{4} P_i[k]. \tag{2.5}$$

The regional ratios follow the final mapping. P1 represents heel loading, P2 represents arch/midfoot loading, P3 represents medial forefoot loading, and P4 represents lateral forefoot loading. A coarse medial-lateral center-of-pressure proxy is computed as

$$y_{\mathrm{CoP}} = \frac{0.45P_3 - 0.45P_4}{P_{\mathrm{tot}}}. \tag{2.6}$$

Positive values indicate more medial forefoot loading, and negative values indicate more lateral forefoot loading. Because only four pressure points are available, these values are coarse engineering proxies rather than pressure-platform measurements. The resulting report uses terms such as loading-pattern candidate and retest suggestion; it does not diagnose clinical gait or foot-posture disorders [6, 7].

Chapter 3

Verification

3.1 Verification Strategy

Verification focused on whether the integrated prototype could collect, transmit, save, and process a walking session. The final report separates measured prototype results from design estimates. The strongest measured evidence comes from the recorded walking session and the short mechanical walking test. Electrical accuracy, pressure-force calibration, IMU absolute angle accuracy, and formal packet-loss stress testing were estimated or partially verified because final bench-test records were not available. Appendix A gives the detailed requirement and verification table.

3.2 Power and Pressure Verification

The power subsystem requirement is that the regulated 3.3 V rail remain within 3.135–3.465 V while supporting the ESP32, IMU, and pressure-divider circuits. The design tolerance calculation estimates a worst-case SY8088AAC output range of 3.181–3.421 V, which stays inside the allowed range. This is a design-supported quantitative result, not a direct oscilloscope measurement. Final idle voltage, WiFi-streaming voltage, and ripple measurements were not available for this report, so the power subsystem is treated as design-supported but not fully verified by measurement.

The pressure-sensing verification objective is to confirm that P1–P4 respond monotonically and repeatably to load. The firmware performs startup no-pressure baseline calibration and maps ADC drop into an approximate 0–500 N engineering scale. A representative walking session showed nonzero pressure activity across all four saved pressure channels, which supports the use of the sensors for relative loading-pattern analysis. The pressure-divider tolerance calculation estimates a midpoint range of 1.519–1.783 V, within the intended 1.650 V $\pm$ 0.15 V electrical target. This result supports the circuit design, but it does not replace force calibration. Static-load error, repeated-load error, and channel-specific calibration curves were not available, so StepWise reports pressure-pattern screening rather than calibrated force measurement.

3.3 IMU Verification

The IMU requirement has two parts: output rate and angle accuracy. The output-rate portion was checked with the representative walking session. The PC saved 1682 valid frames over 16.822 s, giving an estimated session-level sampling rate of

$$\hat{f}_s = \frac{N-1}{T} = 99.93 \text{ Hz}. \tag{3.1}$$

This exceeds the original 50 Hz requirement and is close to the 100 Hz firmware target. For angle accuracy, the design estimate based on a typical MPU6050 accelerometer zero-g offset of ± 20 mg gives about 1.15 degrees of static tilt error, which is below the $\pm 2^\circ$ target. This is still only a design-supported estimate. Static roll/pitch tests against a known angle reference and slow-tilt tests against an inclinometer are still needed to determine whether the $\pm 2^\circ$ static and $\pm 3^\circ$ slow-tilt targets are met on the assembled insole. Walking data alone cannot separate true foot tilt from sensor mounting direction and user motion, so roll-based inversion or eversion feedback remains an engineering screening indicator.

Table 3.1: IMU verification summary.

Requirement	**Evidence**	**Status**
IMU output at no less than 50 Hz	1682 samples over 16.822 s, or 99.93 Hz	Verified
Static angle error within $\pm 2^\circ$	Fixed-angle reference test not completed	Not fully verified
Design-supported static tilt estimate	± 20 mg accelerometer offset gives about 1.15 degrees of tilt error	Estimated
Slow-tilt error within $\pm 3^\circ$	Inclinometer slow-tilt test not completed	Not fully verified

3.4 Embedded Sampling and Wireless Verification

The embedded sampling requirement is that all pressure and IMU fields be acquired and transmitted at no less than 50 Hz. The representative session supports this requirement at the session level: the saved file contained P1–P4, AccX–AccZ, GyrX–GyrZ, Pitch, Roll, and Yaw fields, and Equation (3.1) gives 99.93 Hz. The UDP payload also has sufficient bandwidth margin. Each sample is 52 bytes, so at 100 Hz the raw application payload is 41.6 kbps, far below practical WiFi throughput for the ESP32-S3. A 60 s formal packet-loss test at the same rate would expect about 6000 samples; this test should be repeated with an embedded sequence counter in future firmware.

The remaining wireless limitation is packet-loss measurement. The representative file contained 683 repeated millisecond timestamps. This is interpreted as a PC receive-time

timestamp limitation, not as evidence that the embedded sampler failed, because the total sample count and duration still match the target rate. The available file showed no channel-level dropout longer than 1 s, but a formal 60 s, 3 m line-of-sight packet-count test is still needed to verify session-level packet loss. A future firmware revision should add an embedded sample counter or embedded timestamp so packet loss can be measured directly.

3.5 PC Acquisition and Offline Analysis Verification

The PC software requirement is that valid UDP frames be parsed and saved reliably, and that incomplete or low-quality data not be converted into unsupported gait conclusions. Each valid UDP frame contains 52 bytes, corresponding to 13 little-endian floating-point values. For a UDP datagram with B_{datagram} bytes, the number of complete frames is

$$N_{\text{valid}} = \left\lfloor \frac{B_{\text{datagram}}}{52} \right\rfloor . \tag{3.2}$$

The parsing success rate for a formal test set is

$$S_{\text{parse}} = \frac{N_{\text{valid}}}{N_{\text{total}}} \times 100\%. \tag{3.3}$$

A formal parser stress test with valid, truncated, extra-byte, and empty-session inputs has not yet been completed, so the 95% parser target is not fully verified. The current evidence is still useful: a representative saved session was processed successfully, and the required 13 fields were present in the saved rows.

The offline analysis pipeline was verified on the representative recorded session. It produced processed CSV data, step-level features, JSON summaries, pressure and orientation plots, a technical HTML report, and a user-facing non-diagnostic report. The session contained 1682 samples over 16.822 s, had an estimated sample rate of 99.93 Hz, and yielded 12 detected single-foot stance phases. Figure 3.1 shows the processed pressure signals and detected stance intervals.

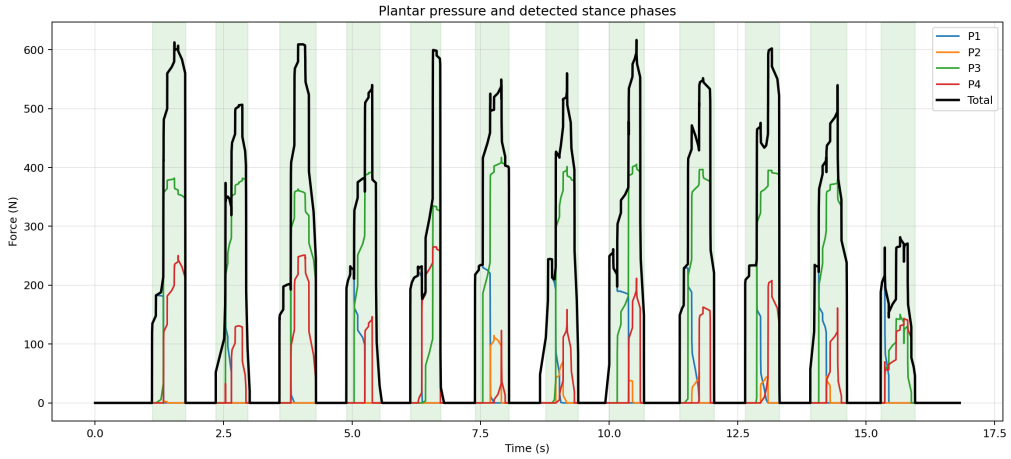

Figure 3.1: Processed plantar-pressure signals and detected stance phases from a representative StepWise session.

The representative output was labeled High data quality and produced a balanced-like engineering screening result. This verifies the saved-file-to-report workflow, but it does not by itself verify clinical accuracy or the uncompleted hardware calibration tests. The user-facing language therefore uses terms such as loading-pattern candidate and retest suggestion rather than clinical diagnostic labels.

3.6 Mechanical and User-Fit Verification

Mechanical verification checked whether the insole could support short-term walking, activate the pressure sensors, and remain mechanically stable. The final TPU-based prototype was tested through a 5 min walking trial with 100 continuous steps, followed by inspection of the insole, sensors, wiring, and side-mounted electronics container. The PLA/PETG prototypes were too rigid to activate the sensors reliably, while the TPU design deformed under body weight and allowed all four pressure channels to respond during walking.

The user completed the short walking test without stopping and reported a comfort rating of 4/5. After 100 steps, there were no cracks, no sensor displacement, no layer separation, no exposed wire damage, and no container failure. The PCB and battery remained outside the plantar contact area. These results verify the mechanical prototype for short-term demonstration use. Longer walking tests and tests with more users are still needed before making broader comfort or durability claims.

3.7 Requirement Verification Summary

Table 3.2: High-level requirement verification summary.

Requirement	Evidence	Status
Data acquisition rate	1682 samples over 16.822 s, estimated 99.93 Hz	Verified
Data integrity	Required fields present; no channel dropout longer than 1 s observed in the representative file; formal packet-loss test not completed	Partially verified
Sensor accuracy	Regulator, divider, and IMU static-angle estimates support design targets; force and angle bench tests not completed	Estimated / not fully verified
PC analysis	Saved-session analysis completed; 12 stance phases, High data quality, balanced-like output; parser stress test not completed	Partially verified
Mechanical fit	5 min, 100-step TPU insole walking test completed	Verified for demo use

Chapter 4

Costs

4.1 Parts and Manufacturing Cost

Table 4.1 summarizes the prototype cost. The retail values are estimates for building one prototype, while the paid values reflect the team's current records and estimates. Lab-owned or reused items should be marked as zero paid cost only if the team confirms that they were not purchased for this project.

Table 4.1: Estimated prototype parts and manufacturing cost.

Item	**Retail estimate (RMB)**	**Paid estimate (RMB)**
Electronic components, including ESP32-S3, MPU6050, CH340C, regulator, connectors, switches, LEDs, and passives	500	478.5
PCB fabrication	200	120.2
3D-printed insole housing	450	450
Mechanical assembly	50	50
Estimated total	**1200**	**1098.7**

4.2 Labor Cost

The ECE 445 guideline recommends estimating labor cost for each partner using

$$C_{\text{labor}} = r_{\text{hourly}} \times h_{\text{actual}} \times 2.5, \tag{4.1}$$

where r_{hourly} is the ideal hourly salary, h_{actual} is the number of hours worked, and 2.5 accounts for overhead. Table 4.2 uses a planning estimate of 50 RMB/h and 50 h per team member.

Table 4.2: Estimated labor cost.

Team member	Primary role	Rate (RMB/h)	Hours	Cost (RMB)
Kerui Xie	Electrical design and validation	50	50	2500
Nuo Pang	PC software and analysis	50	50	2500
Zhichao Chen	Mechanical design and integration	50	50	2500
Xiaorui Zhang	Embedded firmware and communication	50	50	2500
Total				**10000**

4.3 Project Schedule

Table 4.3 summarizes the working schedule used for the project. The schedule was organized around the four main responsibilities in the team: mechanical structure, electrical hardware, embedded firmware, and PC-side analysis.

Table 4.3: Project schedule from initial planning to final report.

Period	Main work	Primary responsibility
W1–W2	Define project scope, requirements, target users, and safety boundaries.	Whole team
W3–W4	Select sensing layout, ESP32-S3 controller, power architecture, WiFi UDP communication, and initial insole form.	ME, EE, CompE
W5–W6	Build first insole CAD, pressure-divider circuits, firmware skeleton, and PC receiver prototype.	Zhichao, Kerui, Xiaorui, Nuo
W7–W8	Integrate four FSR channels, MPU6050, 100 Hz sampling, UDP packets, and live PC plots.	EE and CompE
W9–W10	Iterate TPU insole, route wiring, mount side electronics, and check walking comfort.	ME with team support
W11–W12	Record walking data, run offline analysis, update verification evidence, costs, and safety language.	Whole team
W13	Finalize report, figures, presentation material, and non-diagnostic wording.	Whole team

4.4 Commercialization Considerations

A commercial version would need lower assembly cost, thinner electronics packaging, verified battery operation, improved durability, and a larger validation set. Bulk component purchasing and molded insole manufacturing could reduce unit cost, but StepWise should not be described as commercially ready until comfort, safety, calibration, and user-data handling are tested beyond the short demonstration prototype.

Chapter 5

Conclusion

5.1 Executive Summary

StepWise demonstrates a portable smart-insole workflow for standardized walking tests. The final prototype integrates four plantar-pressure channels, an MPU6050 IMU, ESP32-based 100 Hz acquisition, WiFi UDP transmission, a PC acquisition GUI, and offline rule-based analysis. The strongest verified result is the end-to-end data path: a saved walking session was processed into stance-phase features, plots, and a non-diagnostic screening report. Hardware accuracy and wireless packet-loss requirements are now reported with quantitative design-supported estimates where bench data were not available, but they remain only partially verified until final calibration and stress testing are completed.

5.2 Accomplishments

The project produced an integrated sensing, communication, and analysis pipeline. The embedded system streams pressure, acceleration, gyroscope, and orientation values to the PC. The PC GUI records session data and displays real-time curves. The offline analysis script processes saved sessions into step features, screening findings, risk flags, plots, and user-facing reports.

The representative recorded session shows that the software pipeline can parse and process real sensor data, detect stance phases, and produce a balanced-like engineering screening result. It contained 1682 samples over 16.822 s, corresponding to an estimated sampling rate of 99.93 Hz, and the analysis detected 12 single-foot stance phases. The packet format also gives a clear communication load: 52 bytes per sample at 100 Hz, or 41.6 kbps before WiFi and UDP overhead. These results support the feasibility of the host-side workflow, while still leaving pressure accuracy, IMU angle accuracy, and packet-loss performance as separate measurement items.

5.3 Limitations

The current system is a single-foot prototype with four pressure channels. It cannot directly measure left-right asymmetry, and pressure distribution alone cannot prove clinical conditions such as flat foot, pronation, supination, toe-in, or toe-out. The current PC GUI timestamps packets using receive time, which can create repeated millisecond timestamps and limit fine-grained timing interpretation.

Mechanical testing was limited to short-term use. The TPU lower layer and cotton upper layer improved comfort compared with earlier PLA/PETG prototypes, but long-duration comfort, shoe fit, sweat exposure, and repeated-day durability were not tested. The side-mounted electronics container keeps the PCB and battery away from the plantar surface, but it also increases the size of the prototype.

5.4 Ethics, Safety, and Broader Impact

StepWise collects pressure and motion signals that may reveal behavioral patterns. Session data should be minimized, anonymized, and stored securely. Participants should understand what data are collected and that StepWise provides engineering screening and rehabilitation-support suggestions rather than medical diagnosis.

The project has potential value as a low-cost educational and screening tool, but it also carries risk if users overinterpret results. The team has a responsibility to avoid misleading claims, consistent with the IEEE Code of Ethics [8]. If user data are stored or shared, privacy expectations should be discussed with reference to data-protection principles [9]. Safety considerations include low-voltage electrical operation, battery handling, insulation, mechanical comfort, and avoiding sharp or rigid features in the insole [10, 11].

5.5 Future Work

Future work should add embedded timestamps or sample counters, run formal packet-loss tests, complete pressure calibration, and verify IMU angle accuracy with known-angle fixtures. A larger baseline dataset would allow the rule thresholds to be refined and tested across more users and walking styles. A bilateral version could support left-right gait symmetry, and a video-based foot progression angle measurement could help distinguish toe-in and toe-out from pressure-related loading candidates.

Mechanical future work should improve long-term comfort, sensor mounting, and wearability. Additional pressure sensors could increase spatial resolution, while a thinner electronics package and more protected wire routing would make the system easier to fit into normal shoes.

Appendix A: Requirements and Verification Summary

Table 1: Detailed requirements and verification plan.

Subsystem	Requirement	Verification method	Current result
System data integrity	Four pressure channels plus IMU are transmitted during a walking session, with packet loss no greater than 5% and no channel dropout longer than 1 s.	Run a timed walking or bench session, count received packets, and inspect P1–P4 plus IMU columns for gaps.	Required fields were present in a representative saved file, and no channel-level dropout longer than 1 s was observed. Formal packet-loss counting remains not fully verified.
Power	3.3 V rail remains within 3.135–3.465 V and supports WiFi current spikes.	Measure rail voltage under idle and WiFi-streaming load; measure ripple with an oscilloscope if available.	Design-supported estimate: 3.181–3.421 V worst-case regulator output. Final voltage and ripple measurements remain not fully verified.

Table 1: Detailed requirements and verification plan (continued).

Subsystem	Requirement	Verification method	Current result
Pressure sensing	P1–P4 respond monotonically and meet static/repeated load error targets.	Apply known loads to each channel, record readings, and repeat after loading cycles.	Walking data showed all four channels responding. Divider estimate: 1.519–1.783 V midpoint range. Formal force calibration and repeated-load error remain not fully verified.
IMU	Static angle error is no greater than 2 deg; slow-tilt error is no greater than 3 deg.	Place sensor/insole at known angles and compare pitch/roll output to reference.	Continuous IMU output was recorded. Static tilt design estimate is about 1.15 deg from ± 20 mg accelerometer offset; fixed-angle and slow-tilt measurements remain not fully verified.
Embedded timing	Sensor acquisition and UDP packet generation support the 100 Hz target.	Use recorded session duration and sample count; optionally measure firmware loop timing with a GPIO toggle.	Current evidence: 1682 samples over 16.822 s, estimated 99.93 Hz.
Wireless link	UDP link supports 52-byte packets at 100 Hz over the demo range.	Run a 3 m line-of-sight test and compare expected versus received packet count.	Payload bandwidth is design-verified at 41.6 kbps raw payload. A 60 s test would expect about 6000 packets; formal packet-loss result remains not fully verified.

Table 1: Detailed requirements and verification plan (continued).

Subsystem	Requirement	Verification method	Current result
PC GUI	Valid packets are parsed and saved without program interruption.	Feed valid `<13f` packets and malformed/truncated packets to the GUI.	Representative session was saved successfully; formal parser stress-test rate remains not fully verified.
Offline analysis	Saved sessions are processed into CSV, JSON, PNG, HTML, and text reports without misleading low-quality conclusions.	Process representative and intentionally poor-quality sessions. Inspect outputs and warnings.	Representative output generated 12 stance phases, risk flags, plots, High data quality, and a balanced-like screening result.

Supplemental R&V Rows

Table 2 lists the requirement-and-verification rows related to the IMU, embedded sampling, wireless communication, and PC-side software. These rows supplement the main verification discussion in Chapter 3.

Table 2: R&V rows for IMU, embedded timing, wireless communication, and PC software.

Subsystem	Requirement	Verification and Result	Status
IMU and orientation sensing	IMU data shall be acquired at no less than 50 Hz during walking trials.	A representative saved session contained 1682 frames over 16.822 s, giving an effective rate of 99.93 Hz. The saved rows included AccX–AccZ, GyrX–GyrZ, Pitch, Roll, and Yaw.	Verified

Table 2: R&V rows for IMU, embedded timing, wireless communication, and PC software (continued).

Subsystem	Requirement	Verification and Result	Status
IMU and orientation sensing	Roll and pitch estimates shall be suitable for engineering orientation screening after static or known-direction calibration.	The firmware computes pitch and roll using accelerometer estimates and gyroscope integration through a complementary filter. Walking data confirmed continuous orientation output; absolute clinical angle accuracy was not claimed.	Partially verified
Embedded sampling	The firmware shall sample four pressure channels and one IMU in a 100 Hz acquisition loop.	The firmware uses a 10 ms timer period. The representative session rate of 99.93 Hz supports that the embedded-to-PC path meets the timing target at the saved-file level.	Verified
Embedded data fields	Each transmitted sample shall contain P1–P4, AccX–AccZ, GyrX–GyrZ, Pitch, Roll, and Yaw.	The PC parser unpacks each valid UDP frame as 13 little-endian floats. Saved TXT and CSV outputs contain all 13 fields.	Verified
Wireless communication	The wireless payload rate shall fit within the ESP32-S3 WiFi link.	The application payload is 52 bytes/sample. At 100 Hz, the raw payload rate is 41.6 kbps, which is well below practical WiFi throughput.	Verified by design
Wireless communication	The saved session shall show no channel-level dropout longer than 1 s.	The representative saved file contained all 13 fields across the session, with no observed channel dropout longer than 1 s.	Partially verified
PC acquisition	The GUI shall receive and save valid UDP frames without converting incomplete frames into valid samples.	The GUI processes data in 52-byte blocks using the `<13f>` format. Incomplete trailing bytes are ignored rather than saved as samples.	Partially verified by representative-session processing and code inspection

Table 2: R&V rows for IMU, embedded timing, wireless communication, and PC software (continued).

Subsystem	**Requirement**	**Verification and Result**	**Status**
Offline analysis	The analysis script shall convert a saved TXT session into structured output files and a non-diagnostic report.	The representative session generated processed CSV data, step-level features, JSON summaries, PNG plots, a technical HTML report, and a user-facing report.	Verified
Report safety	User-facing outputs shall not claim clinical diagnosis.	Report wording uses engineering screening terms such as “loading-pattern candidate,” “loading-pattern finding,” and retest suggestions.	Verified by review

This appendix keeps the detailed requirement rows so Chapter 3 can remain concise.

Appendix B: Supplemental Figures and Tables

This appendix is reserved for supplemental figures and large tables that support the main text but would interrupt the main narrative. Every figure or table placed here should still be cited directly from the main text.

Supplemental Analysis Plot

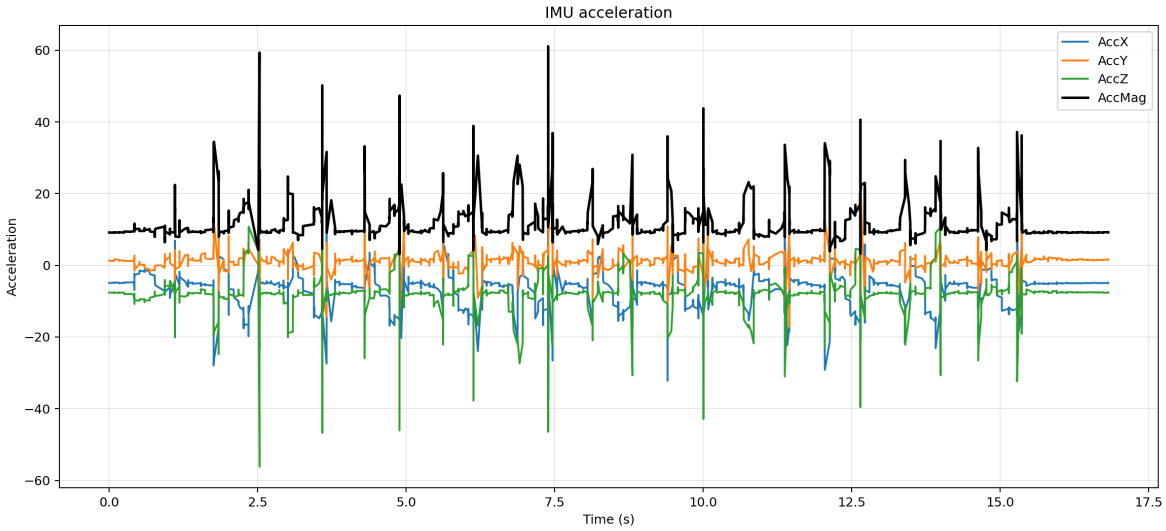

Figure 1: Supplemental acceleration magnitude and axis plot from the representative StepWise session.

Mechanical Modeling Figures

Figure 2: Supplemental CAD views of the final mechanical insole model.

Appendix C: Data and Software Notes

UDP Packet Format and Embedded Fields

The final embedded prototype transmits real-time data using WiFi UDP on port 5005. Each sample is encoded as 13 little-endian 32-bit floating-point values, for a total payload size of 52 bytes. The PC program unpacks each frame using the Python format string `<13f`. Table 3 lists the transmitted fields.

Table 3: UDP packet format for one StepWise sample.

Index	Field	Unit	Description
0	P1	N	Heel pressure channel
1	P2	N	Arch/midfoot pressure channel
2	P3	N	Medial forefoot pressure channel
3	P4	N	Lateral forefoot pressure channel
4–6	AccX, AccY, AccZ	m/s^2	IMU acceleration axes
7–9	GyrX, GyrY, GyrZ	rad/s	IMU gyroscope axes
10	Pitch	deg	Complementary-filter pitch estimate
11	Roll	deg	Complementary-filter roll estimate
12	Yaw	deg	Relative yaw estimate from gyro integration

The application payload rate is

$$52 \text{ bytes/sample} \times 8 \text{ bits/byte} \times 100 \text{ samples/s} = 41.6 \text{ kbps} \tag{1}$$

This calculation excludes UDP/IP and 802.11 overhead, but it is sufficient to show that the sensor stream is small relative to the available WiFi bandwidth.

Embedded Sampling Notes

The firmware targets a 100 Hz sampling rate, corresponding to a 10 ms frame period. During each frame, it reads the four analog pressure channels, reads one MPU6050 sample over I2C, computes pitch, roll, and yaw, and transmits the 13-field UDP frame. The current firmware also includes a batch-send buffer, but the final operating mode used for testing is real-time UDP transmission of one sample frame at a time.

Pressure values are generated from a startup no-load ADC baseline. This conversion provides an engineering force scale for visualization and feature extraction. It is not a clinical-grade pressure calibration curve. The pressure-channel placement used by the final analysis is P1 = heel, P2 = arch/midfoot, P3 = medial forefoot, and P4 = lateral forefoot.

TXT and CSV Session Format

The PC acquisition GUI saves a human-readable TXT file during each session. Each data row contains the fields listed in Table 4. The CSV export uses the same sensor fields in comma-separated form.

Table 4: Saved TXT/CSV session fields.

Field group	**Fields**	**Use in offline analysis**
Sample metadata	Sample, SystemTime	Row indexing and time normalization
Pressure	P1, P2, P3, P4	Contact detection, regional loading ratios, CoP proxy, pressure plots
Acceleration	AccX, AccY, AccZ	Motion-intensity plots and acceleration magnitude
Gyroscope	GyrX, GyrY, GyrZ	Angular-rate plots and gyroscope magnitude
Orientation	Pitch, Roll, Yaw	Foot-orientation trends and roll-range screening features

Offline Analysis Outputs

The offline pipeline reads the saved TXT file and writes the output files listed in Table 5. These files separate machine-readable data, visual inspection plots, and user-facing report content.

Table 5: Offline analysis outputs generated from a saved StepWise session.

Output file	**Purpose**
`processed_gait_data.csv`	Smoothed pressure channels, time-normalized samples, contact state, ratios, and CoP proxies
`gait_steps_analysis.csv`	Step-level stance time, stride time, pressure impulse, regional ratios, roll range, and pattern labels
`session_summary.json`	Sample count, duration, estimated sampling rate, detected stance phases, thresholds, and warnings
`screening_findings.csv/json`	Engineering screening findings and interpretation text
`risk_flags.csv/json`	Grouped risk flags for data confidence, rhythm, landing, pressure transfer, and regional loading
`pressure_stance.png`	Pressure traces with detected stance intervals shaded
`acceleration.png`	Acceleration channels and acceleration magnitude
`orientation.png`	Pitch, roll, and yaw traces
`report.html` and `report.txt`	Technical analysis report
`user_report.html`	User-facing non-diagnostic screening report
`data_guide.html`	Explanation of formulas, sensor mapping, and interpretation limits

Timestamp Limitation

The current GUI assigns timestamps using the PC receive time. This design is simple and useful for debugging, but it has limited precision. Several packets can be processed within the same millisecond, so repeated timestamps can appear in the saved TXT file. In the representative session, the analysis reported 683 repeated timestamps while the session-level sampling rate remained 99.93 Hz.

For this reason, the offline analysis estimates the effective sampling rate from total sample count and total session duration:

$$\hat{f}_s = \frac{N-1}{t_{\text{last}} - t_{\text{first}}} \tag{2}$$

The repeated-timestamp warning should be interpreted as a data-recording limitation rather than proof that the embedded sampler or sensor stream failed. A future firmware revision should add an embedded sample counter or embedded timestamp to each UDP frame. That change would allow the PC program to distinguish wireless packet loss,

sampling jitter, and PC timestamp quantization more accurately.

Rule-Based Screening Logic and Literature Basis

The StepWise offline analysis uses rule-based engineering screening rather than clinical diagnosis. The formulas in Table 6 define the pressure ratios, CoP proxy, baseline comparison, and stride variability used by the software. The risk rules in Table 7 summarize how these metrics are mapped to non-diagnostic user feedback.

Table 6: Core formulas used by the StepWise offline analysis.

Metric	Formula	Use
Total pressure	$P_{\text{tot}} = P_1 + P_2 + P_3 + P_4$	Contact detection and normalization
RearRatio	P_1/P_{tot}	Heel/rearfoot loading
ArchRatio	P_2/P_{tot}	Midfoot/arch loading proxy
MedialRatio	P_3/P_{tot}	Big-toe side loading
LateralRatio	P_4/P_{tot}	Little-toe side loading
CoP_ML proxy	$(0.45P_3 - 0.45P_4)/P_{\text{tot}}$	Positive is medial, negative is lateral
Baseline delta	current metric $-$ baseline metric	Device-specific comparison
Stride CV	$\sigma(T_{\text{stride}})/\mu(T_{\text{stride}})$	Step timing variability

These thresholds are engineering thresholds selected for the StepWise four-sensor prototype. Public datasets and papers justify the direction of the indicators, but the system does not use them as clinical diagnostic cutoffs.

Known Software Limitations

- The current packet format has no embedded sequence number, so packet-loss verification must rely on received sample counts unless sequence numbering is added.
- The current saved timestamp is based on PC receive time, so it is less precise than an embedded sample timestamp.
- The current screening rules are engineering indicators, not clinical classifiers.
- The current pressure mapping assumes the final P1–P4 physical sensor positions match the analysis layout.

Table 7: Rule-based screening indicators and literature basis.

Risk indicator	Trigger evidence	Reference direction
Eversion / over-pronation tendency	Roll delta toward eversion, MedialRatio delta ≥ 0.030, ArchRatio delta ≥ 0.030, or CoP_ML delta ≥ 0.030	Pronated feet show more medial arch and first/second metatarsal loading in the primary foot-posture dataset [6, 7].
Inversion / over-supination tendency	Roll delta toward inversion, LateralRatio delta ≥ 0.050, or CoP_ML delta ≤ -0.040	Supinated feet show more external/lateral loading in the primary foot-posture dataset [6, 7].
Forefoot-first landing	EarlyFrontRatio ≥ 0.650 or delta ≥ 0.200	FSR insole studies use heel/forefoot contact timing to describe gait events [4].
Reduced push-off	Late forefoot push-off proxy < 0.750 or delta ≤ -0.080	Forefoot/metatarsal loading is expected during propulsion; reduced late forefoot loading is treated as weak push-off participation.
High stride variability	Stride CV above the engineering threshold	Stride-time variability is a common temporal gait feature in wearable gait analysis [5].

Bibliography

[1] InvenSense Inc., *MPU-6000 and MPU-6050 Product Specification*, rev. 3.4 ed., 2013.

[2] Espressif Systems, *ESP32-S3-WROOM-1 and ESP32-S3-WROOM-1U Datasheet*, v1.8 ed., 2025.

[3] J. Postel, "User Datagram Protocol," RFC 768, Internet Engineering Task Force, 1980.

[4] A. M. Ngueleu, A. K. Blanchette, L. Bouyer, D. Maltais, B. J. McFadyen, H. Moffet, and C. S. Batcho, "Design and accuracy of an instrumented insole using pressure sensors for step count," *Sensors*, vol. 19, no. 5, p. 984, 2019.

[5] A. M. de-la Herran, B. Garcia-Zapirain, and A. Mendez-Zorrilla, "Gait analysis methods: An overview of wearable and non-wearable systems, highlighting clinical applications," *Sensors*, vol. 14, no. 2, pp. 3362–3394, 2014.

[6] E. Sanchis-Sales, J. L. Sancho-Bru, A. Roda-Sales, M. J. Chiva-Miralles, and C. Garcia-Gomariz, "Foot kinematics and kinetics data for different static foot posture collected using a multi-segment foot model," *Scientific Data*, vol. 11, no. 1307, 2024.

[7] A. C. Redmond, Y. Z. Crane, and H. B. Menz, "Normative values for the foot posture index," *Journal of Foot and Ankle Research*, vol. 1, no. 6, 2008.

[8] IEEE, "IEEE code of ethics." [Online]. Available: https://www.ieee.org/about/corporate/governance/p7-8.html, 2020.

[9] European Union, "General Data Protection Regulation (GDPR)." [Online]. Available: https://gdpr.eu/, 2016.

[10] International Electrotechnical Commission, "IEC 62368-1: Safety of electrical equipment," 2018.

[11] ASTM International, "ASTM D7791: Standard test method for uniaxial fatigue properties of plastics," 2022.