IMPACT INSOLES

INSOLE PRESSURE SYSTEM FOR RUNNING SHOES

By
Joseph Casino
Aarush Sivanesan
Matthew Weng

Final Report for ECE 445, Senior Design, Spring 2026
Professor: Craig Shultz
TA: Xiaodong Ye

4 May 2026
Project No. 68

Abstract

Impact Insoles is a wearable plantar pressure monitoring system designed to provide runners with accessible, real-time gait analysis data. Existing solutions such as clinical force plates are stationary, expensive, and inaccessible to most athletes and coaches on a day-to-day basis. This project addresses that gap by creating an array of twelve thin-film force-sensing resistors (FSRs) onto a flexible PCB insole, routing signals through a multiplexed analog front end to an ESP32-S3 microcontroller, and streaming data wirelessly via Bluetooth Low Energy (BLE) to a browser-based web application. The system samples all sensor channels at ~100 Hz with less than 5% measured packet loss, operates for over one hour on a 3.7 V, 400 mAh LiPo battery, and reports running cadence within ±3 steps per minute (SPM) of a reference measurement. The web application renders a live pressure heatmap overlaid on a foot silhouette, logs gait events, and records session data for post-run analysis. All primary requirements for this project were met or exceeded, validating the design as a lower-cost, portable alternative to laboratory-grade gait analysis equipment.

Contents

1. Introduction

a. Problem and Solution

Running is one of the most popular and accessible forms of physical activity worldwide, offering significant cardiovascular and mental health benefits. However, running is also associated with a high incidence of overuse injuries, with studies estimating that 50% of runners experience a debilitating injury each year. Common causes include excessive impact forces, poor foot-strike mechanics (e.g., heel striking vs. midfoot striking), asymmetrical loading between feet, and inconsistent cadence. Many of these issues stem from improper gait mechanics that go unnoticed by runners until pain or injury develops. Currently, accurate measurement of foot-ground interaction requires gait laboratories equipped with force plates, pressure mats, or instrumented treadmills. These systems are expensive, stationary, and typically limited to clinical, research, or elite athletic settings. While some consumer wearables can estimate cadence or pace using wrist or hip mounted inertial sensors, they do not directly measure foot pressure or impact forces, which are critical indicators of injury risk and running efficiency.

Impact Insoles addresses this gap. This is a thin-film pressure sensor insole system that integrates directly into the running shoes that measures the force applied by the foot to the ground throughout each step during real outdoor running and streams the data wirelessly to a browser-based interface for live visualization and analysis. As shown in Figure 1, the system consists of these integrated subsystems: a flexible sensor insole array, an analog front-end and ADC, an ESP32-S3 microcontroller with BLE telemetry, and a web application running on a smartphone or laptop.

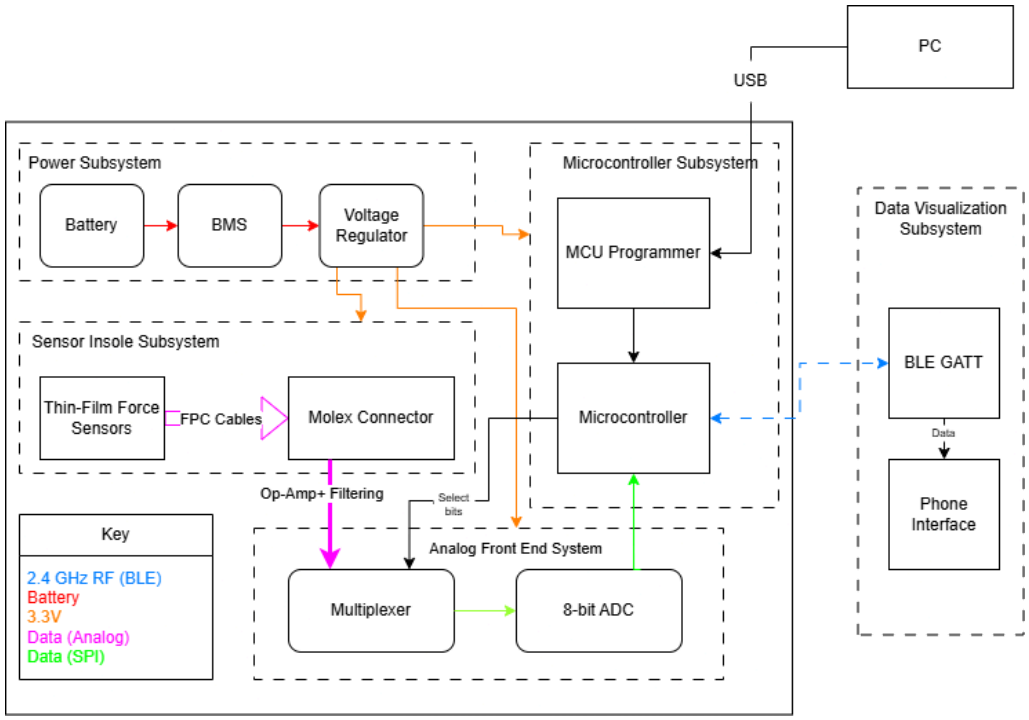

Figure 1.1: Impact Insoles System Block Diagram

A flexible sensor array of 12 thin film force sensors embedded on top of the shoe foam (or placed under the insole) will capture pressure through the foot's main contact regions (forefoot, heel, and midfoot) and track the pressure distribution of the foot throughout the run. A compact electronics module will attach to the shoe heel or tongue and contain the microcontroller, battery, and Bluetooth modules. The MCU will sample the pressure sensor data, detect foot-strike events, and compute basic metrics such as step count and contact time. Data is transmitted wirelessly to a smartphone, where it can be visualized, logged, and analyzed. This approach provides runners with direct, actionable feedback on their running mechanics without requiring expensive lab equipment or professional supervision.
Several design changes were made during the semester. The sensor count was reduced from 16 to 12 to simplify flex PCB routing and reduce fabrication cost while maintaining adequate spatial coverage across all three foot zones. The insole PCB was redesigned from a single full-footprint board to two symmetric polyimide flex PCB hubs joined at a central connector hub. The phone interface was implemented as a browser-based web application using the Web Bluetooth API rather than a native mobile app, eliminating any installation requirement. The planned optional IMU subsystem (LSM6DSOX) was deprioritized and not implemented in the final build.

b. High-Level Requirements List

For our project to be successful in solving our problem, it must meet these standards:

- Efficiency: The system shall sample plantar pressure sensor data at a minimum rate of 100 Hz and transmit the data over Bluetooth Low Energy with no more than 5% packet loss during continuous operation.
    - Justification: to ensure the usability of the device to track changes over time as well as to ensure proper remote operation
- Accuracy: The system shall detect foot-strike events and report running cadence with an accuracy of ±3 BPM compared to a stopwatch or smartwatch reference over a controlled running trial.
    - Justification: to ensure the received data is representative of what is really happening at the foot.
- Continuity/Longevity: The device shall operate continuously for at least 1 hour on battery power while performing active sensing and BLE data streaming.
    - Justification: to ensure the viability of using this device on real runs

The key performance factors in this system are the aggregate sensor sampling rate, BLE packet delivery reliability, and battery longevity. At 100 Hz, each 28-byte BLE notification packet (24 bytes of sensor data plus a 4-byte sequence number) produces a data rate of approximately 2.8 kB/s which is well within BLE 4.2 connection interval limits, but requires careful firmware scheduling to avoid buffer overflows. Battery life is governed by the combined current of the ESP32-S3 during active BLE advertising and the analog front-end, both operating from the 3.3 V regulated rail.

2. Design

a. Physical Design

The Impact Insoles physical system consists of a flexible sensor insole and a compact electronics module. The insole contains thin-film force sensing resistors (FSRs) positioned at key plantar regions: heel, midfoot (arch), and forefoot. There will be 12 sensors arranged around the foot as shown in figure 2.1.

Figure 2.1: FSR sensor placement

The sensors were embedded onto a polyimide flex PCB that matches the geometry of a standard running shoe insole. The total thickness was approximately 0.3 mm to avoid altering the shoe fit or comfort. Flexible FPC ribbon cables route signals from the insole to a small PCB module housed in a low-profile 3D-printed enclosure mounted near the shoe heel or tongue. The enclosure dimensions are approximately:

- Size: 50 mm x 50 mm
- Weight: 50 grams

The enclosure, as shown in figure 2.2 contains the ESP32 MCU incorporated on a PCB with analog front end. The enclosure also has a battery compartment to store the Li-Po battery. The enclosure exposes the USB-C port for charging the battery. Because the enclosure is 3D-printed, the design will be lightweight in construction. This ensures minimal interference with natural running movements while enabling reliable sensor data acquisition.

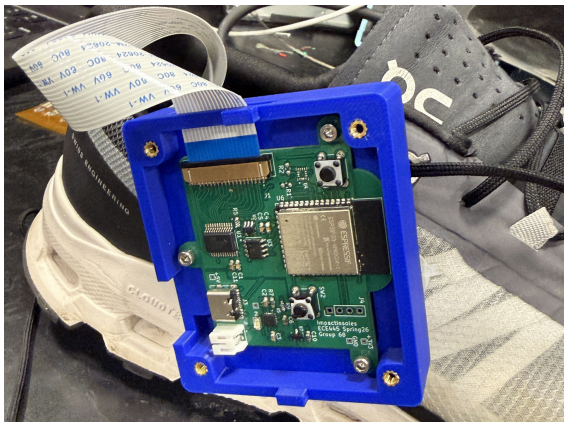

Figure 2.2: 3D-printed main PCB enclosure

b. Design Procedure and Details

i. Subsystem 1: Thin-Film Pressure Sensor Insole Array

This subsystem converts plantar loading during running into electrical signals that can be sampled and processed. Force-sensing resistors were selected because they are thin, flexible, require only a simple voltage-divider bias circuit, and are well-suited to the pressure ranges experienced during running.

The final design uses six Interlink FSR402 sensors at the heel and six FSR400 sensors at the midfoot and forefoot, reduced from the originally planned 16 to simplify flex PCB routing while maintaining sufficient spatial coverage to detect heel strike, mid-stance, and toe-off across all three plantar zones. Each FSR is wired in the voltage-divider configuration shown in Figure 2 with a bias resistor R_bias = 4.7 kΩ and V_cc = 3.3 V:

$$Vout = Vcc \bullet \frac{R_{bias}}{R_{FSR}(F) + R_{bias}} \quad (1)$$

The two insole sub-assemblies connect to the main PCB via an 18-pin Amphenol FPC connector. The total assembly is then laminated to a foam insole (Figure 2.1). Figure 2.3 shows flex PCB layout and final fabricated insole assembly incorporating the FSRs and connectors. The flex PCBs were designed this way to fit the ordering size constraints of the class.

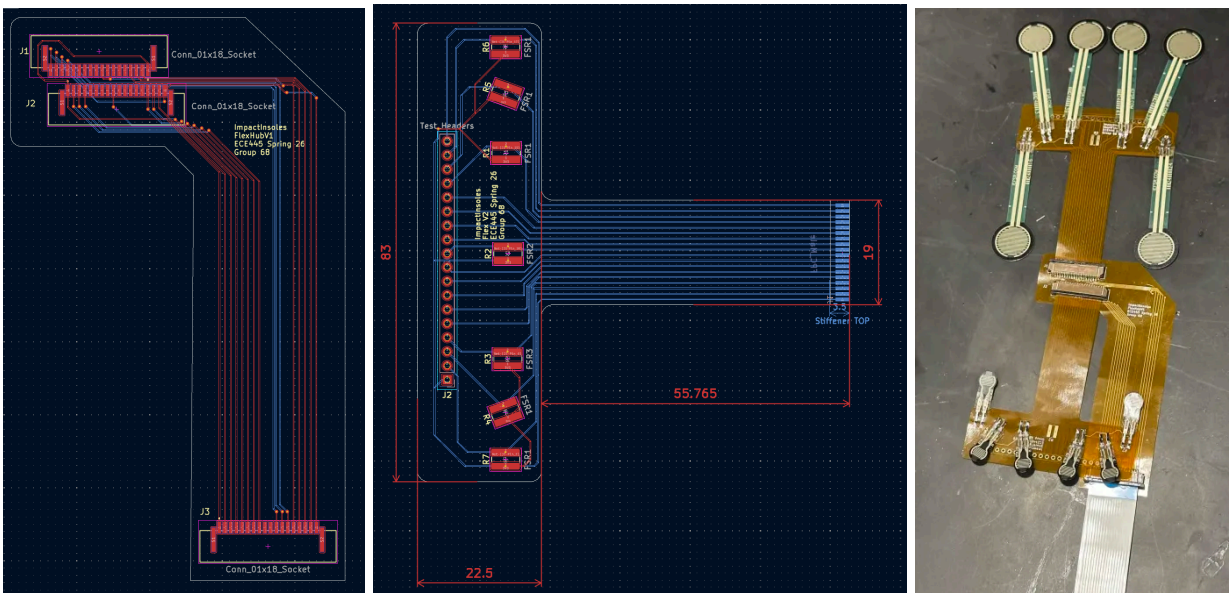

Figure 2.3: Flex PCB Layout and Final Assembly

ii. Subsystem 2: Analog Front-End + ADC Data Acquisition

This subsystem connects to the sensor insole and converts the analog FSR signals into digital data that the MCU can process. Because the ESP32-S3's internal ADC is nonlinear and temperature-sensitive, an external SPI ADC is used, with all 12 sensor channels time-multiplexed onto its single input via a CD74HC4067 16-channel analog multiplexer. The filtered signal then passes through an TLV9062IDR Operational Amplifier, which prevents ADC settling errors caused by the multiplexer on-resistance, before being digitized by an MCP3201 12-bit ADC. The complete schematic for this subsystem is shown in Figure 2.4.

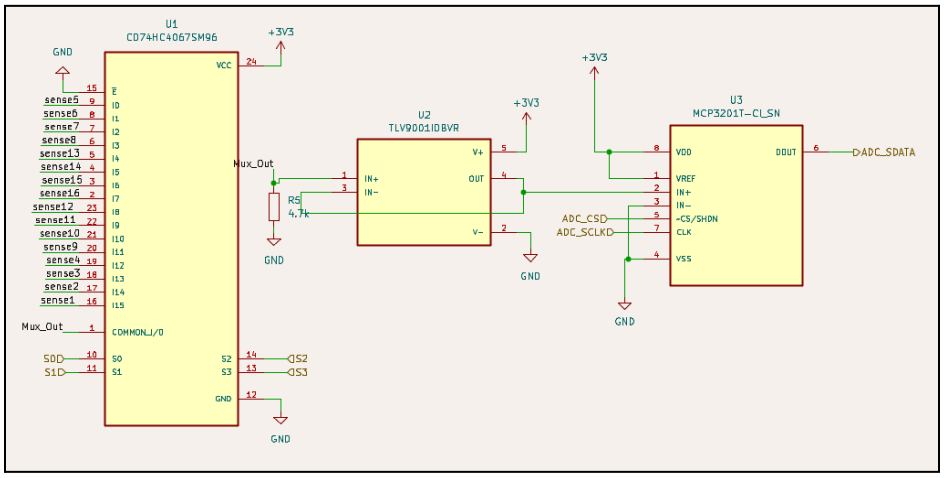

Figure 2.4: Analog Front-End Schematic

iii. Subsystem 3: Microcontroller + BLE Wireless Telemetry

This subsystem is responsible for performing real-time data acquisition and transmitting sensor data through Bluetooth to an external device such as an app on a phone. The ESP32-S3 microcontroller controls the analog acquisition chain by selecting sensor channels via the MUX select bits, triggering ADC conversions, and formatting the resulting data into BLE packets.

Channel selection is performed by writing the four-bit binary representation of the target channel index to the MUX select lines, followed by a 50 µs settling delay to allow the multiplexer output to stabilize before conversion:

```
C/C++
void selectMuxChannel(uint8_t ch) {
  digitalWrite(MUX_S0, (ch >> 0) & 0x01);
  digitalWrite(MUX_S1, (ch >> 1) & 0x01);
  digitalWrite(MUX_S2, (ch >> 2) & 0x01);
  digitalWrite(MUX_S3, (ch >> 3) & 0x01);
  delayMicroseconds(MUX_SETTLE_US);
}
```

Each ADC conversion is performed over SPI using the MCP3201 12-bit ADC. The MCP3201 outputs a null bit as the MSB of the first transferred byte, so the raw 16-bit result must be right-shifted by 1 and masked to extract the 12-bit ADC value:

```C/C++
uint16_t readMCP3201() {
  hspi.beginTransaction(SPISettings(1000000, MSBFIRST, SPI_MODE0));
  digitalWrite(ADC_CS, LOW);
  delayMicroseconds(1);
  uint8_t b1 = hspi.transfer(0x00);
  uint8_t b2 = hspi.transfer(0x00);
  digitalWrite(ADC_CS, HIGH);
  hspi.endTransaction();
  uint16_t raw = ((uint16_t)b1 << 8) | b2;
  return (raw >> 1) & 0x0FFF;
}
```

Each 28-byte notification packet contains twelve 16-bit ADC values packed big-endian into bytes 0–23, and a 32-bit sequence number in bytes 24–27 for packet-loss tracking. The packet is transmitted over BLE as a GATT notification whenever a client is connected:

```C/C++
// Pack 12-bit value into BLE buffer (big-endian)
inline void packSensor(uint8_t slotIndex, uint16_t value) {
  packetBuf[slotIndex * 2]     = (value >> 8) & 0xFF;
  packetBuf[slotIndex * 2 + 1] =  value       & 0xFF;
}

// Pack counter into bytes 24–27
packetBuf[24] = (packetCounter >> 24) & 0xFF;
packetBuf[25] = (packetCounter >> 16) & 0xFF;
packetBuf[26] = (packetCounter >>  8) & 0xFF;
packetBuf[27] =  packetCounter        & 0xFF;

// Notify if connected
if (deviceConnected && bleReady) {
  pCharacteristic->setValue(packetBuf, PACKET_SIZE);
  pCharacteristic->notify();
}
```

The microcontroller schematic is shown in Figure 2.5, showcasing the USB-C connection, reset and boot buttons for programming, GPIO assignments for controlling the select bits, as well as the pin assignments for the external ADC from the analog front-end system.

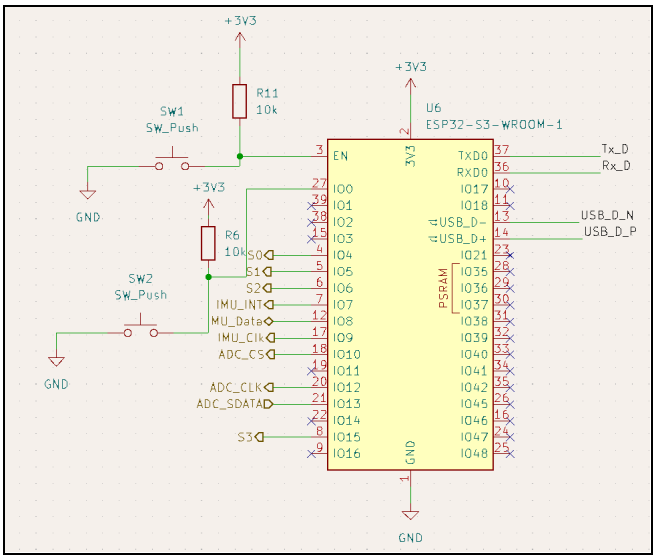

Figure 2.5: Microcontroller Schematic

iv. Subsystem 4: Power Management + Charging

This subsystem provides the board with power through the LiPo battery as well as being able to charge the LiPo battery. The main component of the system is BQ24075RGTR which is a battery management as well as charging regulation. There is a USB-C on the board as well as LiPo battery. The purpose of the USB-C is to charge the battery as well as program the board. BQ24075RGTR is able to control charging by using resistors as reference and has pins to confirm charging is operating properly. A linear MCP1700x-3300xT LDO was used to produce a stable 3.3 V rail for all active circuitry. When the USB-C is plugged in, the board will be powered through the USB-C and use the LiPo otherwise. The LiPo is a 3.7V 400 maH battery that should last the board operating for at least 1 hour. The schematic for this subsystem is shown in Figure 2.6 below.

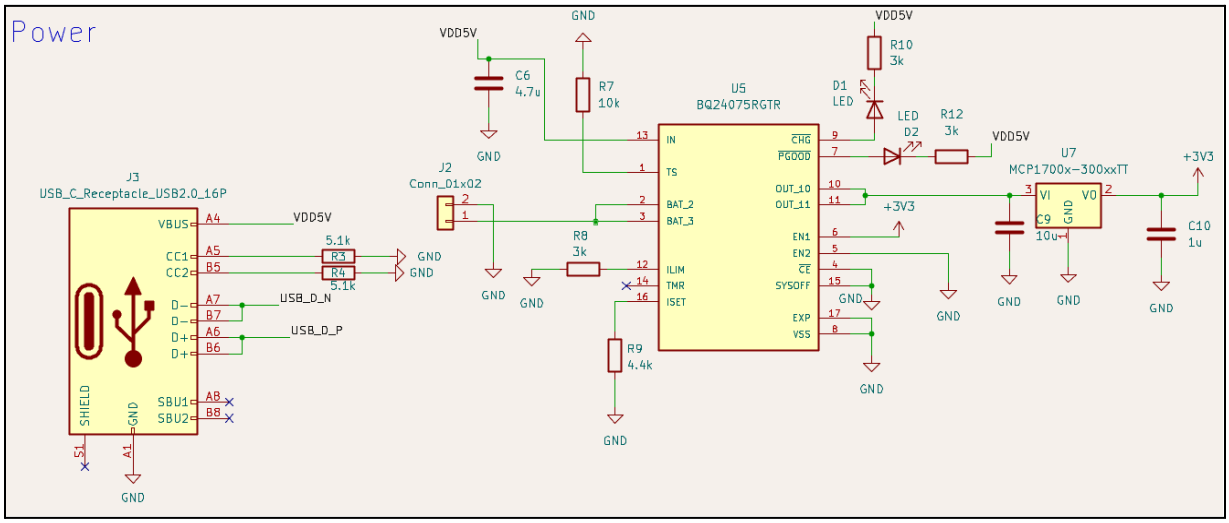

Figure 2.6: Power Circuit Schematic

v. Subsystem 5: Phone Interface / Data Visualization

This subsystem provides the wireless user-facing interface for receiving and displaying the pressure data streamed from subsystem 3. A browser-based web application was chosen over a native iOS/Android app to eliminate separate platform codebases and App Store distribution requirements, using the Web Bluetooth API available in Chrome and Edge to provide full BLE GATT client functionality from JavaScript. The app scans for the Impact Insoles BLE service, subscribes to the notify characteristic, and parses each incoming 28-byte packet into twelve ADC channel values and a sequence number. The pressure heatmap is rendered on an HTML5 Canvas by mapping each ADC value to a blue-to-red color gradient and blending neighboring values via radial gradients overlaid on a foot-shaped SVG silhouette. The dashboard also displays real-time cadence, step count, sample rate, and packet loss, and supports session recording with CSV export. Figure 2.7 shows screenshots of what the web app looks like.

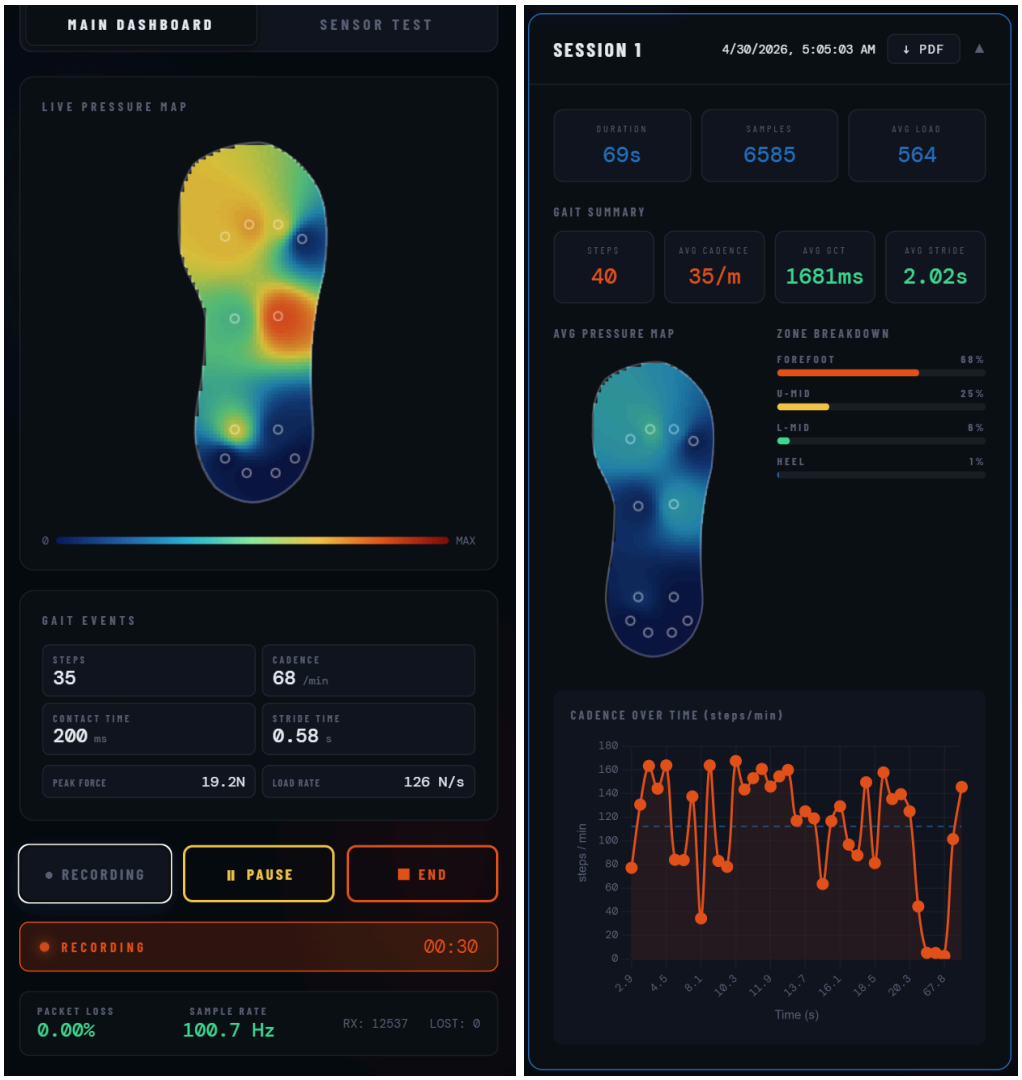

Figure 2.7: App dashboard during a run (Left) and Session Results (Right)

3. Verification and Results

All 12 sensors successfully transmitted data to the ADC under concurrent load conditions, with each channel producing non-zero, changing ADC readings when pressure was applied. Repeated load measurements across all sensors fell within software-correctable margins, confirming repeatable analog signal generation across the array. Figure 3.1 shows the measured resistance across all 12 sensor channels under a standardized load, with error bars indicating one standard deviation over five repeated trials. The distribution confirms consistent behavior between sensor types, with FSR400 sensors clustering around 5686 Ω (SD 463 Ω) and FSR402 sensors around 3497 Ω (SD 359 Ω).

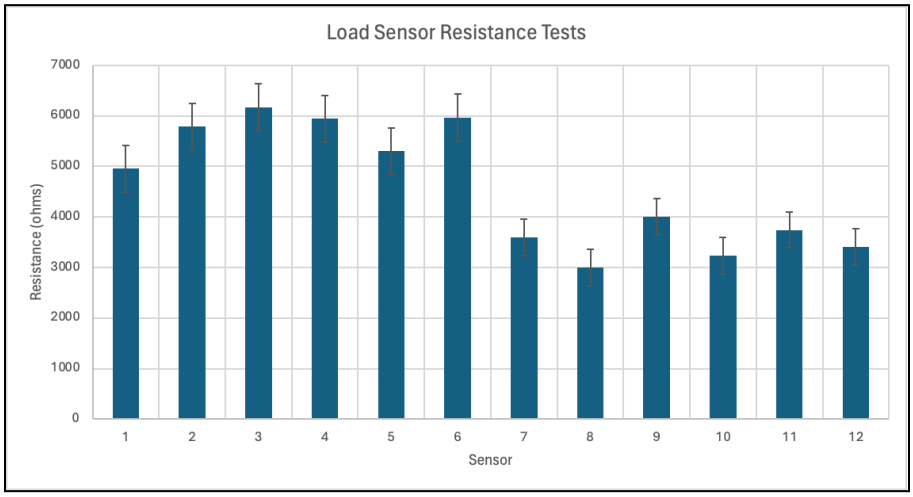

Figure 3.1: Load FSR Bar Chart

The insole assembly was also subjected to an artificial flex cycle test at 45°, repeated 100 times, with no mechanical failure and resistance drift remaining below the 20% threshold.

The analog front-end achieved a measured aggregate sample rate of ~100 Hz across all 12 channels MUX channel selection was verified by monitoring the select lines with an oscilloscope and confirming the correct 4-bit binary sequence for each channel. ADC readings increased with applied load across the full sensor pressure range, confirming that the 12-bit MCP3201 provides adequate resolution for gait-event detection.

The ESP32-S3 programmed successfully over USB-C using the standard USB-serial bootloader, and both the reset and boot buttons functioned as expected. BLE advertising was detected by both Android and iOS devices after powering the board.

Figure 3.2 shows the Pareto distribution of successive-sample ADC differences for Sensor 1 during a test trial, confirming that the vast majority of transitions are small and smooth, with no sudden jumps attributable to packet loss or signal noise.

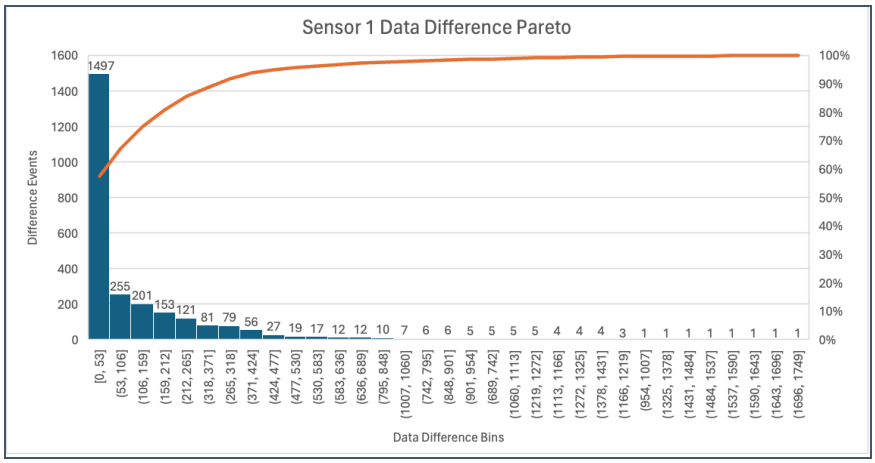

Figure 3.2: Pareto Distribution of Successive Samples for Sensor 1

The BQ24075RGTR charged the LiPo at a measured current of 203 mA, within the ±5 mA tolerance of the 200 mA target. The charge current was verified by measuring the voltage across R_ISET with a multimeter, shown in Figure 3.3, and applying the BQ24075RGTR charge current equation:

$$I_{CHG} = \frac{V_{ISET}}{R_{ISET}} \times 400 = \frac{2.236V}{4400\Omega} \times 400 = 203mA \quad (2)$$

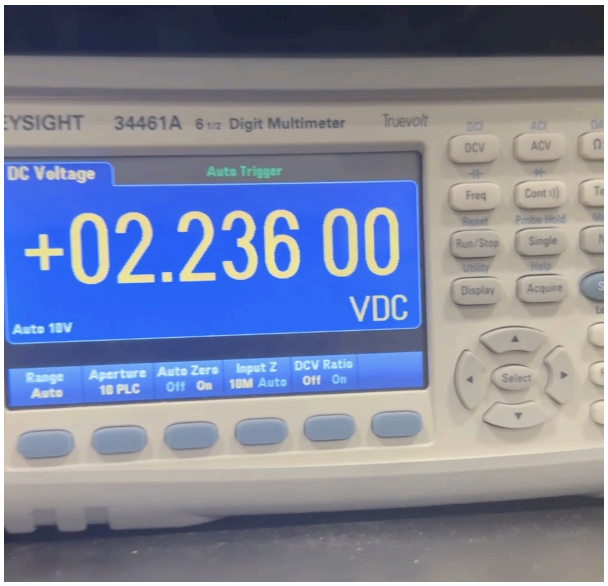

Figure 3.3: Multimeter Reading to Test Charging

The 1-hour battery life and operation requirement was verified by running the system continuously from a full charge while streaming live BLE data to the web application. Figure 3.4 shows the recorded session data from this test, confirming uninterrupted sensor acquisition and BLE transmission over the full duration with no resets, firmware crashes, or dropped packets observed.

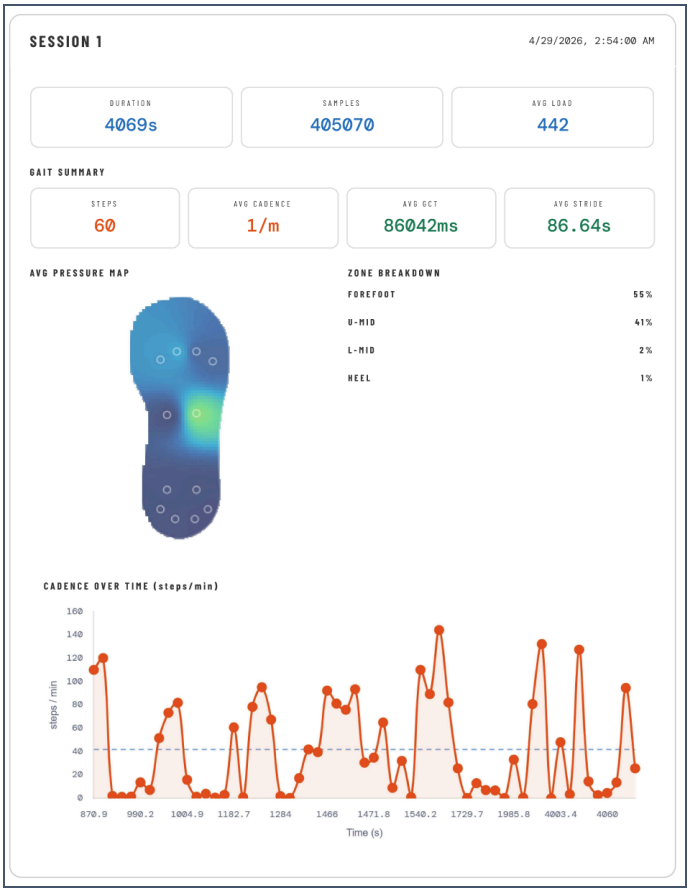

Figure 3.4: PDF from a 1-hour Stress Test

The remaining requirements were verified through a combination of visual inspection and real-time confirmation through the web application. Isolated sensor loading tests confirmed that the phone interface correctly decoded incoming BLE packets without channel corruption. This means pressing a single sensor while keeping all others untouched produced a response in only the corresponding heatmap region. The heatmap updated in real time with no perceptible lag or freezing during rapid tapping, satisfying the continuous display rate requirement.

Cadence accuracy was verified through a controlled walking trial in which a team member walked at a steady pace while a second member counted steps independently using a stopwatch. The cadence reported by the web application was confirmed to be within ±3 BPM of the manually counted reference, satisfying the accuracy requirement.

Finally, the 3.3 V rail stability requirement was verified by measuring the voltage at the 3.3 V testpoint with a multimeter during active BLE streaming, confirming the reading remained within ±0.1 V of nominal throughout the test. Taken together, these results confirm that all subsystem requirements were met, and the system performed as intended across all primary verification criteria. For the full RV tables, see Appendix Tables 1-5.

4. Cost and Schedule

a. Cost Analysis

The total cost to purchase the parts to create one device for one shoe before shipping is $58.35 and if not using parts from the ECE Eshop would be $62.60 (see Table 1 below). Shipping costs are an estimated $19.99 for all parts that are not sourced from the ECE Eshop can be found on Digikey. We can also assume Illinois state tax of 6.25%. This brings the total cost to create one ImpactInsole to be $128.67. We can expect a reasonable salary of $40/hour * 2.5 * 80 hours to complete =$8000 per team member. With our 3 members, this comes to $8000 * 3 = $24,000 in labor costs. Therefore the total costs of the project is $24,128.67.

Note: The cost of the initial creation of ImpactInsoles research and development, as well as the option to create multiple ImpactInsoles will increase this price. The current table is the cost for the build of one ImpactInsole.

Table 1: Bill of Materials

Part Description	Manufacturer	Retail Price per Part ($)	Qty	Retail ($)	Our Cost per Part ($)	Total ($)
FSR 402 Round Force Sensing Resistor	Interlink Electronics	2.99	6	17.94	2.99	17.94
FSR 402 Short Round Force Sensing Resistor	Interlink Electronics	2.99	6	17.94	2.99	17.94
BQ24075RGTR Battery Charger IC	Texas Instruments	2.36	1	2.36	2.36	2.36
MCP3201T-CI/SN 12-bit ADC	Microchip Technology	1.52	1	1.52	1.52	1.52
TLV9001IDBVR Op-Amp	Texas Instruments	0.4	1	0.4	0	0
SFW18R-1STE1LF 18-pin FPC Connector	Amphenol ICC	0.77	4	3.08	0.77	3.08
USB-C Receptacle (GT-USB-7010ASV)	G-Switch	0.52	1	0.52	0.52	0.52
MCP1700-300 LDO Regulator	Microchip Technology	0.51	1	0.51	0.51	0.51
CD74HC4067 16-Channel MUX	Texas Instruments	0.63	1	0.63	0.63	0.63
JST S2B-PH Battery Connector	JST	0.11	1	0.11	0.11	0.11
5-in 18-pin 1mm FFC Cable	Molex	1.14	1	1.14	1.14	1.14
Hot Glue (estimated per-unit share)	Generic	0.25	1	0.25	0.25	0.25
M2.5 Heat Inserts	Generic	0.1	4	0.4	0.1	0.4

ESP32-C3-WROOM Module	Espressif Systems	2.5	1	2.5	0	0
Capacitors 0603 (assorted)	Various	0.05	13	0.65	0	0
Resistors 0603 (assorted)	Various	0.1	5	0.5	0	0
LEDs (0805/1206)	Various	0.1	2	0.2	0	0
Push Button Switch	Generic	0.5	1	0.5	0	0
3.7V 420mAh LiPo Battery	Adafruit	6.95	1	6.95	6.95	6.95
PCB Fabrication	JLC	5	1	5	0	0
Subtotal				62.6		58.35
Illinois Tax (6.25%)				3.91		3.65
Estimated Shipping				19.99		19.99
Grand Total (1 device)				**86.50**		**81.98**

b. Mass Production Estimate

Due to the nature of our product, we believe it would be commercially viable which would change the estimated cost of the production of an ImpactInsole. For this mass production run, we will estimate 1,000 units. This allows for wholesale prices which will make parts such as the ESP32 or the PCB fabrication to be a lot cheaper than it currently is for the price of just one of these parts. Therefore, we estimate that there would be around 37% price reduction leading to the new price of manufacturing an ImpactInsole to be around $50 per unit.

c. Schedule

All members worked throughout the semester to complete the project on time. Table 2 shows the final timeline followed.

Table 2: Project Schedule

Week	Task
February 15 - February 21	**Aarush:** Research into flex PCB materials **Joseph:** Researched and decided frameworks + initialized project **Matthew:** Began hard PCB design
February 22 - February 28	**Everyone:** Design document work **Aarush:** Designed flex PCB v1 for R1 order **Joseph:** Verified designs and started sensor driver functions **Matthew:** Finished hard PCB v1 for R1 order

March 1 - March 7 *March 5:* *2nd PCB Order*	**Everyone:** Ordered parts for v1 PCBs **Aarush:** Revised flex PCB for R3 order **Joseph:** Finished sensor driver functions and other software for breadboard demo **Matthew:** Revised hard PCB for R3 order
March 8 - March 14 *March 9-March 11:* *Breadboard Demo* *March 12: 3rd PCB Order*	**Everyone:** Assembled hardware for breadboard demo **Aarush:** Tested FSR sensor ranges and FPC fit to determine optimal range for final product, redesigned flex PCB for R3 **Joseph:** Finished sensor driver functions, BLE communication, and web front-end for breadboard demo **Matthew:** Probed basic functionality of R1 hard PCB, revised hard PCB design for R3
March 15 - March 21	*Spring Break* **Everyone:** Looked over drivers and design implementation
March 22 - March 28 *March 26:* *Final PCB Order*	**Everyone:** Created mid-project roadmap to guide further progress **Aarush:** Designed FlexHub PCB to bridge sensor flex PCBs, redesigned flex PCB for final round **Joseph:** Added to web UI and adapted it to a PWA for phone **Matthew:** Redesigned hard PCB for final round (mounting points and packaging modifications)
March 29 - April 4	**Everyone:** First joint test of all modules together (hard PCB, flex PCB, software) **Aarush:** Characterized sensor joint and pad durability with different connection methods and solder-spread patterns **Joseph:** Debugged ESP32 issues on board, **Matthew:** Completed soldering of R1 hard PCB, helped debug ESP32 issues on board
April 5 - April 11 *April 6-April 8: Progress Demo*	**Everyone:** Revised working joint-module system for progress demo **Aarush:** Tested flex PCB R3, verified angles and dimensions **Joseph:** Revised sensor and BLE firmware to meet accuracy and efficiency requirements **Matthew:** Tested new ADC/Op-amp circuits on R3 hard PCB
April 12 - April 18	**Aarush:** Designed, printed, and tested CAD for hard PCB enclosure **Joseph:** Finalized web app UI to display full dashboard of foot data **Matthew:** Redesigned hard PCB for self-order due to pinout

	mismatch
April 19 - April 25 *April 20-April 24: Mock Demo* *April 23/April 24: Mock Presentation*	**Everyone:** Put together progress demo, final project integration and testing **Aarush:** Soldered sensors to flex PCBs and connectors to FlexHub, tested total sensor array **Joseph:** Stress tested web app data and BLE connectivity with final version of Impact Insoles **Matthew:** Assembled and tested final PCB
April 26 - May 2	**Everyone:** Integrated final project, created final presentation **Aarush:** Laminated sensor array to insole (final packaging), revised and implemented final enclosure CAD **Joseph:** Stress tested web app data and BLE connectivity with final version of Impact Insoles **Matthew:** Characterized BMS and ADC hardware functionality, helped debug cross-subsystem problems

5. Conclusion

a. Accomplishments and Uncertainties

Impact Insoles successfully demonstrated a fully integrated wearable plantar pressure sensing system at consumer-grade cost. The completed system incorporates 12 force-sensitive resistors distributed across the heel, midfoot, and forefoot zones of a flexible PCB insole. The ESP32-S3 microcontroller samples all 12 channels through a CD74HC4067 multiplexer at 100Hz and streams the data as BLE packets to a browser-based dashboard with around 0.5% measured packet loss. The web application renders a live color heatmap of plantar pressure, detects foot-strike events, and computes running cadence in real time with session recording available to the user. The power subsystem maintained a stable 3.3 V rail throughout testing and sustained over one hour of continuous BLE streaming on a single 420 mAh charge. The battery is able to be recharged as well making the product commercially viable.

Looking at the final product as well as the requirements set out at the beginning, there were no unsatisfactory results. Every verification procedure yielded results within expectations, and the final product had full functionality, as mentioned in Section 3.

b. Ethics and Safety

The main source of ethical consideration in this project is the potential misunderstanding of this system as a medical device. Since the system tracks biometrics and may offer interpretation of that data in the form of statistics or algorithmically-calculated suggestions, users could assume it can assess clinical injury or provide medical recommendations. In accordance with the IEEE Code of Ethics, which emphasizes holding paramount the safety, health and welfare of the

public, we will need to clearly present the device as a training and performance aid only. This means we may also have to include disclaimers describing the product's limitations, as well as advising users to make their own personal medical judgment with the data.

The IEEE Code of Ethics also requires engineers to be honest and realistic in their claims based on the available data. Due to this, the reported metrics from our device will have to be validated against reference measurements. Safety is also a top priority, as an electronic device will be attached to the foot, a crucial part of the human body. The system must remain lightweight, securely mounted, and low-profile to not interfere with normal movement or create a tripping or discomfort hazard. Since the device will operate with a battery, a proper enclosure must be created to prevent overheating and electrical risks during use.

The societal impact of this system is not necessarily widespread, but can still be significant. This product appeals to a niche market of those who care about improving their running stride and maintaining good form. This system can provide a more accessible option for users outside of expensive laboratory environments.The positive effects compound with more miles run, so avid runners in particular can benefit greatly from this.

c. Broader Impact and Future Extensions

Economically, the platform lowers the barrier to entry for coaches and physical therapists who could use real-world pressure data to inform training decisions without requiring clinic infrastructure. There are several future extensions that could expand Impact Insoles to be a more sound product. This includes bilateral sensing, or creating a pair of Impact Insoles for both feet in order to gather difference data between each foot. The web app could incorporate AI-assisted gait coaching, where we could feed the AI the session data to give the user a better understanding of what the data means and how they could improve their foot striking. Lastly, clinical validation with professionals and athletes could further expand the platform's impact well beyond the consumer market, and give it more credibility as a real, impactful product.

6. Citations

[1] National Center for Biotechnology Information (NCBI), "Article available at PMC8500811," PubMed Central (PMC). Available at: https://pmc.ncbi.nlm.nih.gov/articles/PMC8500811/. Accessed Feb. 27, 2026.

[2] ScienceDirect, "Article available at S2950273X24000729," ScienceDirect. Available at: https://www.sciencedirect.com/science/article/pii/S2950273X24000729. Accessed Feb. 27, 2026.

[3] National Center for Biotechnology Information (NCBI), "Article available at PMC7734358," PubMed Central (PMC). Available at: https://pmc.ncbi.nlm.nih.gov/articles/PMC7734358/. Accessed Feb. 27, 2026.

[4] V. Ramachandran et al., "Wearable Plantar Pressure Sensing for Gait Analysis," in Advances in Intelligent Systems, Springer, 2025, ch. 18. Available at: https://link.springer.com/chapter/10.1007/978-981-95-3141-7_18/figures/1. Accessed May 6, 2026.

[5] Interlink Electronics, "FSR 402 Force Sensing Resistor Datasheet," datasheet, 2010. Available at: https://cdn.sparkfun.com/assets/8/a/1/2/0/2010-10-26-DataSheet-FSR402-Layout2.pdf. Accessed May 6, 2026.

[6] Interlink Electronics, "FSR 400 Force Sensing Resistor Datasheet," datasheet, 2010. Available at: https://cdn.sparkfun.com/datasheets/Sensors/ForceFlex/2010-10-26-DataSheet-FSR400-Layout2.pdf. Accessed May 6, 2026.

[7] Interlink Electronics, "FSR 402 Round Force Sensing Resistor," product highlight page. Available at: https://www.digikey.com/en/product-highlight/i/interlink/fsr-402-round-force-sensing-resistor. Accessed Feb. 27, 2026.

[8] Molex LLC, "Premo-Flex FFC/FPC Jumper Cables," product category page. Available at: https://www.mouser.com/c/wire-cable/ffc-fpc-jumper-cables/?m=Molex&tradename=Premo-Flex. Accessed Feb. 27, 2026.

[9] Molex LLC, "5034800400 FFC/FPC Connector," product page. Available at: https://www.molex.com/en-us/products/part-detail/5034800400. Accessed Feb. 27, 2026.

[10] Texas Instruments, "CD74HC4067 High-Speed CMOS Analog Multiplexer/Demultiplexer," datasheet, 2017. Available at: https://www.ti.com/lit/ds/symlink/cd74hc4067.pdf. Accessed Feb. 27, 2026.

[11] Microchip Technology, "MCP3201 2.7V 12-Bit A/D Converter with SPI Serial Interface," datasheet. Available at: https://ww1.microchip.com/downloads/en/DeviceDoc/21290F.pdf. Accessed May 6, 2026.

[12] Texas Instruments, "TLV9001 1.8-V to 5.5-V, 1-MHz, Low-Power Operational Amplifier," datasheet. Available at: https://www.ti.com/lit/ds/symlink/tlv9001.pdf. Accessed May 6, 2026.

[13] Espressif Systems, "ESP32-S3-WROOM-1 & WROOM-1U Datasheet," datasheet, 2022. Available at: https://documentation.espressif.com/esp32-s3-wroom-1_wroom-1u_datasheet_en.pdf. Accessed May 6, 2026.

[14] Mischianti, "ESP32-S3 DevKitC-1 High Resolution Pinout and Specs," reference page. Available at: https://mischianti.org/esp32-s3-devkitc-1-high-resolution-pinout-and-specs/. Accessed May 6, 2026.

[15] Fritzing Forum, "ESP32-S3 HiLetGo Dev Board with Pinout Template," community reference. Available at: https://forum.fritzing.org/t/esp32s-hiletgo-dev-boad-with-pinout-template/5357. Accessed May 6, 2026.

[16] Texas Instruments, "BQ24075 USB-Friendly Li-Ion Battery Charger and Power-Path Management IC," datasheet. Available at: https://www.ti.com/lit/ds/symlink/bq24075.pdf. Accessed Feb. 27, 2026.

[17] Microchip Technology, "MCP1700 Low Quiescent Current LDO Regulator," datasheet. Available at: https://ww1.microchip.com/downloads/en/DeviceDoc/MCP1700-Low-Quiescent-Current-LDO-Regulator-DS20001826E.pdf. Accessed Feb. 27, 2026.

[18] Institute of Electrical and Electronics Engineers (IEEE), "IEEE Code of Ethics," IEEE Board of Directors, June 2020. Available at: https://www.ieee.org/about/corporate/governance/p7-8.html. Accessed Feb. 27, 2026.

7. Appendix

The following tables provide the complete requirement and verification details for all of our subsystems:

a. Table 1: FSR Sensor Array RV Table

Requirements	Verification
• The subsystem will incorporate 8-16 pressure sensors, distributed in the heel, midfoot and forefoot regions.	• Visually inspect the fabricated insole and count total sensors. • Confirm region placement by comparing sensor locations to a foot outline template.
• Each sensor should have a resistance of ~500 ohms when subjected to a static normal force of 10 N ± 0.5 N under nominal bias conditions.	• Place the insole on a rigid flat surface and measure unloaded sensor output voltage V_0 using a DMM. • Apply a calibrated 10 N load centered on the sensor and measure loaded voltage. • Compute $\Delta V = \lvert V_{10N} - V_0 \rvert$ and calculate $R = \frac{V}{I}$ • Repeat this process for all of the utilized sensors.
• Each sensor's tongue-to-insole electrical joint should withstand hundreds of bends of up to 45 degrees from normal, and hold up in real-world conditions	• Place the insole on a rigid flat surface and measure unloaded sensor output voltage V_0 using a DMM. • Flex the insole at the targeted joint by 45 degrees 100 times • Repeat this process for all of the utilized sensors. Inspect joint quality. • Use insoles immediately afterward on a real-world 2 hour run test (2x battery spec). Inspect joint quality.

b. Table 2: Analog Front End RV Table

Requirements	Verification
• This subsystem will be able to sample all 16 sensors at 100Hz	• Run test sampling program on MCU at 100 Hz for 10 seconds. Repeatedly stress all sensors concurrently at high frequency. Can have two people do this with fingers. • Compute deltas between each sample and plot for each sensor. Ensure there is no significant weight at the end of the distribution compared to the rest.
• The subsystem will have enough resolution to track gradual pressure shifts	• Run test sampling program on MCU at 100 Hz for 5 seconds with sensor-insole

across sensors	visualization. Start standing on balls of foot. ● Over 5 seconds, smoothly transition all weight to the heel. ● Visually check that pressure transition is tracked by the sensors. Compute sum of squared differences and ensure values are close to zero.

c. Table 3: MCU and Bluetooth RV Table

Requirements	**Verification**
● The subsystem will acquire sensor data at an effective per-channel sampling rate of ≥100 Hz for the configured number of active channels.	● Enable timestamp logging in firmware for each sample event. ● Collect timestamped data for at least 30 seconds during continuous operation. ● Compute the time difference between consecutive samples of the same channel. Calculate effective per-channel sampling rate. ● Verify minimum observed rate ≥100 Hz across all channels.
● BLE Packet Delivery Requirement. During continuous streaming at full sampling rate, the system shall achieve ≥95% packet delivery at 5 m line-of-sight.	● Embed an incrementing sequence number in each BLE packet. ● Stream data continuously for 5 minutes at full sampling rate. Record the received sequence numbers on the phone (e.g., via nRF Connect log). ● Compute delivery ratio = (received packets / transmitted packets) and verify delivery ≥95%.
● The MCU will be able to sustain simultaneous sampling and BLE transmission for ≥1 hour without reset, buffer overflow, or data stall.	● Fully charge battery and configure system at maximum sampling rate, then begin continuous BLE streaming with live reception. ● Allow the system to run for 1 hour uninterrupted and monitor for resets, dropped streaming, or firmware crashes ● Verify uninterrupted operation and continued packet reception at end of test

d. Table 4: Power Management RV Table

Requirements	Verification
• The power subsystem will have the ability to charge battery at 200mA	• Connect USB-C power cable to USB-C multimeter, and multimeter to charging port. • Commence charging and ensure 200 ± 5 mA charging rate throughout duration
• The subsystem should be able to supply ICs and MCU with stable 3V3 and 5V	• Run the main program for 5 minutes. Measure voltage at 3V3 and 5 V testpoints. • Ensure 3*SD is less than 1% of each.
• The Battery life meets desired specification (1 hour minimum)	• Time the main program until shutoff. Ensure shutoff does not occur before 1 hour.

e. Table 5: Web Application RV Table

Requirements	Verification
• The phone interface will correctly decode the incoming BLE packets into individual sensor channels without channel corruption.	• Apply pressure to only one of the sensors while keeping all others untouched. Observe live data display and confirm only the corresponding channel changes significantly. • Repeat tests for all the different sensor locations. Verify no unintended channels respond during isolated loading.
• The phone interface shall update displayed sensor values or plots at a rate sufficient to appear continuous during active streaming.	• Begin live streaming session. Rapidly tap and release pressure on a sensor at ~2–3 Hz. • Observe visualization and confirm pressure peaks appear in real time without noticeable lag or freezing.
• The phone interface will compute cadence from the pressure-derived foot-strike events within ±3 BPM compared to a reference measurement.	• Conduct a controlled short 3–5 minute running test that measures the cadence independently using a stopwatch or smartwatch. Record feet with 120 fps video camera. • Process foot-strike events and compute cadence over similar time intervals manually based on camera data • Compare sensor-computed cadence to the referenced control test and verify absolute error ≤3 BPM in each test window.