UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

MATHEUS HENRIQUE ORSINI DA SILVA

MIGUEL ESCORSIN FERREIRA DA SILVA

DESENVOLVIMENTO DE UM SISTEMA EMBARCADO VESTÍVEL PARA ANÁLISE DA PRESSÃO PLANTAR

CURITIBA

2026

versão 11.0 (abr. 25)

MATHEUS HENRIQUE ORSINI DA SILVA
MIGUEL ESCORSIN FERREIRA DA SILVA

DESENVOLVIMENTO DE UM SISTEMA EMBARCADO VESTÍVEL PARA ANÁLISE DA PRESSÃO PLANTAR

Wearable embedded system development for plantar pressure analysis

Trabalho de conclusão de curso de graduação apresentado como requisito para obtenção do título de Bacharel em Engenharia Elétrica do curso de Engenharia Elétrica e de Bacharel em Engenharia Mecatrônica do curso de Engenharia Mecatrônica da Universidade Tecnológica Federal do Paraná (UTFPR).
Orientador: Prof. Dr. Narco Afonso Ravazzoli Maciejewski.
Coorientador: Prof. Dr. José Jair Alves Mendes Júnior.

CURITIBA

2026

MATHEUS HENRIQUE ORSINI DA SILVA
MIGUEL ESCORSIN FERREIRA DA SILVA

DESENVOLVIMENTO DE UM SISTEMA EMBARCADO VESTÍVEL PARA ANÁLISE DA PRESSÃO PLANTAR

Trabalho de conclusão de curso de graduação apresentado como requisito para obtenção do título de Bacharel em Engenharia Elétrica do curso de Engenharia Elétrica e de Bacharel em Engenharia Mecatrônica do curso de Engenharia Mecatrônica da Universidade Tecnológica Federal do Paraná (UTFPR).

Data de aprovação: 18 / junho / 2026

Narco Afonso Ravazzoli Maciejewski
Doutorado
Universidade Tecnológica Federal do Paraná

Renata Coelho Borges
Doutorado
Universidade Tecnológica Federal do Paraná

Juan Camilo Castellanos Rodriguez
Doutorado
Universidade Tecnológica Federal do Paraná

CURITIBA
2026

AGRADECIMENTOS

Às nossas famílias, expressamos nossa gratidão pelo apoio e pela paciência ao longo de toda a jornada da graduação. Ao nosso orientador, Prof. Dr. Narco Afonso Ravazzoli Maciejewski, e ao coorientador, Prof. Dr. José Jair Alves Mendes Júnior, agradecemos pela orientação, disponibilidade e contribuições durante o desenvolvimento do trabalho. À Profa. Dra. Leandra Ulbricht, responsável pelo laboratório de ergonomia (LAERG), agradecemos por disponibilizar o seu aparelho EPS+R para a coleta de dados. Ao Gabriel Mellem e ao Jonathan Paulista, agradecemos pela disponibilização dos protótipos em impressão 3D. Por fim, agradecemos a todos que, direta ou indiretamente, contribuíram para a realização deste trabalho.

RESUMO

A análise da pressão plantar é uma ferramenta utilizada para a avaliação biomecânica da marcha, o acompanhamento de condições ortopédicas e a prevenção de lesões. Nesse contexto, foi desenvolvido um sistema embarcado vestível para análise de pressão plantar, integrando aquisição, transmissão, armazenamento e visualização de dados por meio de mapas de calor. A metodologia adotada contemplou etapas informacional, conceitual, prova de conceito e desenvolvimento do protótipo final. Na etapa informacional, realizou-se a análise de patentes nacionais e internacionais, o levantamento de produtos comerciais e a aplicação de um questionário a usuários potenciais, obtendo-se 37 respostas válidas, o que permitiu a definição de requisitos técnicos por meio do desdobramento da função qualidade. A etapa conceitual envolveu a definição de componentes dentro de um conjunto de possibilidades estipuladas para atender aos requisitos levantados na etapa anterior, definindo-se, dessa forma, as especificações de sensores, multiplexador, conversor analógico para digital, microcontrolador e módulos adicionais de comunicação e manipulação de dados. Uma vez que todos os requisitos e componentes foram definidos, realizou-se o desenvolvimento do protótipo, incluindo a montagem do circuito eletrônico, a implementação do software embarcado, o desenvolvimento do aplicativo móvel, o processo de calibração e o software de visualização. O sistema embarcado foi estruturado em tarefas de tempo real para leitura dos 16 sensores resistivos da palmilha, armazenamento dos dados, envio por comunicação sem fio e controle por aplicativo. A aquisição dos sensores foi configurada para operar em até 100 leituras por segundo, permitindo modos de aquisição contínua e temporizada. O processo de calibração foi realizado por comparação com o baropodômetro comercial EPS+R, sendo avaliados o modelo linear e o modelo polinomial para relacionar a condutância dos sensores à pressão medida. O modelo polinomial apresentou melhor ajuste na comparação com o equipamento de referência, com coeficiente de determinação de 0,937, adotando-se o polinômio de segundo grau por apresentar equilíbrio entre qualidade de ajuste e custo computacional. Além disso, foi desenvolvido um software para visualização de dados, utilizando interpolação para a geração de mapas de calor dinâmicos em uma interface gráfica, com possibilidade de análise temporal quadro a quadro, animação contínua e visualização em tempo real. Os resultados obtidos demonstraram a viabilidade funcional do sistema, integrando aquisição, armazenamento, transmissão e visualização da pressão plantar, embora ainda sejam necessários aprimoramentos relacionados à placa de circuito impresso, ao processo de calibração e à validação em condições reais de uso.

Palavras-chave: palmilha sensorizada; baropodometria; sensores resistivos; mapas de calor.

ABSTRACT

Plantar pressure analysis is a tool used for the biomechanical assessment of gait, the monitoring of orthopedic conditions, and injury prevention. In this context, a wearable embedded system for plantar pressure analysis was developed, integrating data acquisition, transmission, storage, and visualization through heat maps. The methodology adopted included informational, conceptual, proof-of-concept, and final prototype development stages. In the informational stage, national and international patents were analyzed, a survey of commercial products was conducted, and a questionnaire was applied to potential users, obtaining 37 valid responses, which allowed the definition of technical requirements through quality function deployment. The conceptual stage involved the definition of components from a set of possible alternatives established to meet the requirements identified in the previous stage, thus defining the specifications of sensors, multiplexers, analog to digital converters, microcontrollers, and additional communication and data-handling modules. Once all requirements and components had been defined, the prototype was developed, including the assembly of the electronic circuit, the implementation of the embedded software, the development of the mobile application, the calibration process, and the visualization software. The embedded system was structured using real-time tasks for reading the 16 resistive sensors of the insole, storing data, transmitting it through wireless communication, and controlling the system via an application. Sensor acquisition was configured to operate at up to 100 readings per second, allowing continuous and timed acquisition modes. The calibration process was carried out by comparison with the commercial EPS+R baropodometer, evaluating both linear and polynomial models to relate sensor conductance to the measured pressure. The polynomial model showed a better fit compared to the reference equipment, with a coefficient of determination of 0.937, adopting the second-degree polynomial because it presented a balance between quality of fit and computational cost. In addition, software was developed for data visualization, using interpolation to generate dynamic heat maps in a graphical interface, with the possibility of frame-by-frame temporal analysis, continuous animation, and real-time visualization. The results demonstrated the functional feasibility of the system, integrating acquisition, storage, transmission, and visualization of plantar pressure, although further improvements are still required regarding the printed circuit board, the calibration process, and validation under real-use conditions.

Keywords: sensorized insole; baropodometry; resistive sensors; heat maps.

LISTA DE FIGURAS

LISTA DE GRÁFICOS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

ADC	Conversor Analógico-Digital, do inglês *Analogic to Digital Converter*
BLE	Bluetooth de Baixa Energia, do inglês *Bluetooth Low Energy*
MMQ	Método dos Mínimos Quadrados
MQTT	Transporte de Telemetria por Enfileiramento de Mensagens, do inglês *Message Queuing Telemetry Transport*
QoS	Qualidade de Serviço, do inglês *Quality of Service*
RTOS	Sistemas Operacionais de Tempo Real, do inglês *Real-Time Operating Systems*
SPI	Interface Serial Periférica, do inglês *Serial Peripheral Interface*
USPTO	Escritório de Patentes e Marcas dos Estados Unidos, do inglês *United States Patent and Trademark Office*
UDP	Protocolo de Datagrama do Usuário, do inglês *User Datagram Protocol*

Acrônimos

INPI	Instituto Nacional da Propriedade Industrial
GUM	Guia para a Expressão da Incerteza de Medição, do inglês *Guide to the expression of Uncertainty in Measurement*

LISTA DE SÍMBOLOS

g	Grama
kg	Quilograma
Hz	Hertz
mm	Milímetros
V	Volts
mV	Milivolts
N	Newtons
kPa	Quilopascal

SUMÁRIO

1 INTRODUÇÃO

A pressão plantar pode ser definida como a distribuição da força exercida pela superfície do pé durante o apoio, seja em condição estática ou dinâmica. Essa permite avaliar como as cargas são distribuídas entre as regiões do pé, fornecendo informações biomecânicas. Como indicado pela revisão da literatura de Carvalho, Berro e Fiorelli (2021), a análise da pressão plantar é um recurso empregado em diversas aplicações, tais como: a compreensão de patologias ortopédicas; a avaliação da efetividade de intervenções cirúrgicas; e o monitoramento durante o tratamento com o uso de palmilhas posturais. Alguns trabalhos apresentados discutem que a redução da pressão plantar máxima, obtida pela utilização de tipos diferentes de palmilhas, pode ocasionar redução de dores na região do pé e da lombar (Almeida *et al.*, 2009). Além disso, a alteração na distribuição da pressão plantar, devido ao uso de palmilhas personalizadas, pode causar impactos na estabilidade do usuário (Seger, 2017). Ademais, no contexto esportivo, a análise da pressão plantar auxilia na obtenção de informações sobre equilíbrio e estabilidade postural, servindo como uma ferramenta para acompanhar a evolução de treinos proprioceptivos (Baldaço *et al.*, 2010).

Visando englobar os diversos cenários em que se exige a análise da pressão plantar, torna-se necessária a aplicação de sistemas de medição, os quais podem ser classificados em sistemas de plataforma fixa e sistemas *in-sole* (Macwilliams; Armstrong, 2000). Tradicionalmente, a medição da pressão plantar é realizada por meio de plataformas rígidas, denominadas baropodômetros, nas quais o usuário pode ficar em posição estática ou realizar uma caminhada, dependendo do tamanho da plataforma. No entanto, para pacientes com passada irregular, essas plataformas podem gerar dificuldades e desconforto, visto que exigem uma pisada normal.

Quanto aos sistemas internos, sensores do tipo filme flexível são inseridos no interior do calçado ou colados na sola do pé do usuário, realizando a aquisição contínua de dados em diferentes condições. No início do século XXI, observa-se uma tendência no desenvolvimento de dispositivos vestíveis. Dessa forma, busca-se proporcionar um entendimento mais aprofundado dos efeitos das modificações estruturais nos calçados sobre a biomecânica do pé, uma vez que a coleta em tempo real de situações cotidianas, como caminhar ou subir escadas, possibilita uma análise mais completa dos parâmetros de modificação (Cavanagh; Hewitt Jr.; Perry, 1992).

Portanto, diante da relevância das análises biomecânicas dos pés, este trabalho propõe o desenvolvimento do protótipo de um sistema embarcado vestível para análise da pressão plantar capaz de captar e processar informações relacionadas à pressão plantar em diversas condições e integrá-las com uma interface gráfica de mapa de calor. Dessa forma, busca-se disponibilizar uma ferramenta para auxiliar tanto profissionais de saúde como atletas no acompanhamento de tratamentos, prevenção de lesões e verificação da melhora postural.

1.1 Problema

Dispositivos de medição da pressão plantar devem ser otimizados para a aplicação específica, garantindo a veracidade e acurácia das leituras em tempo real, além de serem finos, flexíveis e leves para não influenciar a coleta de dados nem causar desconforto ao usuário (Razak *et al.*, 2012). Os baropodômetros não representam adequadamente a caminhada natural, exigindo que o usuário caminhe sobre uma área limitada e que gera um movimento artificial e dificulta a obtenção de dados confiáveis.

Portanto, o problema abordado neste trabalho é o desenvolvimento de um sistema embarcado vestível, capaz de garantir a veracidade e a acurácia das medições da pressão plantar em tempo real, com envio simultâneo de dados para uma interface gráfica.

1.2 Objetivos

Nesta seção são apresentados o objetivo geral e os objetivos específicos do protótipo relativos ao problema apresentado na seção 1.1.

1.2.1 Objetivo geral

Desenvolver um dispositivo vestível de tempo real, destinado à aquisição, transmissão e armazenamento de sinais da pressão plantar, integrado a uma interface gráfica para análise dos dados.

1.2.2 Objetivos específicos

Para se alcançar o objetivo geral proposto, os seguintes objetivos específicos foram definidos:

a) definir a arquitetura eletrônica do sistema com base nos requisitos obtidos pela metodologia de desenvolvimento de produto;

b) desenvolver um algoritmo de leitura baseado em conceitos de sistemas operacionais de tempo real;

c) desenvolver um protótipo funcional capaz de integrar a leitura dos sensores resistivos, o armazenamento local dos dados, a comunicação sem fio e o controle por aplicativo móvel;

d) implementar um sistema de visualização gráfica da pressão plantar utilizando mapa de calor; e

e) gerar a curva de calibração por meio de ensaios instrumentais com cargas controladas e comparação com equipamento comercial de baropodometria.

1.3 Justificativa

Conforme Codinhoto (2020), o número de pesquisas sobre baropodômetros apresentou um crescimento significativo na última década, atingindo 1700 publicações em 2019, um acréscimo de 1200 publicações em comparação com 2010, já no que se refere a palmilhas sensorizadas vestíveis, entre o período de 2000 e 2023 foram publicados 759 artigos (Abdollahi; Zhou; Yuan, 2024), evidenciando a importância do conjunto para análise biomecânica. No entanto, as soluções comerciais disponíveis são difíceis de utilizar devido a fatores de disponibilidade e custo, limitando seu uso.

Com o desenvolvimento desse trabalho, pretende-se preencher a lacuna existente quanto à disponibilidade de uma ferramenta nacional de baixo custo comercial adequada à realidade brasileira para auxiliar tanto profissionais de saúde como atletas no acompanhamento de tratamentos, prevenção de lesões e verificação da melhora postural. Dessa forma, busca-se democratizar o acesso à avaliação biomecânica ao reduzir a dependência de propriedades intelectuais e suporte técnico estrangeiro.

1.4 Estrutura do trabalho

Este documento possui em sua totalidade cinco capítulos, descritos a seguir:

a) Capítulo 1: apresenta a contextualização, o problema de engenharia enfrentado, os objetivos e a justificativa para o desenvolvimento deste trabalho;

b) Capítulo 2: desenvolve-se a fundamentação teórica necessária para a realização do trabalho, abordando temas como a anatomia do pé humano, sistemas operacionais de tempo real, protocolo de comunicação e metrologia;

c) Capítulo 3: apresenta a metodologia utilizada para o desenvolvimento de um protótipo, incluindo o levantamento de dados do cliente transformando-os em requisitos técnicos, realização da seleção de componentes e a confecção da prova de conceito;

d) Capítulo 4: aborda o desenvolvimento do protótipo, tais como o software embarcado, aplicativo móvel, processo de calibração e o software de visualização;

e) Capítulo 5: discorre sobre as conclusões obtidas com o desenvolvimento deste trabalho e recomendações para trabalhos futuros.

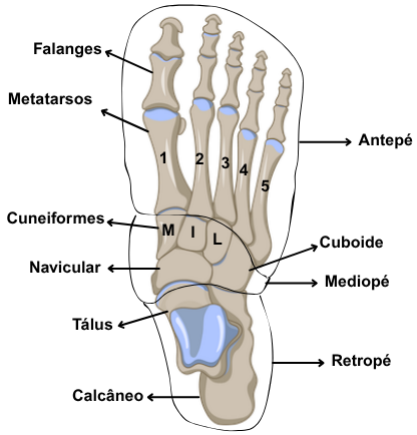
2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo descreve os conceitos teóricos necessários para o entendimento e desenvolvimento do protótipo.

2.1 Anatomia do pé

Conforme Lippert (2013) e apresentado na Figura 1, o pé humano é dividido em três regiões: retropé, mediopé e antepé. O retropé é formado pelos ossos tálus e calcâneo, sendo a primeira parte a fazer contato com o solo. O mediopé é composto pelos ossos navicular, cuboide e cuneiformes, sendo responsável pela estabilidade e mobilidade do movimento de transferência de apoio do retropé para o antepé. Por fim, o antepé é constituído pelos ossos metatarsais e falanges, realizando a adaptação ao solo e propulsão final.

Figura 1 – Anatomia do pé humano



Fonte: Adaptado de DBCLS (2021a).

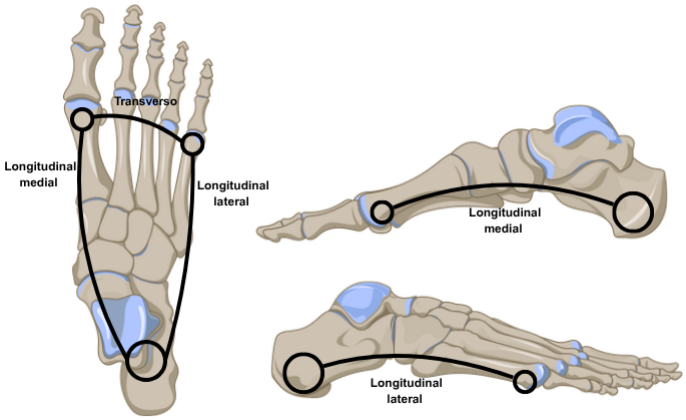
Para que o ser humano possa se manter em pé são necessários três pontos principais de apoio, dois deles localizados nos ossos metatarsais (1 e 4) e o terceiro no calcâneo, possibilitando a formação de três arcos: arco longitudinal medial, arco longitudinal lateral e arco transverso, conforme o descrito por Lippert (2013) e apresentado na Figura 2.

Com base nos arcos, é possível classificar o pé humano em três tipos: pé normal, pé plano e pé cavo. O pé normal apresenta a impressão plantar do mediopé sendo aproximadamente um terço do antepé e arco longitudinal medial normal. O pé plano, toda a sola do pé entra em contato com o solo tendo um arco longitudinal medial mais baixo. O pé cavo ocorre com a elevação do arco longitudinal medial, resultando em uma descontinuidade na impressão do mediopé (Carvalho; Berro; Fiorelli, 2021).

2.2 Sistemas operacionais de tempo real

Sistemas embarcados são amplamente utilizados em setores como automotivo, doméstico, de comunicação, robótica, aeroespacial, clínico e no controle de processos industriais. Tais

Figura 2 – Arcos de apoio do pé humano



Fonte: Adaptado de DBCLS (2021a, 2021b, 2021c).

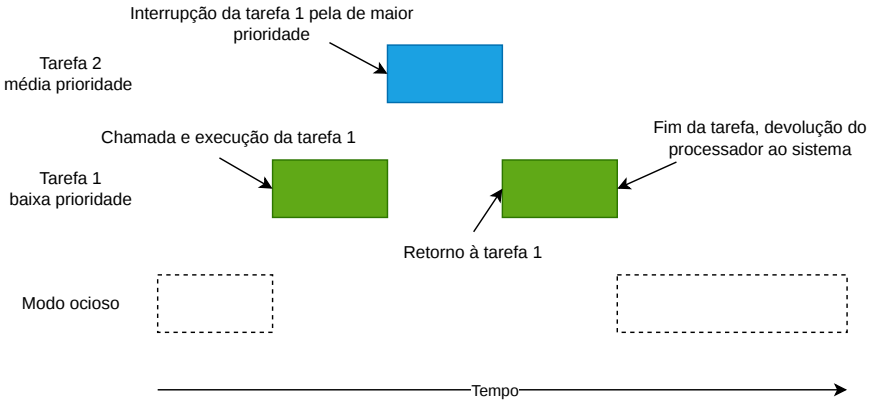
sistemas dedicam-se a executar um conjunto específico de tarefas e, normalmente, contam com um sistema operacional para organizar sua execução. Segundo Denardin e Barriquelo (2019), sistemas operacionais de tempo real (do inglês *real-time operating systems*, RTOS) são uma subclasse de sistemas operacionais nos quais o tempo de resposta a um evento é fixo e deve ser respeitado, resultando em falhas caso os requisitos temporais sejam violados. Tais sistemas podem ser divididos em dois tipos: os de tempo real leve, nos quais, mesmo com quebras temporais, o sistema é capaz de retornar à operação quando os requisitos temporais são novamente respeitados, e os de tempo real rígido, nos quais a violação de requisitos ocasiona o colapso do sistema. A maioria dos sistemas existentes opera com uma combinação desses dois tipos.

No contexto de um RTOS, a tarefa corresponde à execução de uma ação por meio de sequência de instruções. Cada tarefa possui sua própria pilha de memória, destinada ao armazenamento de variáveis locais e dos dados dos registradores da unidade de processamento, permitindo o retorno em caso de interrupção, conhecido como contexto da tarefa. Tipicamente, as tarefas são implementadas como laços infinitos e podem assumir cinco estados: execução, pronto para execução, espera, interrompido ou inativo (Denardin; Barriquelo, 2019).

Outro conceito importante para um RTOS é a preemptividade. Segundo Denardin e Barriquelo (2019), em um sistema com núcleo preemptivo, uma tarefa pode ser interrompida a qualquer momento. A cada interrupção ou chamada de serviço, a fila de tarefas é reavaliada, e o sistema decide se ocorrerá a substituição da tarefa em execução. O método mais comum de gerenciamento é pela definição de prioridades, neste caso, a tarefa com maior prioridade sempre ganha acesso ao processador, tal comportamento é ilustrado na Figura 3. Um fator de atenção em RTOS preemptivos é a inanição de tarefas, tarefas de baixa prioridade podem apresentar dificuldades ou não serem executadas no caso de má configuração do sistema de prioridades.

Alguns erros podem ocorrer no desenvolvimento de um sistema embarcado com RTOS. Como mencionado anteriormente, a inanição de tarefas é um desses casos, mas também podem ocorrer impasses e sobrecargas. O impasse ocorre quando duas tarefas, ou mais, esperam

Figura 3 – Preemptividade em um RTOS



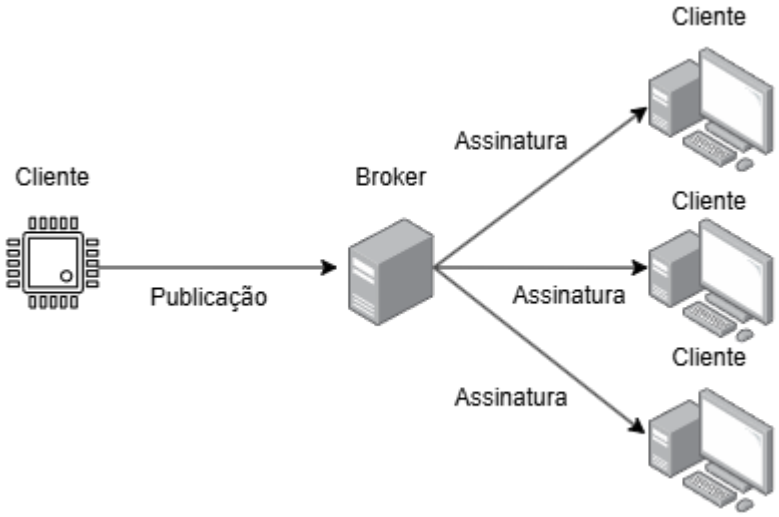
Fonte: Autoria própria (2026).

a alocação de um recurso que está sendo usado por outra. Por exemplo, a tarefa 1 aloca a comunicação serial, e a tarefa 2 aloca uma porta de saída digital, caso a tarefa 1 precise da porta alocada pela tarefa 2, e simultaneamente, a tarefa 2 precise da comunicação serial para finalizar, forma-se um impasse e nenhuma das tarefas é finalizada. Uma estratégia de resolução consiste em alocar, por meio de semáforos, todos os recursos ao iniciar a tarefa e desalocá-los em ordem inversa, além de definir um tempo máximo de espera para alocação. Caso não seja possível a alocação nesse intervalo, a tarefa retorna à fila de tarefas prontas para execução. Já no que se refere à sobrecarga, esta ocorre quando o processador não possui capacidade suficiente para execução das tarefas dentro dos requisitos temporais definidos. A estratégia de resolução, nesse caso, quando já foram realizadas todas as otimizações possíveis, é a substituição do processador por um de maior poder computacional (Denardin; Barriquelo, 2019).

2.3 Protocolo MQTT

Atualmente, com o crescimento do número de sistemas embarcados e a tendência por conectividade, principalmente em aplicações de monitoramento contínuo, surgem desafios relacionados à comunicação entre dispositivos. Dessa forma, o protocolo de *Message Queuing Telemetry Transport* (MQTT) foi desenvolvido em 1999 por Andy Stanford-Clark e Arlen Nipper para aplicações de monitoramento de oleodutos, sendo projetado para ser leve e eficiente. O seu modelo de comunicação baseia-se na arquitetura de publicação e assinatura de mensagens. Para compreender o funcionamento do protocolo, é necessário entender os conceitos de *broker*, clientes e tópicos. O *broker* atua como um agente intermediário, no qual é responsável por gerenciar a comunicação dos publicadores e dos assinadores, realizando o filtro, o roteamento de mensagens e o controle de segurança. Os clientes atuam tanto como publicadores, enviando mensagens, ou como assinadores, se inscrevendo em tópicos. Já os tópicos representam a categorização das mensagens, uma mensagem só pode ser enviada caso esteja associada a um tópico (HiveMQ, c2025). A Figura 4 apresenta um exemplo de uma rede MQTT.

Figura 4 – Rede MQTT



Fonte: Autoria própria (2026).

A conexão no protocolo MQTT é sempre estabelecida entre os clientes e o *broker*. Para inicializá-la, o cliente envia uma mensagem *CONNECT* e aguarda a resposta do *broker* com *CONNACK* acompanhado de um código de status. Com a conexão estabelecida, esta é mantida até o cliente enviar o comando de desconexão ou ocorrer uma falha de rede. Os comandos *PUBLISH* e *SUBSCRIBE* são utilizados, respectivamente, para enviar uma mensagem e assinar a leitura de um tópico. Esses comandos também possuem suas confirmações enviadas pelo *broker* com *PUBACK* e *SUBACK* respectivamente (HiveMQ, c2025).

Além disso, o protocolo MQTT apresenta diferentes níveis de confiabilidade, denominados de qualidade de serviço (do inglês *Quality of Service*, QoS). O QoS 0 é o nível mais básico, não realizando confirmação de recebimento, a mensagem é enviada uma vez e caso seja perdida nenhuma retransmissão ocorre. O QoS 1 garante que o conteúdo da mensagem será entregue pelo menos uma vez, embora possa ocorrer a recepção de mensagens repetidas a depender da qualidade da conexão. Já o QoS 2 apresenta a maior confiabilidade, garantindo que a mensagem seja entregue exatamente uma vez, porém exige mais da troca de pacotes do cliente e do *broker* podendo gerar latências (HiveMQ, c2025).

2.4 Metrologia

Embora protocolos de transmissão eficientes e confiáveis sejam fundamentais para o envio dos dados adquiridos pelo sistema, a qualidade dessa comunicação por si só não garante a validade científica das informações transmitidas. Após assegurar que os dados podem ser enviados de forma íntegra e com níveis adequados de confiabilidade, torna-se essencial avaliar a precisão, exatidão e a rastreabilidade das medições realizadas pelo hardware. Nesse contexto, torna-se indispensável o uso da metrologia, que é definida como a ciência da medição e suas aplicações, que desempenha um papel fundamental no desenvolvimento de qualquer sistema instrumental (Inmetro, 2008). A compreensão dos princípios metrológicos é o que na prática

diferencia um protótipo funcional de um instrumento científico válido e a partir disso fatores como erros, correções e incertezas de medições são inevitáveis para a validação científica do projeto instrumentado.

2.4.1 Erro e correção

Como parte da validação do protótipo, o erro de medição é parte fundamental da análise instrumental, sendo definido como a diferença entre o valor medido de uma grandeza e um valor de referência (Inmetro, 2008). Em um sistema de medição, todo resultado é afetado por diferentes fontes de erro, que são tradicionalmente caracterizadas em duas classes principais: erros sistemáticos ou erros aleatórios.

Erros sistemáticos são os que permanecem constantes ou variam de forma previsível durante medições repetidas sob as mesmas condições, podendo ser derivados de várias fontes, como a não-linearidade dos sensores, histerese, derivação térmica ou até mesmo erros intrínsecos aos componentes utilizados.

Erros aleatórios consistem em flutuações imprevisíveis no sinal medido. Tais erros não podem ser corrigidos, mas podem ser quantificados e mitigados. As maiores causas incluem ruídos eletrônicos, interferência eletromagnética e flutuações na resolução do conversor analógico-digital (do inglês *Analogic to Digital Converter*, ADC).

A correção pode ser feita de inúmeras formas, sendo mais simples para erros sistemáticos, nas quais o próprio método de calibração permite a correção dos erros a partir de uma equação polinomial que aproxima o valor bruto lido do valor verdadeiro. Já os erros aleatórios constituem casos mais complexos, corrigidos através de análises e métodos estatísticos (Figliola; Beasley, 2007).

2.4.2 Calibração

A calibração é a operação que, sob condições especificadas, estabelece uma relação entre os valores indicados por um sistema de medição e os valores correspondentes de um padrão de referência (Inmetro, 2008). O resultado é uma função de calibração, ou curva de calibração, que mapeia a saída do instrumento para o valor estimado do mensurando.

O modelo mais comum e desejável em sistema de medição é o linear. Considera-se que a resposta do sistema y é uma função linear do mensurando x descrito pela equação 1, em que m é a sensibilidade e b é o *offset*.

$$y = mx + b \tag{1}$$

Durante a calibração, n pares de dados (x_i, y_i) são coletados, na qual x_i é o valor aplicado pelo padrão de referência e y_i é a resposta média do sistema. Utiliza-se o Método dos

Mínimos Quadrados (MMQ) para encontrar os coeficientes m e b que minimizam a soma dos quadrados dos resíduos (Figliola; Beasley, 2007).

Para um conjunto de n pares de dados, os coeficientes m e b são calculados por meio das equações 2 e 3, respectivamente, as quais $\bar{x} = \frac{\sum x_i}{n}$ e $\bar{y} = \frac{\sum y_i}{n}$ são as médias dos valores de entrada e saída.

$$m = \frac{n\sum x_i y_i - \sum x_i \sum y_i}{n\sum x_i^2 - (\sum x_i)^2} \tag{2}$$

$$b = \bar{y} - m\bar{x} \tag{3}$$

Após a calibração, a equação é invertida para criar a função de medição, responsável pela correção do erro sistemático. Dada uma nova leitura y_{nova}, o valor corrigido do mensurando $x_{estimado}$ é descrito pela equação 4.

$$x_{estimado} = \frac{y_{nova} - b}{m} \tag{4}$$

A qualidade do ajuste linear é avaliada pelo coeficiente de determinação (R^2), o qual indica a proporção da variância da saída que pode ser explicada a partir da entrada. Além de quantificar a proporção de variância, o coeficiente de determinação R^2 também pode ser interpretado como uma medida indireta da distância entre a curva ajustada e os pontos experimentais (Montgomery; Runger, 2014), como mostra a equação 5, em que y_i representa os valores medidos, $\hat{y}_i$ os valores estimados pelo modelo e $\bar{y}$ a média das medições.

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{5}$$

Todas essas constatações podem ser levadas em consideração no contexto de modelos lineares. Porém, ao analisar modelos não lineares, quando o R^2 for baixo ou se a análise de resíduos não mostrar um padrão claro, um modelo linear é inadequado. Nesses casos, utiliza-se a regressão polinomial, como o modelo quadrático, conforme mostra a equação 6, frequentemente necessário para sensores que exibem saturação ou não linearidade acentuada. Nesse tipo de modelo, cada coeficiente possui uma propriedade, na qual c_0 representa o termo independente, relacionado ao valor de saída quando a entrada é nula, o c_1 descreve a componente linear da resposta, e c_2 é o coeficiente quadrático, que caracteriza a curvatura da função.

$$y = c_0 + c_1 x + c_2 x^2 \tag{6}$$

Embora a determinação dos coeficientes c_i seja mais complexa, pois envolve a solução de um sistema de equações lineares, o procedimento é o mesmo, deve-se encontrar a curva que melhor se ajusta aos dados de calibração. A função de medição torna-se a inversa desse polinômio, a raiz da equação quadrática.

2.5 Tecnologias relacionadas

Uma revisão da propriedade intelectual recente, tanto em âmbito internacional *United States Patent and Trademark Office* (USPTO) quanto nacional Instituto Nacional da Propriedade Industrial (INPI), revela um campo de inovação robusto e diversificado relacionado a palmilhas inteligentes e sensorizadas. A análise desses documentos demonstra uma direção que se afasta da simples medição de pressão plantar, evoluindo para sistemas integrados de monitoramento, diagnóstico e até mesmo terapêuticos. As tecnologias propostas podem ser agrupadas com base em suas inovações de aplicação, arquitetura de hardware e integração de sensores.

Uma vertente significativa das inovações destina-se a aplicações clínicas específicas, com um grande aprofundamento para a prevenção do pé diabético e a reabilitação motora.

Para o monitoramento de pacientes diabéticos, algumas patentes buscam criar um sistema de alerta precoce para ulcerações. A invenção de Alsafran *et al.* (2024) propõe um calçado inteligente que, além dos sensores de pressão, integra sensores de temperatura e de oximetria de pulso. A combinação desses dados permite monitorar não apenas os pontos de alta pressão, mas também indicadores de perfusão sanguínea e inflamação, fornecendo um quadro diagnóstico mais completo. No Brasil, a patente de Souza *et al.* (2025) segue uma linha similar, descrevendo uma órtese plantar sensorizada especificamente voltada para a prevenção do pé diabético, focando no monitoramento da distribuição da pressão.

Em outra aplicação terapêutica, a patente de Colcioni *et al.* (2025) descreve uma palmilha sensorizada para o auxílio à marcha de pacientes hemiplégicos. A inovação apresentada consiste no uso do sistema como uma ferramenta de *biofeedback*. O dispositivo detecta a evolução da pressão ao longo do pé durante a marcha e, por meio de um sistema de processamento, identifica as fases da pisada para auxiliar no reaprendizado motor, demonstrando o potencial de palmilhas como dispositivos de reabilitação ativa.

Nas patentes analisadas, observa-se como tendência comum o avanço na miniaturização e na integração de componentes eletrônicos. A invenção de Finn *et al.* (2025) aborda diretamente a arquitetura do sistema para pressão plantar, propondo um microcontrolador integrado diretamente ao próprio substrato flexível da palmilha. Essa abordagem difere de designs tradicionais que frequentemente alojam o microcontrolador e a bateria em um módulo separado no calcanhar ou tornozelo, permitindo um design mais fino e integrado.

A configuração dos próprios sensores de pressão também constitui um ponto de inovação. Wieczorek (2025) detalha um tênis de corrida sensorizado no qual os sensores de pressão são configurados em um padrão de matriz de alta resolução, com espaçamento de 1 cm. Para gerenciar eficientemente essa grande quantidade de sensores, a patente descreve o uso de técnicas de multiplexação implementadas no microcontrolador.

Diversas patentes buscam enriquecer a análise de dados combinando os sensores de pressão com outras modalidades de sensoriamento. A patente de Chou (2025) descreve palmilhas inteligentes para análise de marcha as quais incorporam sensores inerciais, como acelerômetros e giroscópios, juntamente com os sensores de pressão. Essa fusão de dados permite o

cálculo de parâmetros biomecânicos complexos, como a cadência, o tempo de contato com o solo e, crucialmente, o centro de pressão, proporcionando uma análise de marcha muito mais completa do que a pressão isolada.

De forma ainda mais expansiva, Ko e Liu (2024) propõem uma palmilha que, além dos sensores de pressão, inclui sensores de gases para analisar compostos voláteis emitidos pelo pé. A patente sugere o uso de algoritmos de aprendizado de máquina para analisar o conjunto de dados de pressão, gases e movimento e monitorar a saúde ou desempenho do usuário.

Por fim, um grupo mais específico de invenções foca em melhorar a experiência do usuário e a praticidade do dispositivo. A patente de Chou (2025) menciona a inclusão de carregamento sem fio, eliminando a necessidade de conectar cabos para recarregar a bateria. Ko e Liu (2024) sugerem fontes de alimentação alternativas, como geradores piezoelétricos para captação de energia proveniente da própria pisada.

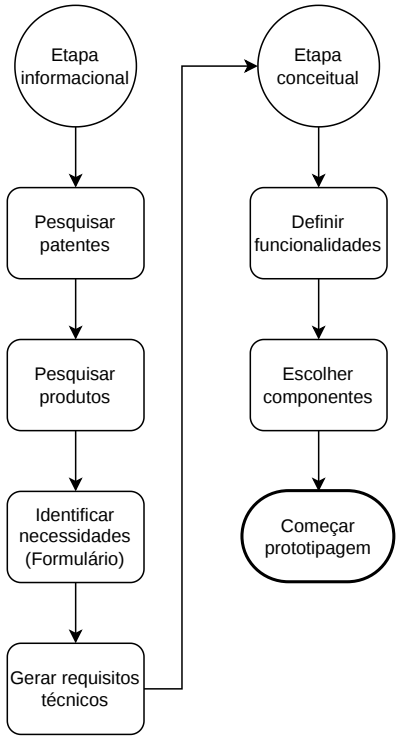
Uma característica não encontrada em outras patentes é apresentada por Wieczorek (2025), que propõe um mecanismo de desinfecção por luz ultravioleta UV-C integrada ao dispositivo, abordando a questão da higiene em dispositivos vestíveis de uso contínuo.

O panorama de patentes demonstra que a tecnologia de palmilhas sensorizadas está em desenvolvimento contínuo, movendo-se em direção a sistemas preditivos e integrados. As inovações não param apenas nos sensores de pressão em si, mas na sua fusão com outros sensores (inerciais, térmicos, químicos), em arquiteturas de hardware mais eficientes e em algoritmos de análise avançados para aplicações que vão da reabilitação clínica ao monitoramento de desempenho esportivo.

3 METODOLOGIA

Para o desenvolvimento do protótipo, adotou-se como referência a metodologia de desenvolvimento de produtos proposta por Pahl *et al.* (2005). Em um primeiro momento, realiza-se a etapa informacional, com o objetivo de compreender melhor o projeto, identificar as necessidades do cliente e definir os requisitos técnicos. Conforme ilustrado na Figura 5, a primeira tarefa será realizar a pesquisa de patentes relacionadas, seguida pela pesquisa de mercado. Dessa forma, é possível compreender de maneira mais precisa o produto em desenvolvimento. Em seguida, é necessário identificar as necessidades do público-alvo do produto e convertê-las em requisitos técnicos. Posteriormente, com os resultados obtidos da etapa informacional, procede-se com a etapa conceitual. Nesta etapa, realiza-se a definição das funcionalidades do produto e a atribuição de um ou mais componentes físicos para o seu cumprimento. Por fim, inicia-se a prototipagem.

Figura 5 – Organização da etapa informacional



Fonte: Autoria própria (2026).

3.1 Etapa informacional

Inicialmente, realizou-se uma pesquisa de patentes com objetivo de conceituar o protótipo e identificar tecnologias aplicáveis. Para tal, foram utilizados dois bancos de dados de patentes, o Instituto Nacional da Propriedade Industrial (INPI) e o *United States Patent and Trademark Office* (USPTO). O critério adotado para considerar uma patente como adequada

consiste na presença de elementos sensores de pressão internos aos calçados e na inclusão de componentes eletrônicos para o tratamento e coleta de dados.

Na pesquisa realizada no INPI, a busca foi conduzida pela busca exata no campo de resumo pelos seguintes termos: "Pressão plantar", "Palmilha sensorizada", e "Palmilha inteligente". Foram obtidas 10, 2 e 4 patentes, respectivamente, das quais quatro foram consideradas adequadas para análise. Quanto à pesquisa no USPTO, realizou-se a busca pelas seguintes combinações de termos: "*Plantar AND Pressure*"; "*Smart AND Insole AND Sensor*"; e "*Insole AND Sensorized*". Obteve-se 21.173, 931 e 63 patentes, respectivamente, das quais cinco foram consideradas adequadas.

Em geral, constataram-se os seguintes pontos:

a) os sensores de pressão utilizados podem ser de tipo filme resistivo ou capacitivos;

b) todos possuem mais de um elemento sensor;

c) software de visualização de dados presente em grande maioria; e

d) utilização de sensores complementares, como temperatura, frequência cardíaca, oxigenação sanguínea e acelerômetros.

Posteriormente, foi realizada uma pesquisa sobre soluções comerciais, com o mesmo objetivo informacional da pesquisa de patentes. Entretanto, as soluções comerciais oferecem tecnologias mais tangíveis e acessíveis em comparação às patentes intelectuais. Dessa forma, o Quadro 1 apresenta os produtos encontrados. Verifica-se que:

a) dependendo da aplicação e dos objetivos do monitoramento o sistema pode ser no formato de plataforma ou palmilha;

b) o tipo do sensor e a quantidade são variados;

c) taxas de medição em grande maioria configuráveis pelo usuário;

d) todos possuem um software de monitoramento em tempo real; e

e) custos elevados.

Conforme Pahl *et al.* (2005), a próxima etapa consiste em elencar os requisitos do cliente. Para tal, aplicou-se um questionário online baseado nas informações obtidas das pesquisas prévias. Para ser considerado estatisticamente válido são necessárias, no mínimo, trinta respostas a fim de garantir uma generalização dos resultados (Back *et al.*, 2008).

O questionário inicia-se com uma primeira pergunta niveladora, com o objetivo de filtrar o grupo de análise. Para ser considerado apto, o respondente deve pertencer a um dos seguintes grupos: pessoas com interesse em monitorar a pressão da pisada, pessoas com condições médicas relacionadas à pressão da pisada, usuário de palmilhas posturais, profissionais da saúde da área ortopédica ou atletas que realizam treinos proprioceptivos. Em seguida, são feitas perguntas de múltipla escolha para avaliar o grau de importância de cada requisito, em uma

Quadro 1 – Resultado da pesquisa de produtos comerciais para medir a pressão plantar

Produto	Sensor	Taxa de medição	Alimentação	Software de monitoramento	Armazenamento	Preço
Pedar Novel®	Capacitivo (até 256 sensores)	Até 400 Hz	Bateria	Sim	Interno	~U$ 3.000
F-Scan Tekscan®	Resistivo (até 966 sensores)	Até 500 Hz	Bateria (2 horas)	Sim	Interno + cartão	~U$ 3.000
ReGo moticon®	Resistivo (16 sensores)	200 Hz	Bateria	Sim	Nuvem	U$ 949,00
Medilogic WLAN®	Resistivo (até 240 sensores)	Até 400 Hz	Bateria	Sim	—	—
emed Novel®	Capacitivo (até 25.344 sensores)	100 Hz	Cabo	Sim	—	—

Fonte: Autoria própria (2026).

escala de um a cinco, na qual um é pouco importante e cinco muito importante. Posteriormente, mais duas perguntas de múltipla escolha são feitas referentes à taxa de medição e ao custo do produto. Ao final, uma caixa de sugestões é disponibilizada para que o respondente preencha com características que não foram apresentadas no questionário indicando, também, seu nível de importância. Ao todo, foram obtidas 37 respostas válidas e, no Quadro 2, apresenta-se a média ponderada de cada tópico, organizada de forma decrescente. Para as perguntas de taxa e custo, foi selecionada a opção com mais votos, excluindo as demais e atribuindo a média máxima.

Quadro 2 – Média ponderada de cada tópico da pesquisa

Tópico	Média ponderada
Fazer entre 1 a 10 medições por segundo	5,0
Custar até 500 reais	5,0
Ser confortável para o uso prolongado	4,7
Ser resistente à água e ao suor	4,7
Gravar dados para análise posterior	4,7
Ser fácil de usar	4,5
Ser preciso nas medições da pressão da pisada	4,5
Ser adaptável à diferentes tipos de calçados	4,4
Ser fácil de limpar	4,3
Ter diferentes modos de medição (em pé, caminhando, correndo)	4,2
Ter grande autonomia energética	4,0
Ser interno ao calçado	3,9
Ser configurável por aplicativo	3,9
Enviar alertas ao usuário quando a pressão da pisada estiver fora de uma faixa saudável	3,8
Gerar relatórios automáticos sobre a pressão da pisada	3,9
Ser capaz de medir aceleração	3,7
Fornecer retorno em tempo real	3,4
Ser capaz de medir batimentos cardíacos	3,2
Ser capaz de medir temperatura	3,0
Ser capaz de medir oxigenação sanguínea	2,8
Ser capaz de medir umidade	2,6
Ser uma plataforma rígida	2,3
Ser discreto e não interferir no visual do calçado	3,7

Fonte: Autoria própria (2026).

Após o encerramento da pesquisa, aplicou-se o método de desdobramento da função qualidade, convertendo os requisitos do cliente em requisitos técnicos para o preenchimento da primeira casa da qualidade, com o objetivo de obter as especificações-meta do protótipo (Hauser; Clausing, 1988). No Quadro 3, apresentam-se as especificações elencadas de forma decrescente e a nota equivalente. Para as próximas etapas, selecionaram-se as 15 maiores notas como necessárias para o protótipo.

Quadro 3 – Especificações-meta obtidas da primeira casa da qualidade

Requisito técnico	Posição	Nota
Sistema de Fixação	1º	260,9
Consumo energético	2º	199,6
Espessura da palmilha	3º	155,4
Sistema de medição de pressão acurado	4º	149,3
Aplicativo funcional	5º	138,4
Tamanho	6º	113,6
Montagem simplificada	7º	103,1
Sistema de armazenamento de dados	8º	102,4
Material flexível	9º	100,9
Interface intuitiva (aplicativo)	10º	96,5
Vedação dos componentes	11º	95,3
Superfície lisa	12º	89,8
Componentes comerciais	13º	82,0
Modos de medição	14º	80,8
Sistema de visualização de dados	15º	57,6
Peso	16º	54,3
Sistema de medição de aceleração	17º	53,5
Sistema de medição de batimentos cardíacos	18º	48,6
Sistema de medição de temperatura	19º	46,6
Sistema de medição de oxigenação	20º	44,6
Sistema de alerta	21º	43,5
Sistema de medição de umidade	22º	42,6
Cores escuras	23º	41,7
Modo de hibernação	24º	39,9
Sistema de relatório automático	25º	38,9

Fonte: Autoria própria (2026).

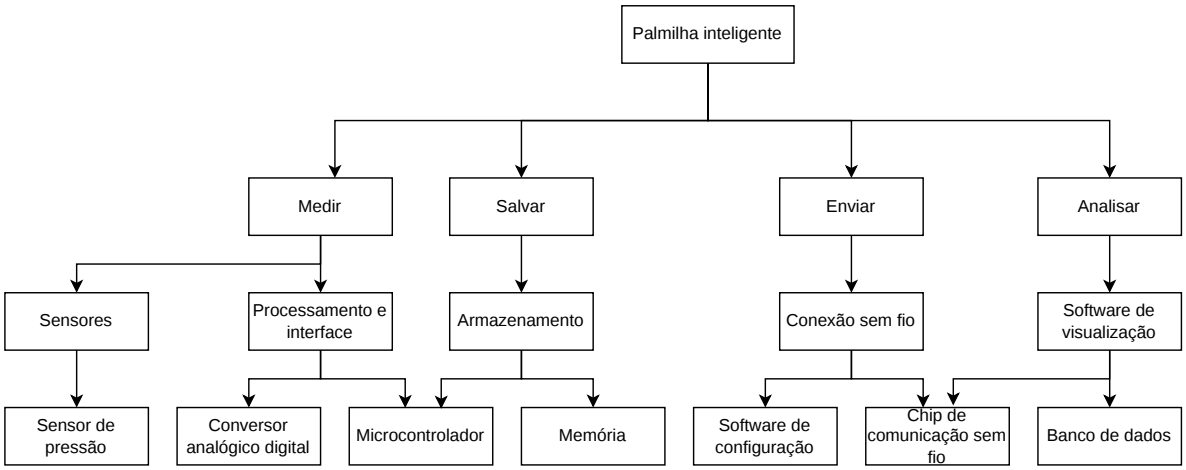
3.2 Etapa conceitual

Com base nas especificações-meta, é possível definir o escopo do protótipo. O protótipo é descrito como uma palmilha inteligente, fina, lisa e flexível, projetada para uso interno ao calçado e com vedação. O dispositivo deve ser de fácil instalação, grande autonomia, tamanho reduzido, possuir armazenamento de dados e elevada confiabilidade dos sensores de pressão. A interface com o usuário ocorre por um aplicativo para configuração intuitiva de modos de medição e sistema de visualização em tempo real de dados.

A partir das funções esperadas para o protótipo, estruturou-se o conjunto mínimo de componentes necessários. Dessa forma, o protótipo, como é apresentado na Figura 6, deve conter: sensores de pressão, ADC, microcontrolador, memória, software de configuração, chip de comunicação sem fio, e banco de dados para o software de visualização.

Grande parte das patentes e produtos comerciais analisados na etapa informacional utiliza sensores de filme resistivo para mensurar a pressão plantar. Algumas soluções apresentam sensores capacitivos e, para casos específicos de tratamento, sensores óticos. A Universidade Tecnológica Federal do Paraná dispõe do sensor FS-INS-16Z da Legact®, composto por 16

Figura 6 – Estrutura funcional do protótipo



Fonte: Autoria própria (2026).

sensores do tipo filme resistivo, com espessura de 0,4 mm, material flexível, resistência à água e poeira com grau IP67, e faixa de sensibilidade de 500 g a 10 Kg (Legact, c2025).

Atualmente, muitos microcontroladores já apresentam soluções integradas para comunicação sem fio, como a linha nRF da Nordic Semicondutor®, linha BG da Silabs® ou a linha ESP da Espressif®. Por questões de disponibilidade de mercado, funcionalidades, tamanho e custo, a placa de desenvolvimento escolhida foi a ESP32-S3FH4R2 que conta com o microcontrolador Xtensa® LX7 de dois núcleos de até 240 MHz, e módulos integrados de Wi-Fi 2.4 GHz e Bluetooth® 5 Low Energy (BLE).

Com base na quantidade de pinos de entrada e saída disponíveis na placa de desenvolvimento, percebe-se que para realizar a leitura simultânea dos 16 sensores, seria necessária a utilização de todos os pinos, impossibilitando outras conexões ao microcontrolador. Dessa forma, é inviável a leitura simultânea fazendo necessário o uso da técnica de multiplexação. Por disponibilidade local e quantidade de canais, foi escolhido o multiplexador analógico CD74HC4067 de 16 para 1 com atraso máximo de propagação de 500 ns, reduzindo assim o uso de 16 pinos para 5 — 4 pinos de saída de controle e 1 pino de entrada do sinal e garantindo velocidade de chaveamento.

Em relação ao ADC, a placa ESP32-S3FH4R2 conta com um ADC de aproximação sucessiva de 12 bits. No entanto, pela demanda de precisão do protótipo, sua utilização não é adequada, visto que as placas ESP não apresentam um circuito tão robusto se comparadas com outras fabricantes ou com chips dedicados, necessitando da utilização de táticas de *oversampling* ou filtros *anti-aliasing*. Dessa forma, optou-se pelo ADC externo de 24 bits ADS1220 do tipo $\Delta\Sigma$ por possuir capacidade de realizar até 2.000 amostras por segundo, o que é suficiente para varrer os 16 canais a uma frequência de 100 Hz, caso necessário. Além disso, conta com ganho interno programável, que oferece maior flexibilidade ao projeto, permitindo as tensões limite da faixa de leitura conforme a necessidade.

Quanto à memória, a utilização de um chip integrado limitaria o usuário, exigindo que importe e remova periodicamente os dados armazenados, além de acrescentar complexidade

ao aplicativo de configuração. Portanto, optou-se pelo armazenamento em cartões de memória do tipo microSD, utilizando o módulo baseado no chip 74LVC125A.

Por fim, o banco de dados necessário ao software de visualização poderá ser obtido a partir de duas fontes, do arquivo armazenado no cartão de memória ou de um banco de dados local populado pela placa ESP por meio do protocolo MQTT. Dessa forma, pode-se realizar tanto a análise posterior quanto a análise em tempo real.

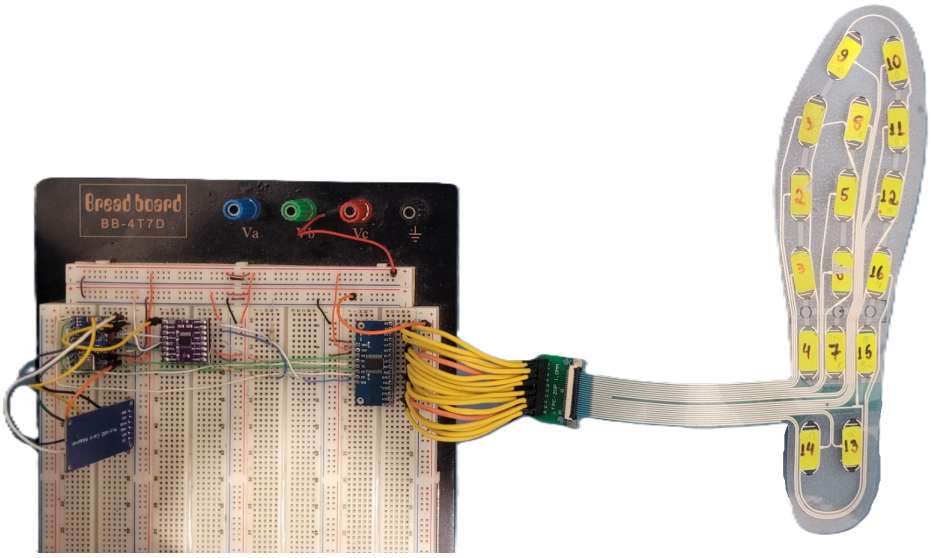
3.3 Prova de conceito

Para verificar a viabilidade técnica dos componentes escolhidos, realizou-se a montagem de um primeiro circuito para a prova de conceito. Os resultados obtidos são apresentados a seguir.

3.3.1 Circuito eletrônico

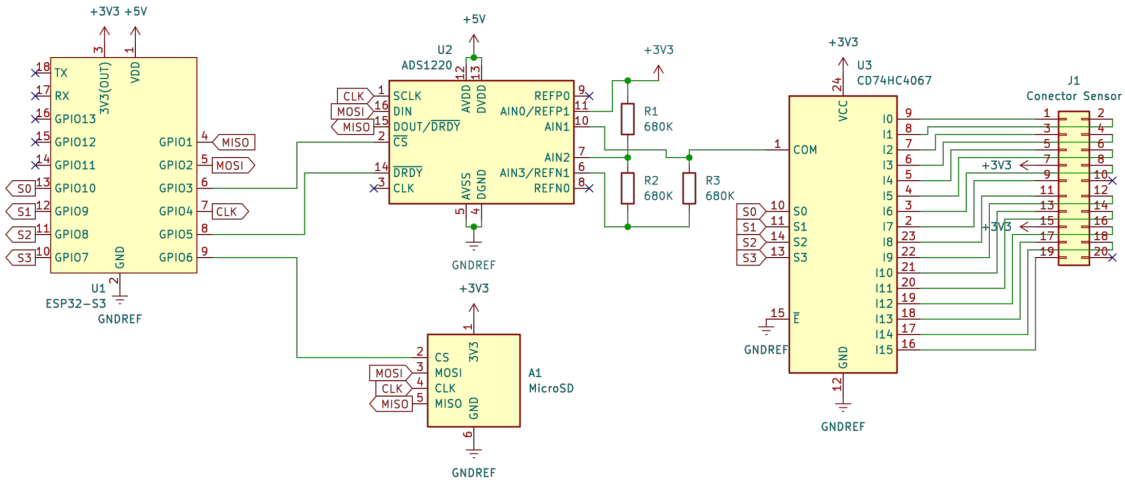
O circuito de teste foi montado em uma placa de ensaio (Figura 7) e segue o esquemático apresentado na Figura 8. No circuito, a placa de desenvolvimento ESP32-S3 está conectada ao ADS1220 e ao módulo microSD por meio do mesmo barramento da Interface Serial Periférica (do inglês *Serial Peripheral Interface*, SPI). O ADS1220 realiza a leitura diferencial da ponte de Wheatstone, utilizando a saída do multiplexador CD74HC4067 para compor o braço variável da ponte. O multiplexador está conectado ao sensor FS-INS-16Z por meio de um conector de 20 vias.

Figura 7 – Protótipo montado em uma placa de ensaio



Fonte: Autoria própria (2026).

Figura 8 – Esquemático elétrico do protótipo



Fonte: Autoria própria (2026).

Em relação ao algoritmo para a prova de conceito, utilizou-se o RTOS FreeRTOS[1] para a codificação de três tarefas fundamentais: leitura, armazenamento no microSD e depuração.

Na tarefa de leitura, realiza-se o preenchimento de uma estrutura composta por uma variável correspondente ao instante da leitura e um vetor de 16 posições contendo os valores de cada sensor, sua prioridade está configurada como a mais baixa. Ao inicializar a leitura dos sensores é obtido o *timestamp* do ESP32-S3, em seguida, envia-se o comando de controle de chaveamento do multiplexador, então é realizada a tentativa de obter o semáforo da SPI. Caso o semáforo seja obtido com sucesso, inicia-se um iterador de varredura percorrendo todos os sensores inserindo o valor lido na sua posição do vetor, ao final, insere a estrutura gerada numa fila de leitura do tipo primeiro a entrar, primeiro a sair e libera o semáforo da SPI. Com base em testes de estresse e nas especificações do *hardware*, a tarefa de leitura possui limite de frequência máxima de 100 Hz, pois, com taxas maiores, ocorrem mais conflitos com as demais tarefas, causando uma sensação de atraso e aumentando a instabilidade do ADC, visto que seu limite operacional é de 2.000 amostras por segundo, equivalente a 125 Hz por sensor.

Na tarefa de armazenamento, ocorre a escrita no microSD da estrutura obtida da fila de leituras, sua execução ocorre a uma taxa de 3 Hz e a prioridade está configurada para a mais alta. Inicialmente, verifica-se a quantidade de estruturas disponíveis na fila, caso seja menor que 32, toda a fila é obtida e caso seja maior, obtém-se as 32 leituras mais antigas. Após isso, realiza-se novamente a tentativa de obtenção do semáforo da SPI, se for possível ocorre a escrita em lote no microSD por separação de valores com ponto e vírgula, posteriormente liberando o semáforo.

Por fim, na tarefa de depuração, o barramento serial é utilizado para o envio da cópia da última leitura a uma taxa de 3 Hz e a sua prioridade é mediana. Sua função é fornecer uma forma de verificação do estado da leitura, permitindo a identificação ágil de eventuais inconsistências através do software de visualização.

[1] Disponível em: https://www.freertos.org/

3.3.2 Mapa de calor

Como parte decorrente da aquisição dos dados brutos de pressão plantar determinados pelo protótipo de hardware, passou a ser necessário o desenvolvimento de uma plataforma de processamento e visualização. Para este fim, foi desenvolvido um software de análise preliminar, cujo objetivo principal é traduzir os dados numéricos de séries temporais, provenientes dos 16 sensores da palmilha, em uma representação visual dinâmica e intuitiva, permitindo a subsequente análise biomecânica dos dados coletados.

Para o desenvolvimento da aplicação, optou-se pela linguagem de programação *Python*. Esta escolha foi motivada pela ampla aplicação e vasta disponibilidade de bibliotecas de código aberto voltadas para a ciência de dados, o que permitiu um desenvolvimento ágil e robusto.

A estruturação dos dados foi gerenciada pela biblioteca Pandas[2], para a qual foi desenvolvido um analisador e separador de caracteres, capaz de ler os arquivos de texto gerados pelo sistema. Este analisador foi projetado para identificar e extrair apenas as amostras válidas, compostas por um registro de tempo e 16 valores de pressão para cada um dos sensores da aplicação, e os dados foram carregados em uma estrutura otimizada para análise de séries temporais.

O resultado central desta etapa de visualização de dados é a geração de um mapa de calor dinâmico, renderizado em uma interface gráfica para usuários construída com a biblioteca *Tkinter*[3]. A etapa consistiu em traduzir a amostragem espacial dos 16 sensores em uma superfície de pressão contínua e visualmente intuitiva. Este processo foi iniciado com um mapeamento geométrico, no qual as coordenadas (X, Y) exatas de cada sensor, bem como o contorno do dispositivo, foram vetorizadas a partir de uma imagem de alta resolução da palmilha.

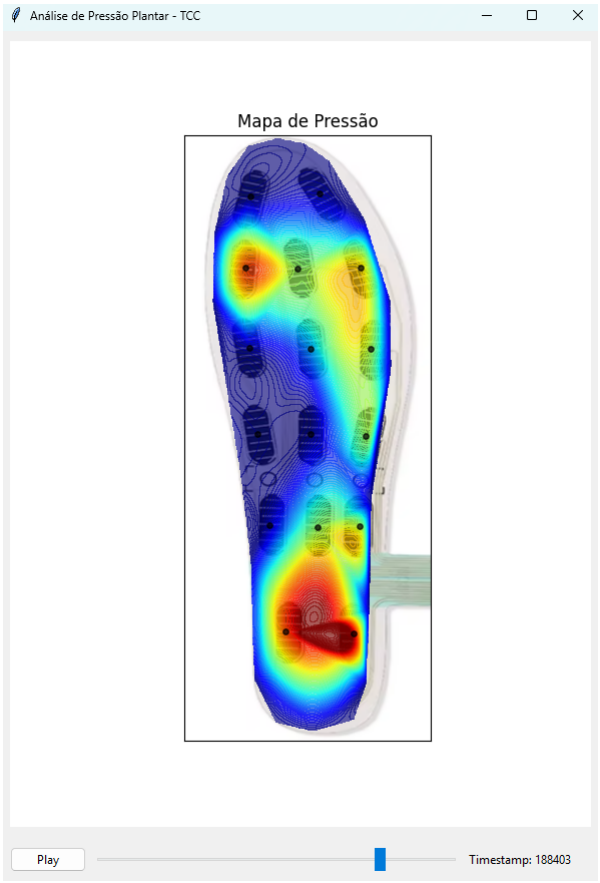
Para estimar os valores de pressão nas áreas entre os sensores, empregou-se um método de interpolação espacial disponível na biblioteca *SciPy*[4] (especificamente a função *griddata* com método cúbico). Uma limitação comum desta técnica é a sua incapacidade de extrapolar dados além da área coberta pelos sensores, limitação que foi solucionada com a implementação de uma rotina que adiciona 'sensores virtuais' com pressão zero ao longo de todo o contorno da palmilha. Esta abordagem força o algoritmo a estender a interpolação até as bordas do dispositivo, resultando em uma transição suave e fisicamente coerente da pressão, tendendo a zero. O contorno vetorizado é utilizado como uma máscara digital, garantindo que apenas a área de interesse da palmilha seja contemplada pela interpolação dos dados.

O software final representado na Figura 9 consolida essas técnicas e ferramentas em uma interface interativa que exibe o mapa de calor interpolado sobreposto, com transparência, à imagem fotográfica da palmilha, facilitando a correlação entre o dispositivo físico e as amostras de dados adquiridas pelo circuito de teste. A ferramenta permite a análise estática de um instante de tempo específico através de um controle deslizante, bem como a análise dinâmica

[2] Disponível em: https://pandas.pydata.org/
[3] Disponível em: https://tkdocs.com/
[4] Disponível em: https://scipy.org/pt/

de todo o ciclo da pisada por meio de uma funcionalidade de animação com acionamento e pausa, com o tempo exato de cada amostra sendo exibido. Adicionalmente, foi implementada uma funcionalidade de exportação que utiliza a biblioteca *Plotly*[5] para gerar um relatório web interativo do *frame* atual, permitindo ao usuário aplicar zoom e inspecionar os valores de pressão em qualquer ponto do mapa.

Figura 9 – Interface de visualização



Fonte: Autoria própria (2026).

Em suma, a ferramenta de software desenvolvida não apenas valida a cadeia de aquisição de dados do protótipo, mas também proporciona uma plataforma para a extração dos resultados biomecânicos preliminares.

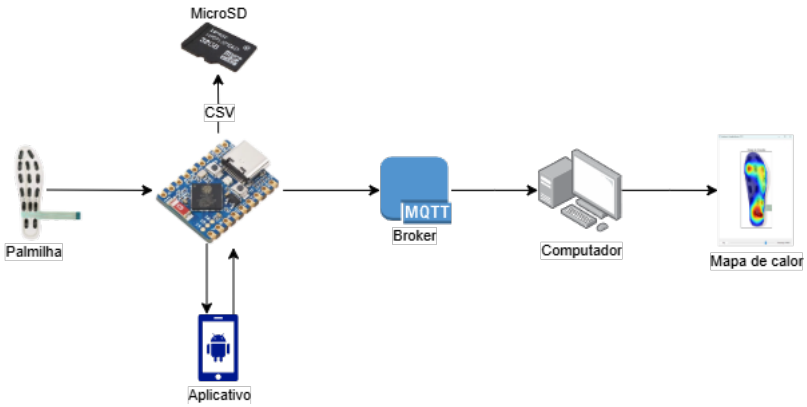
[5] Disponível em: https://plotly.com/python/

4 DESENVOLVIMENTO

Este capítulo apresenta o desenvolvimento do protótipo final, abordando o desenvolvimento do software e os resultados do processo de calibração.

O desenvolvimento do software foi organizado em três partes: *firmware*, software de controle e software de visualização. Cada parte é descrita em seu respectivo tópico. A estrutura geral do protótipo é apresentada na Figura 10.

Figura 10 – Estrutura geral do protótipo



Fonte: Autoria própria (2026).

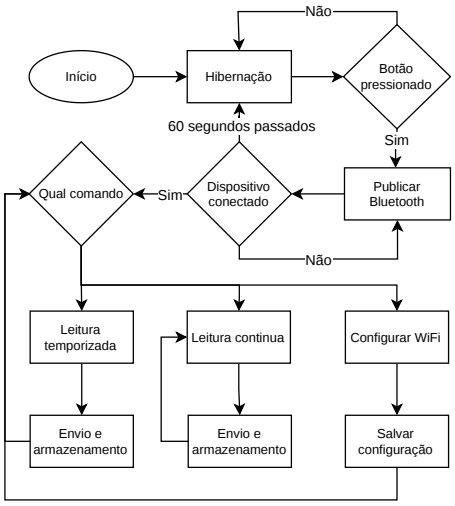
4.1 Firmware

O algoritmo utilizado na etapa da prova de conceito foi aprimorado por meio da implementação de novas funcionalidades, como o envio de dados ao *broker* MQTT e a comunicação com o aplicativo móvel para envio e recebimento de comandos. Inicialmente, o usuário deve pressionar o botão físico para acordar o dispositivo e iniciar a publicação do serviço Bluetooth. Esse processo permanece ativo enquanto a conexão com o aplicativo não é estabelecida, respeitando o limite máximo de 60 segundos. Caso nenhuma conexão seja realizada nesse intervalo, o dispositivo retorna automaticamente ao modo de hibernação.

Após a conexão, disponibiliza-se no aplicativo móvel três comandos: configuração de rede Wi-Fi, na qual o usuário insere o nome da rede e a senha a ser usada armazenando esta informação na memória do dispositivo; a leitura temporizada, com duração de 40 segundos de atividade, com envio ao *broker* e armazenamento no microSD; e a leitura contínua na qual os dados permanecem sendo enviados e armazenados até que o usuário envie o comando de parada pelo aplicativo móvel. Tal funcionamento é exemplificado na Figura 11.

No que se refere ao RTOS, essa versão conta com quatro tarefas: leitura, armazenamento, envio e aplicativo. A tarefa de leitura mantém a taxa de aquisição de 100 Hz, teve sua prioridade elevada e foi configurada para ser executada no segundo núcleo do processador, a fim de reduzir conflitos com as tarefas de Wi-Fi e Bluetooth, normalmente alocadas no núcleo primário pelas bibliotecas utilizadas. O código dessa tarefa é apresentado no Apêndice A.

Figura 11 – Fluxo de utilização



Fonte: Autoria própria (2026).

A tarefa de armazenamento permaneceu praticamente inalterada, com modificações apenas em sua alocação para o núcleo primário e em sua prioridade, definida como a segunda maior do sistema. No Apêndice B está disponível o código da tarefa.

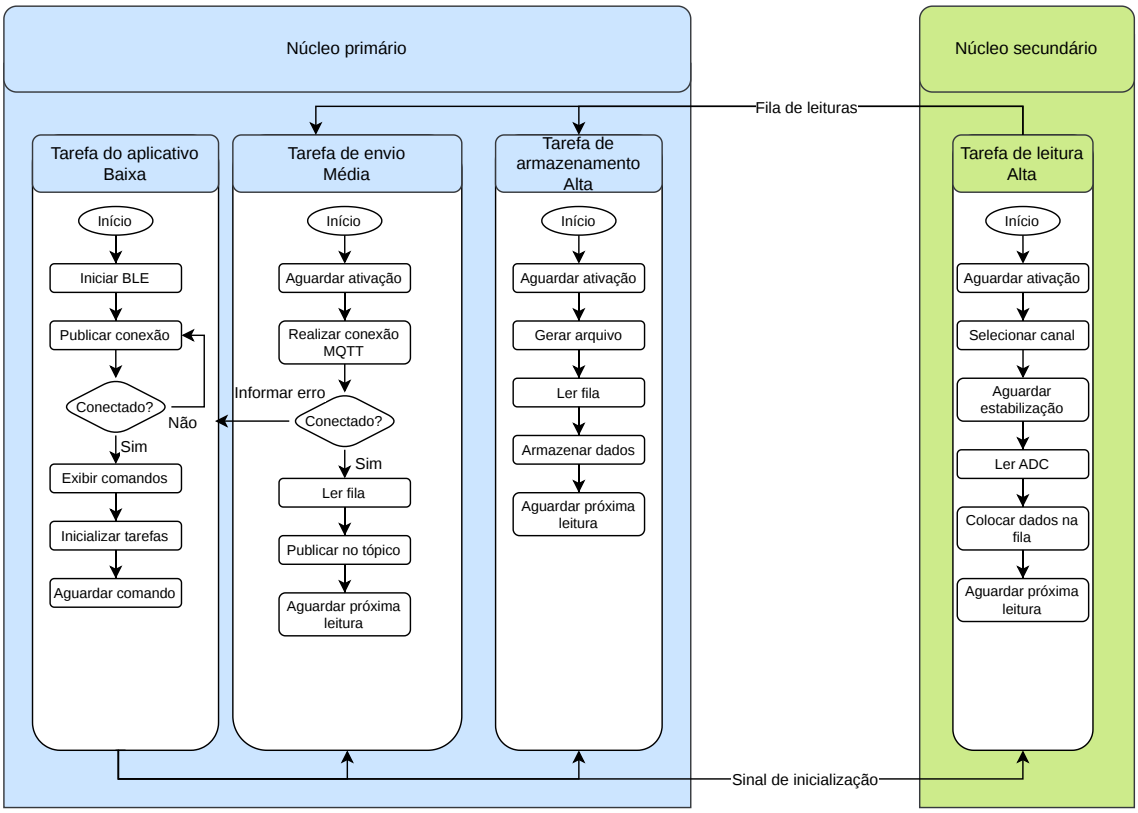
A tarefa de envio, apresentada no Apêndice C, corresponde a um novo desenvolvimento. Ela é executada no núcleo primário, com prioridade inferior às tarefas de leitura e armazenamento, e opera com periodicidade de 100 Hz. Nessa tarefa, ocorre inicialmente o estabelecimento da conexão com o *broker* MQTT e, posteriormente, o envio dos dados em lotes de até 32 leituras. Neste caso, foi escolhida a plataforma HiveMQ *Serverless*[1] para ser o *broker*, pois oferece, em seu plano gratuito, 100 conexões e 10 gigabytes de utilização mensal, capacidade suficiente para as necessidades do projeto.

Por fim, a tarefa do aplicativo é executada a cada segundo, possui a menor prioridade e também é alocada no núcleo primário. Essa tarefa implementa uma máquina de estados responsável por disponibilizar ao usuário as funções adequadas em cada etapa de operação. Por exemplo, após a inicialização do dispositivo, a tarefa de leitura não é ativada imediatamente, sua execução é liberada apenas após a confirmação do usuário para iniciar a leitura temporizada ou contínua. O código é apresentado no Apêndice D.

Um diagrama de fluxo simplificado abordando a alocação das tarefas por núcleo e a sua comunicação é apresentado na Figura 12.

[1] Disponível em: https://www.hivemq.com/agent/

Figura 12 – Diagrama de fluxo dos núcleos primário e secundário



Fonte: Autoria própria (2026).

4.2 Aplicativo móvel

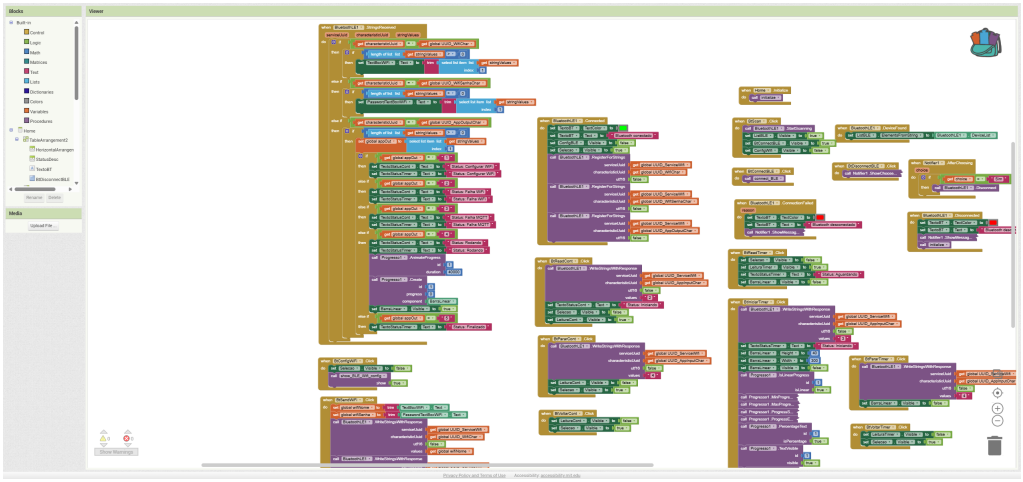
Para o desenvolvimento de um aplicativo móvel capaz de realizar a conexão Bluetooth com o dispositivo para o controle das leituras e configuração da rede, utilizou-se da plataforma disponibilizada pelo *Massachusetts Institute of Technology* chamada de *App Inventor*[2]. Essa plataforma permite a programação em blocos e, por meio da biblioteca BLE, possibilita a leitura e a escrita de características Bluetooth utilizadas para transmitir informações de configuração da rede, comandos de operação do dispositivo e o estado atual da máquina de estados. Além disso, a plataforma permite a criação da interface gráfica do aplicativo, por meio da inserção de botões, imagens, textos e outros componentes visuais. A plataforma também oferece a depuração por meio do aplicativo *Companion* e a compilação de arquivos executáveis para sistemas Android. A Figura 13 demonstra um exemplo da plataforma *App Inventor*.

Dessa forma, o usuário realiza a busca por dispositivos Bluetooth e seleciona o alvo para estabelecer a conexão, como demonstrado na Figura 14.

Após o estabelecimento da conexão, pode-se escolher entre quatro botões disponíveis na tela de menu: Configurar WiFi, Iniciar leitura contínua, Iniciar leitura temporizada e Desconectar dispositivo, apresentados na Figura 15.

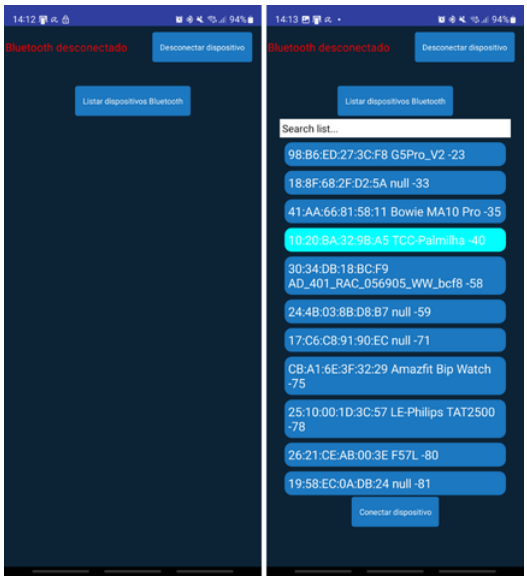
[2] Disponível em: https://appinventor.mit.edu/

Figura 13 – Exemplo da programação no *App Inventor*



Fonte: Autoria própria (2026).

Figura 14 – Processo de conexão Bluetooth



Fonte: Autoria própria (2026).

Ao escolher a opção “Configurar Wi-Fi”, o usuário é direcionado a uma tela com dois campos de texto destinados ao preenchimento do nome da rede Wi-Fi e da senha, permitindo que o dispositivo se conecte à rede e envie os dados ao *broker* MQTT. Essa configuração é necessária somente na primeira inicialização do dispositivo, uma vez que os dados são armazenados em memória não volátil e recuperados sempre que o dispositivo retorna do modo de hibernação. A Figura 16 apresenta a tela de configuração do Wi-Fi.

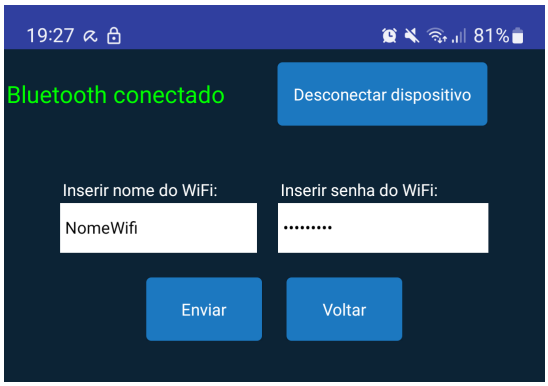
Ao retornar para o menu, o usuário pode escolher o modo de leitura contínua. Após o acionamento do botão, o aplicativo passa a exibir o estado atual do processo, além de disponibilizar as opções de interromper a leitura ou retornar ao menu anterior conforme apresentado na Figura 17. Caso ocorra alguma falha relacionada à conexão com a rede Wi-Fi ou ao *broker* MQTT, a informação é apresentada no campo de texto presente.

Na tela de leitura temporizada, pode-se iniciar a medição, interromper e voltar ao menu. Além disso, uma barra de progresso é exibida para indicar o tempo restante até a coleta to-

Figura 15 – Menu do aplicativo móvel

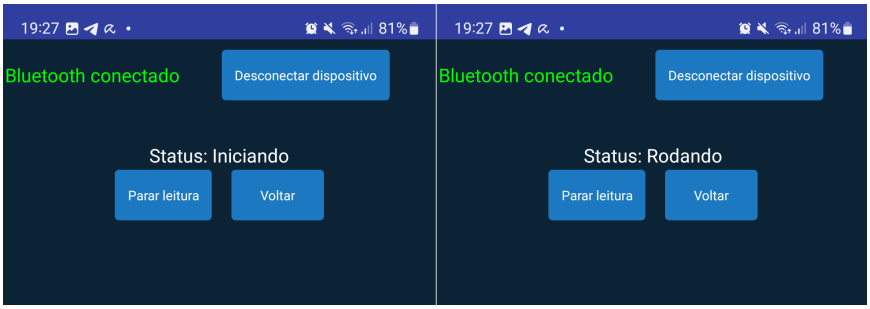
Fonte: Autoria própria (2026).

Figura 16 – Tela de configuração do Wi-Fi no aplicativo móvel



Fonte: Autoria própria (2026).

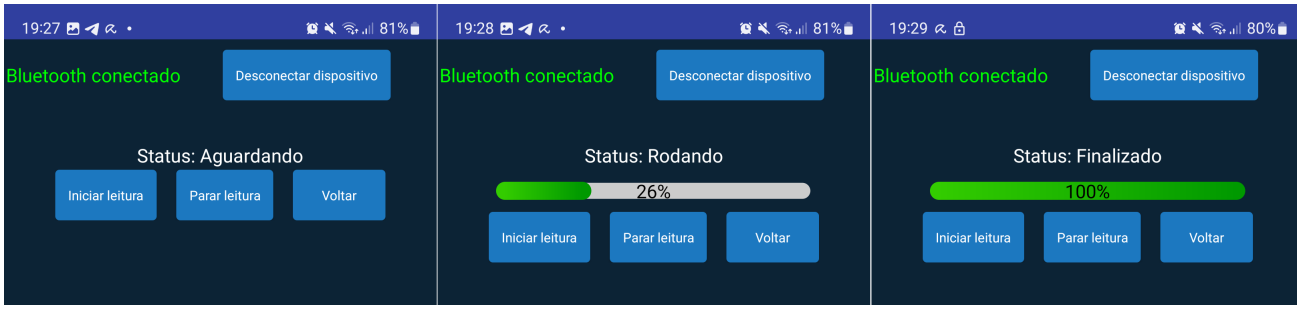
Figura 17 – Tela de leitura contínua no aplicativo móvel



Fonte: Autoria própria (2026).

tal de dados. Conforme a Figura 18, ao atingir 100%, o estado é alterado para finalizado e o usuário permanece na tela, podendo iniciar uma nova medição. Cada medição é salva no cartão microSD em um arquivo nomeado de acordo com o tipo de leitura e com um contador correspondente ao número da medição.

Figura 18 – Tela de leitura temporizada no aplicativo móvel



Fonte: Autoria própria (2026).

4.3 Processo de calibração

Para a obtenção da curva de calibração, adotou-se como referência o equipamento comercial EPS+R, da Loran Engineering®, marca pertencente ao grupo LetSense[3], mostrado na Figura 19. A escolha desse procedimento esteve relacionada à necessidade de comparar a resposta dos sensores resistivos da palmilha desenvolvida com um sistema de medição já consolidado para análise da pressão plantar. A partir dos dados obtidos pelo equipamento de referência, foi possível relacionar a condutância dos sensores da palmilha aos valores de pressão medidos durante os ensaios. As especificações do equipamento são apresentadas no Quadro 4.

Figura 19 – Baropodômetro EPS+R

Fonte: Autoria própria (2026).

O processo para aquisição foi estabelecido com base na norma ABNT NBR 8197:2021 e em manuais de fabricantes como a TekScan®. Para o procedimento, foram aplicadas forças na escala de 0% até 120% da faixa de leitura do sensor FS-INS-16Z, sendo incrementadas em 5% a cada medição e utilizando anilhas metálicas, conforme a Figura 20. Dessa forma, inicia-se a calibração com 0 kg chegando até 12 kg com incrementos de 500 g entre medições.

No primeiro ensaio, o posicionamento de massas inferiores a 5 kg não resultou em pressão suficiente para que o EPS+R realizasse a medição. Além disso, os valores de pressão estavam atrelados à área identificada pelo equipamento, não sendo condizentes com a área

[3] http://www.letsense.net/

Quadro 4 – Especificações do EPS+R

Característica	
Tipo sensor	Resistivo
Número de sensores	2304
Faixa de pressão	50- 350 kPa
Frequência	50 - 100 Hz
Conexão	USB

Fonte: Autoria própria (2026).

Figura 20 – Procedimento da coleta de dados no baropodômetro

Fonte: Autoria própria (2026).

sensora dos elementos da palmilha FS-INS-16Z. Sendo assim, desenvolveu-se um suporte com as mesmas dimensões dos elementos da palmilha, visando sustentar os pesos e padronizar a área de contato percebida pelo EPS+R para a sua estimativa da pressão média.

No Quadro 5 são apresentados os valores obtidos em três processos de medição e a média aritmética dos dados. Após a obtenção dos dados do EPS+R, é necessário realizar o mesmo procedimento para a palmilha FS-INS-16Z. Com o objetivo de aumentar a confiabilidade das conexões do protótipo da prova de conceito e aproximar o protótipo de uma versão mais adequada à industrialização de produto, fabricou-se uma placa de circuito impresso para realizar a acomodação dos componentes eletrônicos.

No entanto, como apresentado na Figura 22, foi identificada uma incompatibilidade de espaçamento entre o módulo do microSD e o módulo TP4056, utilizado para carregar baterias de lítio, causando uma inclinação no microSD. Além disso, também é perceptível, na Figura 21, uma ligação cabeada laranja no módulo microSD. Tal adaptação foi necessária, pois o módulo utiliza deslocadores de nível de 5 V para 3,3 V em todos os pinos de comunicação SPI. Neste caso, ocorria a interferência no pino MISO, deixando inutilizável o barramento. Para o caso do

Quadro 5 – Resultados obtidos na coleta de dados do EPS+R

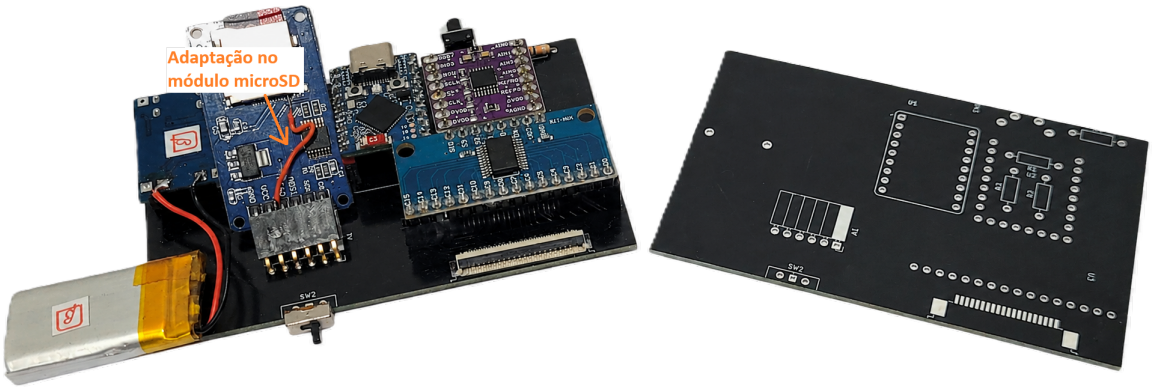
Massa [kg]	Pressão média medida 1 [kPa]	Pressão média medida 2 [kPa]	Pressão média medida 3 [kPa]	Média das medições [kPa]
0,0	-	-	-	-
0,5	-	-	-	-
1,0	-	-	-	-
1,5	-	-	-	-
2,0	31,1	35,3	36,8	34,4
2,5	54,2	56,2	57,1	55,8
3,0	89,8	96,2	99,0	95,0
3,5	99,3	101,8	109,7	103,6
4,0	139,6	144,8	118,5	134,3
4,5	102,3	103,4	104,3	103,3
5,0	135,3	137,9	139,4	137,5
5,5	119,3	119,6	119,9	119,6
6,0	141,6	123,3	123,6	129,5
6,5	146,3	147,9	148,0	147,4
7,0	141,8	142,7	143,5	142,7
7,5	148,4	148,9	149,6	149,0
8,0	158,2	160,2	160,7	159,7
8,5	163,4	163,9	164,6	164,0
9,0	165,4	166,7	167,3	166,5
9,5	171,3	172,6	172,5	172,1
10,0	173,0	173,5	173,9	173,5
10,5	172,7	173,7	173,9	173,4
11,0	184,2	186,2	186,8	185,7
11,5	190,2	190,7	190,8	190,6
12,0	194,8	195,3	195,7	195,3

Fonte: Autoria própria (2026).

ESP32, sua tensão lógica é de 3,3 V não sendo prejudicial realizar essa adaptação. A Figura 22 apresenta a palmilha com o suporte e conectada à placa de circuito impresso para a coleta de dados.

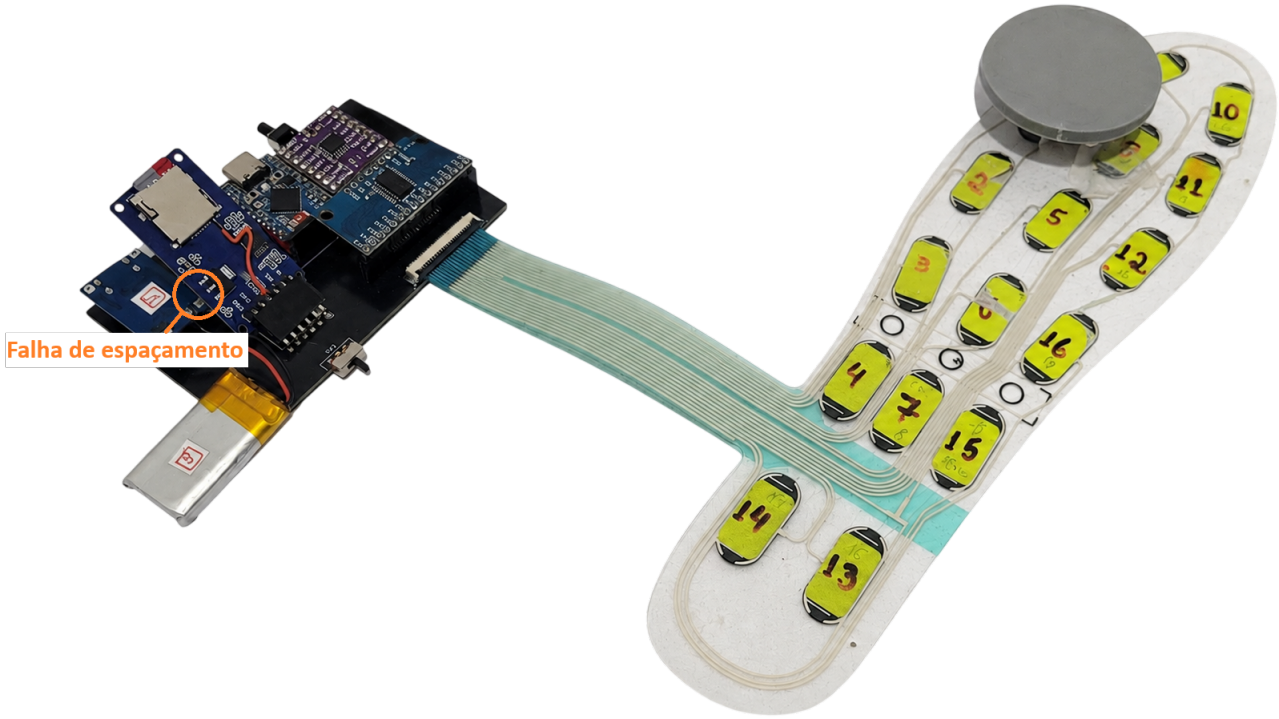
Posterior à coleta de dados da palmilha, realizaram-se duas comparações: a primeira entre os dados obtidos com a massa teórica utilizada; e a segunda com o equipamento comer-

Figura 21 – Placa de circuito impresso com componentes e vazia



Fonte: Autoria própria (2026).

Figura 22 – Placa de circuito impresso com palmilha conectada e suporte



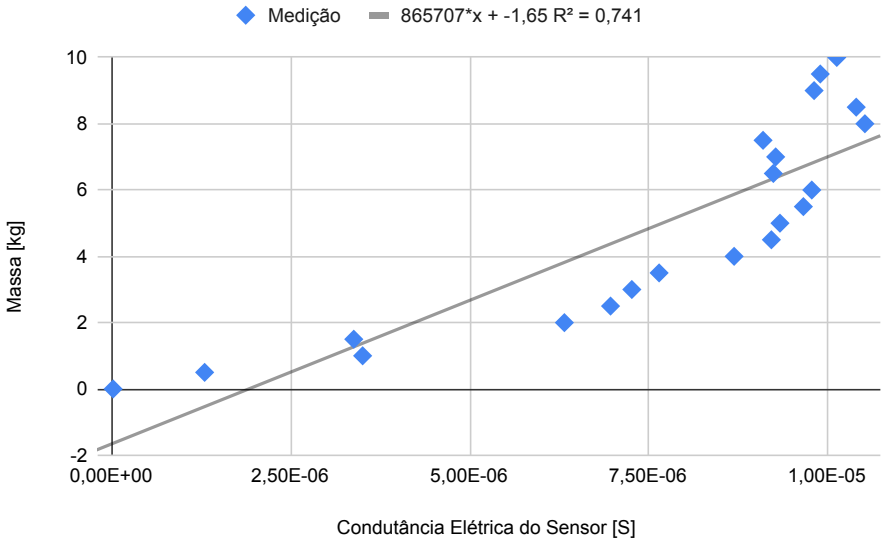
Fonte: Autoria própria (2026).

cial EPS+R. Para sensores de força resistivos, é mais simples a utilização da condutância para calibração e seu valor é proporcional à força aplicada (Tekscan, c2026).

Sendo assim, os valores de condutância média foram inseridos no eixo x e a massa utilizada no eixo y. A partir do gráfico gerado, obtém-se a equação de tendência linear e polinomial, apresentadas no Gráfico 1 e no Gráfico 2, respectivamente, e o fator R^2 de cada ajuste.

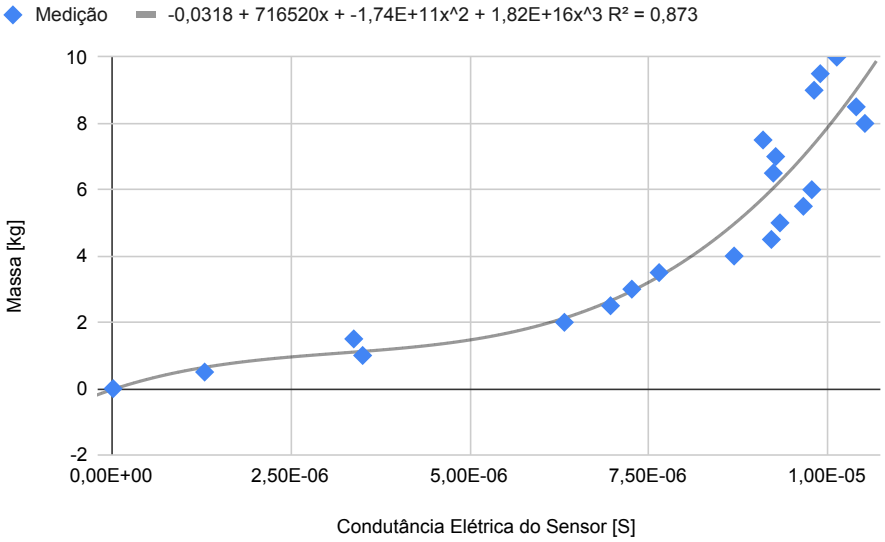
Em uma análise preliminar, observou-se que as medições superiores a 10 kg não apresentam crescimento significativo além de ruídos, pois a especificação da palmilha FS-INS-16Z atua na faixa nominal entre 500 g e 10 kg. Dessa forma, os pontos acima desse limite passaram a interferir negativamente na obtenção das equações de tendência. Com a remoção das

Gráfico 1 – Medições de condutância com base na massa utilizada com modelo linear



Fonte: Autoria própria (2026).

Gráfico 2 – Medições de condutância com base na massa utilizada com modelo polinomial



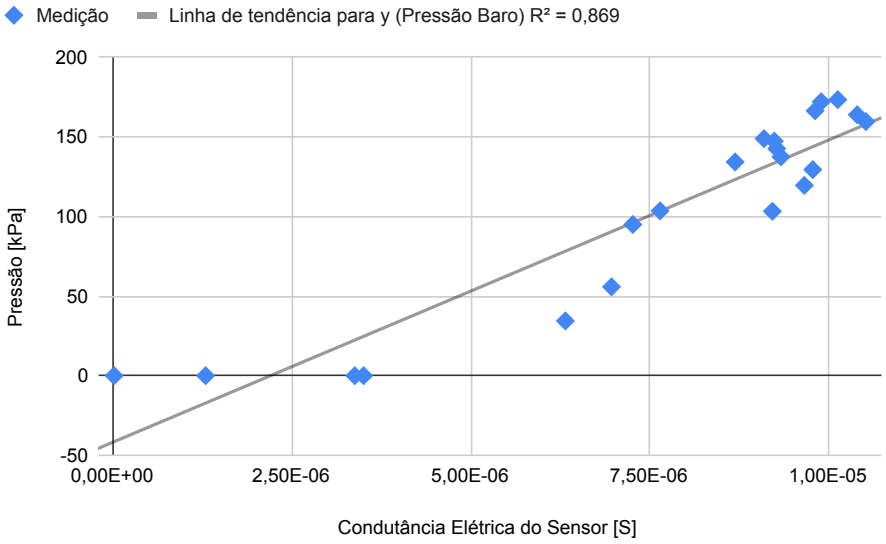
Fonte: Autoria própria (2026).

medições superiores a 10 kg, os modelos linear e polinomial apresentaram fatores R^2 de 0,741 e 0,873, respectivamente. Com base nesse resultado, os valores superiores a 10 kg também foram descartados na análise comparativa com o baropodômetro EPS+R.

Seguindo para a segunda comparação, geram-se novamente os gráficos para obtenção da equação de tendência linear e polinomial, apresentados no Gráfico 3 e no Gráfico 4.

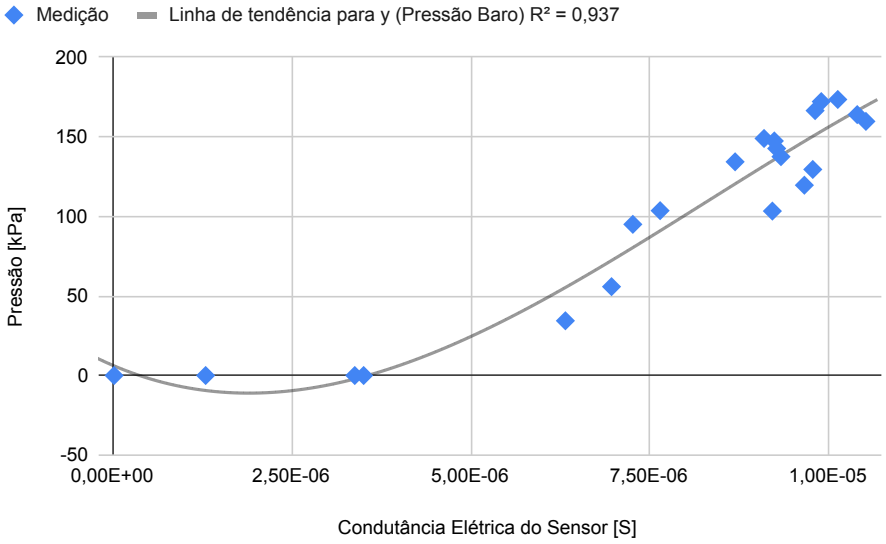
Com base nos resultados obtidos, verifica-se que o modelo linear apresenta boa correspondência em regiões próximas ao fundo da escala, entretanto, não representa de forma satisfatória em seu início. Por outro lado, o modelo polinomial satisfaz boa parte da escala, apresentando um fator R^2 de 0,937. Com isso, escolheu-se prosseguir com a utilização do modelo polinomial, sendo necessário realizar a definição do seu grau.

Gráfico 3 – Medições de condutância com base na pressão do baropodômetro com modelo linear



Fonte: Autoria própria (2026).

Gráfico 4 – Medições de condutância com base na pressão do baropodômetro com modelo polinomial



Fonte: Autoria própria (2026).

A partir da aplicação de polinômios de diferentes graus, verificou-se que modelos superiores ao quarto grau apresentaram ganhos no fator R^2 apenas na terceira casa decimal, não justificando o aumento de complexidade. Além disso, a diferença do fator R^2 entre os modelos de segundo e terceiro grau foi de 0,007 enquanto o gasto computacional incrementa de quatro operações de ponto flutuante para seis. Considerando 16 elementos sensores, com taxa de aquisição de 100 Hz, seriam realizadas 6.400 operações no modelo de segundo grau contra 9.600 no terceiro, o que representa 50% a mais de gasto computacional.

Com base nessa análise, adotou-se como modelo final o polinômio de segundo grau, apresentado a seguir:

$$P(x) = -4{,}77 - 2{,}62 \cdot 10^{6} \cdot x + 1{,}90 \cdot 10^{12} \cdot x^{2}, \tag{7}$$

em que x denota a condutância medida a partir da leitura do sensor, e $P(x)$ corresponde à pressão plantar estimada pelo modelo, expressa em kPa. Dessa forma, cada valor de condutância medido pelo sistema é convertido em uma estimativa de pressão, permitindo a posterior geração dos mapas de calor.

4.4 Software de visualização

O desenvolvimento do software de visualização em ambiente *MATLAB* teve como objetivo criar uma ferramenta científica capaz de ler, processar, armazenar e renderizar dados de pressão plantar provenientes da matriz de 16 sensores. O sistema foi projetado para operar em dois domínios distintos: o processamento offline, por meio da leitura em lote de dados estruturados em cartão microSD; e o monitoramento em tempo real, realizado via protocolo de comunicação MQTT. Durante o ciclo de desenvolvimento, diversas decisões arquiteturais e correções matemáticas foram fundamentais para garantir a fluidez da interface gráfica e a confiabilidade dos dados gerados.

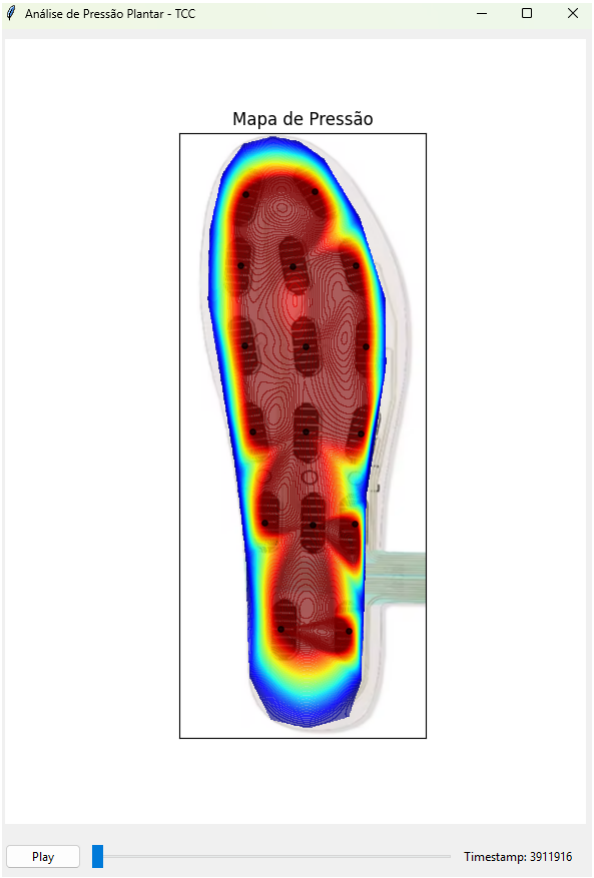
Antes da consolidação da arquitetura definitiva em *MATLAB*, foi desenvolvida uma versão preliminar do sistema utilizando a linguagem de programação *Python*, demonstrada na Figura 23. O objetivo principal desse protótipo inicial consistiu em validar a viabilidade da interpolação de dados baropodométricos sobre uma visualização de anatomia plantar.

Embora o protótipo desenvolvido em *Python* tenha validado com sucesso o algoritmo de interpolação, por meio da função *griddata* de forma estática, a transição para um sistema operacional real e clínico impôs barreiras técnicas associadas à biblioteca gráfica *Tkinter* e ao ecossistema *Matplotlib*[4]. A primeira limitação esteve relacionada ao gerenciamento de concorrência e ao bloqueio de loop causado pelo *Global Interpreter Lock* do *Python*. Essa característica restringia o desempenho quando o sistema precisava processar simultaneamente funções de escuta de portas de rede de alta frequência, como MQTT ou *User Datagram Protocol* (UDP), e disparar chamadas assíncronas de interface, como o *root.after()*, resultando em travamentos constantes na execução do *root.mainloop()*.

Além disso, observou-se latência significativa de renderização e sobrecarga de código. Embutir e atualizar mapas de calor vetoriais gerados pelo *matplotlib.figure* dentro de contêineres de *widgets* nativos do *Tkinter*, por meio do *FigureCanvasTkAgg*, resultavam em latência computacional elevada. Consequentemente, customizações dinâmicas simples, como barras de cores interativas e filtros de suavização em tempo real, exigiam scripts extensos e de difícil otimização.

[4] Disponível em: https://matplotlib.org/

Figura 23 – Protótipo inicial em *Python*/*Tkinter*



Fonte: Autoria própria (2026).

Diante destas limitações, optou-se pela migração completa para o ambiente *MATLAB*. Esse ecossistema oferece um compilador e renderizador nativamente otimizados para álgebra matricial e computação vetorial, além de facilitar o tratamento matemático imediato de sinais e componentes visuais. A Figura 24 demonstra o software no ambiente *MATLAB*.

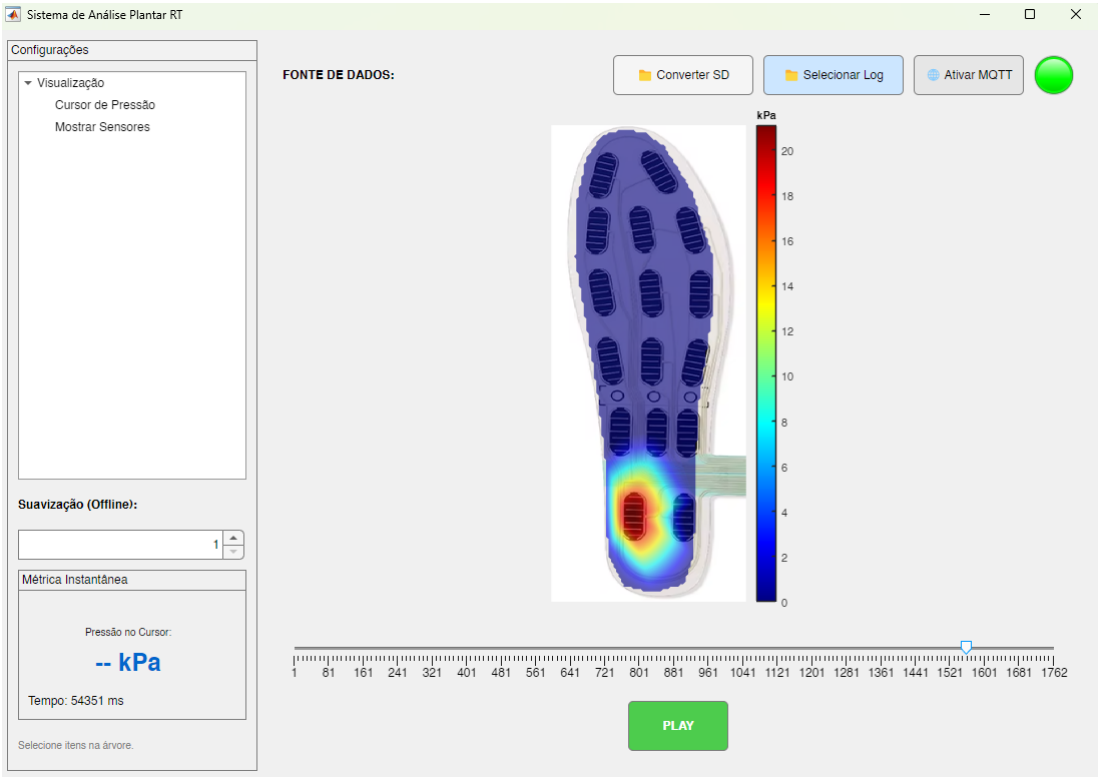
Um dos problemas mais críticos encontrados na fase inicial do processamento de dados foi a inconsistência na conversão de tensão elétrica para pressão física em kPa. Durante os testes com cargas leves, o software registrava pressão nula, ao passo que o sistema apresentava pressões residuais quando os sensores se encontravam em completo repouso.

A curva de calibração inicial utilizava um polinômio de terceiro grau. O software de regressão forçou a curva a interceptar todos os pontos experimentais de teste, gerando um efeito de *overfitting* e criando uma oscilação matemática na zona de baixa condutância. Ao inserir a condutância decorrente de um toque leve na equação, os termos de ordem superior negativos superavam os positivos, resultando em pressões calculadas negativas que o sistema truncava para zero. Por outro lado, em repouso absoluto, a equação retornava valores positivos artificiais.

Para mitigar essa instabilidade numérica, a regressão foi ajustada para um polinômio estável de segundo grau, parametrizada de acordo com a Equação 7.

O coeficiente inicial negativo passou a atuar como um filtro natural de ruído eletrônico de fundo. Em estado de repouso, a equação resulta em um valor negativo, o qual não possui significado físico para a aplicação. Dessa forma, foi implementada uma trava de segurança

Figura 24 – Protótipo final no *MATLAB*



Fonte: Autoria própria (2026).

algorítmica, apresentada na Equação 8, que limita a pressão final ao maior valor entre a pressão calculada pelo modelo e zero.

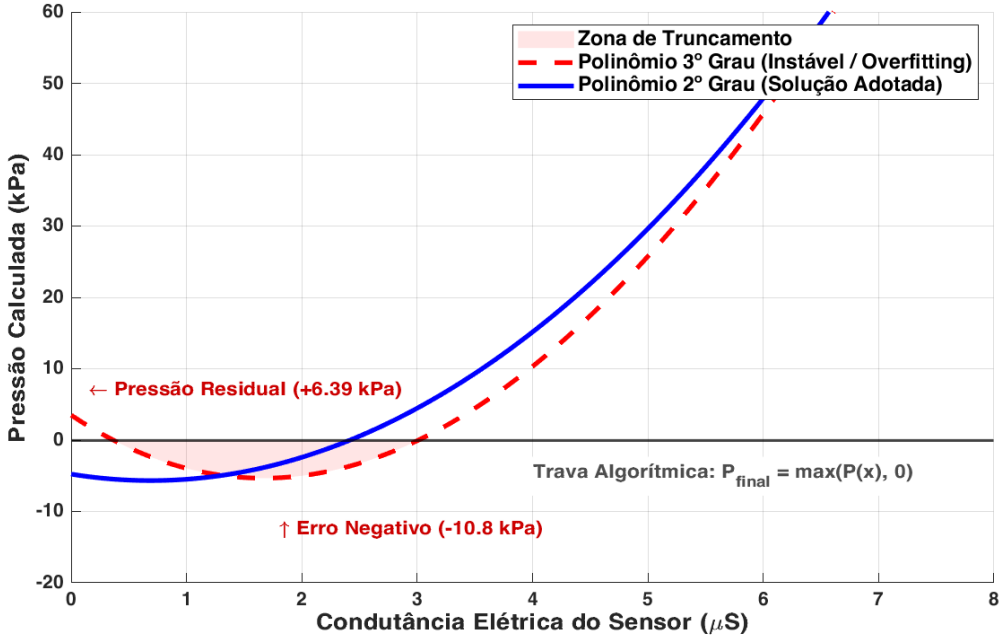
$$P_{final} = \max(P(x), 0), \tag{8}$$

em que $P(x)$ representa a pressão estimada pelo modelo de calibração a partir da condutância x medida pelo sensor, enquanto P_{final} corresponde ao valor de pressão efetivamente utilizado pelo software de visualização. A Figura 25 demonstra a pressão calculada com a utilização do modelo de segundo e terceiro grau bem como a zona de truncamento gerada pela trava algorítmica.

Focando na usabilidade visual, foi implementado um controle de reprodução (*Play/-Pause*) para o modo offline e um botão de alternância de conexão para o modo ao vivo. No entanto, o *MATLAB* apresenta por padrão um comportamento síncrono *single-thread* que congela as interações da interface gráfica do usuário durante a execução de laços de repetição. Devido a este bloqueio, os comandos de pausa eram ignorados pelo sistema operacional enquanto a malha geométrica do pé era renderizada continuamente em loop.

A solução encontrada consistiu em injetar o comando *drawnow* dentro da estrutura interna de repetição. Esse comando força o interpretador do *MATLAB* a interromper temporariamente a execução matemática por uma fração de milissegundo, permitindo realizar duas tarefas essenciais de forma encadeada. Primeiramente, ele esvazia a fila de eventos do sistema operacional, capturando e registrando os cliques de mouse feitos pelo usuário. Imediatamente após, possibilita a atualização e a renderização gráfica dos componentes em tela.

Figura 25 – Comparativo das curvas de calibração



Fonte: Autoria própria (2026).

Adicionalmente, a substituição de soluções personalizadas pelo uso do componente nativo *colorbar*, atrelado diretamente aos eixos geométricos, garantiu uma escalabilidade visual automatizada e indexada estritamente aos limites de tensão.

No que tange à aquisição de dados em tempo real, o projeto original previa a conexão direta do software *MATLAB* ao *broker* MQTT de forma nativa. Contudo, essa abordagem direta revelou entraves técnicos e estruturais significativos durante a fase de integração.

O primeiro obstáculo referiu-se à dependência de pacotes adicionais. A implementação de clientes MQTT nativos no *MATLAB* exige o licenciamento de *toolboxes* específicas, o que reduz a portabilidade do software e impõe restrições de uso em ambientes acadêmicos ou máquinas sem licenças estendidas.

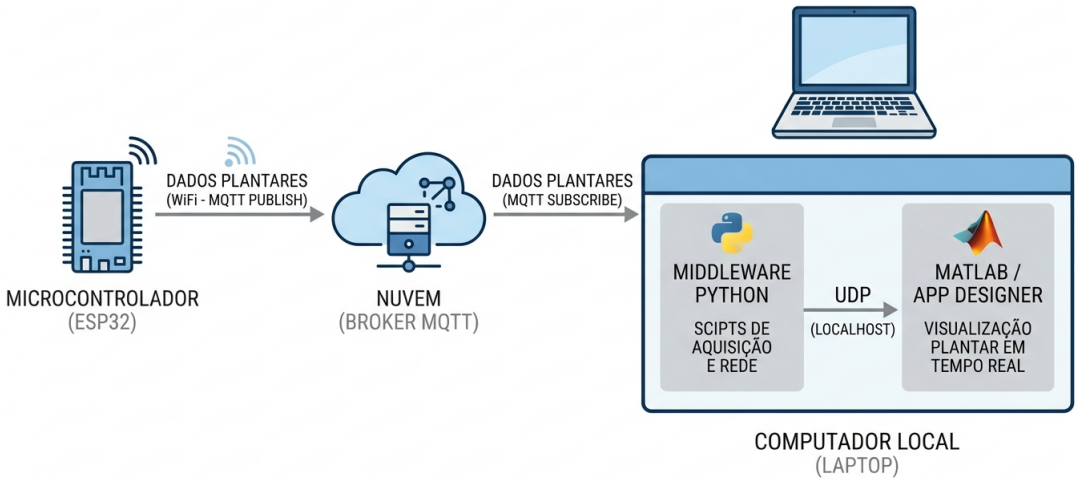
O segundo e mais crítico obstáculo consistiu no gerenciamento de concorrência e rede. O protocolo MQTT exige o tratamento contínuo de *callbacks* assíncronos, mensagens de reconhecimento e verificações de integridade de conexão. Inserir essa carga de processamento de rede diretamente na única *thread* principal do MATLAB, que já estava ocupada com o cálculo matricial e a renderização contínua, resultava em perda de pacotes e instabilidade na comunicação durante altas taxas de amostragem.

Para solucionar essa problemática, desenvolveu-se uma arquitetura de comunicação em camadas como apresentado na Figura 26, delegando as operações de rede a um *middleware* desenvolvido em *Python*. Nessa arquitetura híbrida, a camada de borda é composta por um script *Python* operando em segundo plano, que assume o papel exclusivo de cliente MQTT. Devido à natureza assíncrona nativa de bibliotecas como *paho-mqtt*[5], o *Python* gerencia a co-

[5] Disponível em: https://pypi.org/project/paho-mqtt/

nexão com o *broker*, lida com eventuais quedas de sinal Wi-Fi do microcontrolador e recebe os *payloads* brutos sem latência gráfica. Imediatamente após receber a mensagem, entra em ação a ponte de transporte local, onde o *script* repassa a *string* de dados para o *MATLAB* através de um *socket* UDP operando internamente no computador. Por fim, na camada de aplicação, o *MATLAB* atua apenas como um ouvinte UDP. Como o suporte a *sockets* UDP é nativo nas funções base da plataforma, eliminou-se a necessidade de *toolboxes* pagas.

Figura 26 – Arquitetura do sistema de aquisição em tempo real



Fonte: Autoria própria (2026).

A escolha do UDP deve-se ao fato de ser um protocolo não orientado a conexão e desprovido de confirmação de recebimento. Com isso, o *MATLAB* atua apenas como consumidor dos dados disponíveis no *buffer* UDP, sem a necessidade de gerenciar o estabelecimento de conexão, mensagens de confirmação ou retransmissões. Essa estratégia contribui para manter a continuidade do processamento matemático e da renderização gráfica, reduzindo a interferência das oscilações da rede externa no desempenho da interface de visualização.

A última decisão envolveu a lógica de calibração da barra de cores do mapa de calor. No modo offline, a escala foi programada para identificar os limites mínimo e máximo absolutos contidos no arquivo fonte, assegurando um referencial estático.

Por outro lado, estabelecer um teto fixo no modo ao vivo não se evidenciou como o ideal, pois os limites muito altos ocultariam nuances de pisadas de pacientes leves, enquanto limites estritamente dinâmicos baseados no máximo instantâneo gerariam uma oscilação visual, causando a perda do referencial comparativo. A solução adotada consistiu no desenvolvimento de uma escala dinâmica, fundamentada em uma janela deslizante com análise estatística de percentil.

Para o seu funcionamento, o software inicialmente realiza o armazenamento do histórico, guardando de forma contínua o valor de pico máximo de pressão registrado em cada *frame* dentro de um vetor com tamanho fixo, limitado às últimas 50 iterações. Com esses dados, aplica-se

uma filtragem estatística: a cada novo *frame*, os valores do vetor são ordenados de forma crescente e, em vez de adotar o valor máximo absoluto da janela, calcula-se o percentil 95%. Esta operação funciona computacionalmente como um filtro passa-baixas, evitando que a interface visual apresente picos de ruído transiente. Por fim, implementou-se uma trava de piso de ruído, que consiste em uma restrição matemática para delimitar o encolhimento do limite superior da barra de cores V_{max}, conforme expresso pela Equação 9:

$$V_{\text{max}} = \max(30\,\text{kPa}, P_{95}). \tag{9}$$

A imposição do patamar mínimo de 30 kPa garantiu que, quando o usuário retirar completamente o pé da palmilha, a escala superior não convirja para próximo de zero. Sem essa restrição, pequenos ruídos eletromagnéticos de leitura na ordem de milivolts seriam indevidamente amplificados pela escala reduzida, colorindo o gráfico e prejudicando a interpretação clínica de ausência de carga. Assim, o protótipo desenvolvido posiciona-se como uma solução inicial para análise instrumental da pressão plantar, com potencial de evolução futura para aplicações mais completas, semelhantes às observadas em sistemas descritos na busca de anterioridades e nas pesquisas de patentes da etapa conceitual.

5 CONCLUSÃO

Com base no desenvolvimento realizado, verificou-se que o trabalho resultou em um protótipo funcional capaz de integrar as principais etapas necessárias para a análise instrumental da pressão plantar. O sistema integrou a aquisição de dados por sensores resistivos, armazenamento local em cartão microSD, comunicação sem fio, controle por aplicativo móvel e visualização dos dados por meio de mapas de calor, permitindo tanto o acompanhamento em tempo real quanto a análise posterior das medições.

O desenvolvimento do software embarcado, baseado em tarefas de tempo real, possibilitou a organização das funções de leitura, armazenamento, envio de dados e controle pelo aplicativo. Além disso, a interface de visualização desenvolvida em *MATLAB* permitiu representar graficamente a distribuição da pressão plantar, contribuindo para transformar os dados brutos dos sensores em uma forma visual mais intuitiva para análise.

O processo de calibração permitiu estabelecer uma relação entre a resposta da palmilha FS-INS-16Z e os valores obtidos com o equipamento comercial EPS+R. A análise dos modelos de ajuste demonstrou que o modelo polinomial apresentou melhor correspondência com os dados experimentais em comparação ao modelo linear, adotando o modelo polinomial de segundo grau como solução final, por apresentar melhor equilíbrio entre qualidade de ajuste e custo computacional.

Apesar dos resultados obtidos, o protótipo ainda apresenta limitações que devem ser consideradas. Entre elas, destacam-se a necessidade de correção da placa de circuito impresso, a adaptação realizada no módulo microSD, as limitações do processo de calibração decorrentes da aplicação não contínua de carga e da ausência de análise aprofundada de histerese, além da necessidade de validação com usuários em condições reais de uso. Assim, o sistema demonstrou viabilidade funcional e metrológica preliminar, mas ainda demanda aprimoramentos.

Como trabalhos futuros, recomenda-se o desenvolvimento de uma nova versão da placa de circuito impresso, com a correção das incompatibilidades identificadas e maior integração dos componentes eletrônicos. Também se propõe a expansão do sistema para comunicação entre pares de palmilhas, permitindo a aquisição simultânea dos dados dos pés direito e esquerdo e possibilitando análises comparativas da distribuição de pressão plantar. Além disso, sugere-se o aprimoramento do processo de calibração, com aplicações contínuas e controladas de ciclos de carga e descarga, a fim de avaliar a histerese, a repetibilidade e a estabilidade dos sensores. Por fim, recomenda-se a realização de ensaios com voluntários para a identificação de padrões associados aos diferentes tipos de pisada, como pé normal, pé plano e pé cavo, bem como a coleta de dados em diferentes condições de uso, incluindo caminhada, subida de escadas e atividades esportivas.

REFERÊNCIAS

ABDOLLAHI, M.; ZHOU, Q.; YUAN, W. Smart wearable insoles in industrial environments: A systematic review. **Applied Ergonomics**, Amsterdã, v. 118, n. 1, p. 104250, 2024.

ALMEIDA, J. S. *et al.* Comparação da pressão plantar e dos sintomas osteomusculares por meio do uso de palmilhas customizadas e pré-fabricadas no ambiente de trabalho. **Revista Brasileira de Fisioterapia**, São Carlos, v. 13, n. 6, p. 542–548, 2009.

ALSAFRAN, A. S. *et al.* **Smart shoes for diabetics**. Al-Ahsa, 2024. USPTO. Depositante: King Faisal University. Publicação número: US 2024/0164714 A1.

BACK, N. *et al.* **Projeto integrado de produtos**: planejamento, concepção e modelagem. Barueri: Manole, 2008. 648 p.

BALDAÇO, F. O. *et al.* Análise do treinamento proprioceptivo no equilíbrio de atletas de futsal feminino. **Fisioterapia e Movimento**, Curitiba, v. 23, n. 2, p. 183–192, 2010.

CARVALHO, J. C. V. d. O.; BERRO, C. D.; FIORELLI, A. Palmilhas posturais: Podoposturologia: uma revisão de literatura. **Revista Salusvita:** Ciências biológicas e da saúde, [s. l.], v. 40, n. 4, p. 82–87, 2021.

CAVANAGH, P. R.; HEWITT JR., F. G.; PERRY, J. In-shoe plantar pressure measurement: a review. **The foot**, University Park, v. 2, n. 4, p. 185–194, 1992.

CHOU, Y. S. **Smart gait analysis insoles**. Taipei, 2025. USPTO. Depositante: Decentralized Biotechnology Intelligence Co. Ltd.. Publicação número: US 2025/0151839 A1.

CODINHOTO, J. P. **Hermes: desenvolvimento de software para baropodômetro**. 2020. 66 f. Dissertação (Programa de Pós-Graduação em Engenharia Elétrica) — Universidade Estadual Paulista, Ilha Solteira, 2020.

COLCIONI, W. R. G. *et al.* **Palmilha sensorizada para auxílio à marcha de pacientes com hemiplegia**. Brasil, 2025. INPI. Publicação número: BR 102018015790-6 A2.

DataBase Center for Life Science. Dorsal view of bones of right foot. **TogoTV**. 2021a. Disponível em: https://togotv.dbcls.jp/togopic.2021.124.html. Acesso em: 19 jul. 2025.

DataBase Center for Life Science. Lateral view of bones of right foot. **TogoTV**. 2021b. Disponível em: https://togotv.dbcls.jp/togopic.2021.141.html. Acesso em: 19 jul. 2025.

DataBase Center for Life Science. Medial view of bones of right foot. **TogoTV**. 2021c. Disponível em: https://togotv.dbcls.jp/togopic.2021.139.html. Acesso em: 19 jul. 2025.

DENARDIN, G. W.; BARRIQUELO, C. H. **Sistemas operacionais de tempo real e sua aplicação em sistemas embarcados**. São Paulo: Blucher, 2019. 474 p.

FIGLIOLA, R.; BEASLEY, D. E. **Engenharia de medições**: princípios e aplicações. 4. ed. Rio de Janeiro: LTC, 2007. 482 p.

FINN, T. J. *et al.* **Integrated and predictive smart shoe insole**. Massachusetts, 2025. USPTO. Publicação número: US 2025/0302150 A1.

HAUSER, J. R.; CLAUSING, D. The house of quality. **Harvard Business Review**, [s. l.], v. 66, n. 3, p. 63–73, 1988.

HiveMQ. **MQTT Essentials**: the ultimate guide to the MQTT protocol for IoT messaging. 2. ed. Landshut: HiveMQ, c2025. 90 p. *E-book*. Disponível em: https://www.hivemq.com/mqtt/. Acesso em: 18 out. 2025.

Inmetro. **Guia para a expressão da incerteza e medição**. Rio de Janeiro: Inmetro, 2008. 141 p. Disponível em: https://www.gov.br/inmetro/pt-br/assuntos/metrologia-cientifica/documentos-tecnicos-em-metrologia/gum_final.pdf. Acesso em: 8 nov. 2025.

KO, T.; LIU, X. **Smart insole and application thereof**. Frederick, 2024. USPTO. Depositante: Tao Treasures LLC DBA Nanobiofab. Publicação número: US 2024/0277106 A1.

Legact. **FS-INS-16Z Array Sensor**. Shenzhen, c2025. Legact. Página da Web. Disponível em: https://film-sensor.com/. Acesso em: 19 jul. 2025.

LIPPERT, L. **Cinesiologia Clínica e Anatomia**. 5. ed. Rio de Janeiro: Guanabara Koogan, 2013.

MACWILLIAMS, B.; ARMSTRONG, P. F. Clinical applications of plantar pressure measurement in pediatric orthopedics. *In*: Pediatric Gait: A New Millennium in Clinical Care and Motion Analysis Technology. 2000, Chicago. **Anais [...]** Chicago: IEEE, 2000. p. 143–150.

MONTGOMERY, D. C.; RUNGER, G. C. **Applied Statistics and Probability for Engineers**. 6. ed. Nova Jersey: Wiley, 2014. 832 p.

PAHL, G. *et al.* **Projeto na engenharia**: fundamentos do desenvolvimento eficaz de produtos, métodos e aplicações. 1. ed. São Paulo: Blücher, 2005. 412 p.

RAZAK, A. H. A. *et al.* Foot plantar pressure measurement system: a review of wearable sensors for plantar pressure measurement. **Sensors**, Melbourne, v. 12, n. 7, p. 9884–9912, 2012.

SEGER, F. **Análise da influência de palmilhas personalizadas na distribuição das pressões plantares e no controlo postural**. 2017. 81 f. Dissertação (Mestrado em Engenharia Biomédica) — Faculdade de Engenharia, Universidade do Porto, Porto, 2017.

SOUZA, T. R. de *et al.* **Órtese plantar sensorizada e sistema para monitoramento e prevenção de pé diabético**. Vassouras, 2025. INPI. Depositante: Fundação Educacional Severino Sombra. Publicação número: BR 102023018835-4 A2.

Tekscan. **How Does a Force Sensing Resistor (FSR) Work**. Norwood, c2026. Tekscan. Página da Web. Disponível em: https://www.tekscan.com/blog/flexiforce/how-does-force-sensing-resistor-fsr-work. Acesso em: 16 maio 2026.

WIECZOREK, M. **Pressure sensing running shoe**. San Diego, 2025. USPTO. Publicação número: US 2025/0127259 A1.

APÊNDICE A – Código da tarefa de leitura

```
1   void vTaskLerSensores(void *pvParameters) {
2     TickType_t xLastWakeTime = xTaskGetTickCount();
3     Sensor_t valor;
4     float result = 0.0;
5     long longResult = 0;
6     uint8_t i = 0;
7     while (1) {
8       ulTaskNotifyTake(pdTRUE, portMAX_DELAY);
9       xLastWakeTime = xTaskGetTickCount();
10      runLeitura = true;
11      while (runLeitura) {
12        valor.tempo = millis();
13        xSemaphoreTake(spiMutex, portMAX_DELAY);
14        for (i = 0; i < 16; i++) {
15          SetMux(i);
16          valor.data[i] = Ads.getRawData();
17        }
18        xSemaphoreGive(spiMutex);
19        if (xQueueSend(qSensores, &valor, 0) != pdPASS){
20          Sensor_t lixo;
21          xQueueReceive(qSensores, &lixo, 0);
22          xQueueSend(qSensores, &valor, 0);
23        }
24        if (xQueueSend(qSd, &valor, 0) != pdPASS){
25          Sensor_t lixo;
26          xQueueReceive(qSd, &lixo, 0);
27          xQueueSend(qSd, &valor, 0);
28        }
29        vTaskDelayUntil(&xLastWakeTime, freq_sensores);
30      }
31    }
32  }
```

APÊNDICE B – Código da tarefa de armazenamento

```
1
2  void vTaskEnviarSd(void *pvParameters) {
3    const float VREF = 3.3, GAIN = 1.0, RANGE = 8388607.0;
4    File file;
5    const size_t MAX_BATCH = 32;
6    const size_t BUF_SIZE  = 8192;
7    Sensor_t items[MAX_BATCH];
8    char buf[BUF_SIZE];
9
10   while (1) {
11     ulTaskNotifyTake(pdTRUE, portMAX_DELAY);
12     runSD = true;
13     while (runSD) {
14       UBaseType_t numFila = uxQueueMessagesWaiting(qSd);
15       if (numFila == 0) {
16         vTaskDelay(freq_sd);
17         continue;
18       }
19       if (numFila > MAX_BATCH)
20         numFila = MAX_BATCH;
21
22       size_t n = 0;
23       while (n < numFila && xQueueReceive(qSd, &items[n], 0) == pdPASS) {
24         n++;
25       }
26       if (n == 0) {
27         vTaskDelay(freq_sd);
28         continue;
29       }
30
31       size_t pos = 0;
32       for (size_t k = 0; k < n; ++k){
33         pos += snprintf(&buf[pos], (pos < BUF_SIZE) ? (BUF_SIZE - pos) : 0, "%lu,",
     (unsigned long)items[k].tempo);
34         for (uint8_t i = 0; i < 16; ++i) {
35           float volts = (items[k].data[i] / RANGE) * (VREF / GAIN);
36           pos += snprintf(&buf[pos], (pos < BUF_SIZE) ? (BUF_SIZE - pos) : 0, "%.6f%
     s", volts, (i < 15 ? ", " : ""));
37         }
38         pos += snprintf(&buf[pos], (pos < BUF_SIZE) ? (BUF_SIZE - pos) : 0, ";\r\n")
     ;
39         if (pos > BUF_SIZE - 256) break;
40       }
41       if (xSemaphoreTake(spiMutex, portMAX_DELAY) == pdTRUE){
42         File file = SD.open(arquivo, FILE_APPEND);
43         if (file) {
```

```
44            file.write((const uint8_t*)buf, pos);
45            file.close();
46          }
47          xSemaphoreGive(spiMutex);
48        }
49        vTaskDelay(freq_sd);
50      }
51    }
52  }
```

APÊNDICE C – Código da tarefa de envio

```
1  void vTaskMqtt(void *pvParameters) {
2    const float VREF = 3.3, GAIN = 1.0, RANGE = 8388607.0;
3    Sensor_t valor;
4    TickType_t xLastWakeTime;
5    const size_t MAX_BATCH = 1;
6    const size_t BUF_SIZE  = 2048;
7    Sensor_t items[MAX_BATCH];
8    char buf[BUF_SIZE];
9
10   while (1)
11   {
12     ulTaskNotifyTake(pdTRUE, portMAX_DELAY);
13     runMQTT = true;
14     xLastWakeTime = xTaskGetTickCount();
15     while (runMQTT) {
16
17       if (!mqtt.connected()) {
18         Serial.println("[MQTT] Reconectando...");
19         if (WiFi.status() != WL_CONNECTED)
20           ConectaWifi(nomeWifi, senhaWifi, 20000);
21         ConectaMQTT(20000);
22         xLastWakeTime = xTaskGetTickCount(); // reseta após reconexão longa
23         continue;
24       }
25       mqtt.loop();
26
27       UBaseType_t numFila = uxQueueMessagesWaiting(qSensores);
28       if (numFila == 0) {
29         mqtt.loop();
30         vTaskDelay(pdMS_TO_TICKS(10));
31         continue;
32       }
33       if (numFila > MAX_BATCH)
34         numFila = MAX_BATCH;
35
36       size_t n = 0;
37       while (n < numFila && xQueueReceive(qSensores, &items[n], 0) == pdPASS) {
38         n++;
39       }
40       if (n == 0) {
41         mqtt.loop();
42         vTaskDelay(pdMS_TO_TICKS(10));
43         continue;
44       }
45
46       size_t pos = 0;
```

```
47        for (size_t k = 0; k < n; ++k){
48          pos += snprintf(&buf[pos], (pos < BUF_SIZE) ? (BUF_SIZE - pos) : 0, "%lu,",
      (unsigned long)items[k].tempo);
49          for (uint8_t i = 0; i < 16; ++i) {
50            float volts = (items[k].data[i] / RANGE) * (VREF / GAIN);
51            pos += snprintf(&buf[pos], (pos < BUF_SIZE) ? (BUF_SIZE - pos) : 0, "%.6f%
      s", volts, (i < 15 ? ", " : ""));
52          }
53          pos += snprintf(&buf[pos], (pos < BUF_SIZE) ? (BUF_SIZE - pos) : 0, ";\r\n")
      ;
54          if (pos > BUF_SIZE - 256) break;
55        }
56        bool ok = mqtt.publish(TOPIC_PUB, buf, pos);
57        Serial.printf("[MQTT] publish %d bytes | ok=%d | bufMax=%d | state=%d\n", pos,
       ok, MQTT_MAX_PACKET_SIZE, mqtt.state());
58        xTaskDelayUntil(&xLastWakeTime, freq_mqtt);
59      }
60    }
61  }
```

APÊNDICE D – Código da tarefa do aplicativo

```
1  void vTaskApp(void *pvParameters) {
2    uint32_t appInput = 0;
3    bool flagRunCont = false;
4    while (1) {
5      switch (estado) {
6        case ST_SLEEP:
7        {
8          EncerrarTasks();
9          DeepSleep();
10         break;
11       }
12
13       case ST_PAREANDO:
14       {
15         BLEDevice::startAdvertising();
16         uint32_t t0 = millis();
17         bool pareou = false;
18
19         while ((millis() - t0) < 60000) {
20           if (deviceConnected) {
21             pareou = true;
22             break;
23           }
24           Blink(0,0,10,250,1);
25         }
26
27         if (!pareou) {
28           estado = ST_SLEEP;
29           break;
30         }
31
32         estado = ST_CONECTADO;
33         Blink(0,5,0,500,3);
34
35         WifiNomeChar.setValue(nomeWifi.c_str());
36         WifiNomeChar.notify();
37
38         vTaskDelay(pdMS_TO_TICKS(500));
39
40         WifiSenhaChar.setValue(senhaWifi.c_str());
41         WifiSenhaChar.notify();
42
43         break;
44       }
45
46       case ST_CONECTADO:
```

```
47      {
48        if (xQueueReceive(qAppInput, &appInput, pdMS_TO_TICKS(100)) == pdPASS) {
49          switch (appInput) {
50            case APP_SET_WIFI:
51              if (nomeWifi.length() > 0 && senhaWifi.length() > 0) {
52                SalvaWifi(nomeWifi, senhaWifi);
53                Serial.println("[APP] Wi-Fi salvo no NVS.");
54              } else {
55                Serial.println("[APP] APP_SET_WIFI ignorado.");
56              }
57              break;
58
59            case APP_START_CONTINUOUS:
60              flagRunCont = false;
61              estado = ST_LEITURA;
62              break;
63
64            case APP_START_TIMER:
65              estado = ST_TIMER;
66              break;
67
68            case APP_STOP:
69              estado = ST_CONECTADO;
70              break;
71
72            case APP_DISCONNECT:
73              estado = ST_SLEEP;
74              break;
75
76            default:
77              Serial.println("[APP] Comando desconhecido.");
78              break;
79          }
80        }
81
82        if (!deviceConnected) {
83          estado = ST_PAREANDO;
84        }
85
86        break;
87      }
88
89      case ST_LEITURA:
90      {
91        if (nomeWifi.length() == 0 || senhaWifi.length() == 0) {
92          estado = ST_CONECTADO;
93          AppOutputChar.setValue("1");
94          AppOutputChar.notify();
```

```
95          break;
96        }
97
98        if (!ConectaWifi(nomeWifi, senhaWifi, 20000)) {
99          estado = ST_CONECTADO;
100         AppOutputChar.setValue("2");
101         AppOutputChar.notify();
102         break;
103       }
104
105       if (!ConectaMQTT(20000)) {
106         estado = ST_CONECTADO;
107         AppOutputChar.setValue("3");
108         AppOutputChar.notify();
109         break;
110       }
111
112       if (!flagRunCont) {
113         IniciarTasks();
114         CriarArquivo(true);
115         AppOutputChar.setValue("4");
116         AppOutputChar.notify();
117         flagRunCont = true;
118       }
119
120       if (!deviceConnected) {
121         EncerrarTasks();
122         estado = ST_PAREANDO;
123       }
124
125       vTaskDelay(pdMS_TO_TICKS(250));
126       break;
127     }
128
129     case ST_TIMER:
130     {
131       if (nomeWifi.length() == 0 || senhaWifi.length() == 0) {
132         estado = ST_CONECTADO;
133         AppOutputChar.setValue("1");
134         AppOutputChar.notify();
135         break;
136       }
137
138       if (!ConectaWifi(nomeWifi, senhaWifi, 20000)) {
139         estado = ST_CONECTADO;
140         AppOutputChar.setValue("2");
141         AppOutputChar.notify();
142         break;
```

```
143         }
144
145         if (!ConectaMQTT(20000)) {
146           estado = ST_CONECTADO;
147           AppOutputChar.setValue("3");
148           AppOutputChar.notify();
149           break;
150         }
151
152         IniciarTasks();
153         CriarArquivo(false);
154
155         AppOutputChar.setValue("4");
156         AppOutputChar.notify();
157
158         uint32_t startTime = millis();
159
160         while (millis() - startTime < 40000 && estado == ST_TIMER) {
161           if (!deviceConnected) {
162             estado = ST_PAREANDO;
163             break;
164           }
165
166           if (xQueueReceive(qAppInput, &appInput, pdMS_TO_TICKS(100)) == pdPASS) {
167             if (appInput == APP_STOP) {
168               estado = ST_CONECTADO;
169               break;
170             } else if (appInput == APP_DISCONNECT) {
171               estado = ST_PAREANDO;
172               break;
173             }
174           }
175         }
176
177         EncerrarTasks();
178         AppOutputChar.setValue("5");
179         AppOutputChar.notify();
180
181         estado = ST_CONECTADO;
182         break;
183       }
184
185       default:
186         vTaskDelay(pdMS_TO_TICKS(250));
187         break;
188     }
189
190     Serial.print("[APP] Estado atual: ");
```

```
191     Serial.println(estado);
192
193     vTaskDelay(pdMS_TO_TICKS(1000));
194   }
195 }
```