UNIVERSIDADE DE SÃO PAULO
ESCOLA DE ENGENHARIA DE SÃO CARLOS

Eliel Santos do Espírito Santo

Desenvolvimento de um sistema embarcado de baixo custo para aquisição e transmissão de dados de sensores de força

São Carlos

2025

Eliel Santos do Espírito Santo

Desenvolvimento de um sistema embarcado de baixo custo para aquisição e transmissão de dados de sensores de força

Monografia apresentada ao Curso de Engenharia Elétrica com Ênfase em Eletrônica, da Escola de Engenharia de São Carlos da Universidade de São Paulo, como parte dos requisitos para obtenção do título de Engenheiro Eletricista.

Orientador: Prof. Dr. Alberto Cliquet Júnior

São Carlos

2025

Ficha catalográfica elaborada pela Biblioteca Prof. Dr. Sérgio Rodrigues Fontes da EESC/USP com os dados inseridos pelo(a) autor(a).

Santo, Eliel Santos do Espírito
S237d Desenvolvimento de um sistema embarcado de baixo custo para aquisição e transmissão de dados de sensores de força / Eliel Santos do Espírito Santo; orientador Alberto Cliquet Júnior. São Carlos, 2025.

Monografia (Graduação em Engenharia Elétrica com ênfase em Eletrônica) -- Escola de Engenharia de São Carlos da Universidade de São Paulo, 2025.

1. Sistema Embarcado. 2. ESP32. 3. Sensor de Força. 4. Tecnologia Assistiva. 5. Internet das Coisas. I. Título.

Eduardo Graziosi Silva - CRB - 8/8907

FOLHA DE APROVAÇÃO

Nome: Eliel Santos do Espirito Santo

Título: "Desenvolvimento de um sistema embarcado de baixo custo para aquisição e transmissão de dados de sensores de força"

Trabalho de Conclusão de Curso defendido e aprovado em 27/11/2025,

com NOTA 10,0 (dez), pela Comissão Julgadora:

Prof. Titular Alberto Cliquet Júnior - Orientador - SEL/EESC/USP

Dr. Orivaldo Lopes da Silva - EESC/USP

Dr. Gabriel Augusto Ginja - EESC/USP

Coordenador da CoC-Engenharia Elétrica - EESC/USP:
Professor Associado José Carlos de Melo Vieira Júnior

FOLHA DE AVALIAÇÃO

Nome: Eliel Santos do Espirito Santo
Título: "Desenvolvimento de um sistema embarcado de baixo custo para aquisição e transmissão de dados de sensores de força."

Trabalho de Conclusão de Curso apresentado à
Escola de Engenharia de São Carlos da Universidade de São Paulo.
Curso de Engenharia Elétrica – Ênfase em Eletrônica.

BANCA EXAMINADORA

Prof. Titular Alberto Cliquet Júnior - Orientador - SEL/EESC/USP

Nota atribuída: 10,0 (dez) _____
assinatura

Dr. Orivaldo Lopes da Silva - EESC/USP

Nota atribuída: 10,0 (dez) _____
assinatura

Dr. Gabriel Augusto Ginja – EESC/USP

Nota atribuída: 10,0 (dez) _____
assinatura

Média: 10,0 (dez)

Resultado: APROVADO
(APROVADO/REPROVADO/RECUPERAÇÃO)

Data: 27/11/2025

À minha família, pelo amor inabalável; aos meus amigos, pelo apoio constante; e aos meus mestres, pelo conhecimento compartilhado. Dedico este trabalho a todos vocês, que foram a inspiração e a força essenciais para transformar este projeto em realidade.

AGRADECIMENTOS

A Deus, minha mais profunda gratidão, por ser a força que me sustentou, pela sabedoria que me guiou e pela oportunidade de transformar este desafio em realidade.

Aos meus amados pais e irmão, que foram meu alicerce. Agradeço por cada palavra de incentivo, pelo sacrifício silencioso e por garantirem que eu tivesse a tranquilidade e as condições para me dedicar a este sonho. Esta conquista é tão de vocês quanto minha.

Manifesto um agradecimento especial ao meu orientador, Prof. Dr. Alberto Cliquet Júnior. Sua paciência, conhecimento e orientação segura foram fundamentais para superar os desafios e dar forma a este projeto. Sou imensamente grato pela oportunidade e pela confiança em meu trabalho.

Aos meus amigos, companheiros desta jornada, agradeço por cada momento compartilhado, pelas conversas que aliviaram a pressão, pelo auxílio nos estudos e por tornarem a trajetória muito mais leve e significativa.

"A unidade de medida mais importante na engenharia é a monetária."
Prof. Dr. Marlon Rodrigues Garcia

RESUMO

ESPÍRITO SANTO, E. **Desenvolvimento de um sistema embarcado de baixo custo para aquisição e transmissão de dados de sensores de força**. 2025. 107 p. Monografia (Trabalho de Conclusão de Curso) - Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2025.

A presente obra abrange sobre o desenvolvimento de um sistema embarcado de baixo custo focando na obtenção, processamento e transmissão de dados de sensores resistivos de força (FSR), visando aplicações de tecnologia assistiva, como luvas de sensoriamento (SGS). Tendo como seu objetivo cerne a criação de recursos e ferramentas, tanto em *hardware* quanto em *software*, funcionais, escaláveis e economicamente acessíveis para o público geral, inspirando, por meio de uma estrutura didática, estudantes e pesquisadores a replicação do conteúdo. A metodologia envolve a construção de um circuito eletrônico baseado no microcontrolador ESP32, englobando diversos blocos de sua construção, buscando obter o máximo de qualidade e retorno do dispositivo. Para contornar a não-linearidade intrínseca do conversor analógico-digital (ADC) foi implementada uma correção via software baseada em *Lookup Table* (LUT), melhorando mensuravelmente a precisão das medições. O sistema transmite os dados calibrados simultaneamente por dois canais, via comunicação serial para aplicações locais, e via protocolo MQTT para monitoramento remoto, permitindo a sua associação em aplicações de Internet das Coisas (IoT) ou como dispositivo móvel. Como resultado, obteve-se um protótipo funcional e de baixo custo que oferece uma solução flexível não apenas para FSRs mas para diversos sensores resistivos, demonstrando grande potencial para generalização em projetos de engenharia.

Palavras-chave: Sistema Embarcado. ESP32. Sensor de Força. Tecnologia Assistiva. Internet das Coisas.

ABSTRACT

ESPÍRITO SANTO, E. **Development of a low-cost embedded system for force sensor data acquisition and transmission**. 2025. 107 p. Monograph (Conclusion Course Paper) - Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2025.

This work covers the development of a low-cost embedded system focusing on the acquisition, processing, and transmission of data from resistive force sensors (FSR), aiming for assistive technology applications, such as sensing gloves (SGS). Its core objective is the creation of resources and tools, both in hardware and software, that are functional, scalable, and economically accessible to the general public, inspiring students and researchers to replicate the content through a didactic structure. The methodology involves building an electronic circuit based on the ESP32 microcontroller, encompassing various building blocks to achieve maximum quality and performance from the device. To overcome the intrinsic non-linearity of the analog-to-digital converter (ADC), a software-based correction using a Lookup Table (LUT) was implemented, measurably improving the accuracy of the measurements. The system transmits the calibrated data simultaneously through two channels: via serial communication for local applications, and via the MQTT protocol for remote monitoring, enabling its use in Internet of Things (IoT) applications or as a mobile device. As a result, a functional and low-cost prototype was obtained, offering a flexible solution not only for FSRs but also for various resistive sensors, demonstrating great potential for generalization in engineering projects.

Keywords: Embedded System. ESP32. Force Sensor. Assistive Technology. Internet of Things.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ADC	Conversor Analógico-Digital
BJT	Transistor de Junção Bipolar
BLE	Bluetooth de Baixa Energia
CI	Circuito Integrado
DAC	Conversor Digital-Analógico
EDA	Automação de Design Eletrônico
EMF	Campos Eletromagnéticos
EMI	Interferência Eletromagnética
ESD	Eletricidade Estática
FES	Estimulação Elétrica Funcional
FSR	Force Sensing Resistors (Sensor Resistivo de Força)
HMI	Interfaces Homem-Máquina
IBGE	Instituto Brasileiro de Geografia e Estatística
IDE	Ambiente de Desenvolvimento Integrado
IoT	Internet das Coisas
LED	Light Emitting Diode (Diodo Emissor de Luz)
LER	Lesão por Esforço Repetitivo
LWT	Last Will and Testament
MCU	Microcontrolador
MQTT	Message Queuing Telemetry Transport
PCB	Printed Circuit Board (Placa de Circuito Impresso)
PET	Polietileno Tereftalato
PWM	Pulse Width Modulation (Modulação por Largura de Pulso)
QoS	Qualidade de Serviço

SBC	Single Board Computers
SGS	Sensoring Glove System (Sistema de Luvas Sensoriais)
SMD	Surface-Mount Technology (Tecnologia de Montagem em Superfície)
SoC	System-on-Chip
TCC	Trabalho de Conclusão de Curso
THT	Through Hole Technology (Tecnologia de Furo Passante)
UART	Universal Asynchronous Receiver/Transmitter (Transmissor/Receptor Assíncrono Universal)
USB	Universal Serial Bus
USP	Universidade de São Paulo
USPSC	Campus USP de São Carlos
WLAN	Rede Local Sem Fio

LISTA DE SÍMBOLOS

Ω	Resistência elétrica
β	Coeficiente Beta de um transistor
ρ	Resistividade elétrica do material

SUMÁRIO

1 INTRODUÇÃO

A Engenharia Biomédica é a representatividade da frente da aplicação de princípios de engenharia para solução de desafios complexos na área da saúde. Este campo interdisciplinar tem se demonstrado fundamental no desenvolvimento de ferramentas que vão além do auxílio no diagnóstico e tratamento de enfermidades, avançando também na reabilitação e na melhoria da qualidade de vida de indivíduos com limitações funcionais, estando esse grupo em crescimento nos últimos anos no Brasil. Dentre as diversas áreas de atuação, a criação de tecnologias assistivas se destaca por seu impacto direto na autonomia e reintegração social de pessoas com deficiência.

Sendo uma das condições mais debilitantes a perda ou diminuição da sensibilidade motora ou tátil nas mãos, uma sequela comum de lesões nervosas, como LER, acidentes vasculares cerebrais ou doenças neurodegenerativas. A capacidade de interação com o ambiente, ou seja, de sentir pressão, textura e temperatura é essencial para a manipulação segura de objetos e para a interação com o ambiente, sendo fundamental para qualidade de vida de um indivíduo. A ausência desse *feedback* sensorial compromete a execução de tarefas cotidianas, desde as mais simples, como segurar um copo ou um lápis, até as mais complexas, como operação de maquinário, gerando dependência e impactando negativamente a autoestima do indivíduo, assim como afetando as ações laborais.

Partindo desse contexto, o desenvolvimento de dispositivos que possam restaurar, substituir ou auxiliar essa sensibilidade perdida é de extrema relevância para o desenvolvimento de novos projetos de engenharia. A evolução de sensores e sistemas eletrônicos integrados abriu novas possibilidades para a criação de soluções que traduzem estímulos físicos em sinais perceptíveis pelo usuário, funcionando como uma extensão do sistema nervoso, com o avanço da cultura dos *wearables* pode-se também observar a recepção da população a integração da tecnologia como uma forma de assistente diário. Este trabalho se insere precisamente nesta área, propondo o desenvolvimento de um sistema de luva sensorial que visa a reabilitação e o auxílio a pacientes com perda de sensibilidade tátil. O projeto foi concebido não apenas como uma ferramenta funcional, mas também como um recurso educacional, detalhando cada etapa de sua construção para fomentar o interesse e o desenvolvimento de novos talentos na engenharia aplicada à saúde, mantendo-se constante a conceitos simples e escaláveis a outros métodos de sensoriamento resistivo.

1.1 Objetivo

Esse projeto tem como objetivo principal o desenvolvimento de um circuito protótipo de baixo custo de aplicável a sensores resistivos, com FSR, com foco em uso em luvas de sensoriamento, com todo o documento sendo visado para ser uma ferramenta didática

e funcional. O desenvolvimento do dispositivo foca na simplicidade tanto quanto em sua aplicação prática, focando também em sua construção, permitindo que estudantes e iniciantes na área de eletrônica possam replicar e compreender seu funcionamento. Dessa forma, todo o processo será detalhado de forma a guiar o leitor a compreender os conceitos necessários assim como a motivação por trás das escolhas empregadas, desde a seleção dos componentes até definição de código lógico associado.

Focando na viabilidade econômica da solução, o projeto prioriza o uso de componentes de baixo custo e facilmente acessíveis no mercado, buscando concentrar sempre que possível a maioria das funções em um único componente, buscando a simplificação. A arquitetura do sistema foi pensada para ser generalista, ou seja, aplicável a diferentes cenários e necessidades, embora seu foco técnico esteja na utilização de FSRs, o desenvolvimento busca ser aberto para operação baseada em sensores eletrônicos transdutores que operem com variação da resistência.

O projeto também possui um propósito adicional, servir como um recurso educacional que inspire e capacite novos interessados na área de engenharia eletrônica através de projetos com visualização de resultados em tempo real, demonstrando como a tecnologia pode gerar impacto social positivo, assim como, apresentar uma solução que seja útil à população, oferecendo um caminho para a criação de tecnologias assistivas mais acessíveis e que possam efetivamente melhorar a vida de pessoas com comprometimento sensorial.

2 REVISÃO BIBLIOGRÁFICA

A aplicação de conceitos eletrônicos e mecânicos para auxilio de indivíduos é uma das áreas de aplicação da Engenharia Biomédica, porém ainda assim não engloba por completo todos os seus ramos, abrangendo cenários de conceituação assim como aplicações práticas de soluções teóricas, dessa forma, é extremamente versátil em sua definição.

Há porém de se apurar mais profundamente as suas ramificações, visto que há sobreposição de definições e conceitos que seriam mais organizados se abordados individualmente, dessa forma, o conteúdo a seguir é a exposição parcial de seus conceitos.

2.1 Engenharia Biomédica

A Engenharia Biomédica é a aplicação dos princípios e conceitos de engenharia para compreensão e controle de sistemas biológicos, sendo relacionado a diversas áreas, como aplicação de sistemas biomecânicos, com aplicação em fisioterapias e ergonomia, até biosensoriamento, visando a aplicação de sensores para monitoramento e acionamento variável a atividades de um indivíduo (Bronzino, 2005). Dessa forma, fornecendo auxílio as dificuldades enfrentadas em quaisquer fases da vida, com abordagens médicas preventivas e serviços de auxílios gerais.

2.2 Engenharia de Reabilitação

Aprofundando no conteúdo de Engenharia Biomédica, aborda-se a área relacionada a Engenharia de Reabilitação, tendo sido descrito por James Reswick como (Reswick, 1980) a aplicação de conhecimentos científicos e tecnológicos para assistir indivíduos com deficiências. Partindo dessa definição, vemos então que este campo tem como produto as tecnologias com intuito de auxílio físico e/ou mental, sendo desenvolvidas com o propósito de aumentar ou melhorar as capacidades funcionais de indivíduos com alguma condição incapacitante vigente, seja ela permanente ou não. Dentro dessas tecnologias há uma diversidade de produtos, como as órteses, que aumentam a função de uma extremidade, e as próteses, que substituem uma parte do corpo.

2.2.1 Visão do Cenário Brasileiro

Abordando o cenário da engenharia de reabilitação no Brasil, verifica-se que tem apresentado uma visível expansão, motivada principalmente pela crescente demanda por tecnologias assistivas e pelo crescimento da necessidade de maior inclusão social e profissional de pessoas com deficiência, como o programa Viver Sem Limite (MDHC, 2023) e o Estatuto da Pessoa com Deficiência. As Instituições de ensino superior e centros

de pesquisa estão cada vez mais focados na formação de especialistas para esta área, fornecendo cursos de formação ou certificados especiais como forma de motivação ao ingresso na área, possuindo a EESC uma área focada em engenharia biomédica.

Há também de se considerar o rápido envelhecimento da população brasileira como um dos principais motores dessa expansão, visível na pirâmide etária. De acordo com o Instituto Brasileiro de Geografia e Estatística (IBGE), a proporção de idosos (60 anos ou mais) saltou de 11,3% em 2010 para 15,8% em 2022. Esse fenômeno demográfico influencia o índice de doenças crônicas e condições que afetam a mobilidade e a funcionalidade na população geral, como artrite e derrames, ampliando a necessidade de soluções de reabilitação.

Paralelamente, nas outras faixas etárias, notam-se elevadas taxas de acidentes de trânsito e de trabalho no país gerando um número significativo de lesões e deficiências, como lesões por estresse repetitivo. O Ministério da Saúde aponta os acidentes de trânsito como uma das causas primárias de amputações e lesões medulares, que demandam o uso de próteses e órteses.

Além disso, o aumento da conscientização sobre os direitos das pessoas com deficiência, fortalecido por movimentos sociais e pelo avanço da legislação brasileira, tem se demonstrado como fundamental. A Lei Brasileira de Inclusão da Pessoa com Deficiência (Lei 13.146/2015) representa um marco, promovendo a acessibilidade e incentivando a criação de novas tecnologias assistivas.

Dados do IBGE do censo de 2022 revelam que cerca de 14,4 milhões de brasileiros (7,3% da população) vivem atualmente com alguma deficiência, essa proporção sobe fortemente quando se limita as pessoas com 70 anos ou mais, onde 27,5% da população sofre com algum tipo de deficiência. As limitações que afetam as mãos são particularmente extenuantes, pois impactam diretamente a autonomia em atividades essenciais como alimentação, higiene, laborais e lúdicas. Em resposta, o Brasil tem avançado no desenvolvimento de tecnologias de diversas áreas da engenharia biomédica, como próteses mioelétricas, órteses, dispositivos de estimulação elétrica funcional (FES) e interfaces homem-máquina (HMI), que permitem o controle de aparelhos por meio de métodos menos fisicamente exaustivos.

Apesar desses avanços, o setor enfrenta obstáculos consideráveis. A falta de financiamento consistente para pesquisa e desenvolvimento, muitas vezes dependente de verbas governamentais limitadas e variáveis de acordo com a política pública vigente, podem atrasar a inovação e a produção em larga escala de soluções *ready-to-market*. Outro grande desafio é a acessibilidade, principalmente econômica, já que a distribuição de dispositivos e serviços de reabilitação é desigual, concentrando-se nos centros urbanos afluentes e dificultando o acesso para populações de baixa renda ou de áreas rurais.

Soma-se a isso a carência de profissionais qualificados no cenário laboral brasileiro. A formação de engenheiros de reabilitação com currículos especializados ainda é incipiente e não acompanha o ritmo do avanço tecnológico e da demanda crescente. A adaptação do usuário final às novas tecnologias também representa uma barreira, pois muitos dispositivos exigem uma curva de aprendizado e suporte contínuo para evitar o abandono, sendo essa uma dificuldade particularmente complexa quando visto nos grupos mais idosos da sociedade.

Ainda assim, a perspectiva futura demonstra potencial, o crescente interesse acadêmico e profissional é impulsionado por avanços em áreas como inteligência artificial, robótica e impressão 3D, que viabilizam soluções personalizadas e mais eficientes, além de democratizar o desenvolvimento com seu custo reduzido. O fortalecimento de políticas públicas e o crescimento do mercado de tecnologias assistivas, estão contribuindo para a criação de produtos mais eficazes e acessíveis. Em conjunto, com o novo desenvolvimento da cultura de *open-source* novas ferramentas estão sendo constantemente desenvolvidas, com uma barreira de entrada reduzida e simplificada, permitindo que a disseminação da aplicação seja mais abrangente e generalizada.

2.3 Tecnologia assistiva

O desenvolvimento de toda solução tecnológica exige a aplicação sequencial de de conceitos para construção de um cenário lógico que leve a resolução da problemática (Lundborg; Rosén, 2007), e esse desenvolvimento se demonstra ainda mais fundamental quando se trata de aplicação para um ser vivo. Visando a criação de um dispositivo assistivo há a necessidade de avaliação de todo o arcabouço prático para compreensão de sua segurança e aplicação, desde analisar o indivíduo atuante até o cenário que será abordado. Dessa forma, o processo de criação dessas tecnologias tende a seguir etapas específicas para assegurar um resultado eficaz.

1. **Análise:** A fase inicial consiste em uma investigação aprofundada para compreender por completo a necessidade a ser atendida, envolvendo determinar as características da tarefa, as restrições do ambiente e as particularidades do usuário da solução, geralmente se enquadrando em físicas, cognitivas e/ou psicossociais. Possuindo como resultado desta etapa uma lista detalhada de especificações de desempenho que a solução final deverá possuir.

2. **Síntese:** Esta é a etapa criativa do processo, onde, com base em princípios de engenharia, são concebidas diversas soluções potenciais, onde cada conceito é processado através de esboços e justificativas embasadas em análises técnicas que detalham seu funcionamento.

3. **Avaliação:** Essa é uma etapa decisória do processo, nela as propostas mais promissoras da fase de síntese são submetidas a uma avaliação individual e comparativa, sendo comum nesse momento incluir o desenvolvimento de protótipos ou simulações computacionais. Sendo importante a participação de distintas partes do problema, tanto do usuário quanto do criador além outras partes investidas na resolução, como instituições de ensino.

4. **Decisão:** Com base nos resultados da avaliação, uma solução deve ser selecionada. Nessa etapa é fundamental a consideração das aplicações práticas da solução e das consequências, como investimento necessário para construção assim como preferências do perfil de usuário considerado durante as etapas anteriores.

5. **Implementação:** Uma vez decidida a solução, inicia-se a fabricação e montagem do protótipo funcional. Em seguida, são realizados testes de validação para verificar o conceito na prática, permitindo que ajustes finos sejam feitos para chegar ao projeto definitivo.

É importante notar que a aplicação de uma tecnologia assistiva nem sempre precisa ser diretamente aplicada no indivíduo. A solução pode ser desenvolvida para uso em dispositivos vestíveis (*wearables*), como uma luva sensorial (Mendes, 2010), ou em equipamentos de auxílio externo, como uma muleta instrumentada, que interage com o usuário para melhorar sua funcionalidade.

2.4 Aplicação

A mão humana é fundamental para interação com o ambiente, e é exatamente no estudo da biomecânica que os movimentos são estudados, aplicando conceitos de mecânica associada a cenários biológicos. Um exemplo de tecnologia assistiva aplicada à mão é o *Sensoring Glove System* (SGS - Sistema de Luvas Sensoriais), desenvolvido inicialmente em um estudo (Lundborg; Rosén; Lindberg, 1999) focando no auxílio de indivíduos com a identificação de objetos independente de possuírem uma visão debilitada, foi-se expandindo até a perspectiva que existe nos dias atuais, com diversos projetos utilizando os conceitos aprendidos no século passado para prover assistência via as mãos dos pacientes.

Baseado no princípio da substituição sensorial, o sistema foi desenvolvido para auxiliar pessoas que perderam a sensibilidade tátil, com seu objetivo do a utilização de métodos de identificação de superfícies mesmo possuindo uma redução tátil nas mãos, partindo do uso de microfones localizados nas extremidades dos dedos pode-se então gerar sinais de áudio para relacionar a audição com a sensação tátil que o objeto pudesse fornecer.

Pequenos Microfones foram posicionados nos dedos para captar e amplificar o som em decorrência da fricção produzida pelo toque em diferentes superfícies, nesse caso sendo o microfone e o objeto em análise. Verificou-se que a movimentação sobre texturas distintas gera estímulos acústicos distintos, de forma que o usuário possa realizar a identificação sem o tato ou visão, realizando a compensação por meio da audição. Dessa forma, a pesquisa demonstrou a aplicação prática que as SGS possuem assim como a resiliência e adaptabilidade humana.

3 MATERIAIS UTILIZADOS

3.1 Hardware

3.1.1 Listagem completa

Considerando a manufatura do projeto podem-se dividir os componentes utilizados em algumas categorias: Resistor, Resistor variável (*Trimmer*), Diodo, Diodo Zenner, LED (*Light Emitting Diode*), Capacitor, Fusível & Socket correspondente, Conector fêmea tipo Jack, USB A fêmea, Transistor, Chave de 3 estados, Botão/Chave, Regulador de Tensão, *Terminal Block* de 2 e 4 pinos, barramento de pinos, um motor de vibração, FSR's e por fim um microcontrolador ESP32.

Tabela 1 – Listagem Completa dos Componentes

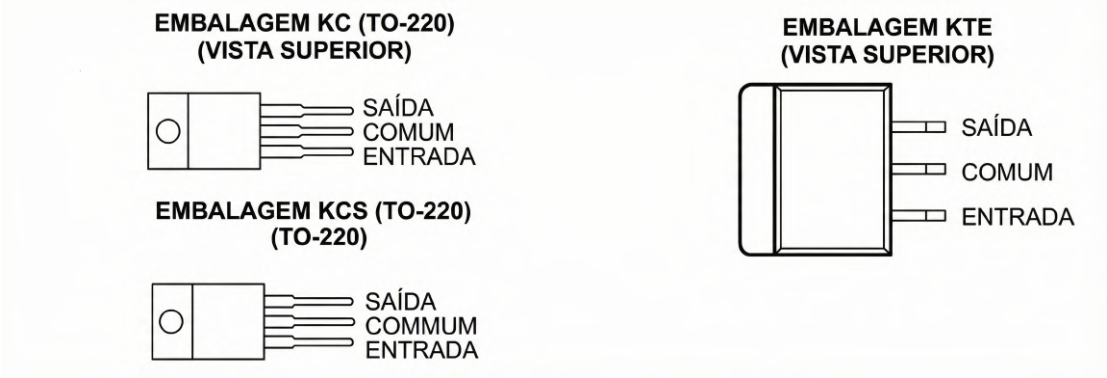
Componente	Descrição	Parte	Código	Quantidade Por PCB
1	Unpolarized capacitor	C	100n	1
2	Unpolarized capacitor	C	330n	2
3	50V 1A General Purpose Rectifier Diode; DO-41	1N4001	1N4001	2
4	1300mW Silicon planar power Zener diodes; DO-41	1N4735A	1N4735A	1
5	Light emitting diode	LED	LED_Low_Batt	1
6	Light emitting diode	LED	LED_On	1
7	Fuse	Fuse	Fuse	1
8	DC Barrel Jack	Jack-DC	Jack-DC	1
9	Terminal Block 2 pin	Screw_Terminal_01x02	Micro_USB_ESP32_Power	8
10	USB Type A connector	USB_A	USB_A	1
11	0.8A Ic; 45V Vce; NPN Transistor; TO-92	BC337	BC337	2
12	Resistor	R	47	1
13	Resistor	R	330	2
14	Resistor	R	470	1
15	Resistor	R	1k	1
16	Resistor	R	2k	1
17	Resistor	R	10k	1
18	Resistor	R	33k	1
19	Resistor	R	70k	5
20	Trim-potentiometer	R_Potentiometer_Trim	50k	5
21	3 Position Switch	SW_DP3T	SW_DP3T	1
22	Push button switch; generic; two pins	SW_Push	SW_Push	1
23	Microcontroller	ESP32-DEVKIT-V1	ESP32-DEVKIT-V1	1
24	Linear Regulator; 5V; TO-220	LM7805_TO220	LM7805_TO220	1
25	Terminal Block 4 pin	Screw_Terminal_01x04	Switch Terminal Block	1
26	Barramento de Pinos p/ ESP32	Barramento 1x15	Barramento 1x15	1
27	Motor de vibração	Motor De Vibração	Cod. 1027; 3V; 10 x 3.2mm	1
28	Force Sensing Resistors (FSR)	FSR Máx XXk	Cód. xxxx	5

Fonte: Autoria própria

3.2 Encapsulamentos

É comum na eletrônica um mesmo componente possuir distintas formas físicas, ou seja, distintos encapsulamentos, enquanto que o seu funcionamento se mantêm quase indistinguível, portanto é importante se atentar a definição do encapsulamento de um determinado material. Um exemplo pode ser visto no componente LM7805 onde no próprio documento de dados do componente (*Datasheet*) é possível observar que na primeira página já se fornece os encapsulamentos fornecidos pela empresa.

Figura 1 – Visão dos encapsuamentos do LM7805



Fonte: Adaptado pelo autor |
https://www.sparkfun.com/datasheets/Components/LM7805.pdf

Além de variar o encapsulamento para os componentes integrados (C.I.), como no LM7805, é essencial distinguir se o componente a ser utilizado será do tipo de montagem *Through Hole Technology* (THT) ou *Surface-mount technology* (SMD). A definição dessas características está fortemente relacionada com a manufatura do circuito. Componentes SMD são menores, ocupando menos espaço na placa e portanto reduzindo o tamanho, e mais baratos, reduzindo assim o custo por placa, porém esse mesmo estilo de montagem fornece também adversidades. Devido ao seu tamanho reduzido é necessária experiência, equipamento especializado ou diversas tentativas e erros. Há também problemática da obtenção do componente, sendo menos comum encontrar uma grande variedade de valores em lojas pequenas. Em suma, deve-se atentar as escolhas durante o processo de produção.

Figura 2 – Comparativo entre componentes SMD e THT

Fonte: Autoria própria

Mesmo para o mesmo tipo de encapsulamentos ainda é comum variar suas dimensões até mesmo para outros componentes mais simples, por exemplo, os encapsulamentos para resistores tipo THT é comum variar com base na potência dissipada pelo componente. Resistores de 1/4W possuem usualmente a mesma dimensão, porém resistores com outras

potências dissipadas irão provavelmente ser distintos, sendo comumente maiores quanto maior a dissipação.

3.3 Resistores

O resistor é o componente mais simples observado na eletrônica, tendo o seu valor denominado na unidade de medida (Ω) e é bastante versátil. Possuindo um relacionamento linear entre a tensão e corrente ele permite diversas funcionalidades, tal qual como divisor de tensão, limitador de corrente elétrica em um circuito, ajuste de ganho de sinal, dissipador de calor e outros.

Figura 3 – Visualização de um resistor típico de 1/4W

Fonte: Autoria própria

A equação que explicita a relação linear entre resistência e tensão ou corrente pode ser vista abaixo.

$$R = \frac{V}{I}$$

R - Representa a resistência em Ohms (Ω).

V - Representa a tensão em Volts (V).

I - Representa a corrente em Amperes (I).

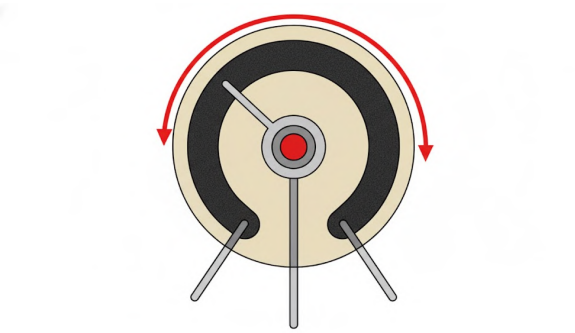
3.4 Resistor variável (Potenciômetro)

Tendo já elucidado a respeito do resistor comum pode-se então abordar um conceito tangencial, porém importante, sendo que resistores com resistência variável existem e são fortemente utilizados. Um potenciômetro (também conhecido como *Trimmer* ou *Trimpot*) é um elemento resistivo ajustável, usualmente manualmente, que permite que, ao contrário dos resistores fixos, varie-se a resistência (Ω) do componente.

Devido à propriedade da variação da resistência os potenciômetros permitem que o seu valor seja modificado conforme necessário durante o processo de montagem, dessa forma permitindo a calibração ou ajuste de uma parte do circuito. Assim como o resistor fixo, o potenciômetro também possui um relacionamento linear da resistência entre a

tensão e corrente e, portanto, também pode ser empregado nas mesmas funcionalidades, para controle da tensão e da corrente.

Figura 4 – Construção interna de um resistor variável



Fonte: Autoria própria

Os potenciômetros geralmente possuem três terminais onde medindo nos extremos obtém-se o maior valor possível de resistência e medindo entre o terminal central e qualquer um dos terminais laterais pode-se então obter uma resistência variável. Assim como os resistores fixos os potenciômetros possuem diversos tipos de encapsulamentos.

A resistência de um material pode ser equacionado de forma a considerar as propriedades do material, como visto na equação abaixo:

$$R = \rho \cdot \frac{l}{A} \tag{3.1}$$

Onde, ρ representa o coeficiente de resistência do material em Ω/m, l representa o comprimento do material em m e A a área da secção transversal do material em m^2. Dada a equação e a construção interna do potenciômetro então pode-se notar que a variação da resistência ocorre pela variação do comprimento l do material.

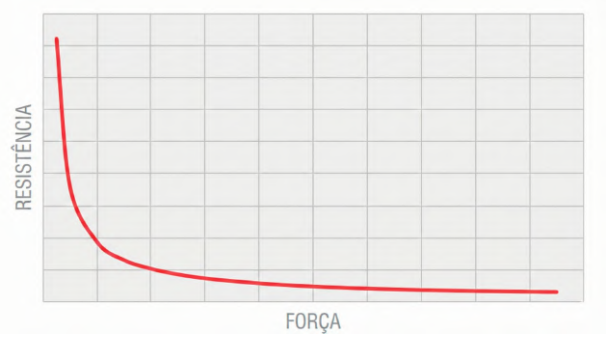
3.5 Force Sensing Resistors (FSR)

Construindo com base no que foi detalhado previamente, temos então uma aplicação prática dos conceitos previamente dispostos. Com base na equação de resistência (3.1), podemos notar a relação entre a resistência de um material e o seu comprimento ou área, expandindo nesse conceito, podemos então controlar a resistência de um material controlando o seu comprimento ou área, visto que o seu coeficiente de resistência do material (ρ) se manteria constante dadas condições normais de temperatura e pressão (CNTP) na maioria dos cenários.

Desenvolvendo nessa análise temos então a aplicação de sensores com resistência variável com base no efeito piezoresistivo intrínseco da sua construção, como os *Force*

Sensing Resistors (FSR). Esse sensor passívo possui uma resistência variável com base na força perpendicular à sua superfície, de forma que possuem sua resistência máxima quando em repouso. Este comportamento piezoresistivo o torna um transdutor eficaz para converter uma grandeza mecânica (força) em um sinal elétrico, daí o seu nome de resistor sensível a força (FSR).

Figura 5 – Curva típica de um FSR



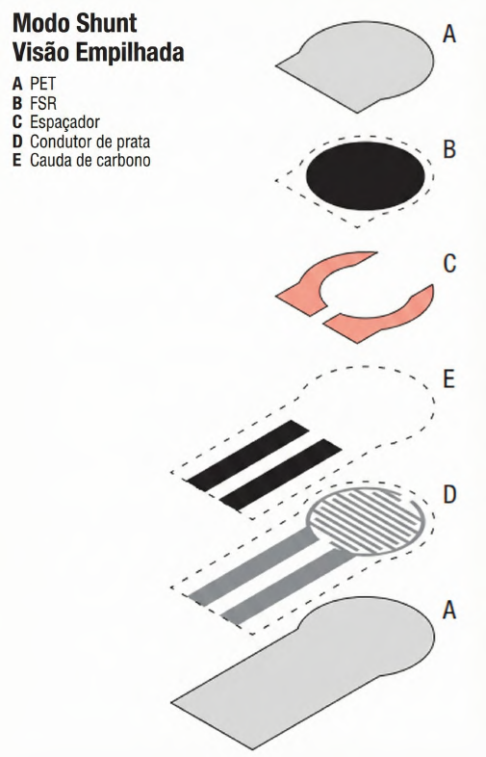
Fonte: Adaptado pelo autor | https://www.mouser.com/pdfdocs/Ohmite-FSR-Integration-Guide-V1-0_11-01-18.pdf

3.5.1 Construção e Princípio de Funcionamento

É fundamental elucidar que o comportamento de um FSR é uma consequência direta de sua construção. O dispositivo é fabricado a partir de um polímero robusto, usualmente denominado de "Membrana"e construído com material plástico Polietileno Tereftalato, comumente conhecido como PET, que garante durabilidade e flexibilidade similar a garrafas comuns de mercado. Existem diversos tipos de sensores FSR, porém a sua estrutura tende a seguir o mesmo padrão:

- **Camada Semicondutora:** Uma camada de substrato flexível (PET) revestida com um material semicondutor.
- **Espaçador Adesivo:** Uma camada intermediária, com expessura típica de 50-125 microns, com um recorte que define a área ativa e cria um espaço vazio entre camadas.
- **Camada Condutiva:** Uma segunda camada de substrato (PET) com traços condutivos impressos, tipicamente em um padrão de eletrodos intercalados.
- **Adesivo Traseiro:** Permite a fixação e proteção do do sensor.

Figura 6 – Construção de um FSR do tipo *shunt*



Fonte: Adaptado pelo autor | https://www.mouser.com/pdfdocs/Ohmite-FSR-Integration-Guide-V1-0_11-01-18.pdf

O seu método de construção está fortemente conectado a sua aplicação desejada, sendo que tipicamente os FSRs podem ser catalogados em quatro tipos:

1. **Ponto Único (*Single-point*)**
   Com formas e tamanhos variados, desde pequenos círculos até longas tiras, um sensor de ponto único reportará a força aplicada apenas no eixo Z. Sensores de ponto único podem ser individuais ou múltiplos sensores podem ser fabricados em um único conjunto para criar matrizes (*arrays*) e outros arranjos de sensores personalizados.

2. **Potenciômetro Linear (*Linear Potentiometer*)**
   Tipicamente em configuração de tira ou roda de rolagem (*scroll-wheel*), um potenciômetro linear reportará simultaneamente a posição (X ou Y) e a força aplicada (Z) para um único ponto de toque.

3. **Toque Único 3D (*3D Single-touch*)**
   Um trackpad que pode reportar a posição em duas dimensões, ou seja, X e Y, além da força aplicada Z, simultaneamente para um único ponto de toque.

4. **Multitoque 3D (*3D Multi-touch*)**
   Um trackpad que reporta simultaneamente X, Y e Z para múltiplos pontos de toque.

Pode-se catalogar os FSRs também com base na sua construção, sendo eles o *Shunt Mode* e *Thru Mode*, não divergindo muito do já elucidado, porém com a principal diferença que o *Thru Mode* tem a sua camada FSR impressa sobre uma área condutiva único, ocorrendo tanto na camada superior quanto na inferior, dessa forma, o condutor em cada camada possui somente um único terminal, dessa forma a corrente transfere de uma camada para a outra.

Devido a essa mínima diferença de construção, os sensores FSR do tipo *Shunt Mode* possuem uma maior faixa de força mensurável, tipicamente chegando até 5kg. Curva de variação da resistência com relação a força mais suave, proporcionando melhor controle, especialmente em forças mais elevadas. Sendo porém a sua característica mais atrativa para esse projeto o fato de que necessita de menos camadas de impressão e menos tinta de prata são necessárias, portanto o custo é normalmente muito menor que o dos sensores do tipo *Thru Mode*.

Para o FSR em foco, ou seja, o *Shunt Mode*, o princípio de funcionamento baseia-se no efeito de curto-circuito resistivo (*shunting effect*). Em repouso, o espaçador mantém as camadas separadas, resultando em um circuito aberto com uma resistência de repouso (*stand-off resistance*) extremamente elevada, usualmente superior a 1 MΩ. Ao aplicar uma força, a camada semicondutora é pressionada contra os eletrodos. O contato entre as camadas cria múltiplos caminhos paralelos para a corrente. Com o aumento da força, a área de contato se expande, "curto-circuitando"mais eletrodos e, consequentemente, diminuindo a resistência total do dispositivo.

Além da sua construção e aplicação, também pode-se segmentar os FSRs de acordo com a sua geometria, sendo tipicamente dividida em quatro tipos:

- **FSRs Circulares**: Disponíveis em diversos diâmetros, são projetados para detectar força em um ponto localizado. São ideais para a criação de botões sensíveis à pressão em interfaces homem-máquina ou como pontos de contato em garras robóticas.

- **FSRs Quadrados e Retangulares**: Possuem uma área de detecção mais ampla, sendo adequados para detectar a presença ou a distribuição de pressão de um objeto maior. Suas aplicações incluem o monitoramento de pacientes em leitos hospitalares e a detecção de ocupantes em assentos automotivos.

- **FSRs em Tira (*Strip*)**: Sensores longos e finos, projetados para detectar a pressão ao longo de uma linha. Permitem não apenas detectar a presença, mas também a posição de um toque ao longo de seu comprimento, sendo úteis em interfaces de controle e análise de distribuição de peso.

3.5.2 Considerações adicionais

Embora os FSRs estejam relacionados a um dos componentes mais simples da eletrônica (resistores), eles porém possuem conceitos relacionados a sua aplicação prática fundamentais, tanto para escolha quanto para a operacionalidade. A fim de aplicar corretamente um FSR, o designer do circuito deve considerar suas características elétricas e mecânicas intrínsecas, tendo algumas das mais fundamentais estando listadas abaixo:

Curva da relação entre Força aplicada e resistência mensurada: A resposta do sensor é altamente não linear, seguindo uma curva tipicamente inversa da logarítmica, ou seja, a resistência diminui drasticamente com forças iniciais e de forma mais suave com forças elevadas. Dessa forma tornando a sua faixa de aplicação, que embora seja ampla (até 5Kg), limitada ao *range* esperado da aplicação. Uma boa prática é identificar qual a faixa de medição esperada para o projeto, e com base nesse espaço de valores, encontrar com base em *datasheets* aqueles sensores FSR que possuam maior comportamento linear nessa faixa. Portanto reduzindo a variação brusca de valores e consequentemente podendo também reduzir faixas de erros e medições em outras áreas do projeto, como com relação ao conversor de tensão utilizado ou conversor Analógico para Digital.

Histerese: A curva de resposta para o carregamento (aumento da força) é diferente da curva para o descarregamento (diminuição da força). Isso significa que, em um mesmo ciclo, o valor de resistência pode variar dependendo da direção da mudança de força, ou seja, a medição ao aplicar a força e ao remover tal força não seria exatamente igual, geralmente variando com base na faixa de valor definido no *datasheet* do componente. Há também a variação da medição do componente com diferenças medições porém *inputs* equivalentes, ou seja, para uma mesma força a medição pode varia levemente entre momentos, isso pois há a deformação do material com o tempo, variações de temperatura, interferência, e outros fatores que podem estar relacionados.

Deriva (*Drift*): Sob uma carga estática constante, a resistência do FSR tende a diminuir lentamente ao longo do tempo. Este efeito é conhecido como deriva e deve ser considerado em aplicações de medição de longa duração. Esse é um efeito comum para todos os tipos de sensores com construção baseada em polímeros, uma boa prática é a limitação da magnitude da carga, ou seja, da faixa de valores de medição e tempo de duração da aplicação da carga.

Tempo de Resposta: O FSR possui uma resposta mecânica muito rápida, tipicamente entre 1 a 2 milissegundos, permitindo sua utilização em aplicações dinâmicas. Porém é importante considerar que a sua construção é baseada em um polímero, tipicamente PET, portanto é importante evitar deformações inelásticas no material, a fim de conservar a sua aplicação a longo prazo, assim como considerar um arcabouço eletrônico necessário para acompanhar a sua velocidade de resposta a fim de evitar gargalos no processo.

Vida Útil: Devido a sua construção simples e robusta, principalmente devido a sua utilização de polímero PET, sendo amplamente famoso e difundido na indústria, a sua operação contínua permite uma longa vida útil, geralmente superior a 1 milhão de atuações sob condições normais de operação desde que respeitando as informações de uso previamente dispostas.

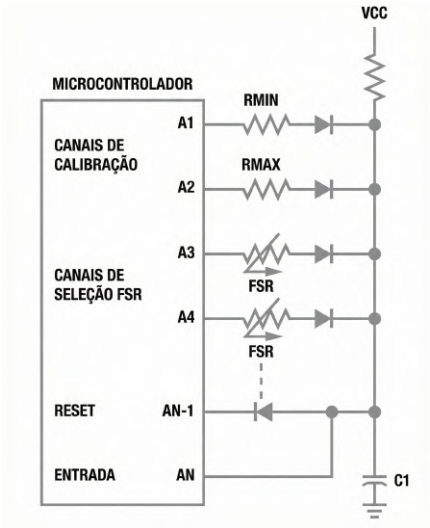
Força de Atuação (*Actuation Force*): Refere-se à força mínima necessária para que o sensor comece a exibir uma mudança significativa em sua resistência. Este valor depende do sensor específico, porém geralmente encontra-se no *datasheet* fornecido pelo fabricante.

3.5.3 Circuitos de aplicação

Os FSRs, em resumo, são sensores com variação de resistência, portanto, os métodos típicos de aplicação para mensuração de sensores dessa modalidade também podem ser aplicados, porém é importante compreender que cada sensor resistivo, embora façam parte da mesma classe, possui a sua forma ideal de aplicação, alguns podem requerer compensação, como os termistores (sensores de temperatura) que tipicamente utilizam uma construção sistêmica para reduzir erros, porém podem ser utilizados de maneira geral, embora com redução de resolução, de forma análoga aos FSRs.

É prático o uso de microcontroladores (MCU) integrados a conversores de tensão de analógico para Digital visando a leitura, tratamento e envio imediato dos dados em conjunto com programação baseado em lógicas pré-definidas.

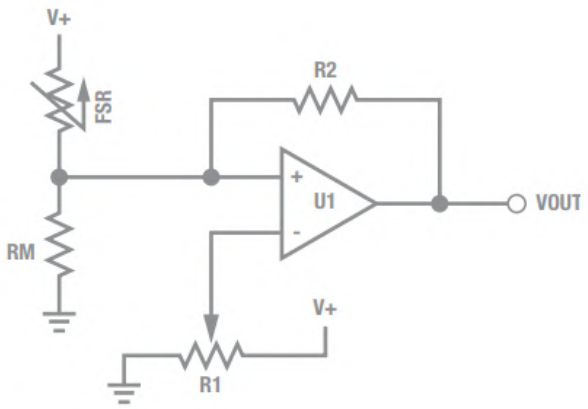
Figura 7 – Exemplo de aplicação da leitura de um FSR utilizando um microcontrolador



Fonte: Adaptado pelo autor | https://www.mouser.com/pdfdocs/Ohmite-FSR-Integration-Guide-V1-0_11-01-18.pdf

Um método clássico da leitura dos sensores resistivos é com a aplicação de amplificadores operacionais (Op-Amp).

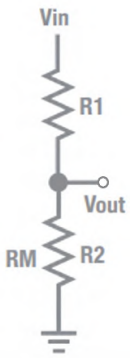
Figura 8 – Aplicação típica para leitura de Sensor FSR com Op-Amp



Fonte: Adaptado pelo autor | https://www.mouser.com/pdfdocs/Ohmite-FSR-Integration-Guide-V1-0_11-01-18.pdf

Independente do método de leitura, o arcabouço eletrônico associado para a leitura do sensor resistivo se mantém, em seu cerne, igual. A aplicação de sensores resistivos em divisores de tensão é o método mais comum de aplicações em diferentes cenários.

Figura 9 – Exemplo de um divisor de tensão



Fonte: Autoria própria

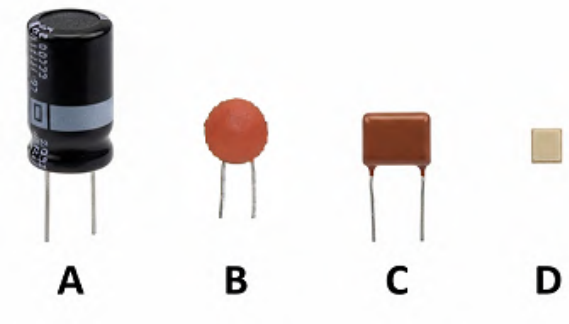
3.6 Capacitor

Um capacitor é um componente eletrônico passivo que consiste em dois condutores na forma de uma placa separados por um isolante, denominado dielétrico, com capacidade de armazenar energia na forma de um campo elétrico e então a liberar em um circuito.

Ao aplicar tensão entre os terminais do capacitor há então o acumulo de cargas elétricas nas placas condutoras, criando um campo elétrico entre elas. A unidade de medida associada com os capacitores é Farads (F), estando diretamente relacionado com o acúmulo de cargas no componente e, portanto, com a capacidade de carregar e descarregar

do mesmo, porém, é fundamental se atentar a polarização, visto que alguns tipos de capacitores são dependentes de polaridade apropriada.

Figura 10 – Exemplos de capacitores: (A) Eletrolítico; (B) Cerâmica; (C) Poliêster; (D) SMD]



Fonte: Autoria própria

Além de se atentar a polarização do capacitor, é importante observar a tensão máxima permitida assim como para a capacitância (F). Capacitores podem ter diversas utilizações como acoplamento de sinais, filtragem de ruído, armazenamento de energia, temporização e redução da variação da tensão, comumente denominado re *ripple*, na saída de reguladores de tensão.

3.7 Fusível

Tendo já abordado os componentes que possuíam uma alteração significativa no circuito, ou seja, alterando de alguma forma a corrente ou a tensão, como os resistores e capacitores, pode-se então abordar um elemento cuja finalidade principal é a proteção passiva do circuito, cuja finalidade é estar invisível nas operações normais do circuito, salvo em casos de necessidade, sendo ele o fusível. Este é um dispositivo de segurança projetado para interromper o fluxo de corrente elétrica quando esta excede um valor predeterminado, sacrificando-se, ou seja "queimando-se", para proteger componentes mais complexos e de maior importância. Dessa forma o fusível tem a função de 'elo mais fraco' do circuito.

O fundamental para de definição do fusível o dimensionamento de corrente do circuito, portanto ele é tipicamente escolhido por último, porém em cenários onde a corrente não é calculada, pode-se medir a corrente durante a operação do circuito e então realizar a decisão de aplicação do fusível. Com relação aos seu posicionamento do circuito, tipicamente é posicionado em série com a alimentação de entrada.

O fusível consiste em um filamento ou lâmina metálica, fabricado com um material de baixo ponto de fusão, encapsulado em um invólucro, tipicamente de vidro, e nas suas extremidades estão as conexões metálicas. Quando a corrente que atravessa o filamento ultrapassa sua corrente nominal, a temperatura é suficiente para fundir o metal, dessa

forma abrindo fisicamente o circuito e cessando o fluxo de elétrons, portanto evitando dano aos demais componentes. É importante notar que a sua ação, embora imediata, não é instantânea, ou seja, é possível que haja dano a demais componentes antes da ativação do fusível.

Figura 11 – Imagem de um fusível típico

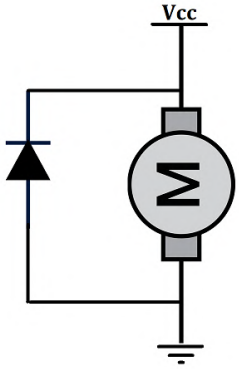
Fonte: Autoria própria

Devido à sua função crítica, a escolha de um fusível deve considerar não apenas a corrente nominal, mas também a tensão de operação do circuito e sua característica de atuação, que pode ser de ação rápida (*fast-acting*), como os tipicamente utilizados em aplicações de eletrônica, ou com retardo (*slow-blow*), para acomodar picos de corrente temporários, como os que ocorrem na partida de motores.

3.8 Diodo

Distintamente dos resistores, que apresentam uma relação linear entre tensão e corrente, o diodo é um componente semicondutor que exibe um comportamento não linear, possuindo como característica fundamental permitir a passagem de corrente elétrica em um único sentido, assim como possuir uma queda de tensão fixa, geralmente sendo utilizado para evitar uma inversão do fluxo de corrente.

Figura 12 – Aplicação de um diodo como flyback de um motor



Fonte: Autoria própria

Sua construção é baseada em uma junção PN, a união de um material semicondutor tipo P (com excesso de "lacunas"ou portadores de carga positiva) e um tipo N (com excesso de elétrons). Este arranjo cria uma barreira de potencial que dita o comportamento do diodo, tipicamente possui dois métodos de polarização a depender da sua aplicação.

Na polarização direta, há a aplicação de tensão ao terminal anodo (lado P) e aterramento ao catodo (lado N), dessa forma a barreira de potencial é superada, permitindo a passagem de corrente com uma queda de tensão pequena e relativamente constante (*threshold voltage*). Na polarização reversa, ao inverter a polaridade da tensão, a barreira de potencial se alarga, bloqueando quase que completamente o fluxo de corrente.

Devido a essa propriedade unidirecional, os diodos são essenciais em aplicações como a retificação de sinais, ou seja, na conversão de corrente alternada para contínua, proteção de circuitos contra inversão de polaridade e em circuitos de chaveamento.

3.9 Diodo Zenner

Elaborando com base no conceito tangencial ao do diodo comum apresentado no capítulo anterior, o diodo Zener é um tipo especial de diodo projetado não para retificar a corrente, mas para atuar como um regulador de tensão de precisão. Enquanto um diodo convencional é danificado se a tensão reversa exceder seu limite, o diodo Zener é fabricado para operar de forma confiável exatamente nesta condição.

Seu princípio de funcionamento baseia-se no "Efeito Zener", ou seja, quando polarizado reversamente, ele bloqueia a corrente até que a tensão atinja um valor definido em seu *datasheet*, denominado Tensão Zener (V_Z). Nesse ponto, o diodo entra em seu modo de operação típico (*breakdown*) e passa a conduzir corrente, mantendo a tensão entre seus terminais praticamente constante e igual a tensão pré-determinada (V_Z), mesmo que a corrente reversa varie.

Essa capacidade de manter uma tensão estável o torna um componente prático para uso em circuitos de regulação e estabilização de tensão, atuando como uma referência de tensão ou protegendo outros componentes eletrônicos contra sobretensões.

Figura 13 – Imagem de um diodo zener típico

Fonte: Autoria própria

3.10 LED - Light Emitting Diode

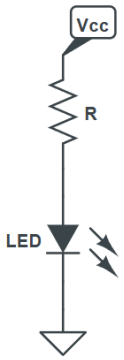
O Diodo Emissor de Luz, ou LED (*Light Emitting Diode*), é uma derivação da tecnologia do diodo semicondutor, porém, diferente do diodo comum o seu foco não é o controle de corrente, mas sim a conversão de energia elétrica em luz. Porém, assim como um diodo padrão, o seu sentido de corrente é unidirecional, ou seja, ele permite a passagem de corrente em um único sentido.

Com seu funcionamento baseando-se no fenômeno da eletroluminescência, quando o LED é polarizado diretamente, os elétrons da camada N se recombinam com as lacunas da camada P na junção. Durante este processo de recombinação, a energia excedente dos elétrons é liberada na forma de fótons, que são as partículas elementares da luz. A cor da luz emitida é determinada pela composição química do material semicondutor utilizado na junção.

Uma consideração fundamental na aplicação de LEDs é a necessidade de um resistor limitador de corrente conectado em série, geralmente denominados de R_{LED}. Sem este resistor, o LED tenderia a conduzir uma corrente excessiva, o que levaria à sua queima imediata, porém, como já visto no decorrer desse documento, os resistores possuem a capacidade de limitação de corrente.

Devido à sua alta eficiência energética, longa vida útil e robustez, os LEDs suplantaram outras tecnologias de iluminação e são amplamente utilizados como indicadores visuais em painéis, em *displays* de informação e em sistemas de iluminação geral. Podendo também variar o seu encapsulamento, podendo ser SMDs (*Surface Mounted Device*) ou estarem embutidos em outro dispositivo, como *displays* de 7 segmentos. A configuração comum de aplicação de um LED pode ser vista abaixo.

Figura 14 – Aplicação de um Led



Fonte: Autoria própria

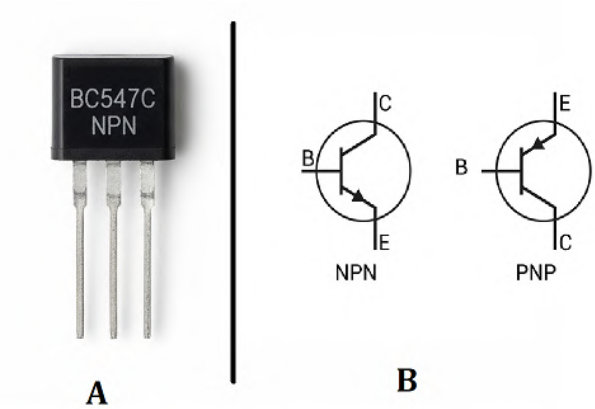
3.11 Transistor de Junção Bipolar

Tendo já abordado os componentes semicondutores de dois terminais, como os diodos, pode-se então avançar para o elemento fundamental para toda a eletrônica moderna: o transistor. Este é um dispositivo semicondutor de três terminais cuja função principal é amplificar sinais elétricos ou atuar como uma chave controlada eletronicamente, através da atuação na saturação ou corte. Sua invenção representou uma revolução, permitindo a miniaturização e a complexidade dos circuitos que conhecemos hoje.

O princípio do funcionamento de um transistor é baseado no controle do fluxo de corrente entre dois de seus terminais (o coletor e o emissor, em um transistor BJT) através da aplicação de uma pequena corrente ou tensão em um terceiro terminal (a base). Uma pequena variação no sinal de controle pode gerar uma variação muito maior no sinal de saída, caracterizando a amplificação. Quando operado nos extremos (corte ou saturação), ele funciona como uma chave, permitindo ou bloqueando completamente o fluxo de corrente, sendo este o princípio fundamental da lógica digital utilizada em microcontroladores e computadores. A sua construção se assemelha bastante a junção de três diodos, tendo esse sido o começo do seu desenvolvimento.

Existem diversos tipos de transistores, sendo os mais comuns o Transistor de Junção Bipolar (BJT) e o Transistor de Efeito de Campo de Óxido Metálico (MOSFET). A escolha entre eles depende da aplicação, considerando fatores como a potência, a velocidade de chaveamento e a impedância de entrada. Devido à sua versatilidade, são encontrados em praticamente todos os dispositivos eletrônicos, desde simples amplificadores de áudio até os processadores mais avançados. Para efeitos desse documento há somente a aplicação do transistor do tipo BJT.

Figura 15 – Visão de um transistor: (A) Encapsulamento; (B) Símbolo esquemático



Fonte: Autoria própria

Torna-se fundamental na escolha de um transistor se atentar algumas características principais, como a polarização, geralmente denominada como PNP ou NPN. O seu Beta, denominado pela letra β, assim como os seus limites de operação e faixa recomendada, todos esses podendo ser acessados no *datasheet* do fabricante.

3.12 Conectores de Entrada/Saída

Para a integração de um circuito funcional a outros equipamentos ou etapas de processos é necessário realizar uma conexão, podendo ser em formato de software, com envio via *bluetooth*, internet ou outros vínculos sem fio OtA(*over the air*), ou em formato de hardware, com ligações físicas, como cabeamento, soldagem ou outros canais rígidos. Portanto realizar a escolha ideal da conexão adequada é fundamental.

Devido a necessidade da interoperabilidade de dispositivos, é crucial que os sistemas eletrônicos possam operar em conjunto entre si, visando então esse cenário, as interfaces de conexão de entrada e saída se tornam o ponto inicial da análise, necessitando que os componentes eletromecânicos possuam uma forte aceitação de conexão pelo público-alvo, assim como possuam uma forte robustez e características desejadas por outros designers. As conexões podem variar de método, podendo ser soldadas, de forma a ser fortemente robustas porém dificilmente alteráveis, ou então modulares, permitindo maior flexibilidade porém com susceptividade ao ruído.

A escolha de quais conectores específicos utilizar é uma escolha estratégica realizada pelo designer do circuito, porém geralmente é realizada seguindo padrões pré-estabelecidos da industria, como optando por conectores universais como USB ou conector Jack para conexões externas soldadas, permitindo a associação com ecossistemas bem definidos, porém internamente pode-se escolher opções como barramentos de pinos e outros conectores modulares, visando a flexibilidade de possibilidade de consertos ou alterações futuras,

sendo fundamentais para a depuração de código e a fácil substituição de módulos.

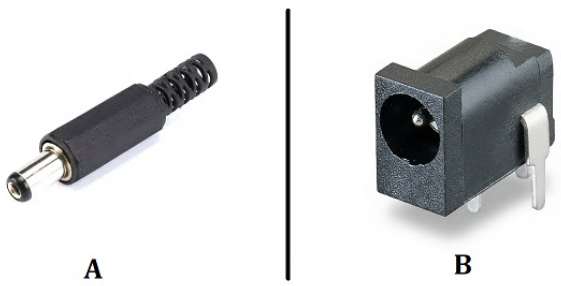
3.12.1 Conector tipo Jack

Visando a alimentação DC de um circuito, a utilização do componente eletromecânico denominado de conector tipo Jack é uma padronização comum e robusta altamente aplicada na interconexão de dispositivos, sendo principalmente aplicado em cenários de alimentação contínua. Sendo um conjunto segmentado em dois modos, o plugue (macho) e o soquete (fêmea), a sua construção mais comum é realizada de forma que o conector seja de formato cilíndrico, com dois contatos isolados entre si, com o pino central servindo como entrada de tensão (+Vcc) e a armadura (*shield*) que está ao redor como sendo o negativo ou *ground* (GND).

No contexto deste projeto, o conector Jack é uma solução robusta e padronizada para fornecer alimentação externa ao circuito, oferecendo uma conexão mais segura e confiável do que pinos ou fios expostos. Portanto, visando a sua aplicação prática, o soquete é tipicamente posicionado diretamente na placa de circuito impresso (PCB) o visando a utilização conveniente, geralmente com sua saída localizada nas extremidades da placa.

Quando utilizando o conector Jack é importante a verificação da polaridade, já que o padrão de de polaridade pode variar de acordo com o fabricante, de forma que uma inversão pode causar danos irreparáveis ao circuito. Além da polaridade também é fundamental a observação ao valor máximo de corrente aceito pelo conector, assim evitando um derretimento devido a excesso de temperatura.

Figura 16 – Exemplo de um conector jack: (A) Tipo macho; (B) Tipo fêmea



Fonte: Autoria própria

3.12.2 USB A

Em quase todos os dispositivos eletrônicos atualmente irá ser identificada a utilização de uma conexão USB, sendo ela altamente disseminada e também regida por protocolos bem definidos e resguardados internacionalmente, sendo o USB tipo A a sua construção clássica e mais típica de se encontrar, dessa forma, se tornando a opção mais clara para aplicações de dispositivos caso se busque a generalização da solução.

Passível do fornecimento tanto de comunicação de dados na forma serial quanto de energia, a sua construção interna permite o uso eficiente para aplicações práticas. Com quatro canais, sendo dois dedicados a alimentação DC (VCC e GND) e dois que formam um par diferencial para transmissão de dados (D+ e D-), a aplicação em cenários variados faz com que o USB seja a escolha ideal em todos os meios.

Nesse projeto, embora abranja comunicação serial, não irá utilizar os canais de dados do USB, utilizando-o exclusivamente como porta de alimentação de energia, essa decisão provêm da praticidade e conveniência para o usuário final que esse conector gera, possibilitando que o dispositivo final seja alimentado por fontes comuns, como carregadores de telefoes ou computadores, até a cenários visando a mobilidade, com utilização de bancos de bateria (*power banks*), tornando o uso do equipamento mais flexível e acessível em diferentes cenários.

Durante a utilização do conector USB é importante que o *shield* seja apropriadamente aterrado, garantindo visando uma apropriada blindagem contra interferência eletromagnética (EMI), e evitando também possíveis danos relacionados a descarga de eletricidade estática (ESD), sendo fundamentais as proteções para cenários de aplicação de dispositivos em seções de testes com voluntários.

3.12.3 Terminal Block

Considerando a inclusão de sensores e dispositivos com alta necessidade de manutenção, é comum a utilização de *terminal blocks*. Sendo uma conexão extremamente robusta, com foco para fixação de fios e cabos de forma semi-permanente através de parafusos gerando força suficiente para evitar deslizes e solturas. Sendo bastante análogo as dispositivos empregados em eletrotécnica, como tomadas e disjuntores, ele opera da mesma maneira porém em um cenário de eletrônica, sendo reduzido e com limites de corrente e tensão geralmente reduzidos, permitindo a conexão a uma fiação externa qualquer.

Sendo especialmente indicado para usos onde as conexões requerem uma maior capacidade de corrente, ou com conexões cujos terminais fogem dos conectores padrões, como entradas de alimentação de alta potência ou entradas de motores, nesse projeto ele é utilizado para conexões com sensores que necessitam de manutenção e calibração constante, como é o caso dos FSRs, que também necessitam estar empregados distante da placa, dessa forma utilizando cabos para extensão.

É importante considerar no momento de projetar o circuito a seleção de modelos de *terminal blocks* que sejam suficientes para aplicação, ou seja, verificando os parâmetros de distancia entre fios, bitola do fio utilizada e capacidade de corrente para a aplicação. Porém, é ainda mais fundamental considerar que o método de conexão utilizada por esse conector é passível de ruídos, principalmente em cenários de corrosão do fio/cabo ou caso não haja pressão suficiente para evitar sua movimentação.

Figura 17 – Imagem de um conector borne

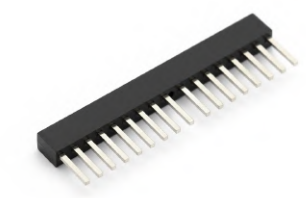
Fonte: Autoria própria

3.12.4 barramento de pinos

Análogo a um *protoboard*, o barramento de pinos opera de forma os dispositivos eletrônicos possuam uma conexão simples e versáteis ao circuito, sendo utilizado principalmente pelo seu custo extremamente reduzido mas permitindo ainda uma forte flexibilidade para remoção dos dispositivos, geralmente para *debugging*, podendo remover um MCU ou circuito integrado (CI)..

Justamente devido a sua simples construção, sendo padronizada pelo espaçamento entre terminais (ex. 2,54 mm), é muito comum encontrar a sua aplicação associada a dispositivos progamáveis, geralmente sendo necessário para futuras atualizações ou manutenções do sistema, habilitando que um sistema seja facilmente reconfigurado ou atualizado sem a necessidade de soldagem e dessoldagem. Porém, assim como os *terminal blocks*, o uso dos barramentos de pinos torna o sistema vulnerável a ruídos.

Figura 18 – Imagem de um barramento de pinos eletrônicos



Fonte: Autoria própria

3.12.5 Chave de 3 estados

Considerando a alteração de estados, a aplicação de chaves é o método mais fundamental, seja aplicando um BJT como chave ou até um interruptor comum, nesse cenário, há também a aplicação de chaves de 3 estados, fornecendo um controle com três posições operacionais possíveis, ou seja, ele permite o direcionamento de um polo comum até três posições físicas distintas, no cenário do projeto, do sinal de entrada do circuito para três fontes possíveis.

A aplicação de uma chave de três estados é fundamental para a seleção dos modos de operação de um circuito, gerando possíveis cenários independentes entre si, com uma isolação física estrita entre os estados. Sendo o seu uso preferível em cenários de conexão a fontes de energia, onde cada modo se associa a alimentações distintas, porém sendo ainda necessário se atentar que deve-se aplicar para baixos valores de tensão insuficientes para gerar um arco elétrico entre os terminais da chave.

É importante considerar que ao utilizar esse componente o efeito do contato entre terminais ao realizar a mudança de estado irá gerar ruídos ou picos de tensão no momento da comutação, portanto a aplicação de capacitores em paralelo é recomendado. Também é crucial verificar no *datasheet* do componente se suas especificações de corrente e tensão são adequadas para a carga que ele irá comutar.

Figura 19 – Imagem de uma chave seletora de três estados

Fonte: Autoria própria

3.12.6 Botão e Chaves

Na eletrônica, é bastante comum que a terminologia de componentes eletromecânicos simples como botões e chaves de um estado, que compartilham a função simples de abrir ou fechar um contato elétrico, sejam utilizados alternadamente, porém para fins de dinstinção é importante olhar para o seu modo de atuação, os botões (*push-button*) são dispositivos de ação curta e breve, ou seja, ele estabelece uma conexão somente quando pressionado, e quando deixado de interagir, retorna a sua posição padrão. Já a chave (*switch*) opera de

forma mais permanente, ou seja, mantém o seu estado mesmo após o fim da interação.

A aplicação desses componentes no circuito não poderia ser mais simples, eles são inseridos em série com alguma carga ou seção do circuito que se deseje controlar, por exemplo, uma chave é aplicada em série com a entrada de alimentação de um circuito, e quando desativada serve como *shut-off*. Já o botão tem o padrão comum de ser utilizado como forma de *reset* em um sistema, fornecendo um sinal de curta duração a um pino específico de um sistema, geralmente um microcontrolador, atuando dessa forma como gatilho para outra ação futura.

Portanto, análogo aos outros conectores discutidos, o ruído perdura como um male associado ao seu uso, porém sendo menos comum. Para cenários de aplicação de chaves é importante observar e se precaver contra arcos elétricos que inutilizem a aplicação da chave, já em botões é importante evitar que o seu valor fique em um estado "flutuante"quando associado a um microcontrolador, geralmente necessitando adicionar resistores de *pull-up* ou *pull-down*, que irão forçar a entrada a um nível lógico definido.

3.13 motor de vibração

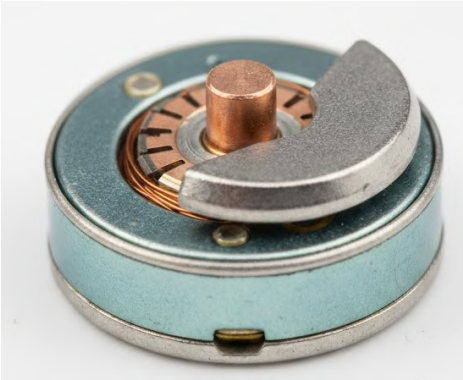
O motor de vibração é um atuador eletromecânico cuja finalidade principal, como disposto em seu nome, é realizar vibração tátil e perceptível com base na utilização de um motor. Transformando assim esse transdutor em um dispositivo de *feedback* háptico, provendo uma resposta física ao usuário por meio do sentido do tato. A construção consiste em um motor, geralmente sendo de corrente contínua, de pequeno porte onde no eixo é acoplada uma massa excêntrica, ou seja, um peso desbalanceado geralmente de forma cilíndrica com um chanfrado, dessa forma quando operando com uma rotação em alta velocidade esse peso desbalanceado gera uma força centrífuga irregular, se manifestando como a vibração sentida em todo o corpo do motor e então transferida ao usuário.

A aplicação de vibradores é extremamente presente no dia a dia, escondido em um aparelho cotidiano, sendo fortemente associados a alertas silenciosos, estando principalmente aplicados nos dispositivos móveis, como telefones celulares, associados a função *vibracall*. A sua forte difusão demonstra claramente a sua eficácia, com atuação análogo a qualquer outro motor DC, ele é tratado como uma carga de saída associada a uma lógica de controle.

Expandindo o seu uso para microcontroladores, é comum que o MCU não seja capaz de fornecer a corrente necessária para acionamento do motor, sendo nesse caso a interface de conexão sendo realizada através de um transistor BJT atuando como chave, ou seja, oscilando entre o corte ou saturação. Partindo então dessa conexão, pode-se utilizar o controle via *software* para traduzir em uma resposta física e tangível para o usuário, como um alerta ou a confirmação de um comando.

É porém fundamental considerar que o motor de vibração é ainda um motor DC típico, ou seja, é importante considerar a sua propriedade indutiva, portanto, assim como os demais, o motor de vibração irá armazenar energia em seu campo magnético enquanto em operação, porém quando desligado a energia irá ser liberada de forma como um pico de tensão reversa, podendo então danificar os demais componentes associados, principalmente os relacionados ao controle. Dessa forma, para resguardar contra os efeitos das cargas indutivas, é típico a adição de um diodo de *flyback*, sendo conectado em paralelo com o motor, embora em polaridade reversa. Portanto irá então prover um caminho seguro para a corrente induzida, servindo de proteção para o restante do circuito. Sendo também prudente a isolação dos demais componentes dos motores, de forma a evitar o ruido elétrico gerado pela sua operação.

Figura 20 – Imagem de um motor de vibração, visualizando o peso desbalanceado



Fonte: Autoria própria

3.14 Circuitos Integrados

Um Circuito Integrado (CI), é um circuito eletrônico completo, miniaturizado em uma única e minúscula pastilha de material semicondutor, geralmente silício, envolucro em uma estrutura plástica e usualmente seguindo algum dos padrões pré-definidos de encapsulamento. A sua maior vantagem é com relação a simplificação de funções, economia de espaço, custo, velocidade e confiabilidade em comparação com circuitos montados com peças individuais, visto que em estrutura através das técnicas avançadas de manufatura é possível manipular algumas características dos componentes discutidos previamente, como transistores, resistores e capacitores.

3.14.1 Regulador de tensão LM7805

O LM7805 é um circuito integrado operando como regulador de tensão positiva que se estabeleceu como uma solução fundamental, robusta e altamente difundida no âmbito de fontes de alimentação para circuitos eletrônicos. Sua função principal opera de forma crítica para o funcionamento apropriado dos circuitos, sendo como encargo receber

uma tensão de entrada mais elevada, geralmente instável ou ruidosa, e fornecer em sua saída uma tensão contínua, estável e precisa de +5 volts. Dessa forma o LM7805 é um sistema de regulação completo, oferecendo uma solução sólida e de fácil implementação para criar o ambiente de alimentação estável que a grande maioria dos circuitos digitais e microcontrolados exige para operar de forma confiável.

Tendo como seu encapsulamento o TO-220, esse componente opera com somente três terminais: entrada (*Input*), terra (*Ground*) e saída (*Output*). Utilizando esses três pontos de conexão o LM7805 consegue operar como um intermediário que garante uma alimentação limpa e constante, sendo fortemente associado a uma fonte de energia instável, como uma bateria cuja tensão cai com o uso, evitando um comportamento errático nos componentes sensíveis.

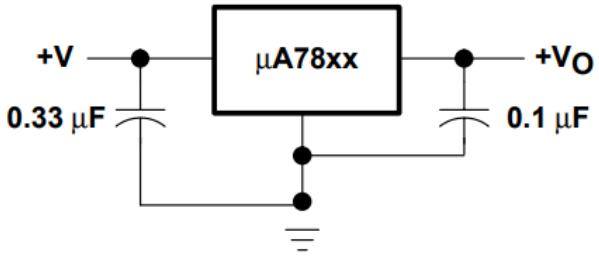
É fundamental o uso um capacitor na entrada (tipicamente 0,22µF ou 0,33µF) e outro na saída (tipicamente 0,1µF), tendo o capacitor de entrada a função principal a de filtrar ruídos da fonte, enquanto o capacitor de saída é essencial para suavizar a tensão entregue à carga, evitando picos, servindo geralmente como capacitor de desacoplamento.

Os Reguladores da familia do LM78xx dissipam a energia excedente na forma de calor, e a quantidade de calor gerado é proporcional à diferença entre a tensão de entrada e saída, dessa forma tornando a gestão térmica um conceito importante durante o *design* dos protótipos. Portanto é importante o dimensionamento da tensão de entrada e saída, aplicando quando necessário o uso de um dissipador de calor (*heatsink*) acoplado à aba metálica do componente.

Em cenários em que a tensão de entrada decair mais rapidamente que a tensão de saída do LM7805 é possível que a junção do componente seja danificado, dessa forma é aplciado um diodo em reverso (*reverse-bias*) para proteção.

Adicionalmente, para todo regulador de tensão deve-se atentar ao fenômeno do *ripple*, que é a ondulação residual presente em uma tensão DC obtida a partir da retificação de uma fonte AC. O LM7805 atua ativamente para filtrar essa variação, entregando uma tensão de saída muito mais limpa e contínua do que a que recebe. No entanto, para que ele funcione corretamente, a tensão de entrada, mesmo no ponto mais baixo da ondulação, deve ser sempre superior à tensão de saída somada à tensão de *dropout* (aproximadamente 2V). Portanto é fundamental que o componente opere dentro da sua faixa normal, podendo ser consultado do *datasheet*, ignorar essas condições pode fazer com que o regulador perca sua capacidade de regulação nos vales da ondulação, comprometendo a estabilidade da alimentação.

Figura 21 – Recomendação de aplicação de um LM7805



Fonte: Adaptado pelo autor | https://cdn.sparkfun.com/assets/1/7/7/3/2/LM7805.pdf

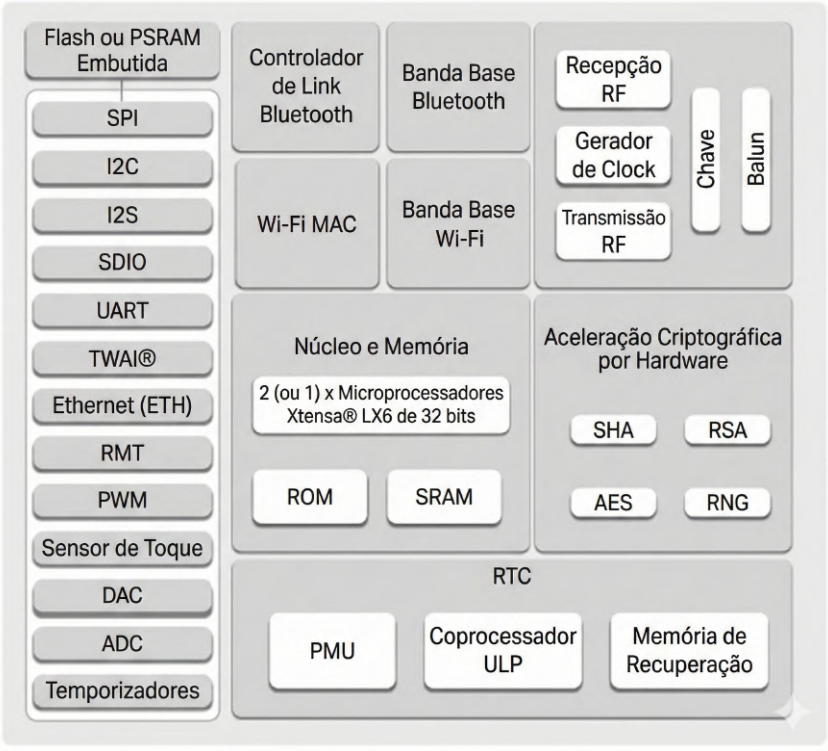
3.14.2 microcontrolador ESP32

Entrando no conceito de processamento lógico, é preciso primeiro compreender alguns conceitos básico. Um Microcontrolador (MCU) é um dispositivo que une um centro de processamento com armazenamento de memória, também possuindo muitas vezes outras funções visando aplicações em eletrônica, como periféricos de entrada e saída, visando ser uma solução completa de pronto uso.

Aprofundando no cenário de MCUs, o kit de desenvolvimento baseado no ESP32 surge como uma solução bastante popular e de alto desempenho, com o seu forte diferencial sendo a sua integração nativa com conectividades Wi-Fi e Bluetooth a um custo de aquisição extremamente baixo quando comparado com seus similares de mercado

Nos projetos, ele é frequentemente disponibilizado em uma placa de desenvolvimento (*devkit*) que permite acesso aos seus pinos através de um barramento, facilitando a prototipagem e a conexão modular ao restante do circuito, usualmente associado a utilização de um pino de barramentos, já discutido previamente no decorrer desse texto.

Figura 22 – Diagrama de bloco interno da ESP32

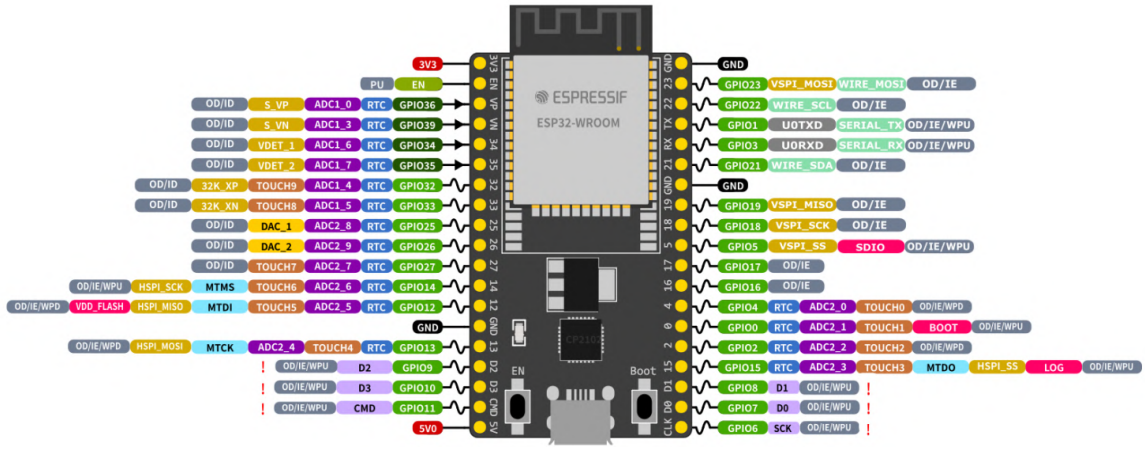


Fonte: Adaptado pelo autor | https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf

A arquitetura do ESP32 utiliza o conceito de *System-on-Chip* (SoC), onde o seu poder de processamento está contido em seu núcleo com um ou dois processadores de 32 bits. Atuando como unidade central de controle, esse núcleo é responsável por executar a lógica de programação criada pelo programador e gerenciar todos os periféricos integrados, utilizando então os seus outros módulos como suporte para exercer a lógica criada. Devido a essa capacidade de processamento robusta o ESP32 consegue executar desde tarefas simples de controle, como operação de transistores como chaves, até algoritmos mais complexos, como o processamento de sinais ou a execução de um servidor web local.

Para integração com circuitos, o ESP32 depende de um conjunto diversificado de periféricos integrados ao kit de desenvolvimento, podendo ler sensores analógicos através de seus múltiplos Conversores Analógico-Digital (ADC) de 12 bits e gerar sinais de tensão analógica por meio de seus Conversores Digital-Analógico (DAC) de 8 bits. A comunicação com outros circuitos integrados, sensores e módulos é realizada diferentes interfaces de comunicação serial, que incluem múltiplas portas UART, SPI e I2C, possibilitando a aplicação de diferentes procolos de comunicação, além de métodos de comunicação sem fio, como Wi-Fi e Bluetooth.

Figura 23 – Diagrama de pinos do kit de desenvolvimento da ESP32



Fonte: Adaptado pelo autor | https://docs.espressif.com/projects/esp-dev-kits/en/latest/esp32/esp-dev-kits-en-master-esp32.pdf

O grande diferencial da ESP32 com relação aos seus similares na mesma faixa de preço é sua capacidade de comunicação sem fio. O dispositivo integra um rádio de 2.4 GHz que suporta tanto Wi-Fi (padrão 802.11 b/g/n) quanto Bluetooth (Clássico e de Baixa Energia - BLE). Essa característica realiza um *upgrade*, transformando em um dispositivo de capacitado para operar com Internet das Coisas (IoT) nativamente. Através do Wi-Fi, ele pode se conectar a redes locais (WLAN) e à internet para enviar dados para servidores na nuvem, como por *Message Queuing Telemetry Transport* (MQTT), receber comandos remotamente ou interagir com APIs. O Bluetooth, por sua vez, permite a comunicação direta com smartphones, e outros periféricos próximos transformando o circuito associado a ESP32 em um *gadget*.

Para realizar ações e reações com o kit de desenvolvimento é necessário primeiro realizar a programação lógica associada, realizada em um Ambiente de Desenvolvimento Integrado (IDE), sendo o Arduino IDE uma das plataformas mais populares para este fim, devido a sua forte integração e difusão no meio de desenvolvedores. Do ponto de vista da aplicação, a viabilidade de um projeto depende também de uma alimentação de energia apropriada, tendo o ESP32 a necessidade de uma tensão de entrada regulada e estável, tipicamente de 5V pelo pino V_{in} ou via sua conexão USB. Contanto com uma de suas características mais importantes para projetos modernos sendo seu baixo consumo de corrente, com capacidade ainda para otimização por meio de modos de economia de energia, como o *Deep Sleep*. Essa eficiência energética permite que o ESP32 opere por longos períodos a partir de uma bateria externa, tornando-o a escolha ideal para a criação de dispositivos móveis, portáteis e autônomos.

3.15 Software

Seguindo a mesma metodologia já empregada, nesta seção será exploradas as necessidades para a realização e funcionamento apropriado do código lógico requerido para operar o sistema. Nesse momento será falado o "O que é utilizado?", e em seções posteriores será explorado o "Como é utilizado?".

3.15.1 Linguagens e estruturas de programação

A aplicação de diversas linguagens de programação em um único projeto se tornou uma prática comum no mercado atual, portanto, esse projeto não será exceção. Cada linguagem é selecionada com base em suas aplicações para a tarefa necessária, seja para o controle de baixo nível do microcontrolador, via uma linguagem mais simples e rápida, ou para a flexibilidade na manipulação de dados e criação de interfaces no computador. Podendo então dividir as aplicações realizadas via distintas linguagens como *software* e *firmware*.

Para programação na ESP32 foi utilizada a linguagem C++, utilizando como base o *Framework* do Arduino. Utilizada para o desenvolvimento do *firmware*, responsável pelo controle do microcontrolador ESP32. A escolha se mostra ideal pois a linguagem combina alto desempenho e velocidade de execução, essenciais para o processamento de dados em tempo real, com uma implementação simplificada graças ao seu uso em sala de aula.

Para a programação do *software* foi utilizado o Python. Fundamental na aplicação do lado do computador, ela foi escolhida por ser uma linguagem bastante abrangente e difundida, permitindo a integração de tarefas de diversas áreas com um único *script*, ainda mais pela sua simplicidade e praticidade de busca por fóruns ou locais de pesquisa.

Embora não seja uma linguagem de programação, o JSON (*JavaScript Object Notation*) desempenha um papel fundamental como o formato de transmissão de dados que unifica a comunicação entre os dois sistemas. Possuindo uma padronização mundial, ele atua como uma "linguagem"comum entre dispositivos, o *firmware* em C++ no ESP32 estrutura os dados dos sensores e os codifica em uma *string* de texto JSON. Em seguida essa *string* é então transmitida e, ao ser recebida pela aplicação em Python, é decodificada de volta para uma estrutura de dados nativa (um dicionário), permitindo então uma interoperabilidade entre sistemas distintos.

3.15.2 Ambientes de desenvolvimento

Para escrita dos códigos foram utilizados Ambientes de Desenvolvimento Integrados (IDEs), sendo eles responsáveis por gerenciar e visualizar o código assim como compilar e depurar. Para um projeto com componentes distintos como este, é comum utilizar IDEs diferentes, cada uma especializada em otimizar o fluxo de trabalho para a linguagem e a

plataforma em questão, seja ela o *firmware* do dispositivo embarcado ou a aplicação no computador, porém geralmente estão associadas com a experiência que o usuário possui com cada uma delas.

Para o C++ foi utilizado o Arduino IDE, servindo como o ambiente de desenvolvimento integrado para escrever, compilar e carregar o *firmware* no ESP32. Sua simplicidade, gerenciamento integrado de placas e bibliotecas, e o monitor serial embutido a tornam uma ferramenta eficiente para o desenvolvimento e depuração rápida do sistema embarcado. Assim como o seu fórum dedicado, tornando a busca e resolução de dúvidas mais direcionada.

Considerando a linguagem Python, foi decidido utilizar o PyCharm. É uma IDE profissional e gratuita que oferece ferramentas avançadas de depuração, análise de código e gerenciamento de projetos, assim como de customização visual.

3.15.3 Bibliotecas empregadas

Visando o desenvolvimento de aplicações é sempre comum a associação a conjuntos de bibliotecas para a sua construção, trabalhando como um conjunto lógico pré-escrito altamente eficiente para realização de tarefas, por vezes simples, de maneira dinâmica, permitindo ao programador focar no que importa. Evitar "reinventar a roda"possui o efeito importante de reduzir o tempo e complexidade necessária para a aplicação da solução, viabilizando o *go-to-market* da solução. Segmentando em duas partes, tem-se as bibliotecas empregadas para cada IDE no decorrer dessa seção, iniciando com o *firmware* do ESP32 (C++):

- **ArduinoJson**: Sendo necessária para criar e serializar dados no formato JSON dentro do ESP32, o uso dessa biblioteca foi motivado pela sua alta praticidade e baixo consumo de memória, sendo fundamental para a operação limpa porém organizada do processamento de dados, assim como também sendo referência para aplicações envolvendo JSON. Garantindo que os dados sejam enviados em um formato padronizado, compatível com diversos dispositivos distintos e facilmente analisável, ela permite a integração com outros sistemas, como aplicativos móveis ou *softwares* de terceiros.

- **WiFi**: Sendo a biblioteca padrão do ESP32 para gerenciar a conectividade com redes *Wi-Fi*, seu uso é essencial, pois estabelece a camada de rede necessária para que o protocolo MQTT possa se comunicar com a internet. Sem a aplicação dessa biblioteca não teria a possibilidade de aplicação do módulo de *Wi-Fi* do kit de desenvolvimento.

- **PubSubClient**: Empregada para implementar a comunicação via protocolo MQTT. Esta biblioteca se utiliza da base fornecida pela biblioteca *WiFi* e oferece um cliente

MQTT leve e funcional, permitindo que o ESP32 publique mensagens em um *broker* de forma simples e confiável, sendo a peça-chave para a funcionalidade IoT do projeto, sendo também extremamente famosa, com diversos exemplos possíveis de estudo nos fóruns e em seu repositório.

Partindo para a análise da parte do desenvolvimento relativo a linguagem Python, tem-se então a aplicação das seguintes bibliotecas:

- **serial (pyserial)**: Sendo aplicado para estabelecer a comunicação com dispositivos pela porta USB/Serial, essa biblioteca pode ser utilizada para abrir, configurar e ler o fluxo de dados enviado por um dispositivo. Necessitando porém da definição de algumas variáveis de início, como o *baudrate* e porta associada, como a 'COM3'.
- **json**: Empregada como a contraparte da ArduinoJson, sendo ela utilizada como receptora. Sua função foi decodificar (*parse*) a *string* de texto JSON recebida, convertendo-a em um dicionário Python para que os dados de cada sensor pudessem ser facilmente acessados e processados.
- **OpenCV (cv2)**: Biblioteca central para o processamento de imagem e visualização do Python. Foi utilizada para carregar uma imagem de fundo, nesse caso foi utilizado a visualização de uma mão, criar uma janela de exibição e, principalmente, para modificar dinamicamente os *pixels* dessa imagem de acordo com o nível do sensor, representando graficamente os dados de força em tempo real.
- **NumPy**: Utilizada como a base para as operações do OpenCV, necessitando para trabalhar em conjunto. As imagens no OpenCV são representadas como *arrays* NumPy, e esta biblioteca forneceu as ferramentas para as manipulações matemáticas necessárias para alterar a imagem a nível de *pixel*.
- **threading**: Empregada para gerenciar a execução de tarefas concorrentes via *multithreading* do processador do computador. No código, foi utilizada para iniciar a plotagem dos gráficos de cada sensor em uma *thread* separada. O objetivo dessa abordagem é melhorar a responsividade da aplicação, evitando que o fluxo principal do programa, principalmente da leitura dos dados enviados pela comunicação serial, seja bloqueado por operações que possam ser demoradas.

3.15.4 Protocolos de comunicação aplicados

É fundamental que ambos os lados da comunicação, o responsável por enviar e o responsável por receber, estejam com seus parâmetros de configuração alinhados entre si para que a troca de informações em qualquer protocolo de comunicação seja bem-sucedida. Sem essa definição prévia sobre os parâmetros da comunicação, os dados podem ser

perdidos ou interpretados de forma incorreta, geralmente ocorrendo por meio de leitura parcial.

A Comunicação Serial, via hardware UART (*Universal asynchronous receiver/ transmitter*), é um dos métodos mais fundamentais de comunicação digital, geralmente disponível em quase qualquer placa de desenvolvimento. Operando sobre um par de fios, TX para transmissão e RX para recepção, de forma que o compartilhamento de informação é realizado de forma assíncrona, com o tempo é sincronizado pelo *baudrate*, sendo esse um parâmetro que define a velocidade da transmissão de *bits* por segundo.

No projeto, a Serial é a atua servindo a múltiplos propósitos, primeiro serve por método de *upload* de código, porém é também o meio pelo qual mensagens de status são enviadas para realização de depuração. Por fim, é também o canal pelo qual o fluxo contínuo de dados JSON é enviado para a aplicação local em Python. Tendo também o como forte vantagem a realização de comunicação de forma offline, o serial é um método bastante útil para comunicação adicional.

O MQTT (*Message Queuing Telemetry Transport*) é um protocolo de mensagens projetado especificamente para aplicação na Internet das Coisas, operando com base na conexão com internet, ele utiliza uma arquitetura de Publicação/Assinatura (*Publish/Subscribe*) para envio e recepção dos dados.

O seu grande diferencial do modelo cliente-servidor tradicional, é a definição de uma entidade central, denominada *broker* responsável por gerenciar as comunicações entre dispositivos de forma a operar como um "canal"para os dispositivos. Aqueles dispositivos atuando como *Publishers*, similar ao ESP32 nesse projeto, enviam mensagens para "tópicos"específicos no *broker*, sem definição de quem as receberá. Por outro lado, os dispositivos atuando como *Subscribers* podem se inscrever nesses mesmos tópicos para receber as mensagens de interesse. Dessa forma, tem-se o *broker* operando como uma central de distribuição, desacoplando completamente os produtores de informação dos consumidores.

A aplicação do MQTT combina perfeitamente com o IoT, devido a sua leveza e praticidade de aplicação, assim como economia de banda e energia, crucial para dispositivos embarcados, e por fim devido a sua forte difusão no meio de comunicação atual. Há também de se considerar que o desacoplamento entre o *Publisher* e *Subscriber* permite uma escalabilidade imensa, ou seja, múltiplos dispositivos, como ESP32, podem publicar dados e inúmeros clientes, como aplicativos móveis e servidores, podem se inscrever para recebê-los, sob anonimidade.

O protocolo também define mecanismos avançados como Qualidade de Serviço (QoS), que garante diferentes níveis de entrega de mensagem, e o *Last Will and Testament* (LWT), que permite ao *broker* notificar outros clientes caso um dispositivo se desconecte abruptamente, definindo também, em alguns casos, a persistência de uma última mensagem.

No projeto, o MQTT é o que transforma o sistema de um dispositivo local para uma verdadeira plataforma IoT, permitindo o monitoramento remoto e a integração com um ecossistema global de aplicações.

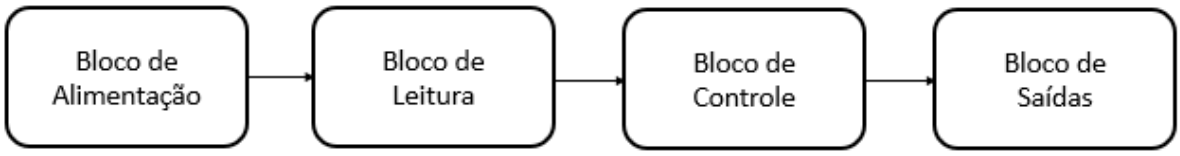
4 METODOLOGIAS EMPREGADAS

Nessa seção, tendo já discorrido sobre os materiais que compõe o projeto, será agora abordado os métodos em que tais materiais foram utilizados, ou seja, se na etapa anterior foi mencionado o "O que?", agora será abordada a sua aplicação, ou seja, o "Como?".

4.1 Hardware

O circuito pode ser considerado como possuindo 4 partes distintas, porém que operam em conjunto. Cada bloco possui sua interação direta ou indiretamente com os demais, porém todos contribuem para que seja realizado o processo final de medição de sinal.

Figura 24 – Diagrama de blocos do circuito

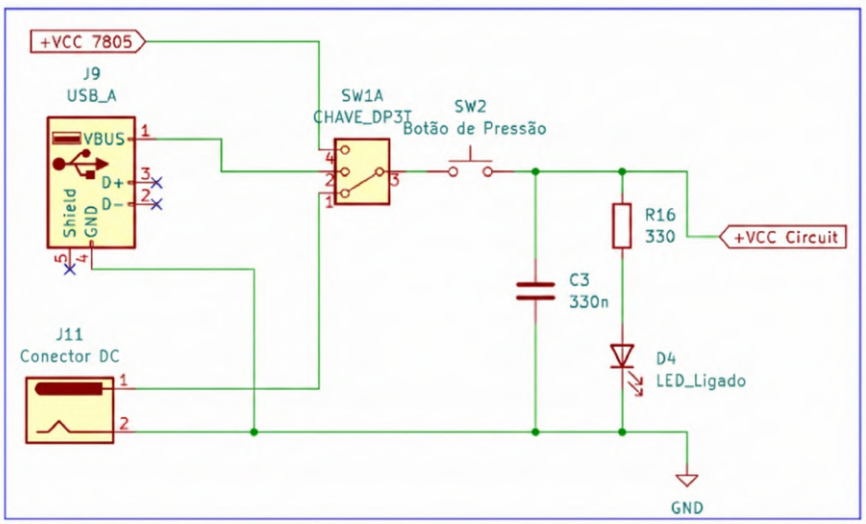


Fonte: Autoria própria

4.2 Alimentação

A alimentação é a base da construção do circuito, com o restante do circuito contando com a sua estabilidade e capacidade, portanto, o circuito proposto nesse projeto não é diferente. Contanto com quatro formas distintas de possibilidade de alimentação, a sua construção foi focada com base na versatilidade e usabilidade em diferentes cenários, tendo todos os seus componentes já discutidos em capítulos anteriores. Sendo importante salientar que uma das conexões, a utilizando o *terminal block* (J1) foi utilizada como forma de *debugging*, focado em conexão direta com uma fonte externa durante momentos de prototipagem.

Figura 25 – Bloco de entrada de alimentação do circuito



Fonte: Autoria própria

Começando pela chave seletora de três estados (SW1A), ela serve como elemento decisório para o método de alimentação do circuito, cada um com uma entrada individual. Sendo porém importante salientar que embora haja alteração na fonte, os terras são conectados, podendo ser alterado por uma outra chave seletora, se o leitor acreditar ser mais apropriado, portanto é recomendado a utilização de somente uma forma de alimentação por vez, por isso a chave seletora foi adicionada.

As conexões diretas do tipo USB tipo A e o conector tipo Jack servem principalmente para conexão de fontes de alimentação externas, tal qual fontes de outros dispositivos que o leitor possa possuir, desde que atendendo as especificações de tensão (5V) e corrente suficiente ($\geq$ 1A). Para o USB tipo A também foi considerada a utilização de bancos de bateria externos, tais quais os comumente vendidos como acessórios para celulares, dessa forma tornando a placa para um dispositivo móvel, capaz de ser atrelado a indivíduos ou outros aparelhos.

O botão (SW2) em série com a chave seletora tem como função a remoção completa de tensão ou corrente do circuito, abrindo-o de forma a cessar a passagem de energia, ou seja, operando como um botão de ligar/desligar. Em paralelo com a saída do circuito tem-se um capacitor (C3) de 330nF, operando como filtro de ruído, servindo também para estabilizar a tensão, suavizando quaisquer variações mínimas, e garantir a longevidade dos seus componentes devido a maior qualidade do sinal, sendo sempre presentes em aplicações de fontes de tensão ou similares.

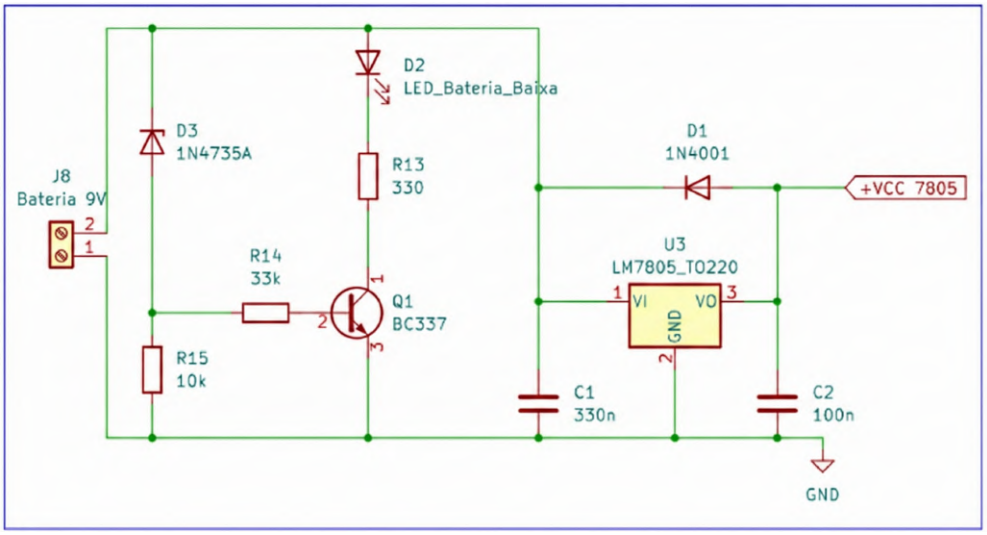
Em paralelo ao capacitor (C3) há então um led (D4) responsável pela indicação visual de operação do circuito, estando somente aceso em cenários em que há tensão sendo aplicado sobre ele. Como discutido previamente, o led necessita de um resistor (R16) em

série para a limitação da corrente, dessa forma evitando uma deterioração do componente. O valor do resistor foi escolhido com base na equação abaixo:

$$R_{Led} = \frac{V_{cc} - V_{Led}}{I_{Led}}$$

Tendo o valor de V_{cc} equivalente a 5V e os valores de V_{Led} e I_{Led} tendo sido consultados previamente em *datasheet* respectivo ao led, para esse cenário, decidiu-se reduzir a corrente de led para metade do seu valor máximo e considerou-se uma faixa de variação da tensão de led, portanto os valores foram $I_{Led} = 10mA$ e $1,8V \geq V_{Led} \geq 2,0V$, por fim, aproximou-se o valor do resistor para aquele perto de um valor comercial.

Figura 26 – Bloco de entrada de alimentação e regulação do circuito



Fonte: Autoria própria

Partindo então para o segundo bloco do circuito de alimentação, esse opera então com base em alimentação por meio de uma fonte externa conectada ao *terminal block* (J8), foi considerado e calculado o uso de uma bateria de 9V, trabalhando em conjunto com um regulador de tensão LM7805.

O Led (D2) opera em conjunto com o resistor (R13), estando ele operando como limitador da corrente, para indicação do nível de tensão restante da bateria, ou seja, informando em cenário em que a tensão da bateria caia para níveis insuficientes para a operação do regulador LM7805, tendo sido configurado de forma que o led (D2) irá gradualmente aumentar a luminosidade até o ponto máximo, de forma que indicará que a bateria está com de acordo com o requerido ($V_{bat} \geq 7,5$V).

O diodo zener (D3) opera de forma a aplicar uma tensão fixa de 6,8V sobre si, então seguindo com base nesse pressuposto, e considerando a tensão de bateria como V_{bat} teremos então que a queda de tensão sobre o resistor R15 será equivalente a $V_{R15} = V_{bat} - V_z$.

Para um transistor operar em corte algumas condições devem ser satisfeitas, uma delas é ter o $V_{BE} < 0,7V$. Em um cenário no qual a tensão da bateria (V_{bat}) vá decaindo, porém ainda superior a tensão necessária para o zener (D3), então haverá um cenário onde a tensão do resistor R15 (V_{R15}) será inferior a 0,7V, dessa forma garantindo que $V_{BE} < 0,7V$. Sendo que esse cenário ocorre quando: $0,7V < V_{bat} - V_z$, e supondo $V_z = 6,8V$ como fixo, então $V_{BE} < 0,7V$ quando $V_{bat} < 7,5V$.

Quando o transistor estiver em corte, então também haverá uma redução da luminosidade do led (D2), dessa forma o apagando. Portanto, quanto mais próximo do corte o transistor for se tornando, menos luminoso o led (D2) estará até o momento que ele se apague, dessa forma, indicando que a bateria deve ser recarregada ou substituída.

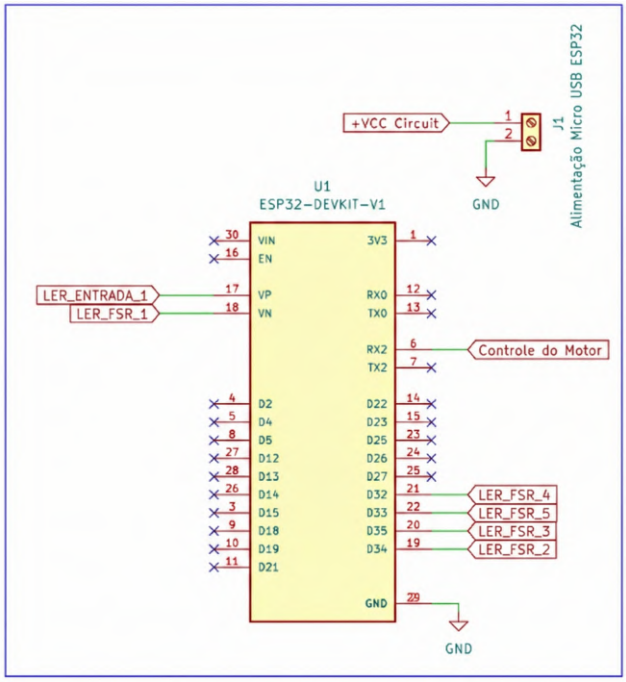
Os capacitores C1 e C3 servem para garantir a operação do LM7805 e são recomendados em seu *datasheet*, para operação como regulador de tensão fixa. Já o diodo (D1) posicionado no modo *reverse-bias* opera para garantir a segurança do LM7805 em casos de queda de tensão na entrada superior a saída.

É importante mencionar que a alimentação ao kit de desenvolvimento da ESP32 não está inclusa na alimentação das fontes, isso devido a alta variação de metodologias de alimentação externas nos kits vendidos atualmente no mercado, é recomendado que para a alimentação seja utilizada uma fonte externa ou conectado diretamente ao USB do *devkit* da ESP32.

4.3 Controle

Visando a seção de controle do circuito é então aplicado o kit de desenvolvimento (*devkit*) baseado na ESP32 para definição de um controle lógico e tratamento dos sinais e dados. Ela opera utilizando tanto os seus Conversores Analógico-Digital (ADC) quanto os seus Conversores Digital-Analógico (DAC), ambos com o intuito de simplificar a operação em um só dispositivo assim como reduzir o custo de entrada para replicação desse projeto.

Figura 27 – Bloco de controle do circuito



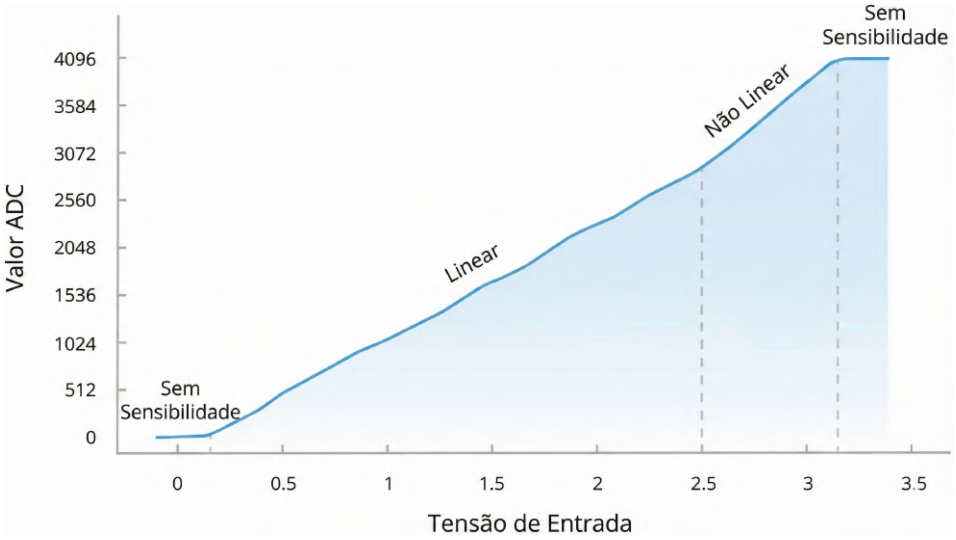
Fonte: Autoria própria

4.3.1 Conversor ADC

Para leitura do sinal é utilizado o conversor Analógico-Digital da placa, porém a sua curva de leitura não é linear ou facilmente aproximável por meio de equações matemáticas de forma que possa realizar a compensação em tempo real de sua leitura incorreta com erro mínimo. A sua curva de leitura varia de acordo com cada dispositivo, sendo única para cada, porém todas tendem a seguir o mesmo padrão.

Possuindo dois ADCs distintos em sua construção, ambos com 12 bits de leitura do sinal, portanto possuindo 4096 posições para resposta da leitura digital, e estando também limitado a leituras de sinais de 0V a 3,3V pode-se então calcular que o degrau (*step*) teórico de leitura do ADCs é de 0,879mV, porém isso seria somente em um cenário ideal, considerando a leitura em todo o range, linearidade para toda leitura e valor de referência do ADC (V_{ref}).

Figura 28 – Identificação da região linear do ADC



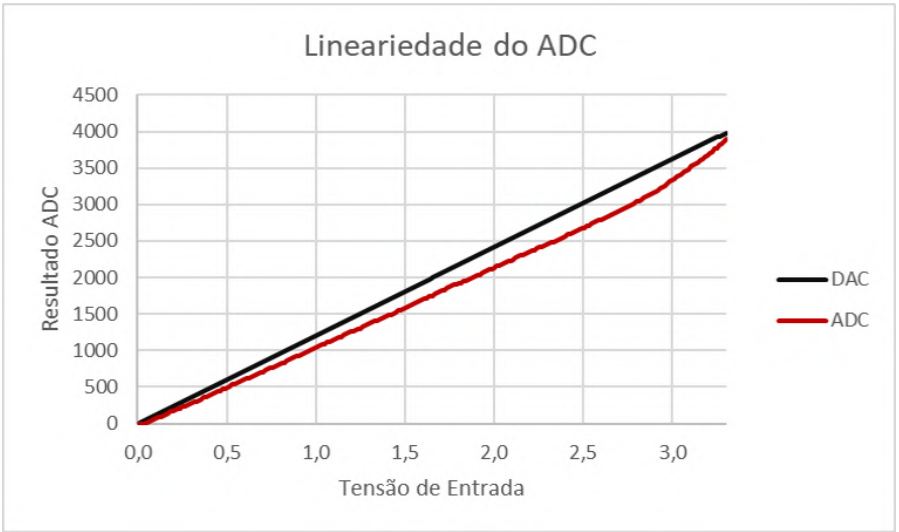
Fonte: Adaptado pelo autor | https://arduinokitproject.com/esp32-basics-adc/

Com base na imagem acima, nota-se então que é recomendado que o ADC esteja posicionado fora da região de insensibilidade, porém também fora da região não-linear. Portanto é importante o posicionamento adequado do sinal a ser medido. Porém há também de se considerar o ruido e erros de leitura associado a variações de construção do ADC.

4.3.2 Aplicação de Look-up Table

Mesmo com a aplicação da medição do sinal em local apropriado na curva do ADC da ESP32 ainda há problemas a serem mitigados, sendo um dos principais problemas o fato de que mesmo a faixa linear do ADC não é apropriadamente linear, podendo portanto haver ainda variações de leitura do sinal.

Figura 29 – Visão da curva não-linear em comparação com uma entrada linear



Fonte: Autoria própria

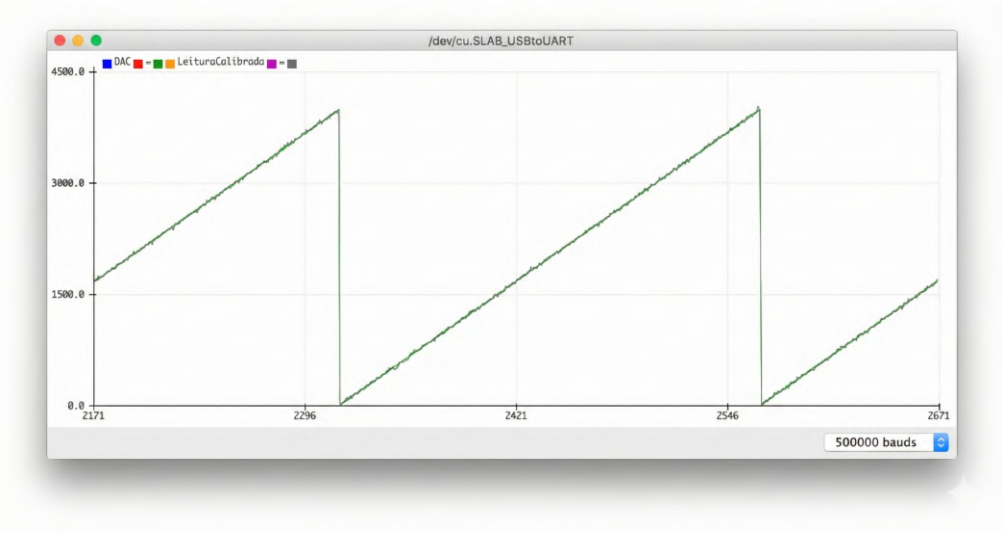
Com o intuito de reduzir o erro de leitura, pode-se então realizar uma correção que é individual para o kit de desenvolvimento do usuário, sendo a aplicação de uma correção via software para a leitura do ADC via calibração do mesmo. Tendo a calibração necessariamente devendo ocorrer antes da inserção do ESP32 ao circuito, como um passo preparatório a sua aplicação.

Para calibração do ADC foram estudados dois métodos, o primeiro é utilizar a função "calculate _voltage _linear()"e "read _efuse _vref()", sendo que a partir da primeira semana de 2018, todos os chips da ESP32 produzidos pela espressif já foram medidos individualmente e possuem o valor de V_{ref} no chip, dessa forma, pode-se então utilizar uma formula polinomial para aproximação. Podendo ambas as formulas serem obtidas a partir da biblioteca "esp _adc _calc.c"disponibilizada pela fabricante em seu github.

O segundo método de cálculo é a partir da calibração do ADC via tabela de consulta, ou em inglês *Look-up Table* (LUT), onde realiza-se uma associação entre os valores medidos e os valores reais, dessa forma compensando de forma mais rápida e precisa com relação aos erros de leitura, porém considerando que haverá uma maior necessidade de armazenamento.

Para a criação de uma LUT é recomendado que, por praticidade, utilize o DAC do próprio kit de desenvolvimento, dessa forma estaria alimentando a saída do DAC, localizada no GPIO 25, com o canal de entrada 1 do ADC, localizado no GPIO 35 (ADC7). Com a conexão via cabo (*jumper*) pode-se então realizar a calibração com o código e obter a tabela LUT.

Figura 30 – Comparativo entre o ADC e DAC com compensação

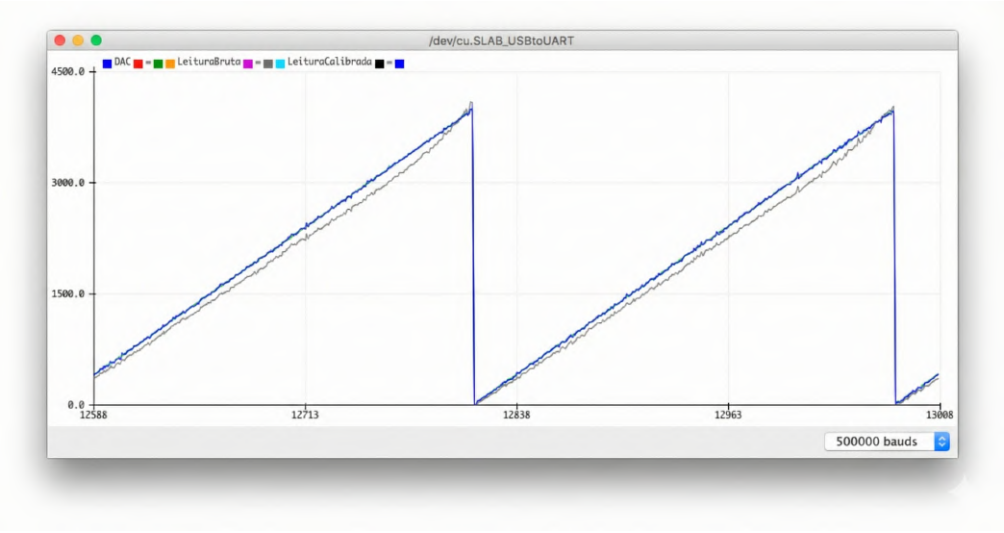


Fonte: Adaptado pelo autor | https://github.com/e-tinkers/esp32-adc-calibrate

Com a calibração pode-se notar que a qualidade de leitura do sinal se torna muito mais elevada, sendo que há uma redução tangível do erro porém sem a elevação de custo,

tendo sido realizado via *software* de forma a também ser aplicável a qualquer dispositivo baseado em ESP32.

Figura 31 – Comparativo entre o ADC e DAC sem compensação



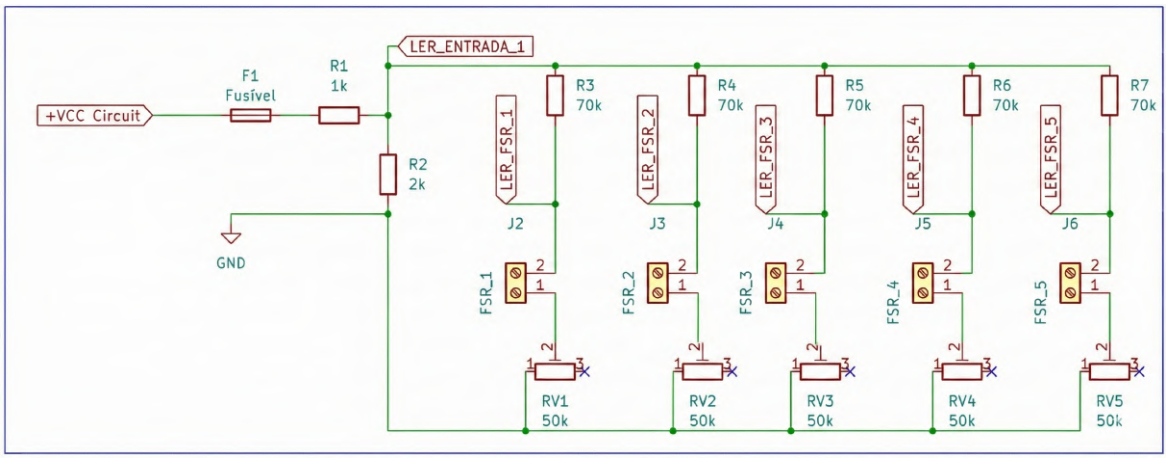
Fonte: Adaptado pelo autor | https://github.com/e-tinkers/esp32-adc-calibrate

Comparativamente pode-se também observar a divergência de leituras sem a aplicação da calibração, demonstrando então a importância da sua aplicação para um projeto de sensoriamento.

4.4 Entrada de Sinal

Com base no que já foi descrito, partindo do fato do circuito ser alimentado com uma tensão contínua regulada, pode-se então abordar a leitura de dados dos sensores, ou seja, entrada de dados no circuito.

Figura 32 – Bloco de leitura dos sensores do circuito



Fonte: Autoria própria

O fusível (F1) serve como dispositivo de proteção a uma das partes mais frágeis do circuito, sendo que o kit de desenvolvimento da ESP32 já possui métodos próprios de proteção intrínsecos a si.

Relembrando dos limites de tensão dos ADCs da ESP32, torna-se necessário a redução da tensão aplicada aos sensores a fim de evitar possíveis cenários de dados permanente. Portanto, os resistores R1 e R2 operam como um divisor de tensão, reduzindo a tensão de entrada de 5V para uma tensão inferior (3,3V), de forma ser trabalhada pelos sensores.

$$V_{R2} = V_{cc}\left(\frac{R_2}{R_1 + R_2}\right)$$

Com base no mesmo efeito de divisor de tensão, as medições dos ADCs são realizadas a partir da aplicação de um resistor de *pull-up* e em seguida com base em um resistor variável a fins de melhor posicionar a leitura do sinal em uma área linear de leitura dos ADCs da ESP32.

$$V_{FSR_1} = V_{R2}\left(\frac{R_3}{R_3 + R_{FSR_1} + R_{V1}}\right)$$

A equação acima é importante para a definição do valor do sensor, visto que o resistor variável (RV1) serve como método para melhor posicionar a medição do FSR na região linear de leitura do ADC, da mesma forma, o resistor R3 serve para realizar o cálculo de divisor de tensão, assim como também para reduzir a tensão aplicada sobre o FSR, dessa forma reduzindo levemente a tensão aplicada sobre ele. Portanto, sabendo o valor do resistor variável (RV1) e do resistor fixo (R3) pode-se então saber qual a resistência do sensor FSR, visto que a sua tensão será medida pelo ESP32 na variável V_{FSR_1}.

$$R_{FSR_1} = \left(\frac{V_{R2}}{V_{FSR_1}}\right).R_3 - (R_3 + R_{V1})$$

A partir da equação acima pode-se notar que a partir da medição infere-se o valor da resistência do sensor, e portanto, pode-se então consultar a relação de força e resistência do FSR com base em uma equação polinomial ou via *lookup-table* definida em código armazenado na ESP32.

Para uma própria determinação do valor do resistor variável é importante identificar qual a faixa de força esperada para medição e portanto planejar de acordo, de forma a posicionar a leitura média da tensão no meio da região linear do ADC. Para os demais divisores de tensão do circuito a aplicação é direta, realizando os mesmos passos já descritos previamente.

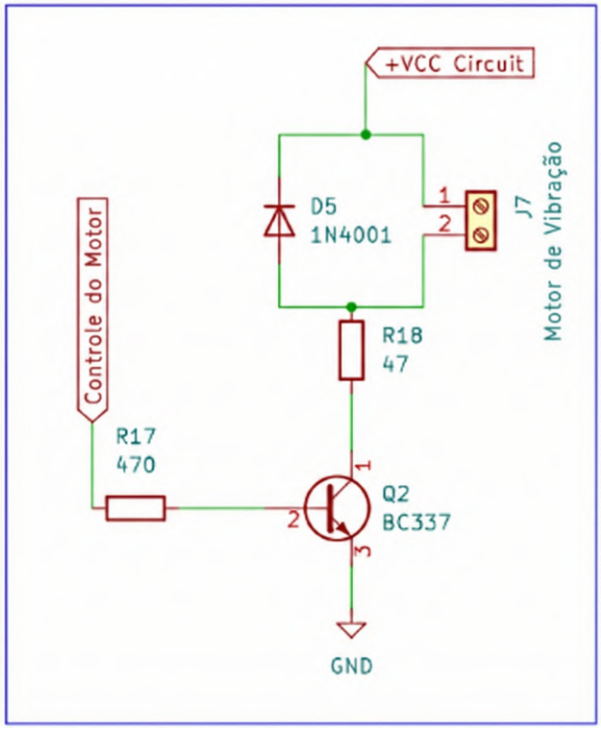
Para inclusão dos sensores FSR ao circuito são utilizados os *terminal blocks* J2 a J6, soldando um fio a cada um dos terminais do FSR e então fixando eles aos conectores com força apropriada.

A leitura via "READ_INPUT _1"serve somente como checagem em caso de necessidade da operacionalidade do circuito, garantindo que, caso necessário, possa ser realizada a leitura da tensão aplicada aos sensores.

4.5 Saídas do Circuito

Para que um sistema eletrônico seja verdadeiramente interativo, ele precisa não apenas receber informações, mas também comunicar suas respostas e estados ao usuário. Essas respostas são manifestadas através de atuadores, que funcionam como as "saídas"do circuito, traduzindo sinais elétricos em ações físicas.

Figura 33 – Bloco de saída do circuito



Fonte: Autoria própria

Para essa aplicação específica foi escolhido a utilização de um motor vibrador, dado ao seu preço reduzido, fácil capacidade de compreensão do sinal sem necessidade de prestar atenção e também pelo fato de que pode ser encontrado em qualquer dispositivo celular, seja antigo ou novo, portanto pode-se obter de uma pilha de itens eletrônicos descartados.

O pequeno motor está sendo alimentado pelo V_{cc}, possuindo o diodo (D5) como diodo de *flyback*, dessa forma protegendo contra o efeito indutivo do motor. O transistor

(Q2) está atuando como chave nesse circuito, ou seja, está variando entre o corte e a saturação de forma a ligar/desligar o motor. Já o controle do sinal de base é realizado pela ESP32.Por fim, os resistores R17 e R18 servem como limitadores de corrente, evitando cenários de excesso de corrente que podem levar a danos em qualquer componentes do circuito, sendo o resistor R18 possui como principal propósito a limitação da corrente requerida pela fonte ($I = V_{cc}/R_{18} \approx 106mA$), já que visando o uso de motores de vibração, como os de celulares, se torna opcional visto que o motor é uma carga resistiva e indutiva e portanto não necessita de um resistor limitador de corrente, pois só consumirá a corrente de que necessita. Por fim, para conexão do motor de vibração é utilizado o *terminal block* J7 para conexão.

4.6 Construção do Circuito

Esta plataforma serve como a fundação do projeto, provendo não apenas o suporte mecânico para todos os componentes, mas também constituindo o sistema de interconexões elétricas através de um conjunto de trilhas de cobre. A elaboração da PCB deste projeto foi um processo deliberado, realizado com o auxílio de ferramentas de software de Automação de Design Eletrônico (EDA), onde o layout foi cuidadosamente planejado para otimizar tanto o desempenho elétrico quanto a usabilidade do dispositivo final.

Um dos aspectos mais críticos no design de uma PCB robusta é o tratamento da malha de terra. Evitando rotear o terra como uma trilha fina e singular, optou-se por uma abordagem mais eficaz: a utilização de uma malha de terra (*ground plane*), visando empregar esta técnica para preencher as áreas vazias da placa com cobre e conectá-las ao potencial de terra do circuito. A partir da implementação de uma malha de terra pode-se oferecer um caminho de baixa impedância para o retorno da corrente de todos os sinais, o que é fundamental para a estabilidade do sistema. Além disso, essa extensa área de cobre atua como uma blindagem, oferecendo proteção contra campos eletromagnéticos (EMF) e interferências (EMI), absorvendo ruídos externos e contendo as emissões do próprio circuito, garantindo a integridade dos sinais mais sensíveis.

Também foi considerada a dimensão das trilhas condutoras durante o design. A largura de cada trilha não foi definida de forma arbitrária, mas sim dimensionada com base nas recomendações do próprio software, de forma a trilha suporte a corrente elétrica destinada a ela sem superaquecer e também a geometria das trilhas foi otimizada para reduzir o efeito antena que pode ocorrer a frequência elevadas.

Por fim, a organização dos componentes na placa foi planejada com o objetivo de facilitar a montagem, o teste e a conexão pelo usuário final. Os conectores de entrada e saída, como a porta USB, o conector Jack e os blocos de terminais para os sensores, foram posicionados nas bordas da placa para garantir um acesso desobstruído. Internamente, os componentes de sub-circuitos específicos, como o regulador de tensão e seus capacitores

de filtro, foram mantidos fisicamente próximos para tornar a manutenção futura mais fácil, assim como os resistores variáveis foram posicionados ao lado um dos outros. Esse layout criterioso resulta em um dispositivo que não é apenas eletricamente funcional, mas também prático e ergonômico em seu uso.

4.7 Software

O desenvolvimento de lógica da programação do circuito ocorre em duas etapas, sendo a primeira relacionada ao desenvolvimento de um *firmware* para o microcontrolador e a segunda sendo o desenvolvimento de um *software* para um computador.

Abordando a primeira etapa, o dispositivo realiza a leitura das portas pré-definidas do seu ADC para identificar qual a queda de tensão, em seguida, a partir de uma lógica da programação baseada na utilização de uma *look-up table* identifica-se o valor mais apropriado da leitura, e por fim, realiza-se o envio da leitura via MQTT ou comunicação serial.

Na segunda etapa, ocorre principalmente o recebimento dos dados enviados, podendo ser processados novamente, porém nesse momento está sendo focado na interface humano-máquina, ou seja, na leitura e interpretação dos dados.

4.7.1 Acquisição da Look-Up Table

Para o desenvolvimento e aquisição da *look-up table* foi utilizada a metodologia criada pelo e-tinker(Cheung; Kalitin; Weber, 2019), um dos maiores sites de projetos *open-source* para dispositivos eletrônicos.

4.7.2 Firmware da ESP32

Como todo software, geralmente inicia-se pela definição das bibliotecas a serem requisitadas pelo código, portanto, aquelas bibliotecas explicadas previamente serão requisitadas nesse trecho.

```
1 // --- BIBLIOTECAS ---
2 #include <ArduinoJson.h>
3 #include <WiFi.h>
4 #include <PubSubClient.h>
5
6 // --- OBJETOS GLOBAIS ---
7 WiFiClient espClient;
8 PubSubClient client(espClient);
9
```

Em conjunto com as biblioteca também são definidos os objetos globais necessários para o uso das bibliotecas, onde o objeto WiFiClient é criado para gerenciar a conexão de rede de baixo nível (TCP/IP). Logo em sequência, este objeto é repassado ao PubSubClient para realização da comunicação MQTT. Isso é necessário pois o MQTT depende de uma camada de transporte de dados confiável para funcionar, e o WiFiClient fornece exatamente essa funcionalidade.

Para armazenamento e manipulação dos dados é necessário a definição de variáveis previamente ao uso, isso sendo uma necessidade da linguagem de programação utilizada (C++), em oposição a outras, como o Python, onde pode-se criar a variável em conjunto com a sua aplicação. Por escolha, nesse início as variáveis são definidas como constantes, visto que são valores fixos que não devem se alterar, servindo principalmente para fins de configuração, e também por aumentar a otimização do código. Começando pela definição da variável da tabela de referência do ADC, tendo sido suprimida nesse texto para prover uma leitura mais organizada.

```
1  // --- DEFINIÇÃO DA LOOKUP TABLE (LUT) ---
2  const float SENSOR_LUT[4096] = {
3      0.0000, 7.3621, 16.8967, 18.7303, 20.6698, 22.9096,
4      24.6291, 26.8393, 28.5342, 30.2974, 31.9839, 33.7291,
5      34.6141, 36.3150, 37.1009, 38.2713,
6      // ... (restante da tabela com 4096 valores) ...
7      4006.3198, 4008.2036, 4010.2319, 4012.2471, 4014.8879,
8      4017.3755, 4019.4678, 4021.5791, 4032.5591
9  };
10
```

Em seguida, são declaradas as variáveis necessárias para as conexões, sendo elas as definições de WiFi, como *Service Set Identifier* (SSID) que é o nome da rede WiFi a ser conectada, assim como a senha dessa mesma rede. Logo abaixo são definidas as informações necessárias para a conexão com um servidor MQTT, nesse caso foi utilizado um *broker* gratuito e de uso difundido, com capacidade para aplicações de MQTT 5.0, 3.1.1, e 3.1.

```
1
2  // --- CONFIGURAÇÕES WiFi ---
3  const char *ssid = "VIVOFIBRA-C451";       // <-- Altere para o nome da
   ↪  sua rede
4  const char *password = "H55x3CbGm8"; // <-- Altere para a senha da sua
   ↪  rede
```

```
5
6  // --- CONFIGURAÇÕES MQTT ---
7  const char *mqtt_broker = "broker.emqx.io";
8  const int mqtt_port = 1883;
9  const char *mqtt_username = "emqx";
10 const char *mqtt_password = "public";
11 const char *MQTT_TOPIC_PUB = "esp32/sensor_data"; // Tópico para
   ↪  PUBLICAR os dados
12
```

Como é um *broker* aberto há a possibilidade dos dados serem acessados por qualquer indivíduo, porém seria necessário conhecimento do tópico associado a aplicação, tornando então a probabilidade reduzida. Embora, caso haja a necessidade de aplicação em cenários mais sigilosos, pode-se utilizar o IBM Watson para aplicação, sendo gratuito até um certo limite de dados e mensagens diárias, permitindo também a definição de senhas. Para esse projeto foi utilizado um *broker* gratuito e aberto que não há a necessidade de definição de senha ou criação de tópicos, sendo hábil para pronto uso. É válido mencionar que não há a necessidade da abertura de porta no roteador.

```
1
2  // --- CONFIGURAÇÕES DOS SENSORES E MOTOR ---
3  const int NUM_SENSORES = 5;
4  int sensorPins[NUM_SENSORES] = {32, 33, 34, 35, 36};
5  const int VALOR_MAXIMO_ADC = 4095;
6  const int MOTOR_PIN = 16;
7  const int PINO_SENSOR_GATILHO = 32;
8  const int PWM_FREQ = 5000;
9  const int PWM_CHANNEL = 0;
10 const int PWM_RESOLUTION = 8;
11 const int SENSOR_DEADZONE = 50; // Leituras do ADC abaixo deste valor
   ↪  serão ignoradas (consideradas 0)
12
13
```

As definições das constantes para sensores e motor são dependentes do circuito e sensores aplicados, portanto irão variar de acordo com o equipamento utilizado. Nesse cenário são definidos as quantidades de sensores inclusos no circuito, assim como quais os pinos do ADC que estão conectados aos sensores e motor. Como um PWM é aplicado ao motor, também é definidas as configurações associadas a sua configuração.

Nesse cenário, a vibração do motor varia de acordo com a força aplicada ao sensor conectado ao pino 32. Uma zona morta foi definida de forma a que define um limiar mínimo para a leitura do sensor. A sua definição foi necessária pois sensores analógicos podem ter pequenas flutuações, geralmente devido a ruído, mesmo quando em repouso. Esta "zona morta"evita que o motor reaja a esses ruídos, garantindo que ele só seja acionado por uma pressão intencional e evitando variação constante do PWM e do motor, aumentando a vida útil.

É sempre recomendado em um código a aplicação de funções auxiliares a fins de melhoria de leitura do código, dessa forma o código fica mais fácil de compreender inicialmente assim como evita repetições, mantendo a lógica principal limpa.

```
1
2  // --- FUNÇÃO PARA CONECTAR AO WIFI ---
3  void setup_wifi() {
4    delay(10);
5    Serial.println();
6    Serial.print("Conectando-se a ");
7    Serial.println(ssid);
8    WiFi.begin(ssid, password);
9    while (WiFi.status() != WL_CONNECTED) {
10     delay(500);
11     Serial.print(".");
12   }
13   Serial.println("");
14   Serial.println("WiFi conectado!");
15   Serial.print("Endereço IP: ");
16   Serial.println(WiFi.localIP());
17 }
18
```

A função auxiliar *setup_wifi()* tem como intuito a definição da conexão a rede WiFi, utilizando a função interna da biblioteca *WiFi.begin()* para realizar a conexão e a função *WiFi.status()* para verificar se a conexão obteve sucesso. Em caso de falha a função continuará se repetindo até obter sucesso, isso é aplicado devido ao fato de ser necessário para o envio de dados via MQTT. A função *WiFi.localIP()* tem como dever somente demonstrar o endereço do dispositivo na rede para casos de diagnóstico de erro.

```
1
2  // --- FUNÇÃO DE CALLBACK PARA MENSAGENS MQTT RECEBIDAS ---
```

```
3  void callback(char *topic, byte *payload, unsigned int length) {
4    Serial.print("Mensagem recebida no tópico: ");
5    Serial.println(topic);
6    Serial.print("Mensagem: ");
7    for (int i = 0; i < length; i++) {
8      Serial.print((char)payload[i]);
9    }
10   Serial.println();
11   Serial.println("-----------------------");
12 }
13
```

A função *callback()* tem como intuito a conexão ao tópico MQTT como *subscriber* e em seguida publicar na porta serial, dessa forma pode-se utilizar como método de diagnóstico.

```
1
2  // --- FUNÇÃO PARA RECONECTAR AO MQTT ---
3  void reconnect_mqtt() {
4    while (!client.connected()) {
5      Serial.print("Tentando conectar ao broker MQTT...");
6      String client_id = "esp32-client-";
7      client_id += String(WiFi.macAddress());
8      if (client.connect(client_id.c_str(), mqtt_username, mqtt_password))
       ↪ {
9        Serial.println("conectado!");
10
11       // Após conexão, increve-se no tópico para receber de volta o que
         ↪  enviamos.
12       client.subscribe(MQTT_TOPIC_PUB);
13       Serial.print("Inscrito no tópico: ");
14       Serial.println(MQTT_TOPIC_PUB);
15
16     } else {
17       Serial.print("falhou, rc=");
18       Serial.print(client.state());
19       Serial.println(" tentando novamente em 2 segundos");
20       delay(2000);
21     }
```

```
  }
}
```

É comum a queda da conexão com o *broker* do MQTT, dessa forma é importante realizar uma verificação contínua antes da tentativa de envio e esperar a conexão ser refeita de forma a continuar com o código. Em caso de falha, é então exposto na porta serial a informação de erro necessária para realizar o diagnóstico.

```
// --- FUNÇÃO DE SETUP PRINCIPAL ---
void setup() {
  Serial.begin(115200);
  setup_wifi();
  client.setServer(mqtt_broker, mqtt_port);
  client.setCallback(callback);
  analogSetAttenuation(ADC_11db);
  ledcSetup(PWM_CHANNEL, PWM_FREQ, PWM_RESOLUTION);
  ledcAttachPin(MOTOR_PIN, PWM_CHANNEL);
}
```

Na função *setup()* são realizadas as inicializações importantes para a continuação do código, porém ela só ocorre uma vez, toda vez que a ESP32 é iniciada. Nessa função é realizada a conexão com o serial e WiFi, chamando cada um a sua função representante. Em seguida se configura o cliente MQTT com *client.setServer()* e registra a função *callback*. Por fim, configura os periféricos de hardware: o ADC com *analogSetAttenuation()* para leituras precisas na faixa completa de 0-4095, e o sistema de PWM com *ledcSetup()* e *ledcAttachPin()* para preparar o controle do motor.

```
// --- FUNÇÃO DE LOOP PRINCIPAL ---
void loop() {
  if (!client.connected()) {
    reconnect_mqtt();
  }
```

```
7   client.loop();
8
9   StaticJsonDocument<512> doc;
10  doc["qtd_sensores_utilizados"] = NUM_SENSORES;
11  JsonObject sensores = doc.createNestedObject("sensores");
12
13  int raw_adc_gatilho = 0;
14
15  for (int i = 0; i < NUM_SENSORES; i++) {
16    int raw_adc = analogRead(sensorPins[i]);
17    float forca_calibrada = 0;
18    if (raw_adc < 4096) {
19        forca_calibrada = SENSOR_LUT[raw_adc];
20    }
21
22    if (sensorPins[i] == PINO_SENSOR_GATILHO) {
23      raw_adc_gatilho = raw_adc;
24    }
25
26    char sensorKey[12];
27    sprintf(sensorKey, "sensor_0%d", i);
28    JsonObject sensorData = sensores.createNestedObject(sensorKey);
29    sensorData["forca"] = forca_calibrada;
30  }
31
32  int pwm_duty_cycle = 0;
33  if (raw_adc_gatilho > SENSOR_DEADZONE) {
34    pwm_duty_cycle = map(raw_adc_gatilho, SENSOR_DEADZONE,
35                                 VALOR_MAXIMO_ADC, 0, 255);
36  }
37  pwm_duty_cycle = constrain(pwm_duty_cycle, 0, 255);
38  ledcWrite(PWM_CHANNEL, pwm_duty_cycle);
39
40
41  char json_buffer[512];
42  serializeJson(doc, json_buffer);
43  client.publish(MQTT_TOPIC_PUB, json_buffer);
44  serializeJson(doc, Serial);
45  Serial.println();
```

```
46
47    delay(200);
48  }
49
```

A função *loop()* implementada no código representa a parte fundamental para operação do dispositivo. A cada iteração, ela executa uma sequência precisa de tarefas que transformam os dados de acordo com a lógica programada. O ciclo começa com uma etapa de auto-verificação e manutenção, onde o dispositivo garante que sua conexão com o *broker* MQTT está ativa, reconectando-se automaticamente se necessário utilizando a função *reconnect_mqtt()* previamente discutida.

Em seguida é definida um espaço na memória para um objeto JSON que será populado. Onda a biblioteca ArduinoJson gerencia essa memória. Logo depois entra-se na fase de aquisição de dados. Metodicamente será lido o valor analógico de cada um dos cinco pinos do ADC, porém esse dado será transformado, executando uma etapa crítica de calibração, usando a leitura bruta como um índice para consultar a tabela de referência (*SENSOR _LUT* para obter um valor de força preciso e significativo. Simultaneamente, também é identificado e armazenado a leitura do sensor específico que controla o motor.

Com os dados em obtidos, o sistema passa para a fase de ação, nela a lógica de acionamento do motor é acionada. Nessa etapa o motor tem a sua velocidade ajustada de forma análoga à pressão aplicada no sensor. Utilizando a função *map()*, é transformada a intensidade da força medida em um nível de potência PWM correspondente, garantindo uma resposta suave e controlada.

Por fim, na fase de comunicação, é consolidado todos os dados calibrados dos sensores em uma estrutura JSON padronizada para envio. Este estado do sistema é então transmitido simultaneamente por dois canais, primeiramente é publicada via MQTT para o *broker*, permitindo o monitoramento remoto, e em seguida enviada pela porta Serial. O *loop()* se encerra com um *delay()* de 200ms, sendo feito para definir uma taxa de amostragem (aproximadamente 5 vezes por segundo) e, crucialmente, para evitar o envio excessivo de dados pela rede, o que poderia sobrecarregar o broker MQTT e causar uma desconexão devido a excesso de dados.

Por fim, abaixo é possível visualizar um exemplo da estrutura do JSON criado para envio via terminal e MQTT.

```
1
2   {
3     "qtd_sensores_utilizados": 5,
4     "sensores": {
```

```
    "sensor_00": {
      "valor_maximo": 1023,
      "valor_minimo": 0,
      "forca": 450
    },
    "sensor_01": {
      "valor_maximo": 1023,
      "valor_minimo": 0,
      "forca": 512
    },
    "sensor_02": {
      "valor_maximo": 1023,
      "valor_minimo": 0,
      "forca": 320
    },
    "sensor_03": {
      "valor_maximo": 1023,
      "valor_minimo": 0,
      "forca": 680
    },
    "sensor_04": {
      "valor_maximo": 1023,
      "valor_minimo": 0,
      "forca": 150
    }
  }
}


```

4.7.3 Software em Python

Com o código do *firmware* realizado e propriamente diagnosticado, pode-se então partir para a execução do *software* responsável por receber os dados.

```

import serial
import numpy as np
import cv2 as cv
import json
```

```
6  import threading
7
8
9  # # Cria graph como uma variavel global; Armazena então o plot nessa
   ↪  variavel
10 # graph=1
11 qtd_graficos = 5
12 graph = list(range(qtd_graficos))
13
```

Iniciando-se pela declaração das bibliotecas, sendo necessárias para a realização e uso de funções para realização de tarefas generalistas. Em seguida são declaradas as importação de funções, sendo necessárias para aplicação do código de maneira mais organizada. Também são definidas variáveis globais para uso em todos os cenários, principalmente para as funções auxiliares.

```
1
2  def square_overleay(imagem, pixel_central_quadrado, lateral, valor_max,
   ↪  valor_min, valor_atual):
3      pixel_inicial_quadrado = [pixel_central_quadrado[0] - lateral,
   ↪  pixel_central_quadrado[1] - lateral]
4      pixel_final_quadrado = [pixel_central_quadrado[0] + lateral,
   ↪  pixel_central_quadrado[1] + lateral]
5
6      valor_atual = max(valor_atual, valor_min)
7      valor_atual = min(valor_atual, valor_max)
8
9      imagem[pixel_inicial_quadrado[0]:pixel_final_quadrado[0],
10     pixel_inicial_quadrado[1]:pixel_final_quadrado[1], 2] = 0 +
   ↪  (valor_atual / valor_max) * 255
11     imagem[pixel_inicial_quadrado[0]:pixel_final_quadrado[0],
12     pixel_inicial_quadrado[1]:pixel_final_quadrado[1], 1] = 255 -
   ↪  (valor_atual / valor_max) * 255
13     imagem[pixel_inicial_quadrado[0]:pixel_final_quadrado[0],
14     pixel_inicial_quadrado[1]:pixel_final_quadrado[1], 0] = 0
15
16     return imagem
17
18
```

```
19 def circle_overleay(imagem, pixel_central_circulo, raio, valor_max,
   ↪ valor_min, valor_atual):
20     pixel_inicial_quadrado = [pixel_central_circulo[1] - raio,
   ↪ pixel_central_circulo[0] - raio]  # X Y
21     pixel_final_quadrado = [pixel_central_circulo[1] + raio,
   ↪ pixel_central_circulo[0] + raio]  # X Y
22
23     # cv.rectangle(imagem, pixel_inicial_quadrado,
   ↪ pixel_final_quadrado, (0, 0, 0), 10)
24
25     valor_atual = max(valor_atual, valor_min)
26     valor_atual = min(valor_atual, valor_max)
27
28     # print("valor x_inicial: {}. valor x_final:
   ↪ {}".format(pixel_inicial_quadrado[1], pixel_final_quadrado[1]))
29     # print(imagem.shape)
30     for m in range(pixel_inicial_quadrado[0], pixel_final_quadrado[0]):
31         for n in range(pixel_inicial_quadrado[1],
   ↪ pixel_final_quadrado[1]):
32             modulo = np.sqrt((abs(m - pixel_central_circulo[1])) ** 2 +
   ↪ (abs(n - pixel_central_circulo[0])) ** 2)
33
34             if (modulo <= raio):
35                 imagem[n, m, 2] = 0 + (valor_atual / valor_max) * 255
36                 imagem[n, m, 1] = 255 - (valor_atual / valor_max) * 255
37                 imagem[n, m, 0] = 0
38
39     # for i in range(1,raio):
40     #         cv.circle(imagem,
   ↪ (pixel_central_circulo[1],pixel_central_circulo[0]), i, (0 +
   ↪ (valor_atual / valor_max) * 255,255 - (valor_atual / valor_max) *
   ↪ 255,0), 10)
41
42     return imagem
43
44
```

As funções de *circle _overlay()* e *square _overlay()* são funções auxiliares que recebem uma imagem, nesse caso sendo uma imagem de uma mão, as coordenadas de um

ponto central (dedos da mão) e um valor de sensor. Com base nesse valor, ela desenha um círculo colorido sobre a imagem, onde a cor do círculo varia de verde (baixa intensidade) até vermelho (alta intensidade), criando um mapa de calor visual da força aplicada similar a um semáforo. Somente a função do círculo é utilizada, representando um FSR circular, porém em cenários onde o sensor possui uma forma quadrada pode ser aplicada a função para desenhar um quadrado.

```
1
2  # https://medium.com/@vinay.dec26/
   ↪  real-time-plotting-tool-in-opencv-1d1263b7fc99
3  # https://github.com/2vin/opencv-plot
4  # Plot values in opencv program
5  class Plotter:
6
7      def __init__(self, plot_width, plot_height):
8          self.width = plot_width
9          self.height = plot_height
10         self.color = (255, 0, 0)
11         self.val = []
12         self.plot_canvas = np.ones((self.height, self.width, 3)) * 255
13
14     # Update new values in plot
15     def plot(self, val, sensor_numero, step_numeros, label="plot"):
16         #
   ↪  self.val=collections.deque(maxlen=len(self.plot_canvas[0])-1)
17         self.val.append(int(val))
18
19         # if (len(self.plot_canvas[0])> len(self.val)+step_numeros):
20         #     self.val.append(int(val))
21         # else:
22         #     shift(self.val, step_numeros)
23         #     self.val[len(self.val)-1]=int(val)
24
25         while len(self.val) > self.width:
26             self.val.pop(0)
27         # print("Dentro do Plot: val = {}".format(self.val))
28         self.show_plot(label, sensor_numero, step_numeros)
29
```

```
30      ############## - UTILIZAR ESSE PLOT PARA NUMEROS SOMENTE POSITIVOS
    ↪  - ####################
31      # Show plot using opencv imshow
32      def show_plot(self, label, sensor_numero, step_numeros):
33          global graph
34
35          self.plot_canvas = np.ones((self.height, self.width, 3)) * 255
36          cv.line(self.plot_canvas, (0, int(self.height - 10)),
    ↪  (self.width, int(self.height - 10)), (0, 255, 0), 1)
37          offset_X_line = 10
38
39          # if (len(self.plot_canvas[0]) <=
    ↪  len(self.val)+step_numeros+20):
40          #
41          #     shift(self.val, step_numeros+20)
42          #     self.val[(len(self.val)- 1 - step_numeros +
    ↪  20):(len(self.val)- 1 )] = 0
43
44          # print("len plot_canvas: {}. len val:
    ↪  {}".format(len(self.plot_canvas[0]), len(self.val)))
45          cv.line(self.plot_canvas, (offset_X_line, 10), (offset_X_line,
    ↪  int(self.height)), (0, 255, 0), 1)
46          for k in range(sensor_numero, len(self.val) - 1, step_numeros):
47              # print("valor k= {}".format(k))
48              cv.line(self.plot_canvas, (k + offset_X_line - (step_numeros
    ↪  - 1), int(self.height - 10) - self.val[k]), (
49                      k + 1 + offset_X_line - (step_numeros - 1) *0,
    ↪  int(self.height - 10) - self.val[k + step_numeros]),
50                      self.color, 1)
51          graph[sensor_numero] = self.plot_canvas
52          # cv.imshow(label, self.plot_canvas)
53          # cv.waitKey(10)
54
55
56
```

É criada a classe "Plotter", sendo ela uma classe customizada, criada para gerar os gráficos de linha em tempo real. O método *plot()* recebe um novo valor, o adiciona a uma lista histórica e chama o *show _plot()*, portanto ele efetivamente desenha o gráfico (eixos

e linhas) sobre um "*canvas*"(uma imagem em branco criada com NumPy) e armazena o resultado em uma variável global para uso futuro.

```

# Carregando imagem a ser demonstrada
img_hand = cv.imread("hand_resized.png", cv.IMREAD_UNCHANGED)
# converte de BGR para RGB
img_hand_RGB = cv.cvtColor(img_hand, 4)

# Parâmetros para os sensores


pixel_central = [[143, 51], [73, 117], [50, 162], [89, 212], [250, 280]]  # Y X

N = [15, 15, 15, 15, 17]  # Define o tamanho do quadrado

# Tamanhos pro resize da imagem img_hand_RGB


resize_X = 320
resize_Y = 640

min_pixel = 0 / qtd_graficos
max_pixel = resize_Y / qtd_graficos

dim = (resize_X, resize_Y)

# Create a plotter class object
p = Plotter(int(resize_X), int(max_pixel))  # X Y

# make sure the 'COM#' is set according the Windows Device Manager
ser = serial.Serial('COM3', 115200, timeout=1)
ser.reset_input_buffer()

qtd_points = 150
for i in range(qtd_points):
    line = ser.readline().decode("utf-8")  # read a byte string

```

```
36      try:
37          dict_json = json.loads(line)
38
39          qtd_graficos = dict_json.get("qtd_sensores_utilizados")  #
    ↪ Equivalente a quantidade de sensores
40          valor_maximo =
    ↪ list(range(dict_json.get("qtd_sensores_utilizados")))
41          valor_minimo =
    ↪ list(range(dict_json.get("qtd_sensores_utilizados")))
42          valor = list(range(dict_json.get("qtd_sensores_utilizados")))
43          threads_grafico =
    ↪ list(range(dict_json.get("qtd_sensores_utilizados")))
44          # value =
    ↪ list(range(dict_json.get("qtd_sensores_utilizados")))
45
46          for i in range(qtd_graficos):
47              string_sensores_key = "sensor_00"
48              string_list = list(string_sensores_key)
49              string_list[len(string_sensores_key) - 1] = str(i)
50              string_sensores = ''.join(string_list)
51
52              valor_maximo[i] =
    ↪ (dict_json.get("sensores").get(string_sensores).get("valor_maximo"))
53              valor_minimo[i] =
    ↪ (dict_json.get("sensores").get(string_sensores).get("valor_minimo"))
54              valor[i] =
    ↪ dict_json.get("sensores").get(string_sensores).get("forca")
55
56              # print('iteração: {}. valor_maximo: {}. valor: {}. sensor:
    ↪ {}'.format(i,valor_maximo[i],valor[i],string_sensores))
57
58              # for i in range(qtd_graficos):
59              value = ((valor[i] - valor_minimo[i]) / (valor_maximo[i] -
    ↪ valor_minimo[i])) * max_pixel
60
61              # print('iteração: {}. VALOR: {}. value: {}. sensor:
    ↪ {}'.format(i, valor[i],  value, string_sensores))
62
63              # threads_grafico[i]= threading.Thread(target=Plotter)
```

```
64              threads_grafico[i] = threading.Thread(p.plot(value, i,
    ↪ qtd_graficos))
65              threads_grafico[i].start()
66              img_hand_RGB = circle_overleay(img_hand_RGB,
    ↪ pixel_central[i], N[i], valor_maximo[i], valor_minimo[i],
67                                             valor[i])
68
69          #   Multithreading:  https://www.geeksforgeeks.org/
    ↪ multithreading-python-set-1/
70          for j in range(qtd_graficos):
71              threads_grafico[j].join()
72              if j < 1:
73                  graph_final = graph[j]
74              else:
75                  graph_final = np.vstack((graph_final, graph[j]))
76
77          # graph_final = np.array([graph[0], graph[1], graph[2],
    ↪ graph[3], graph[4]]) # https://stackoverflow.com/
    ↪ questions/44517809/ concatenate-multiple-numpy-arrays-in-one-array
78          # print(graph_final.shape)
79          # print(img_hand_RGB.shape)
80          both = np.hstack((img_hand_RGB, graph_final))
81
82          cv.namedWindow("canvas")
83          cv.imshow("canvas", both)
84          cv.waitKey(10)
85          # cv.destroyAllWindows()
86
87          # winsound.Beep(440+round(3*int(valor[0])), 200)
88
89
90      except json.JSONDecodeError as e:
91          print("JSON:", e)
92
93  ser.close()
94
```

O código começa com o carregamento da imagem, sendo ela denominada "*hand_resized.png*"e carregada na memória, tendo como função servir de fundo para a visualização

dos círculos coloridos. Em seguida são definidos os parâmetros para coordenada dos círculos e para a conexão serial, sendo conectado na porta COM, e é importante que o *baudrate* seja igual para ambos o *firmware* e *software*.

Em seguida parte-se para a parte de construção lógica do programa, onde ele lê continuamente os dados, processa as informações e atualiza a tela. O código pode ser segmentado em quatro etapas:

1. **Leitura e Decodificação:** A função *ser.readline()* lê uma linha de texto da porta serial, sendo que espera-se que essa linha seja uma *string* no formato JSON. O bloco *try/except* garante que, caso a *string* estiver malformada, o programa não irá travar, apenas imprimirá uma mensagem erro utilizada em diagnóstico. A função *json.loads()* converte então a *string* em um dicionário Python.

2. **Extração e Visualização:** Um *loop* utilizando um *for()* percorre os dados de cada sensor extraídos do dicionário, onde para cada sensor, irá chamar a função *circle_overleay()* para desenhar ou atualizar o círculo colorido na imagem da mão. Em sequência chama o método *p.plot()* para adicionar o novo ponto de dados ao gráfico correspondente e redesenhá-lo.

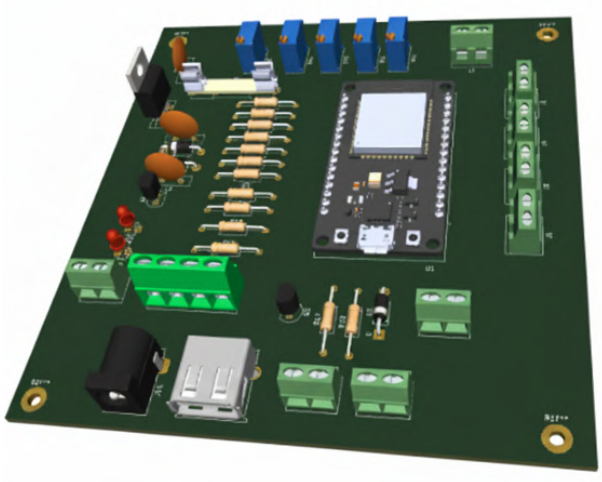
3. **Composição da Tela:** Após processar todos os sensores, o *script* utiliza as funções *np.vstack()* e *np.hstack()* do NumPy para juntar todas as imagens geradas, a imagem de fundo da mão e os 5 gráficos, em uma única imagem final.

4. **Exibição:** O comando *cv.imshow()* exibe essa imagem composta em uma janela na tela. A função *cv.waitKey(10)* faz uma pequena pausa, essencial para que o OpenCV consiga processar os eventos da janela e redesenhá-la.

5 RESULTADOS OBTIDOS

Conforme proposto no decorrer desse documento, a criação de um dispositivo de reabilitação de baixo custo, apoiado tanto por *hardware* e quanto suplementado via *software*, foi o foco do desenvolvimento. Culminando em um circuito com preço aproximado de mercado de R$ 70,00, representando 4,6% do salário mínimo brasileiro mensal (c. 2025), tendo seu custo majoritariamente composto pelo dispositivo ESP32 e sensores FSR, podendo o seu custo ser ainda reduzido caso o leitor possua um microcontrolador similar a sua disposição.

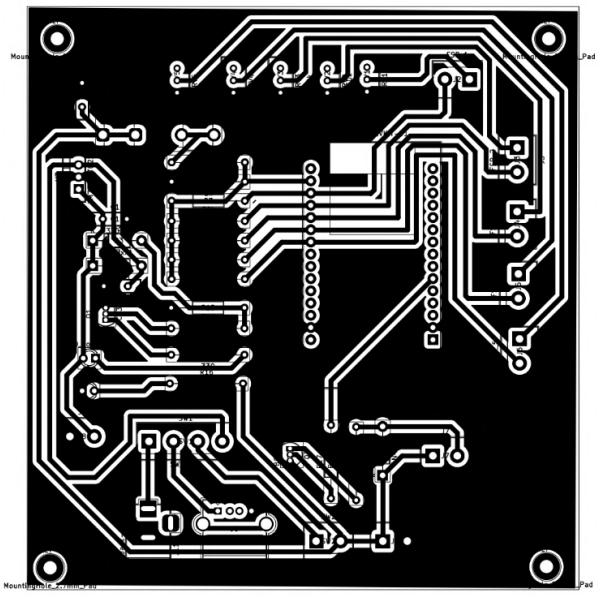
Figura 34 – Visão tridimensional da placa do circuito



Fonte: Autoria própria

A placa do circuito impresso, também denominado de PCB (*Printed Circuit Board*), foi construído de forma a minimizar o espaço assim como tornar a usabilidade e o diagnóstico o mais simples possível. Contanto também com *mounting holes*, ou seja, espaço para parafusos com configuração M3 diretamente na malha de terra, permiti-se então a montagem a qualquer superfície e integração ao terra já existente.

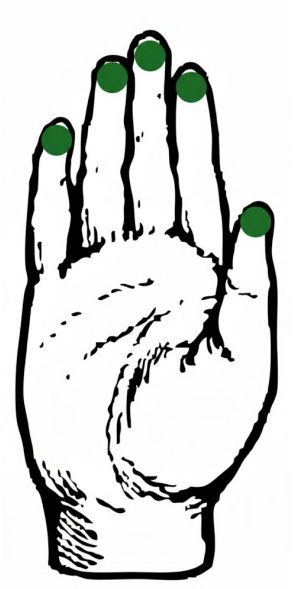
Figura 35 – Visão do PCB do circuito



Fonte: Autoria própria

Associando o circuito a sua aplicação foi desenvolvido também códigos para aplicação tanto no microcontrolador assim como em dispositivos de controles externos, como computadores ou SBCs (*Single Board Computers*) como Raspberrys. Dessa forma pode-se implementar o controle do dispositivo de formas distintas, aumentando a diversidade de aplicações.

Figura 36 – Visão da imagem de saída do *software*



Fonte: Autoria própria

Com base nas otimizações e integrações foi desenvolvido um dispositivo que obteve sucesso na criação de um sistema de alta performance para a mensuração em tempo real. Um avanço fundamental foi a notável melhoria do sinal obtida através da aplicação de uma *Lookup Table* (LUT), que converteu as leituras analógicas brutas e não-lineares do ADC em dados calibrados e com maior precisão, eliminando ruídos e imprecisões. A capacidade de envio de dados em dois canais permite uma operação diversificada, com capacidade para envio via cabo e a distância. É de se destacar que toda esta funcionalidade foi alcançada mantendo um custo total extremamente baixo, viabilizado pela utilização de componentes eletrônicos básicos e da plataforma de baixo custo ESP32.

Para além da performance imediata, a arquitetura do sistema demonstra escalabilidade e potencial de generalização, visto que a aplicação de protocolos de comunicação como MQTT permite que o sistema seja facilmente expandido. Adicionalmente, a generalização do circuito e do código-base foi projetado com uma estrutura de forma modular, permitindo que os sensores de força (FSRs) sejam substituídos por qualquer outro tipo de sensor analógico baseados na variação da resistência, como sensores de temperatura, luz ou umidade, com alterações mínimas. Da mesma forma, o controle do atuador pode ser adaptado para outros dispositivos, como servomotores ou relés, tornando este projeto não apenas uma solução para um problema específico, mas uma plataforma flexível e adaptável para uma vasta gama de futuras aplicações em automação e Internet das Coisas.

6 CONCLUSÃO

A conclusão deste trabalho culminou no desenvolvimento de uma solução prática, escalável e de simples execução, alcançando com sucesso os objetivos propostos. Através da integração de componentes de baixo custo e softwares de código aberto, foi possível construir um sistema robusto para a aquisição, transmissão e visualização de dados de sensores em tempo real. Possuindo uma arquitetura que emprega uma comunicação dupla, tanto Serial quanto MQTT, e uma calibração precisa via *Lookup Table*, demonstrou ser não apenas funcional, mas também flexível e pronta para expansão, validando a viabilidade da abordagem escolhida.

O resultado final materializa-se em um protótipo funcional de um dispositivo para controle e sensoriamento, onde embora sejam empregados componentes simples eles são impulsionados por lógicas de *software* para aumentar a sua eficiência, integrando de forma coesa múltiplos conceitos da engenharia biomédica. O projeto abrangeu todo o espectro do desenvolvimento, partindo da seleção e montagem do hardware (microcontrolador, sensores e circuito de acionamento), passando pelo desenvolvimento do *firmware* embarcado para processamento e controle, até a criação do *software* de visualização que traduz dados brutos em informação visualmente intuitiva. Essa jornada completa, do físico ao digital, exemplifica a interdisciplinaridade inerente à área da engenharia eletrônica.

Torna-se também fundamental ressaltar que a prototipagem foi concebida não como um fim em si mesma, mas como uma plataforma robusta para validação e iteração futura. O sistema foi desenvolvido com uma clara orientação para a aplicação prática, visando uma futura avaliação de desempenho em ensaios clínicos e a utilização em cenários reais, seguindo sempre as normativas associadas a segurança do indivíduo. A precisão obtida com a calibração, aliada à capacidade de monitoramento remoto, estabelece uma base sólida para que este protótipo evolua para um dispositivo clinicamente e didaticamente relevante, capaz de coletar dados para diagnóstico, seja em reabilitação ou pesquisa.

Este projeto, portanto, serve como um evidência da missão crucial que a engenharia biomédica desempenha na criação de soluções inovadoras visando o auxílio geral à sociedade. Ao conectar conhecimentos de diversas áreas, como eletrônica, programação e fisiologia, é possível desenvolver tecnologias acessíveis e eficazes que respondem a necessidades humanas concretas.

REFERÊNCIAS

BRONZINO, J. 1 - biomedical engineering: A historical perspective. *In*: ENDERLE, J. D.; BLANCHARD, S. M.; BRONZINO, J. D. (ed.). **Introduction to Biomedical Engineering**. Second. Boston: Academic Press, 2005, (Biomedical Engineering). p. 1–29. ISBN 978-0-12-238662-6. Disponível em: https://www.sciencedirect.com/science/article/pii/B9780122386626500033.

CHEUNG, H.; KALITIN, C.; WEBER, H. **ESP32 ADC Calibrate**: Source code for esp32 adc calibration. GitHub, 2019. Repositório de código-fonte. Disponível em: https://github.com/e-tinkers/esp32-adc-calibrate/blob/master/src/main.cpp. Acesso em: 30 nov. 2025.

LUNDBORG, G.; ROSéN, B. The sensor glove in preoperative conditioning and postoperative rehabilitation. *In*: **Hand Transplantation**. Milano: Springer, 2007.

LUNDBORG, G.; ROSéN, B.; LINDBERG, S. Hearing as substitution for sensation: A new principle for artificial sensibility. **The Journal of Hand Surgery**, SBC, p. 219–2024, 1999.

MDHC, M. dos Direitos Humanos e da C. **Plano Novo Viver sem Limite**. 2023. Disponível em: https://www.gov.br/mdh/pt-br/navegue-por-temas/pessoa-com-deficiencia/acoes-e-programas/plano-novo-viver-sem-limite. Acesso em: 30 nov. 2025.

MENDES, R. M. **Desenvolvimeno e aplicação de um modelo de luva sensorial**. 2010. Dissertação (Dissertação de Mestrado) — Universidade de São Paulo - Faculdade de Medicina de São Paulo, 2010.

RESWICK, J. Rehabilitation engineering. **Annual review of rehabilitation**, v. 1, p. 55–79, 1980.

ANEXOS

ANEXO A – CÓDIGO PARA OBTENÇÃO DA LOOK-UP TABLE

```
1
2  #include <Arduino.h>
3  #include <driver/dac.h>
4
5  // https://github.com/e-tinkers/esp32-adc-calibrate/
   ↪ blob/master/src/main.cpp
6  // Based on original work from Helmut Weber
   ↪ (https://github.com/MacLeod-D/ESP32-ADC)
7  // that he described at
   ↪ https://esp32.com/viewtopic.php?f=19&t=2881&start=30#p47663
8  // Modified with bug-fixed by Henry Cheung
9  //
10 // Build a ESP32 ADC Lookup table to correct ESP32 ADC linearity issue
11 // Run this sketch to build your own LUT for each of your ESP32, copy
   ↪ and paste the
12 // generated LUT to your sketch for using it, see example sketch on how
   ↪ to use it
13 //
14 // Version 2.0 - switch to use analogRead() instead of esp-idf function
   ↪ adcStart()
15 // Version 1.0 - original adoptation and bug fix based on Helmut Weber
   ↪ code
16
17 // #define GRAPH      // uncomment this for print on Serial Plotter
18 #define FLOAT_LUT     // uncomment this if you need float LUT
19 #define ADC_PIN 35    // GPIO 35 = A7, uses any valid Ax pin as you
   ↪ wish
20
21 float Results[4097];
22 float Res2[4096*5];
23
24 void dumpResults() {
25   for (int i=0; i<4096; i++) {
26     if (i % 16 == 0) {
27       Serial.println();
28       Serial.print(i); Serial.print(" - ");
```

```
29      }
30      Serial.print(Results[i], 2); Serial.print(", ");
31    }
32    Serial.println();
33  }
34
35  void dumpRes2() {
36    Serial.println(F("Dump Res2 data..."));
37    for (int i=0; i<(5*4096); i++) {
38        if (i % 16 == 0) {
39          Serial.println(); Serial.print(i); Serial.print(" - ");
40        }
41        Serial.print(Res2[i],3); Serial.print(", ");
42      }
43      Serial.println();
44  }
45
46  void setup() {
47      dac_output_enable(DAC_CHANNEL_1);     // pin 25
48      dac_output_voltage(DAC_CHANNEL_1, 0);
49      analogReadResolution(12);
50      Serial.begin(500000);
51      delay(1000);
52  }
53
54  void loop() {
55
56      Serial.print(F("Test Linearity "));
57      for (int j=0; j<500; j++) {
58        if (j % 100 == 0) Serial.print(".");
59        for (int i=0;i<256;i++) {
60            dac_output_voltage(DAC_CHANNEL_1, (i & 0xff));
61            delayMicroseconds(100);
62            Results[i*16]=0.9*Results[i*16] + 0.1*analogRead(ADC_PIN);
63        }
64      }
65      Serial.println();
66      // dumpResults();
67
```

```
68      Serial.println(F("Calculate interpolated values .."));
69      Results[4096] = 4095.0;
70      for (int i=0; i<256; i++) {
71         for (int j=1; j<16; j++) {
72            Results[i*16+j] = Results[i*16] + (Results[(i+1)*16] -
              ↪  Results[(i)*16])*(float)j / (float)16.0;
73         }
74      }
75      // dumpResults();
76
77      Serial.println(F("Generating LUT .."));
78      for (int i=0; i<4096; i++) {
79          Results[i]=0.5 + Results[i];
80      }
81      // dumpResults();
82
83      Results[4096]=4095.5000;
84      for (int i=0; i<4096; i++) {
85         for (int j=0; j<5; j++) {
86            Res2[i*5+j] = Results[i] + (Results[(i+1)] - Results[i]) *
              ↪  (float)j / (float)10.0;
87         }
88      }
89      // dumpRes2();
90
91      for (int i=1; i<4096; i++) {
92          int index;
93          float minDiff=99999.0;
94          for (int j=0; j<(5*4096); j++) {
95              float diff=fabs((float)(i) - Res2[j]);
96              if(diff<minDiff) {
97                  minDiff=diff;
98                  index=j;
99              }
100         }
101         Results[i]=(float)index;
102     }
103     // dumpResults();
104
```

```
105     for (int i=0; i<(4096); i++) {
106         Results[i]/=5;
107     }
108 
109 #ifdef GRAPH
110 
111     while(1) {
112       for (int i=2; i<256; i++) {
113         dac_output_voltage(DAC_CHANNEL_1, (i & 0xff));
114         delayMicroseconds(100);
115         float r = Results[analogRead(ADC_PIN)];
116         Serial.print(i*16); Serial.print(" "); Serial.println(r);
117       }
118     }
119 
120 #else
121 
122     Serial.println();
123 
124 #ifdef FLOAT_LUT
125     Serial.println("const float ADC_LUT[4096] = { 0,");
126     for (int i=1; i<4095; i++) {
127        Serial.print(Results[i],4); Serial.print(",");
128        if ((i%15)==0) Serial.println();
129     }
130     Serial.println(Results[4095]);
131     Serial.println("};");
132 #else
133     Serial.println("const int ADC_LUT[4096] = { 0,");
134     for (int i=1; i<4095; i++) {
135       Serial.print((int)Results[i]); Serial.print(",");
136       if ((i%15)==0) Serial.println();
137     }
138     Serial.println((int)Results[4095]);
139     Serial.println("};");
140 #endif
141     while(1);
142 #endif
143 
```

```
144 }
145
```