UNIVERSIDADE FEDERAL DE PERNAMBUCO

Universidade Federal de Pernambuco
Centro de Tecnologia e Geociências
Departamento de Eletrônica e Sistemas

Graduação em Engenharia Eletrônica

Danillo Rodrigues Aguiar

Projeto de Sistema Eletrônico de Aquisição de Dados para um Veículo do Tipo Baja SAE

Recife

2023

Danillo Rodrigues Aguiar

Projeto de Sistema Eletrônico de Aquisição de Dados para um Veículo do Tipo Baja SAE

Trabalho de Conclusão apresentado ao Curso de Graduação em Engenharia Eletrônica, do Departamento de Eletrônica e Sistemas, da Universidade Federal de Pernambuco, como requisito parcial para obtenção do grau de Bacharel em Engenharia Eletrônica.

Orientador(a): Prof. Hermano Andrade Cabral, Ph.D.

Recife
2023

Ficha de identificação da obra elaborada pelo autor,
através do programa de geração automática do SIB/UFPE

Aguiar, Danillo Rodrigues .
Projeto de Sistema Eletrônico de Aquisição de Dados para um Veículo do Tipo Baja SAE / Danillo Rodrigues Aguiar. - Recife, 2023.
93 : il.

Orientador(a): Hermano Andrade Cabral
Trabalho de Conclusão de Curso (Graduação) - Universidade Federal de Pernambuco, Centro de Tecnologia e Geociências, Engenharia Eletrônica - Bacharelado, 2023.
Inclui referências, apêndices.

1. Engenharia Automotiva. 2. Aquisição de dados. 3. Desenvolvimento projetos. 4. Modelagem. 5. Baja SAE. I. Cabral, Hermano Andrade. (Orientação). II. Título.

620 CDD (22.ed.)

Danillo Rodrigues Aguiar

Projeto de Sistema Eletrônico de Aquisição de Dados para um Veículo do Tipo Baja SAE

Trabalho de Conclusão apresentado ao Curso de Graduação em Engenharia Eletrônica, do Departamento de Eletrônica e Sistemas, da Universidade Federal de Pernambuco, como requisito parcial para obtenção do grau de Bacharel em Engenharia Eletrônica.

Aprovado em: 22/05/2023

Banca Examinadora

Documento assinado digitalmente
gov.br
HERMANO ANDRADE CABRAL
Data: 25/05/2023 16:36:55-0300
Verifique em https://validar.iti.gov.br

Prof. Hermano Andrade Cabral, Ph.D.
Universidade Federal de Pernambuco

Documento assinado digitalmente
gov.br
GUILHERME NUNES MELO
Data: 25/05/2023 17:19:42-0300
Verifique em https://validar.iti.gov.br

Prof. Guilherme Nunes Melo, D.Sc.
Universidade Federal de Pernambuco

Agradecimentos

Eu gostaria de agradecer a todos vocês que me ajudaram

durante esta jornada, especialmente para:

Meus pais,

Irmão, familiares e companheira.

Além dos amigos de graduação, professores

A Equipe Mangue Baja e a UFPE.

Resumo do Trabalho de Conclusão de Curso apresentado ao Departamento de Eletrônica e Sistemas, como parte dos requisitos necessários para a obtenção do grau de Bacharel em Engenharia Eletrônica(Eng.)

Projeto de Sistema Eletrônico de Aquisição de Dados para um Veículo do Tipo Baja SAE

Danillo Rodrigues Aguiar

Este trabalho tem como objetivo principal desenvolver, prototipar e utilizar um *datalogger* para aquisição de dados em testes reais com veículos do tipo Baja SAE e utilizar esses dados adquiridos para a modelagem do veículo, visando o aprimoramento e consequente aumento da *performance*. Para atingir esse objetivo, foi realizado um planejamento detalhado para identificar as grandezas determinantes do desempenho do veículo, levando em consideração fatores como aceleração, velocidade, torque e outras variáveis relevantes. Em seguida, foi realizado o projeto do *hardware* e do *software*, incluindo a fase de prototipação e validação desses componentes. Posteriormente, foram realizados os testes e aquisição dos dados previamente planejados, resultando em uma melhoria de 15% na dinâmica longitudinal do veículo ao longo de um ano. Esses resultados destacam a importância do *datalogger* como uma ferramenta indispensável para otimizar o desempenho dos veículos Baja SAE.

Palavras-chave: *datalogger*; aquisição de dados; modelagem; baja, projeto.

Abstract of Course Conclusion Work, presented to Departament of Eletronic and Systems, as a partial fulfillment of the requirements for the degree of Bachelor of Electronic Engineering(Eng.)

Electronic Data Acquisition System Project for a Baja SAE Type Vehicle

Danillo Rodrigues Aguiar

The main purpose of this work is to develop, prototype and use a datalogger for data acquisition in real tests with Baja SAE type vehicles and use these acquired data for vehicle modeling, aiming at improving and consequently increasing performance. For this purpose, a detailed plan was carried out to identify the determinants of vehicle performance, taking into account factors such as control, speed, torque and other relevant variables. Then, the hardware and software design was carried out, including a prototyping and validation phase of these components. subsequently, the previously planned tests and data acquisition were carried out, followed by a 15% improvement in the longitudinal dynamics of the vehicle over a year. These results highlight the importance of the datalogger as an indispensable tool to optimize the performance of Baja SAE

Keywords: datalogger; data acquisition; modeling; baja; design.

Sumário

Lista de Ilustrações

Lista de Abreviações

A2D	Analog to digital
ABS	Anti-lock Braking system
ARM	Advanced RISC Machine
BOM	Bill of Material
CI	Circuito Integrado
CAN	Controller Area Network
CSV	Valores Separados por Vírgulas
CVT	Continuously Variable Transmission
DSP	Digital Signal Process
FPGA	field-programmable gate array
GPU	Graphics Processing Unit
ISR	Interrupt Service Routine
LED	Light-Emiting Diode
MCU	Micro Controller Unit
MEMS	Microelectromechanical systems
MOSFET	Metal Oxide Semiconductor Field Effect Transistor
PCB	Placa de Circuito Impresso
PTH	Pin Through Hole
PWM	Pulse Width Modulation
HW	Hardware
RPM	Rotação por Minuto
SAE	Society of Automotive Engineers
SD	Secure Digital
SPI	Serial Peripheral Interface
SW	Software
UFPE	Universidade Federal de Pernambuco

Capítulo 1

Introdução

Como qualquer automóvel, o Baja SAE é um veículo motorizado que utiliza vários mecanismos para poder se locomover. A título de exemplo, podemos citar o sistema de transmissão, responsável por conduzir a potência gerada pelo motor para as rodas. A forma que pode ser entregue essa potência são diversas, e a escolha dos parâmetros de configuração do sistema mudam sensivelmente o resultado de performance de todo o veículo. A escolha destes podem ser feita de modo empírico, testando diversos valores e observando a mudança, ou utilizando modelos computacionais para o cálculo de cada parâmetro. Os dois métodos costumam ser utilizados em conjunto, balanceando o uso para atingir o melhor custo benefício entre o erro computacional e o tempo/custo do processo. Contudo, o *feedback* detalhado do comportamento do sistema é muito importante para determinar o rumo das alterações, dessa forma, o uso de equipamentos para aquisição de dados reais para serem empregados nos modelos matemáticos se traduz em um uma configuração mais perto da *performance* máxima esperada.

1.1 Delimitação do problema

Uma das maiores dores para equipes estudantis de competição é a falta de dados obtidos em campo nos métodos computacionais de modelagem matemáticas nas etapas de projeto, gerando um erro acumulado que custa perda de pontos importantes.

Nesse sentido, um equipamento eletrônico para aquisitar, salvar e processar os dados reais, permitindo a comparação com as curvas teóricas esperadas é imprescindível para proporcionar bons resultados na competição.

1.2 Justificativa

O viés automobilístico em que o veículo está imerso faz com que a necessidade de escolhas dos sistemas, arquiteturas e configurações sejam o mais perto possível do ótimo. Para cada sistema existem parâmetros a serem observados para identificar a região de trabalho em que o conjunto está operando. Essas análises só são possíveis com a medição quantitativa dos dados em condições reais de uso, para que posteriormente possam ser elaboradas as curvas características e comparar os valores encontrados com os teóricos esperados.

Dessa forma, na busca de um veículo competitivo, se faz necessário a elaboração de um projeto eletrônico para a aquisição de dados, possibilitando a construção de um veículo de alto rendimento.

1.3 Objetivos

Nesta seção será abordados os objetivos do presente trabalho, sendo dividido em objetivo geral do projeto, em que é relatado a ideia central, enquanto os resultados específicos esperados estão listados na seção Objetivos Específicos detalhando os resultados individuais a serem alcançados.

1.3.1 Objetivo Geral

O principal objetivo deste trabalho é o desenvolvimento, prototipagem e uso de um *data logger* para aquisição de dados em testes reais com veículos do tipo Baja SAE, com a finalidade de fornecer relatórios e dados para análises e modelagens, sobretudo visando o aprimoramento de desempenho dos veículos da equipe Mangue Baja.

Para isso, é necessário definir as informações pertinentes à observação, o método de aquisição, o plano de teste e a avaliação dos resultados obtidos. Assim, será possível direcionar projetos utilizando dados reais do protótipo e não só dados encontrados em referências bibliográficas.

1.3.2 Objetivos específicos

- Investigação do problema:
    - Entendimento da arquitetura de funcionamento do veículo;
    - Selecionar os parâmetros observados nos subsistemas determinantes para o desempenho.
- Desenvolvimento do projeto:
    - Definir plataforma de desenvolvimento de HW e SW;
    - Definir os métodos de aquisição dos dados e requisitos do sistema;
    - Desenvolver o esquema elétrico e a PCB.
- Prototipação e testes:
    - Prototipar o projeto;
    - Definir e realizar de teste para verificação de funcionamento.
- Avaliação de resultados:
    - Tratar dados e avaliar das curvas características obtidas.

Capítulo 2

Fundamentação Teórica

Neste capítulo, será discutido um resumo da teoria usada para desenvolver o trabalho. Nesse sentido, são introduzidos conceitos de física, engenharia automotiva, eletrônica, desenvolvimento de *hardware* e *software* para facilitar o entendimento dos modelos desenvolvidos.

2.1 Engenharia automobilística

Antes de tudo, entender a indústria automotiva será importante para se beneficiar das boas práticas desenvolvidas durante os últimos cem anos. Com isso, A indústria automotiva é definida como todas as empresas e atividades envolvidas na fabricação de veículos automotores, incluindo a maioria dos componentes. Colocando em perspectiva histórica a indústria automobilística é recente em comparação com a de muitas outras indústrias, entretanto tem um interesse excepcional por causa de seus efeitos na história a partir do século XX (RAE, 2023).

2.1.1 Indústria automotiva moderna

A indústria automobilística moderna é enorme. Nos Estados Unidos, é o maior ramo de manufatura individual em termos de valor total do produto, é responsável por 4,5mi de empregos nos EUA e 4mi na EU, correspondendo cerca de 1/3 dos

empregos industriais nestes continentes(BLS, 2022)(ACEA, 2022). Estar alinhado com as práticas da indústria automotiva moderna é importante para o desenvolvimento profissional na graduação. A produção de veículos automotores está nas mãos de algumas empresas muito grandes, e os pequenos produtores independentes são quase inexistentes. Isso se deve à produção em massa, que exige investimentos significativos em equipamentos e ferramentas.

O processo de montagem tem um padrão bastante uniforme em todo o mundo. Normalmente, existem duas linhas de montagem principais, carroceria e chassi, que por vezes são unidos numa única estrutura, o monobloco. O monobloco é uma estrutura mais moderna que facilita a construção do carro, diminuindo os custos de produção. Contudo, no caso de veículos como o Baja SAE, para se obter uma melhor dinâmica e rigidez, não se usa um monobloco, e sim uma estrutura tubular para o chassi, com a carroceria afixada a parte.

Primeiro, os painéis da carroceria são soldados, as portas e janelas são instaladas e a carroceria é pintada e aparada. Na segunda fila, o quadro tem molas, rodas, caixa de direção, motor e transmissão, além dos freios e sistema de escapamento. As duas linhas se fundem onde o carro é finalizado, exceto para pequenos itens e testes e inspeções necessários (ROCHA, 2005).

2.1.2 Desenvolvimento de projeto de novos carros

O processo de trazer carros novos para o mercado é amplamente padronizado. À medida que a indústria se torna cada vez mais competitiva e internacional, os fabricantes usam vários meios para reduzir o tempo desde o projeto até a produção para menos de três anos (IEA, 2022)

Os fornecedores atuam como parceiros de desenvolvimento e fornecedores de sistemas e componentes, mas também são responsáveis pelo controle de vários processos operacionais, como design, mudança e gestão da qualidade. Portanto, eles são cada vez mais responsáveis por desenvolver novas tecnologias e disponibilizá-las para produtos do tipo *Original Equipment Manufacturer*. Apesar das pressões de

custo consideráveis, eles estão cada vez mais impulsionando a inovação e exigindo investimentos significativos (HERLT, 2022).

2.1.3 Sistemas de um veículo

O sistema proposto neste trabalho possui uma grande dependência do entendimento mecânico do veículo, portanto é importante compreender a contribuição de cada parte mecânica para o desempenho do projeto. Pela complexidade de um carro é comum dividir o funcionamento em subsistemas a fim de facilitar a abordagem. Nesse sentido, abordaremos nas próximas seções nove partes do sistema: motor, transmissão, ignição, exaustão, eletroeletrônica, freios, corpo, suspensão e direção. (MILLIKEN, 1995) (STONE, 2004)

Motor e transmissão

O motor de combustão interna queima combustível dentro dos cilindros e converte a força de expansão da combustão em força rotativa usada para impulsionar o veículo. Existem vários tipos de motores de combustão interna: motores de pistão alternativo de dois e quatro ciclos, turbinas a gás, pistão livre e motores de combustão rotativa. Entretanto, o motor alternativo de quatro ciclos foi refinado a tal ponto que tem domínio quase completo no campo automotivo (ZHANG, 2018).

Ele é responsável por converter a energia química em energia mecânica. Para operar, ele requer ar limpo para o combustível, água para resfriamento, eletricidade para gerar a centelha e óleo para lubrificação, além de uma bateria e um motor de arranque elétrico para fazê-lo dar partida (CROLLA, 2009).

O sistema de transmissão do automóvel consiste em vários componentes. Esses componentes trabalham juntos para transmitir o movimento rotativo no virabrequim de forma suave e eficiente para as rodas da estrada. As principais transmissões utilizadas hoje são a manual, em que o motorista deve usar a embreagem para controlar a transferência de torque do motor para a transmissão; a automática, muda as marchas automaticamente enquanto o veículo está em movimento; e a

transmissão variável contínua, que possui trocas de marchas continuamente, esta é a utilizada no BAJA SAE.

No caso da transmissão variável contínua (CVT), também conhecida como transmissão sem deslocamento, há uma quantidade contínua ou ilimitada de marchas. Ao contrário de outros tipos, a transmissão CVT não usa engrenagens como meio de produzir velocidades variadas, mas depende de um *design* acionado por correia de duas polias de borracha ou metal. As duas polias trabalham em sincronia, imitando o efeito produzido quando engrenagens de diâmetros diferentes são engatadas. A capacidade das polias de aumentar e diminuir seus diâmetros efetivos permite que a transmissão CVT se mova com fluidez através de uma gama ilimitada de engrenagens efetivas. Quanto mais marchas uma transmissão tiver, melhor ela operará em uma ampla faixa de velocidades.

Ignição e exaustão

A única função do sistema de ignição é gerar uma tensão muito alta para a vela de ignição para incendiar a mistura ar-combustível no motor. Os principais componentes deste sistema incluem a chave de ignição, a bateria, a bobina, a árvore de camés e o braço do rotor (ZHANG, 2018).

O sistema de ignição do carro deve funcionar em perfeita harmonia com o resto do motor. O objetivo é acender o combustível exatamente no momento certo para que os gases em expansão possam fazer a quantidade máxima de trabalho. Se o sistema de ignição for acionado na hora errada, a energia cairá e o consumo de combustível e as emissões poderão aumentar.

Quanto ao sistema de exaustão, quando o motor está funcionando, ele produz uma grande quantidade de resíduos. Essas substâncias são prejudiciais tanto para as pessoas quanto para o meio ambiente. Portanto, o sistema de escape, funciona para filtrar, canalizar e expulsar os gases de escape para longe do carro.

Eletroeletrônica

Este sistema é a parte principal do trabalho, compreende a maior parte do desenvolvimento. Consiste de duas partes, o sistema elétrico, responsável pela alimentação de componentes elétricos, e o sistema eletrônico, responsável pelo gerenciamento de atuadores e processamento de sinais e dados.

O sistema elétrico consiste principalmente de bateria, motor de partida e alternador. A bateria atua como uma unidade de armazenamento de energia enquanto o alternador envia energia de volta para a bateria do motor em funcionamento, mantendo-a carregada. O sistema de carregamento é composto por alternador e bateria. Esta bateria é usada para alimentar o motor de partida ajuda o motor a começar a funcionar, enquanto o alternador é usado para carregar a bateria e outros componentes elétricos do veículo (BHOSALE, 2023).

Todos esses componentes elétricos são conectados à bateria por meio de fios, que transportam a corrente elétrica, e fusíveis, que protegem a fiação. Os fios variam em espessura dependendo de sua função dentro do sistema elétrico. Eles devem ser do tamanho apropriado para lidar com a quantidade de corrente transferida ou podem superaquecer, queimar um fusível.

Atualmente, a eletrônica automotiva tem sido alvo de consideráveis investimentos em pesquisa e desenvolvimento em todo o mundo. Estima-se que cerca de 90% da inovação no setor automotivo esteja concentrada na área de elétrica e eletrônica. À medida que a tecnologia avança, os veículos modernos têm cada vez mais suas funções controladas por *software*. A expansão das funcionalidades dos veículos, baseadas em *software*, tornou-se uma tendência padrão na indústria automotiva atual (CONRAD, 2023).

Anteriormente, a eletrônica em automóveis era usada principalmente para controlar o motor. Mas, agora, os avanços mais recentes fornecem sistemas eletrônicos relacionados às operações automobilísticas, como melhorar o ato de dirigir, a eficiência de combustível, o conforto dos motoristas e dos pilotos.

Os sistemas embarcados desempenham um papel essencial na arquitetura auto-

mobilística, já que eles são usados no sistema de ABS, telemática, sistema de música e *airbags* de segurança, rádio, capacidade de estacionamento, etc. Além disso, é importante saber que as peças eletrônicas usadas em automóveis podem ser controladas digitalmente para a maioria dos operações do carro (DENTON, 2017).

A eletrônica automotiva são sistemas distribuídos que podem ser classificados em diferentes tipos com base em diferentes domínios, como eletrônica do motor, eletrônica de transmissão, eletrônica do chassi, segurança passiva, assistência ao motorista, conforto do passageiro, sistemas de entretenimento, sistemas eletrônicos e integrados de *cockpit*.

Corpo

A carroceria é uma variedade bastante complexa de grandes seções de aço. Essas seções foram estampadas em formas específicas que compõem a carroceria do seu carro. Essas peças são projetadas para fazer muitos trabalhos ao mesmo tempo; proteger os ocupantes dos elementos e em colisões, fornecer suportes sólidos para todos os outros sistemas e cortar o ar com resistência mínima (CROLLA, 2009).

O termo “chassis” é usado para designar o carro completo menos a carroceria. O chassi, portanto, consiste no motor, sistema de transmissão de energia e sistema de suspensão, todos adequadamente fixados ou suspensos de uma estrutura estruturalmente independente. Embora esta construção seja amplamente utilizada, um número quase igual de fabricantes de automóveis emprega um projeto no qual a estrutura e a carroceria são soldadas para formar uma unidade integral.

Os principais requisitos de projeto do chassi do automóvel, seja ele estruturalmente independente ou parte integrante da carroceria, é que ele forneça grande resistência com peso mínimo. Deve ser suficientemente rígido para absorver os impactos da estrada e os choques transmitidos pelas rodas e eixos, e deve ser capaz de suportar as tensões de torção encontradas nas condições de operação.

Freios

O sistema de freio consiste no pedal de freio, servo-freio, cilindro mestre de freio, linhas e mangueiras de freio, pinças e pistões de freio, pastilhas de freio a disco, rotores de freio a disco, fluido de freio e ABS. Mas os componentes do sistema de freio variam de acordo com o modelo e os tipos.

De maneira geral é um dispositivo mecânico projetado para restringir o movimento, absorvendo energia de um sistema em movimento, geralmente por meio de fricção. É usado para desacelerar ou parar um veículo em movimento, rodas, eixo, etc. A desaceleração é conseguida pelo fluido hidráulico, que muitas vezes é sangrado para obter o melhor desempenho da frenagem (MILLIKEN, 1995).

Suspensão e direção

Embora tecnicamente dois sistemas separados, os sistemas de suspensão e direção trabalham para manter o veículo sob controle. O sistema de suspensão é composto por seus amortecedores e suportes. O sistema de direção permite ao condutor dar prumo ao veículo.

O chassi do veículo é conectado às rodas dianteiras e traseiras por molas, amortecedores e eixos. Um sistema de suspensão refere-se a todas as peças que trabalham juntas para proteger contra choques. As molas conectam o chassi automotivo aos eixos de forma indireta. Isso é feito para proteger a carroceria do veículo de choques na estrada causados por salto, inclinação, rolagem ou oscilação. Esses choques de estrada proporcionam um passeio acidentado e sobrecarregam a estrutura e o corpo do carro.

Portanto o sistema de suspensão é um conjunto de conexões mecânicas, molas e amortecedores que conectam as rodas ao chassi. Tradicionalmente, ele desempenha duas funções: gerenciar o manuseio e a frenagem do veículo para segurança e manter os passageiros confortáveis contra choques, vibrações e outros fatores .

O sistema de suspensão do carro é responsável por suavizar o passeio e manter o controle do veículo. Para oferecer estabilidade de direção e boa dirigibilidade, o

sistema de suspensão aumenta o atrito entre os pneus e a estrada (PACEJKA, 2012).

2.2 Sistemas embarcados

O sistema embarcado é um sistema de *hardware* de computador baseado em microprocessador com *softwar* projetado para executar uma função dedicada, seja como um sistema independente ou como parte de um grande sistema. (NAVET; SIMONOT-LION, 2017)

As complexidades variam de um único microcontrolador a um conjunto de processadores com periféricos e redes conectados; de nenhuma interface de usuário a interfaces gráficas de usuário complexas. A complexidade de um sistema embarcado varia significativamente dependendo da tarefa para a qual foi projetado. (MARTIN, 2016)

Os sistemas embarcados são gerenciados por microcontroladores ou processadores de sinal digital DSP, ASIC ou FPGA. Esses sistemas de processamento são integrados a componentes dedicados ao manuseio de interfaces elétricas e mecânicas.

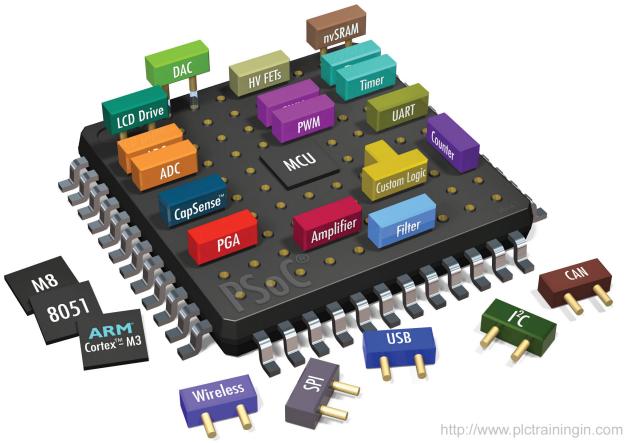
As instruções de programação de sistemas embarcados, conhecidas como *firmware*, são armazenadas em memória somente de leitura ou chips de memória flash, rodando com recursos limitados de *hardware* do computador. Os sistemas embarcados se conectam com o mundo externo por meio de periféricos, conectando dispositivos de entrada e saída. Um sistema didático de um *chip* e seus componentes pode ser visto na Figura 2.1.

2.2.1 Estruturas básicas de um sistema embarcado

- Sensor – Mede a grandeza física e a converte em um sinal elétrico que pode ser lido por um observador ou por qualquer instrumento eletrônico como um conversor A2D. Um sensor armazena a quantidade medida na memória;
- Conversor AD – Um conversor analógico-digital converte o sinal; analógico enviado pelo sensor em um sinal digital;

- Processador e ASICs – Processadores processam os dados para medir a saída e armazená-los na memória;
- Conversor DA – Um conversor digital-analógico converte os dados digitais alimentados pelo processador em dados analógicos;
- Atuador - Um atuador compara a saída fornecida pelo conversor DA com a saída real (esperada) armazenada nele e armazena a saída aprovada.

Figura 2.1: Estrutura de um microcontrolador.



Fonte: Tevatron Technologies[1]

Componentes de um sistema embarcado

Para o funcionamento de um sistema embarcado de qualquer aplicação é necessário a presença de vários componentes, nos quais abordaremos em seguida (WILLIAMS, 2023).

O primeiro deles são as fontes de energia, servindo de alimentação para o sistema embarcado. Pode ser fornecida a partir de uma bateria ou de um adaptador. Dependendo do aplicativo em que o sistema embarcado está sendo usado.

O processador atua como o cérebro principal em um sistema embarcado, normalmente é usado um microcontrolador ou um microprocessador.

[1]Disponível em https://www.tevatrontech.com/. Acesso em: 20 de set. 2022

Memórias, como a ROM usada para armazenar um código fonte do programa, uma vez que é inicializado, e é a partir dessa que o sistema obterá o código necessário para operar. Outro tipo é a memória RAM, uma memória volátil, que é usada para armazenar dados temporariamente e é normalmente mais rápida que as memórias ROM (ELECTRONICSHUB, 2017).

Temporizadores e Contadores são para aplicações que requerem contagem de tempo para funcionar ou operar. Portanto, a maioria dos microcontroladores está equipada com um ou mais sistemas de temporização de precisão que podem ser usados para executar uma variedade de funções de temporizador de precisão, incluindo a geração de eventos em horários específicos, a determinação da duração entre dois eventos ou a contagem de eventos.

Portas de comunicação são utilizadas para estabelecer conexões com outros dispositivos embarcados. Essas portas são especialmente projetadas para suportar protocolos de comunicação conhecidos, que desempenham trocas de mensagens entre sistemas de computador. Por meio desses protocolos, é possível enviar e receber mensagens de forma consistente, garantindo uma comunicação confiável e eficiente. Essas portas de comunicação oferecem a capacidade de interconectar dispositivos e permitir a transferência de dados em conformidade com os padrões estabelecidos pelos protocolos utilizados.

O escopo das portas I/O em um sistema embarcado inclui sensores e atuadores, conversão de sinais entre o domínio analógico e digital e etc.

2.2.2 Protocolos de comunicação

O uso de protocolos de comunicação embarcados foi extenso no projeto, conhecer o funcionamento auxiliará o entendimento do projeto. De maneira geral, os protocolos são um conjunto de regras que permitem que dois ou mais sistemas de comunicação comuniquem dados através de um meio físico. As regras, regulamentos, sincronização entre sistemas de comunicação, sintaxe a ser seguida e semântica são todos definidos pelo termo protocolo. Os protocolos podem ser implementados

por *hardware* e *software* ou pela combinação de ambos. Os sistemas de comunicação analógicos e digitais utilizam amplamente vários protocolos de comunicação. Além disso, cada protocolo possui sua própria área de aplicação. Os processos como configuração do sistema, seleção da taxa de transmissão e transmissão e recebimento de dados estão associados à camada de aplicação (LEENS, 2009).

Em sistemas embarcados, os protocolos de comunicação ocupam um lugar especial, pois abrem maneiras de trocar dados entre dispositivos com eficiência. Existem muitos protocolos, mas em geral, os protocolos de comunicação estão associados à camada física que descreve os sinais, intensidade do sinal, mecanismos de *handshake*, arbitragem de barramento, endereçamento de dispositivos, com ou sem fio e linhas de dados .

Protocolos de comunicação

Um canal de comunicação pode ser usado para se comunicar em uma direção de cada vez ou em ambas as direções ao mesmo tempo. Os termos half-duplex e full-duplex descrevem esses modos de transmissão (UNIT-V, 2023).

Dispositivos *half-duplex* só podem transmitir em uma direção de cada vez. Portanto, embora os dados possam se mover em duas direções, não podem ser ao mesmo tempo. Ambos os dispositivos são capazes de transmitir e receber, portanto, quando um dispositivo está enviando, o outro está recebendo.

O termo *full-duplex* descreve a transmissão e recepções simultâneas de dados em um canal. Um dispositivo full-duplex é capaz de transmissões de dados de rede bidirecionais ao mesmo tempo.

Protocolo SPI

O SPI é um protocolo *full-duplex* que utiliza a configuração mestre e escravo para estabelecer a comunicação. A Motorola inventou este protocolo em 1980.

O protocolo SPI é composto por quatro pinos:

- SCLK: *Serial Clock* (Fonte de relógio para transmissão de dados);

- MOSI: *Master Output Slave Input* (Dados de saída do Master);
- MISO: *Master Input Slave Output* (Saída de dados do Slave);
- SS: *Slave Select* (Seleção de escravo de vários escravos pelo mestre).

Pinos SS adicionais são necessários quando o dispositivo mestre precisa controlar mais de um escravo, já que um mestre pode controlar vários escravos. O protocolo SPI suporta transferência de dados de alta velocidade com frequências de barramento normalmente em torno de 50Mhz.

Protocolo I2C

I2C é um protocolo de comunicação de barramento serial de dois fios inventado em 1982 pela Philips Semiconductors. É uma interface de comunicação de dois fios e comumente usada para conectar dispositivos de baixa velocidade com os microcontroladores (UNIT-V, 2023).

I2C usa dois pinos para estabelecer comunicação:

- SDA: Linha de dados seriais;
- SCL: Linha de Relógio Serial;

O protocolo I2C suporta diferentes velocidades de dados com base em seu modo de operação. Velocidade de transferência de dados 100 kbit/s quando operando em modo padrão e 400 kbit/s quando operando em modo *full speed*.

Protocolo UART

UART é um protocolo de comunicação serial desenvolvido pela Digital Equipment Corporation em 1960. Suporta comunicação *full duplex, half duplex e simplex*. Este é um dos protocolos mais populares e esta em quase todos os microcontroladores (UNIT-V, 2023).

A comunicação em UART ocorre através de dois pinos:

- RX: Recebe os dados de entrada;

- TX: Transmite os dados de saída.

A transmissão de dados em UART ocorre por meio de pacotes de dados que geralmente consistem em 8 *bits* de dados, *bits* de início e *bit* de paridade para realizar a correção de erros.

Protocolo CAN

O protocolo *Controller Area Network* ou CAN foi projetado para compartilhar dados entre várias interfaces para reduzir a complexidade da conectividade e da fiação. Isso foi desenvolvido e usado pela primeira vez em automóveis para estabelecer comunicação entre diferentes partes do automóvel. A característica única do protocolo CAN é que os dados transmitidos de um dispositivo ou nó CAN estarão disponíveis para todos os dispositivos conectados no barramento (UNIT-V, 2023).

Na rede CAN padrão, qualquer nó pode transmitir dados através do barramento para qualquer nó. O quadro de dados CAN começa com um *bit* inicial, seguido por um identificador de 11 *bits* que define a prioridade da mensagem. A mensagem com maior prioridade ganha acesso ao barramento CAN, quando dois dispositivos tentam enviar mensagens ao mesmo tempo. A verificação de redundância cíclica e os *bits* ACK seguem então e são usados para verificação de erros para garantir a transmissão adequada dos dados. Um final de quadro de 7 *bits* que marca o fim da mensagem.

Protocolo ETHERNET

O *ethernet* é o protocolo de rede mais popular usado para estabelecer conexões diretamente de LAN para WAN. Com o crescimento da IOT, este protocolo tornou-se indispensável em sistemas Embarcados. Este protocolo de comunicação passou por uma série de revisões ao longo dos anos para melhorar sua conectividade e velocidade. Apesar de não ter sido utilzado no projeto, uma variação do protocolo *ethernet*, chamado de *ethenet* automotiva, vem se tornando um padrão para as redes automotivas, já que cada vez se trafegam mais dados, e protocolos com maiores

capacidades de banda são necessários.

2.3 Placas de circuito impresso e prototipação

Uma placa de circuito impresso, como na Figura 2.2, é uma estrutura rígida que contém circuitos que consistem em superfícies metálicas incorporadas chamadas traços e áreas metálicas maiores chamadas planos. Os componentes são soldados à placa em *pads* de metal que se conectam aos circuitos da placa. Isso permite que os componentes sejam conectados uns aos outros. Uma placa de circuito pode consistir em uma, duas ou mais camadas de circuitos(RITCHEY; ZASIO, 2003).

Figura 2.2: Placa de circuito impresso.

Fonte: Página da Axis Communications[2]

As placas de circuito são construídas com um material de núcleo dielétrico menos condutor para garantir o transporte de circuito limpo e espaçado com metal adicional e camadas dielétricas conforme necessário. O material dielétrico padrão usado para

[2]Disponível em https://www.axis.com/pt-br/products/axis-tp1903-pcba-housing-t92. Acesso em: 22 de set. 2022

placas de circuito é um composto retardador de chama de tecido de fibra de vidro e epóxi chamado FR-4, enquanto os traços de metal e planos usados para circuitos são geralmente compostos de cobre.

Tanto a escolha do material e método de fabricação, quanto as normas que serão adotadas tem um impacto significativos no desenvolvimento e na confiabilidade do projeto, com isso é importante conhecer as opções para julgar as melhores alternativas para o projeto.

2.3.1 Classificações

As placas de circuito impresso são usadas para uma variedade de propósitos. Uma característica distintiva dos PCBs é sua classe um, dois ou três, segundo a IPC-6011. Essa classe da PCB indica sua confiabilidade geral e qualidade de projeto e é de extrema importância para nossa aplicação, que se encontra um ambiente extremo durante todo o uso do dispositivo (PETERSON, 2022).

Classe 1

Placas de classe 1 designam um eletrônico de consumo. A primeira classe de produtos eletrônicos é chamada de categoria “eletrônica geral”. Este consiste em placas com os mais baixos requisitos de qualidade e é encontrado principalmente em produtos com um ciclo de vida curto esperado.

Classe 2

Placas de classe 2 são encontradas em dispositivos onde a alta confiabilidade é importante, mas não crucial. Esses dispositivos tentam minimizar as falhas. Os dispositivos eletrônicos de classe 2 abrangem todos os eletrônicos onde o desempenho contínuo e um ciclo de vida estendido são necessários, em que um serviço ininterrupto é desejado, mas não crítico. Em outras palavras, esses são itens em que uma falha no início do ciclo de vida o aborreceria o consumidor, mas não colocaria sua vida em risco.

Classe 3

As placas Classe 3 representam os padrões de fabricação mais exigentes de um PCB. Simplificando, se uma prancha Classe 3 falhar, vidas estarão imediatamente em jogo. A terceira classe de placas de circuito está sujeita a diretrizes rígidas devido à sua importância no campo. Enquanto os eletrônicos de Classe 1 são geralmente itens baratos e facilmente substituíveis e os eletrônicos de Classe 2 são mais importantes e requerem um ciclo de vida mais longo, os eletrônicos de Classe 3 são itens de missão crítica. Seja um marcapasso ou um radar militar, um produto que precisa atender aos requisitos IPC Classe 3 deve usar componentes eletrônicos de alta confiabilidade para garantir um serviço ininterrupto.

2.3.2 Material e rigidez

Por motivos semelhantes ao tópico anterior, conhecer os materiais é importante para evitar falhas mecânicas precoces durante o ciclo de vida, visto que o baja é um veiculo de corrida *off-road*.

Placas rígidas

Placas de circuito rígido são tipicamente a grande maioria das placas de circuito que um projetista encontrará, onde o layout da placa de circuito está contido em um substrato rígido feito por um processo de laminação térmica de alta pressão. O material comum para essas placas é o FR-4, mas dependendo das necessidades específicas do projeto, ele pode ser modificado para enfatizar ou melhorar certas propriedades da placa (CIRCUITS, 2022b).

Placas flexíveis

As placas flexíveis são compostas por materiais menos rígidos que permitem maior deflexão. O material parece uma reminiscência de um rolo de filme ao toque, e a espessura da placa geralmente é muito menor do que as placas rígidas padrão. Embora tenham visto alguns usos, as folhas flexíveis prometem inaugurar o próximo

passo na tecnologia vestível e remover as limitações planares inerentes aos atuais dispositivos de folhas rígidas.

Núcleo de metais

PCBs com núcleo de metal são um ramo do design de placa de circuito rígido com a capacidade de melhorar a dissipação de calor em toda a placa para proteger circuitos sensíveis. Este estilo pode ser usado como opção para projetos de alta corrente para evitar desgaste térmico e falhas.

2.3.3 Métodos de fabricação

Existem principalmente cinco tecnologias padrão usadas na fabricação de PCB, e é importante conhece-las para explorar as possibilidades e usar o melhor método para cada objetivo (CIRCUITS, 2022a).

Por usinagem

Inclui furação, furação e roteamento em PCB com maquinário padrão existente e também novas tecnologias como corte a laser e jato de água. A resistência da placa precisa ser levada em consideração durante a usinagem para obter diâmetros de furos precisos. Furos pequenos tornam esse método caro e menos confiável devido à relação de aspecto reduzida e também dificultando o revestimento.

Por Imagem

Transfere o desenho do circuito para camadas individuais. PCBs de face simples ou dupla face podem usar a tecnologia de impressão de tela simples para criar os padrões em uma base de impressão e gravação. Mas isso tem uma limitação na largura de linha mínima alcançável. Para placas de linhas finas e placas multicamadas, é usado o Photoimaging.

Por laminação

Este processo é usado principalmente para a fabricação de placas multicamadas, ou os laminados de base de placas de face simples/dupla. Folhas de vidro impregnadas com resina epóxi de estágio B são pressionadas entre as camadas usando uma prensa hidráulica para unir as camadas.

Por gravura

É a remoção de metal e dielétrico indesejados da placa ocorre por processos secos ou úmidos. A uniformidade da gravação é a principal preocupação neste estágio e para estender as capacidades de gravação em linha fina, novas soluções de gravação anisotrópica estão sendo desenvolvidas.

2.3.4 Encapsulamento de componentes

Tradicionalmente, a tecnologia *through hole* (THT) era usada para construir a grande maioria dos PCBs. Nos últimos anos, no entanto, o uso da tecnologia de montagem em superfície (SMT) cresceu em popularidade e é cada vez mais usado no lugar da tecnologia de furos passantes.

Os dispositivos de montagem em superfície (SMD) terão caudas planas coplanares ou condutores que permitem que o componente se apoie em uma pista plana exposta na PCB. Portanto não sendo necessários furos no PCB. A tecnologia *through hole* envolve a inserção de cabos de componentes em furos na placa de circuito impresso. As ligações podem então ser soldadas em almofadas ou terras no lado da solda da placa.

2.3.5 Tipos de PCB

Outro ponto importante é o tipo da PCB, quando falamos em complexidade e densidade podemos classificar o design nos pontos abordados nas próximas seções (CIRCUITS, 2022a).

PCB de camada única

Os circuito impresso de camada única são as mais simples de projetar e fabricar. Essas resultados têm uma única camada de material condutor em apenas um lado de um substrato não condutor.

PCB de camada dupla

Um PCB de duas camadas tem uma camada condutora na parte superior do substrato não condutor e uma camada condutora na parte de trás. As duas superfícies condutoras podem ser conectadas por meio de orifícios pintados no substrato que se conectam às almofadas em ambos os lados do PCB.

PCB multicamada

Este termo refere-se a uma placa de circuito que possui três ou mais camadas condutoras. As camadas condutoras estão na parte superior e inferior, bem como pelo menos uma camada condutora ensanduichada entre o substrato não condutor.

PCB de interconexão de alta densidade

Os PCBs HDI aproveitam a tecnologia de fabricação baseada em precisão para reunir o máximo de funcionalidade em um pequeno espaço. Isso é feito usando muitas camadas condutoras, microvias perfuradas a laser, linhas finas e tolerâncias e materiais laminados avançados. Os PCBs HDI podem acomodar o roteamento complexo de chips de alta contagem de pinos e outros componentes miniaturizados de alta tecnologia.

PCB de alta frequência

A principal diferença na fabricação de placas de circuito impresso de alta frequência está no próprio design. Esses PCB são projetados para facilitar os sinais acima de 1 gigahertz. Dependendo da aplicação, placas de circuito impresso de alta frequência podem exigir o uso de materiais laminados avançados e impedância controlada.

2.3.6 Desenvolvimento de PCB

O primeiro passo de um projeto de PCB é fazer um esquema elétrico do circuito. O esquema servirá como um modelo para traçar os traços e colocar os componentes no PCB. Além disso, o *software* de edição de PCB pode importar todos os componentes, *footprints* e fios para o arquivo PCB, o que facilitará o processo de *design*.

Uma vez o esquemático pronto, a criação do *layout* da placa é o próximo passo. Geralmente, um *layout* denota a maneira como as partes de um item específico são dispostas ou organizadas. Na mesma linha, um *layout* de PCB é um termo amplo que indica vários processos necessários no projeto de um PCB. Envolve fazer traços, montar recortes de furos, rotular e especificar a localização dos componentes, entre outros(CHARRAS; TAPPERO; STAMBAUGH, 2018) (DALMARIS, 2022).

2.4 Linguagem e estruturas de programação

As linguagens de programação definem e compilam um conjunto de instruções para a CPU realizar qualquer tarefa específica. Cada linguagem de programação tem um conjunto de palavras-chave junto com a sintaxe que ela usa para criar instruções.

Algumas dessas linguagens fornecem menos ou nenhuma abstração, enquanto as outras fornecem uma abstração muito alta. Com base nesse nível de abstração, existem dois tipos de linguagens de programação:

- Linguagem de baixo nível;
- Linguagem de alto nível.

A principal diferença entre linguagens de baixo e alto nível é que qualquer programador pode entender, compilar e interpretar uma linguagem de alto nível de forma viável em comparação com a máquina. As máquinas, por outro lado, são capazes de compreender a linguagem de baixo nível de forma mais viável em comparação com os seres humanos.

2.4.1 Linguagem C

A linguagem C foi desenvolvida por Dennis Ritchie em 1972 nos Laboratórios Bell . Em 1989, a linguagem C foi padronizada com o padrão ANSI de 1989 para C. A linguagem C evoluiu de três linguagens estruturadas diferentes ALGOL, BCPL e B Language . Ele usa muitos conceitos dessas linguagens enquanto introduziu muitos novos conceitos, como tipos de dados, struct, ponteiro e etc.

Programas escritos em linguagem C levam muito pouco tempo para serem executados e quase executam na velocidade das instruções em linguagem assembly . (As instruções de nível de montagem nada mais são do que comandos diretos para se comunicar com o *hardware* do computador)

Inicialmente, a linguagem C foi usada principalmente para escrever programas em nível de sistema, como projetar sistemas operacionais, porque nos anos 80 (1980-89) a luta para tornar um sistema operacional estável e aceito mundialmente estava acontecendo. Hoje é a linguagem de programação mais popular na área de *software* para o desenvolvimento de dispositivos eletrônicos embarcados e desempenha um papel fundamental na execução de atividades em sistemas dedicados (TOULSON; WILMSHURST, 2016) (MARTIN, 2016).

2.4.2 Linguagem Python

Python é uma linguagem de programação de alto nível de uso geral amplamente utilizada. Foi criado por Guido van Rossum em 1991 e desenvolvido pela Python Software Foundation. Ele foi projetado com ênfase na legibilidade do código e sua sintaxe permite que os programadores expressem seus conceitos em menos linhas de código.

É uma linguagem de programação interpretada, orientada a objetos e de alto nível com semântica dinâmica. Suas estruturas de dados integradas de alto nível. A sintaxe simples e fácil reduzindo o custo de manutenção do programa, também suporta módulos e pacotes, o que incentiva a modularidade do programa e a reutilização de código (VANDERPLAS, 2022).

2.4.3 Estruturas de programação

Um *framework* em programação é uma ferramenta que fornece componentes prontos ou soluções customizadas para acelerar o desenvolvimento. Um *framework* pode incluir uma biblioteca, mas é definido pelo princípio de inversão de controle (IoC). Com a programação tradicional, o código personalizado chama a biblioteca para acessar o código reutilizável. Com o IoC, a estrutura chama partes de código personalizadas quando necessário. O principal intuito de na utilização de *frameworks* é economizar tempo e garantir que as melhores práticas sejam seguidas no desenvolvimento (KRIGER, 2022).

Mbed OS

O Mbed OS é um sistema operacional de código aberto para placas Cortex-M da Internet das Coisas (IoT) que inclui todos os recursos necessários para desenvolver um produto conectado baseado em um microcontrolador: baixa potência, restrita e conectada. O Mbed OS fornece uma camada de abstração para os microcontroladores em que é executado, para que os desenvolvedores possam escrever aplicativos C/C++ que sejam executados em qualquer placa habilitada para Mbed, incluindo segurança, conectividade, um RTOS e *drivers* para sensores e dispositivos (MBED, 2023).

Jupyter

O notebook Jupyter é um IDE de código aberto, padrões abertos e serviços para computação interativa em várias linguagens de programação usado para criar documentos Jupyter que podem ser criados e compartilhados com códigos ativos. Além disso, é um ambiente computacional interativo baseado na *web*.

Um documento do Jupyter Notebook é um REPL baseado em navegador que contém uma lista ordenada de células de entrada/saída que podem conter código, matemática, gráficos e etc . Abaixo da interface, um *notebook* é um JSONdocumento, seguindo um esquema versionado, geralmente terminando com a extensão

.ipynb (ORG, 2023).

2.5 Baja SAE

A Baja SAE consiste em competições que simulam projetos reais de engenharia e seus desafios associados. Estudantes de engenharia são encarregados de projetar e construir um veículo todo-o-terreno que possa suportar punições severas em terrenos acidentados. O objetivo de cada equipe é projetar e construir um carro esportivo *off-road* de um lugar com o motorista no corpo, Figura 2.3.

O objetivo da competição é oferecer aos alunos da SAE um projeto desafiador que inclui tarefas de *design*, engenharia e fabricação que surgem ao trazer um novo produto ao mercado em uma indústria de consumo. As equipes competem entre si para obter permissão para uma empresa fictícia produzi-las. Os alunos devem trabalhar em equipe não apenas para projetar, construir, testar, promover e testar um veículo de acordo com as regras, mas também para obter apoio financeiro para seu projeto e gerenciar suas prioridades educacionais (SAE, 2023).

Figura 2.3: Veículo Baja SAE.

Fonte: O autor.

Capítulo 3

Materiais e Métodos

3.1 Materiais

Neste capítulo discutiremos sobre os materiais utilizados para a execução das tarefas propostas durante a justificativa e objetivos do projeto.

3.1.1 Materiais para desenvolvimento do *hardware*

Para criação do esquemático e desenho existem muitos *softwares* no mercado, um deles, o KiCad, que é um conjunto de *softwares* de código aberto para Electronic Design Automation (EDA). Nele é possível fazer elaboração do esquemático e layout de PCB com saída Gerber. Podendo ser executado nas plataformas Windows, Linux e macOS (BAUTISTA, 2023).

Para a confecção da placa foi utilizado papel fotográfico 270 gramas em que era impresso a laser o design, e juntamente com uma superfície aquecida, muitas vezes um ferro de passar, e um pano era passado o desenho para fenolite. Também foi utilizado uma estação de solda para componentes eletrônicos, fluxo de solda e uma mini-retífica para a fixação dos componentes na placa.

Além disso, foram utilizados equipamentos eletrônicos para medição e testes elétricos e de funcionamento no dispositivo, como multímetro e osciloscópio, a fim de medir continuidade e comportamento de sinais digitais e analógicos.

3.1.2 Materiais para desenvolvimento do *software*

Um *framework* em programação é uma ferramenta que fornece componentes prontos ou soluções customizadas para acelerar o desenvolvimento. Com a programação tradicional, o código personalizado chama a biblioteca para acessar o código reutilizável. Com o IoC, a estrutura chama partes de código personalizadas quando necessário (KRIGER, 2022).

O principal intuito na utilização de *frameworks* é economizar tempo e garantir que as melhores práticas sejam seguidas no desenvolvimento. Neste trabalho foram utilizados o Mbed OS e o Jupyter Notebook.

O Mbed OS é um sistema operacional de código aberto para placas Cortex-M da Internet das Coisas (IoT) que inclui todos os recursos necessários para desenvolver um produto conectado baseado em um microcontrolador: baixa potência, restrita e conectada. O Mbed OS fornece uma camada de abstração para os microcontroladores em que é executado, para que os desenvolvedores possam escrever aplicativos C/C++ que sejam executados em qualquer placa habilitada para Mbed, incluindo segurança, conectividade, um RTOS e drivers para sensores e dispositivos(MBED, 2023).

O notebook Jupyter é um IDE de código aberto, padrões abertos e serviços para computação interativa em várias linguagens de programação usado para criar documentos Jupyter que podem ser criados e compartilhados com códigos ativos. Além disso, é um ambiente computacional interativo baseado na web (ORG, 2023)
.

3.2 Metodologia

Nas próximas seções deste capítulo serão abordados os métodos e as etapas envolvidas no progresso do desenvolvimento.

A fase inicial de qualquer projeto deve analisar o problema a partir de uma perspectiva macro. Entendendo o que se deseja alcançar, identificando as necessidades

e viabilidade. Para isso a primeira fase mencionada será o planejamento.

3.2.1 Planejamento

Inicialmente o foco foi definir os requisitos básicos do projeto, baseado na motivação da execução do projeto. Portanto, o primeiro passo decorre da identificação das necessidades para a solução (INSTITUTE, 2017).

As necessidades do projeto nascem dos problemas de outros subsistemas que impedem o alcance das metas e objetivos identificados na seção 1.3.2. De maneira geral, a metodologia empregada para a identificação dos pontos relevantes foi baseada em observações empíricas, pesquisas, modelagens, entrevistas e consultas em livros especializados.

Baseado no que foi apresentado na seção 2.5, o veículo possui subsistemas, e em cada um existe a necessidade de que a expectativa de funcionamento seja alcançada. Para controlar isso é preciso que esse comportamento seja observado e medido, a fim de produzir curvas características e dados de performance do veículo.

O método para identificação das grandezas determinantes para o desempenho foi baseado no estudo da regra da competição que atribui pontos para cada prova que é responsável por medir a *performance* do carro em cada subsistema, dessa forma cruzando isso as bibliografias utilizadas pela equipe foi possível saber quais grandezas estão envolvidas em cada prova.

3.2.2 Desenvolvimento do projeto

Nesta segunda fase do projeto, foi realizado o desenvolvimento do *hardware* e *software* com base nas expectativas previamente estabelecidas durante a fase de desenvolvimento inicial.

Para garantir a efetividade e o êxito do projeto, foram tomadas decisões cuidadosas em relação à escolha dos *hardwares* e arquiteturas de *softwares* utilizadas. Essas decisões foram baseadas em uma análise detalhada das necessidades e requisitos específicos do projeto, bem como nas considerações de viabilidade técnica e

econômica.

Ao selecionar os componentes de *hardware*, foram levados em consideração fatores como desempenho, capacidade de processamento, consumo de energia, tamanho físico e custo. Uma análise comparativa de diferentes opções de *hardware* foi conduzida, avaliando suas especificações técnicas, disponibilidade no mercado e suporte de desenvolvimento. Com base nessa análise, foram escolhidos os *hardwares* mais adequados para atender aos requisitos do projeto.

No que diz respeito às arquiteturas de *softwares* utilizadas, foram realizadas pesquisas e análises extensivas para identificar as opções disponíveis. Foram considerados critérios como flexibilidade, escalabilidade, modularidade, capacidade de integração com os componentes de *hardware* selecionados e facilidade de desenvolvimento. Após uma análise criteriosa, as arquiteturas de *softwares* mais adequadas foram selecionadas para garantir um desenvolvimento eficiente e uma implementação bem-sucedida do projeto.

3.2.3 Elaboração de Testes

Na fase de Elaboração de Testes, foram desenvolvidos e implementados procedimentos e critérios de teste para avaliar a eficácia e a confiabilidade do sistema desenvolvido. A finalidade desses testes é verificar se o sistema atende aos requisitos estabelecidos, identificar possíveis falhas e garantir seu desempenho adequado.

Para elaborar um conjunto abrangente de testes, serão considerados diferentes aspectos do sistema, como funcionalidade, desempenho, segurança e usabilidade. Cada um desses aspectos será avaliado por meio de testes específicos, garantindo uma cobertura completa e confiável.

Durante a elaboração de testes, foram registrados os resultados para que ao final desta fase todos os testes sejam utilizados para as tomadas de decisão em relação a ajustes finais e melhorias do Baja.

3.2.4 Análise de Resultados

Na fase de Análise de Resultados, os dados coletados durante os testes e experimentos foram tratados e analisados utilizando técnicas de programação em Python. O objetivo principal desta fase é interpretar os resultados obtidos e obter *insights* relevantes para o projeto, validando as hipóteses iniciais estabelecidas.

Para realizar a análise de resultados, serão utilizadas bibliotecas populares de Python, como NumPy, Pandas e Matplotlib. Inicialmente, os dados brutos serão importados para estruturas de dados apropriadas, como DataFrames do Pandas, para facilitar o processamento e manipulação dos dados.

Durante a análise de resultados, será importante realizar uma avaliação crítica dos achados. Será necessário discutir a confiabilidade e validade dos resultados obtidos, levando em consideração possíveis viéses e limitações do estudo. Essa avaliação crítica permitirá uma interpretação mais precisa e uma discussão embasada dos resultados.

Capítulo 4

Desenvolvimento

Neste capítulo, abordaremos o planejamento, a análise dos parâmetros de estudo, o desenvolvimento do projeto e todos os aspectos relacionados ao *hardware e software*. Iniciaremos com a seleção dos componentes, considerando critérios específicos e levando em conta as necessidades do projeto. Em seguida, detalharemos o *design* do *hardware*, incluindo o esquema elétrico, a disposição dos componentes e a verificação do projeto, garantindo sua integridade e funcionalidade.

Além disso, será discutido questões relacionadas ao desenvolvimento do *software*. Exploraremos a estrutura do código, destacando as principais funcionalidades e módulos envolvidos. Serão abordados os processos de implementação das funções, levando em consideração as melhores práticas de programação e a adequação aos requisitos do projeto.

Ao longo deste capítulo, forneceremos uma visão detalhada de todo o processo de desenvolvimento, desde a concepção das ideias até a concretização do projeto, tanto no âmbito do *hardware* quanto do *software*.

4.1 Planejamento

Inicialmente o foco foi definir os requisitos básicos do projeto, baseado na motivação da execução do projeto. Portanto, o primeiro passo decorre da identificação das necessidades para a solução (INSTITUTE, 2017).

4.1.1 Caracterização dos parâmetros de desempenho

Grandezas determinantes para a performance

Para a determinação dos dados aquisitados visando definir subsistemas e parâmetros determinantes para o desempenho, foram listados os seguintes:

- Transmissão:
    - Rotação do motor;
    - Velocidade do veículo;
    - Temperatura do motor;
    - Aceleração Longitudinal.
- Freio:
    - Pressão na linha de freio;
    - Desaceleração.
- Suspensão:
    - Aceleração lateral;
    - Aceleração vertical.

Detalhamento dos parâmetros

A fim de evitar problemas futuros inesperados e evitáveis, será abordado mais profundamente os dados listados anteriormente para traçar estratégias de aquisição condizentes com o ambiente e o tipo do dado.

Transmissão A transmissão é provavelmente o subsistema de maior valor agregado devido ao impacto na pontuação que o comitê do Baja SAE atribui durante a competição.

Rotação do motor A curva de rotação do motor é um dado primordial para o *setup* da redução variável, pois a relação entre rotação do motor e a da roda é a principal curva característica para definição dos pesos da CVT abordado na seção 2.1.3 (MILLIKEN, 1995).

A informação pode ser obtida através do eixo do motor, ou do conjunto da polia motora, ou, por fim, do pulso elétrico produzido pela vela no conjunto da ignição. Devido a simplicidade de montagem, a terceira opção foi escolhida e pode ser implementada conectando a ferramenta à vela através de um fio condutor para a captação do sinal.

O sinal gerado pelo motor é da ordem de 50V pico a pico e será necessário ferramentas para tratamento do sinal antes da recepção deste no microcontrolador, esse assunto será abordado com mais detalhes nas seções 4.2.1 e 4.2.2.

Velocidade do veículo A velocidade do veículo é, pela mesma motivação da rotação do motor, um dado de alta prioridade, e pode ser obtido através do conjunto da polia movida, ou disco de freio traseiro, ou discos de freio dianteiro.

Foi escolhido a aquisição pelos discos de freios dianteiros, já que a roda traseira é tracionada e pode patinar em acelerações. Essa aquisição é feita através de um sensor indutivo detectando os furos do disco.

Temperatura do motor A temperatura do motor pode ser captada diretamente através do termistor que já existe nos motores padronizados da categoria. O dado não é tão relevante para a performance, mas pode ser necessário em um eventual projeto de arrefecimento.

Aceleração Longitudinal A aceleração longitudinal é uma informação importante para avaliação de performance cinemática do veículo, e é amplamente utilizado para *Key Performance Indicator* entre os ano-modelo dos veículos. Existem duas maneiras de obter o dado, primeiro a partir de um acelerômetro embarcado, ou segundo, através da derivação numérica do dado de velocidade no tempo. No

projeto, ambos serão possíveis já que o acelerômetro, como vamos ver mais a frente, é um item necessário para avaliação do conjunto de suspensão, e o dado de velocidade no tempo também já é um item demandado anteriormente.

Freio Os freios também tem uma participação relevante no carro, apesar de não impactar significativamente em desempenho, eles são um item de segurança fundamental no veículo.

Pressão na linha de freio A pressão na linha de freio é um parâmetro de projeto no sistema que pode indiretamente medir a força aplicada no disco de freio, verificar a vedação do sistema e ainda servir de um dado base para ergonomia do sistema. O sensoriamento pode ser feito através de um sensor de pressão conectado ao conduíte da linha, que transforma a pressão em um sinal analógico de tensão.

Desaceleração A desaceleração é o principal *output* do sistema, quanto maior o valor, mais rapidamente o carro irá voltar ao repouso, entregando mais segurança ao condutor e pessoas próximas. A aquisição pode ser feita através de um acelerômetro de 3 eixos, ou da derivada numérica da velocidade no tempo.

Suspensão A suspensão desempenha um papel fundamental no comportamento dinâmico lateral e vertical do veículo, sendo considerado o segundo sistema mais importante. Ela é responsável pelo controle e estabilidade do veículo durante a sua operação.

Curso da suspensão O curso da suspensão é crucial para a capacidade do seu veículo de transportar cargas e sobrepor obstáculos, além de determinar o conforto do seu passeio em terrenos variáveis e o comportamento ao frear, acelerar e fazer curvas. A aquisição é normalmente feita através de potenciômetros lineares fabricados para essa finalidade, que são acoplados junto ao amortecedor.

Aceleração lateral e vertical As acelerações vertical e lateral ajudam a diagnosticar o comportamento do veículo, possibilitando uma comparação quantitativa com o comportamento esperados de acordo com as simulações realizadas. Normalmente aquisitado por acelerômetro de três ou seis eixos posicionados em um ponto fixo no chassi.

4.2 Desenvolvimento do *hardware*

4.2.1 Escolha dos componentes eletrônicos

A seleção dos componentes eletrônicos foi baseada em argumentos fundamentados e apresentados em cada seção de definição dos componentes. Esses argumentos forneceram diretrizes para as escolhas, levando em consideração os critérios relevantes para cada aplicação específica como desempenho, confiabilidade, disponibilidade, custo e compatibilidade.

A decisão final de cada componente foi tomada levando em conta a melhor combinação de critérios e a adequação às necessidades do projeto.

Definição do microcontrolador

Ao avaliar os microcontroladores mais comumente utilizados, encontramos a família ATmega, baseada na arquitetura AVR, presente nos produtos Arduino, que são amplamente utilizados por makers e estudantes para projetos rápidos e prototipagem. Por outro lado, os microcontroladores Cortex, baseados na arquitetura ARM, dominam o mercado profissional. Além disso, a arquitetura Xtensa LX6 da Espressif Systems tem apresentado um crescimento significativo. Para este projeto, consideramos os microcontroladores STM32, da família Cortex, fabricados pela ST, e o ESP32, fabricado pela Espressif, ambos amplamente utilizados para prototipagem e projetos de desenvolvimento rápido (ODUNLADE, 2020).

Devido à disponibilidade e *performance*, o STM32 foi a família escolhida para a aplicação, com o microcontrolador STM32F103C8T6 que possui uma placa de

desenvolvimento chamada *Blue Pill*. que já possui os componentes básicos para o funcionamento do cortex, como osciladores, reguladores de tensão e conexões de *boot e reset*.

Definição do conversor de tensão

A tensão fornecida pela bateria do veículo é de 12V, incompatível com o nível de tensão do microcontrolador escolhido, que opera em 3,3V, então o uso de um regulador *buck* é necessário para converter a tensão de entrada para uma adequada ao projeto.

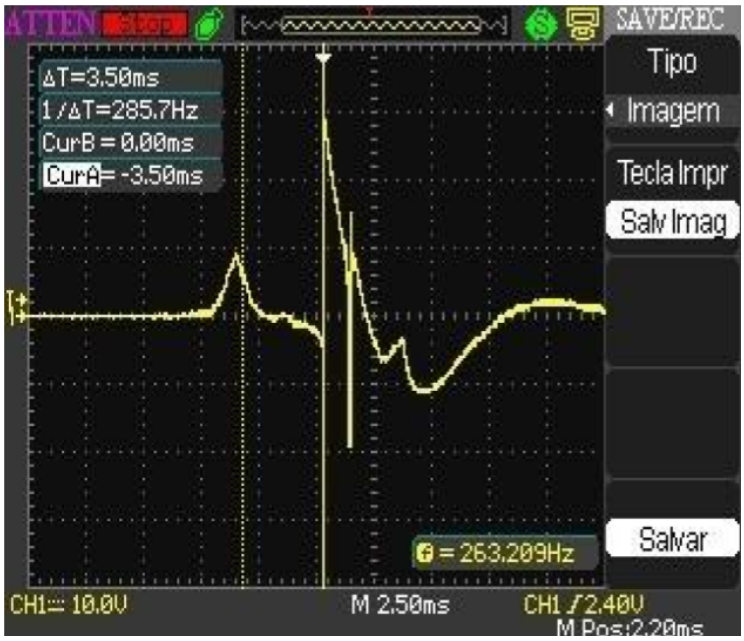
Para isso, foi escolhida a fonte chaveada LM2596, com 3A de capacidade e um pino de *feedback* para controle utilizando um circuito de amplificadores operacionais em cascata para comparação do sinal de saída e o sinal esperado, em que o nível de tensão na saída é controlado por um divisor de tensão formado entre os resistores R1 e R2, por vezes substituído por potenciômetros para um circuito de tensão de saída ajustável (WATSON, 2023).

Este já possui um *kit* de desenvolvimento, uma placa manufaturada, sendo assim mais facilmente integrado ao design da PCB, evitando a soldagem dos componentes menores necessários para a utilização.

Definição do acoplador óptico

Na Figura 4.1 é possível ver como se comporta o sinal produzido pela vela durante a rotação do motor, produzindo tensões de pico a pico próximos a 50V que podem destruir muitos dos componentes utilizados. O uso do acoplador óptico, ou optoacoplador, permite separar fisicamente o sinal do motor de outros componentes da placa. O CI 4N25 foi escolhido devido à alta facilidade de compra no mercado local e estar acima das margens mínimas de tensão exigidas para o projeto (ENRIQUE, 2022).

Figura 4.1: Forma de onda produzida pelo motor do Baja.



Fonte: O autor

Definição da memória

Em relação ao armazenamento dos dados aquistados, foi proposto um sistema de memória removível do tipo *flash*, que pode ser facilmente encontradas no formato de cartões SD. Estes possuem dois modos de comunicação amplamente utilizados o *SD ou SPI protocol*. Neste trabalho foi utilizado o protocolo SPI pela familiaridade com o protocolo (OPENLABPRO, 2023).

Definição do transmissor CAN

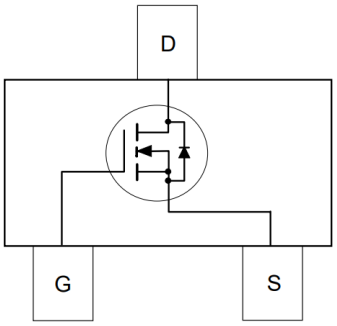
Para o transceptor CAN foi escolhido o modulo TJA1050 da NXP, pela compatibilidade com os níveis de tensão aplicadas ao projeto, além da facilidade comercial do módulo (NXP, 2023).

Definição do conversor nível de dado digital

Como alguns sensores, por exemplo, o indutivo e capacitivo, trabalham em níveis de tensão acima do permitido para o microcontrolador, foi necessária a elaboração de circuito para converter os níveis lógicos diferentes. Enquanto o sensor indutivo

trabalha com 5V, o microcontrolador é projetado para 3,3V. portanto foi proposto um conversor de nível lógico baseado em MOSFETs de canal N e resistores de pull-up. Foi escolhido o MOSFET BSS138, Figura 4.2, com montagem SOT-23 para essa finalidade.

Figura 4.2: Transistor de efeito de campo canal N BSS138.



Fonte: *datasheet* do CI[1]

Definição dos conectores

Os conectores selecionados foram os conectores de parafuso no bloco do terminal, que são amplamente utilizados em prototipagem, por possuírem baixo custo e boa conexão com os fios dos sensores.

4.2.2 Esquemático elétrico

Para acelerar a produção do primeiro protótipo muitos componentes foram utilizados com seus respectivos *kits* de desenvolvimento, mas em contrapartida abordaremos mais detalhadamente o funcionamento de cada componente utilizado, a fim de entender mais profundamente o funcionamento do circuito.

Esquemático elétrico do microcontrolador

O microcontrolador é o destino de toda entrada de dados do sistema, portanto será o componente com mais conexões na placa como visto na Figura 4.3. Para

[1]Disponível em https://cdn.sparkfun.com/datasheets/BreakoutBoards/BSS138.pdf. Acesso em: 23 de dezembro de 2022

facilitar o entendimento, foi dividido em 4 grupos:

- Alimentação:
  - Vbat;
  - 5V;
  - 3v3.
- Sinais Analógicos:
  - Três entradas de sinais analógicos.
- Sinais digitais:
  - RPM;
  - Velocidade.
- Protocolos de comunicação:
  - SPI;
  - CAN;
  - I2C;
  - Uart.

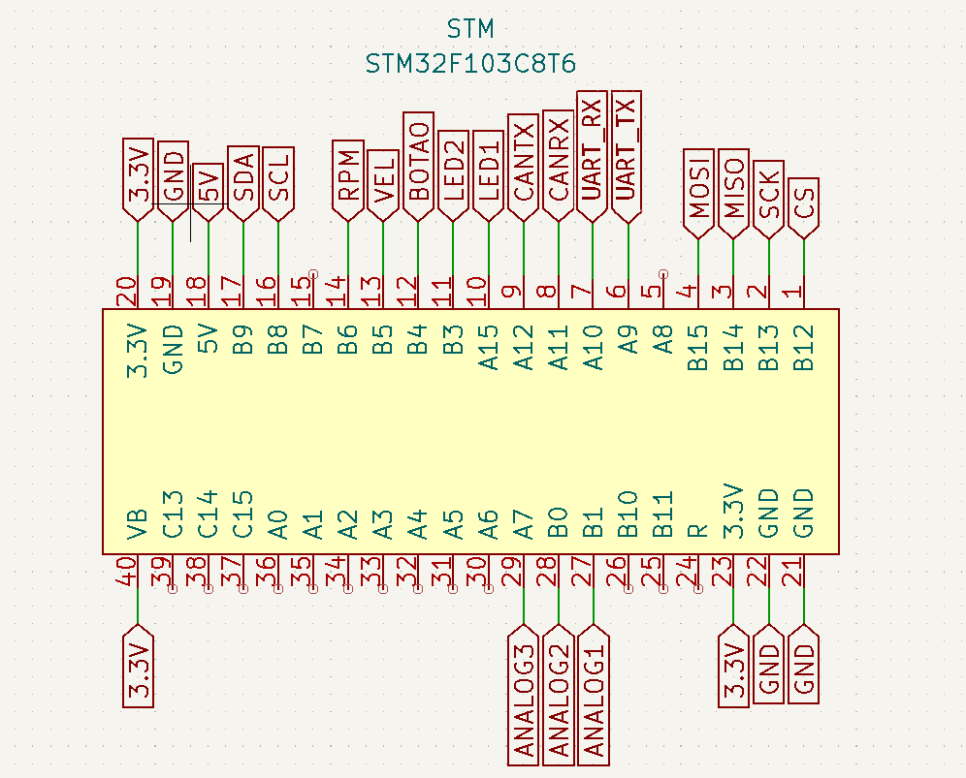
Esquemático elétrico do conversor de tensão

Na Figura 4.4 é possível ver o esquemático utilizado no projeto do conversor de tensão. O pino dois (OUT) é a saída do regulador e está conectado com três componentes, um diodo zener e dois capacitores, para a estabilização da tensão. Além disso, está conectado no pino de *feedback* em série com um potenciômetro que é usado para definir o nível de tensão da saída.

Como resultado o circuito encontrado para comercialização pode ser visto na Figura 4.5.

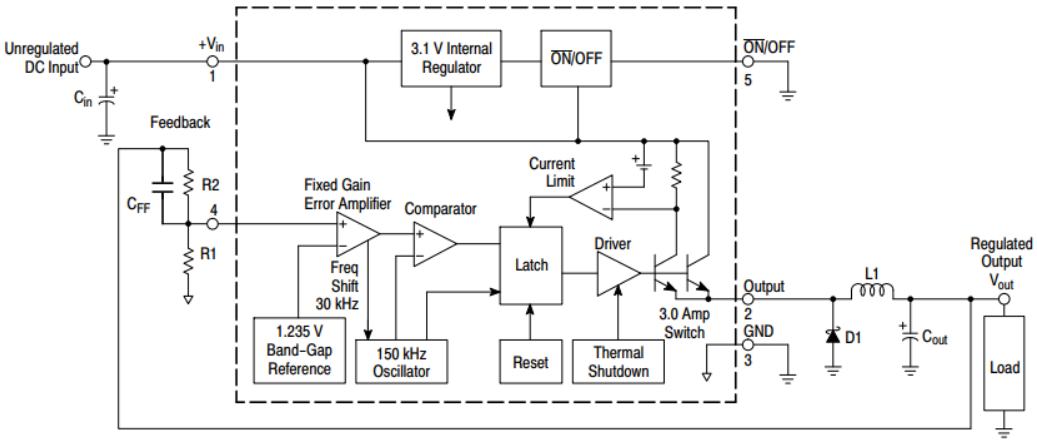
[2]Disponível em https://oshwlab.com/tiege/LM2596s-erDnJ53oK. Acesso em: 15 de abril de 2022

Figura 4.3: Conexões do kit de desenvolvimento do microcontrolador



Fonte: O autor

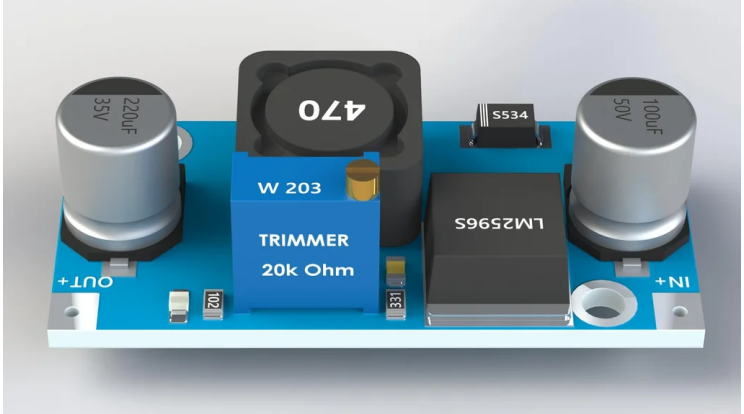
Figura 4.4: Esquemático elétrico do conversor de tensão *step-down* utilizando LM2596.



Fonte: Open Source Hardware Lab[2]

[3]Disponível em https://www.instructables.com/How-to-Use-DC-to-DC-Buck-Converter-LM2596/. Acesso em: 22 de abril de 2022

Figura 4.5: Placa de desenvolvimento do conversor de tensão *step-down* utilizando LM2596

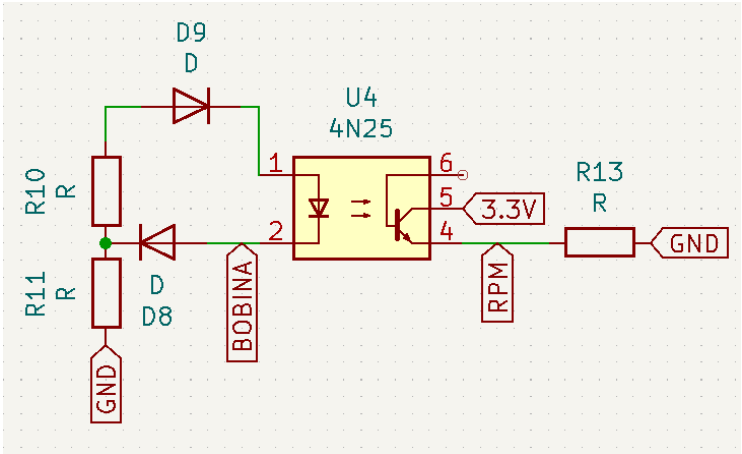


Fonte: Autodesk Instructables[3]

Esquemático elétrico do acoplador óptico

O esquemático elétrico para a aquisição do pulso da bobina é apresentado na Figura 4.6, onde o optoacoplador é usado na entrada do sinal na placa como um dispositivo de proteção que separa fisicamente o microcontrolador do pulso potencialmente danoso que vem do motor.

Figura 4.6: Circuito para aquisição do RPM do motor



Fonte: O autor

Os diodos D8 e D9 são usados para polarização, herdados do *design* inicial do projeto e podem ser retirados, permitindo apenas o ciclo negativo da onda do motor, contudo o ciclo positivo teria o mesmo efeito. Eles fazem com que quando o semi ciclo negativo chegue ao 4N25 o LED acenda e o transistor fotossensível passe a

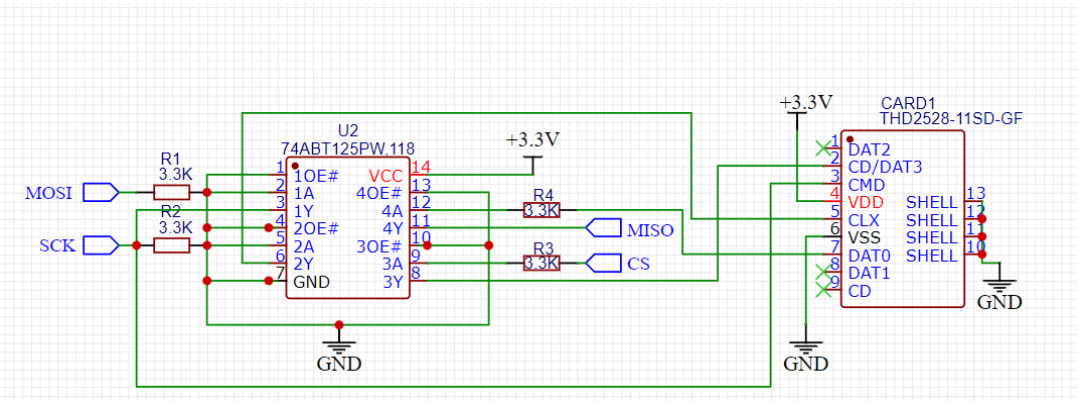
conduzir funcionando como uma chave eletrônica. O resistor R13 funciona como um resistor de *pull down*, todos os resistores possuem valor de $10k\Omega$.

Esquemático elétrico da memória

Para utilização da memória removível no formato SD card, é necessário a conexão do microcontrolador com o cartão para uso do protocolo SPI. De maneira geral, a conexão pode ser em geral a conexão direta entre o *socket* e o microcontrolador.

Para projetos rápidos pode ser utilizado um *kit* que já possui um circuito adequado. O circuito é apresentado na Figura 4.7. Na figura Nota-se que possui um conversor de nível lógico, que para nossa aplicação não é obrigatório, mas tão pouco problemática pois não interfere no funcionamento.

Figura 4.7: Conexão do cartao SD por protocolo SPI no *kit* de desenvolvimento.



Fonte: Acme Systems[4]

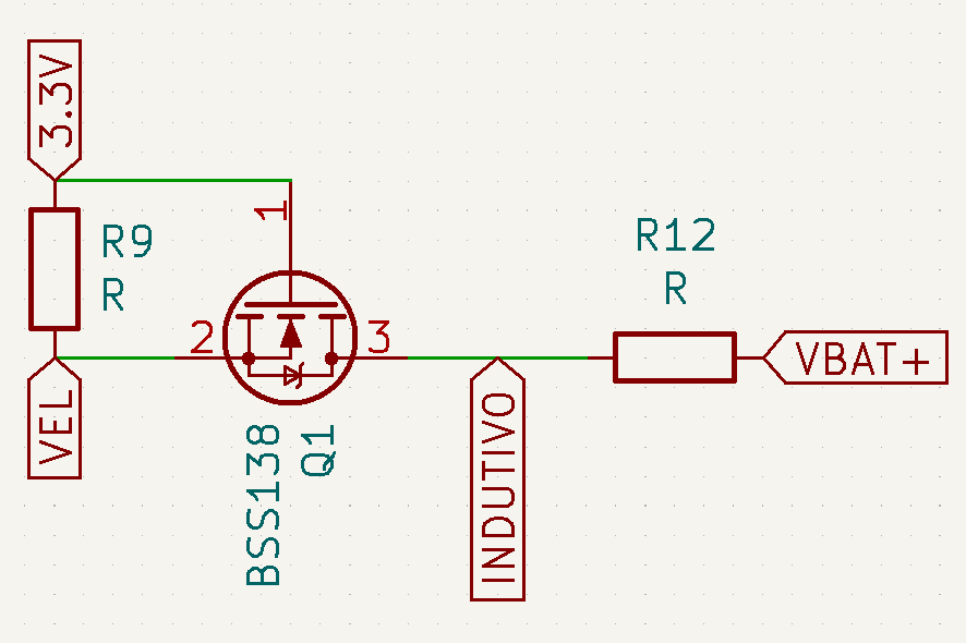
Esquemático elétrico do conversor de nível de dado digital

Para o conversor de tensão para comunicação entre sensores e micro com níveis de tensão diferentes, foi utilizado um circuito de comunicação bidirecional, neste caso usado apenas como unidirecional, com MOSFET de canal N e resistores de *pull-up*, utilizando alimentação da bateria 12V do veículo, VBAT. Este foi o caso para o sensor de velocidade, em que foi usado um sensor indutivo de nível lógico

[4]Disponível em https://www.acmesystems.it/pcb$_{m}icrosd.Acessoem$: $23demaiode2022$

12V que precisa se comunicar com um sistema operando em 3.3V. Os resistores R9 e R10 possuem um valor de $10k\Omega$.

Figura 4.8: Circuito do conversor de nível lógico de dado digital aplicado no sensor indutivo.



Fonte: O autor

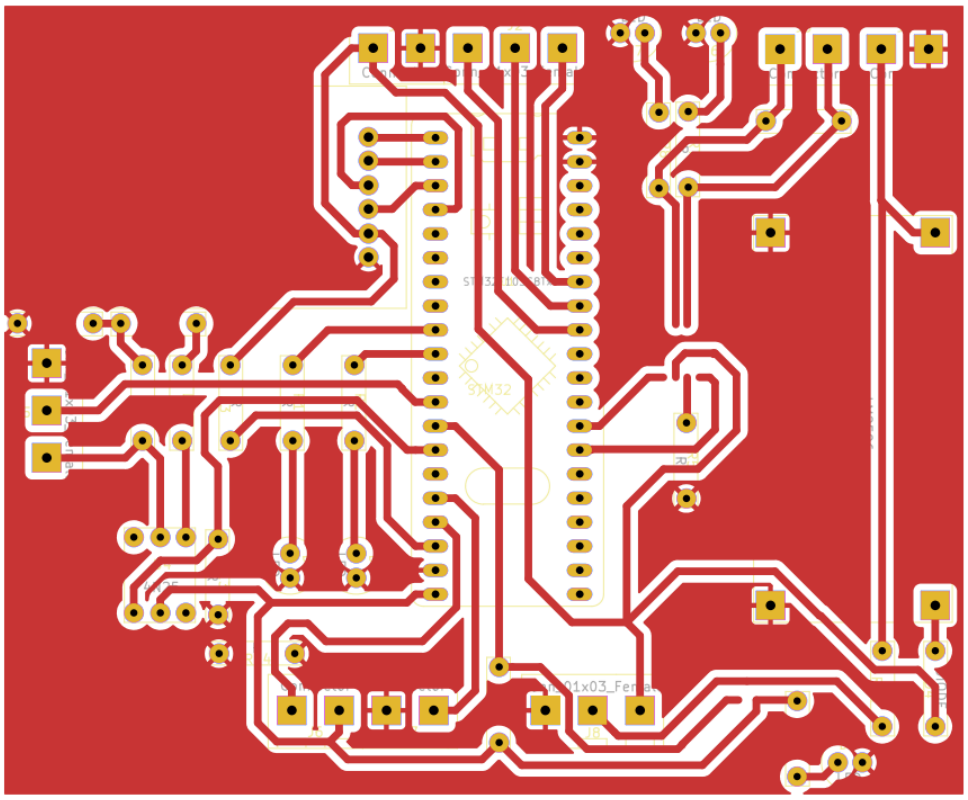
4.2.3 *Layout da PCB*

Roteamento e *design* da PCB

O SW utilizado para o *design* do *layout* foi o mesmo que no esquemático elétrico, o Kicad. Para a elaboração do *layout*, uma das primeiras decisões é o processo de fabricação que será usado, neste caso, como a finalidade é de prototipação, foram escolhidos componentes *Pin Through Hole (PTH)*, consequentemente usando apenas uma *layer* para o roteamento, visando a facilidade na fabricação. Os primeiros componentes posicionados foram o microcontrolador, o módulo do SD card e o regulador de tensão devido ao tamanho das placas. Com o posicionamento dos módulos maiores foram posicionados os resistores e CI presentes no esquemático, por fim os conectores, que por sua vez foram colocados nas bordas da placa para facilitar o uso durante os testes. No fim do processo obtivemos o *design* mostrado

na Figura 4.9 que será utilizada para conexão dos componentes eletrônicos.

Figura 4.9: Design da PCB para prototipação.



Fonte: O autor

4.3 Desenvolvimento do *software* embarcado

4.3.1 *Framework*

O *framework* escolhido para o desenvolvimento foi o Mbed devido à familiaridade com a plataforma e à compatibilidade com a arquitetura de *hardware* usada e a boa disponibilidade de ferramentas, como bibliotecas, documentações e aplicações de exemplo para os componentes utilizados.

4.3.2 *Setup*

Includes e defines

Para usar alguns dispositivos foram utilizado bibliotecas que facilitam a implementação. Os primeiros *includes* são relacionados ao *framework* utilizado, o mbed.h, cabeçalho de ferramentas da plataforma, além dos stdio.h e errno.h, relacionado a funções de entrada e saída, e macros para reconhecer erros na execução do código. Há outros três *includes* relacionados há bibliotecas de componentes, o SDBLockDevice.h e a FATFileSystem.h que auxiliam na comunicação com cartões SD e na criação de arquivos em memórias portáteis, e por fim o LSM6DS3.h que é usado para implementar o acelerômetro de seis eixos.

Quanto às pré-definições, foram setados 3 parâmetros, o tamanho do *buffer*, número de pacotes a serem salvo em uma única vez, e a frequência de aquisição; respectivamente sendo BUFFER.SIZE, SAVE.WHEN, SAMPLE.FREQ.

Variáveis Globais

Foram declaradas ferramentas de entrada e saída para aquisição dos dados, três do tipo AnalogIn que aquistarão dados analógicos, e duas interrupções para aquisição de dados de digitais. Outras variáveis foram declaradas para utilização de *hardwares* periféricos, o acelerômetro LSM6DS3, o SDBlockDevice e o FATFileSystem para utilização do cartão SD.

Criado duas ferramentas de *debug*, a primeira do tipo serial chamada PC para comunicação com o computador para acompanhar a execução do código e entender falhas, enquanto a PwmOut que será usada para gerar sinais PWM para testes de canais de entradas digitais durante o desenvolvimento do código. Outras ferramentas importantes são os pinos que cumprem a função de comunicação com o usuário por meio de LED, em que é possível acompanhar sem outras ferramentas a execução do código, como os LEDs de *logging* e *warning*.

Todos os dados são salvos em uma *struct* chamada packet_t que por sua vez é armazenada em um *buffer* circular que serve de memória temporária para os dados

da *struct* em seguida é salvo no cartão SD em forma arquivo iteradamente.

Outras variáveis são utilizadas para auxiliar na execução do código, sejam elas contadoras como a *buffer-counter* e *pulse-counter1*, *booleanas* para identificar o estado da máquina como *bool running* e a *bool StorageTrigger*, além de outras variáveis para marcação temporal como o *Timer t* e a *Ticker acq*.

Por fim outras funções ISR para interrupção foram criadas a void sampleISR(), freq-channel1-ISR(), freq-channel2-ISR() e toggle-logging();

4.3.3 Código principal

Inicialmente é criado uma variável do tipo *packet-t* para salvar os dados e uma variável ponteiro para o arquivo em que os dados serão salvos permanentemente. Em seguida é iniciado a função *begin* da biblioteca do LSM6DS3 em que é configurado a escala do giroscópio e do acelerômetro e seus *bandwidth*. Após isso é iniciado um *loop* que espera até que o arquivo no cartão de memória esteja pronto para ser escrito. É feito a vinculação do *InterruptIn* do pino PB4 com a função ISR do *toggle-logging*, essa função é responsável por trocar o estado inicial da *running* que por sua vez funciona como o gatilho para o progresso na execução, uma vez que o pino é colocado em nível lógico zero o código continua a ser executado, fazendo a abertura do arquivo pela primeira vez e associando as funções ISR para os canais de aquisição digitais.

A função sampleISR é responsável pela interrupção que ativa o StorageTrigger, ela está atrelada ticker acq que será ativado repetidamente em intervalos definidos pelo SAMPLE-FREQ. Uma vez que o StorageTrigger está em estado verdadeiro, todos os dados previstos no *struct packet-t* são salvos, é feito um *push* no *buffer* com os dados do acq-pck e somado ao contador para controlar o tamanho das iteraçoes, por fim é colocado StorageTrigger em falso novamente.

Em seguida é realizado uma verificação do estado do *buffer* em que caso ele se encontre cheio é feito o fclose(fp) e o *pop*, retirando a iteração e salvando os dados do *packet-t* presente nos *buffer* no arquivo

4.4 Desenvolvimento do *software* para processamento dos dados coletados

4.4.1 *Framework*

Para o desenvolvimento do *software* de pós-processamento dos dados, foi escolhida a linguagem *Python* pela facilidade de implementação da linguagem em ciência de dados.

Para ajudar construção do código foi utilizado o jupyter notebook e o google colabs para fazer uma execução parte a parte do programa. Foi escolhido o CSV de um dos melhores setups para ilustrar o desenvolvimento do código.

4.4.2 Ferramentas e bibliotecas

Foram instaladas bibliotecas para servir de ferramenta de suporte para o tratamento dos dados. A primeira ferramenta utilizada foi o Pandas, biblioteca focada em tratamento de arquivos CSV, outra importante biblioteca instalada foi o matplotlib que disponibiliza a elaboração de gráficos para visualização dos dados. Por fim, outras bibliotecas matemáticas foram usadas para facilitar os cálculos numéricos no tratamento dos dados, são numpy e a scipy.

4.4.3 Código principal

O primeiro passo para o tratamento dos dados é a retirada destes dados do CSV, os de maior interesse são os dados digitais referentes a rotação do motor e rotação da roda dianteira, f1 e f2 na Figura 4.10. Nesse teste, os demais dados não são representativos.

As duas colunas de dados digitais são atribuídas a dois *arrays* em que cada posição significa as quantidade de pulsos detectados em um intervalo de tempo de 5 milissegundos. Para transformar esses dados em números representativos das grandezas observadas foram necessárias a adoção de algumas medidas. A primeira

Figura 4.10: Exemplo do data frame encontrado nas aquisições dos dados.

	fxosaccx	fxosaccy	fxosaccz	lsmaccx	lsmaccy	lsmaccz	lsmangx	lsmangy	lsmangz	a0	a1	a2	a3	a4	a5	f1	f2	timestamp
0	9312	1052	-6788	0	0	0	0	0	0	16383	15749	25422	25470	23975	37205	0	0	5
1	6764	1520	-8448	0	0	0	0	0	0	9132	11716	19511	22530	24024	37207	0	0	10
2	-3704	-1240	-15160	0	0	0	0	0	0	12398	15071	20404	22966	23977	37171	0	0	15
3	588	-4340	-21296	0	0	0	0	0	0	12403	16436	21905	22313	24016	37289	0	0	20
4	-1916	-2800	-23696	0	0	0	0	0	0	9197	11211	20939	19486	23898	37190	1	0	25
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
9854	1792	-832	-15128	0	0	0	0	0	0	9116	11720	15084	16408	12361	16078	0	0	49275
9855	1632	-672	-14564	0	0	0	0	0	0	10425	11179	11215	9868	12430	16160	0	0	49280
9856	1700	-748	-14940	0	0	0	0	0	0	5332	9750	11240	12735	12275	16096	0	0	49285
9857	1580	-936	-14780	0	0	0	0	0	0	9164	9209	12739	14137	12260	16058	0	0	49290
9858	1580	-936	-14780	0	0	0	0	0	0	9164	9209	12739	14137	12260	16058	0	0	49290

9859 rows × 18 columns

Fonte: O autor

delas foi uma espécie de filtro, que atuou somando 10 intervalos e transformando em apenas um, a fim de aumentar o intervalo de tempo de cada amostra para 50 milissegundos, como mostrado na Figura 4.11.

Figura 4.11: Filtro para aumentar o intervalo de tempo das amostras.

```
for i in range(int(len(f1)/10)):
    rot_bruto.append(sum(f1[i*10:i*10+10]))
    vel_bruto.append(sum(f2[i*10:i*10+10]))
```

Fonte: O autor

Após isso os dados serão convertidos para as unidades de medidas utilizadas nas modelagens, sendo rotações por minuto para o conta giro do motor e quilômetros por hora para a velocidade do veículo, como mostrado na Figura 4.12.

Figura 4.12: Parâmetros e tratamento para transformar os dados em valores representativos.

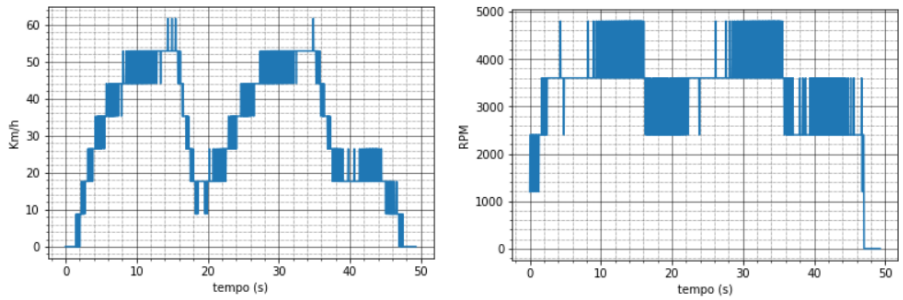
```
RPM_factor = 20 * 60
km_h_factor = 20 * 3.6
furos_disco =  15
raio_pneu = 0.584
pi = 3.1415

rot = [j * RPM_factor for j in rot_bruto]
vel = [i * raio_pneu * pi * km_h_factor / furos_disco for i in vel_bruto]
```

Fonte: O autor

Uma vez que já se tem os dados nas unidades de medida corretas, pode-se verifica-los através dos gráficos destes no tempo, como na Figura 4.13, é possível perceber

um certo ruído devido a forma de aquisição escolhida, mas em contrapartida o comportamento no tempo é facilmente identificado, mesmo não sendo possível fazer nenhuma análise qualitativa da relação.

Figura 4.13: Gráfico dos dados brutos de velocidade e RPM no tempo.



Fonte: O autor

Como explicado na fundamentação teórica, a curva mais importante para o *setup* da transmissão do *powertrain* é a curva de velocidade por rotação, e tentando verificar a correlação dos dados foi realizado o *plot* desses dados na Figura 4.14. No entanto, a propagação do ruído gerado quando os dois dados são combinados torna a visualização muito difícil. Dessa forma, não é possível realizar nenhuma análise de comportamento da relação de transmissão.

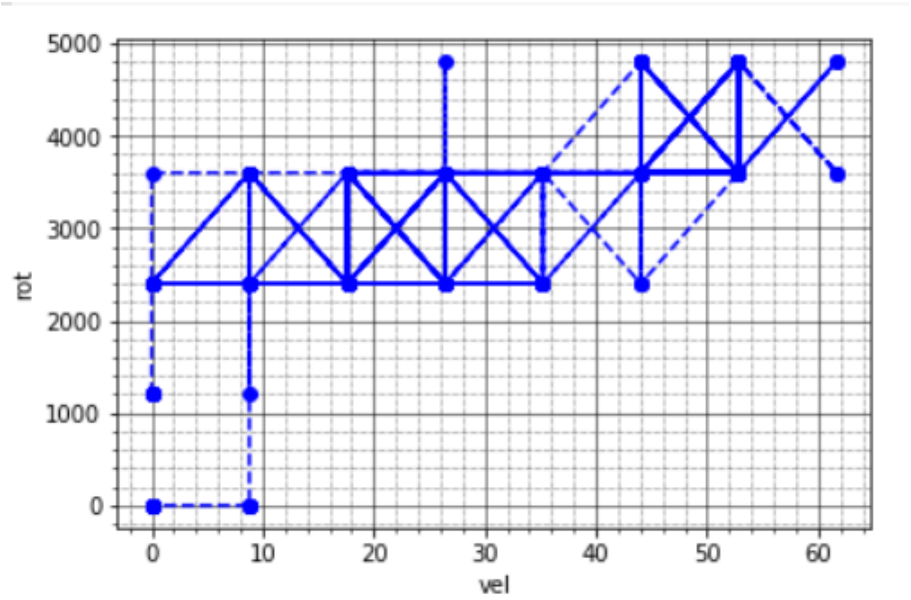
Portanto foi necessário o uso de um filtro *butterworth* de ordem quatro, em que para a sua construção foi utilizada a função *butter* para gerar os quocientes do filtro e a *filtfil* para fazer a filtragem dos dados, ambas as funções foram disponibilizadas pela biblioteca do *scipy*. A partir disso foi realizado um novo *plot* com os dados, como na Figura 4.15.

A partir dos novos dados de velocidade e rotação o gráfico de correlação entre eles podem ser novamente traçados.

Como pode ser visto na Figura 4.16, já é possível determinar a região de trabalho da CVT com clareza e utilizar isso para modelar qualitativamente e quantitativamente o desempenho da transmissão do carro.

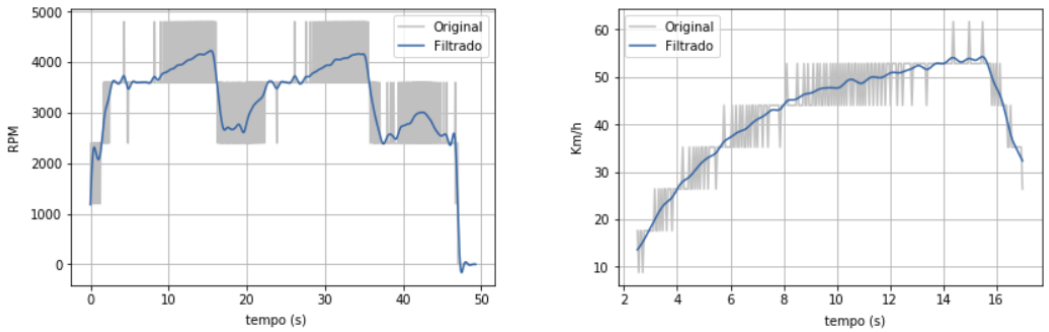
Outro dado aquisitado no testes foi a aceleração a partir do MEMS de seis eixos do sistema, o dados pode servir para *setup* de suspensão, mas nesse teste

Figura 4.14: Gráfico gerado pela relação dos dados brutos.



Fonte: O autor

Figura 4.15: Sinais de velocidade e RPM após o filtro BTW.



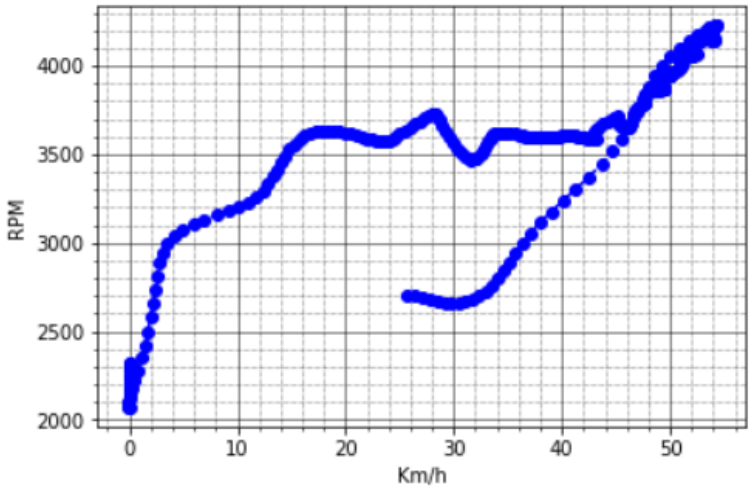
Fonte: O autor

será utilizado para validação dos dados aquisitados por meio da comparação entre os dados de aceleração do sensor e a derivada numérica do velocidade obtida. Os dados de aceleração no estado sem tratamento é visto na Figura 4.17.

Da mesma forma que os dados colhidos no formato de pulsos digitais necessitam de tratamento para poder ser utilizado. Seguindo as mesmas funções usadas para a análise dos dados anteriores, o filtro *butterworth* de quarto grau serviu como um crivo para fornecer dados de forma útil, como na Figura 4.18.

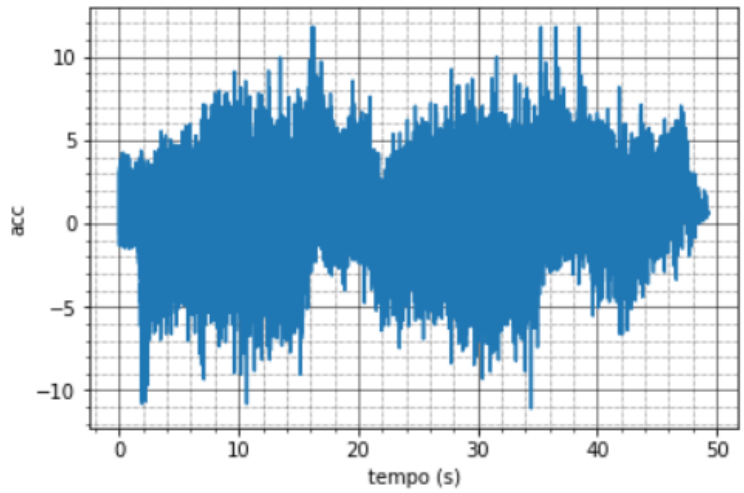
Comparando todos os dados até agora coletados conseguimos ver uma correlação

Figura 4.16: Cruzamento dos sinais filtrados de velocidade e rotação.



Fonte: O autor

Figura 4.17: Dados de aceleração recebidos do acelerômetro.



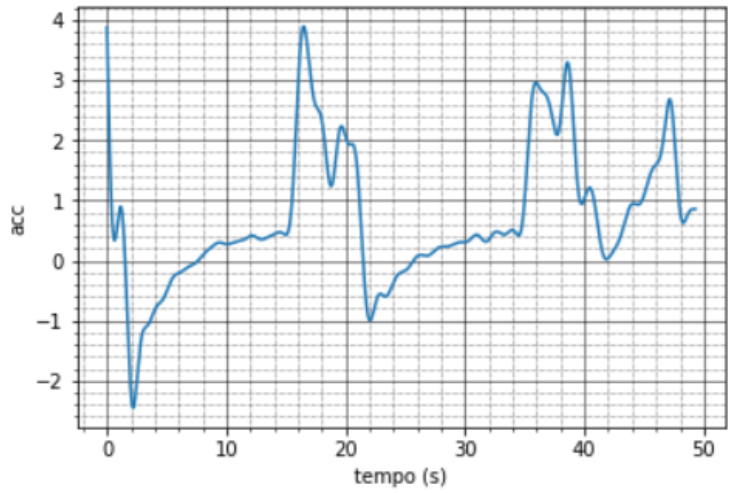
Fonte: O autor

que corrobora com a fidelidade dos dados, como mostrado na Figura 4.19.

Para mais uma verificação dos valores, foi realizado a derivada numérica dos dados de velocidade para ser comparada com os dados de aceleração aquisitadas pelo acelerômetro. A realização desse procedimento pode ser visto na Figura 4.20.

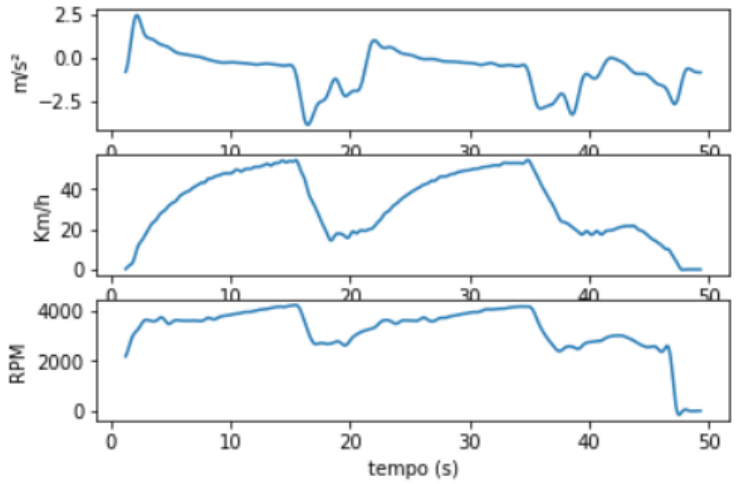
A partir disso é possível comparar as duas curvas de aceleração. Pode ser visto na Figura 4.21 que apesar dos dois dados terem sofridos com filtros agressivos que se utilizados de maneira errônea podem descaracterizar os dados, os dados são niti-

Figura 4.18: Dados tratados de aceleração recebidos do acelerômetro.



Fonte: O autor

Figura 4.19: Visualização dos dados de aceleração, velocidade e rotação do motor.



Fonte: O autor

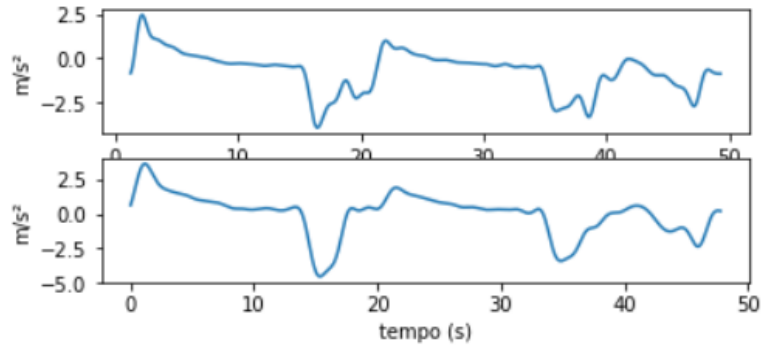
Figura 4.20: Código da derivação numérica dos dados de velocidade.

```
t2 = np.linspace(1, len(sig_vel), len(sig_vel))
t2 = [int(k) for k in t2]

accxn = [-1*(sig_vel[k-1]-sig_vel[k])/(0.05*3.66) for k in t2[:-1]]
```

Fonte: O autor

damente compatíveis e com poucas perdas de informações relevantes.

Figura 4.21: Código da derivação numérica dos dados de velocidade.



Fonte: O autor

Por fim, para facilitar o uso de pessoas leigas, a equipe construiu a partir desse código um executável para utilização do programa de tratamento de dados. Para isso, foi utilizado a biblioteca Python *tkinter*.

4.5 Resultados

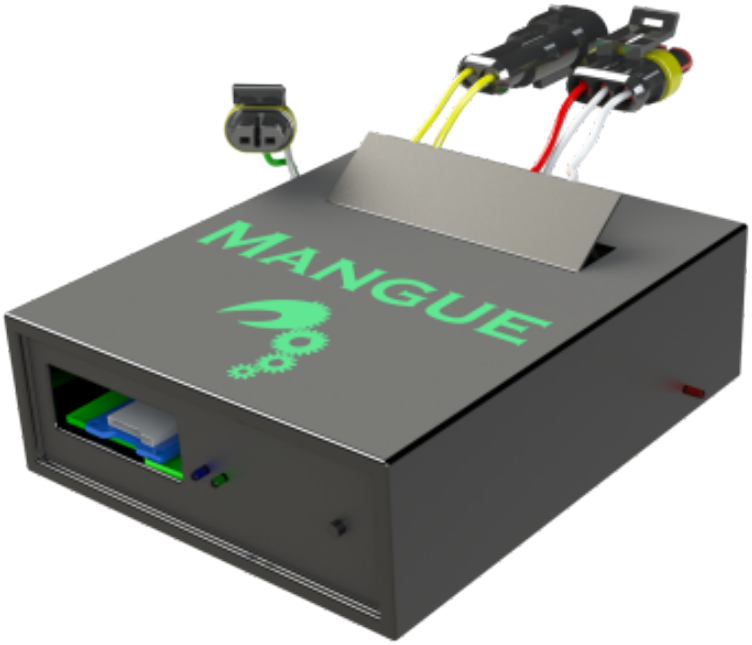
Nas próximas seções serão abordados os resultados obtidos pelo projeto.

4.5.1 Prototipagem

A partir do projeto finalizado, é iniciada a fase de elaboração de um produto piloto que servirá para o uso da equipe nas validações de componentes e aquisição de dados para modelagem de outros sistemas. Em CAD foi feito a montagem para servir de uma expectativa de produto final para testes clínicos. O resultado pode ser visto na Figura 4.22.

A placa de circuito impresso foi feita em uma empresa terceira, e a soldagem dos componentes por meio de uma estação de soldagem de componentes eletrônicos. Após a soldagem, foram realizados testes de continuidades em todas as trilhas e planos de terra para verificar a integridade das ligações e verificar curtos em algum *pad* da placa. Por fim, foi feita uma análise visual para identificar soldas frias

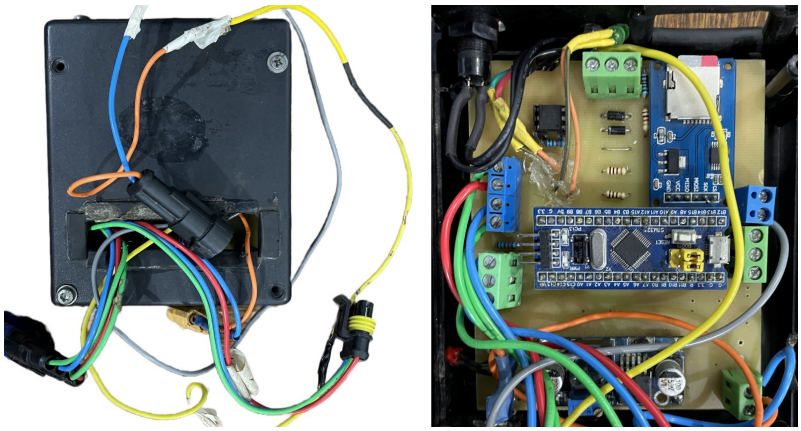
Figura 4.22: Expectativa de montagem do datalogger.



Fonte: O autor

ou problemas mecânicos no processo. Passado por todas essas fases, a placa foi aprovada e passou para fase de validação do sistema. Na Figura 4.23 é possível ver como ficou a montagem final do dispositivo.

Figura 4.23: Fotos do datalogger por fora e por dentro.

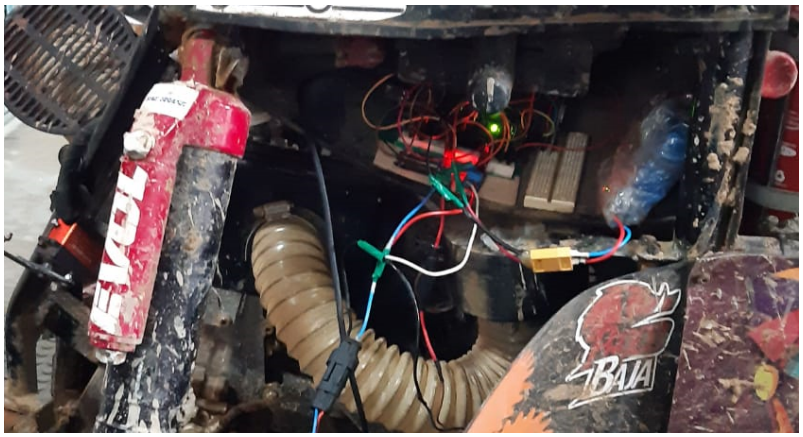


Fonte: O autor

4.5.2 Validações

Durante todo o desenvolvimento foram realizados testes para acompanhar o funcionamento do *software* e *hardware*, as primeiras validações e verificações foram realizadas ainda em *protoboards* na fase de estudo e propostas dos circuitos e algorítimos de aquisição para avaliar, como ser visto na Figura 4.24.

Figura 4.24: Validação de sistema ainda na protoboard.



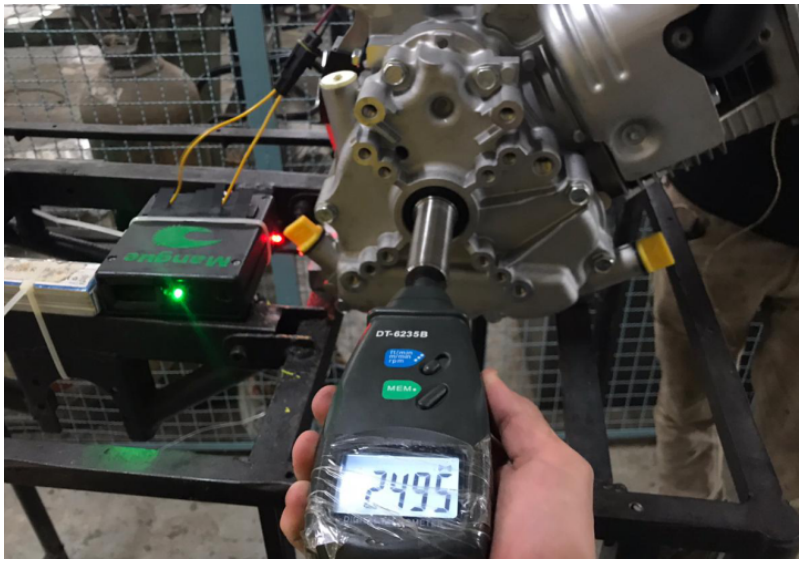
Fonte: O autor

Outras baterias de testes para verificação do sistema foram realizadas. Como exemplo, para a validação da aquisição da rotação do motor foram utilizados um tacômetro e a bancada de motor, como mostra a Figura 4.25, com a finalidade de cruzar os dados aferidos em cada um dos métodos e atestar a fidelidade do que foi salvo pelo *data logger* e a realidade. Por outro lado, para o acelerômetro foi utilizado um *hardware* com um microcontrolador de teste e uma mesa de alinhamento para verificar os cálculos de rolagem do veículo.

4.5.3 Testes

Visando o cumprimento do objetivo geral de projetar um equipamento eletrônico que ajude no desenvolvimento de projetos veiculares do tipo baja SAE, foram realizados cerca de 196 testes catalogados com *datalogger*. Um deles é possível ser visto na Figura 4.26.

Figura 4.25: Validação da aquisição do RPM do motor a partir de um tacômetro.



Fonte: O autor

Figura 4.26: Montagem e testes usando o datalogger

Fonte: O autor

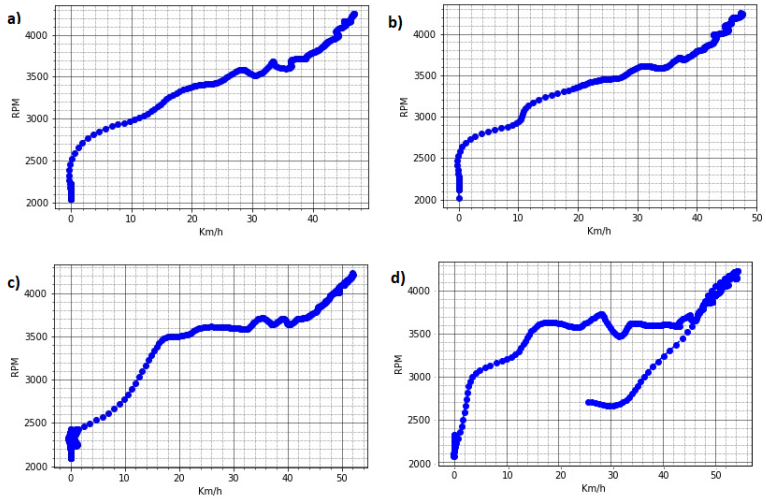
4.5.4 Resultados

O foco da equipe na temporada em questão foi na melhoria do sistema de *power-train*, então o uso do equipamento foi em testes de *setup* desse das relações fixas e variáveis das reduções.

Portanto, foram realizados mais de 190 testes de aceleração e velocidade para diferentes *setups*, em que cada um destes alimentava uma rotina de modelagem em matlab com a finalidade de chegar na melhor performance longitudinal possível,

como na figura 4.27.

Figura 4.27: Evolução da curva da relação entre RMP e velocidade.



Fonte: O autor

É possível observar uma clara evolução na transmissão, chegando mais rápido e mantendo o RPM do motor por mais tempo no pico de potência, que ocorre por volta das 3500 rotações por minuto. Além disso, conseguiu-se atingir velocidades superiores ao *setup* utilizado originalmente.

Com essa novas configurações de transmissão o veículo passou a ser muito mais competitivo, um fato que se mostrou verdade na competição seguinte, como pode ser visto na Figura 4.28. O carro subiu da vigésima segunda posição para a primeira posição em apenas um ano, sem nenhuma mudança de arquitetura relevante, apenas modificando o setup da redução do sistema de transmissão.

Figura 4.28: Carro da equipe Mangue Baja é primeiro colocado na competição Baja SAE Brasil 2020 em provas dinâmicas

#	Equipe	Aceleração	Retomada	Tração	Manobrabilidade	Suspensão	Super Prime	Total
27	Mangue Baja 2 Multimoldes Universidade Federal de Pernambuco	39.91	37.82	17.17	30.98	70	30	225.88
11	CEFAST Baja SAE Centro Federal de Educação Tecnológica de Minas Gerais	37.75	36.31	27.44	33.93	68.14	15	218.57
18	Equipe Car-Kará Baja Universidade Federal do Rio Grande do Norte	29.89	34.86	29.55	42.05	68.91	10	215.26
29	FEI Baja 1 Centro Universitário FEI	36.95	35.71	20.25	37.62	69.44	15	214.97
34	Baja de Galpão - UNISC Universidade de Santa Cruz do Sul	35.41	32.86	20.72	38.36	68.65	15	211
49	UNICAMP Baja SAE Universidade Estadual de Campinas	31.09	27.75	27.35	40.57	55.28	12	194.04

Fonte: Baja SAE[5]

[5]Disponível em https://resultados.bajasaebrasil.online/. Acesso em: 22 de abril de 2022

4.6 Próximos Passos

Como abordamos na fase inicial do projeto, há outros subsistemas que possuem um alto impacto na performance do carro, como a suspensão e o sistemas de freio. Portanto, o projeto do *datalogger* foi pensado em aplicações que necessitem dados de outras origens. Na Figura 4.29, por exemplo, temos duas possibilidades de aquisição de dados, a pressão na linha de freio e o curso da suspensão, ambos fatores importantes na modelagem de seus sistemas.

Figura 4.29: Sensores posicionados para aquisitar dados analógicos.

Fonte: O autor

Além disso, melhorias possíveis podem ser a criação de uma interface gráfica mais amigável para visualização dos dados, além da possibilidade do recebimento de dados em tempo real por radiofrequência que aumentaria bastante a produtividade dos testes visto que o equipamento não precisaria ser desmontado para a transferência dos dados.

Capítulo 5

CONSIDERAÇÕES FINAIS

Com tudo que foi apresentado, foi possível perceber a importância da eletrônica no auxílio de projetos automotivos, fornecendo dados reais do protótipo para servir de base para modelagem dos subsistemas. No caso concreto apresentado, tivemos uma melhoria de performance em aceleração de cerca de 15%, suficiente para chegar como melhor equipe da competição no quesito de performance dinâmica, com baixíssimo custo para obter os resultados, visto que foi gasto apenas para a montagem do equipamento, que em contraste com qualquer outro projeto mecânico que pode custar milhares de Reais, o projeto teve um custo total de 200 Reais.

Foram percebidas diversas dificuldade durante o desenvolvimento do projeto, sobretudo em questões relacionadas a infraestrutura para teste e confecção de placas de circuito impresso e integração com o veículo que por sua aplicação extrema em *off road* representa um nível altíssimo de desafio. Outra questão importante foi o tempo necessário para desenvolver o trabalho, que durou quase 2 anos, sendo preciso 2 meses apenas para fazer testes de aquisição de dados de *powertrain*.

Como sugestões para trabalhos futuros, destaca-se a criação de uma interface gráfica mais amigável para a visualização dos dados coletados, o que facilitaria a interpretação e análise dos resultados. Além disso, a possibilidade de receber dados em tempo real por radiofrequência poderia significativamente aumentar a produtividade dos testes, permitindo uma monitorização mais eficiente do desempenho do protótipo. Essas melhorias potenciais poderiam ampliar ainda mais a aplicabilidade

e utilidade do projeto, contribuindo para o avanço da área de projetos automotivos e eletrônica embarcada.

Referências

ACEA. *Employment trends in the EU automotive sector*. 2022. Disponível em: ⟨https://www.acea.auto/figure/employment-trends-in-eu-automotive-sector/⟩. Acesso em: 17 jan. 2023.

BAUTISTA, R. F. *About KiCad*. 2023. Disponível em: ⟨https://www.kicad.org/about/kicad/⟩. Acesso em: 11 jan. 2023.

BHOSALE, A. J. *Introduction to Automotive Electrical Systems*. 2023. Disponível em: ⟨https://www.gcoeara.ac.in/learning_material/auto/Unit%201-AEE.pdf⟩. Acesso em: 29 jan. 2023).

BLS, U. B. O. L. S. *Automotive Industry: Employment, Earnings, and Hours*. 2022. Disponível em: ⟨https://www.bls.gov/iag/tgs/iagauto.htm⟩. Acesso em: 11 jan. 2023.

CHARRAS, J.-P.; TAPPERO, F.; STAMBAUGH, W. *KiCad Complete Reference Manual Full Color Version*: Full color version. [S.l.]: 12th Media Services, 2018. 508 p. ISBN 9781680921281.

CIRCUITS, S. *Design for Manufacturing Handbook*. [S.l.]: Sierra Circuits, 2022.

CIRCUITS, S. *PCB Material Design Guide*. [S.l.]: Sierra Circuits, 2022.

CONRAD, M. *Understanding Automotive E/E Systems*. 2023. Disponível em: ⟨https://www.vdiconference.com/event/grundlagen-ee-systeme/⟩. Acesso em: 26 fev. 2023.

CROLLA, D. *Automotive Engineering: Powertrain, Chassis System and Vehicle Body*. [S.l.]: Elsevier Science Technology, 2009. ISBN 9781856175777.

DALMARIS, P. *KiCad 6 Like A Pro - Fundamentals and Projects*. [S.l.]: Elektor International Media, 2022.

DENTON, T. *Automobile Electrical and Electronic Systems*. [S.l.]: Routledge, 2017. ISBN 0415725771.

ELECTRONICSHUB. *Basics of Microcontrollers – History, Structure and Applications*. 2017. Disponível em: ⟨https://www.electronicshub.org/microcontrollers-basics-structure-applications/⟩. Acesso em: 25 fev. 2023.

ENRIQUE. *4N25 Phototransistor Optocoupler IC*. 2022. Disponível em: ⟨https://microcontrollerslab.com/4n25-optocoupler-pinout-features-examples-datasheet/⟩. Acesso em: 13 jan. 2023.

HERLT, A. *Smartphones on wheels: New rules for automotive-product development*. 2022. Disponível em: ⟨https://www.mckinsey.com/industries/automotive-and-assembly/our-insights/smartphones-on-wheels-new-rules-for-automotive-product-development⟩. Acesso em: 28 jan. 2023.

IEA. *Trends and developments in electric vehicle markets*. 2022. Disponível em: ⟨https://www.iea.org/reports/global-ev-outlook-2021/trends-and-developments-in-electric-vehicle-markets⟩. Acesso em: 28 jan. 2023.

INSTITUTE, P. M. *A Guide to the Project Management Body of Knowledge (PMBOK® Guide)*. [S.l.]: Project Management Institute, 2017. ISBN 9781628251845.

KRIGER, D. *O QUE SÃO FRAMEWORKS E POR QUE SÃO IMPORTANTES PARA OS DEVS*. 2022. Disponível em: ⟨https://kenzie.com.br/blog/framework/⟩. Acesso em: 10 fev. 2023.

LEENS, F. An introduction to I2C and SPI protocols. *IEEE Instrumentation &amp Measurement Magazine*, Institute of Electrical and Electronics Engineers (IEEE), v. 12, n. 1, p. 8–13, feb 2009.

MARTIN, T. *Designer's Guide to the Cortex-M Microcontrollers*. [S.l.]: Elsevier Science Technology, 2016. 490 p. ISBN 9780081006290.

MBED, A. *An introduction to Arm Mbed OS 6*. 2023. Disponível em: ⟨https://os.mbed.com/docs/mbed-os/v6.16/introduction/index.html⟩. Acesso em: 12 jan. 2023.

MILLIKEN, W. F. *Race car vehicle dynamics*. [S.l.]: SAE International, 1995. 890 p. ISBN 1560915269.

NAVET, N.; SIMONOT-LION, F. *Automotive Embedded Systems Handbook*. [S.l.]: Taylor Francis Group, 2017. 470 p. ISBN 9781315222301.

NXP. *CAN Transceivers*. 2023. Disponível em: ⟨https://www.nxp.com/products/interfaces/can-transceivers:MC_53485⟩. Acesso em: 25 jan. 2023.

ODUNLADE, E. *TOP 10 POPULAR MICROCONTROLLERS AMONG MAKERS*. 2020. Disponível em: ⟨https://www.electronics-lab.com/top-10-popular-microcontrollers-among-makers/⟩. Acesso em: 01 fev. 2023.

OPENLABPRO. *Interfacing Microcontrollers with SD Card*. 2023. Disponível em: ⟨https://openlabpro.com/guide/interfacing-microcontrollers-with-sd-card/⟩. Acesso em: 15 jan. 2023.

ORG jupyter. *About Us Project Jupyter's origins and governance*. 2023. Disponível em: ⟨https://jupyter.org/about⟩. Acesso em: 12 jan. 2023.

PACEJKA, H. B. *Tire and Vehicle Dynamics*. [S.l.]: Butterworth-Heinemann, 2012. ISBN 0080970168.

PETERSON, Z. *IPC Classes and Complying with IPC Standards for PCB Design*. 2022. Disponível em: ⟨https://resources.altium.com/p/complying-with-ipc-standards-for-pcb-design⟩. Acesso em: 26 fev. 2023.

RAE, J. B. *Automotive Industry*. 2023. Disponível em: ⟨https://www.britannica.com/technology/automotive-industry⟩. Acesso em: 12 jan. 2023.

RITCHEY, L. W.; ZASIO, J. *Right the First Time*: a practical handbook on high speed pcb and system design. [S.l.]: Speeding Edge, 2003. 288 p. ISBN 9780974193601.

ROCHA, E. V. M. *Métodos e sistemas de gestão de produção de veículos sob a ótica das tendências de produção sob encomenda e de customização em massa*. Dissertação (dissertação) — Pontifícia Universidade Católica, Rio de Janeiro, RJ, 2005.

SAE. *Baja Nacional*. 2023. Disponível em: ⟨https://saebrasil.org.br/programas-estudantis/baja-sae-brasil/⟩. Acesso em: 25 fev. 2023.

STONE, R. *Automotive engineering fundamentals*. [S.l.]: SAE International, 2004. ISBN 0768009871.

TOULSON, R.; WILMSHURST, T. *Fast and Effective Embedded Systems Design Applying the ARM Mbed*: Applying the arm mbed. [S.l.]: Elsevier Science Technology Books, 2016. 510 p. ISBN 9780081008805.

UNIT-V. *Embedded Communications Protocols*. 2023. Disponível em: ⟨http://www.gpcet.ac.in/wp-content/uploads/2018/08/UNIT-V.pdf⟩. Acesso em: 18 fev. 2023.

VANDERPLAS, J. *Python Data Science Handbook*: Essential tools for working with data. [S.l.]: O'Reilly Media, 2022. 548 p. ISBN 9781491912058.

WATSON, D. *LM2596 Buck Converter Datasheet, Pinout, Features, Applications*. 2023. Disponível em: ⟨https://www.theengineeringprojects.com/2020/09/lm2596-buck-converter-datasheet-pinout-features-applications.html⟩. Acesso em: 13 jan. 2023.

WILLIAMS, L. *Embedded Systems Tutorial: What is, History Characteristics*. 2023. Disponível em: ⟨https://www.guru99.com/embedded-systems-tutorial.html⟩. Acesso em: 18 fev. 2023.

ZHANG, Y. *Automotive Power Transmission Systems*. [S.l.]: Wiley Online Library, 2018. ISBN 9781118964897.

Apêndice A

Apêndice: Código do software embarcado

```
#include "mbed.h"
#include <stdio.h>
#include <errno.h>
#include "SDBlockDevice.h"
#include "FATFileSystem.h"
#include "LSM6DS3.h"

#define BUFFER_SIZE 200
#define SAVE_WHEN 50
#define SAMPLE_FREQ 200

/* Debug */
PwmOut signal.wave(PB_3);

/* I/O */
Serial pc(PA_2, PA_3);
LSM6DS3 LSM6DS3(PB_9, PB_8);
SDBlockDevice   sd(PB_15, PB_14, PB_13, PB_12);
FATFileSystem   fileSystem("sd");
DigitalOut warning(PA_15);
DigitalOut logging(PA_12);
```

```
22 InterruptIn start(PB.4,PullUp);
23 InterruptIn freq.chan1(PB.5,PullUp);
24 InterruptIn freq.chan2(PB.6,PullUp);
25 AnalogIn pot0(PB.1),
26          pot1(PB.0),
27          pot2(PA.7);
28
29 /* Data structure */
30 typedef struct
31 {
32     int16.t acclsmx;
33     int16.t acclsmy;
34     int16.t acclsmz;
35     int16.t anglsmx;
36     int16.t anglsmy;
37     int16.t anglsmz;
38     uint16.t analog0;
39     uint16.t analog1;
40     uint16.t analog2;
41     uint16.t pulses.chan1;
42     uint16.t pulses.chan2;
43     uint32.t time.stamp;
44 } packet.t;
45
46
47 Timer t;
48 Ticker acq;
49 CircularBuffer<packet.t, BUFFER.SIZE> buffer;
50 int buffer.counter = 0;
51 int err;
52 bool running = false;
53 bool StorageTrigger = false;
54 uint16.t pulse.counter1 = 0,
55          pulse.counter2 = 0,
56          acc.addr = 0;
```

```
57
58 void sampleISR();
59 uint32_t count_files_in_sd(const char *fsrc);
60 void freq_channel1_ISR();
61 void freq_channel2_ISR();
62 void toggle_logging();
63
64 int main()
65 {
66     pc.printf("\r\nDebug 1\r\n");
67     logging = 0;
68     int num_parts = 0,
69         num_files = 0,
70         svd_pck = 0;
71     char name_dir[12];
72     char name_file[20];
73     FILE* fp;
74     packet_t temp;
75     signal.wave.period_us(50);
76     signal.wave.write(0.5f);
77
78
79     /* Initialize accelerometer */
80     acc_addr = LSM6DS3.begin(LSM6DS3_G_SCALE_245DPS, LSM6DS3_A_SCALE_2G
       , \
81                              LSM6DS3_G_ODR_208, LSM6DS3_A_ODR_208);
82
83     /* Wait for SD mount */
84     do
85     {
86         /* Try to mount the filesystem */
87         pc.printf("Mounting the filesystem... ");
88         fflush(stdout);
89
90         err = fileSystem.mount(&sd);
```

```
91          pc.printf("%s\n", (err ? "Fail :(" : "OK"));
92          if (err)
93          {
94
95              pc.printf("No filesystem found, formatting... ");
96              fflush(stdout);
97              err = fileSystem.reformat(&sd);
98              pc.printf("%s\n", (err ? "Fail :(" : "OK"));
99              if (err)
100             {
101                 error("error: %s (%d)\n", strerror(-err), err);
102             }
103         }
104     }while(err);
105
106     pc.printf("\r\nDebug 2\r\n");
107
108     pc.printf("\r\nDebug 3\r\n");
109
110     num.files = count.files.in.sd("/sd");
111     sprintf(name.dir, "%s%d", "/sd/RUN", num.files + 1);
112
113     pc.printf("\r\nDebug 4\r\n");
114     pc.printf("\r\nNum.files = %d\r\n", num.files);
115
116     start.fall(&toggle.logging);
117
118     while(!running)
119     {
120         warning = 1;
121         pc.printf("\r\nrunning=%d\r\n", running);
122     }
123
124     /* Create RUN directory */
125     mkdir(name.dir, 0777);
```

```
126     warning = 0;
127     //sprintf(name.file , "%s%s%d", name.dir , "/part", num.parts++);
128     sprintf(name.file , "%s%s%d", name.dir , "/part", num.parts+1);
129     fp = fopen(name.file , "a");
130     freq.chan1.fall(&freq.channel1.ISR);
131     freq.chan2.fall(&freq.channel2.ISR);
132     acq.attach(&sampleISR , 1.0/SAMPLE.FREQ);
133
134     while(running)
135     {
136         if(StorageTrigger)
137         {
138             packet.t acq.pck;
139             static uint16.t last.acq = t.read.ms();
140
141             /* Store LSM6DS3 data if it's connected */
142             if (acc.addr != 0)
143             {
144                 LSM6DS3.readAccel();
145                 LSM6DS3.readGyro();
146
147                 acq.pck.acclsmx = LSM6DS3.ax.raw;
148                 acq.pck.acclsmy = LSM6DS3.ay.raw;
149                 acq.pck.acclsmz = LSM6DS3.az.raw;
150                 acq.pck.anglsmx = LSM6DS3.gx.raw;
151                 acq.pck.anglsmy = LSM6DS3.gy.raw;
152                 acq.pck.anglsmz = LSM6DS3.gz.raw;
153             }
154             else
155             {
156                 acq.pck.acclsmx = 0;
157                 acq.pck.acclsmy = 0;
158                 acq.pck.acclsmz = 0;
159                 acq.pck.anglsmx = 0;
160                 acq.pck.anglsmy = 0;
```

```
161                 acq.pck.anglsmz = 0;
162             }
163
164             acq.pck.analog0 = pot0.read.u16();
165             acq.pck.analog1 = pot1.read.u16();
166             acq.pck.analog2 = pot2.read.u16();
167             acq.pck.pulses.chan1 = pulse.counter1;
168             acq.pck.pulses.chan2 = pulse.counter2;
169             acq.pck.time.stamp = t.read.ms();
170
171             pulse.counter1= 0;
172             pulse.counter2= 0;
173             buffer.push(acq.pck);
174             buffer.counter++;
175
176             StorageTrigger = false;
177         }
178
179         if(buffer.full())
180         {
181             fclose(fp);
182             warning = 1;
183             pc.putc('X');
184         }
185         else if(!buffer.empty())
186         {
187             pc.putc('G');
188
189             /* Remove packet from buffer and writes it to file */
190             buffer.pop(temp);
191             buffer.counter--;
192             fwrite((void *)&temp, sizeof(packet.t), 1, fp);
193             svd.pck++;
194
195
```

```
196                 if(svd.pck == SAVE.WHEN)
197                 {
198                     svd.pck = 0;
199                 }
200             }
201 
202             /* Software debounce for start button */
203             if((t.read.ms() > 10) && (t.read.ms() < 1000))
204                 start.fall(toggle.logging);
205         }
206 
207 
208         fclose(fp);
209         logging = 0;
210         NVIC.SystemReset();
211         return 0;
212 }
213 
214 void sampleISR()
215 {
216         StorageTrigger = true;
217 }
218 
219 uint32.t count.files.in.sd(const char *fsrc)
220 {
221         DIR *d = opendir(fsrc);
222         struct dirent *p;
223         uint32.t counter = 0;
224 
225         while ((p = readdir(d)) != NULL)
226         {
227             if(strcmp(p->d.name, ".Trash-1000"))
228                 counter++;
229         }
230         closedir(d);
```

```
231     return counter;
232 }
233
234 void freq.channel1.ISR()
235 {
236     pulse.counter1++;
237 }
238
239 void freq.channel2.ISR()
240 {
241     pulse.counter2++;
242 }
243
244 void toggle.logging()
245 {
246     running = !running;
247     start.fall(NULL);
248 }
```

Apêndice B

Apêndice: Código Python

```
from tkinter import *
from tkinter import filedialog as fd
from tkinter import messagebox as msb
from ctypes import *
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from scipy import signal
from glob import glob
import os


window = Tk()
window.title("iLOGGER Mangue Baja")
window.geometry('450x140')
photo = PhotoImage(file = 'icone_window.png')
window.wm_iconphoto(False, photo)
```

```
def grafico():

    if txt5.get() and ":/" in txt5.get():
        df = pd.read_csv(txt5.get())
        tabela1 = df.set_index('f1')
        tabela2 = df.set_index('f2')
        f1 = tabela1.index.values
        f2 = tabela2.index.values

        vel_bruto = []
        rot_bruto = []
        rel = []
        cont = 0
        for i in range(int(len(f1) / 10)):
            rot_bruto.append(sum(f2[i * 10:i * 10 + 10]))
            vel_bruto.append(sum(f1[i * 10:i * 10 + 10]))

        raio = 0.29  # em mil metros
        if txt4.get().isnumeric():
            furos_do_disco_de_freio = int(txt4.get())
            msb.showinfo("Disco de Freio","Usando quantidade de furos no
            txt4.delete(0, "end")
        else:
            furos_do_disco_de_freio = 24
            msb.showerror("ERRO", "Quantidade inv lida , configurando par
            txt4.delete(0, "end")
```

```
rot = [j * 20 * 60 for j in rot_bruto]
vel = [i * 2 * raio * 3.1415 * 20 * 3.6 / furos_do_disco_de_freio
# vel = [i * 0.584 * 3.1415 * 20 * 3.6 / 24 for i in vel_bruto]

t = np.linspace(0, 0.05 * len(f1) / 10, int(len(f1) / 10))
data = {
    'RPM': rot,
    'vel': vel}
csv = pd.DataFrame(data, columns=['RPM', 'vel'])
csv.to_csv('AV_data.csv')

b, a = signal.butter(4, 0.10, analog=False)
impulse = np.zeros(1000)
impulse[500] = 1
imp_ff = signal.filtfilt(b, a, impulse)
imp_lf = signal.lfilter(b, a, signal.lfilter(b, a, impulse))

# PLOT_ROTA O
sig_rot = signal.filtfilt(b, a, rot)
plt.plot(t, rot, color='silver', label='Original')
plt.plot(t, sig_rot, color='#3465a4', label='Filtrado')
plt.grid(True, which='both')
plt.legend(loc="best")
plt.xlabel('tempo (s)')
plt.ylabel('RPM')
plt.title("Rota   o do Motor")
plt.show()

c, d = signal.butter(4, 0.15, analog=False)
```

```
impulse = np.zeros(1000)
impulse[500] = 1
imp_ff = signal.filtfilt(c, d, impulse)
imp_lf = signal.lfilter(c, d, signal.lfilter(c, d, impulse))

# PLOT_VELOCIDADE
sig_vel = signal.filtfilt(c, d, vel)
plt.plot(t[50:340], vel[50:340], color='silver', label='Original'
plt.plot(t[50:340], sig_vel[50:340], color='#3465a4', label='Filt
plt.grid(True, which='both')
plt.legend(loc="best")
plt.xlabel('tempo (s)')
plt.ylabel('Km/h')
plt.title("Velocidade")
plt.show()

# PLOT_RPM_VELOCIDADE
plt.plot(sig_vel[5:350], sig_rot[5:350], marker='o', linestyle='-
plt.xlabel('Km/h')
plt.ylabel('RPM')
plt.minorticks_on()
plt.grid(which='major', linestyle='-', linewidth='0.5', color='bl
plt.grid(which='minor', linestyle=':', linewidth='0.5', color='bl
plt.title("Rota   o por Velocidade")
# plt.savefig('Vel x Rot.jpeg')
plt.show()

data = {
    'RPM': sig_rot,
```

```
            'vel': sig_vel}
        csv = pd.DataFrame(data, columns=['RPM', 'vel'])
        csv.to_csv('AV_filt.csv')

        tabela3 = df.set_index('lsmaccx')
        tabela4 = df.set_index('lsmaccy')
        tabela5 = df.set_index('lsmaccz')

        accx = tabela3.index.values * 0.00036
        accy = tabela4.index.values * 0.00036
        accz = tabela5.index.values * 0.00036
        t1 = np.linspace(0, 0.005 * len(f1), len(f1))
        e, f = signal.butter(4, 0.007, analog=False)

        sig_accx = signal.filtfilt(e, f, accx)
        sig_accy = signal.filtfilt(e, f, accy)
        sig_accz = signal.filtfilt(e, f, accz)
    else:
        msb.showerror("ERRO", "O Caminho acima    inv lido")


def file_select():
    filename = fd.askopenfilename()
    if ".csv" in filename:
        txt5.delete(0, "end")
        txt5.insert(0, filename)
    else:
        msb.showerror("ERRO", "Arquivo Inv lido!")
        txt5.delete(0, "end")
```

```
def path_select():
    pathname = fd.askdirectory()
    txt2.delete(0, "end")
    txt2.insert(0, pathname)


def runs_select():
    pathname = fd.askdirectory()
    files = glob(os.path.join(pathname, "RUN*"))
    if files:
        txt1.delete(0, "end")
        txt1.insert(0, pathname)
    else:
        msb.showerror("ERRO", "O diret rio selecionado n o possui nenhu
        txt1.delete(0, "end")


def generate_csv():
    cfunctions = CDLL("./libshared_read_object.dll")
    if txt1.get() and txt2.get() and (":/" in txt2.get()) and txt3.get().
        cfunctions.read_struct(txt1.get().encode("utf8"), txt2.get().enco
    elif not(txt3.get().isnumeric()):
        msb.showerror("ERRO", "Voc digitou algo inv lido para ser lido
        txt3.delete(0, "end")
    else:
        msb.showerror("ERRO", "Algum dos caminhos acima est inv lido!"
```

```
lbl1 = Label(window, text="Selecione o diret rio das RUNs:")
lbl1.grid(column=0, row=0)
txt1 = Entry(window, width=30)
txt1.grid(column=1, row=0)
txt1.focus()
btn1 = Button(window, text="Procurar", command=runs_select)
btn1.grid(column=2, row=0)

lbl2 = Label(window, text="Selecione um diret rio para salvar:")
lbl2.grid(column=0, row=1)
txt2 = Entry(window, width=30)
txt2.grid(column=1, row=1)
txt2.focus()
btn2 = Button(window, text="Procurar", command=path_select)
btn2.grid(column=2, row=1)

lbl3 = Label(window, text="Digite o n mero da RUN:")
lbl3.grid(column=0, row=2)
txt3 = Entry(window, width=30)
txt3.grid(column=1, row=2)
txt3.focus()
btn3 = Button(window, text="Gerar CSV", command=generate_csv)
btn3.grid(column=2, row=2)

lbl4 = Label(window, text="N mero de furos no disco de freio:")
lbl4.grid(column=0, row=3)
txt4 = Entry(window, width=30)
txt4.grid(column=1, row=3)
```

```
txt4.focus()



lbl5 = Label(window, text="Selecione o arquivo para plot:")
lbl5.grid(column=0, row=4)
txt5 = Entry(window, width=30)
txt5.grid(column=1, row=4)
txt5.focus()
btn5 = Button(window, text="Procurar", command=file_select)
btn5.grid(column=2, row=4)
btn6 = Button(window, text="Plotar Gr ficos", command=grafico)
btn6.grid(column=1, row=5)


window.mainloop()
```