Relatório de Case:

Bancada de Testes

Amanda J. Nakamura “Sagu”
Candidato

Rafael E. I. Rasoto “Lambari”
Tutor

Processo Seletivo 2023 - Equipe Imperador

RESUMO

Os testes e validações de toda a parte eletrônica é uma etapa imprescindível antes das competições, não somente por ser requisito da SAE Brasil em relação a segurança, mas também pelas avaliações, pontuação e pelas informações que um sistema de telemetria funcional pode fornecer sobre o desempenho do veículo como um todo. Porém, esta etapa só é possível quando todos os demais sistemas estão funcionando, uma vez que o protótipo deve estar em movimento e adquirindo dados para realizar a maioria dos testes. Desta forma, o presente trabalho tem como função projetar uma “bancada de testes” que simula o funcionamento da parte eletrônica do veículo.

INTRODUÇÃO

O projeto BAJA SAE foi desenvolvido nos Estados Unidos com o objetivo de proporcionar aos alunos de graduação de engenharia, uma oportunidade prática de desenvolver um veículo off-road desde sua concepção, projeto, construção e testes, além de incentivar o trabalho em equipe, responsabilidade, e qualificação para o mercado de trabalho. O BAJA SAE Brasil atua desde 1994 através da promoção de competições regionais e nacionais realizadas anualmente entre equipes universitárias (BAJA NAC., Rev.2).

A equipe Imperador, fundada em 2008, representa a Universidade Tecnológica Federal do Paraná - campus Curitiba, e atua como uma das melhores equipes brasileiras, obtendo o 6º lugar geral na competição nacional.

Tendo em vista a excelência na entrega dos trabalhos, a equipe é organizada em subsistemas. O subsistema de eletrônica tem como principais objetivos garantir a segurança do piloto e de todos ao redor seguindo as regulamentações estabelecidas pelo BAJA SAE Brasil, além de desenvolver softwares de telemetria para coleta de dados através de sensores posicionados no veículo para validação, acompanhamento de desempenho, possibilitando também o feedback ao piloto sobre sua estratégia e condução durante as provas.

Os sensores utilizados e suas funções no protótipo Jaguara13 (J13) estão listados a seguir:

- Reed Switch localizado dentro do motor para contagem do RPM, acionado por campo magnético através do fechamento do circuito registrado na passagem de um ímã;
- Sensor indutivo para aferir a velocidade através da detecção de um objeto metálico que fornece uma resposta lógica;
- Sensor infravermelho (MLX90614) para acompanhamento da temperatura da CVT;
- Acelerômetro (MPU6050), que funciona detectando mudanças na capacitância de estruturas eletromecânicas quando acelerado, convertendo a informação em sinais elétricos para determinação de movimento e orientação espacial;
- Sensor de pressão que mede a força exercida pelo fluido de freio por unidade de área;
- Sensor de ativação do diferencial ainda não definido;
- GPS integrado vinculado ao módulo 4G;

Todos os sinais, digitais e analógicos, recebidos desses sensores, são enviados para a ECU que se localiza no painel com display digital, através de protocolos de comunicação como I2C, SPI e UART, e então, passado para o servidor web de telemetria para possibilitar o acesso aos dados tratados em tempo real à distância. Para backup, na ECU, os dados também são salvos em um cartão um SD.

Para teste e validação de todo o sistema eletrônico, atualmente, é necessário a implementação e presença do carro apto para rodar, o que exige que os outros subsistemas estejam preparados ou que os integrantes de eletrônica estejam presentes durante os testes de outras partes do carro, com horários extraordinários como aos finais de semana conforme a disponibilidade de todos.

Sendo assim, o objetivo deste trabalho foi desenvolver uma bancada de testes que simula o funcionamento da parte eletrônica do veículo. Isso foi feito a partir da geração de sinais digitais ou analógicos (conforme o módulo), semelhantes aos enviados pelos sensores à ECU.

Outro ponto solicitado, seria uma forma de testar os próprios sensores do J13 através de comparação com dados obtidos de testes controlados usando sensores comerciais. Isso seria feito através de simulações da ação captada pelos mesmos, e pretende-se implementar essa funcionalidade no futuro.

TIPOS DE SINAIS E PROTOCOLOS DE COMUNICAÇÃO

Os sinais transmitidos dos sensores até a ECU podem ser digitais ou analógicos dependendo do tipo e modelo do sensor.

Os sinais analógicos consistem em um tipo de sinal contínuo que representa variações de uma quantidade física e podem assumir qualquer valor dentro de um intervalo contínuo, sendo representados por formas de onda que podem ser senoidais, triangulares, quadradas, entre outras (Horowitz, 2015).

Os sinais digitais representam dados através de valores discretos, geralmente utilizando o sistema binário (0s e 1s). Esses sinais são essenciais para a operação de computadores e dispositivos eletrônicos modernos. Uma grande vantagem deste tipo de sinal é sua robustez, sendo menos suscetíveis a ruído e interferência. (Tenenbaum, 2011).

Os sinais digitais são enviados do sensor até a ECU através de conjuntos de regras que permitem a troca de informações entre dispositivos, denominados protocolos de comunicação. Neste trabalho, os tipos usados são de comunicação serial: UART, I2C e SPI.

O protocolo UART (Universal Asynchronous Receiver/Transmitter) é assíncrono, dessa forma não requer um sinal de clock compartilhado entre os dispositivos que se comunicam. Ele utiliza dois fios principais de transmissão de dados: um para transmissão (TX) e outro para recepção (RX), como pode-se referenciar no próprio nome do protocolo.Este tipo de comunicação é simples de implementar e adequado para comunicação ponto a ponto (Harold, 2007).

Já o I2C (Inter-Integrated Circuit) é síncrono e por isso utiliza um sinal de clock (SCL) para sincronizar a transferência de dados, além de um outro fio, SDA, para transmissão dos dados propriamente ditos. Uma de suas características principais é que ele suporta múltiplos dispositivos mestres e escravos no mesmo barramento, e isso é possível pois cada dispositivo possui um endereço que é enviado juntamente com a solicitação de informação. (Smiley. J, 2011)

O protocolo SPI (Serial Peripheral Interface), também é um protocolo de comunicação serial síncrona que usa quatro fios principais: MOSI (Master Out Slave In), MISO (Master In Slave Out), SCK (Serial Clock), e SS/CS (Slave Select/Chip Select). Ele permite uma comunicação mais rápida em relação ao I2C.

DAC E ADC

Os conversores digital-analógico (DAC) e analógico-digital (ADC) desempenham papeis cruciais na interface entre sistemas digitais e o mundo analógico. Um DAC converte sinais digitais em sinais analógicos, permitindo que dispositivos digitais, como microcontroladores e processadores, interajam com componentes analógicos, como alto-falantes e motores.

Por outro lado, um ADC converte sinais analógicos em sinais digitais, permitindo a amostragem e processamento de sinais do mundo real por sistemas digitais.

A conversão de dados digitais para volts é um processo essencial na eletrônica, onde os sinais digitais são transformados em sinais analógicos que podem ser utilizados em diversos componentes, ou seja, os DACs pegam valores binários discretos e os convertem em uma tensão contínua que representa o valor original do dado digital.

DESENVOLVIMENTO

Este projeto consiste na quarta etapa do processo seletivo Imperador 2024 e foi lançado no dia 9 de julho de 2024 com entrega no dia 11 de julho de 2024, além deste relatório, há a apresentação do case e do sistema em funcionamento junto com todo o material complementar. Isso, feito simultaneamente a atividades complementares de rotação e participação em reuniões de 3 subsistemas diferentes, e apresentação de um checklist de conhecimento em eletrônica.

MATERIAIS E MÉTODOS

O desenvolvimento do *case* foi dividido em etapas dado o prazo de entrega. A leitura do documento com o enunciado e entendimento completo dos objetivos e instruções fornecidas foram a base do planejamento desenvolvido em etapas.

O contato com o gerente de eletrônica durante esse período inicial foi crucial para adquirir mais dados e especificações para o projeto, além de materiais complementares, arquivos e informações prévias sobre o J13 que foram importantes para o desenvolvimento da bancada de testes.

Foram utilizadas ferramentas de planejamento e organização como o notion, notion calendar e google drive. Além de sites para pesquisa, artigos científicos e chatGPT versão 4.0.

Tendo uma visão geral do que seria necessário, foi feito um checklist básico com as etapas e conhecimento necessários para o desenvolvimento do código propriamente dito para a simulação dos sinais.

Cada pesquisa foi desenvolvida e salva com informações essenciais em tópicos em arquivos .docx para consultas posteriores.

A plataforma utilizada para o desenvolvimento do código foi a Arduino IDE junto com um ESP32 WROOM-32D adquirido previamente para testes. Para para configuração do ambiente foi necessário instalar o driver CH340, uma vez que a porta serial não estava sendo identificada, e encontrar a biblioteca de placas da Espressif para identificar a placa usada (DOIT ESP32 DEVKIT V1).

Foi necessário aprender a programar em C++ e pesquisar profundamente sobre sistemas embarcados e configuração de pinos pois não havia experiência alguma nesses tópicos. Uma das atividades extra do checklist de eletrônica contribuiu nesse quesito. A atividade solicitava a programação de um código na Serial( ) que imprimia a frase “Hello world”, além de conter as instruções para piscar o LED do ESP32 utilizando interrupção por timer. Apesar do processo de entendimento dos conceitos e da biblioteca esp_timer.h ter sido complicado, o propósito foi concluído e auxiliou em atividades posteriores.

Após o cumprimento dos estudos do checklist estabelecidos, iniciou-se a etapa de planejamento e início do desenvolvimento do projeto em si. Foi necessário, em primeiro plano, o entendimento dos tipos de sensores utilizados no J13 e suas funções para descobrir quais sensores enviavam quais tipos de ondas para a ECU a fim de reproduzi-las. Para isso consultou-se o datasheet de cada módulo utilizado, e as conclusões foram as seguintes: a aferição de velocidade, RPM, e freio eram enviadas através de sinais analógicos representados através de ondas high/low; enquanto temperatura e aceleração, através de protocolo I2C, e para testar o ADC (Conversor analógico-digital) de tensão, corrente e pressão do freio, foi necessário simular o DAC, ou seja, o processo inverso.

A escolha dos pinos foi feita através da análise do datasheet da placa. Dos 39 pinos GPIO alguns tinham funções específicas ou atuavam somente como input. Os pinos escolhidos neste trabalho foram: 16, 17 e 18 para velocidade das rodas dianteiras e do eixo traseiro; 20 para RPM, 21 para freio, 22 para temperatura, 23 para aceleração e 25 para DAC.

Para a construção do código (disponível em https://github.com/AJNkamura/ImperadorBaja), foi necessário o estudo de diversas bibliotecas. A ideia inicial era produzir a onda analógica através das funções delay( ) e millis ( ), porém o gerente instruiu a utilização de interrupção por timer por permitirem que tarefas sejam executadas periodicamente sem a necessidade de uma sondagem constante (polling) do tempo.

Isso foi feito configurando o timer para gerar interrupções em um intervalo específico correspondente à metade do período calculado*1000 (referente a transformação para milissegundos) a partir da frequência de inversão das ondas que difere para cada sensor. Cada timer chama uma função específica que inverte o sinal da onda a cada intervalo de tempo. Além disso, a cada timer é feita a verificação de erro ao iniciá-lo.

Para a definição do período das ondas de velocidade e rpm foi necessário avaliar o código da ECUV3 que recebia e tratava os dados. Lá estavam definidos o diâmetro da roda (0.51916m) e comprimento (CONST_PI * DIAMETRO), e a velocidade era calculada através da fórmula:

$$\Delta tempo = (COMP * k * 3.6)/velocidade$$

COMP: Comprimento da roda
k: constante referente a caixa de redução = $(1000/0.7)$
velocidade: velocidade escolhida para a simulação

As velocidades escolhidas para os testes foram definidas no início do código e foram: 15, 20, 35 km/h.

Para a aceleração, o período foi definido como:

$$\Delta tempo = 60000/rpm$$

Sendo 60.000 a quantidade de milissegundos por minuto, necessário para a conversão. O período usado para a simulação do freio é gerado aleatoriamente através da função random( ).

Para o freio, foi feito um trecho de código na função void loo( ). A chamada da simulação do sinal de freio foi definida para ocorrer a cada 5 segundos utilizando o tempo da última ocorrência da aferição e da função millis( ).

A função utilizada para DAC tinha como propósito gerar uma onda triangular com valores de 0 a 255 (correspondente a 3V). Quando o valor máximo era atingido, a onda voltava ao início. Essa função é chamada no void loop ( ) a cada 50 milissegundos, definidos pela função delay( ).

Apesar de ter sido solicitado, não foi possível simular o I2C. Uma pesquisa foi realizada sobre o assunto, e foi criado um esquema de como seria feito, mas a implementação não ocorreu devido ao prazo de entrega do trabalho.

A ideia para a simulação do I2C seria encontrar uma maneira de identificar a solicitação e o endereço do dispositivo da leitura de dados da ECUV3. Através desta identificação, usaria-se uma estrutura condicional para verificar o endereço do dispositivo escravo para então chamar uma outra função que geraria a informação e a transformaria em byte para que fosse transmitida para a ECU.

CONCLUSÃO

As funcionalidades do código proposto, de simular as ondas de velocidade, RPM, freio e DAC foram cumpridas (Disponível em < https://github.com/AJNkamura/ImperadorBaja>). Esta parte do código compila corretamente mas não foi possível verificar seu funcionamento pela falta do osciloscópio que tornaria possível a visualização das ondas, e por um problema no ESP32 que não estava permitindo a passagem do código.

Deseja-se fazer a implementação da simulação da I2C visto que já existe o esquema de como isso pode ser realizado. É necessário somente estudar a fundo as partes das bibliotecas utilizadas e entender melhor como funciona a solicitação de dados do mestre para os dispositivos escravos.

Github:< https://github.com/AJNkamura/ImperadorBaja >

REFERÊNCIAS

1. Horowitz, P., & Hill, W. (2015). The Art of Electronics (3rd ed.). Cambridge University Press.
2. Tanenbaum, A. S., & Wetherall, D. J. (2011). Computer Networks (5th ed.). Pearson Education.
3. Harold, L. (2007). "UART Communications." In Embedded Systems Design with the Atmel AVR Microcontroller (pp. 115-136). Springer.
4. Smiley, J. (2011). "Using the I2C Bus." In Arduino Workshop: A Hands-On Introduction with 65 Projects (pp. 287-304). No Starch Press.
5. https://www.crescerengenharia.com/post/como-instalar-driver-ch340#viewer-9os1qFranco, S. (1994). Design with Operational Amplifiers and Analog Integrated Circuits. McGraw-Hill.
6. Laker, K. R., & Sansen, W. M. C. (1994). Design of Analog Integrated Circuits and Systems. McGraw-Hill.
7. Schreier, R., & Temes, G. C. (2005). Understanding Delta-Sigma Data Converters. Wiley-Interscience.
8. https://espressif-docs.readthedocs-hosted.com/projects/arduino-esp32/en/latest/api/i2c.html
9. randomnerdtutorials.com/esp32-i2c-scanner-arduino/
10. https://circuitdigest.com/microcontroller-projects/esp32-timers-and-timer-interrupts
11. https://randomnerdtutorials.com/esp32-i2c-communication-arduino-ide/
12. Franco, S. (1994). *Design with Operational Amplifiers and Analog Integrated Circuits*. McGraw-Hill.
13. Laker, K. R., & Sansen, W. M. C. (1994). *Design of Analog Integrated Circuits and Systems*. McGraw-Hill.
14. Schreier, R., & Temes, G. C. (2005). *Understanding Delta-Sigma Data Converters*. Wiley-Interscience.
15. https://www.imbaja.com/equipe
16. https://saebrasil.org.br/quem-somos/