Thiago Zipper Melato e Fabio Ivo

Trabalho de Sistemas em Tempo Real (STR) - Concorrência e multithread em STR

Este relatório contém a implementação de um algoritmo de simulação da rede CAN de um carro, utilizando FreeRTOS.

Universidade do Vale do Itajaí - UNIVALI
Engenharia da Computação
Sistemas em Tempo Real

Orientador: Felipe Viel

Brasil – 2023

Sumario

Introdução

O seguinte trabalho tem por objetivo simular discretamente a rede CAN (Controller Area Network) de um carro, esta rede serve para interligar centrais de comando de diferentes funcionalidades, como motor, câmbio, conveniência, segurança etc.

Visto que alguns destes sistemas possuem grande importância e não podem sofrer com atrasos, como o sistema de airbags, é preciso que haja uma prioridade no processamento e exibição das informações.

Todos estes sistemas precisam se comunicar entre si via rede CAN, mas também precisam fornecer algumas dessas informações ao condutor, isso é feito por meio do computador de bordo ou pela central multimidia em alguns carros.

Essa exibição também exige um tempo de resposta em tempo real, visto que em alguns casos como de superaquecimento do motor o condutor precisa tomar medidas imediatas, ou quando o sensor frontal detecta um risco de colisão esta informação precisa ser repassada para o computador de bordo sem atraso.

Esta versão do trabalho tem como requisito rodar a aplicação em FreeRTOS em uma placa ESP32 utilizando SDK ESP-IDF, utilizando multithread e paralelismo.

Enunciado

A problemática deste trabalho consiste na simulação de um novo sistema de monitoramento do comportamento de várias áreas de um veículo. Sendo esses: Motor, Frenagem, Equipamentos de Suporte a Vida e Luzes, Vidros e Travas (LVT).

Cada sistema possui alguns sensores que devem ser monitorados, o sistema do motor deve controlar a injeção eletrônica e temperatura. Já na frenagem deve ser controlado apenas o ABS das duas rodas dianteiras. Para os equipamentos de suporte a vida é necessário controlar um airbag e cinto de segurança. E por fim o sistema de LVT controla as luzes dos faróis dianteiros, vidros elétricos dianteiros e as travas das portas dianteiras.

Todos estes sistemas devem se conectar ao computador de bordo que deve controlar estes sistemas e exibir suas informações. Para fins de simulação será atribuída uma tecla para executar as ações que deveriam ser obtidas por sensores, como uma colisão, pedal de freio, pedal do acelerador.

É necessário observar que a propagação das informações pelos fios leva 10us e deve-se manter dentro das deadlines necessárias para cada sistema, sendo elas:

- Injeção eletrônica: 500 us após alteração no pedal
- Temperatura do motor: 20 ms após detecção de temperatura acima do limite
- ABS: 100 ms após acionamento no pedal
- Airbag: 100 ms após detecção de choque
- Cinto de segurança: 1 segundo após carro em movimento
- LVT: 1 segundo após interação do usuário

A empresa exige que você expanda sua solução para trabalhar com multithread e utilizando o FreeRTOS na placa ESP32 e SDK ESP-IDF. Considere também que o sensor demora 1 us para fazer a aquisição da amostra corretamente e que o controlador demora 5 us para agir. Além disso, agora ela gostaria de monitorar a velocidade média e o consumo médio de gasolina e atualizar a cada instante no sensor.

Implementação

Para implementar o sistema solicitado foi optado por separar os subsistemas em tasks diferentes, com o fim de se atribuir prioridades diferentes para cada uma, podendo assim manter as tasks dentro das deadlines solicitadas. Sendo assim foram criadas 8 tasks, sendo elas:

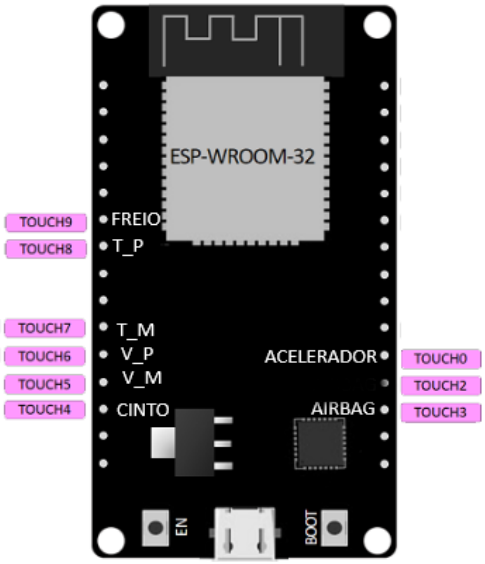
abs_task
airbag_task
cinto_task
ltv_task
motor_task
temperature_task
touch_pad_read_task

A simulação foi estruturada em várias tarefas (tasks), cada uma dedicada a um subsistema específico do veículo. O uso do multithreading, uma característica fundamental do FreeRTOS, permite que cada tarefa opere de forma independente. Esta abordagem é essencial para simular a natureza paralela dos sistemas de um veículo real, onde múltiplos processos ocorrem simultaneamente.

Cada task executa uma ou mais ações do carro, as tasks do abs, airbag e cinto controlam apenas os recursos de seu nome, no caso da task do ltv ela controla todos dos vidros, travas e luzes do carro, isso é possível pois estes recursos possuem um deadline maior, então não seria prejudicado pelo menor desempenho.

A task do motor carrega uma lógica para aumentar a velocidade com base na aceleração e freio e manipular o consumo com base nas mesmas variáveis, o caso da task de temperatura é muito semelhante, aumentando a temperatura do motor com base na aceleração e disparando uma variável global caso o motor superaqueça.

Dentro da touch_pad_read_task existe uma lógica para alterar as variáveis globais de cada sensor quando a interrupção no touch pad for acionada, seguindo o seguinte diagrama:



Mapeamento dos botões

A sincronização entre as tarefas é assegurada através de semáforos, evitando conflitos no acesso às variáveis globais. O FreeRTOS oferece um controle refinado sobre a priorização das tarefas, permitindo que tarefas críticas, como a ativação do airbag, tenham prioridade sobre outras menos críticas. Esta gestão de prioridades é crucial para o realismo e eficácia da simulação.

Cada subsistema tem requisitos específicos de tempo de resposta que foram considerados na implementação. Por exemplo, o sistema de airbag exige uma resposta rápida em caso de colisão. A implementação ajusta o tempo de resposta de cada tarefa para refletir com precisão o comportamento do sistema real, utilizando técnicas como delays e medição do tempo de execução das tarefas.

A tarefa de exibição desempenha um papel crucial, fornecendo feedback em tempo real (1 segundo de refresh) sobre o estado de cada subsistema. Esta funcionalidade não só ajuda na verificação do funcionamento correto da simulação, mas também emula a interface que um motorista teria com os sistemas do veículo, incluindo métricas como consumo médio de combustível e velocidade média.

Para configurar a prioridade de cada tarefa foi avaliada o tipo da deadline de cada uma, seguindo a tabela:

Task	Tipo Deadline	Justificativa
Abs	Hard	Sistema de suporte a vida
Airbag	Hard	Sistema de segurança
Cinto	Soft	Atraso no aviso não é fatal
Ltv	Soft	Controle de sistemas de conveniência não é fatal
Motor	Hard	Injeção eletrônica possui pequena deadline
Temperatura	Hard	Temperatura do motor precisa ser verificada o quanto antes
Touch Pad	Hard	Obtenção das interrupções é tão importante quanto o tratamento delas

Para obter os tempos de execução de cada tarefa foi utilizado a função esp_timer_get_time() capturando o tempo de execução entre o inicio e fim da task e armazenando os dois valores em variáveis globais do tipo uint64_t para serem calculados de uma vez na tarefa do display.

Desempenho

Com o fim de avaliar o desempenho geral da aplicação foram executadas análises de melhor, médio e pior caso. A metodologia para cada um deles foi:

- **Melhor caso:** Placa resetada e ativação de apenas uma interrupção por vez com intervalos maiores.
- **Médio caso:** Placa resetada e ativação de algumas interrupções com intervalos médios.
- **Pior caso:** Placa executando por um longo período de tempo e ativação de todas as interrupções simultaneamente e repetidamente.

Para adquirir os dados da execução em cada caso foi desenvolvido um programa em python para se comunicar com a ESP32 e armazenar os dados brutos em um arquivo .txt.

A partir deste arquivo foi possível utilizar outro programa desenvolvido em python para analisar os dados, tirando a média dos valores adquiridos e apresentando de forma organizada o resultado das execuções, permitindo avaliar o desempenho médio em cada caso, como mostram as tabelas:

Pior Caso	
Sensor	**Tempo**
Injeção Eletrônica	27 us
ABS	27.11 us
Airbag	27 us
Cinto	27 us
Luz	43 us
Trava Motorista	71.5 us
Trava Passageiro	72.66 us
Vidro Motorista	27.14 us
Vidro Passageiro	43.2 us
Temperatura	27 us

Médio Caso	
Sensor	**Tempo**
Injeção Eletrônica	27 us
ABS	27 us
Airbag	27.1 us
Cinto	24 us
Luz	43 us
Trava Motorista	46.2 us
Trava Passageiro	54.33 us
Vidro Motorista	54 us
Vidro Passageiro	47.1 us
Temperatura	27 us

Melhor Caso	
Sensor	**Tempo**
Injeção Eletrônica	27 us
ABS	27 us
Airbag	27 us
Cinto	27 us
Luz	27 us
Trava Motorista	27 us
Trava Passageiro	27 us
Vidro Motorista	27 us
Vidro Passageiro	27 us
Temperatura	27 us

Conclusão

A implementação desta simulação de rede CAN em um veículo, utilizando FreeRTOS, ESP32 e ESP-IDF, demonstra a eficácia dessas tecnologias no desenvolvimento de sistemas automotivos complexos. A abordagem adotada para a simulação oferece uma representação realista e detalhada dos diversos subsistemas de um veículo, destacando a importância de um gerenciamento de tarefas eficiente e de uma sincronização precisa para o sucesso de sistemas embarcados em tempo real.

A capacidade de simular com precisão o comportamento de sistemas críticos, como o airbag e o sistema de frenagem ABS, dentro dos limites de tempo estabelecidos, é um testemunho da robustez do design e da implementação. Além disso, a modularidade e a clareza do código contribuem para a manutenção e a compreensão do sistema, aspectos fundamentais em projetos de engenharia de software.

Em resumo, este projeto não apenas atinge seus objetivos de simulação, mas também serve como um exemplo prático da aplicação de sistemas operacionais em tempo real, multithreading e sincronização em sistemas embarcados, abrindo caminho para futuras inovações e melhorias no campo da tecnologia automotiva

Referencias

MELATO, Thiago Zipper; IVO, Fabio. **TRABALHO de Sistemas em Tempo Real (STR) – Concorrência e multithread em STR.** [S. l.], 23 out. 2023. Disponível em: https://github.com/sergio-venturi/Trabalho-M2/tree/main Acesso em: 23 out. 2023.