Avaliação do uso do Sistema Operacional Embarcado FreeRTOS em Aplicações Automotivas

Vinicios Luneburger Anacleto[1], **Anderson L. Fernandes Perez**[2]

[1]Laboratório de Automação e Robótica Móvel – Universidade Federal de Santa Catarina (UFSC)
88906-100 – Araranguá – SC – Brasil

viniciosanacleto@hotmail.com, anderson.lfp@gmail.com

Abstract. *With the rise of complexity about electronic gadgets, domestic or specific application, a embedded operating system is indispensable. A embedded operational system offers abstraction about the hardware details and a set of resources for the embedded systems programmer. The modern vehicles have many control units that provide some functions to the driver, this is the cause for software development for the automotive market be so hard and request a programmer with knowledge about hardware and automotive software engineering rules. In this paper is accomplished a case study with the FreeRTOS operational system in a automotive application, a cruise control system, also knowed how auto pilot.*

Resumo. *Com o aumento da complexidade dos dispositivos eletrônicos, seja de consumo doméstico ou mesmo em aplicações específicas, o uso de um sistema operacional embarcado é indispensável. Um sistema operacional embarcado permite a abstração dos detalhes do hardware e ainda disponibiliza um conjunto de funcionalidade para o programador de sistemas embarcados. Os veículos modernos são dotados de várias unidades de controle que provem as mais diversas funcionalidades para o motorista, em virtude disso o desenvolvimento de software para o mercado automotivo não é tarefa fácil e portanto exige do programador conhecimento do hardware e de algumas normas existentes para a engenharia de software automotiva. Neste trabalho é realizado um estudo de caso com o sistema operacional FreeRTOS em uma aplicação automotiva, um sistema de controle de cruzeiro, também conhecido como piloto automático.*

1. Introdução

Atualmente os veículos automotores possuem funções implementadas por software que gerenciam e controlam subsistemas que necessitam de métodos, processos e ferramentas adequadas para as fases de desenvolvimento, produção e serviços [BONNICK 2001]. O nível e disponibilidade de conteúdos de eletroeletrônica automotiva vêm sendo caracterizados como fatores diferenciais aos consumidores para um mercado dinâmico.

O desenvolvimento de sistemas embarcados automotivo envolve conhecimentos de diversas áreas, destacando-se Mecânica, Elétrica, Eletrônica e Computação, em que podem ser caracterizados em um nível de complexidade elevado para ser gerenciados e controlados. Em face ao elevado nível de tecnologia embarcada nos veículos automotores,

os softwares embarcados em uma ECU (*Electronic Control Unit*) , são atualmente considerados como artefatos que podem diferenciar modelos e variantes de veículos, sendo considerados como componentes em todo ciclo de desenvolvimento, produção e serviços [NAVET and SIMONOT-LION 2009].

A indústria automobilística vem trabalhando em conjunto com fornecedores e desenvolvedores de ferramentas para a definição de padrões para desenvolvimento de software automotivo. Têm-se como exemplos de padrões conhecidos o OSEK/VDX e mais recentemente o AUTOSAR.

O padrão OSEK (*Open Systems and the Corresponding Interfaces for Automotive Electronics*) foi inicialmente definido em 1993 por um consórcio de empresas alemãs BMW, Bosch, Daimler-Chrysler, Opel, Siemens, VW em parceria com a Universidade de Karlsruhe. Já o padrão VDX (*Vehicle Distributed eXecutive*) foi definido na França pelos fabricantes de veículos PSA e Renault. Em 1994 os dois padrões, o alemão e o francês, foram fundidos com objetivo em formar a norma única denominada por OSEK/VDX.

O padrão AUTOSAR (*AUTomotive Open System ARchitecture*) é um consórcio definido em 2002 por fabricantes de veículos em parceria com fornecedores de componentes eletroeletrônicos para automação veicular, utilizando como base, vários aspectos já realizados no padrão OSEK/VDX como as características do Sistema Operacional [Pop et al. 2013]. O AUTOSAR define uma metodologia para o desenvolvimento de software veicular baseado em uma arquitetura em camadas com interfaces bem definidas e a total independência entre software de aplicação e hardware, tornando a interoperabilidade fator benéfico aos fabricantes e fornecedores de peças [WOLF 2008].

Este artigo está organizado em mais 5 (cinco) seções, além desta introdução. Na Seção 2 são descritas as principais características do sistema operacional FreeRTOS. A Seção 3 descreve o funcionamento das *Eletronic Control Units* (ECUs), bem como os padrões de desenvolvimento de software adotados pela indústria automotiva. Na Seção 4 é apresentado o funcionamento de um sistema de controle de cruzeiro (piloto automático). A Seção 5 descreve a implementação do sistema de cruzeiro desenvolvido no FreeRTOS bem como os resultados obtidos. Na Seção 6, são apresentadas as considerações finais sobre o trabalho desenvolvido e algumas propostas para trabalhos futuros.

2. Sistema Operacional FreeRTOS

O FreeRTOS é um sistema operacional de tempo real (RTOS - *Real Time Operating System*) livre de licenças. Foi desenvolvido para o mercado de aplicações microcontroladas por um consórcio entre as maiores fabricantes de componentes eletrônicos do mundo. Por possuir qualidades como, fácil implementação de código, facilidade de uso e robustez, o FreeRTOS hoje é líder no mercado de desenvolvimento de sistemas operacionais embarcados [FreeRTOS 2015a].

O FreeRTOS é um sistema operacional multitarefas, sendo uma tarefa definida por uma função em linguagem C onde está definido o código a ser executado. O escalonador de tarefas consegue executar múltiplas tarefas a partir da divisão do tempo de processamento com base na prioridade definida em cada tarefa [Melot 2009].

Além das funções controladas automaticamente pelo escalonador de tarefas, o desenvolvedor pode utilizar funções disponíveis na API (*Application Programming In-*

terface) do FreeRTOS para fazer manualmente a suspensão e retomada de execução das tarefas [Barry 2006].

O FreeRTOS implementa um mecanismo de fila de mensagens para comunicação entre tarefas. Uma fila de mensagens é um vetor de tamanho *N* implementado com a política de inclusão e retirada de elementos do tipo FIFO (*First In First Out)*. Além das filas de mensagens o FreeRTOS provê um mecanismo de sincronização de tarefas baseado em semáforos.

Por possuir pouca memória disponível, o uso da função *malloc()* para alocação de memória em sistemas embarcados pode causar uma série de problemas. Para evitar erros relacionados a alocação de memória, o FreeRTOS possui cinco maneiras de alocar memória dinamicamente[FreeRTOS 2015b], são elas:

- Heap 1: é a forma mais simples e não permite liberação de memória.
- Heap 2: permitida a liberação da memória mas não permite união de blocos de memória livres.
- Heap 3: funciona como a função padrão *malloc()* mas com segurança garantida para memória.
- Heap 4: permite união de blocos de memria liberados para evitar fragmentação.
- Heap 5: como o Heap 4, entretanto com a capacidade de estender o Heap em diversas áreas de memória não-adjacentes.

Cabe ao projetista da aplicação embarcada escolher corretamente a melhor forma de alocação de memória que será utilizada no projeto.

3. Sistemas de Controle Automotivo

Uma ECU (*Electronic Control Unit*) é um módulo eletrônico que tem a função de controlar determinadas funções no automóvel como o controle dos vidros elétricos, a injeção de combustível ou mesmo o acionamento dos freios. A ECU atua lendo variáveis do ambiente externo através de sensores [Wisniewski 2003].

Um sistema de controle automotivo pode ser baseado em uma única ECU ou pode conter várias ECUs. A arquitetura centralizada consiste na leitura de dados do ambiente via sensores e controle de atuadores administradas por uma única ECU. Embora a estrutura seja simplificada qualquer acréscimo de funções faz-se necessário uma modificação completa de hardware e software [Junior 2012].

Em um sistema distribuído várias ECUs estão interligadas através de canais de comunicação, e se diferenciam entre si quanto às demandas solicitadas pelas aplicações [SANTOS 2010]. A utilização de várias ECUs em rede distribuída, embora torne o sistema muito mais complexo, é importante para facilitar a retificação de sistemas, por tornar cada ECU mais independente. É importante salientar que a estrutura física de uma ECU é construída de acordo com as funções e sensores que ela irá controlar, tornando assim cada ECU mais eficiente. A Figura 1 ilustra a diferença entre o modelo centralizado e o distribuído.

Considerando que cada automóvel possui um hardware e um software específico, os fabricantes notaram que a necessidade de readaptar o sistema exigia muito tempo de desenvolvimento. Percebendo a complexidade para se criar um sistema eficiente, algumas

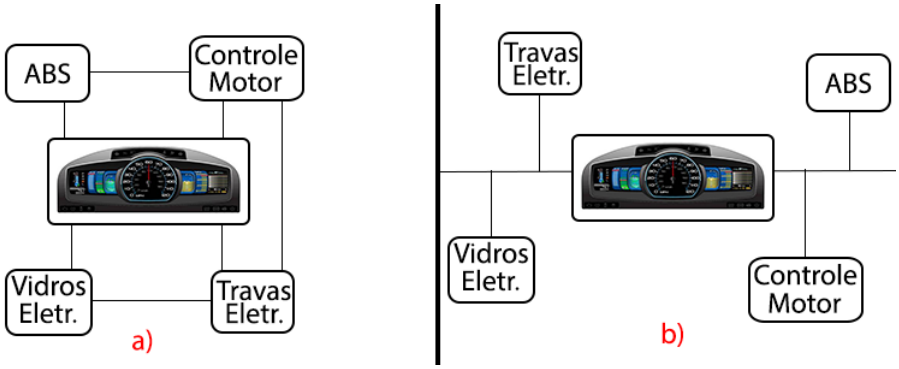

Figura 1. Tipos de estruturas de ECUs. Centralizada (a). Distribuida (b).

empresas se juntaram para tentar criar um padrão de técnicas utilizadas no desenvolvimento eletrônico do automóvel. Da união de algumas das principais empresas do setor surgiram iniciativas como a OSEK/VDX e ASAM que tiveram um relativo sucesso mas não conseguiram unir toda a indústria automotiva [Silva 2015].

Em 2002 surgiu o AUTOSAR, criado a partir da união das maiores empresas do ramo automotivo e fabricantes de componentes eletrônicos: BMW, Bosch, Continental, DaimlerChrysler e Volkswagen. Tendo como objetivo diminuir a complexidade na implementação de software em múltiplos modelos de hardware, a flexibilidade na modificação e escalabilidade do produto [AUTOSAR 2015]. O AUTOSAR possui uma arquitetura disposta em três camadas, conforme Figura 2.

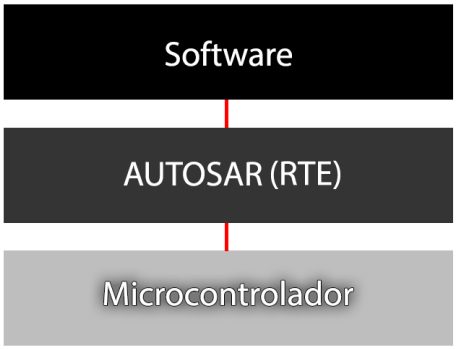

Figura 2. Disposição em camadas da arquitetura AUTOSAR.

Na primeira camada está localizado o software, onde está a aplicação em si. Com o uso do AUTOSAR o projetista só precisa se preocupar em desenvolver a aplicação em si, pois ela é totalmente independente do tipo de hardware usado.

A segunda camada é representada pelo RTE (*Runtime Environment*), onde é localizado o AUTOSAR. Essa camada é responsável por fazer a comunicação entre a camada superior (software) e a camada inferior (hardware).

A última camada é onde está presente o hardware, a única camada que não é abstrata do meio físico. É nesta camada que é controlado os sensores e atuadores da

ECU.

Ainda no padrão AUTOSAR existe o VFB (*Virtual Functional Bus*) que é um meio de comunicação que permite a troca de informações entre aplicações mesmo que elas não estejam na mesma ECU. A aplicação que utiliza o VFB não necessita saber com qual outra aplicação ela está se comunicando, basta que ela disponibilize a saída de dados, pois o VFB se responsabiliza por guiar a informação até a aplicação requisitante [Silva 2015].

4. Controle de Velocidade de Cruzeiro (Piloto Automático)

Com o avanço da tecnologia automotiva os fabricantes foram motivadas a investir na segurança e conforto de seus passageiros, surgindo assim, novas tecnologias que tornaram mais prático e seguro o ato de dirigir um automóvel. O *Cruise Control* (Controle de Velocidade de Cruzeiro), também conhecido como Piloto Automático, é um sistema que permite, após a aceleração manual do veículo, o acionamento de um sistema que fará o controle automático da velocidade.

O *Cruise Control* funciona ligado ao velocímetro do carro e ao controle de aceleração, assim, ao ser acionado o sistema deve controlar a velocidade regulando a aceleração do motor, sem intervenção do motorista. Em caso de emergência ou necessidade de desativação, o sistema se desliga e permite que o motorista tome o controle ao pressionar um botão de desativação ou ao acionar o pedal do freio [Bendix 1965].

Figura 3. Controle de Velocidade de Cruzeiro.
Fonte: Internet.

O uso de controles e assistências de velocidade em carros topo de linha evoluíram muito a partir do *Cruise Control* padrão. Com a associação desses sistemas à sensores de maior precisão e confiabilidade, algumas empresas investem no ACC (*Adaptative Cruise Control*).

O ACC tem como proposta manter uma distância segura entre dois veículos, sem intervenção humana, utilizando o *Cruise Control* em harmonia com sensores de distância altamente precisos. O sistema funciona como um complemento ao *Cruise Control* e utiliza a velocidade controlada pelo mesmo, juntamente aos sensores de distância, para calcular e manter um espaço seguro entre veículos na rodovia [Vahidi and Eskandarian 2003]. A Figura 4 ilustra o funcionamento do ACC.

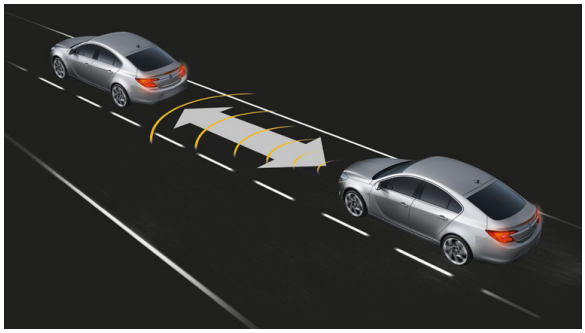

Figura 4. Distância mantida pelo ACC.
Fonte: Internet.

Há alguns fatores que contrariam o sentido de evolução dessa tecnologia, o principal é o fator humano. Apesar do sistema de ACC garantir a segurança do veículo controlado, o risco ocorre por causa dos veículos conduzidos por humanos nas rodovias. Outros fatores estão ligados as próprias rodovias as quais se encontram em más condições ou mal sinalizadas.

5. Experimento com o Controle de Cruzeiro Implementado no FreeRTOS

O experimento proposto neste artigo visa simular uma ECU de Controle de Velocidade de Cruzeiro(*Cruise Control*) que se comunique com outras ECUs automotivas através de tarefas implementadas no FreeRTOS que se comunicam por meio de filas. Para um melhor entendimento esta Seção foi dividida em quatro tópicos, a saber: descrição do hardware, descrição das tarefas, sistema de comunicação e experimento realizado.

5.1. Hardware

Para simular os componentes do hardware foi utilizado o software de simulação de circuitos eletrônicos Proteus. Na Figura 5 é possível observar a utilização de potenciômetros para simular os pedais de freio e acelerador, um servo para controle de aceleração do motor e um botão para acionamento do Cruise Control. Também é possível observar o microcontrolador ATMEGA328P, onde é executado o sistema operacional FreeRTOS.

5.2. Tarefas

Em uma primeira etapa, para testar as funcionalidades do FreeRTOS atuando em uma ECU, foram criadas tarefas no sistema operacional, onde cada tarefa simboliza uma ECU em funcionamento. As tarefas criadas são:

- Ler o pedal de acelerador e controlar o servo de aceleração do motor.
- Ler o pedal de freio.
- Controle de Velocidade de Cruzeiro.

Nota-se que que cada tarefa em uma função bem definida no sistema e o principal objetido foi o de simular a comunicação entre as três ECUs envolvidas no sistema de *Cruise Control*.

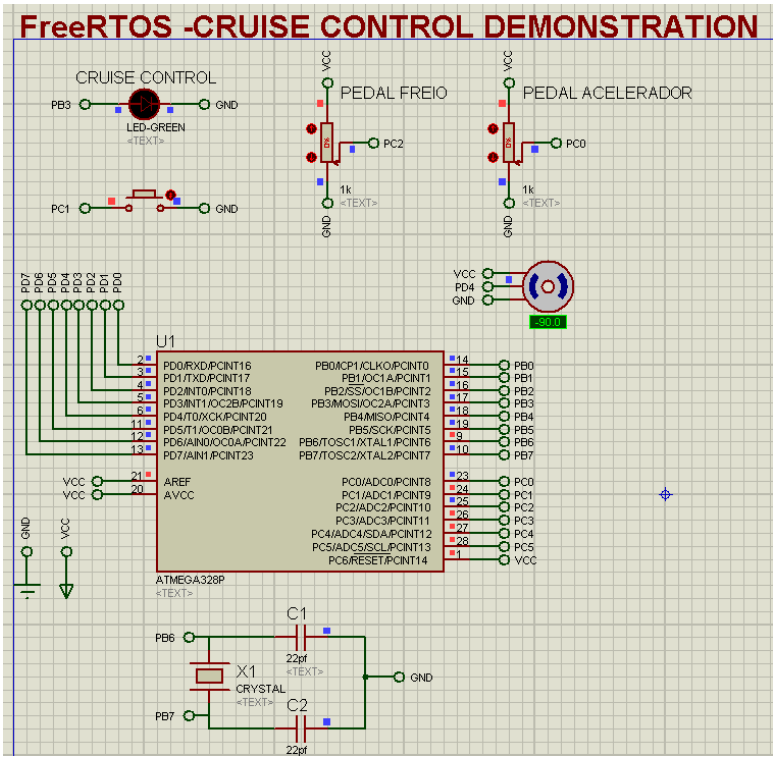

Figura 5. Componentes de hardware utilizados na experimentação simulados no Proteus.

5.3. Comunicação

Por se tratar de uma simulação de ECUs a comunicação entre as mesmas não poderia ser por meio de variáveis globais adotando o conceito de que as ECUs estariam executando em diferentes microcontroladores em uma situação real. Assim sendo, foram utilizadas as funções de filas implementadas no FreeRTOS. A utilização de filas simula o envio e recepção de dados por meio de um VFB (*Virtual Functional Bus*).

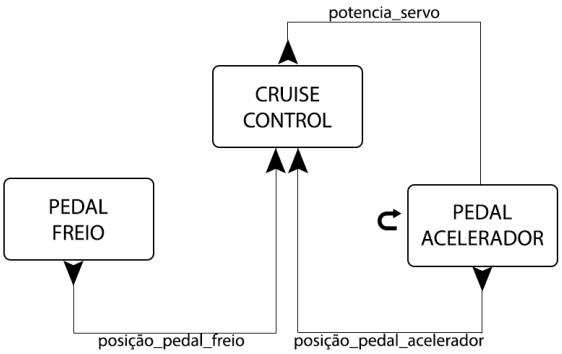

Figura 6. Diagrama de comunicação entre tarefas.

Como pode ser observado no diagrama da Figura 6, a tarefa **CRUISE CONTROL** faz a recepção de dados do pedal de freio e acelerador, a partir da ativação do sistema, ou

desativação, é tomada uma decisão que é enviada para o pedal de acelerador que controlará o servo de aceleração do motor. É importante ressaltar que a tarefa **PEDAL ACELERADOR** possui uma redundância de segurança que, caso a tarefa que controla o *Cruise Control* venha a falhar, não prejudique o funcionamento da aceleração do motor, fazendo com que o controle fique por conta do motorista.

5.4. Experimento Realizado

Ao iniciar a simulação sem a ativação do *Cruise Control*, onde conforme a Figura 7, o acelerador foi acionado 70% e gerou uma resposta do servo motor que se move aproximadamente 50°.

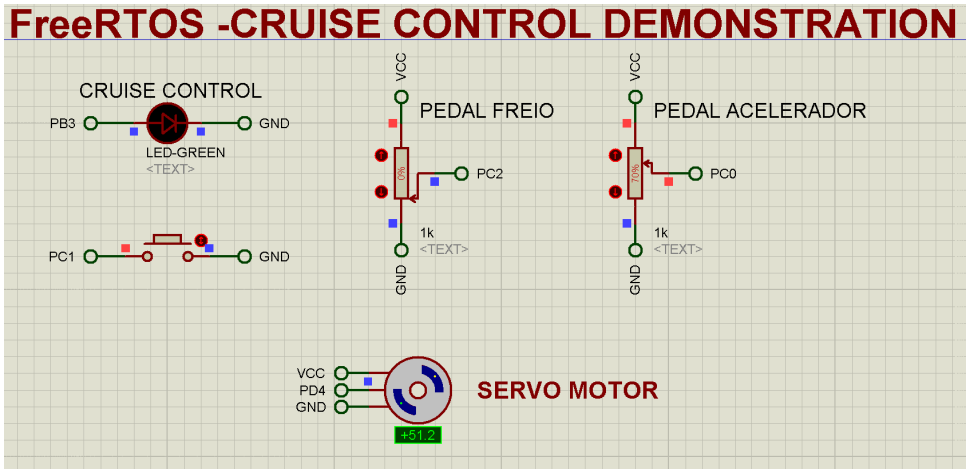

Figura 7. Acelerador acionador 70% gerando uma resposta de aproximadamente 50° no ângulo do servo motor.

Em seguida o *Cruise Control* foi ativado e o pedal do acelerador foi colocado em 0% e o servo motor mantem sua posição de aproximadamente 50°, conforme ilustra a Figura 8.

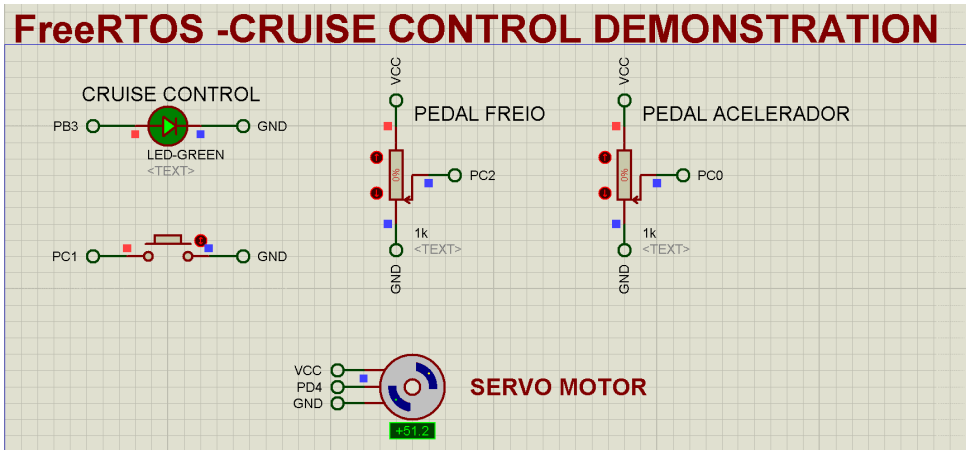

Figura 8. *Cruise Control* ativado e posição do servo motor mantida.

Como pode ser observado na Figura 9, ao acionar o pedal de freio em 20% o *Cruise Control* é desativado e o servo motor toma a posição de -90°, equivalente a 0% de

acionamento do pedal de acelerador. O comportamento de desativação do Cruise Control também ocorre caso o botão de acionamento seja pressionado novamente ou o acelerador seja ativado.

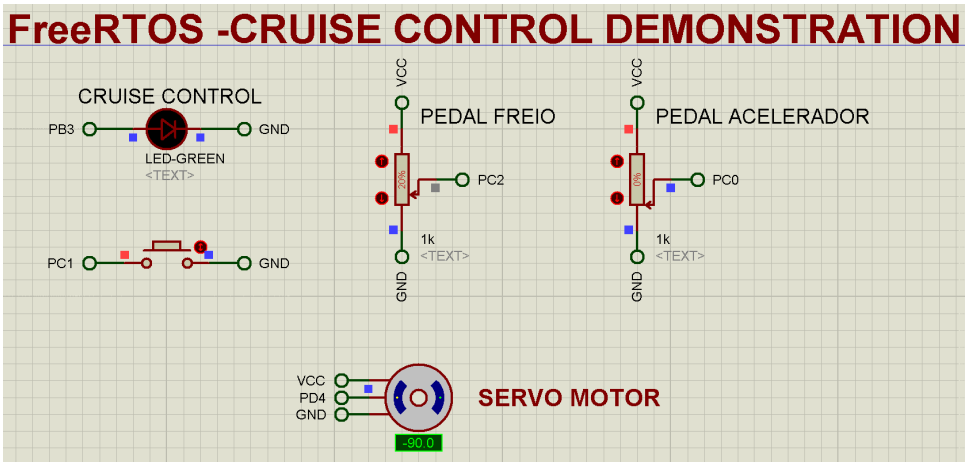

Figura 9. Pedal de freio acionado 20% e *Cruise Control* desativado.

6. Considerações Finais e Trabalhos Futuros

Neste trabalho foi apresentado um estudo de caso envolvendo o desenvolvimento de uma aplicação automotiva baseada no sistema operacional embarcado FreeRTOS. A aplicação desenvolvido é um controle de cruzeiro, também conhecido como piloto automático, que teve o hardware simulado no sistema Proteus com algumas tarefas implementadas no FreeRTOS simulando a comunicação entre *Eletronic Control Units* (ECUs).

O sistema desenvolvido foi baseado no padrão AUTOSAR, uma padrão aberto para o desenvolvimento de software automotivo, que foi proposto por várias empresas do ramo automobilístico. O controle de cruzeiro desenvolvido se mostrou totalmente confiável, apesar do sistema operacional FreeRTOS não seguir as normas especificadas no padrão AUTOSAR.

Com os resultados obtidos foi possível demonstrar que o sistema FreeRTOS se adequa bem aos requisitos do padrão AUTOSAR, sobretudo ao VFB (*Virtual Functional Bus*) que foi simulado com o recurso de troca de mensagens via fila de mensagens disponível no FreeRTOS.

Para trabalhos futuros pretende-se adaptar o FreeRTOS para o padrão AUTOSAR, assim como construir ECUs utilizando diversos microcontroladores. Deste modo poderia ser testada a segurança e confiabilidade do sistema, para enfim ser implementado em uma situação que utilize um veiculo real, tornando assim o estudo mais preciso.

Referências

AUTOSAR (2015). About autosar.

Barry, R. (2006). Freertos-a free rtos for small embedded real time systems.

Bendix (1965). Automotive cruise control. US Patent 3,207,252 A.

BONNICK, A. W. M. (2001). *Automotive Computers Controlled Systems - Diagnostics Tools and Techniques*. Butterworth Heinemann.

FreeRTOS (2015a). About freertos.

FreeRTOS (2015b). Memory management.

Junior, H. T. (2012). Estudo dos protocolos de comunicação das arquiteturas eletroeletrônicas automotivas, com foco nas suas características e respectivas aplicações, visando o direcionamento para o uso adequado e customizado em cada categoria de veículo.

Melot, N. (2009). Study of an operating system: Freertos.

NAVET, N. and SIMONOT-LION, F. (2009). *Automotive Embedded Systems Handbook*. CRC Press.

Pop, T., Hnětynka, P., Hošek, P., Malohlava, M., and Bureš, T. (2013). Comparison of component frameworks for real-time embedded systems. *Knowledge and Information Systems - An International Journal (online)*, 40(1):1–44.

SANTOS, M. M. D. R. d. (2010). comunicação automotiva: características, tecnologias e aplicações. *São Paulo: Érica*.

Silva, B. S. d. S. d. (2015). Desenvolvimento de software embarcado automotivo aderente ao padrão autosar.

Vahidi, A. and Eskandarian, A. (2003). Research advances in intelligent collision avoidance and adaptive cruise control. *Intelligent Transportation Systems, IEEE Transactions on*, 4(3):143–153.

Wisniewski, A. (2003). Vehicle electronic control units. US Patent 6,669,505.

WOLF, W. H. (2008). *Computer as Components: principles of embedded computing system design*. Morgan Kaufmann.