Estratégias de Construção de Firmware baseada em Sistemas Operacionais de Tempo Real para Veículos Seguidores de Trilha

Marcelo Barros de Almeida
Fac. Eng. Elétrica
Univ. Fed. de Uberlândia
Uberlândia, Brasil
marcelo.barros@ufu.br

Thomás de C. S. Pitombeira
Fac. Eng. Elétrica
Univ. Fed. de Uberlândia
Uberlândia, Brasil
thomas.csp@hotmail.com

Márcio Jose da Cunha
Fac. Eng. Elétrica
Univ. Fed. de Uberlândia
Uberlândia, Brasil
mjcunha@ufu.br

***Resumo* - Neste trabalho é apresentada uma proposta de construção de firmware para veículos seguidores de trilha baseada em sistema operacional de tempo real. Essa proposta é discutida e comparada com a estratégia tradicional *super loop* (ou sequencial). Como será visto, o RTOS possibilita uma divisão das tarefas executadas pelo controlador, otimizando o tempo do processador e evitando a ociosidade. Além disso, pode ser notado um maior determinismo nas operações de controle, desejável neste tipo de aplicação.**

***Palavras Chaves* - Sistema operacional de tempo real, seguidor de trilha, firmware**

Firmware strategy based on Real-Time Operating System for building track follower vehicles

***Abstract* - In this paper is presented a firmware building proposed for track followers vehicles based on real-time operating system. This proposal is discussed and compared with the traditional approach *super loop* (or sequential). As will be seen, the RTOS enables a division of the tasks performed by the controller, optimizing the processor time and avoiding idleness. Moreover, it can be noticed increased determinism in control operations, desirable in such applications.**

***Keywords* - real-time operating system, track follower, firmware**

I. INTRODUÇÃO

Neste artigo são apresentadas e comparadas duas estratégias de construção de veículos seguidores de trilha. Na primeira, denominada de *super loop* [1], [2], o código é colocado de forma sequencial e repetido indefinidamente. Movimentos atuais como **Arduino** [3] e **mbed** [4] acabaram por consolidar a prática do *super loop*, em geral aceitável para pequenos projetos e protótipos mas questionável em projeto maiores.

A segunda estratégia desenvolvida é baseada na utilização de um sistema operacional de tempo real (RTOS) [1], [2]. Com um RTOS é possível definir prioridades para cada tarefa e fatias máquinas de uso do processador. Via um escalonador e uma política de escalonamento, o RTOS realiza a execução periódica de tarefas, de acordo com as definições do usuário.

XIV CEEL - ISSN 2178-8308
03 a 07 de Outubro de 2016
Universidade Federal de Uberlândia - UFU
Uberlândia - Minas Gerais - Brasil

II. SEGUIDORES DE TRILHA IMPLEMENTADOS COM SUPER LOOP

A estrutura de firmware de um veículo autônomo seguidor de linha básico pode ser vista na Figura 1.

Basicamente, através de uma série de sensores infravermelho colocados em linha e voltados para o solo, é feita a inferência da posição da trilha a ser seguida. Um sensor de ultrassom, capaz de medir a distância do veículo a um obstáculo posicionado a sua frente, completa o conjunto mínimo de sensores. Os dados destes sensores são processados por uma unidade de processamento central (CPU), gerando comandos para os motores acoplados às rodas (esquerda e direta) do veículo. A velocidade desse motores é controlada através de um sinal pulsado modulado (PWM), sendo possível também ter controles digitais de direção associados (para frente ou para trás). Um rodízio é posicionado na frente do veículo, fazendo o papel da terceira roda.

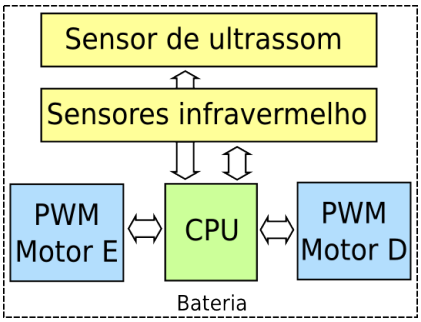

Fig. 1. Esquema básico de um veículo seguidor de trilha

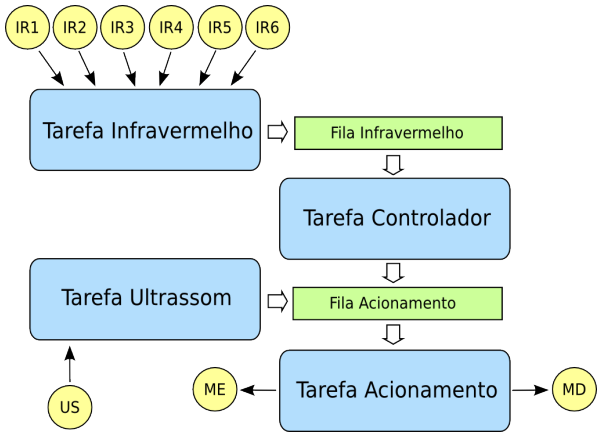

Fig. 2. Implementação Baseada em RTOS

Este veículo não possui *encoders* para cálculo de distâncias.

De maneira simplificada, através da estratégia *super loop*, o código de acionamento desse veículo pode ser descrito uma repetição contínua dos passos 1 a 3, descritos a seguir:

1) Ler sensores infravermelho e ultrassom.
2) Processar as informações dos sensores, decidindo a velocidade e direção dos motores. Estratégias mais complexas de desvio e movimento podem ser incluídas neste passo, assim como controladores digitais do tipo PID [5].
3) Gerar um valor de PWM (velocidade) e direção para cada roda do veículo.

A estratégia *super loop* traz alguns inconvenientes. O primeiro e mais grave é a falta de determinismo na execução dos passos citados pois a cada rodada, dependendo do caminho adotado nas estratégias de controle, o tempo de execução pode variar.

Do ponto de vista de controle digital, a amostragem com frequência fixa é necessária para a aplicação de um algoritmo de PID digital, devendo ser garantida pelo implementador. É até possível contornar este problema através da inserção de um passo adicional, responsável por aguardar uma determinada de quantidade de tempo que faça com que todos os ciclos sejam iguais. O custo, nesse caso, é ter um tempo ocioso de CPU e, pior, um veículo não responsivo durante esta espera.

O segundo problema decorrente do uso de *super loops* é a impossibilidade de reagir rapidamente a eventos externos. Por exemplo, a detecção de um obstáculo muito próximo pelo sensor de distância pode ser percebida, no pior caso, somente depois de um ciclo completo. Enquanto isso possa ser tratado no código através de leituras diversas espalhadas dentro do *super loop*, não é necessariamente uma estratégia elegante, necessitando de ajustes do tipo "tentativa e erro".

Na implementação realizada, um terceiro problema ainda existe, ligado ao acionamento dos motores. Para evitar uma aceleração brusca, como possíveis derrapagens, é executada uma rampa de aceleração para cada novo valor de velocidade calculado. Apesar de rápida, é preciso esperar que seja totalmente cumprida, ao final do *super loop*.

III. SEGUIDORES DE TRILHA IMPLEMENTADOS COM RTOS

Através de uma modelagem do veículo seguidor de trilha, os seus elementos podem ser traduzidos em tarefas e mecanismos de comunicação, permitindo uma implementação baseada em RTOS. No caso, foi empregado o RTOS disponível no mbed, uma versão do RTX [6]. Na Figura 2 são apresentados os elementos desta modelagem, descritos a seguir.

Uma primeira tarefa foi criada e nomeada de Tarefa "Infravermelho" possui a finalidade de realizar a leitura dos valores analógicos fornecidos pelos sensores infravermelhos. Cada sensor fornece um valor analógico entre 0 e 1, onde 1 implica totalmente sobre a linha e 0 totalmente fora da linha. Após todos lidos, uma soma ponderada é realizada afim de criar uma posição virtual da distância da trilha aos sensores centrais, dada pela Equação 1, onde IR_n representa o n-ésimo sensor (de 1 a 6). Os sensores são dispostos da esquerda para a direita, de IR_1 até IR_6.

$$S = (0.6IR_1 + 0.25IR_2 + 0.15IR_3) - (0.15IR_4 + 0.25IR_5 + 0.6IR_6) \quad (1)$$

Esta soma ponderada é a referência para determinar quando os dois sensores centrais estão sobre a linha. No caso ideal, o valor é 0, isto é, quando quando IR_3 e IR_4 estão sobre a linha. Quando o carrinho se desloca, ela atualiza seu valor, podendo ser negativo ou positivo conforme a posição dos sensores estar deslocada à esquerda ou direita. A variável S será a referência virtual de distância do veículo, representando o quão longe ela está do valor ideal. Os valores que ponderam os sensores são obtidos a partir de testes empíricos para esta construção de veículo. Após calculado o valor de S, ele deve ser armazenado em uma fila denominada fila "Infravermelho", compartilhada com a tarefa de Controle.

Uma segunda tarefa denominada "Controlador" terá a função de ler da fila Infravermelho o valor de referência calculado na tarefa anterior. Depois disto, deverá realizar a escolha das velocidades dos motores para que os mesmos façam com que o veículo se mantenha em movimento sobre a linha, com base em um controlador PID digital, alterando o valor de saída (velocidades) de acordo com o valor de erro realimentado. Outra função deste tipo de controlador é suavizar as variações da saída para que o sistema se movimente de maneira estável. O valor de saída do PID é enviado para a tarefa de controle do motor através da fila "Acionamento".

Uma terceira tarefa denominada de tarefa "Ultrassom" foi criada para realizar a rotina de leitura dos valores de distância, verificando se existem ou não objetos a frente. O sensor utilizado espera um pulso de 10 microssegundos no seu pino de *trigger* e então avalia a distância através do pino de eco. A duração do pulso no pino eco guarda uma relação com a distância do obstáculo mais próximo. Distâncias muito grandes não geram o pulso, não influenciando no funcionamento. Todo valor lido por esta tarefa é também colocado na fila "Acionamento", estando disponível rapidamente para a tarefa de acionamento.

Finalmente, a quarta e última tarefa, denominada de "Acionamento", é onde de fato será realizado o controle dos motores. O controle da velocidade dos motores se dá pela porcentagem do ciclo de PWM gerado para cada um. Logo, em 100% o motor estaria em sua velocidade nominal e, para cada valor de porcentagem abaixo, a sua velocidade seria reduzida proporcionalmente até chegar a 0%, com motor parado.

Nesta tarefa, o primeiro passo é obter os dados da fila de controle e, com base neles, tomar as decisões necessárias. A prioridade aqui é não deixar que sejam colocadas variações abruptas de tensão sobre o motor. Para isso é realizada uma rampa de tensão com dez degraus. O intervalo de espera entre cada degrau é de 10 microssegundos. A construção da rampa foi feita a partir da comparação do valor atual que deve ser gerado para os motores com os valores anteriores, de acordo com a Equação 2, com a variável i indo de 0 a 9 e V, V_a e V_o sendo, respectivamente, a velocidade atual, a nova velocidade e a velocidade usada previamente.

$$V = V_o + \frac{V_n - V_o}{10 - i} \quad (2)$$

Nesta tarefa também é feita a verificação da distância fornecida pela tarefa do ultrassom, podendo tomar ações rápidas de parada ou de desvio em situações em que o ultrassom reporte algum problema.

IV. SISTEMA IMPLEMENTADO

Para a executar a comparação entre as duas estratégias de construção de firmware, foi construído um veículo (Figura 3) seguidor de trilha, com os seguintes componentes:

- Conjunto de oito sensores infravermelhos QTR-8A, da Pololu [7]. Foram usados 6 sensores, posicionados voltados para o chão, a cerca de 5mm.
- *Driver* de controle de dois motores TB6612FNG, também da Pololu.
- Sensor de ultrassom HC-SR04, também disponível na Pololu, posicionado na frente do veículo e apontando para frente.
- Bateria de 7.2V com capacidade de 1000mAh.
- Regulador de tensão LM7805.
- Dois micromotores com torque máximo de 0.064 Nm e até 1000 RPM de velocidade nominal, acoplados a rodas traseiras.
- Processador Cortex M4F, através de uma placa Freescale FRDM-K64F [8], rodando o sistema mbed e RTOS RTX.
- Rodízio e chassi do veículo.

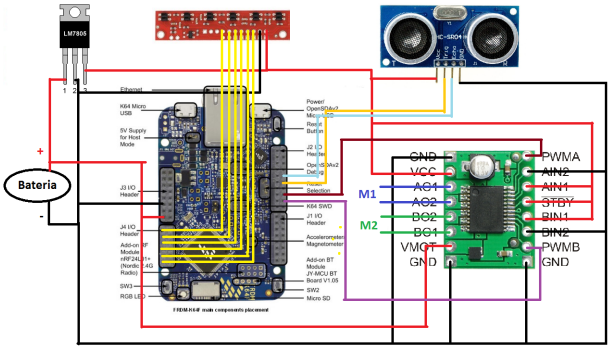

Fig. 3. Montagem realizada

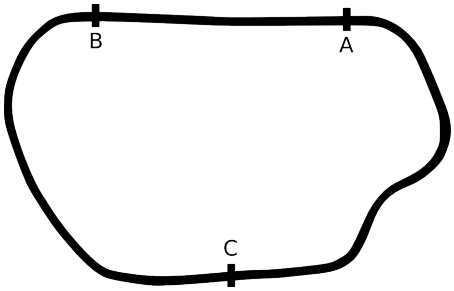

Fig. 4. Pista de testes traçada no chão

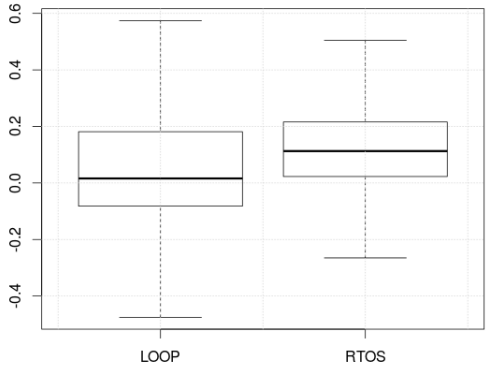

Fig. 5. Boxplot para todo o circuito

Um pequeno circuito foi construído (Figura 4), composta de 3 segmentos. O veículo inicia o percusso no ponto A, percorrendo seu primeiro trajeto que vai deste ponto até o ponto B com a distância de 88 cm, denominado aqui de trecho ou trajeto A. O trecho B inicia no ponto B e vai até o ponto C, composto por uma curva mais aberta, de velocidade mais alta, e apresenta 157 cm de comprimento. E por último, o trecho C, que vai do ponto C até o ponto A, composto por uma curva em forma de "S", de velocidade mais baixa, com 123 cm de percurso.

Foi empregado um controlador PID digital [5], calibrado experimentalmente, através de várias tentativas. Para o caso *super loop*, foram usados $P = 1,15$, $I = 0,00023$ e $D = 0,44$. Para o sistema com RTOS, os valores foram $P = 0,5$, $I = 0,0001$ e $D = 3$.

V. EXECUÇÃO DO EXPERIMENTO E ANÁLISE DOS RESULTADOS

Para cada caso, considerando o sistema com *super loop* e com RTOS, foram realizados 5 voltas. Em cada um das voltas foram amostrados os valores da soma dos sensores infravermelhos, que indica o quão longe a referência está do ponto desejado. Estes valores obtidos foram também separados pelos 3 trajetos descritos previamente afim de comparar os dois casos em distintas situações de percurso. O tempo para percorrer cada trajeto também foi armazenado para comparar o desempenho do RTOS com a programação *super loop* com relação a velocidade.

Para uma análise da distribuição das amostras de distância com relação aos seus quartis, são apresentados gráficos do tipo boxplot, onde a linha central representa a mediana das amostras obtidas para cada caso. O primeiro gráfico (Figura 5) reúne os valores de amplitudes durante todo o trajeto, considerando todas as voltas, para firmwares com *super loop* e RTOS. A distância é calculada segundo a Equação 1. Já a Figura 6 apresenta esta mesma variável para ambos os casos, mas separada por trechos.

Tanto quando se analisa o boxplot total ou por trechos, é visível uma diferença na variabilidade da distribuição dos valores medidos, apresentando a estratégia *super loop* uma maior dispersão. Isto indica não somente a maior capacidade de repetir a atuação, volta a volta, para o sistema com RTOS, mas também uma menor dispersão ao redor da trilha seguida, com consequente incremento em velocidade. Desta forma, no trecho A, o comportamento se reflete em uma maior estabilidade na reta, assim como uma curva de alta velocidade (trecho B) também com uma saída relativamente constante, corrigida pela atuação do sistema.

Nesses mesmos trechos, para o caso *super loop*, pode-se perceber um trecho de reta menos estável (trecho A) e maior variabilidade também em curvas de alta velocidade (trecho B).

Deve-ser observar também que a velocidade do

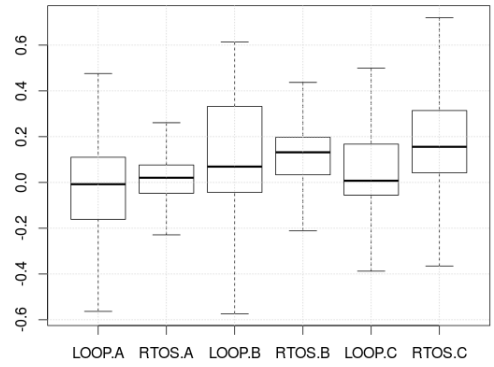

Fig. 6. Boxplot por trechos

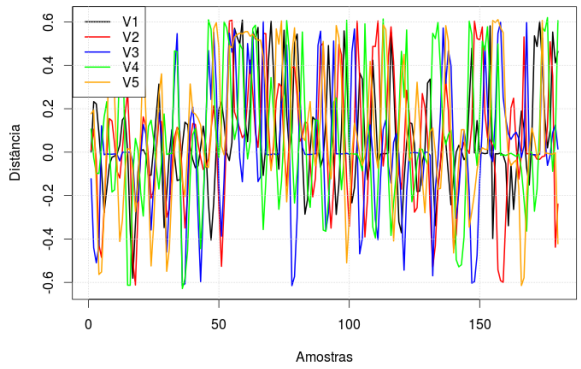

Fig. 7. Cinco voltas com *super loop*

veículo com RTOS é sempre, pelo menos, duas vezes maior (ver Tabela II), o que naturalmente tenderia a gerar erros um pouco maiores quando comparado com um mesmo sistema em velocidade menor.

Já o trecho C é uma parte onde não se infere facilmente o comportamento do veículo diretamente pelo boxplot. Aparentemente, o veículo com *super loop* tem um erro menor. No entanto, a observação do teste evidencia que o veículo com *super loop* tem a tendência de passar reto, não respeitando muito bem a trilha.

Uma possível causa é a menor taxa de execução do loop de controle PID para o caso *super loop*. Isto é facilmente percebido através do número de amostras obtidas por volta. As amostras são geradas a cada execução do algoritmo de PID, com um total aproximado de apenas 180 amostras/volta para o *super loop*, um valor baixo perto das 1761 amostras/volta para o sistema com RTOS. Com a avaliação do erro sendo mais frequente, a correção também acaba sendo e o veículo tem a tendência de seguir melhor o traçado proposto.

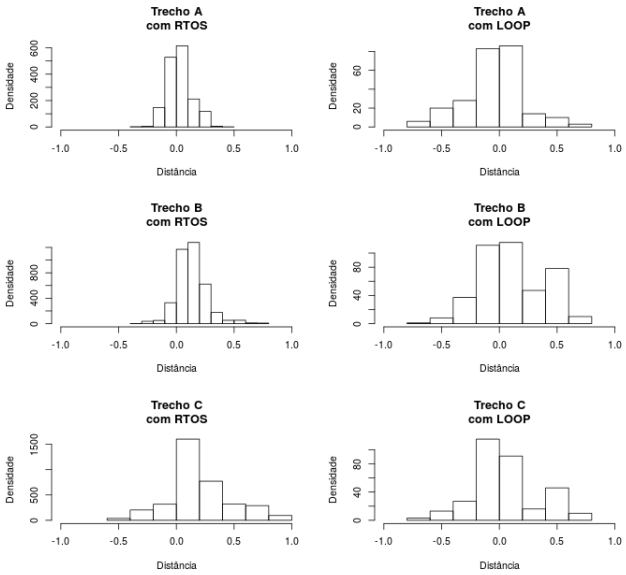

Fig. 8. Histogramas

O efeito dessa sub-amostragem, no trecho C, é não seguir bem o traçado, tendendo a passar reto. Já quando executado com RTOS, além da velocidade maior que traz consigo maiores erros, tem-se também um execução mais fiel do traçado. Infelizmente, não foi possível reproduzir nesse trabalho este comportamento dada a ausência de uma câmera no teste realizado.

Através de uma análise de variância via *Anova One Way* e teste com o emprego do método *Tukey* com nível de confiança de 95%, foi possível verificar que existe uma diferença estatística entre as variações *super loop* e RTOS, quando todos os dados são considerados ($p = 0$) [9].

Quando os dados são avaliados por trecho, foi possível obter diferença estatística no trecho A ($p = 0,0001470$) e no trecho C ($p = 0$). Já no trecho B não foi possível provar nenhuma diferença estatística ($p = 0,8299718$). Não foi confirmado mas acredita-se que, dada uma mesma velocidade para *super loop* e RTOS, exista uma diferença estatística em todos os trechos.

Os histogramas de cada trecho revelam uma estimativa da distribuição (Figura 8) e evidenciam a dispersão das medidas. É possível ver a tendência de erros também, como no caso do trecho B, onde a distribuição apresenta mais valores do lado direito, indicando a tentativa de saída da curva.

No entanto, nem o boxplot ou o histograma revelam como o veículo se comportou no tempo,

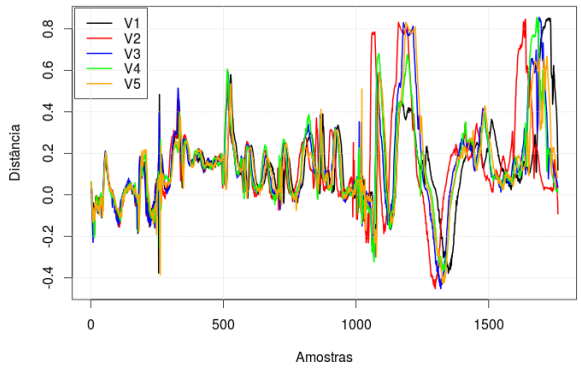

Fig. 9. Cinco voltas com RTOS

sendo interessante visualizar as cinco voltas executas de forma superposta, para os firmware construídos com *super loop* e RTOS.

Para o *super loop*, mesmo em velocidades pelo menos duas vezes menores, a capacidade de o sistema repetir o comportamento da volta anterior fica bastante comprometida (Figura 7). Já o emprego do RTOS revela um comportamento repetitivo (Figura 9), com decisões similares sendo tomadas durante o percurso do veículo.

Percebe-se que ocorre uma leve defasagem à medida que a volta é dada e acredita-se que isso se deva aos pequenos erros que vão se acumulando durante a volta. O enfraquecimento da bateria também é um fator relevante nessa análise.

Finalmente, os dados de velocidade e tempo de volta, assim como os seus valores médios e desvios, são apresentados nas Tabelas I e II. A superioridade do sistema com RTOS é comprovada em todos os trechos.

Tabela I

TEMPO POR TRECHO (MS)

Tempo por trecho (ms) para Super Loop				
Volta	**A**	**B**	**C**	**Total**
1	5941	9357	8113 2	3411
2	6032	9311	7492	22835
3	5214	8254	6848	20316
4	5067	8565	7273	20905
5	5256	10840	6607	22703
Média	5502	9265,4	7266,6	22034
Desvio	401,6	894,8	524,9	1201
Tempo por trecho (ms) para RTOS				
Volta	**A**	**B**	**C**	**Total**
1	1551	3829	3809	9189
2	1521	3718	3481	8720
3	1576	3808	3660	9044
4	1608	3787	3591	8986
5	1605	3810	3665	9080
Média	1572,2	3790,4	3641,2	9003,8
Desvio	33	38,6	107	156,6

Tabela II

VELOCIDADE POR TRECHO (CM/S)

Velocidade por trecho (cm/s) para Super Loop				
Volta	**A**	**B**	**C**	**Total**
1	16,0	16,8	15,2	16
2	14,6	16,9	16,4	15,97
3	16,9	19	18	17,97
4	17,4	18,3	16,9	17,53
5	16,7	14,5	18,6	16,6
Média	16,32	17,1	17,02	16,81
Desvio	1,1	1,5	1,2	0,8
Velocidade por trecho (cm/s) para RTOS				
Volta	**A**	**B**	**C**	**Total**
1	56,7	41	32,3	43,33
2	57,9	42,2	35,3	45,13
3	55,8	41,2	33,6	43,5
4	54,7	41,5	34,3	43,5
5	54,8	41,2	33,6	43,2
Média	55,98	41,42	33,82	43,74
Desvio	1,2	0,4	1	0,7

VI. CONCLUSÃO

Neste trabalho foi apresentado uma forma de avaliar a diferença de comportamento entre veículos seguidores de trilha implementados com RTOS e *super loop*. A partir dos dados apresentados é possível concluir a superioridade do sistema com RTOS em quase todos os aspectos medidos. Enquanto o resultado possa ser, de certa forma, esperado, a contribuição no formalismo que permite a comparação, assim como o exemplo de modelagem, são contribuições relevantes para outros leitores.

REFERÊNCIAS

[1] Q. Li and C. Yao, *Real-Time Concepts for Embedded Systems*. CMP Books.

[2] E. White, *Making Embedded Systems: Design Patterns for Great Software*. O'Reilly Media.

[3] Y. A. Badamasi, "The working principle of an arduino," in *Electronics, Computer and Computation (ICECCO), 2014 11th International Conference on*, Sept 2014, pp. 1–4.

[4] J. Bungo, "Embedded systems programming in the cloud: A novel approach for academia," *IEEE Potentials*, vol. 30, no. 1, pp. 17–23, Jan 2011.

[5] K. Ogata, *Engenharia de Controle Moderno*. Pearson.

[6] RTX real-time operating system. [Online]. Available: http://www.keil.com/rl-arm/kernel.asp

[7] Pololu robotics and eletronics. [Online]. Available: https://www.pololu.com/

[8] FRDM-K64F development board. [Online]. Available: https://developer.mbed.org/platforms/FRDM-K64F/

[9] S. M. Ross, *Introduction to Probability and Statistics for Engineers and Scientists*. Elsevier.