Relatório do Laboratório 2 de Sistemas de Tempo Real

André Dornelas Sanches
170099369@aluno.unb.br

Hiago dos Santos Rabelo
160124492@aluno.unb.br

1. Objetivos

Este laboratório tem como objetivo o desenvolvimento de uma aplicação que utilize conceitos de sistemas de tempo real. Assim, propõe-se um sistema crítico de tempo real para monitoramento e acionamento de um sistema de recalque de água com envio de dados via web.

2. Introdução

2.1. Introdução teórica

Um sistema com requisitos temporais no sentido da necessidade de se realizar uma tarefa usando o tempo disponível e corretamente é chamado de sistema de tempo real. Portanto, nesse tipo de sistema uma tarefa ser realizada de maneira correta é tão importante quanto ser finalizada no tempo certo. Esses sistemas estão presentes desde aeronaves até o mercado financeiro.

Sistemas de tempo real tem forte vínculo com o meio físico, dado que é necessário receber algum estímulo para realizar alguma tarefa ou tomada de decisão. Para entender melhor o funcionamento desse tipo de sistema é preciso entender alguns conceitos importantes como periodicidade das tarefas, deadline e precedência.

Além disso, fica estabelecido que uma tarefa é algo a ser feito perante algum estímulo recebido, como a chegada de dados de um sensor de nível de água. Portanto, tarefas podem ser periódicas, esporádicas ou aperiódicas. Assim:

- Tarefas Periódicas: São tarefas que a ativação ocorre com intervalo de tempo determinado.
- Tarefas Esporádicas: São tarefas cuja ocorrência é imprevisível, porém há um período mínimo entre as ocorrências.
- Tarefas Aperiódicas: São tarefas que nada se sabe sobre sua ativação, ou seja, o período de tempo entre ocorrência é incerto.

Já o deadline, que é o prazo máximo para uma tarefa ser finalizada, pode ser dividido em três tipos, são eles:

- Deadline Hard: Ocorre quando a perda do prazo pode gerar consequências catastróficas.
- Deadline Firm: Ocorre quando a perda do prazo não gera consequências catastróficas e não existe valor em terminar a tarefa após o prazo.
- Deadline Soft: Ocorre quando a perda do prazo não gera consequências catastróficas e existe valor ainda que a tarefa tenha finalizado após o prazo.

O conceito de precedência está relacionado com dependências das tarefas uma com as outras, havendo a necessidade de garantir a execução de uma tarefa antes de poder executar a outra, porém esse conceito não foi utilizado neste projeto.

Com esses conceitos como base, o que se busca neste trabalho é implementar um sistema capaz de monitorar os níveis dos reservatórios de água e estabelecer as tarefas e prazos para realizar o recalque de água de um reservatório inferior a um reservatório superior.

2.2. Recalque de água

O sistema de recalque de água é aquele que, por meio da elevação de uma pressão bombeada, ajuda o líquido a fluir em uma tubulação até determinada altura, como representado na Figura 1 abaixo. Normalmente, esse sistema de recalque de água é usado para o transporte da água de um reservatório inferior a outro em um nível mais alto que não pôde ser alimentado diretamente devido a pressão insuficiente de enchimento.

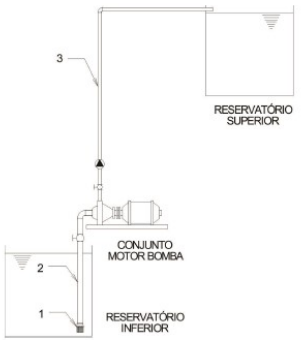

Figura 1. Esquemático de um sistema de recalque de água.

Assim, o sistema de recalque de água proposto neste trabalho fará uso de reservatórios de água, bomba de água,

sensores de nível, sistema de processamento e de exposição de dados com o objetivo de emular um sistema encontrado em condomínios residenciais, por exemplo. O detalhamento dos materiais utilizados é apresentado na seção seguinte.

3. Materiais e métodos

Inicialmente, o sistema foi projetado para a utilização de uma mini bomba de água de 12. Onde, é necessário levar em consideração o requisito temporal do enchimento do reservatório superior precisar ser feito em até 1.5 minuto. Atendendo a esse requisito temporal, opta-se por uma bomba sem grau de proteção IP para líquidos, possibilitando uma redução de custo de projeto visto que não existe a necessidade da bomba permanecer submersa. Como a coluna máxima do nível de água pra o protótipo não ultrapassa 2 metros, foi escolhida a mini bomba de água com motor RS-385, mostrada na Figura 2, que será utilizada para levar a água do reservatório inferior para o superior. A alimentação da parte do motor é feita através da bateria de 12v mostrada na Figura 3.

Figura 2. Mini bomba de água 12V

Durante a implementação do projeto, deparou-se com um alta interferência eletromagnética gerada pelo motor, o qual impossibilitava a utilizava a utilização do circuito montado. Devido a isso, realizou-se uma troca por um motor com uma corrente consumida menor. Como substituto para a modelagem da tarefa eletromecânica, foi escolhido o micro motor de corrente contínua, mostrado na Figura 4. O micro motor necessita de uma alimentação de 3 a 6v, o qual consome 1,6 A em stall. Devido a isso, a bateria de 12v, mostrada na Figura 3, foi substituída por uma bateia de 6v, a qual é mostrada na Figura 5.

O módulo do microcontrolador utilizado é um ESP32

Figura 3. Bateria lipo de 12V

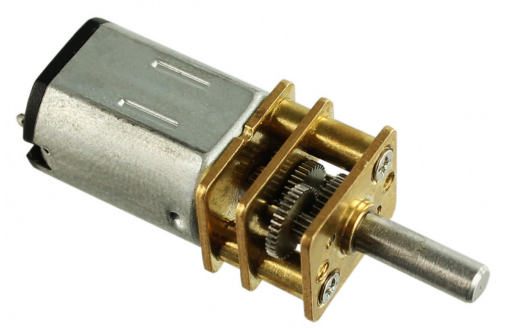

Figura 4. Micro motor 6V

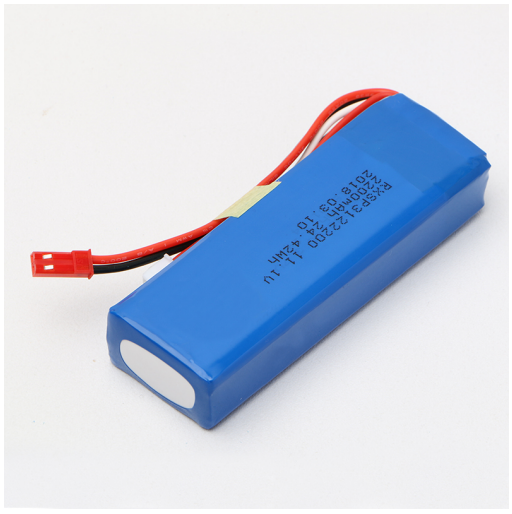

Figura 5. Bateria lipo de 6V

DevKit V1, o qual possui CPU Xtensa® Dual-Core 32-bit, conexão wi-fi de 2.4GHz e bluetooth BLE 4.2 integrados. Para possibilitar o acionamento no motor, é utilizado um relê eletromecânico selado com 2 polos reversíveis, o qual

suporta 2A ao ser alimentado com 24V.

3.1. Tarefas

Para o desenvolvimento do sistema, são necessárias as implementações das tarefas concorrentes em um sistema microkernel simples. As tarefas são:

- **Monitoramento do motor:** É a tarefa responsável por ler os sensores referentes ao motor: o medidor de corrente (ACS712-5A) e o de temperatura (DHT22). Durante a execução dessa tarefa, fez-se necessário a utilização de um *Mutex* para proteger a variável global utilizada, uma struct que continha os valores de temperatura e de corrente. Portanto, usou-se duas funções, *Get()* e *Set()*, que ao serem chamadas, bloqueiam o acesso à variável global *MotorInfo* para garantir que não esteja sendo utilizada na escrita no dashboard ou no envio para web. Caso os valores lidos de temperatura e corrente ultrapassem limiares definidos pelo projetista do sistema, a bomba é desligada e mantida assim até os os valores lidos fiquem abaixo dos limiares, sendo essa ação realizada mesmo que seja solicitado um acionamento manual.

- **Botões:** Sendo na verdade dividida em três tarefas, uma para cada botão: Stop, Manual e Automático. A tarefa é responsável por alterar o modo do sistema de acordo com o botão pressionado, o qual é chamado através de uma interrupção de subida e verificado sua consistência por meio de um parâmetro de *debounce*. Este parâmetro serve para suavizar o pressionamento do botão, não permitindo que seja possível apertá-lo mais de uma vez a cada 500ms, logo, o período dessa tarefa tem de ser ao menos 500ms. A tarefa do botão STOP deve ser a de maior prioridade pois o botão é usado como emergência para parar o motor.

- **Envio para web:** A partir do monitoramento de temperatura e corrente consumida pela bomba, esses dados são enviados para um IP via *POST*. Também é anexada a informação de nível do tanque para o envio.

- **Modo automático:** Responsável por acionar a bomba caso as variáveis associadas ao nível do reservatório indiquem que o tanque está vazio ou desligar o motor caso o nível do tanque indique que está cheio. Neste modo o botão de acionamento manual do motor não está habilitado.

- **Modo manual:** Responsável por fornecer ao usuário a escolha de quando a bomba deve ou não ser acionada e parada. Os botões de acionamento devem ser lidos nesse modo.

- **Dashboard de visualização:** Através de um display LCD, mostra os valores associados às variáveis de temperatura do motor, corrente consumida, nível de água e modo operante do sistema.

- **Entradas do usuário:** Realiza a leitura dos botões de acionamento, parada, modo de atuação e parada de emergência. Necessária a implementação de uma exclusão mútua nessa etapa.

3.2. Métodos

Inicialmente foi pensado que o sistema poderia ser implementado na forma de um executivo cíclico com tratador de interrupções, já que a tarefa mais crítica seria o botão de emergência, que possui um *deadline hard*. Porém, como a leitura e escrita das variáveis de nível de água acontecem de forma rápida e concorrente e também atuam sob o funcionamento do motor, indicando quando ele deve ligar ou desligar, foi decidido que o sistema mais apropriado seria um microkernel.

A utilização de um *microkerknel* no sistema é feita através do uso da biblioteca FreeRTOS. Dentre as várias ferramentas a qual o *FreeRTOS* oferece suporte, destacam-se o MUTEX e *multitask*. O MUTEX é utilizado amplamente utilizados no trabalho para gerenciar as seções críticas acessadas pelas funções de leitura e escrita de dados.

O suporte para criação de *multitasks* é utilizado para permitir que sejam definidas tarefas com prioridades e períodos variados, permitindo assim que tarefas de mais alta prioridade, como por exemplo o gerenciamento da bomba, tenham a sua execução priorizada em detrimento da execução de tarefas de mais baixa prioridade, como por exemplo a tarefa de envio de dados para a *cloud*.

4. Resultados

A função *gettimeofday* foi utilizada para medir o momento prévio antes da execução das linhas de código dentro da tarefa. É realizada uma segunda chamada após a execução da última instrução da tarefa e em seguida utilizada a função *timersub* para obter o tempo de resposta de cada uma das tarefas analisadas. Para a coleta dos tempos de resposta medidos, esses valores foram enviados para a porta serial do ESP32, os quais foram lidos no computador através de um *script* em python e salvos em um arquivo txt.

Seguindo a lista de prioridades e períodos da tabela 1 para o sistema de tempo real utilizando o FreeRTOS, obtêm-se o gráfico da figura em 6. Na tabela, as informações de período para os botões não existem devido a ser uma tarefa esporádica, porém, possui um intevalo mínimo entre as chamadas de 500 milissegundos devido ao *debouncing* utilizado. O *debouncing* é a remoção de ruídos indesejados no sinal coletado pelos botões. Uma forma efetiva de realizar essa remoção é incorporar um tempo mínimo entre cada

Tarefa	Prioridade	Período (ms)
HANDLE STOP	1	X
PUMP	2	500
MEASURE MOTOR	3	2000
MEASURE WATER	4	500
HANDLE AUTOMATIC	5	X
HANDLE MANUAL	6	X
LCD	7	500
UPLOAD STATUS	8	2000

Tabela 1. O valor X na coluna de período denota as tarefas esporádicas.

leitura para possibilitar que os ruídos gerados pela ativação dos botões sessem.

Pode ser notado no gráfico da Figura 6 que nenhuma tarefa ultrapassa o tempo de resposta medido de 1 segundo. Também é importante ressaltar que o tempo de reposta máximo medido de cada tarefa diminui a medida que a prioridade da tarefa aumenta, onde a única tarefa que quebra essa exceção é a *HandlePump*. Tal comportamento na tarefa mencionada reflete o caso em que solicita as informações de temperatura e corrente do motor, porém a tarefa é bloqueada devido à não poder acessar a variável devido ao *lock* realizado pela tarefa *MeasureMotor*. O *lock* realizado pela tarefa *MeasureMotor* evita que alguma tarefa tente acessar as informações do motor enquanto ela está atualizando os valores, onde essa proteção da seção crítica protege contra a leitura incorreta de valores.

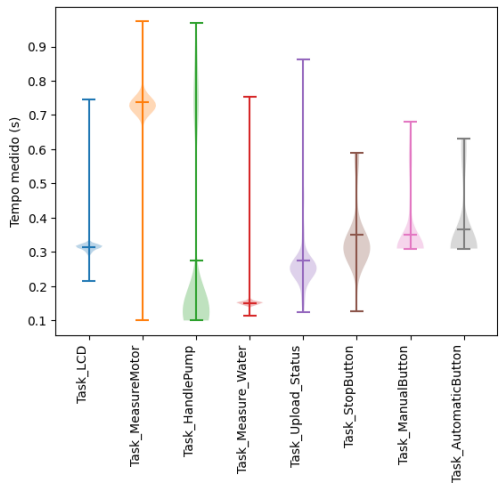

Figura 6. Violinos com a informação de mínimo, máximo, média e distribuição do tempo de resposta medido de cada uma das tarefas.

Os resultados detalhados são mostrados nas subseções a seguir.

4.1. Tarefa LCD

É a tarefa responsável por coletar a informação de estado desejado pelo usuário, onde os estados podem assumir o modo automático, manual e stop, e acionar a bomba ou pará-la de acordo com o estado desejado. Essa tarefa também é responsável por acionar ou parar a bomba de acordo com o nível de água, caso o sistema esteja em modo automático. Por fim, independentemente o modo ao qual o sistema esteja, a tarefa também coleta informações de temperatura e corrente do motor, as quais são utilizadas para parar a bomba caso ultrapassem valores pre-definidos.

O histograma criado a partir dos tempos de resposta medidos, mostrado na Figura 8, em conjunto com a visão do tempo de resposta para cada caso de teste, mostrado na figura7, apresentam a baixa variabilidade do tempo de resposta da tarefa de mostrar as informações para o usuário.

Obtido um HWM(100) de 744 milissegundos, HWM(99) de 361 milissegundos para 1207 amostras coletadas, o valor de HWM(99) se mostra plausível para o deadline da tarefa analisada visto que existe uma baixa variabilidade do tempo de resposta medido para os casos de teste utilizados.

Para o deadline proposto foi obtido um *factor-skip* de 34 perdas de deadlines dentre as 1207, o que é algo plausível visto a baixa prioridade da tarefa e sua não criticidade no sistema. O *factor-skip* S mede a mínima quantidade de ativações da tarefa entre 2 perdas consecutivas de deadlines.

A partir do deadline proposto, utiliza-se uma janela deslizante de tamanho 20 para a análise do parâmetro *mk-firm*. O *mk-firm* utiliza uma janela deslizante de tamanho k para analisar quantas m ativações da tarefa foram realizadas respeitando o deadline. Para a tarefa analisada, foi obtido um *mk-firm* de (18,20), onde para cada 20 ativações são perdidos apenas 2 deadlines. Essa perda de deadlines na tarefa de mostrar as informações para o usuário é algo tolerado visto que não será percebida visualmente.

4.2. Tarefa Gerencia Bomba

É a tarefa responsável por coletar informações do sistema e acionar a bomba. O histograma criado a partir dos tempos de resposta medidos, mostrado na 10, em conjunto com a visão do tempo de resposta para cada caso de teste, mostrado na Figura 9, apresentam a baixa variabilidade do tempo de resposta da tarefa de mostrar as informações para o usuário.

Obtido um HWM(100) de 274 milissegundos, HWM(99) de 273 milissegundos para 1284 amostras coletadas, o valor de HWM(99) se mostra plausível para o deadline da tarefa analisada visto que existe uma alta variabilidade do tempo de resposta medido para os casos de teste utilizados.

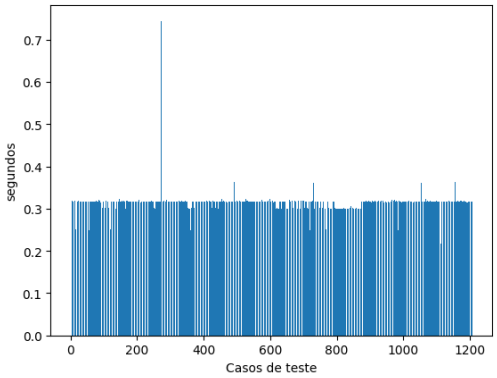

Figura 7. Tempos de resposta medidos para a tarefa responsável por mostrar as informações do sistema ao usuário através de um display LCD.

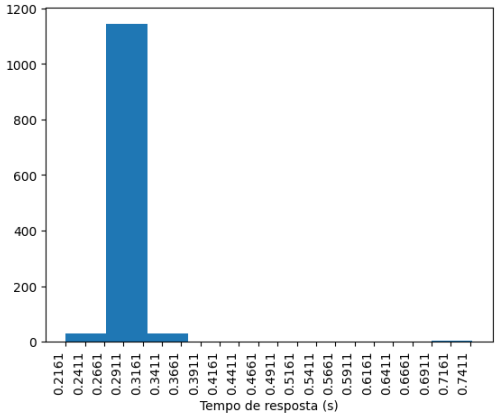

Figura 8. Histograma da tarefa responsável por dar get nas informações do sistema de mostrar no display LCD.

Para o deadline proposto foi obtido um *factor-skip* de 68 perdas de deadlines dentre as 1284 amostras, o que não é a princípio pode parecer algo não tolerado devido a criticidade desse tarefa.

Para um estudo mais detalhado, utiliza-se o deadline proposto e uma janela deslizante de tamanho 20 para a análise do parâmetro *mk-firm*. Para a tarefa analisada, foi obtido um *mk-firm* de (19,20), onde para cada 20 ativações é perdido 1 deadline.

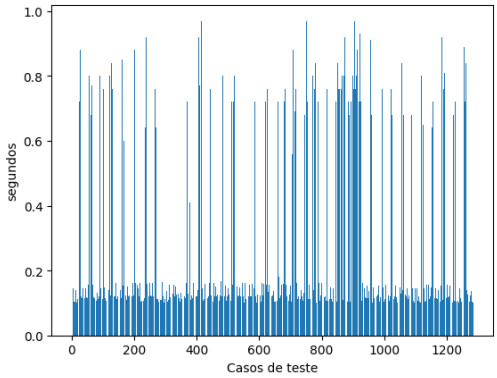

Figura 9. Tempos de resposta coletados para a tarefa responsável por gerenciar o acionamento da bomba.

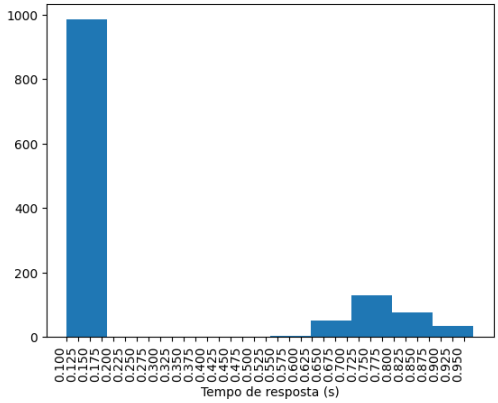

Figura 10. Histograma da tarefa responsável por dar get nas informações de temperatura, corrente, altura media pelo ultrassom e botões pressionados, e a partir disso decidir se liga ou desliga o motor.

4.3. Tarefa Botão Stop

Embora a tarefa não tenha sido implementada como um tratado de interrupção, atende aos requisitos temporais desejado de um

As medições do tempo de resposta para 41 amostras resultaram em um HWM(100) de 590 milissegundos e HWM(99) de 590 de milissegundos. É possível observar que para um deadline de 590 milissegundos não existe perda de deadlines. Os gráficos são mostrados nas figuras 11 e 12.

4.4. Tarefa Botão Manual

Embora a tarefa não tenha sido implementada como um tratado de interrupção, atende aos requisitos temporais desejado de um sistema de tempo real.

A partir das medições do tempo de resposta para 71 amostras, tem-se um um HWM(100) de 680 milissegundos

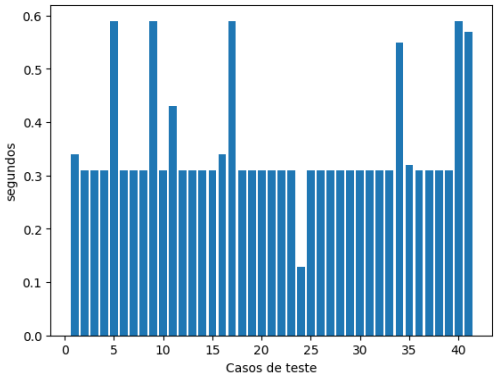

Figura 11. Tempo de resposta da tarefa liberada para execução após o mutex de stop ter sido liberado pela interrupção gerada a partir do aperto do botão stop.

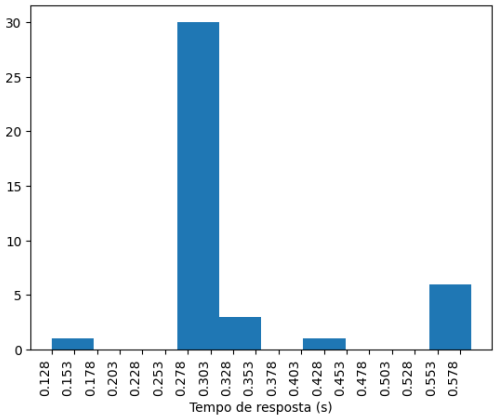

Figura 12. Histograma da tarefa tempo de resposta do tratador de interrupção gerado a partir do aperto do botão stop.

e HWM(99) de 680 de milissegundos. É possível observar que para um deadline de 680 milissegundos não existe perda de deadlines. Os gráficos são mostrados nas figuras 13 e 14.

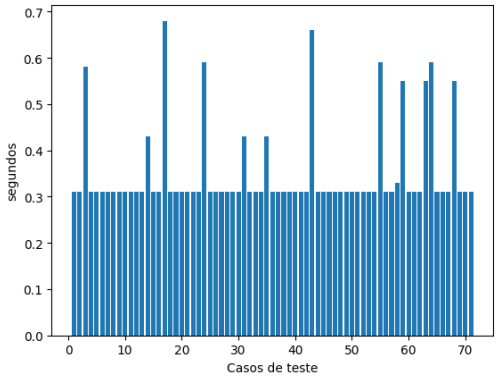

Figura 13. Tempo de resposta da interrupção gerada a partir do aperto do botão manual.

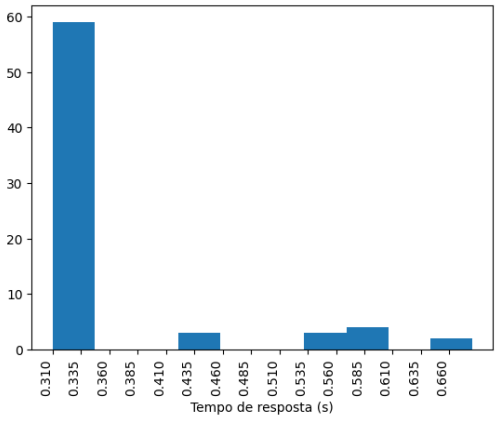

Figura 14. Histograma da interrupção responsável gerpor inserir o estado ”Manual”na variável global de botão, caso o mesmo tenha sido pressionado. Também insere a quantidade de vezes que o botão foi pressionado.

4.5. Tarefa Botão Automático

Embora a tarefa não tenha sido implementada como um tratado de interrupção, atende aos requisitos temporais desejado de um sistema de tempo real.

A partir das medições do tempo de resposta para 37 amostras, tem-se um um HWM(100) de 630 milissegundos e HWM(99) de 630 de milissegundos. É possível observar que para um deadline de 630 milissegundos não existe perda de deadlines. Os gráficos são mostrados na Figura 15 e Figura 16.

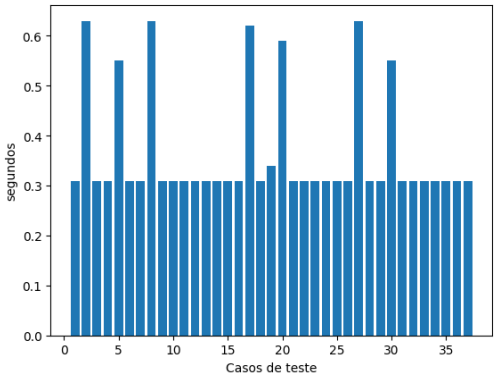

Figura 15. Template da tarefa de monitorar a chave de modo de acionamento do sistema.

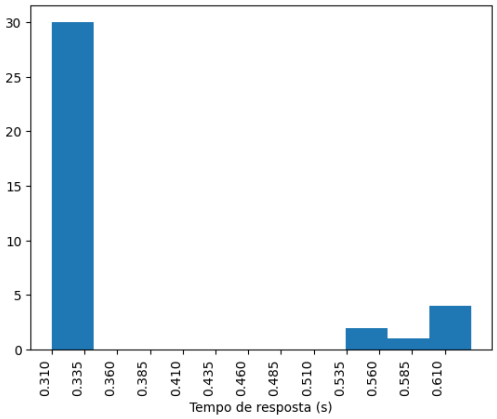

Figura 16. Histograma da tarefa responsável por inserir o estado *Automático* na variável global de botão, caso o mesmo tenha sido pressionado. Também insere a quantidade de vezes que o botão foi pressionado.

4.6. Tarefa mede motor

É a tarefa responsável por coletar informações do sistema e atualizar a struct global para que possa ser usada em tarefas ao usarem o método get.

O histograma criado a partir dos tempos de resposta medidos, mostrado na Figura 18, em conjunto com a visão do tempo de resposta para cada caso de teste, mostrado na Figura 17, apresentam a baixa variabilidade do tempo de resposta da tarefa. A variabilidade que está presente nos gráficos é devido ao bloqueio da tarefa durante a leitura dos valores dos valores por tarefas de mais baixa prioridade.

Obtido um HWM(100) de 973 milissegundos, HWM(99) de 969 milissegundos para 320 amostras coletadas, o valor de HWM(99) se mostra plausível para o deadline da tarefa analisada visto que existe uma baixa variabilidade do tempo de resposta medido para os casos de teste utilizados.

Para o deadline proposto foi obtido um *factor-skip* de 9 perdas de deadlines dentre as 320 amostras. A partir do deadline proposto, utiliza-se uma janela deslizante de tamanho 20 para a análise do parâmetro *mk-firm*. Para a tarefa analisada, foi obtido um *mk-firm* de (17,20), onde para cada 20 ativações são perdidos 3 deadlines.

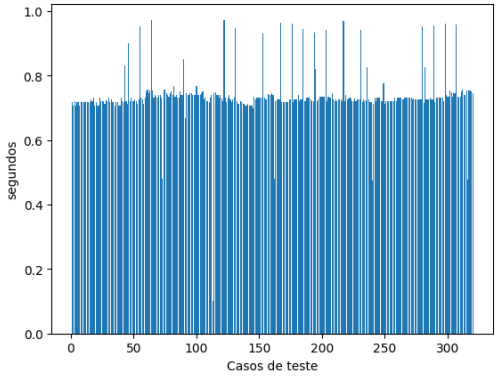

Figura 17. Tempos de resposta coletados para a tarefa responsável por coletar informações de temperatura e corrente do motor.

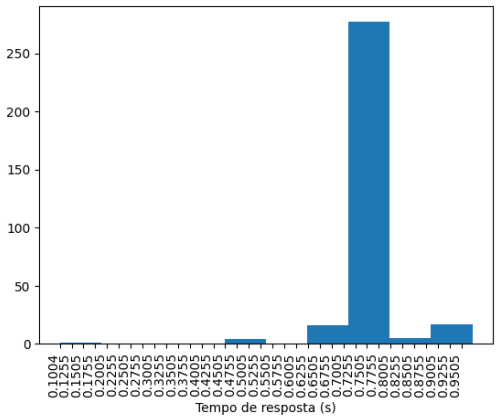

Figura 18. Histograma da tarefa responsável por coletar a temperatura e corrente do motor e atualizar a struct global para que possa ser acessado por get.

4.7. Tarefa mede altura

É a tarefa responsável por coletar informações do sistema e atualizar a struct global do nível de água para que possa ser usada em tarefas ao usarem o método get.

O histograma criado a partir dos tempos de resposta medidos, mostrado na Figura 20, em conjunto com a visão do tempo de resposta para cada caso de teste, mostrado na Figura 19, apresentam a baixa variabilidade do tempo de resposta da tarefa. A variabilidade que está presente nos gráficos é devido ao bloqueio da tarefa durante a leitura dos valores dos valores por tarefas de mais baixa prioridade.

Obtido um HWM(100) de 754 milissegundos, HWM(99) de 160 milissegundos e HWM(95) de 153 milissegundos para 1244 amostras coletadas, o valor de HWM(95) se mostra plausível para o deadline da tarefa analisada visto que existe uma baixa variabilidade do tempo de resposta medido para os casos de teste utilizados.

Para o deadline proposto foi obtido um *factor-skip* de 4 perdas de deadlines dentre as 1244 amostras. A partir do deadline proposto, utiliza-se uma janela deslizante de tamanho 20 para a análise do parâmetro *mk-firm*. Para a tarefa analisada, foi obtido um *mk-firm* de (16,20), onde para cada 20 ativações se perde apenas 4 deadline.

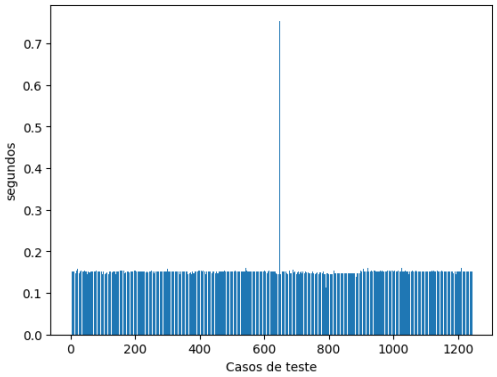

Figura 19. Tempos de resposta coletados para o caso de teste na tarefa responsável por coletar o nível da água através do ultrassom.

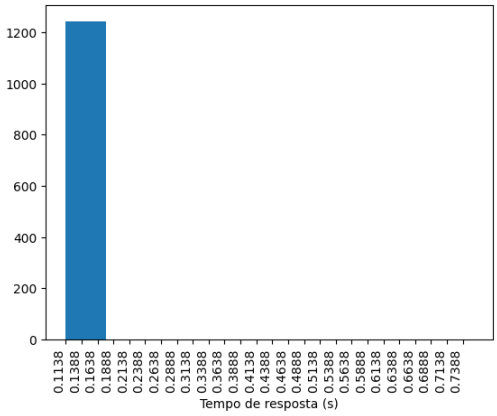

Figura 20. Histograma da tarefa responsável por coletar a medida do sensor ultrassônico e atualizar a variável global para que possa ser acessado por get.

4.8. Tarefa upload infos

É a tarefa responsável por coletar informações do sistema e enviá-las para a *cloud* para uma posterior utilização em um dashboard online para acesso remoto.

O histograma criado a partir dos tempos de resposta medidos, mostrado na Figura 22, em conjunto com a visão do tempo de resposta para cada caso de teste, mostrado na Figura 21, apresentam a alta variabilidade do tempo de resposta da tarefa devido a intereferência causada por tarefas de mais alta prioridade e pelo interefência causada pelos mutex durante a solicitação de leitura dos dados.

Obtido um HWM(100) de 863 milissegundos, HWM(99) de 824 milissegundos e HWM(95) de 458 para 282 amostras coletadas, o valor de HWM(95) se mostra plausível para o deadline da tarefa analisada visto que existe uma alta variabilidade do tempo de resposta medido para os casos de teste utilizados.

Para o deadline proposto foi obtido um *factor-skip* de 1 perdas de deadlines dentre as 282, o que é algo plausível visto a baixa prioridade da tarefa e sua não criticidade no sistema.

A partir do deadline proposto, utiliza-se uma janela deslizante de tamanho 20 para a análise do parâmetro *mk-firm*. Para a tarefa analisada, foi obtido um *mk-firm* de (16,20), onde para cada 20 ativações são perdidos apenas 4 deadlines. Essa perda de deadlines na tarefa de envio de informações na nuvem é algo tolerado visto que não será percebida visualmente.

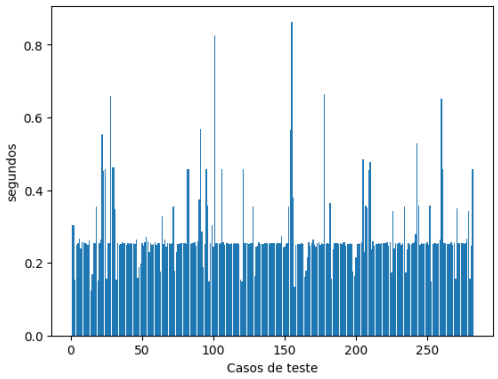

Figura 21. Tempos de resposta coletados para a tarefa responsável por envio de informações do sistema usando método *POST*.

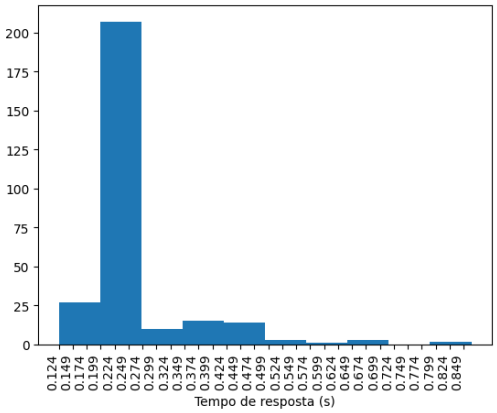

Figura 22. Histograma da tarefa responsável por dar get nas informações de temperatura, corrente, altura media pelo ultrassom, e enviar esses dados para a cloud.

As subfiguras contidas em Figura 23 mostram o dashboard visualizado de forma remota pelo usuário.

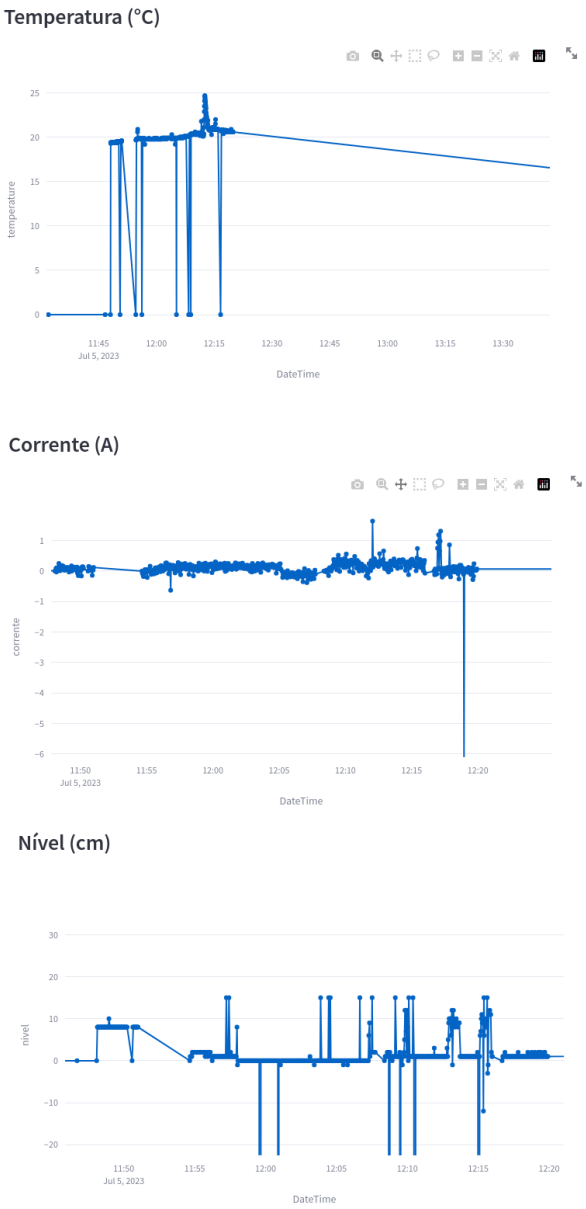

Figura 23. Informações mostradas de forma remota.

A partir das informações obtidas nos datasheets e também por estipulação de um consumo de água fictício, restringiu-se o valor para cada uma das variáveis controladas, sendo eles:

- O nível de água do reservatório superior precisa ficar entre 20% e 80%
- A temperatura máxima do motor deve ser de 50ºC
- A corrente máxima deve ser de 2A.

Com essas informações, usou-se a aba *Verifier* para verificar se haveria deadlock. O resultado pode ser visto na

Figura 24, onde é possível observar que não está previsto nenhum deadlock para os valores de entrada aleatórios gerados pelo software.

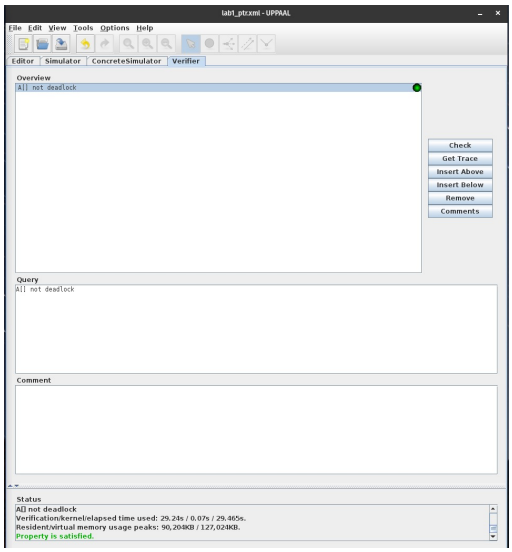

Figura 24. Simulação do sistema para verificação de existência de deadlock.

5. Discussões e conclusões

A simulação através do *UPPAAL* possibilitou a estimativa de um sistema que executaria sem deadlocks a partir de variáveis aleatórias, além de viabilizar os requisitos temporais de deadlines e garantir que a escrita, leitura e atualização dos dados sejam respeitadas.

Apesar da resposta positiva, sabe-se que para um modelo real existem outras dificuldades e atrasos que são difíceis de serem modelados sem efetivamente construir o projeto.

O sistema proposto foi construído e apresentando os resultados esperados para todas as tarefas inicialmente desejadas.

Para trabalhos futuros relacionados à uma atualização do sistema de recalque de água, propõe-se a implementação da tarefa de parar o motor como um tratador de interrupções caso o botão de stop tenha sido apertado.

6. Bibliografia

Uppaal SMC Tutorial, Alexandre David, Kim G. Larsen, Axel Legay, Marius Mikučionis, Danny Bøgsted Poulsen. International Journal on Software Tools for Technology Transfer (STTT), January 2015, Volume 17, Issue 4, pp 397-415. Springer Berlin Heidelberg. ISSN 1433-2787, 1433-2779.