Aplicação do RTOS no projeto Sistema Cyber Físico de Eficiência Energética e Conforto ambiental

Rafael Ifanger Ribeiro[1], **Antonio Pestana Neto**[1]

riribeiro@cti.gov.br, antonio.pestana@cti.gov.br

[1]**Divisão de Robótica e Visão Computacional – DRVC**
CTI/MCTI Renato Archer – Campinas/SP

Abstract. *Handling environmental comfort depends on various structures, ranging from acquisition of data with multiple sensors to data storage, comprehension, and inference. Since the goal of the project is to manage multiple threads, and the time they will be executed is of the utmost importance - due to periodical reading measurements - an RTOS (Real-Time Operating System) kernel was proposed as a solution to control the execution of the tasks; in this case, freeRTOS. This article presents a brief explanation of the concept of RTOS and the manner in which it was applied to the project, as well as the platform developed in order to handle the data gathered, aiming to identify energy consumption and estimate environmental comfort.*

Resumo. *O tratamento do conforto ambiental depende de várias estruturas, desde a aquisição de dados com múltiplos sensores até armazenamento, compreensão e inferência desses dados. Em vista que o objetivo do projeto é gerir múltiplas tarefas e o tempo em que elas serão executadas é de extrema importância - devido à medições periódicas - um escalonador RTOS (Real-Time Operating System) foi proposto como solução para o controle da execução das tarefas; neste caso, freeRTOS. Esse artigo apresenta uma breve explicação dos conceitos de RTOS e a maneira que foi aplicado no projeto, assim como a plataforma desenvolvida para tratar os dados coletados, visando identificar o consumo de energia e estimar o conforto ambiental.*

1. Introdução

O conforto ambiental influencia o bem-estar de um indivíduo em local fechado e, por conseguinte, a qualidade de trabalho que este realiza. Há maneiras para estimar esse conforto, que depende das variáveis físicas do ambiente, como a umidade relativa do ar, a troca de calor por radiação, a velocidade do ar e a temperatura, por exemplo.

Esse artigo relata o desenvolvimento de um sistema cyber físico para monitoramento das condições ambientais de uma sala fechada com proposta de aplicação do sistema operacional RTOS (*Real-Time Operating System* - Sistema Operacional de Tempo-Real) para controle da execução de tarefas, além da medição de fatores energéticos de equipamentos e buscar eficiência energética em seu uso. Procura-se também como objetivo uma aplicação de baixo custo para facilitar a replicação.

O monitoramento depende, primeiramente, da aquisição de dados dos sensores presentes na sala monitorada. Criou-se módulos de sensoriamento com os microcontroladores ESP8266 e ESP32 e sensores de umidade, temperatura, concentração de gás, corrente e tensão. A leitura feita pelos microcontroladores ocorre

periodicamente em intervalos pré definidos no código em execução e as informações são enviadas por protocolo MQTT em formato JSON para uma plataforma desenvolvida com *Node-Red* que armazena, interpreta e apresenta graficamente os dados.

A importância do controle ambiental também está prescrito na Consolidação das Leis Trabalhistas, como por exemplo, a obrigatoriedade do uso de ventilação artificial para situações que a ventilação natural não esteja de acordo com o conforto térmico do indivíduo.

Para analisar o conforto ambiental existem diversos métodos que podem ser combinados para aumentar a confiabilidade do sistema, tanto por uma questão de redundância quanto a possibilidade das informações que resultam de cada método se complementarem.

2. Conceitos do sistema operacional de tempo real - RTOS

Para aplicações cujo tempo de acionamento é mais importante que a realização de atividades em massa, o sistema operacional de tempo-real se adequa devido à capacidade de manipulação de tarefas com base no tempo de início e uma baixa faixa de atraso, além do gerenciamento multitarefa. No projeto foi utilizado um dos sistemas de RTOS mais empregado, o sistema de código aberto freeRTOS [1].

A respeito do sistema operacional, *scheduler*, *tasks* e seus estados são conceitos necessários para compreensão do RTOS, mais especificamente com a abordagem do freeRTOS:

2.1 Tasks (tarefas)

Atividades específicas efetuadas pelo sistema embarcado e categorizadas pela prioridade em relação às demais. O *scheduler* utiliza o valor da prioridade para definir a próxima *task* a ser executada, selecionando a de maior prioridade. Na criação da tarefa com a função *xTaskCreate*, define-se como alguns dos parâmetros o tamanho da pilha (*stack*) da função que deve ser alocada na memória e a prioridade da task, valor inteiro limitado por 0 e (*configMAX_PRIORITIES* – 1), *configMAX_PRIORITIES* uma macro que configura a faixa máxima de inteiros para a prioridade.

2.2 Scheduler

Escalonador, ou agendador, é a parte do software responsável pela decisão de quais tarefas devem ser executadas, definindo a ordem de prioridade das *tasks* no consumo da CPU (*Central Processing Unit* - central de processamento que executará as tasks), que implica na ordem de seleção da lista de tasks a serem realizadas, sendo a atividade com maior prioridade escolhida primeiramente. Entre os algoritmos de trabalho do scheduler, pode-se separar os preemptivos e cooperativos. Os algoritmos preemptivos consideram a interrupção de uma *task* antes do tempo previsto para que outra prioritária possa utilizar a CPU. Essa descontinuação é seguida por uma troca de contexto, recuperação do estado dos registradores da CPU para que seja retomada a *task* futuramente.

Pela contagem do tempo a partir de um relógio, a preempção remaneja as tarefas com base na reavaliação das prioridades em intervalos de tempo.

Haja visto o gerenciamento do sistema, as tarefas ociosas, denominadas *idle*, sempre estarão na lista de execução, no entanto, com menor prioridade de modo que as outras *tasks* possuam maior prioridade. Um exemplo de tarefa *idle* é o gerenciamento de memória RAM.

A divisão do tempo de uso da CPU em intervalos é chamada *Time Slicing* (fatiamento de tempo), a qual designa o tempo de uma unidade básica de tempo de alocação na CPU às tarefas e, após esse período, há a preempção para definir a execução da próxima *task*, havendo troca de contexto caso a tarefa não esteja finalizada. Não existindo outra tarefa, o escalonador retornará à *task* executada antes da preempção. No projeto, o algoritmo preemptivo com *Time Slicing* é usado pelo *scheduler*.

O algoritmo cooperativo, diferente do preemptivo, não interrompe a tarefa durante sua execução. Dessa maneira, a *task* permanece em execução até que seja finalizada ou exija sua troca de contexto para que a próxima entre no estado de execução.

3.1 States

Estados em que uma *task* pode se encontrar. Estes estados são alterados com a chamada de funções ou eventos, sendo o estado em que a tarefa se encontra em execução pela CPU denominado *Running state*. O número de *tasks* no *Running state* depende de quantas CPUs estão livres para uso.

O *Blocked state* é o estado em que a tarefa está parada esperando que algum evento ocorra e não pode ser selecionada pelo scheduler. Por exemplo, caso outras *tasks* estejam sendo executadas e não há CPU disponível, ou a *task* esteja dentro do tempo de *delay* após chamada da função *vTaskDelay* [1]. Espera para sincronização com estruturas de dados com semáforos, queues e outros também podem ser casos da tarefa em *Blocked state*.

Outro estado é *Suspended state*, similar ao blocked state, a tarefa não pode ser selecionada pelo *scheduler*, entretanto, ocorre como se a task fosse “desativada”. Uma tarefa entra neste estado pela chamada da função *vTaskSuspended* e sai da suspensão com a chamada da *vTaskResume*.

Quando a tarefa pode ser executada pela CPU, não estando no *Duspended* ou *Blocked state* e esperando sua seleção pelo scheduler, ela se encontra no *Ready state*.

Os estados e a mudança entre eles está apresentada na máquina de estados da Figura 1.

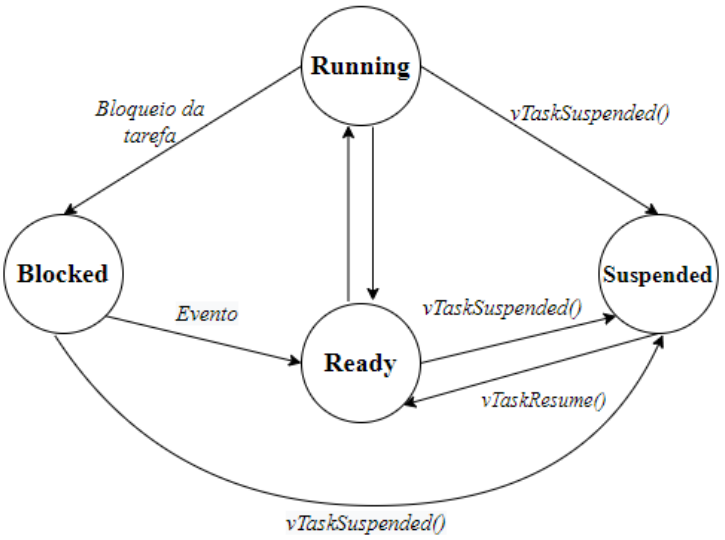

Figura 1. Máquina de estados do RTOS.

Um exemplo pode ser observado na Figura 2, com três tarefas (*task 1, task 2 e Idle*). A *task 1* possui maior prioridade em relação à *Idle* e menor em relação à *task 2*. Inicialmente, a *task 2* está em *Blocked state* e, portanto, não é selecionada pelo *scheduler*. Assim, a *task 1* é a de maior prioridade no *Ready state*, sendo esta selecionada até que *task 2* entra no *Ready state* no instante t1, fazendo com que a *task 1* sofra preempção. Entre os instantes t1 e t2, a *task 1* passa para o *Blocked state*. No instante t2, a *task 2* é suspensa. Dessa maneira, em t2, não há tarefa com maior prioridade que a *Idle task* e esta é selecionada. A Figura 2 ilustra no tempo as *tasks* que estão em *Running state*.

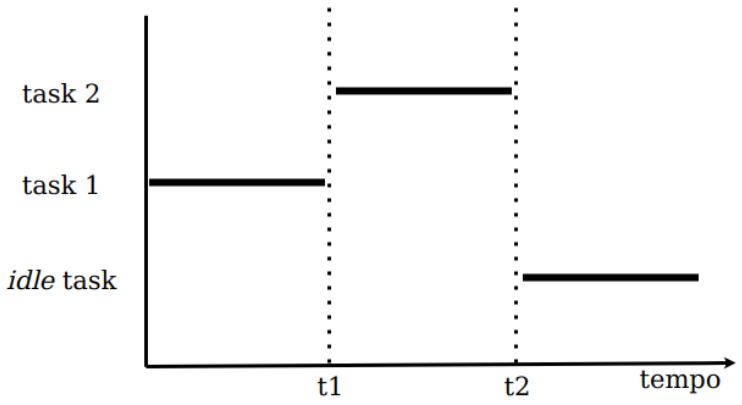

Figura 2. Exemplo de sequência de execução de tasks

3. Implementação das tarefas com RTOS no projeto

Em vista da existência de um ou mais sensores funcionando continuamente durante todo o dia, torna-se necessário a sincronização de leitura do sensor no tempo, todos estando com a mesma referência temporal para o momento de interpretar as informações.

A abordagem utilizada no projeto foi sincronizar a leitura com um tempo inicial definido a partir do protocolo SNTP (*Simple Network Time Protocol*). Este protocolo é uma simplificação do protocolo NTP, que possui a função de sincronização de tempo de um sistema em uma conexão entre diferentes aparelhos. Tendo em vista o uso de diversos dispositivos, como ESP8266's distribuídas pelos laboratórios, por exemplo, é necessário a sincronia dos dados. No caso do projeto, utiliza-se o protocolo para sincronizar os microcontroladores e, desse modo, obter uma leitura síncrona dos sensores, ou seja, ler os valores de temperatura e umidade em um mesmo momento pelos embarcados.

Utilizou-se o sistema operacional de tempo-real (RTOS - *Real Time Operating System*) para controle do tempo de execução das tarefas com a função *vTaskDelay* do freeRTOS e a verificação do tempo seguindo o SNTP. Uma vez armazenada esse tempo inicial, que é periodicamente atualizado por uma *task*, há garantia do *clock* do microcontrolador para incrementar o valor do tempo.

Tem-se uma rotina de uma *task* gerenciada pelo freeRTOS que periodicamente atualiza o segundo do SNTP armazenado. Com o tempo do protocolo é gerada uma estrutura cujo um dos membros é um valor de 0 a 59, que representa o resto da divisão inteira do tempo por 60, referindo-se a um minuto. Localmente no microcontrolador é incrementado o valor do tempo atualizado desde a última execução da *task* que sincroniza o tempo com o protocolo.

No algoritmo é definida a macro STARTSEC que representa o segundo dessa janela de um minuto que se inicia o ciclo de cálculos ou medições que se deseja sincronizar, um valor configurável e menor que o período T.

A sincronização ocorre com base no sntp, ciclicamente, em uma janela de um segundo quando a tarefa executa seu código de interesse (medições ou cálculos). Dessa maneira, existem três instantes importantes para sincronia dentro do intervalo de um minuto: o STARTSEC; a leitura; e um instante anterior à leitura definido para assegurar que a *task* não esteja bloqueada no momento que deveria estar realizando a leitura. Foi definida uma janela de um segundo antes do instante de leitura em que a tarefa pode efetuar o código de leitura e envio de dados.

Quando o segundo dentro do intervalo de um minuto for coincidente com o STARTSEC ou um instante posterior, executa-se o código da task, entrando em *Blocked state* com a função *vTaskDelay* por um tempo igual ao período menos uma estimativa (indicado por x na Figura 3) de tempo para que a *task* esteja no R*eady state* antes no instante de leitura. Após sair do *Blocked state* é verificado se a tarefa está dentro da janela de um segundo e, enquanto não estiver na janela, faz a chamada de *delays* curtos como pequenos passos até entrar na janela de leitura.

Há imprecisão em relação ao tempo T exato que a leitura deveria ser executada e esta depende do *Hardware* do microcontrolador e da definição de STARSEC e os tempos de *delay*.

A Figura 3 representa em um diagrama os procedimentos realizados na janela de um minuto.

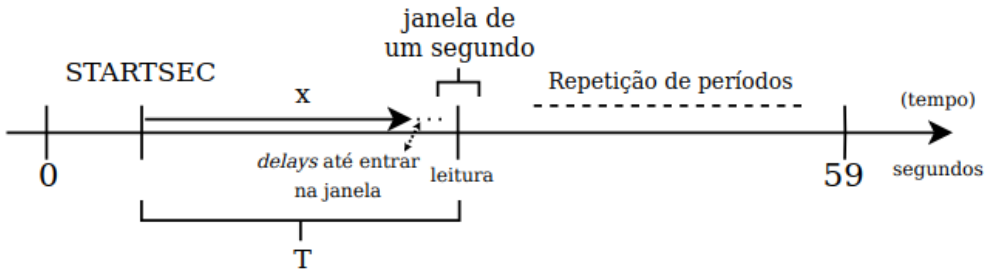

Figura 3. Diagrama da janela de execução das tasks de leitura.

4. Desenvolvimento dos módulos de energia e ambiental

Para leitura dos sensores de umidade, temperatura e concentração de gás, utilizou-se o ESP8266, os sensores DHT11 e a família MQ-x de sensores de gás catalíticos, compondo os três o módulo ambiental.

De acordo com o manual de produto do módulo de medição de temperatura e umidade DHT11, o sensor possui um dispositivo resistivo para obtenção do valor de umidade e um termistor do tipo NTC, resistor de operação em região não linear cuja variação da resistência depende da temperatura de maneira inversamente proporcional, aumenta a resistência ao diminuir a temperatura e vice-versa.

Os dados são passados por uma única porta digital em uma comunicação serial de 8 bits no qual os dados são passados na sequência: 8 bits com o valor inteiro da umidade; em seguida, 8 bits com as casas decimais do valor da umidade; 8 bits do valor inteiro da temperatura; 8 bits para o valor fracionado da temperatura; e por último, 8 bits de paridade para detectar erros nas transmissões.

Dentre os diversos tipos de sensores para detecção de gás, como por exemplo, eletroquímicos, condutividade térmica, semicondutores, os sensores catalíticos [2] são os mais utilizados. O sensor catalítico é construído a partir de um pelistor (*pellistor*), dispositivo no estado sólido utilizado para detecção de gases. Este é formado por uma bobina de platina revestida por uma base de cerâmica de alumina e na camada mais externa um catalisador disperso em um substrato [2].

O sensor é formado por duas bobinas, uma de referência, que é inativa e, por conseguinte, usada com um sinal de referência inerte devido à não resposta ao gás, o que permite usar como base de comparação com a bobina ativa, pois compensa as mudanças do ambientes que afetam a temperatura do sensor e, consequentemente, a resposta gerada.

A outra bobina é a ativa, que é coberta por um catalisador (*catalyst*). Nessa camada ocorre a reação de oxidação do gás, que gera calor e por sua vez altera a resistência da bobina de platina (ativa). Essa resistência é medida em uma ponte de

wheatstone. Assim, essa variação, comparando com a condição normal da bobina de referência, é proporcional à quantidade do gás.

Relativo ao algoritmo do módulo ambiental, são criadas três tasks, sendo duas tasks destinadas à leitura periódica dos sensores, uma para umidade e temperatura e a outra para concentração do gás, e uma task para sincronizar o tempo de referência com o protocolo SNTP a cada uma hora.

Na task de temperatura, estando na subjanela de aquisição dos dados, é chamada uma função para leitura do sensor DHT11. Se ocorrer a leitura corretamente, um JSON é formatado e enviado por MQTT contendo a região da sala em que o módulo se encontra, padronizado; a identificação da sala de monitoramento; e os valores de umidade e temperatura. Caso haja algum erro na leitura, é enviado uma mensagem de LOG no terminal.

A segunda *task* é responsável por ler o valor analógico do sensor de gás e enviar o valor de concentração do gás em partes por milhão (ppm) por MQTT ao broker em um JSON, similar ao realizado com a *task* do DHT11.

O primeiro passo para a leitura do sensor analógico é a configuração do canal de leitura analógico e a inicialização das variáveis que armazenam os valores da resistência interna do módulo MQ135, a resistência de referência e as constantes para o cálculo da concentração em ppm do gás.

Dentro da *task* é chamada a função *adc_read*, que retorna a situação de leitura, se ocorreu corretamente ou não, e recebe um ponteiro que aponta para o inteiro que armazena o valor do adc. Além da leitura, calcula-se a concentração de gás com a lei de potência ax^b, com a e b duas constantes, que podem ser calibradas através de um segundo sensor de gás, e x a razão entre Rs (a resistência do sensor na presença de gás) e R0 (resistência de referência). Rs é obtido a partir do valor do adc lido.

Ambas as *tasks* do módulo ambiental entram em *Blocked state*, aguardando o momento de sincronização para passar para o *Ready state*, ser selecionado ao *Running state* pelo *scheduler* e executar a leitura dos sensores. Nenhuma das tarefas entra em *Suspended state.*

A estrutura do código desenvolvida para o módulo de energia também se baseia nesse princípio de sincronização da *task*. Obtém-se N amostras de tensão e corrente para calcular os valores de potência (ativa, reativa, aparente e fator de potência), tensão e corrente rms, e a distorção harmoniza com o algoritmo FFT (*Fast Fourier Transform* - Transformada Rápida de Fourier).

Além do módulo ambiental, desenvolveu-se utilizando a ESP32 o módulo de energia, cujo objetivo é ler em intervalos constantes de tempo a corrente e tensão de um circuito elétrico para estimar o fator de potência bem como a potência ativa, reativa e aparente.

Com um transdutor de tensão e um transdutor de corrente, realiza-se a leitura destas variáveis do circuito elétrico com uma frequência de amostragem, no caso dos

60Hz de frequência padrão, igual 200µ segundos. Portanto, a cada $200\mu s$ é coletado um valor de tensão e corrente. Assim, em vista de um período de onda $T = \frac{1}{60}s \simeq 0,0167s$, para 250 pontos de amostragem ($50ms$), são obtidos valores de exatos 3 ciclos $((0,05)/(0,01\overline{6}) = 3)$.

Estipula-se uma defasagem entre os sensores de $1200\mu s$, que é compensado numericamente por um atraso de 6 pontos ($1200\mu s/200\mu = 6$). são utilizados valores Desse modo, em um buffer que armazena os 250 pontos, ao fim do ciclo de amostragem, no cálculo das variáveis de energia, mantém-se um vetor fixo para tensão o de corrente é adiantado em 6 pontos ($1200\mu s$). Para o cálculo da FFT de um vetor, é necessário que este vetor tenha dimensão de tamanho 2^n, n inteiro. Na situação obtém-se 250 pontos, zerando-se os últimos 6 valores do vetor.

Realizando testes de 0 à 10 pontos de adiantamento, (números inteiros, pois se trata da posição de um vetor), 6 pontos de atraso apresentou o melhor resultado de medição de corrente rms, tensão rms e potência média (produto da corrente rms e da tensão rms) quando comparado com um sensor externo medindo as mesmas variáveis.

A tensão rms V_{rms} é calculada a partir das medidas discretas de tensão adquiridas pelo sensor, sendo $V_{rms} = \sqrt{\frac{1}{n}(\sum_{i=0}^{n-1} V_i^2)}$, onde n é o número total de pontos de leitura e V_i o valor da i-ésima leitura discreta. O cálculo da corrente rms I_{rms} é análogo ao cálculo da tensão rms e a potência aparente S é obtida pelo produto $S = I_{rms} \cdot V_{rms}$.

A potência ativa P com o conjunto de valores discretos de tensão e corrente (V_i e I_i) é dado por $P = \frac{1}{n}\sum_{i=0}^{n-1}(V_i \cdot I_i)$ e o fator de potência $FP = P/S$.

Para identificação das harmônicas, emprega-se o algoritmo FFT (*Fast Fourier Transform* - Transformada Rápida de Fourier) para calcular as componentes de frequência do sinal lido em pontos ao longo do tempo a partir da Transformada de Fourier.

O algoritmo de Transformada Discreta de Fourier (DFT - *Discrete Fourier Transform*), requer um número N de amostras do sinal o qual se deseja representar por uma soma de senóides para representar o sinal no domínio de frequência com as componentes de frequência e suas respectivas amplitudes.

A equação utilizada no DFT é $X(k) = \sum_{n=0}^{N-1} x_n \cdot e^{\frac{-j2\pi nk}{N}}$ (I) onde k representa a k-ésima frequência, $X(k)$ o coeficiente complexo da frequência, N o número total de amostras e x_n a n-ésima amostra de sinal temporal.

Pela fórmula de Euler, $e^{i\theta} = cos\theta + isen\theta$. Assim o termo e^{-jp}, $p = \frac{2\pi nk}{N}$, pode ser expandido em $e^{-jp} = cos(-p) + jsen(-p) = cos(p) - jsen(p)$.

Logo, $X(k) = \sum_{n=0}^{N-1} x_n \cdot [cos(\frac{2\pi nk}{N}) - jsen(\frac{2\pi nk}{N})] = \sum_{n=0}^{N-1} x_n cos(\frac{2\pi nk}{N}) - j\sum_{n=0}^{N-1} x_n sen(\frac{2\pi nk}{N})$. Para o termo real, $A_k = \sum_{n=0}^{N-1} x_n cos(\frac{2\pi nk}{N})$ e o termo imaginário, $B_k = -\sum_{n=0}^{N-1} x_n sen(\frac{2\pi nk}{N})$ $\Rightarrow X(k) = A_k + jB_k$.

O algoritmo FFT, resolve o mesmo problema do DFT. No entanto, relativo ao tempo de processamento para encontrar $X(k)$, o FFT é computacionalmente mais vantajoso, pois reduz o número de operações necessárias em relação ao DFT, haja visto que durante o cálculo de $X(k)$ no DFT, alguns termos aparecem repetidos entre diferentes ks, enquanto o algoritmo FFT não recalcula termos.

No estudo dos sinais lidos pelos transdutores de tensão e corrente, destaca-se a análise de distorções harmônicas. Em um sistema elétrico, harmônicas são frequências múltiplas de uma frequência fundamental, 60 Hz, sendo que transdutores de tensão e corrente não devem introduzir harmônicas ao sinal original.

A distorção harmônica total (DHT) é uma medida do efeito de distorção de todas harmônicas presentes no sinal, calcula-se $DHT = \frac{1}{V_1}\sqrt{\sum_{i=2}^{H} V_i^2}$, onde H é o número total de componentes de frequência do sinal e V_i a tensão RMS da i-ésima componente de frequência, que pode ser obtida pela gráfico de amplitude espectral resultante do algoritmo DFT ou FFT

5. Resultados

A aplicação do *Kernel* freeRTOS cumpriu com as necessidades do projeto na execução multitarefa, controlando o estado das *tasks* em tempo real e considerando a periodicidade e sincronização da leitura dos sensores. Dessa maneira, com o desenvolvimento dos módulos de sensoriamento, pode-se estabelecer, por exemplo, os pares de temperatura e umidade em função do tempo, que são parâmetros para obter o índice de calor.

Observou-se que com uma frequência amostral que permite recolher pontos de ciclos completos é possível calcular os valores de potência e dos harmônicos do sinal para compreender o estado da rede elétrica. No caso do módulo de energia foi utilizado

o algoritmo FFT, que calculou com sucesso as harmônicas do sinal de entrada e amplitude de cada frequência.

A visualização das variáveis físicas ao longo do tempo, como é a estrutura que se organizou ao utilizar o sistema operacional de tempo-real, enviando periodicamente medições, pode ser observada graficamente de diversas formas, como mostra a Figura 4, onde os comportamentos e tendências da temperatura e umidade, bem como a relação entre elas, é analisada

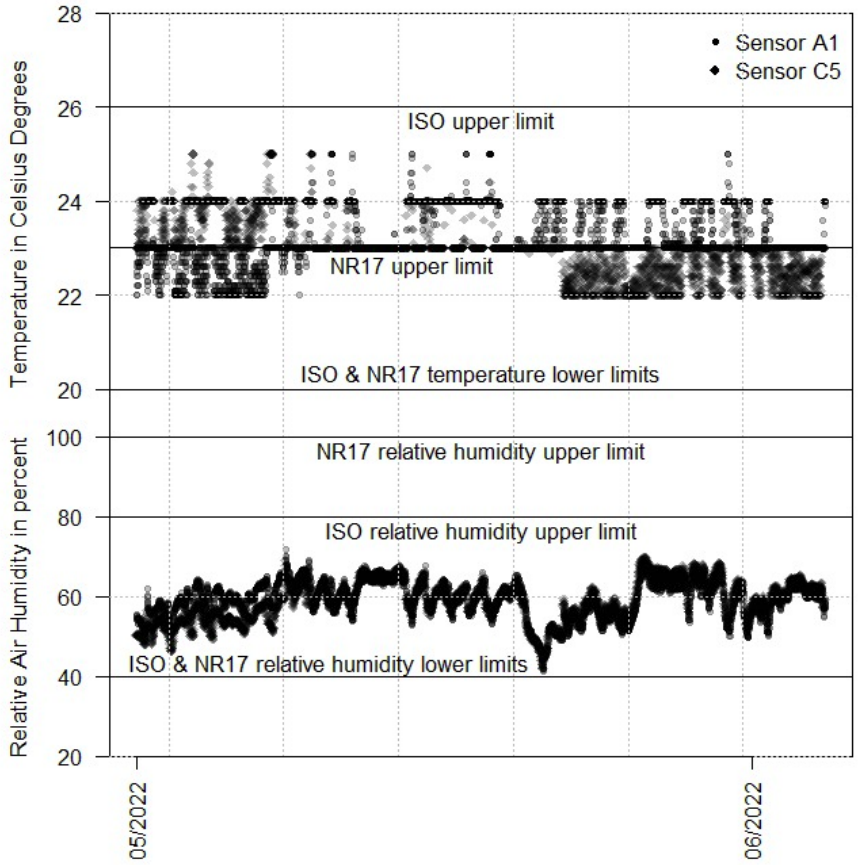

Figura 4. Leituras de umidade e temperatura de 05/2022 à 06/2022.

A definição periódica das *tasks* com aplicação das janelas utilizando o freeRTOS permitiu executar as tarefas com sincronização. Portanto, essa precisão de tempo e a qualidade dos dados lidos garantem que, após o envio dos valores, os dados estejam organizados em uma série temporal no banco de dados. Essa ordenação é útil para análise das informações, tanto na questão de avaliar o conforto térmico quanto no consumo energético, considerando que monitorar o gasto energético possui importância ambiental e financeira.

Como resultado da aplicação do RTOS, se identifica a possibilidade de observar graficamente os dados em uma plataforma desenvolvida em *Node-Red*, responsável também pela conexão com o banco de dados e a aplicação dos algoritmos estatísticos e

de outros métodos que podem ser implementados. Esta plataforma, além de apresentar os valores ao usuário, também tem um campo para avaliar o conforto em relação a temperatura e a umidade.

6. Conclusão

Analisar um ambiente fechado para definir o nível de conforto de um usuário em tempo real é viável, portanto, usando por base a leitura de sensores de baixo custo gerenciados por um microcontrolador com RTOS, como validado ao longo da pesquisa. Dessa maneira, garante-se a continuidade do estudo para o passo de condicionar por meio de atuadores o conforto esperado pelo usuário, em vista que, com os dados coletados mais métodos de análise de conforto ambiental, pode-se estimar a mudança necessária nas variáveis físicas do ambiente para se aproximar o máximo possível da condição ideal de conforto.

O índice de calor reflete a temperatura aparente devido a umidade relativa e a temperatura, um modo simples de começar a entender o conforto do usuário, em vista que uma alta umidade pode aumentar a temperatura aparente e causar incômodos.

Segundo a NR 17 [3], a temperatura no ambiente de trabalho precisa estar entre 20 e 23 °C sem que a umidade relativa fique abaixo de 40%. Com base nas regulamentações, espera-se que os dados pertençam pelo menos à região de conforto delimitada por essas faixas de temperatura e umidade. Levando em consideração a NR 17, ao calcular o índice de calor com os valores das leituras realizadas pela ESP8266 periodicamente, identifica-se, portanto, se a temperatura aparente se encontra ou não dentro desta região.

O sistema desenvolvido, cuja interface está apresentado na figura 5, por consequência, faz com que seja viável a utilização de diferentes métodos de estudo do conforto ambiental em consequência dos dados enviados pelos módulos criados. O índice PMV (*Predicted Mean Vote*), por exemplo, é um estimativa para a sensação de conforto térmico para um grupo grande de pessoas que considera, além de fatores físicos da sala fechada, a relação entre uma pessoa e o ambiente, como a troca de calor do corpo considerando a resistência térmica da vestimenta. Este índice, bem como outras estimativas, será objeto de estudo na continuação do projeto de iniciação científica.

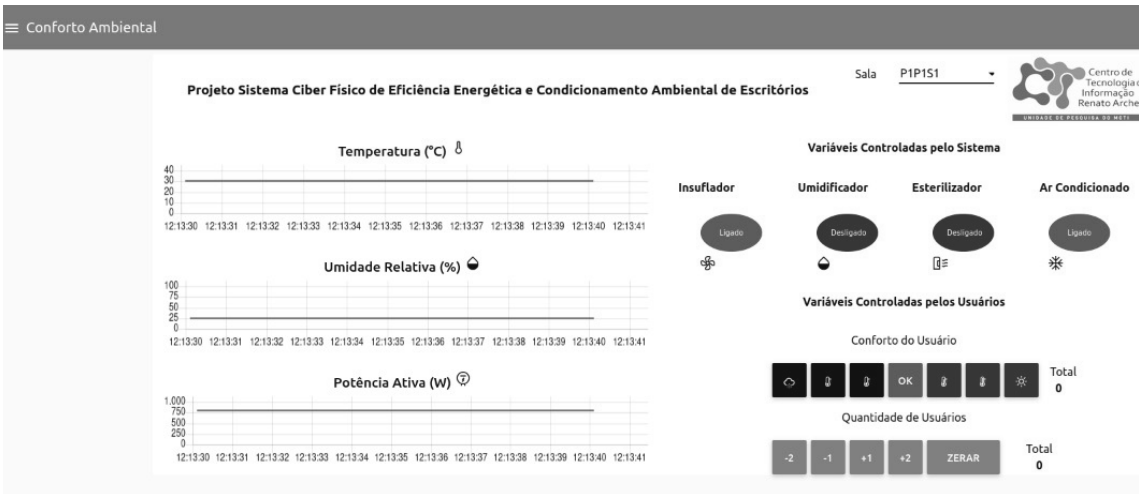

Figura 5. Dashboard de conforto ambiental criado com Node-Red.

5. Agradecimentos

Os resultados deste trabalho refletem o suporte do Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) e do Centro de Tecnologia da Informação Renato Archer (CTI), bem como a constante orientação no desenvolvimento projeto do Antonio Pestana Neto e o auxílio técnico do também bolsista Ranulfo Acir de Oliveira Resende.

7. Referências

[1] Barry, R. Mastering the FreeRTOS™ Real Time Kernel: A Hands-On Tutorial Guide. Real Time Engineers Ltd. Ed. 161204. 2016. Disponível em: <https://freertos.org/Documentation/161204_Mastering_the_FreeRTOS_Real_Time_Kernel-A_Hands-On_Tutorial_Guide.pdf>. Acesso em: 31 ago. 2022.

[2] PERCON. Como funciona o detector de gás?. [S. l.], [201-]. Disponível em: https://acessopercon.com.br/percon/funcionamento-de-detectores-de-gases/. Acesso em: 31 ago. 2022.

[3] ——, “Portaria nº 3.214, de 8 de junho de 1978. Aprova as Normas Regulamentadoras - NR - do Capítulo V, Título II, da Consolidação das Leis do Trabalho, relativas a Segurança e Medicina do Trabalho.”. Disponível em: <https://www.gov.br/trabalho-e-previdencia/pt-br/composicao/orgaos-especificos/secretaria-de-trabalho/inspecao/seguranca-e-saude-no-trabalho/ctpp-nrs/norma-regulamentadora-no-17-nr-17>. Acesso em: 31 ago. 2022