Controlador de ar-condicionado

Marina Luz[1]

[1]UFSC

March 25, 2022

1 Introdução

Neste projeto foi utilizado o Espressif ESP32-DevKitC (como mostrado no site oficial da tensorFlowLite), um potenciômetro (o mesmo foi utilizado por apresentar mesmo função operacional que os dois botões, que o autor não possuía para implementar o projeto), um botão para ligar ou desligar o sistema do ar-condicionado , módulo WiFi embarcado na placa ESP32 utilizada, a porta serial-UART para enviar os logs para o programa de admin, o código do software do pc foi desenvolvido no visualStudioCode em c++, o compilador escolhido foi o g++ e o padrão c++17 está sendo usado no projeto do computador, o sistema operacional é MacOS distribuição Monterey.

2 Sistemas embarcados

2.1 Diagrama de classes

A entrega final deste projeto, trás a maior parte dos requisitos pedidos pelo professor. Foi de muito aprendizado ao longo do desenvolvimento devido às requisições do projeto que traziam muitos novos desafios.

De longe o ponto de maior dificuldade foi a junção dos diferentes blocos (as classes foram desenvolvidas em arquivos isolados para validação primeiro) na rotina principal, que exigia o funcionamento de servidores de WiFi, comunicação com porta UART, interrupções de timer, leitura de pinos analógicos(potenciômetro) e digitais (botão) e rotina de machine learning. Como demonstra o diagrama na Figura 1.

As sessões abaixo estão divididas nas classes principais

2.1.1 Main routine:

- A rotina principal inicializa a GeneralController class
- Também inicializa a classe de WiFi e módulo NTP pelo qual é possível pegar informações de horário em tempo real, como timestemp e o horário escolhido é GMT-4 (por conta de um servidor NTP apresentar dados de horário mais realistas e portanto fáceis de conferir no programa de Admin foi utilizado o servidor NTP ao invés da classe time/clock apresentada pelo professor), foi escolhido para versão final devido ao fato da rede neural treinada para execução de análise temporal neste trabalho ter apenas datasets em regiões diferentes, o exemplo encontrado utiliza Miami e para poder validar os resultados foi mantido o mesmo padrão.

2.1.2 GeneralConttroler Class:

- A classe de LogHandler está implementando uma lista com um buffer circular que salva os dados de logs e envia os via serial quando o usuário desliga o ar condicionado ou quando a memória do aparelho está próxima de estar completa, os logs são enviados via serial(UART).
- Não foi possível transformar o botão e nem potênciometro em uma função de interrupção, devido a bugs e erros nas tentativas de implementação.
- No entanto, tendo os dois no loop principal da rotina main, já é o suficiente para detectar qualquer variação de temperatura ou estado do ar condicionado inserida pelo usuário em questão de milissegundos.
- A classe LogHandler poderia ser atribuída em forma de herança para o GeneralController, isso seria um bom passo na otimização deste trabalho.
- O protocolo de handshake implementado é bem simples e é explicado na parte do sistema do computador

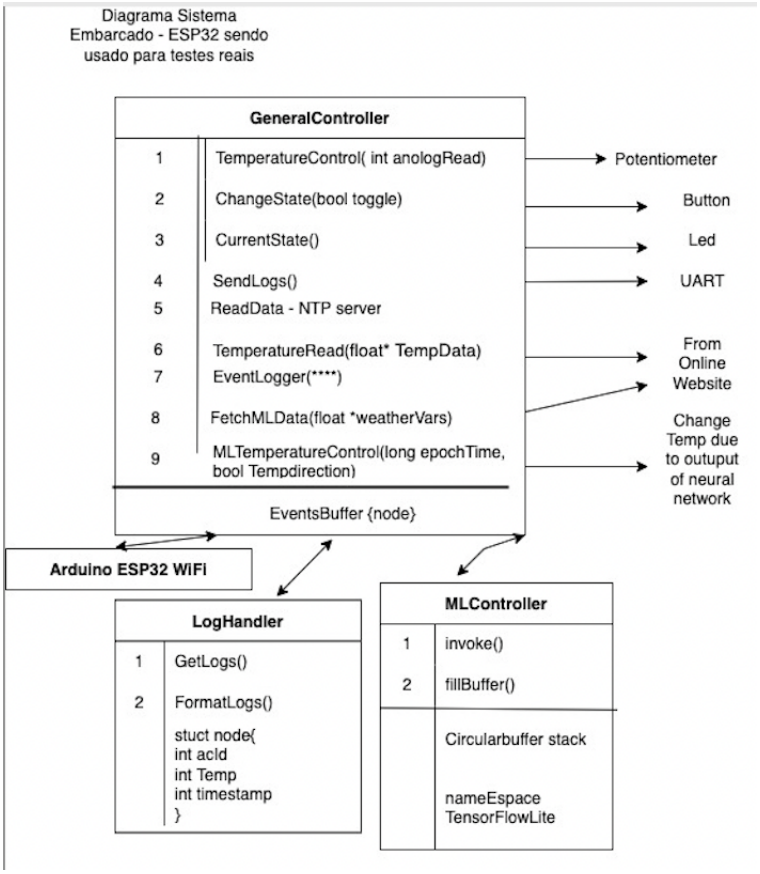

Figure 1:

- Através do método fetchMlWeather é possível obter os dados em tempo real de pressão, humidade e temperatura atual em Miami (input da rede neural).
- Os dados de temperatura são obtidos em JSON e foi utilizado a classe ArduinoJson para fazer o parsing desses dados. No mais é necessário normalizar a informação antes de enviar a rede neural
- O site escolhido foi :

  https://thecustomizewindows.com/2019/06/esp32-arduino-fetching-current-weather-data-no-json-parsing/
- Foi gerado uma API key e assim conseguia obter a temperatura.

2.1.3 Jupyter Lab:

- Foi utilizado um arquivo chamado training.ipynb , que está no github juntamente com este relatório para gerar a rede neural utilizada neste trabalho.
- A implementação foi baseada no código exposto no github de :

  https://github.com/AIWintermuteAI/Seeed_Arduino_Sketchbook
- É então exportado em um arquivo de tflite os pesos e parâmetros dessa rede neural, para ser usada de forma estática no sistema embarcado.
- A rede neural implementada permite obter probabilidade de precipitação e também o tipo do tempo sendo dividido em alguns diferentes tipos como : cloudy, rainy, sunny e etc. . .

2.1.4 MLController Class:

- Esta classe infelizmente não foi completada, foi utilizado um arquivo de tensorflow lite mais simples que foi treinado de exemplo para modelar uma senoide, com o qual o esp32 conseguiu reagir muito bem. link do exemplo:

  https://www.digikey.com.br/en/maker/projects/intro-to-tinyml-part-1-training-a-model-for-arduino-in-tensorflow/8f1fc8c0b83d417ab521c48864d2a8ec

- O problema é que a layer de expansão de dimensão adicionada pela camada convolucional da rede criada com Jupyter Lab não possui implementação para registro pelo TensorFlow Lite versão 2.3 do ESP32.
- Várias tentativas foram feitas pelos próprios issues abertos no github do tensorFlowLite e foi recomendado dar um dowgrade na versão do python usado para treinar a rede (jupyter lab) e tentar novamente com um arquivo tflite, isto foi feito, também foram feitas tentativas de utilizar outras versões do tensorflowlite no sistema embarcado. Infelizmente essas tentativas foram mal sucedidas.
- Também foi tentado utilizar a rede com layers dinâmicas (adicionando as layer manualmente através do comando addbuilin do tensorflow lite) e trocar o register de ExpandDims por Quantize conforme indicação do compilador, mas o problema persistiu.
- No entanto o método de invoke foi implementado de maneira correta, dentro da classe MLController.
- Também o processo de alimentar o buffer circular de stack que alimentaria o parâmetro de input da rede com o os dados provenientes do site utilizado na Classe GeneralController, para aquisição de dados.
- A lógica seria que quando o buffer estivesse cheio seria feito uma análise do tipo de tempo e tomada uma decisão de aumentar ou abaixar a temperatura de acordo (por exemplo se a previsão fosse de um dia ensolarado o ar seria ajustado de forma mais fria, dias chuvosos seria aumentada a temperatura).
- Essa lógica foi implementada parcialmente para demonstração.
- Conforme é exposto no diagrama de classes do sistema embarcado os métodos utiliados são:
    1. fillBuffer → preenche o buffer circular.
    2. Invoke → joga os novos dados de data para a rede neural computar a previsão temporal.
    3. MLTemperatureControl (no generalController) → atualiza a temperatura , conforme é atualizado para um usuário pedindo alteração pelo potenciômetro, e loga no buffer interno do esp32 a mudança com timeStamp, id do aparelho, conforme requisitado pela definição do projeto.

3 Sistema do Computador

O sistema do computador utiliza basicamente uma interface UART com a parte serial do computador na qual o ESP32 esta conectado para conseguir escrever informações pedindo logs da aplicação e também receber estes logs após a solicitação ser efetuada, em uma Thread diferente para não ser perdido informações, dado que a serial do ESP32 é halfDuplex a thread contém toda a parte de escrita e leitura das informações enviadas ao sistema Embarcado.

Existe um protocolo simples para um handShake entre a aplicação administradora e o sistema embarcado, basicamente é enviado pelo sistema embarcado o tamanho de informação em bytes que será enviado via serial e o sistema administrador após receber esses dados envia uma resposta para o sistema embarcado. Após a resposta o sistema administrador espera 3 segundos para ver os dados no buffer serial do computador, mais que o suficiente para o envio de dados visto que a baud rate da porta serial é de 9600 bits/segundo.

Após a implementação de handshake o sistema administrador conseguiu efetuar a leitura de mais de 90 eventos de mudança de temperatura e convertê-los de volta na struct utilizada no sistema embarcado sem problemas. Esse foi um dos maiores desafios na implementação do trabalho na parte do computador.

Ao invés de uma interface lógica, o sistema de admin é todo desenvolvido por command line, sendo possível buscar os dados nos quais a temperatura do ar condicionado foi modificada, com um ID de identificação do aparelho e também com a timestamp do momento que o evento ocorreu.

Também é possível obter o intervalo de tempo no qual o sistema está ligado pelo painel de administrador.

A lógica central é aplicada na função main que inicializa o código.

A Figura 2 temos o Diagrama do Sistema do Computador.

4 Sistema Celular

Será implementado um cliente HTTP para executar queries para o servidor do sistemaEmbarcado, possibilitando desligar o mesmo e alterar a temperatura através de POST requests e GET requests para adquirir os dados de logs.

Como não foi possível implementar a parte mobile , devido ao trabalho necessário na correção e melhorias da parte embarcada e do computador. Não foi criado nenhum tipo de aplicativo mobile.

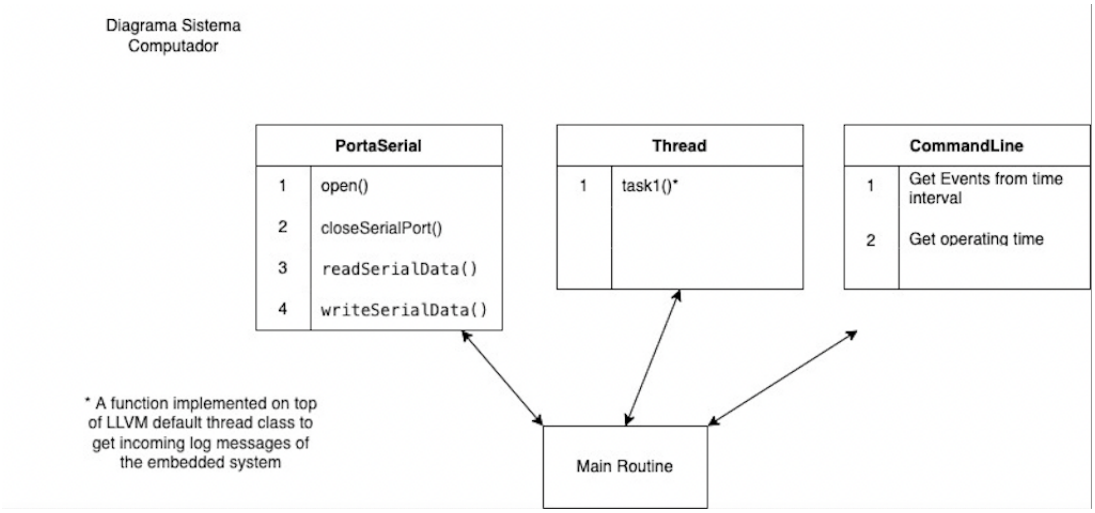

Figure 2:

No entanto conforme descrito acima o aplicativo mobile seria bem simples, se comunicando via WiFi com o sistema embarcado para enviar comandos e receber dados. Uma UI simples seria criada com apenas alguns menus e botões.

5 NTP(Network Time Protocol)

NTP significa Network Time Protocol e é um protocolo de rede para sincronização de relógio entre sistemas de computador. Em outras palavras, ele é usado para sincronizar os horários do relógio do computador em uma rede.

6 LINKS

Link GitHub :
https://github.com/Marinaluz51/EEL7323-Embedded-Systems-Programming-in-C-/tree/main/projeto_final_4
Link Youtube :
https://www.youtube.com/watch?v=0y9gbCQT2iE

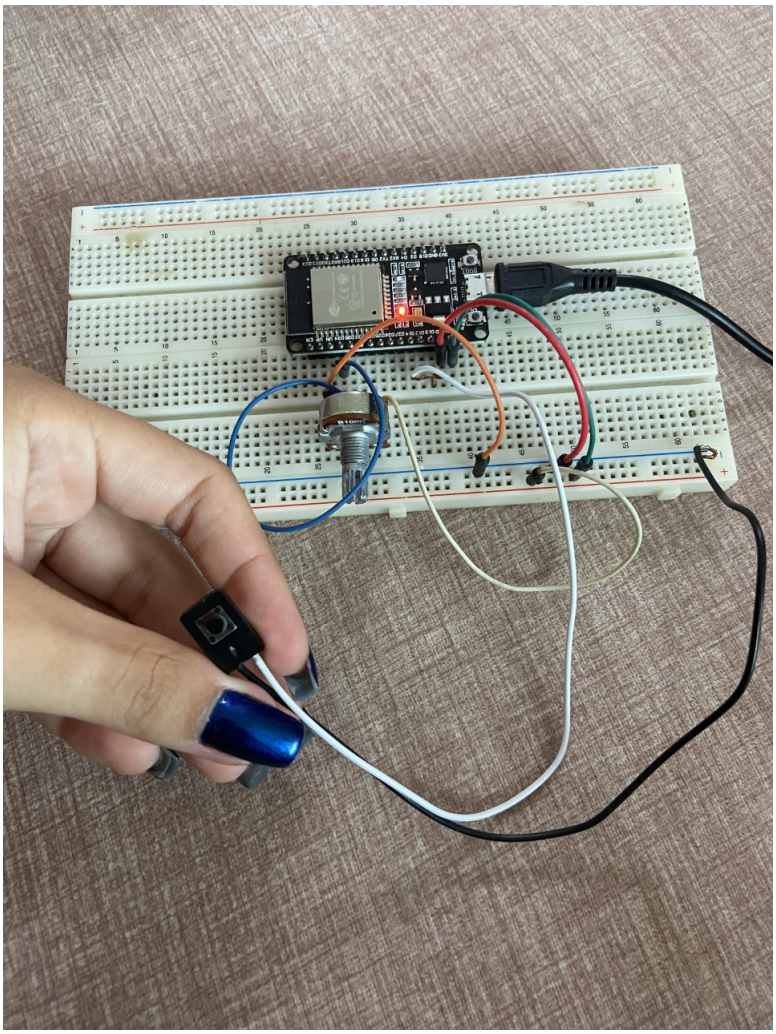

Figure 3: Foto do sistema de testes no estado atual deste relatório

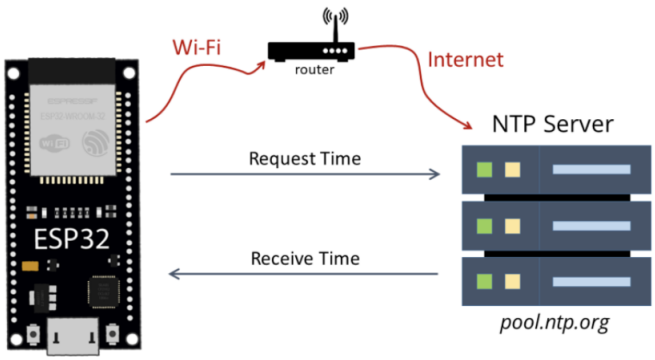

Figure 4: