Luis Antonio Spader Simon (18250065)

Projeto - Controlador de ar-condicionado

Relatório parcial 2 - 10/03 - Software do sistema embarcado, e documentação atualizada (30%) - EEEL7323-08235 (20212) - Programação C++ para Sistemas Embarcados.

Professor: Eduardo Bezerra

Florianópolis SC

2021

1 Sumário

List Of Images

2 Objetivo

- Implementação de um sistema para diminuir o consumo energético em condicionadores de ar.
- Implementação do sistema em Programação Orientada a Objetos em Sistemas Embarcados.

3 Descrição do sistema

3.1 Resumo do sistema OK

O sistema proposto executará a tarefa de monitorar a temperatura externa e ajustar a temperatura do dispositivo de ar-condicionado (AR)de forma a melhor conforto térmico e menor consumo de energia.

O sistema conta com um microcontrolador que fará toda a parte de controle da temperatura, monitoramento, registro do log de operações em uma lista e envio desta lista para um computador através de comunicação serial UART.

3.2 Descrição do Sistema Físico (Hardware)

A figura abaixo representa o funcionamento do Controlador Embarcado de Baixo Custo com sistema de controle através da utilização de algoritmos de IA (CEBC-IA).

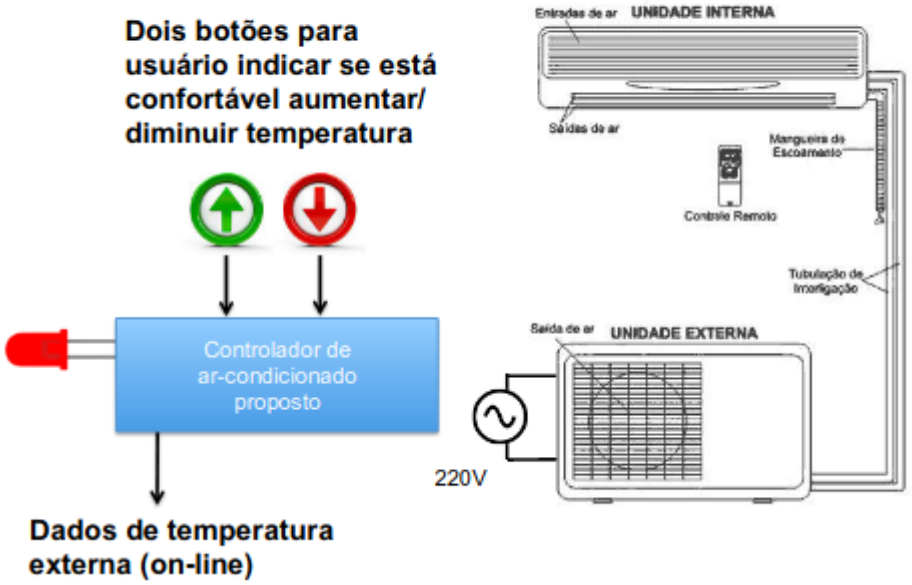

Figure 1. Sistema Físico

Através de 2 botões para aumentar ou diminuir a temperatura, o usuário informa ao sistema a necessidade de mudança de temperatura.

O algoritmo de IA, através de 2 informações (temperatura externa e o desejo de mudança da temperatura pelo usuário), controla a temperatura do dispositivo AR. O envio de comandos para o dispositivo AR deve ser implementado em uma das duas formas a seguir:

- LED Infravermelho: comando enviado diretamente ao dispositivo AR.

- Bluetooth: comando enviado para app de smartphone, para então alterar a temperatura do AR

3.3 Descrição do Sistema Digital (Software)

3.3.1 Funcionamento Software vs Hardware

A figura abaixo representa em diagrama de blocos a interação entre o sistema embarcado e todos os componentes os quais pode se comunicar. Cada componente e sua função serão descritos a seguir:

1. **Microcontrolador (Raspberry Pi Pico):** utilizado para fazer a interface Software vs Hardware, controlando e registrando mudanças de temperatura quando necessário.
2. **Botões:**
    a. **Botão Aumentar Temperatura:** ao ser pressionado aumenta a temperatura do dispositivo e reajusta o algoritmo de controle com IA (se adequa às preferências do usuário).
    b. **Botão Diminuir Temperatura:** ao ser pressionado diminui a temperatura do dispositivo e reajusta o algoritmo de controle com IA (se adequa às preferências do usuário).
3. **Sensor de Temperatura Externa (DHT11):** O DHT11 é um sensor de temperatura e umidade que permite fazer leituras de temperaturas entre 0 a 50 Celsius e umidade entre 20 a 90%.
4. **Envio de comandos bluetooth:** o comando é enviado do microcontrolador para o app de Smartphone para então alterar a temperatura do AR.
5. **Transmissão de dados de Log**
    a. **Notebook:** Interface para recebimento da lista de eventos salva no microcontrolador e visualização da mesma.
    b. **Smartphone:** Interface para recebimento da lista de eventos salva no microcontrolador e visualização da mesma.

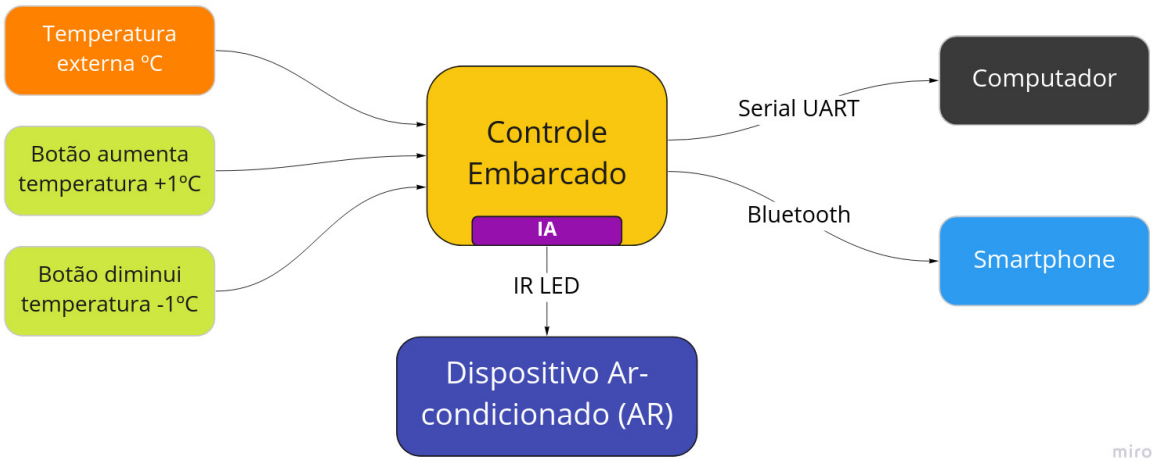

Figure 2. Diagrama de blocos do sistema

4 Descrição dos programas

4.1 Sistema Embarcado

4.1.1 Funcionamento

O sistema embarcado possui apenas 2 tipos de entradas:

- Temperatura externa: leitura obtida em tempo real através do sensor DHT11 com interface feita através da classe 'Sensor_temp_DHT11' em C++;
- Botões de temperatura: interface através da classe 'Botoes';

O objetivo do sistema embarcado é continuamente monitorar as entradas e decidir entre a mudança ou não da temperatura do dispositivo de ar-condicionado (AR) ou do envio da lista de eventos (log) a um computador.

Em loop o sistema de controle embarcado monitora estas entradas, alterando a temperatura quando necessário ou altera seu estado (mudança do estado 'Controle Ativo' para o estado 'Log'). Esta mudança de estado só ocorre quando os 2 botões físicos são pressionados. Uma melhor visualização da lógica do sistema embarcado pode ser vista na Figura 3.

4.1.2 Fluxograma de lógica funcional

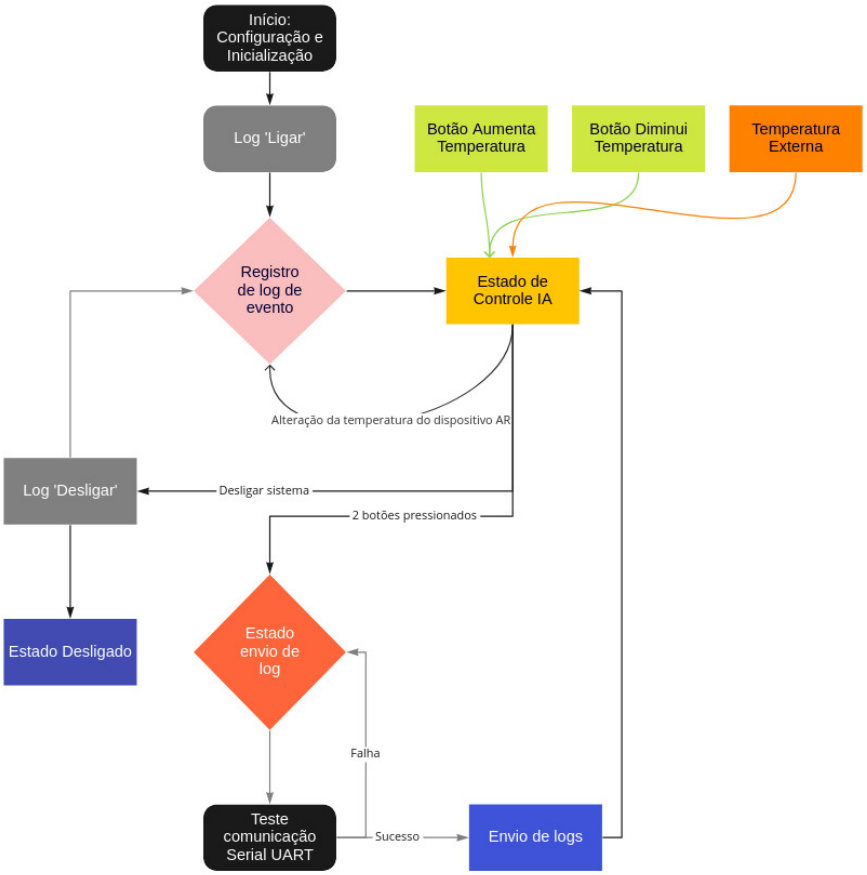

Figure 3. Lógica funcional sistema embarcado

4.1.3 Diagramas

4.1.3.1 Diagrama de Classes

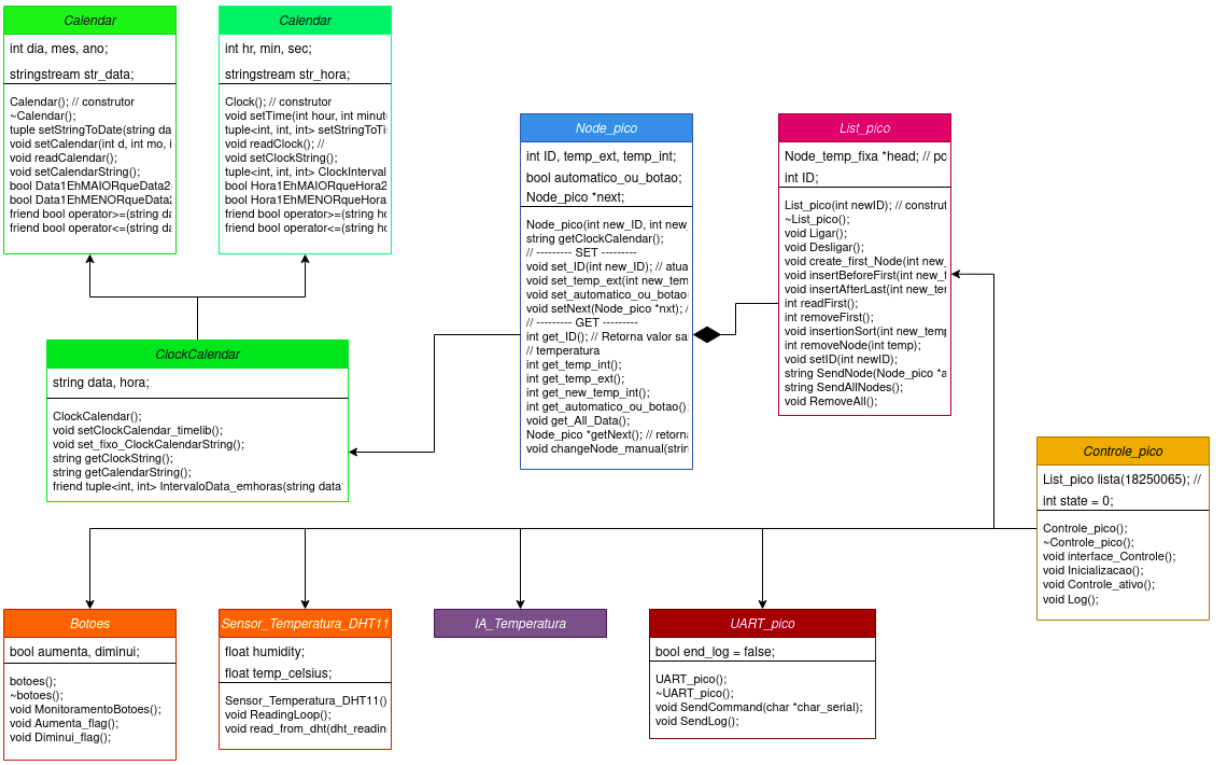

Figure 4. Diagrama de classes software Embarcado

4.1.3.2 Diagrama de transição de estados (máquina de estados)

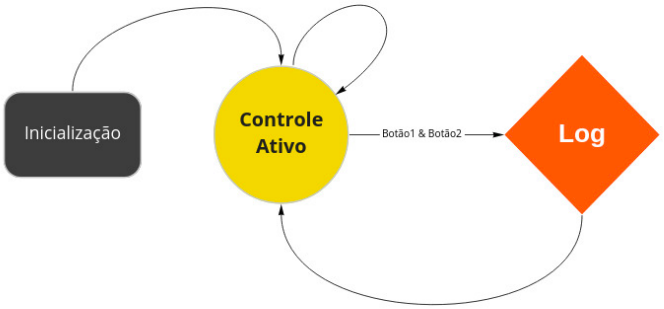

Figure 5. Máquina de estados do software embarcado

4.1.4 Descrição dos Componentes

4.1.4.1 Componentes de Hardware

4.1.4.1.1 Microcontrolador Raspberry Pi Pico

Raspberry Pi Pico é uma placa microcontroladora de baixo custo e alto desempenho com interfaces digitais flexíveis. Na figura abaixo pode-se ver o desenho da placa e a indicação das possibilidades de uso para cada pino.

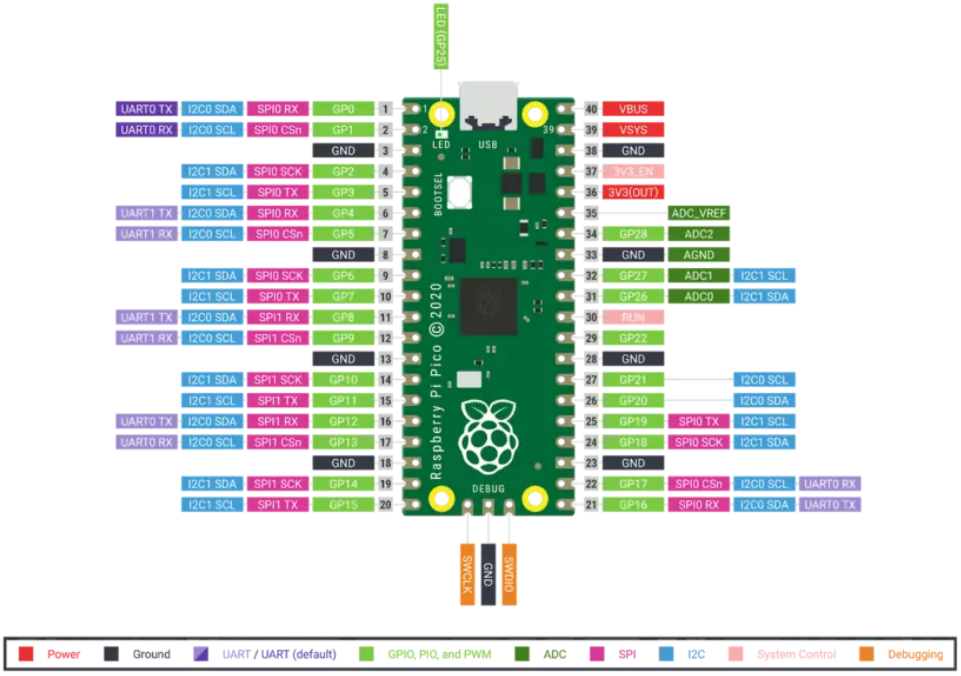

Figure 6. Microcontrolador Raspberry Pi Pico

Os principais recursos incluem:

- Chip microcontrolador RP2040 projetado pela Raspberry Pi
- Processador Dual-core Arm Cortex M0+, clock flexível rodando até 133 MHz
- 264 KB de SRAM e 2 MB de memória Flash integrada
- USB 1.1 com suporte a dispositivo e host
- Modos de suspensão e inatividade de baixo consumo
- Programação de arrastar e soltar usando armazenamento em massa via USB
- 26 × pinos GPIO multifuncionais
- 2 × SPI, 2 × I2C, 2 × UART, 3 × ADC de 12 bits, 16 × canais PWM controláveis
- Relógio e temporizador precisos no chip
- Sensor de temperatura
- Bibliotecas de ponto flutuante aceleradas no chip
- 8 × máquinas de estado de E/S programáveis (PIO) para suporte periférico personalizado
-

4.1.4.1.2 Sensor de Temperatura e umidade DHT11

O Sensor de Umidade e Temperatura DHT11 é um sensor de temperatura e umidade que permite fazer leituras de temperaturas entre 0 a 50 Celsius e umidade entre 20 a 90%

O elemento sensor de temperatura é um termistor do tipo NTC e o sensor de Umidade é do tipo HR202, o circuito interno faz a leitura dos sensores e se comunica a um microcontrolador através de um sinal serial de uma via.

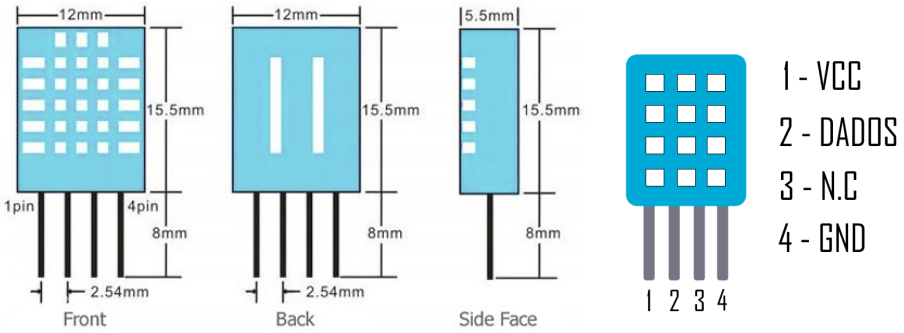

Figure 8. Componente: Sensor de umidade e temperatura DHT11

Abaixo pode-se ver as conexões necessárias entre o componente e o microcontrolador (Raspberry Pi Pico).

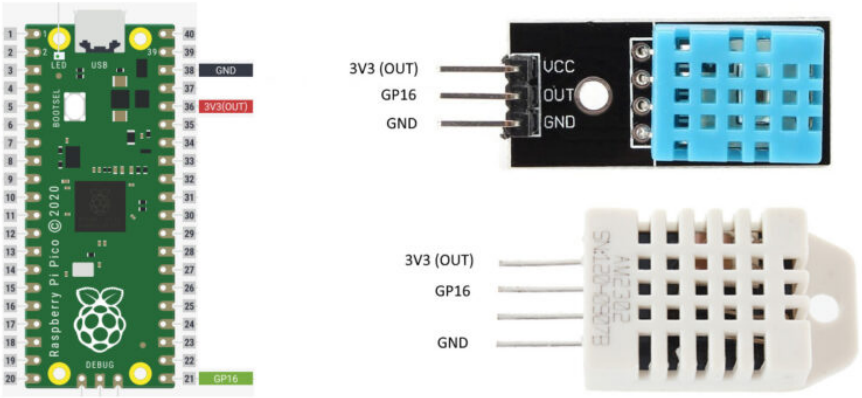

Figure 9. Indicação de ligação entre sensor DHT11 e Microcontrolador

Especificações:

- Modelo: DHT11 (Datasheet)
- Faixa de medição de umidade: 20 a 90% UR
- Faixa de medição de temperatura: 0º a 50ºC
- Alimentação: 3-5VDC (5,5VDC máximo)
- Corrente: 200uA a 500mA, em stand by de 100uA a 150 uA
- Precisão de umidade de medição: ± 5,0% UR
- Precisão de medição de temperatura: ± 2.0 ºC
- Tempo de resposta: 2s

– Dimensões: 23 x 12 x 5mm (incluindo terminais)

4.1.4.1.3 Conversor USB para TTL CH340G

Placa baseada no chip CH340G que funciona como um conversor USB para serial TTL, permitindo a interface de dispositivos TTL para USB. Possui 6 pinos e pode fornecer tensão de 3.3V e 5V.

Especificações:

Chip: CH340
LEDs indicadores de comunicação e tensão
Tensão selecionável 5V e 3.3V
Pinagem: 5V, VCC, 3V3, TXD, RXD, GND
Interface: USB tipo A
Tamanho: 52mm x 16mm x 8mm
Peso: 6g

É necessário colocar um resistor de 10 kΩ entre o VCC e o pino de dados (conforme Figura 10), para atuar como um pull up de força média na linha de dados.

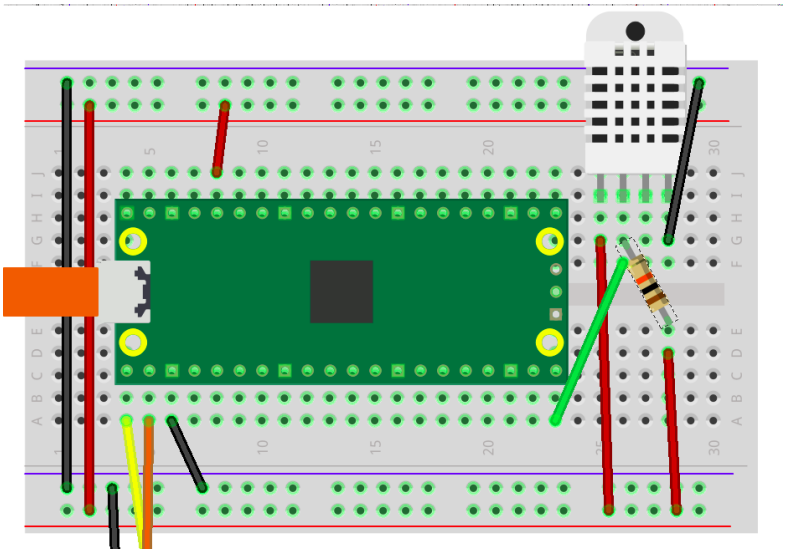

Figure 10. Ajuste de hardware para sensor e microcontrolador

4.2 Computador

4.2.1 Funcionamento

O programa de computador é basicamente uma interface para lidar com a comunicação Serial com o microcontrolador para requisitar o log de eventos e opções de visualização da lista de eventos obtida. A lógica funcional do software e sua máquina de estados podem ser vistas nas Figuras 11 e 13 respectivamente.

4.2.2 Fluxograma de lógica funcional

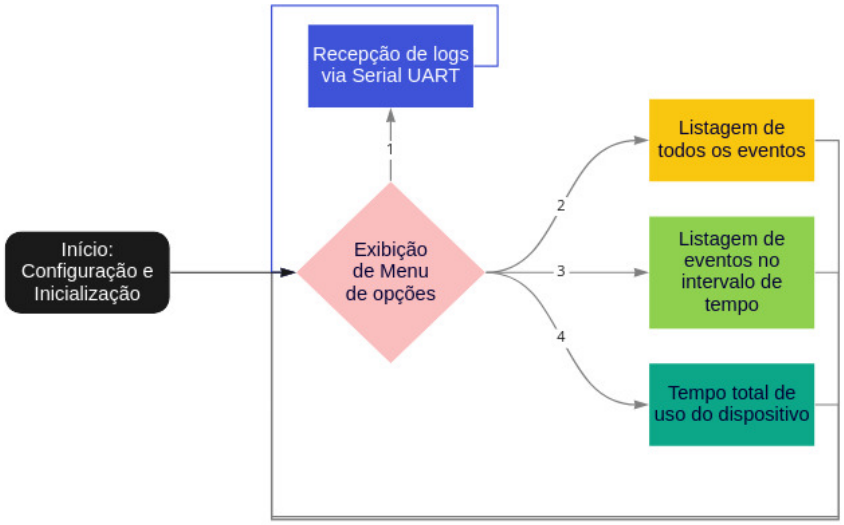

Figure 11. Fluxograma de lógica funcional: Computador

4.2.3 Diagramas

4.2.3.1 Diagrama de Classes

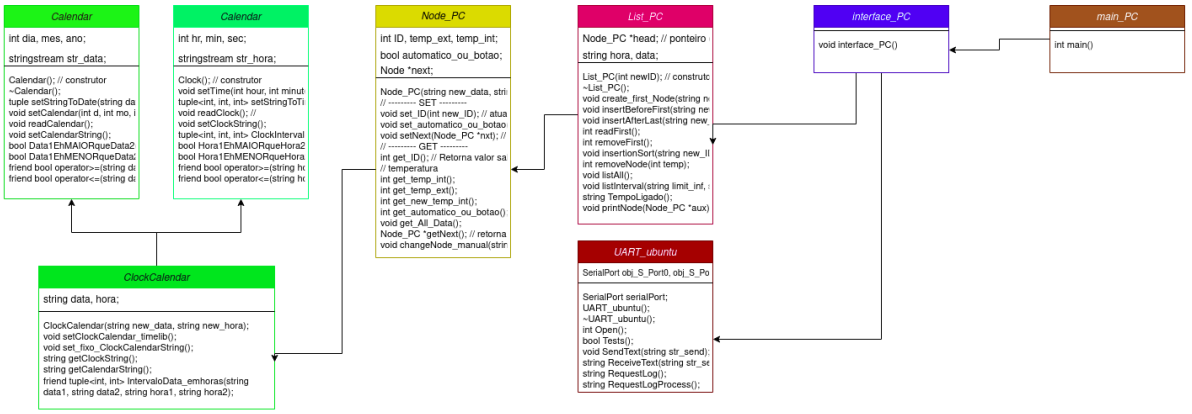

Figure 12. Diagrama de classes software Computador

4.2.3.2 Diagrama de transição de estados (máquina de estados)

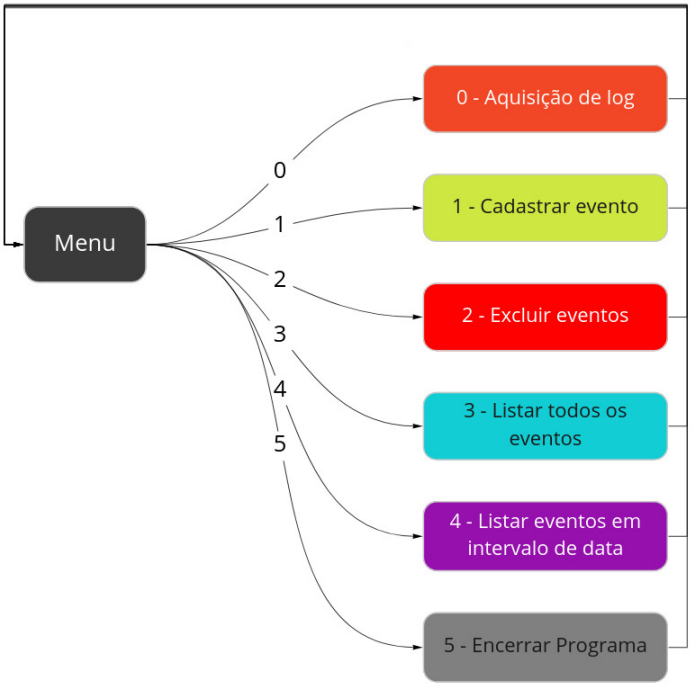

Figure 13. Máquina de estados software Interface Computador

4.3 Smartphone (versões futuras)

4.3.1 Funcionamento

4.3.2 Diagramas

4.3.2.1 Diagrama de Classes

4.3.2.2 Diagrama de transição de estados (máquina de estados)

4.3.3 Descrição dos Componentes

4.3.3.1 Componentes de Hardware

4.3.3.2 Componentes de Software

5 Estruturas de dados utilizadas

A estrutura de dados se baseou no uso de lista encadeada, com as classes **Nó, Lista e UART** se diferenciando entre o dispositivo **embarcado** e o **computador**. A diferenciação se deve às necessidades de inicialização das classes de cada setor, manipulação e visualização da lista de eventos. Esta diferenciação pode ser vista nas Figuras 4 e 12 (Diagramas de classes).

A manipulação da lista no computador possui a opção de visualização com intervalo, dependendo da escolha no menu de Interface no Computador.

```
-----
hora init: 08:00:00
hora end: 10:00:00
data init: 05/02/2022
data end: 08/02/2022
------------------------------------------------------------
========== Lista Intervalo: 05/02/2022 - 08/02/2022 | 08:00:00 - 10:00:00; ==========
|   Data     |   Hora    |    ID    |Temp. Externa|Temp. Dispositivo |Acionamento (1 = aut.)|
| 06/02/2022|  20:28:25| 18250065 |     30      |       28         |          0           |
| 06/02/2022|  20:28:25| 18250065 |     30      |       9999       |          1           |
| 06/02/2022|  20:28:25| 18250065 |     30      |       27         |          0           |
| 06/02/2022|  20:28:25| 18250065 |     30      |       26         |          0           |
| 06/02/2022|  20:28:25| 18250065 |     30      |       1111       |          1           |
------------------------------------------------------------
```

Figure 14. Exibição de eventos em intervalo de tempo e data

6 Infraestrutura de comunicação

6.1 Transmissão de lista de eventos (log de eventos)

Conforme já citado no resumo do sistema, a comunicação Serial se dá por via UART. A placa microcontroladora possui 2 portas UART, onde é utilizada para o projeto apenas uma delas.

Através de 2 classes de comunicação (UART_ubuntu & UART_pico) há a inicialização das portas e troca de dados. A Figura 15 representa o funcionamento das classes e sua comunicação.

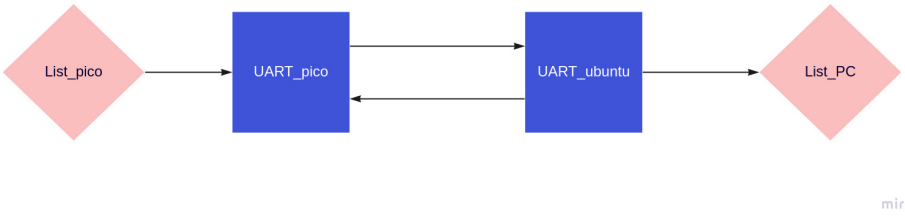

Figure 15. Lógica de transferência de dados entre Microcontrolador e Computador

7 Repositório Github

https://github.com/LuisSpader/EEL7323-Programacao-C-para-Sistemas-Embarcados.git