EEL7323 - Programação C++ para Sistemas Embarcados

Projeto Ar-Condicionado

Aluno:
Felipe Hugo Costa de Oliveira (17104483)

Professor:
Eduardo Bezerra

Março, 2022

Lista de figuras

1 Diagrama do Sistema Ar-Condicionado 3
2 Bloco funcional do Desktop 5
3 Diagrama FSM do Desktop 6
4 Diagrama de classes Desktop 7
5 Bloco funcional do sistema embarcado 9
6 Diagrama FSM do Sistema Embarcado 10
7 Diagrama de Classes do Sistema Embarcado 11
8 Extensões utilizadas para implementar o software 12
9 Linha de comandos para se obter a Toolchain 13
10 Linhas de comando para se obter o SDK 13
11 PICO SDK PATH 14
12 Portas de comunicação do Raspberry Pi Pico 15
13 Modulo Conversor Serial-USB 16
14 Sensor de temperatura DHT11 17
15 Protoboard 18

Sumário

1 Objetivo **3**

2 Descrição do sistema **4**
2.1 Descrição do software do Desktop 4
2.2 Descrição do Software do Sistema Embarcado 7
2.3 Descrição do aplicativo do smartphone 11

3 Implementação dos sistemas **12**
3.1 Implementação do Desktop 12
3.2 Implementação do Sistema Embarcado 13

4 Equipamentos e ferramentas **15**
4.1 Raspberry Pi Pico 15
4.2 Conversor Serial-USB 16
4.3 Sensor de temperatura e umidade DHT11 17
4.4 Protoboard 18

5 Conclusão **19**
5.1 GitHub 19

1 Objetivo

Este projeto tem por objetivo desenvolver um controlador de ar-condicionado visando a economia de energia, conforto e agregação de valor ao produto, utilizando machine learning a um baixo custo. Na figura 1, podemos visualizar o diagrama do sistema ar-condicionado.

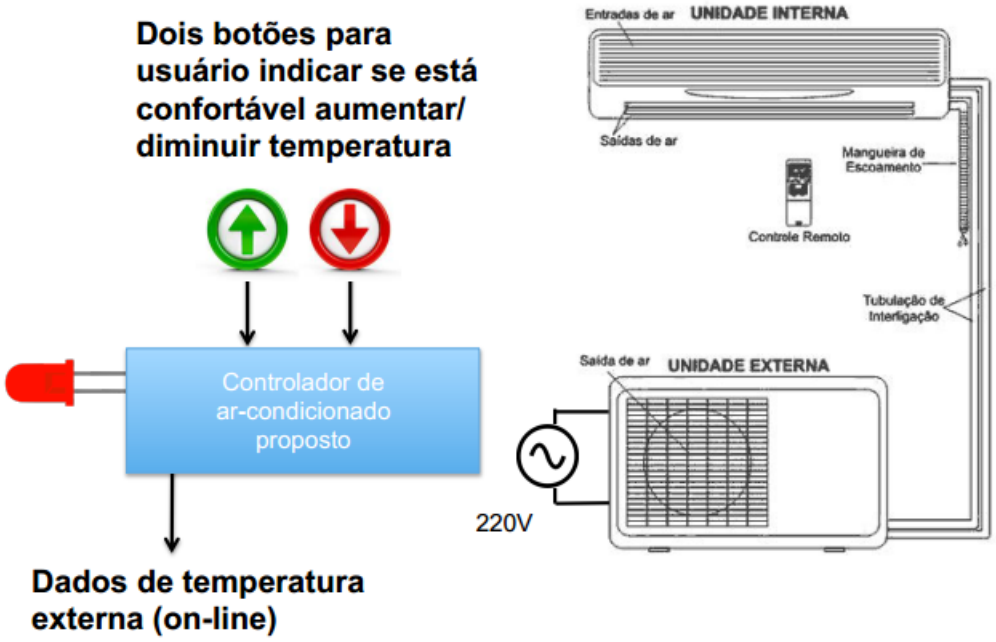

Figura 1: Diagrama do Sistema Ar-Condicionado

2 Descrição do sistema

O sistema conta com um Raspberry pi Pico e ele será responsável por ler 2 parâmetros de entrada, sendo eles: temperatura externa, obtidos via internet(para o caso prático, utilizaremos o sensor DHT11 para obter as temperaturas) e botões para que o ajuste seja feito pelo usuário caso esteja insatisfeito com a configuração atual. Toda vez que a temperatura for alterada, independente do parâmetro, a inteligência artificial, utilizando o TensorFlow Lite, receberá esses dados e irá se adequar às novas exigências e, internamente, será guardadas as informações de identidade do controlador, parâmetro único para cada controlador, e a data e hora que o ajuste na temperatura foi feita. Esses dados salvos poderão ser enviados para o sistema Desktop, onde também serão salvos para que possam ser acessados futuramente.

2.1 Descrição do software do Desktop

O desktop será responsável por receber os dados, via porta serial UART do microcontrolador e guardá-las em uma estrutura de dados tipo lista. O usuário poderá solicitar a listagem de todos os eventos ocorridos em um determinado intervalo de datas, sendo da mais antiga para as mais recentes e também obter o tempo total em um intervalo de datas em que o ar-condicionado ficou em funcionamento.
Quando o sistema é iniciado, primeiramente é configurada a porta serial uart, para que possa receber os dados enviados pelo microcontrolador. A porta é configurada utilizando a função openSerial, que faz parte da classe abstrata Uart e é implementada em UartLinux. Em seguida entraremos no loop while, que continuamente ficará observando se a porta serial está conectada ou se alguma opção no menu foi selecionada. O menu conta com 2 opções para o usuário obter informações, na primeira opção pode-se obter uma lista com todos os eventos registrados em um intervalo de datas, da mais recente escolhida até a ultima. Na segunda opção o usuário poderá selecionar um intervalo de datas e obter o tempo total que o sistema ficou ligado dentro do intervalo.Os dados recebidos pela porta serial são salvos em uma estrutura de dados do tipo lista, pois assim é mais fácil e prático organizar e listar o conteúdo salvo.
As blibiotecas utilizadas para o software do desktop se limitam apenas as padrões da linguagem c++ como iostream e string. A seguir, temos a figura 2, que mostra resumidamente as partes do sistema, figura 3 que mostra a

maquina de estado do software e figura 4 que mostra as classes utilizadas para a implementação do software do Desktop.

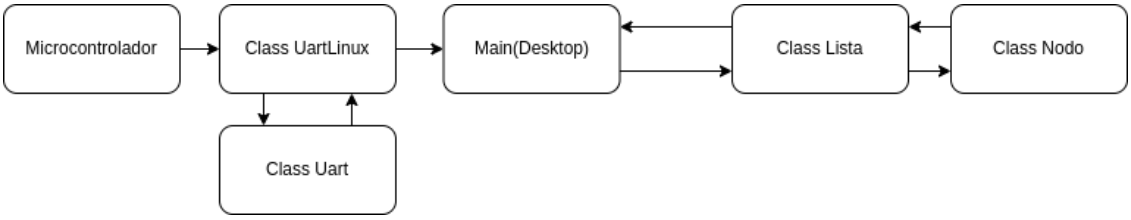

Figura 2: Bloco funcional do Desktop

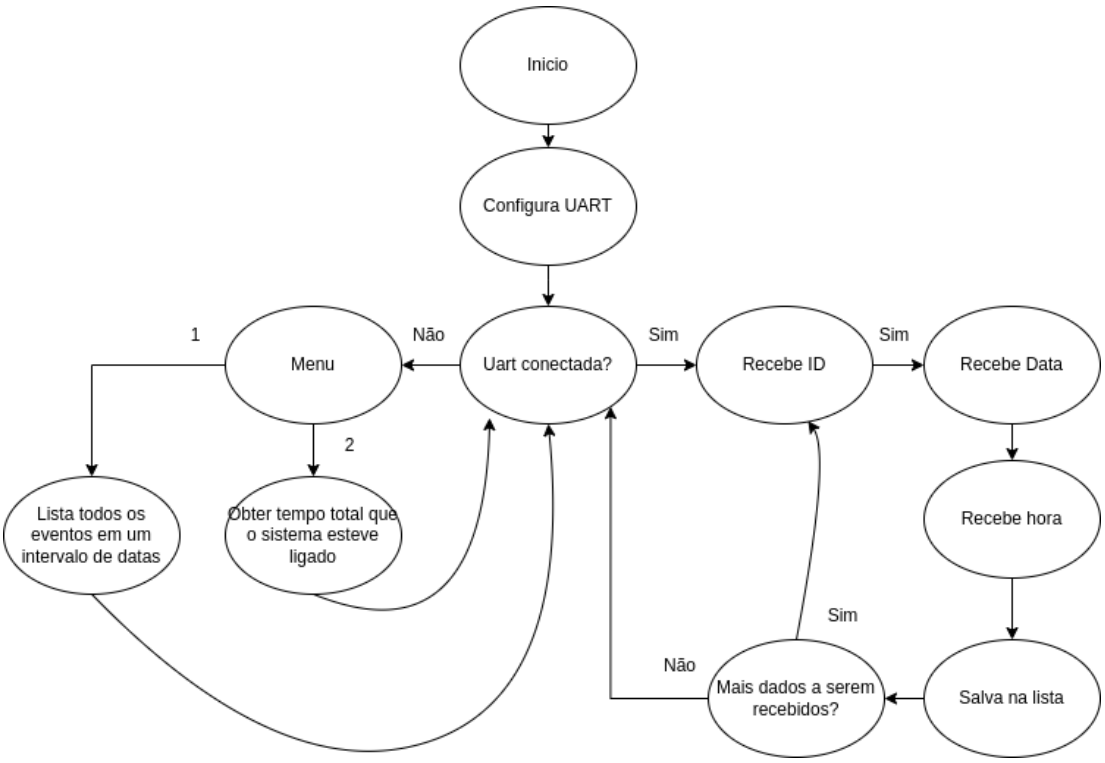

Figura 3: Diagrama FSM do Desktop

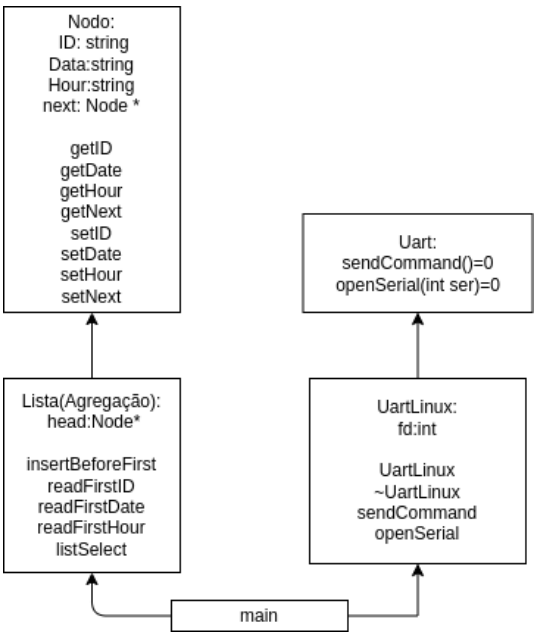

Figura 4: Diagrama de classes Desktop

2.2 Descrição do Software do Sistema Embarcado

O software do sistema embarcado será o cerne do projeto, por ele será recebido os parâmetros de temperatura, envio de comandos ao ar-condicionado via aplicativo smartphone, registro de eventos como o ajuste do ar-condicionado por conta de variações na temperatura externa ou por insatisfação do usuário, será utilizado estrutura de dados tipo fila, envio de dados via UART para serem armazenados no desktop.

Inicialmente é necessário configurar a porta uart serial para que esta já fique pronta para o uso posteriormente. Essa configuração é feita pelas bibliotecas do Raspberry, esta que deve ser configurada seguindo as mesmas configurações feitas no desktop, como bit de paridade, quantidade de dados a serem transmitidos etc para um bom funcionamento. Para esse projeto não utilizaremos bit de paridade e o baud rate foi configurado para 9600 para evitar ao máximo perdas de dados.

Tendo a uart configurada, entraremos no loop while do sistema embarcado que ficará monitorando a temperatura externa, a entrada do botão e a conexão uart. Em caso de haver variação na temperatura ou ativação do botão, o sistema converte a data/hora atual para string e insere esses dados junto com

a ID do controlador em uma estrutura de dados do tipo fila. A ideia por trás de se utilizar apenas strings vem do fato de que cada caracter da string tem tamanho de 1 byte e como a porta serial, tanto do microcontrolador como do desktop utilizam 1 byte para envio e 1 byte a ser recebido, respectivamente, a execução fica simplificada, diminuindo erros.
A temperatura e obtida através do sensor DHT11 por meio da classe TempSense, pelo método getTemperature, este que pertence à classe abstrada temp.h. A utilização de uma classe abastrata se faz necessário por se tratar de um projeto que tem por objetivo funcionar com qualquer marca de ar-condicionado, com cada uma tendo seu metodo para medir a temperatura. Portanto, com algumas modificações, o projeto funcionará com qualquer equipamento do mercado. A fila é uma estrutura em que só pode ser inserido um novo item depois do último e o item a ser removido é sempre o primeiro. A fila é implementada utilizando agregação da classe Nodo. Em seguida o sistema envia esse novo parâmetro de temperatura para o tensorflow que imediatamente se adequa aos novos parâmetros e por fim o comando para a nova temperatura é enviada para o ar-condicionado pelo módulo Wifi ESP-31, já que o Raspberry Pico não tem conexão Wireless embarcada.
O sistema também monitora a porta serial uart e caso esta seja conectada, inicia o processo de esvaziamento da fila, enviando o primeiro item e em seguida o excluindo e repetindo até esvaziar por completo. O envio é feito pela função uart puts, que envia uma string.
Para dar suporte ao software foram utilizadas algumas bibliotecas, tanto nativas do C++ como do Raspberry Pico.

A stdlib.h faz a agregação de um subconjunto principal de bibliotecas Raspberry Pi Pico SDK usadas pela maioria dos executáveis, juntamente com alguns métodos utilitários adicionais. A uart.h tem o conjunto de funções que permitem a comunicação uart serial de forma simples. Para mais informações detalhadas sobre as bibliotecas do Pico acesse: *https://raspberrypi.github.io/pico-sdk-doxygen/modules.html*. Já as bibliotecas nativas do C++, foram utilizadas a iostream, que nos fornece serviços básicos de entrada e saída para programas em C++, string, que apresenta diversas funções para trabalhar e manipular strings.
A seguir, temos a figura 5, que mostra as partes do sistema, figura 6 que mostra a maquina de estado do software embarcado e figura 7 que mostra as classes do software embarcado.

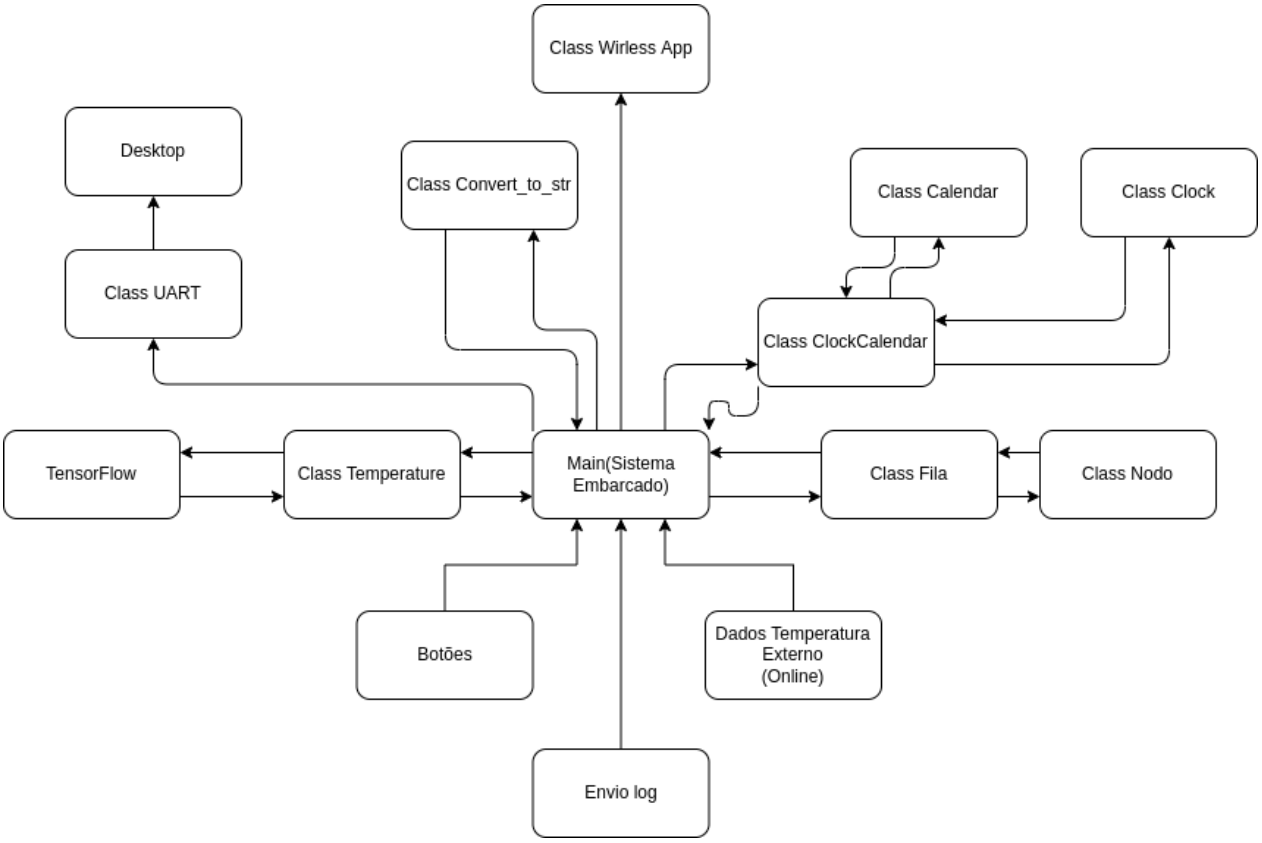

Figura 5: Bloco funcional do sistema embarcado

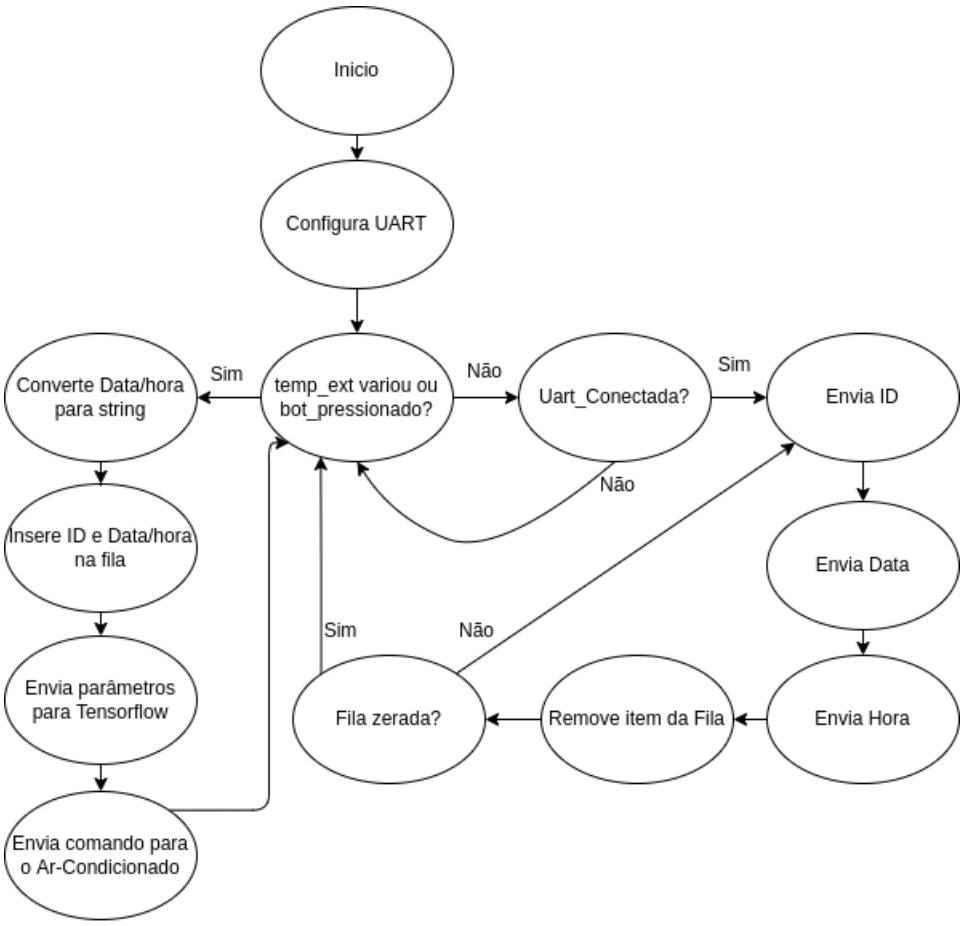

Figura 6: Diagrama FSM do Sistema Embarcado

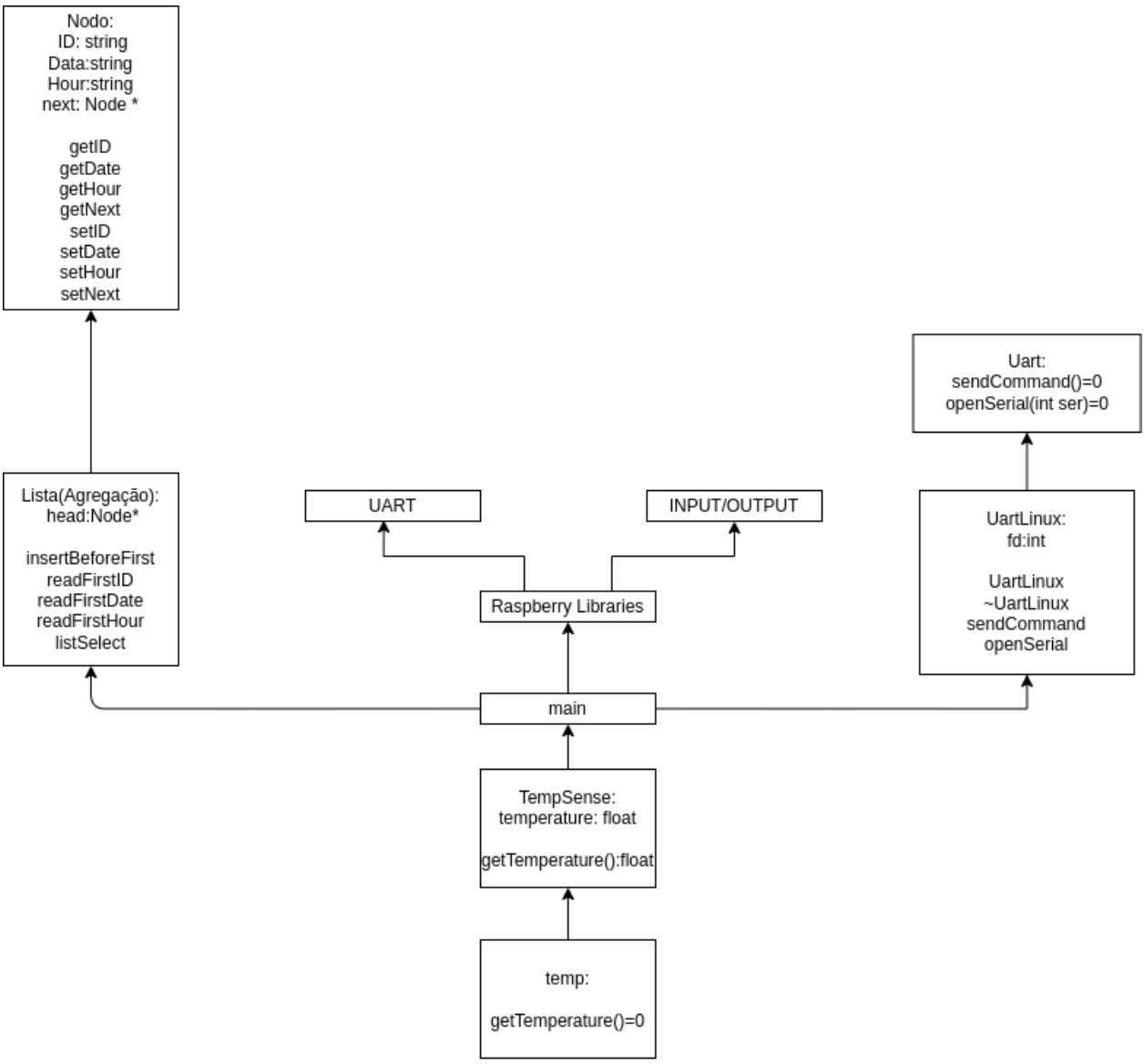

Figura 7: Diagrama de Classes do Sistema Embarcado

2.3 Descrição do aplicativo do smartphone

3 Implementação dos sistemas

3.1 Implementação do Desktop

Para se implementar o software do desktop foi utilizado a ferramenta Visual Studio com as seguintes extensões:
- C/C++ intelliSense, debugging e code browsing
- C/C++ Extension Pack
- C/C++ Themes
- C/C++ Compile Run(com gcc)

Com essas extensões instaladas e o programa escrito, basta teclar f6 para compilar e rodar o programa.

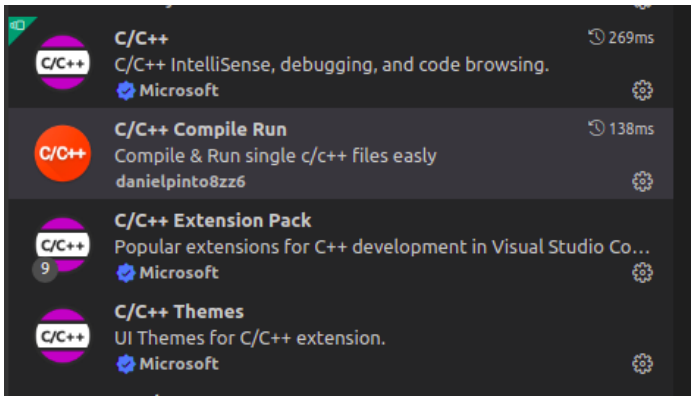

Figura 8: Extensões utilizadas para implementar o software

3.2 Implementação do Sistema Embarcado

A Raspberry pi facilitou a compilação de programas para o Pico. Seguindo os passos do guia *Getting started with Raspberry Pi Pico*, é pissivel obter o repositório do GitHub com inumeros exemplos pré programados que podem servir de base para projetos maiores, com algumas modificações no arquivo do exemplo. E para compilar, o processo também é bem simples. Após baixar o repositório, será necessário instalar Toolchain, como o GNU Embedded Toolchain for Arm, já que o chip R2040 conta com processador da arquitetura Arm e baixar o repositório SDK. As figuras 9 e 10 demonstram o passo a passo.

```
$ sudo apt update
$ sudo apt install cmake gcc-arm-none-eabi libnewlib-arm-none-eabi build-essential
```

Figura 9: Linha de comandos para se obter a Toolchain

```
$ cd pico-sdk
$ git pull
$ git submodule update
```

Figura 10: Linhas de comando para se obter o SDK

Estando em um sistema operacional Linux, abra o terminal, vá até o diretório de exemplos comentados anteriormente e:
1° Crie um diretório com o nome **build** dentro da pasta **pico-examples**;
2° Defina **PICO SDK PATH** com o comando mostrado a seguir

```
$ export PICO_SDK_PATH=../../pico-sdk
```

Figura 11: PICO SDK PATH

3° Prepare o diretório de compilação cmake executando **"cmake .."**;
4° Entre no diretório que está os arquivos a serem compilados junto deles as bibliotecas necessárias;
5° Execute **"make -4j"** e então a compilação se iniciará;
6° Copie os arquivos **.uf2** gerados para o diretório do Raspberry Pi Pico.
O guia pode ser acessado em: *https://datasheets.raspberrypi.com/pico/getting-started-with-pico.pdf*,

4 Equipamentos e ferramentas

4.1 Raspberry Pi Pico

Como foi dito anteriormente, o microcontrolador a ser utilizado nesse projeto é o Raspberry Pi Pico. Ele foi escolhido pelo seu baixo custo e alta aplicabilidade, sendo de fácil configuração. O chip utilizado foi produzido pela própria Raspberry Fundation, RP2040, o mesmo conta com 2 núcleos ARM Cortex-M0+ com um clock de até 133 MHz.

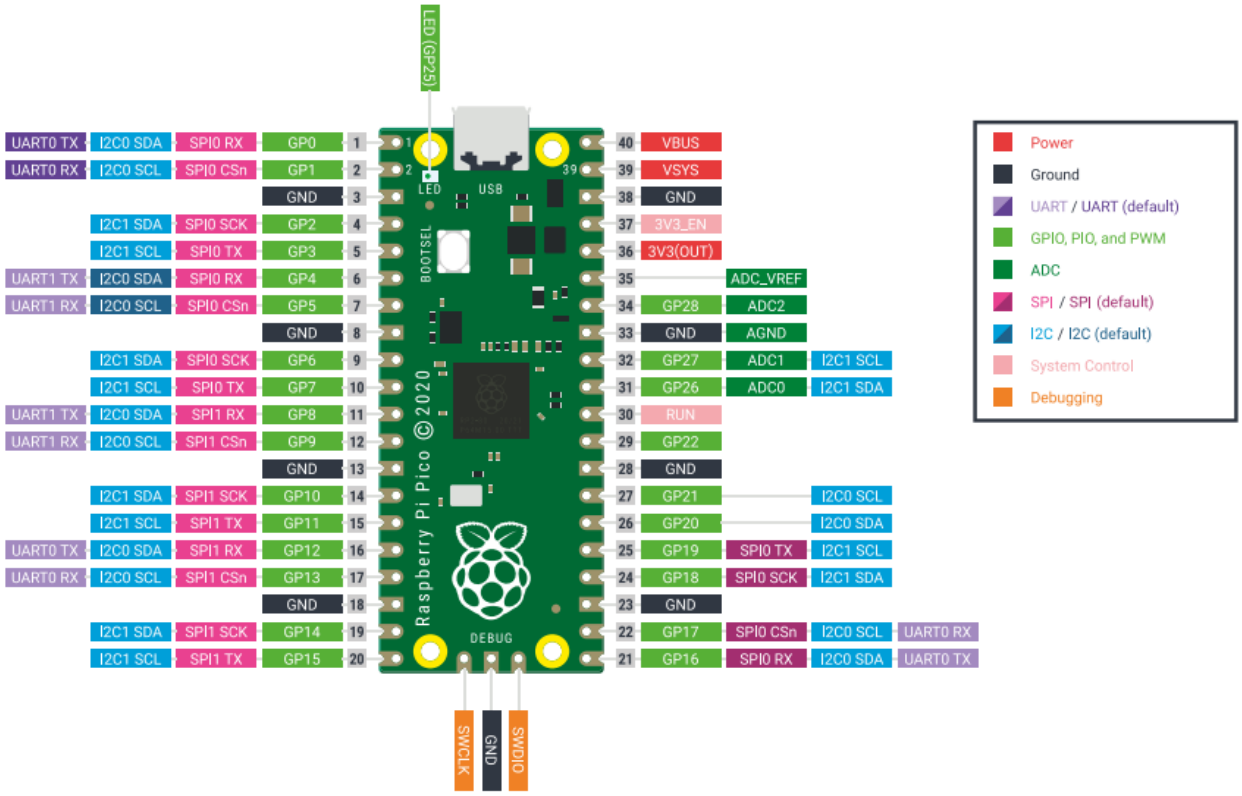

Figura 12: Portas de comunicação do Raspberry Pi Pico

4.2 Conversor Serial-USB

Foi necessário utilizar um conversor Serial-USB para transferência de dados do microcontrolador para o desktop. Foi utilizado o RS232-PL2303HX. A conexão é feita ligando o pino TX do equipamento ao TX do conversor que vai enviar os dados e na outra ponta, o outro equipamento deve habilitar sua porta RX, para receber os dados. Nesse projeto, apenas a entrada TX foi utilizada, já que só tem envio de dados em uma direção (microcontrolador para Desktop).

Figura 13: Modulo Conversor Serial-USB

4.3 Sensor de temperatura e umidade DHT11

Para medir a temperatura, foi utilizado o sensor DHT11 que é recomendado para temperaturas de 0°C à 50°C, com uma precisão de 2 graus, e umidade entre 20 a 90%, com precisão de 5%. O DHT11 possuí um controlador de 8 bits que converte o sinal de temperatura e umidade dos sensores e um sinal serial e envia ao microcontrolador através do pino de dados (Data). Suporta alimentação de 3.3 ou 5V.

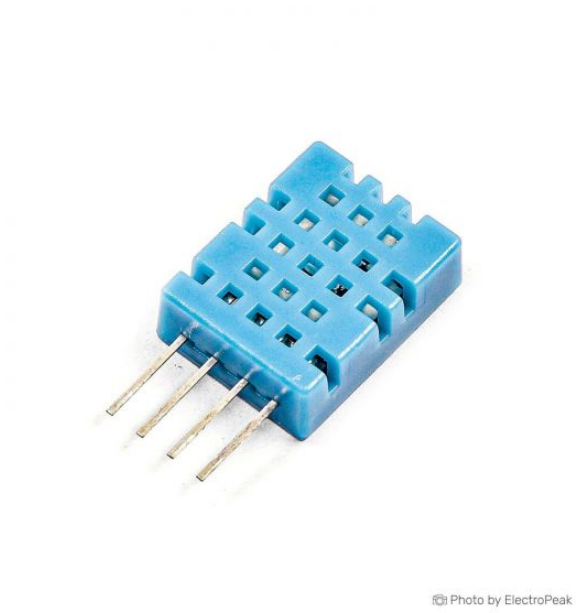

Figura 14: Sensor de temperatura DHT11

4.4 Protoboard

Para facilitar a conexão dos componentes foi utilizado uma protoboard de 830 conexões.

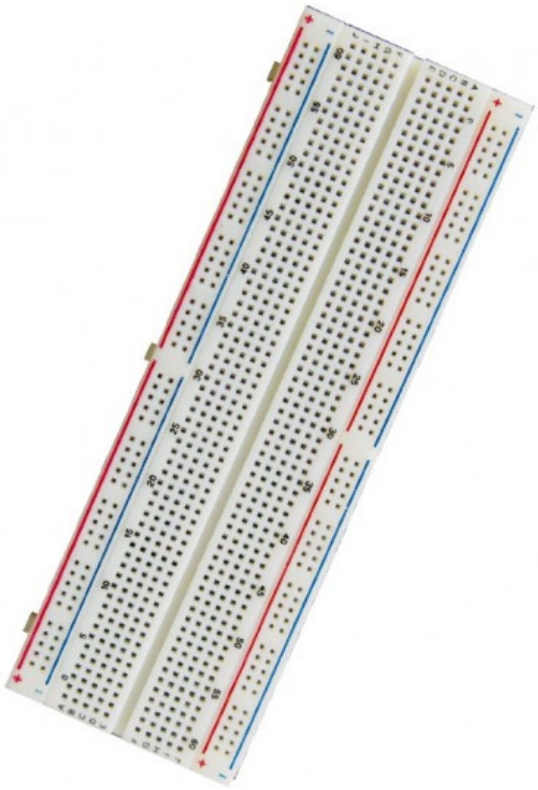

Figura 15: Protoboard

5 Conclusão

5.1 GitHub

O link para acessar o GitHub com os arquivos desse projeto:
https://github.com/Felipehxtz/EEL-7323/tree/main/EEL7323/Projeto$_{A}r_{C}$ondicionado/Codes