EEL7323-08235 Programação C++ para Sistemas Embarcados

Bússola eletrônica de 2 eixos

Gustavo Batistell[1]

[1]UFSC - Universidade Federal de Santa Catarina ,
Curso de Graduação em Engenharia Eletrônica

9 de novembro de 2023

1 Introdução

O objetivo deste projeto é desenvolver um magnetômetro eletrônico capaz de apresentar, entregar e salvar os dados de *heading*, ou direção em relação ao campo magnético terrestre, ou seja, a direção apontada pelo equipamento. Os magnetômetros eletrônicos são parte importante na instrumentação de embarcações, aeronaves, submarinos, satélites, etc e durante a realização do estágio no Laboratório do Grupo Drakkar Atlantec em Florianópolis, foi constatada a demanda para dominar tanto o ajuste quanto a calibração deste tipo de instrumento. Apesar da empresa contar com as boninas de Helmhotz com controlador e um magnetômetro de referência, tanto a parte eletrônica quanto os softwares embarcados carecem de atualização, melhoria e nova implementação para completo domínio do funcionamento. Essa necessidade de dominar (novamente) por completo o sistema surgiu devido perda de documentação e dos códigos originais com o passar dos anos.
Para isso, a tarefa do desenvolvimento inicial foi dividida em duas partes: uma parte para acionamento (driver) e outra parte para um magnetômetro, este último no qual fiquei incumbido.
Assim, utilizado um módulo micro-controlado ARM RP2040 Raspberry Pi Pico [1], um módulo magnetômetro para medir o campo magnético em pelo menos 2 eixos (seja o HMC5883 ou HoneyWell HMC1002/2300 disponíveis no laboratório), realizar o acionamento de leds para apresentar o *heading*, realizar log de dados e suportar acesso via UART/USB e Wireless. Ver diagrama da Figura 1.

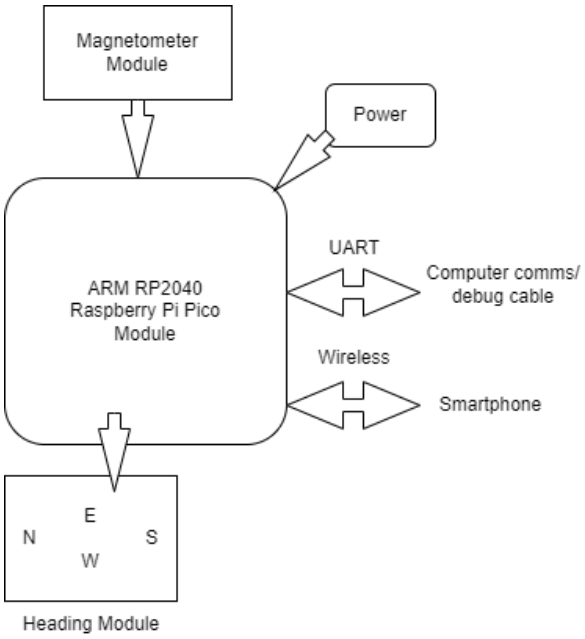

Figura 1: Diagrama de blocos revisado do projeto.

2 Requisitos

Os requisitos de projeto foram combinados com a demanda do Laboratório do Grupo Drakkar Atlantec e as especificações solicitadas pelo professor, aproveitando-se da portabilidade e modularidade com a linguagem de programação C++ e os conceitos de orientação a objetos, polimorfismo, classes, etc. Assim, foram listados os requisitos de desenvolvimento combinados para uma Bússola Eletrônica conforme segue:

- Apresente em mostrador simples de LEDs o *heading* em 8 direções: N, NE, E, SE, S, SW, W e NW.
- Utilize um módulo de magnetômetro de pelo menos 2 eixos (seja analógico ou digital) para obtenção das componentes de campo magnético.
- Com as componentes X e Y de campo magnético calcular a direção (em graus de 0 a 360) que também pode ser utilizada para o *heading*.
- Realizar log da ID, timeStamp e dados.

- Aceitar a consulta e envio deste log para um *host*.
- Aceitar comunicação via UART/USB.
- Aceitar comunicação *wireless*.
- O *host* pode conferir a listagem do log.
- O *host* pode conferir o tempo em que a Bússola Eletrônica se manteve ativa

3 Design Proposto

Como o microcontrolador ARM RP2040 [1] é *dual-core* foi decidido implementar 2 máquinas de estado. Uma delas para a aplicação da Bússola Eletrônica propriamente dita, que ao ser iniciada segue diretamente para o estado de aquisição de dados brutos, depois para um estado que calcula a direção a partir das componentes X e Y dos dados brutos, outro estado que armazena o log e finalmente um estado que exibe o *heading*. Essa primeira máquina de estados na *Trhead 1* é apresentada na Figura 2 e é importante destacar a importância dela na gravação/escrita em RAM. Já a segunda máquina de es-

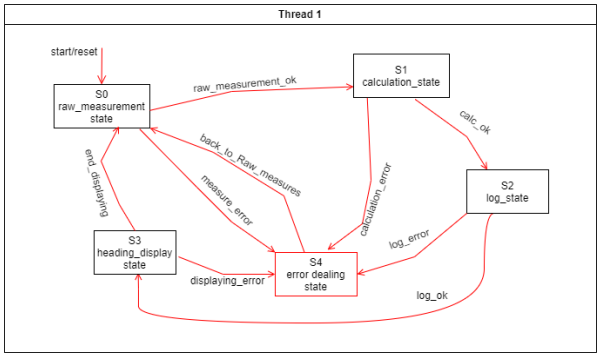

Figura 2: Maquina de estados principal (com gravação em memória).

tados na *Trhead 2* é incumbida de controlar a conectividade UART e *Wireless*, conforme mostrado na a Figura 3. Esta máquina de estados tem um estado de *idle* para quando não houver conexão, um estado para conexão *wireless* estabelecida, um estado para o wirelessRequest, um estado para quando a conexão UART é estabelecida, um estado para o UARTrequest e um estado para tratamento de erros. Esta máquina desta *Trhead 2* apenas acessa os dados. Essa distinção entre gravação e acesso deste projeto em específico é importante para evitar problemas de leitura e escrita usuais em programação concorrente quando mais de uma *trhead* é possível.

Para garantir a modularidade do projeto, o diagrama de classes proposto utilizou de 5 níveis de abstração que são listados abaixo em uma abordagem "de dentro para fora":

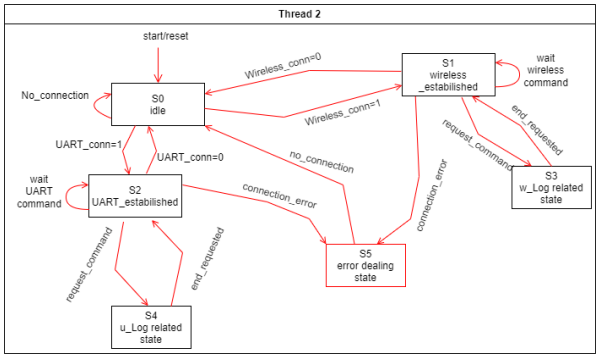

Figura 3: Maquina de estados secundária para conexões UART e Wireless (apenas acesso a memória).

- Hardware Abstraction Layer (HAL): Responsável por abstrair o hardware subjacente, incluindo os módulos do SDK da Raspberry Pi Pico.
- Device Drivers Layer: Gerencia os drivers para GPIO, UART, comunicação sem fio e o magnetômetro.
- Processing Layer: Lida com o processamento dos dados, incluindo a classe 'DirectionCalculator'. Esta camada também pode ser usada para implementações futuras de técnicas de Digital Signal Processing como média móvel, dizimação, etc.
- Control/Application Layer: Responsável por controlar as partes funcionais do equipamento. É o ponto central onde as operações funcionais são coordenadas.
- User Interaction Layer: É a camada mais externa que lida com a interação do usuário, incluindo as classes 'HeadingDisplay', 'Log', e as interfaces de comunicação com o usuário.

Essa organização se apresentou clara e permite uma separação adequada de responsabilidades em cada camada de classes do sistema, o que é uma boa prática para melhor documentar, estruturar e facilitar a compreensão e manutenção do projeto. O diagrama de classes pode ser visto na Figura 4 do Apêndice A.

Referências

[1] R. Pi. Rp2040 datasheet a microcontroller by raspberry pi. [Online]. Available: https://datasheets.raspberrypi.com/rp2040/rp2040-datasheet.pdf

Apêndice A

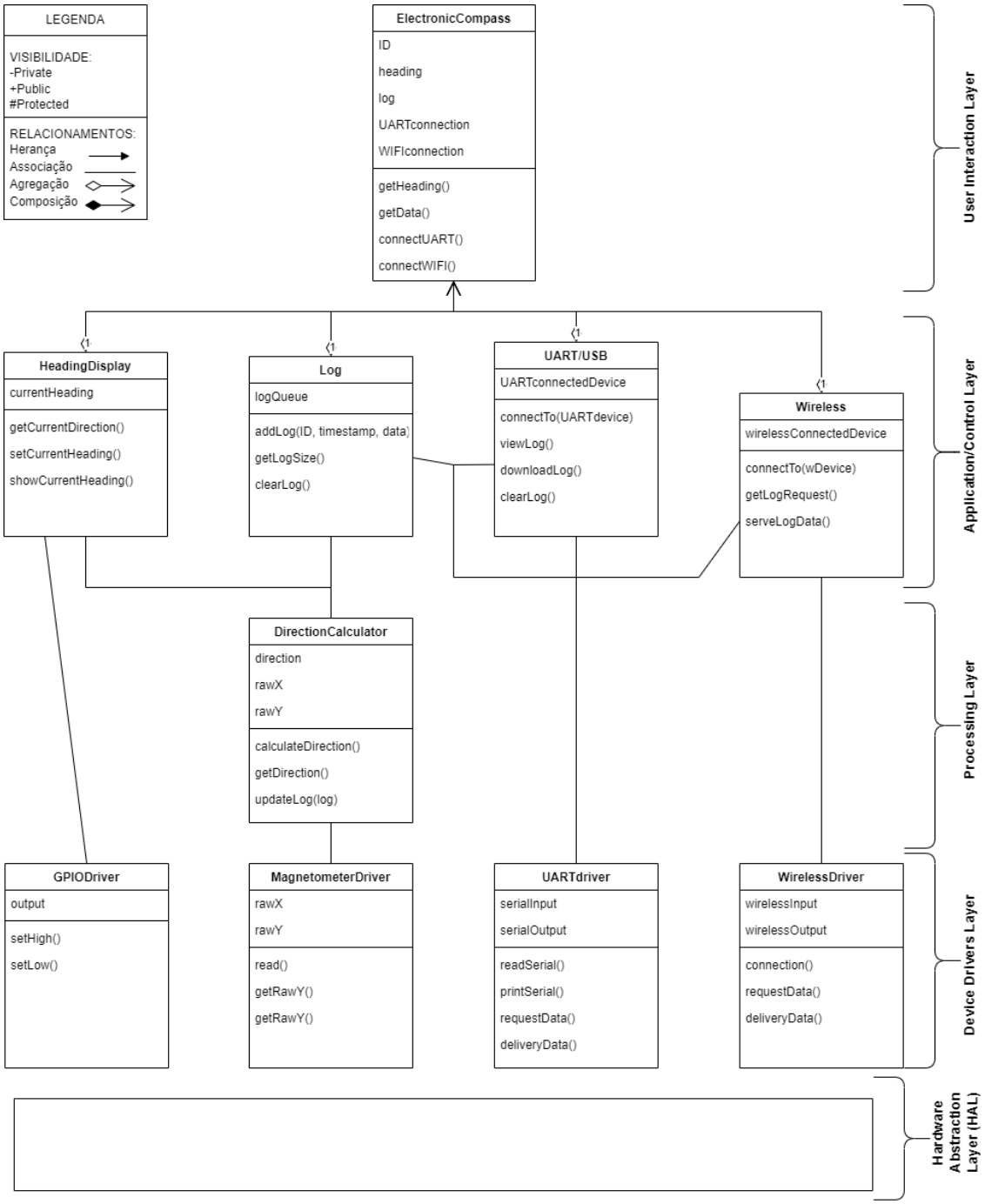

Figura 4: Diagrama de classes do projeto organizado em camadas.