Medição de Sinais Vitais em Bovinos Leiteiro Utilizando Datalogger

Elevandro Cazalli

e.cazalli@grad.ufsc.br, Eng. Eletrônica

1 Introdução

O monitoramento contínuo dos sinais vitais de vacas leiteiras é essencial para a qualidade e quantidade de leite produzido. Ele permite intervenções rápidas em caso de doença, melhorando a produtividade. Além disso, ajuda os produtores a otimizar as práticas de manejo através da análise de dados ao longo do tempo.

2 Projeto Proposto

Este projeto envolve o desenvolvimento de um sistema de monitoramento de sinais vitais para vacas leiteiras, visando otimizar a produção de leite ao garantir a saúde animal por meio do rastreamento contínuo de temperatura e frequência cardíaca. O objetivo principal é facilitar intervenções rápidas em caso de anomalias de saúde, aumentando assim a produtividade e possibilitando o refinamento das práticas de manejo por meio da análise de dados longitudinais. O sistema consiste em três componentes principais. O primeiro é um dispositivo embutido de registro de dados, encarregado da coleta e processamento interno de dados vitais. O segundo é um computador hospedeiro, designado para armazenar um log de eventos e exibir as informações coletadas para análises posteriores. O terceiro é um aplicativo de smartphone hospedeiro, que atua como uma interface para a aquisição de dados em tempo real e alertas do sistema. Todas as funcionalidades do sistema são projetadas usando uma abordagem orientada a objetos em C++, estruturadas para serem escaláveis para a futura integração de recursos adicionais de monitoramento, como dados de ECG e acelerômetro. O foco do projeto está no desenvolvimento de software, com considerações de hardware limitadas ao essencial necessário para as operações básicas do sistema. A arquitetura do sistema proposto é detalhada na Figura 1.

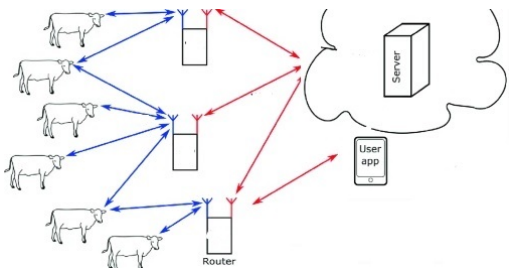

Figure 1: Arquitetura

3 Sistema Embarcado

Optou-se pelo microcontrolador ESP32-WROOM-32 da Espressif, que possui uma configuração de dois núcleos, como o coração do hardware do sistema embarcado. A configuração de hardware é complementada por um sensor de temperatura digital LM35 e o módulo MAX30102 para monitoramento da frequência cardíaca, que transmite dados por meio de uma interface I2C.

O software do sistema embarcado foi criado com o uso do ESP-IDF, o ambiente de desenvolvimento oficial da Espressif, que traz uma versão customizada do FreeRTOS, facilitando a criação de aplicações multitarefas. O firmware foi estrategicamente projetado para maximizar esses recursos, distribuindo as operações em três tarefas principais.

A tarefa primária é encarregada de ler os dados dos sensores, armazená-los e avaliar sua normalidade. Uma segunda tarefa gerencia a atualização contínua do relógio e do calendário. A terceira e última tarefa administra a interface serial, processando os comandos recebidos e transmitindo dados.

A comunicação entre o sistema e o dispositivo 'host' ou smartphone é estruturada seguindo um protocolo mestre-escravo. O dispositivo mestre envia um pacote contendo o ID do microcontrolador, o tipo de comando, o tamanho dos dados e os próprios dados. O sistema embarcado, atuando como escravo, processa esse pacote e, em resposta, envia um pacote contendo o ID, o tamanho dos dados e, conforme necessário, os dados solicitados, mensagens de erro ou uma confirmação de recebimento quando o comando se refere apenas à configuração. A classe "Conf" é responsável por manter a configuração do sistema, disponibilizando dados e métodos essenciais para a configuração do mesmo.

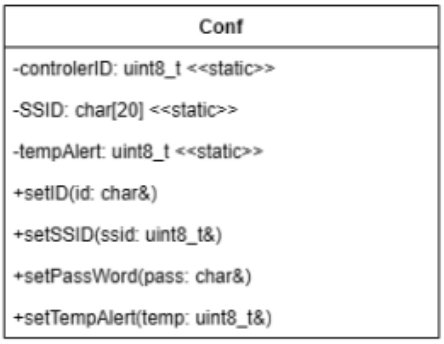

Figure 2: Classe Conf

A Classe 'ClockCalendar' administra e fornece ao sistema um relógio integrado com calendário, cujas informações são utilizadas para o registro temporal dos logs. A classe 'DataLogger' atua como o componente central do sistema, encarregada de gerenciar uma fila de dados, denominada 'dataQueue', e um semáforo, 'logMutex', para controlar o acesso concorrente aos dados. Possui funcionalidades para armazenar dados coletados (storeData) e para processar a comunicação tanto por UART (processLogUARTTask) quanto por Wi-Fi (processLogWiFiTask). A fila de dados é implementada pela classe 'Queue', que permite a inclusão (enQueue) e a remoção (deQueue) de dados genéricos, além

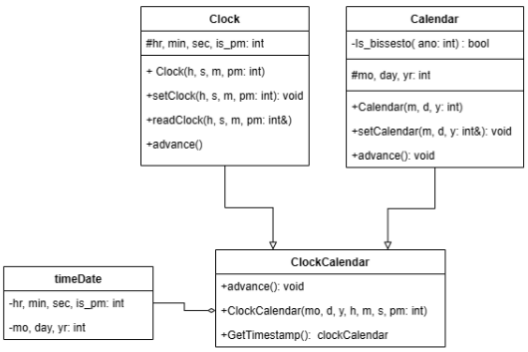

Figure 3: Classe ClockCalendar

de verificar se a fila está vazia (empty) e definir seu tamanho máximo (setMaxSize). Cada item na fila é um Node, que contém os dados e um ponteiro para o próximo item. A estrutura 'logData' é utilizada para armazenar os registros de dados, que incluem o ID do controlador, a data e hora do registro, assim como as leituras dos sensores de temperatura.

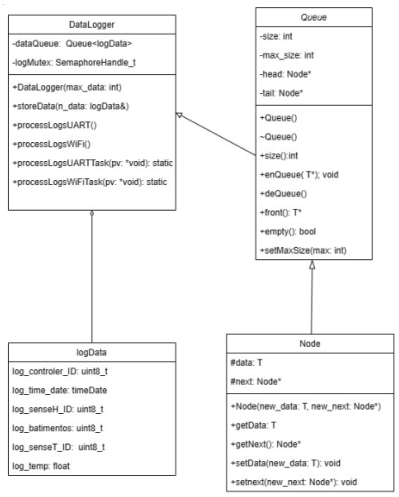

Figure 4: Datalogger

Dentro da arquitetura do sistema, a classe abstrata Sensor serve como um modelo fundamental para os sensores, estabelecendo uma estrutura comum para diferentes tipos de medições. As classes derivadas 'HeartSensor' e 'TempSensor' especializam-se na aquisição de dados, com a primeira monitorando a frequência cardíaca e a segunda medindo a temperatura, cada uma processando as informações específicas conforme a sua aplicação.

As interfaces HandlerUART e HandlerWiFi são componentes críticos do sistema, fornecendo a infraestrutura necessária para a comunicação via UART e Wi-Fi, respectivamente. Ambas possuem um conjunto de métodos essenciais que facilitam a interação com dispositivos externos: 'processCommand' para inter-

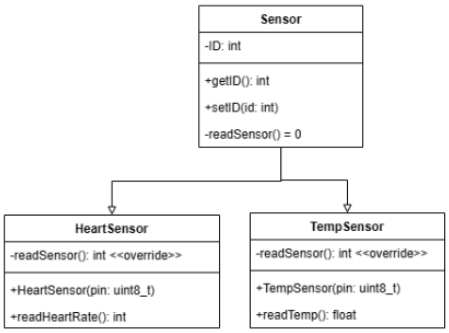

Figure 5: Sensores

pretar comandos recebidos, ‘receive‘ para acolher dados de entrada, ‘sendData‘ para a transmissão de informações e ‘sendCommand‘ para enviar comandos específicos. Essas interfaces são a espinha dorsal para o fluxo de dados bidirecional e controle de comandos entre o sistema embarcado e o mundo externo.

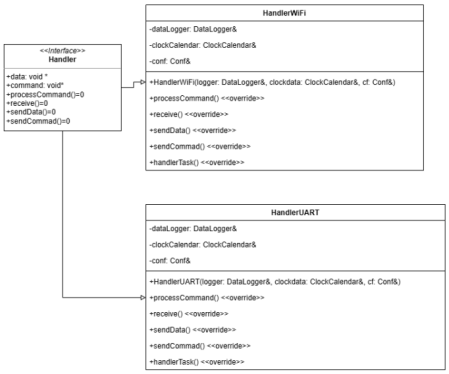

Figure 6: Interfaces

Na estrutura do sistema, a classe ’Actuator’ estabelece a base para os atuadores, oferecendo um conjunto de funcionalidades genéricas para dispositivos de ação. A classe derivada ’WarningLight’ é especializada na gestão de um atuador luminoso, especificamente uma luz de aviso, desempenhando a função de alerta visual. Complementando esta configuração, temos o ’FeedbackController’, que é responsável por controlar o atuador ’WarningLight’, permitindo que ele atue como um indicador visual que reage aos dados recebidos dos sensores, garantindo assim que o feedback visual apropriado seja fornecido em resposta a condições específicas detectadas pelo sistema.

4 Host - PC

O componente ”Host - PC” no sistema de monitoramento serve como uma ponte entre o usuário e o sistema embarcado. Ele desempenha um papel fundamental na recepção de dados do sistema embarcado, utilizando a classe ‘SerialCom-

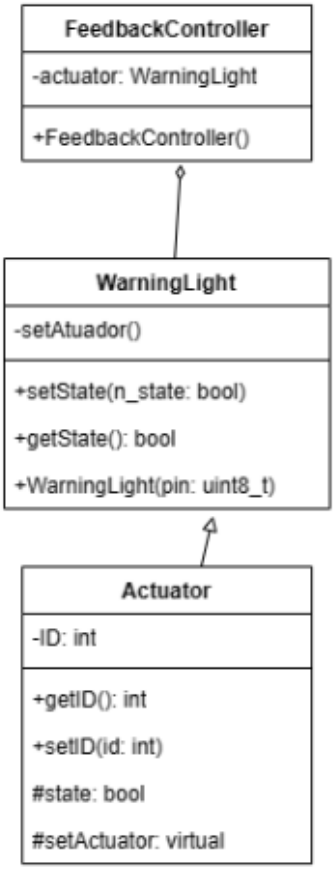

Figure 7: Controlador

munication‘ para tal. Após receber os dados, o ”Host - PC” os processa e os armazena eficientemente no ‘LogDatabase‘. Isso possibilita que o usuário acesse, visualize e analise os dados, o que é essencial para um monitoramento eficaz dos sinais vitais dos bovinos. A interação harmoniosa entre o hardware e o software garante uma monitorização precisa e uma tomada de decisões informadas na gestão do gado. A estrutura do software, desenvolvida em C++ e orientada a objetos, inclui diversas classes interligadas que gerenciam a comunicação de dados e a interface com o usuário. A interface de linha de comando (CLI), implementada no programa principal (‘main‘), disponível no GitHub (https://github.com/ecazalli/ProjetoFinalEEL7323), oferece uma interação direta e simplificada com o usuário, permitindo o envio de comandos ao sistema embarcado e a listagem de dados dos eventos coletados.

4.1 Estrutura de Classes

- **LogDatabase:** Esta classe é encarregada do armazenamento e gerenciamento dos dados recebidos do sistema embarcado. Ela armazena informações detalhadas de cada evento, como ID do controlador, data e

hora, frequência cardíaca e temperatura, em uma estrutura de lista encadeada. As funcionalidades da classe incluem a adição de registros de log e a listagem de eventos em um intervalo de datas específico. Isso facilita a análise de tendências e a identificação de problemas de saúde nos animais. A Figura 8 ilustra o diagrama da classe LogDatabase, mostrando sua relação com as classes herdadas e as estruturas de dados utilizadas.

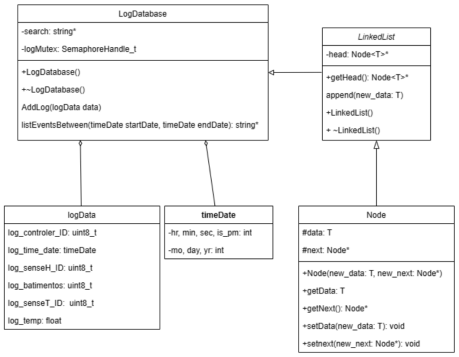

Figure 8: Base de dados

- **SerialCommunication:** Esta classe estabelece e gerencia a comunicação entre o sistema embarcado e o PC através de uma porta serial. É fundamental para o envio de comandos e para a recepção de logs de dados do sistema embarcado. A classe utiliza a estrutura 'logData' para receber, processar e encaminhar os dados coletados para armazenamento. Uma análise mais detalhada dessa classe e suas funcionalidades pode ser vista na Figura 9.

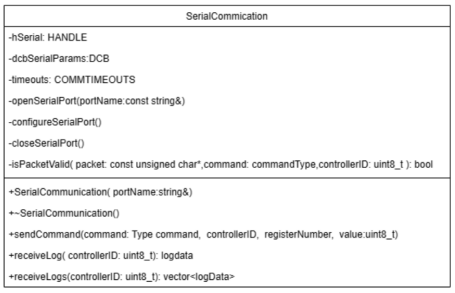

Figure 9: Comunicação Serial

5 Host - Sistema Mobile

Optei por utilizar o 'framework' multiplataforma Qt para o desenvolvimento do aplicativo do sistema 'host'. O Qt permite desenvolver aplicativos de forma unificada, habilitando a compilação em diversas plataformas com mínimas modificações no código fonte. Esta abordagem assegura que o aplicativo seja compatível tanto com dispositivos móveis quanto com PCs. Para adaptar o sistema

'host' a este cenário, foi necessário modificar algumas classes existentes. Além disso, introduzimos uma nova classe denominada 'WifiCommunication'. Esta classe é projetada para replicar as funcionalidades da comunicação serial prévia, mas agora utilizando comunicação Wi-Fi. As figuras 10 e 11 ilustram as classes de comunicação recém-integradas ao sistema.

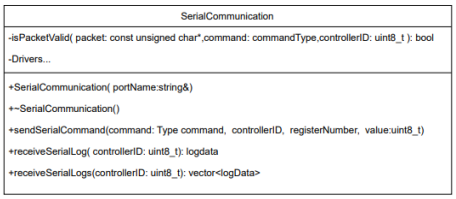

Figure 10: Comunicação Serial Aplicativo

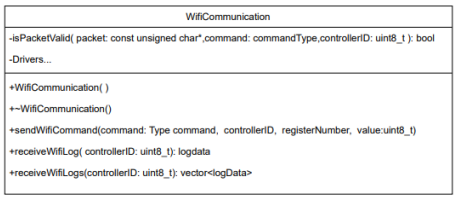

Figure 11: Comunicação Wi-Fi Aplicativo

Para avançarmos nos testes do aplicativo e expandirmos suas capacidades, efetuamos alterações na classe 'LogDatabase', visando aprimorar a persistência de dados. Implementamos duas novas funcionalidades: 'LoadLogsFromFile()' e 'AddLogToFile()'. Essas funções operam de maneira automática dentro da classe. Quando o aplicativo é inicializado, os dados são lidos do arquivo correspondente e carregados na memória. Além disso, qualquer novo dado recebido é imediatamente armazenado no arquivo, assegurando uma atualização contínua e eficiente do banco de dados de logs.

5.1 Interface

A interface do aplicativo foi projetada para ser extremamente intuitiva e simples, consistindo em apenas uma tela que contém todas as funcionalidades disponíveis. Durante o processo de compilação, certos elementos serão ocultados com base no sistema operacional de destino. Por exemplo, na versão compilada para Android, os campos associados à interface serial e as configurações do sistema embarcado serão desabilitados, restringindo o uso ao pedido de Logs do sistema. Além disso, a interface inclui uma seção para a pesquisa de logs armazenados. Aqui, o usuário pode buscar logs dentro de um intervalo de datas específico; se houver logs correspondentes a esse período, eles serão exibidos em uma lista, com cada log ocupando uma linha separada.

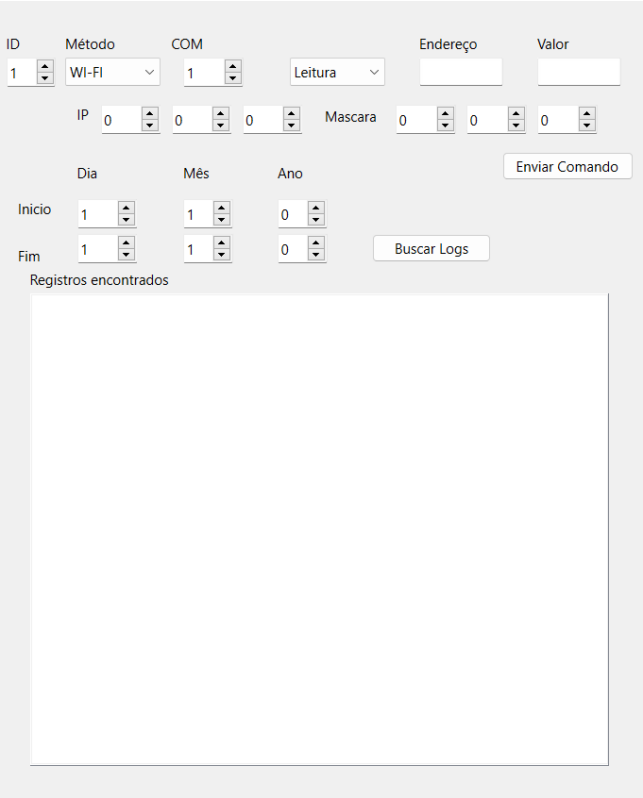

Figure 12: Layout Aplicativo