UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO
CENTRO TECNOLÓGICO
DEPARTAMENTO DE ENGENHARIA ELÉTRICA
PROJETO DE GRADUAÇÃO

JOÃO AGRICIO LOPES BARBOSA

PROJETO DE SISTEMA EMBARCADO PARA CONTROLE SEGURO DE TRAVAS ELETRÔNICAS

VITÓRIA
2021

JOÃO AGRICIO LOPES BARBOSA

PROJETO DE SISTEMA EMBARCADO PARA CONTROLE SEGURO DE TRAVAS ELETRÔNICAS

Parte manuscrita do Projeto de Graduação do aluno **João Agricio Lopes Barbosa**, apresentado ao Departamento de Engenharia Elétrica do Centro Tecnológico da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do grau de Engenheiro Eletricista.

Orientador: Prof. Dr. Camilo Arturo Rodríguez Díaz
Coorientador: Prof. Dr. Hans Jorg Andreas Schneebeli

VITÓRIA
2021

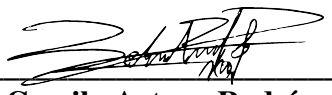
JOÃO AGRICIO LOPES BARBOSA

PROJETO DE SISTEMA EMBARCADO PARA CONTROLE SEGURO DE TRAVAS ELETRÔNICAS

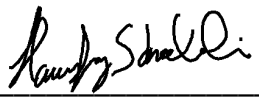
Parte manuscrita do Projeto de Graduação do aluno **João Agricio Lopes Barbosa**, apresentado ao Departamento de Engenharia Elétrica do Centro Tecnológico da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do grau de Engenheiro Eletricista.

Aprovada em 13 de outubro de 2021.

COMISSÃO EXAMINADORA:

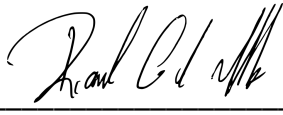


Prof. Dr. Camilo Arturo Rodríguez Díaz
Universidade Federal do Espírito Santo
Orientador



Prof. Dr. Hans Jorg Andreas Schneebeli
Universidade Federal do Espírito Santo
Coorientador

Profa. Dra. Eliete Maria de Oliveira Caldeira
Universidade Federal do Espírito Santo
Examinador



Prof. Dr. Ricardo Carminati de Mello
Universidade Federal do Espírito Santo
Examinador

Aos meus familiares e amigos.

AGRADECIMENTOS

Agradeço a minha família pelo suporte durante todos os momentos vividos. Agradeço também aos meus amigos pelas boas conversas, discussões e experiências que certamente auxiliaram em meu desenvolvimento pessoal e acadêmico.

Agradeço também aos meus orientadores pelos ensinamentos, paciência e apoio. A orientação prestada foi primordial para a execução deste trabalho.

RESUMO

Neste trabalho é apresentado um modelo generalista para um sistema embarcado capaz de controlar travas eletrônicas de forma segura. A estrutura proposta é desenvolvida por meio da análise de trabalhos semelhantes em conjunto com a aplicação de métodos bem estabelecidos na literatura a respeito do desenvolvimento de sistemas embarcados seguros. Diferente de abordagens convencionais, em que o sistema é construído com foco em funcionalidades e apenas ao fim do desenvolvimento adicionam-se camadas de segurança, neste trabalho, características de segurança são consideradas desde a concepção e levantamento de requisitos. Desta forma, são explicitados o processo de desenvolvimento e as justificativas das escolhas tomadas para o estabelecimento da estrutura geral proposta. Ademais, é implementada uma prova de conceito utilizando o exemplo de um bicicletário eletrônico para comprovar a utilidade da estrutura proposta, por meio da qual pôde-se demonstrar a utilidade das medidas gerais de segurança adotadas para o desenvolvimento do sistema. São abordados no projeto: o levantamento dos requisitos de um dispositivo, a definição da arquitetura de um sistema embarcado seguro, o processo de escolha da estrutura de *software,* o processo de definição do *hardware* e a implementação de um sistema mínimo.

Palavras-chave: Sistemas embarcados. Travas eletrônicas. *Software*. *Hardware*.

ABSTRACT

This graduation project presents a generalist model for an embedded system capable of safely controlling electronic locks. The proposed structure is developed through the analysis of similar works together with the application of well-established methods available in the literature on the development of secure embedded systems. Unlike conventional approaches, in which the system is built with a focus on functionality and the security layers are left to be added only by the end of development process, in this work, security features are considered from the conception and requirements gathering. In this way, development process and justifications for the choices made to establish the proposed general structure are explained in detail. Furthermore, a proof of concept is implemented using the example of an electronic bicycle rack to prove the usefulness of the proposed structure, through which the usefulness of the general security measures adopted for the development of the system could be demonstrated. The project addresses: the definition of the device requirements, the definition of a secure architecture for an embedded system, the process of choosing the software structure, the process of defining the hardware and the implementation of a minimal system.

Keywords: Embedded systems. Electronic locks. Software. Hardware.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

API	*Application Programming Interface*
CC	Controlador Concentrador
DFD	*Data Flow Diagram*
HW	*Hardware*
HAL	*Hardware Abstraction Layer*
IHM	Interface Homem-Máquina
IoT	*Internet of Things*
LED	*Light-Emitting Diode*
MCU	*Microcontroller*
PC	*Personal Computer*
RTOS	*Real-Time Operating System*
SW	*Software*
TAD	Tipo Abstrato de Dados
UART	*Universal Asynchronous Receiver-Transmitter*
UPS	*Uninterruptible Power Supply*
USB	*Universal Serial Bus*

SUMÁRIO

1 INTRODUÇÃO

Os avanços das tecnologias de comunicação e de microcontroladores possibilitam que sistemas embarcados estejam cada vez mais presentes em objetos de uso cotidiano. A popularização destes sistemas se dá, em grande parte, por conta da diminuição, ao longo do tempo, do preço e do espaço ocupado por eles em conjunto com o aumento das funcionalidades adicionadas por meio de sua aplicação (MATTERN; FLOERKEMEIER, 2010). Encontram-se diversas definições para sistemas embarcados na literatura, as quais costumam se diferenciar por algumas particularidades. Neste trabalho, define-se um *sistema embarcado* como um sistema eletrônico, composto por microprocessador e *software* (SW), que possua a capacidade de realizar um conjunto estabelecido de tarefas (KLEIDERMACHER, KLEIDERMACHER, 2012, p. 3).

A tendência de popularização de sistemas embarcados vai ao encontro do conceito de internet das coisas (IoT, do inglês *Internet of Things*). A aplicação de IoT visa que objetos do cotidiano passem a estar conectados ao mundo virtual. Com essa mudança, espera-se revolucionar as funcionalidades presentes nos objetos, sobre os quais ter-se-á poder de controle e monitoramento por meio de comunicação remota (MATTERN; FLOERKEMEIER, 2010). Pode ser observada a implementação desses sistemas em produtos como geladeiras inteligentes (AHELEROF et al., 2020), babás eletrônicas (ZIGANSHIN; NUMEROV; VYGOLOV, 2010), equipamentos para monitoramento remoto de pacientes em situação de risco (CHIUCHISAN; GEMAN, 2016), dentre outros. No entanto, existem muitas áreas da sociedade em que soluções desta natureza ainda não foram amplamente desenvolvidas.

Uma aplicação em especial é destacada neste trabalho: o controle de travas eletrônicas. Sistemas de travas estão presentes em diversos ambientes, como por exemplo, para gerenciamento de guarda-volumes e para controle de estacionamentos de veículos. Estes, por sua vez, ainda são majoritariamente não computadorizados e, em consequência, não possuem o nível de automatização, controle e monitoramento que pode ser alcançado com o uso de sistemas embarcados.

Parte da relutância à implementação de sistemas computadorizados para solucionar o problema de controle de travas pode ser justificado pela necessidade de segurança e robustez, tendo em

vista que um sistema responsável por controlar o acesso a bens de terceiros não pode falhar. Por sua vez, encontra-se na literatura uma grande lista de casos em que dispositivos IoT apresentaram falhas de segurança e expuseram seus usuários a situações de vulnerabilidade (MENEGHELLO et al., 2019).

O termo *segurança* é definido neste texto como a mitigação dos riscos de dano mediante a presença de ameaças (KLEIDERMACHER, KLEIDERMACHER, 2012, p. 1). Ou seja, representa o quão resiliente o sistema é às tentativas de violação. A segurança é uma das características que compõem um sistema *robusto*. Além de ser seguro, para que o sistema seja considerado robusto, este deve possuir formas preestabelecidas para lidar com situações adversas. Por exemplo, um sistema que prevê reação mediante a existência de alguma falha em sua alimentação de energia ou em seus periféricos, é mais robusto que um sistema semelhante que não considera a possibilidade de falhas. Para um sistema de controle de travas eletrônicas, abrir as travas mediante ausência de energia é uma falha grave de robustez.

Neste trabalho será apresentada uma solução geral, com foco em segurança, para sistemas de controle de travas eletrônicas. Para demonstrar uma aplicação e comprovar a generalidade da solução, será utilizado o caso exemplo de um bicicletário eletrônico. A escolha do caso exemplo se dá tendo em vista que bicicletas representam um método de locomoção alternativo capaz de contribuir para a mobilidade urbana e que medidas têm sido tomadas, ao redor do mundo, para incentivar seu uso (DILL, 2009). No entanto, a alta taxa de furtos é um fator que desencoraja a aderência de usuários (LIEROP; GRIMSRUDEL-GENEIDY, 2015). Sendo assim, o emprego de tecnologias que permitam o aumento das funcionalidades presentes nos bicicletários pode contribuir para a popularização do ciclismo como alternativa de transporte.

1.1 Justificativas

O trabalho proposto neste documento tem o potencial de estruturar uma solução que pode ser aplicada a diversos dispositivos. Na literatura, trabalhos que envolvem o controle de travas eletrônicas apresentam soluções não gerais. São exemplos o guarda-volumes para agências bancárias utilizando biometria (HTWE; HTUN; TUN, 2015) e a implementação de um armário compartilhado para recebimento de encomendas (ZE-HONG; GUANG-YUAN, 2015). Destaca-se que o controle de travas eletrônicas possui diversas características comuns a

diferentes aplicações e, além disso, pode exigir medidas rígidas quanto à robustez e à segurança. Porém, ambos os temas são inexistentes ou brevemente mencionados nos trabalhos analisados.

A apresentação de uma solução geral, utilizando conceitos de projeto de sistemas reutilizáveis (BENINGO, 2017) e de código limpo (MARTIN, 2009), permite que novos projetos se beneficiem dos métodos aplicados neste trabalho e, com isso, possam dedicar mais recursos às especificidades da aplicação intencionada.

1.2 Objetivos

O objetivo geral do projeto é desenvolver e apresentar o processo de desenvolvimento de um sistema embarcado, aplicado ao controle de travas eletrônicas, capaz de se comunicar e receber comandos de usuários de forma remota, considerando-se os aspectos de segurança quanto ao *hardware* (HW) e quanto ao SW. Este trabalho se detém à implementação do sistema desconectado da internet (apesar de prever estruturalmente a possibilidade de conexão) haja vista que existem diversos trabalhos consolidados que apresentam métodos para conexão segura de sistemas à rede.

Para alcançar o objetivo geral, foram definidos os seguintes objetivos específicos:

a) Desenvolver uma arquitetura reutilizável para solução do problema de travas eletrônicas;
b) Implementar medidas que garantam a atuação correta do sistema sobre as travas;
c) Demonstrar o processo de desenvolvimento de um sistema embarcado seguro e de *software* reutilizável.

1.3 Metodologia

O trabalho exposto apresenta o desenvolvimento de um sistema para controle de travas eletrônicas, de forma segura e robusta, que pode ser utilizado para diferentes aplicações. A abordagem apresentada diferencia-se de abordagens tradicionais pelo foco dado, desde a fase de concepção do sistema, aos requisitos de segurança e robustez.

São analisadas, de forma comparativa, estruturas consolidadas na literatura para o desenvolvimento de sistemas semelhantes e é escolhida a que melhor atende aos requisitos da solução geral intencionada.

Com base na estrutura de desenvolvimento do sistema, define-se o problema em maior nível de detalhe por meio da especificação de requisitos de segurança, robustez e funcionalidade e da definição das partes componentes do sistema e suas interfaces. São utilizados trabalhos afins como base para definição de requisitos funcionais.

Mediante o estabelecimento dos requisitos gerais, é feito o levantamento de funcionalidades de HW, SW e comunicação necessárias para que se atenda ao propósito do sistema. Comparações quantitativas e qualitativas são empregadas.

É utilizado o caso exemplo de um bicicletário para que se demonstre a funcionalidade do sistema desenvolvido e a generalidade da solução proposta. São apontadas de forma comparativa as funcionalidades da solução geral e os requisitos da solução específica utilizada.

Por fim, é apresentado o desenvolvimento de um sistema físico simples como prova de conceito e são realizados testes empíricos que reproduzem situações reais de funcionamento. Os resultados dos testes são utilizados para comprovar o atendimento dos requisitos estabelecidos para o sistema, tanto em solução geral quanto em aplicação específica.

1.4 Estrutura do Trabalho

Este trabalho é dividido em 5 capítulos e estes são apresentadas sucintamente a seguir.

No Capítulo 1 é exposta a introdução ao tema e ao projeto desenvolvido. É abordada a situação atual de pesquisas relacionadas, aplicações semelhantes, importância e justificativa do trabalho, os objetivos e a metodologia aplicada. Em seguida, no Capítulo 2, é exposto o embasamento teórico utilizado para o correto desenvolvimento de sistemas embarcados no que tange o HW e o SW. Ademais, é apresentado o problema do controle seguro de travas eletrônicas e a especificação de uma solução generalista com base nos conceitos abordados.

No Capítulo 3, o caso exemplo é apresentado em grande nível de detalhe. São explicitadas as generalidades e as particularidades da solução e são expostos os requisitos e as diretivas de segurança. É realizada também a definição das diretivas de projeto do HW e do SW, passando por levantamento de requisitos funcionais e de segurança além de demonstrar as aplicações de técnicas que resultam em um SW reutilizável e pouco dependente do HW.

No Capítulo 4 é explicitado o processo de implementação com base nas diretivas definidas nos capítulos 2 e 3 e são mostrados também os resultados quanto ao HW e ao SW. Por fim, o Capítulo 5 é dedicado a conclusões e recomendações de trabalhos futuros.

2 DESENVOLVIMENTO DE SISTEMAS EMBARCADOS SEGUROS – TRAVAS ELETRÔNICAS

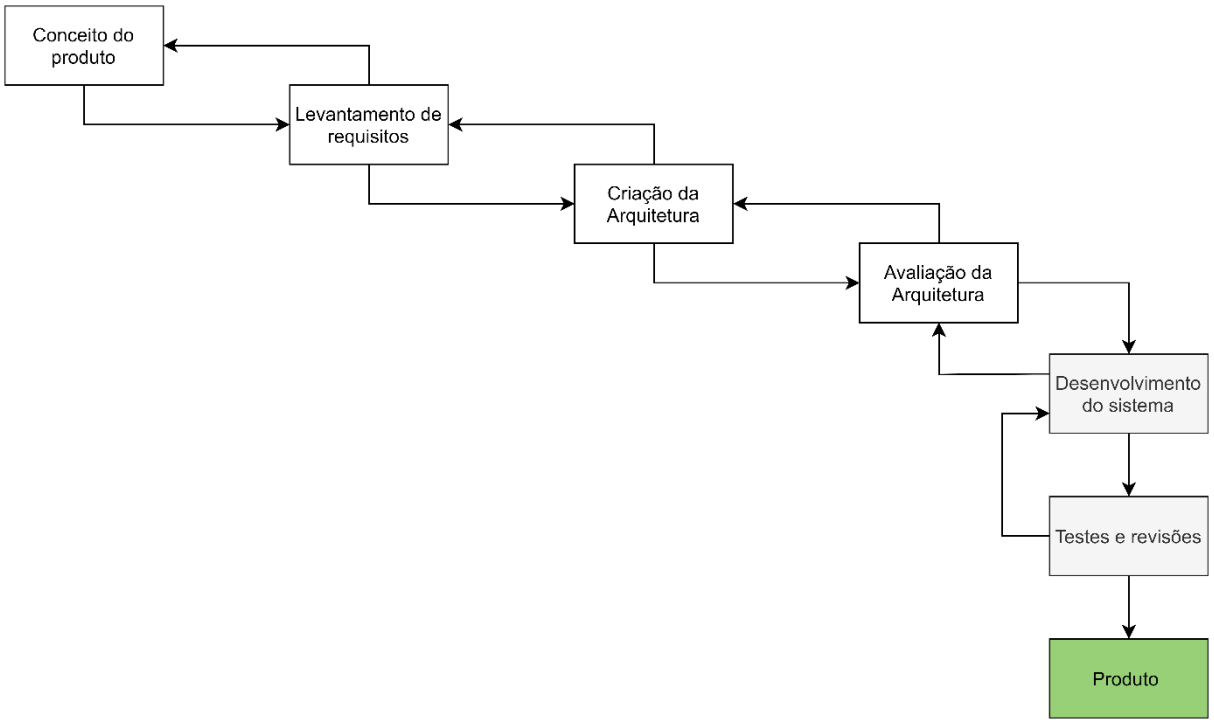
O processo de desenvolvimento de sistemas embarcados seguros envolve diversas etapas e diferentes metodologias podem ser aplicadas, a depender do tipo de sistema que se deseja construir. Neste Capítulo serão apresentados métodos de projeto de sistemas seguros adotados neste trabalho, suas diferenças em relação aos métodos convencionais, e a aplicação dos métodos para o desenvolvimento de uma arquitetura para solucionar o problema de controle de travas eletrônicas.

Dentre os métodos aplicados para o desenvolvimento de sistemas embarcados, os mais primordiais são referentes a qual será o modelo de ciclo de desenvolvimento adotado. Neste quesito, destacam-se quatro mais comuns (NOERGAARD, 2005, p.7). São eles:

a) **Modelo *big-bang* –** praticamente não há planejamento ou processos definidos antes e nem durante o início do desenvolvimento;
b) **Modelo *code-and-fix* –** os requisitos são definidos, mas não há planejamento dos processos antes do início do desenvolvimento;
c) **Modelo cascata** – há planejamento e definição dos processos para desenvolver o produto e chegar ao resultado. O fim de cada etapa leva à próxima;
d) **Modelo espiral** – há planejamento e definição de processos e, além disso, ao decorrer das etapas, *feedbacks* são obtidos e incorporados de volta aos processos.

Os dois primeiros modelos, *big-bang* e *code-and-fix*, não são recomendados e costumam ser utilizados em aplicações não profissionais ou extremamente simples, tendo em vista que, de forma geral, não há planejamento e estabelecimento de processos. Os modelos cascata e espiral, por sua vez, são empregados em desenvolvimento de produtos comerciais e com maior nível de complexidade, tendo em vista a melhor organização, previsibilidade e confiabilidade que se pode atingir por meio deles. Ambos são considerados modelos estruturados e uma versão do modelo espiral, apresentada na Figura 1, foi utilizada no desenvolvimento deste trabalho.

Figura 1 – Ciclo de desenvolvimento convencional baseado no modelo espiral



Fonte: Noergaard (2005).

Nota: Adaptada pelo autor.

2.1 Métodos Convencionais para Desenvolvimento de Sistemas Embarcados

O projeto de um sistema embarcado tem como etapa inicial, em se tratando de desenvolvimento estruturado, o estabelecimento do objetivo. Ou seja, descrever o que o produto é e o que deve ser capaz de fazer. Os elementos de projeto comumente citados na literatura para desenvolvimento de sistemas embarcados em geral, são os expostos a seguir (BALL, 2002, p. 1).

a) **Requisitos do produto** – descrição do que o produto é;
b) **Requisitos funcionais** – descrição do que o produto deve ser capaz de fazer;
c) **Arquitetura do sistema** – abstração do produto em elementos que compõem o sistema e a interface entre eles;
d) **Especificação do *hardware* –** descrição de como os *hardwares* específicos devem ser projetados;
e) **Especificação do *software* –** descrição das diretivas aplicadas ao desenvolvimento do *software*.

2.1.1 Definição dos Requisitos

A documentação dos requisitos do sistema deve ser uma descrição precisa e concisa do produto e de suas funcionalidades. Sua correta definição é crucial, principalmente para sistemas embarcados menores, pois previne, por exemplo, que posteriormente se descubra que o *hardware* selecionado não é suficiente para cumprir os objetivos do produto.

Há certa liberdade, porém, sobre o tipo de abordagem que é adotada para a documentação de requisitos. Estas podem ser extensas, cobrindo todas as interfaces, interações, e condições de erros que o sistema pode possuir ou uma lista simples contendo tudo que o produto precisa ser capaz de fazer quando finalizado. São, no entanto, necessários os seguintes itens: a finalidade do sistema, sua interface com o mundo real e sua interface com o usuário, caso exista.

Tendo em vista o alto nível de complexidade que os sistemas podem atingir e a incapacidade que os seres humanos possuem de associar e compreender grandes quantidades de informação de forma simultânea (MARWEDEL, 2018, p. 27), a utilização de hierarquias e abstrações torna-se uma ferramenta útil para facilitar a interpretação das partes componentes. O uso de diagramas permite representação de sistemas, em suas diversas camadas de abstração, de forma simples e direta. Para a fase de definição do produto e seus requisitos funcionais, além da visualização dos potenciais usos para o produto, o diagrama *Use Case* costuma ser utilizado (MARWEDEL, 2018, p. 39).

2.1.2 Definição da Arquitetura

A arquitetura de um sistema embarcado é uma abstração que inclui os elementos do sistema, os elementos que interagem com o sistema, suas propriedades e suas relações. No nível da arquitetura, o *hardware* e o *software* são representados como elementos abstraídos e esta abordagem confere um ambiente comum para convergência das decisões entre as duas áreas.

Com a representação dos elementos e suas relações é possível realizar, ainda nas etapas iniciais do projeto, análises sobre limitações de custo, confiabilidade, capacidade requerida para o *hardware* e *software etc.*

2.1.3 Especificação de *Hardware* e *Software*

Em sistemas embarcados de aplicação específica (diferente de computadores de uso geral, por exemplo), HW e SW são intrinsecamente relacionados. O SW em sistemas deste gênero costuma ser denominado *firmware,* tendo em vista sua interação direta com o HW e seu caráter de baixa mutabilidade ao longo do tempo.

A especificação do HW tem a função de documentar: os requisitos gerais para o cumprimento da proposta do sistema, como o HW especificado será aplicado para implementar as funcionalidades e a interface entre HW e SW.

A especificação do SW contém os requisitos de SW e como serão implementados os requisitos do sistema, além das possíveis interfaces com outro SW.

2.2 Desenvolvimento de Sistemas Embarcados Seguros

Sistemas embarcados estão se tornando cada vez mais complexos e mais presentes nas atividades cotidianas, fato que torna aspectos como segurança cada vez mais importantes (KLEIDERMACHER, KLEIDERMACHER, 2012, p. 4). Sistemas seguros devem, além de atingir seus requisitos funcionais, ser capazes de proteger seus recursos de possíveis ameaças (sejam elas geradas intencionalmente ou não). As medidas de segurança podem ser aplicadas tanto em nível de SW quanto em nível de HW e é comum que envolvam algum compromisso quanto à performance, ao custo do produto ou afins.

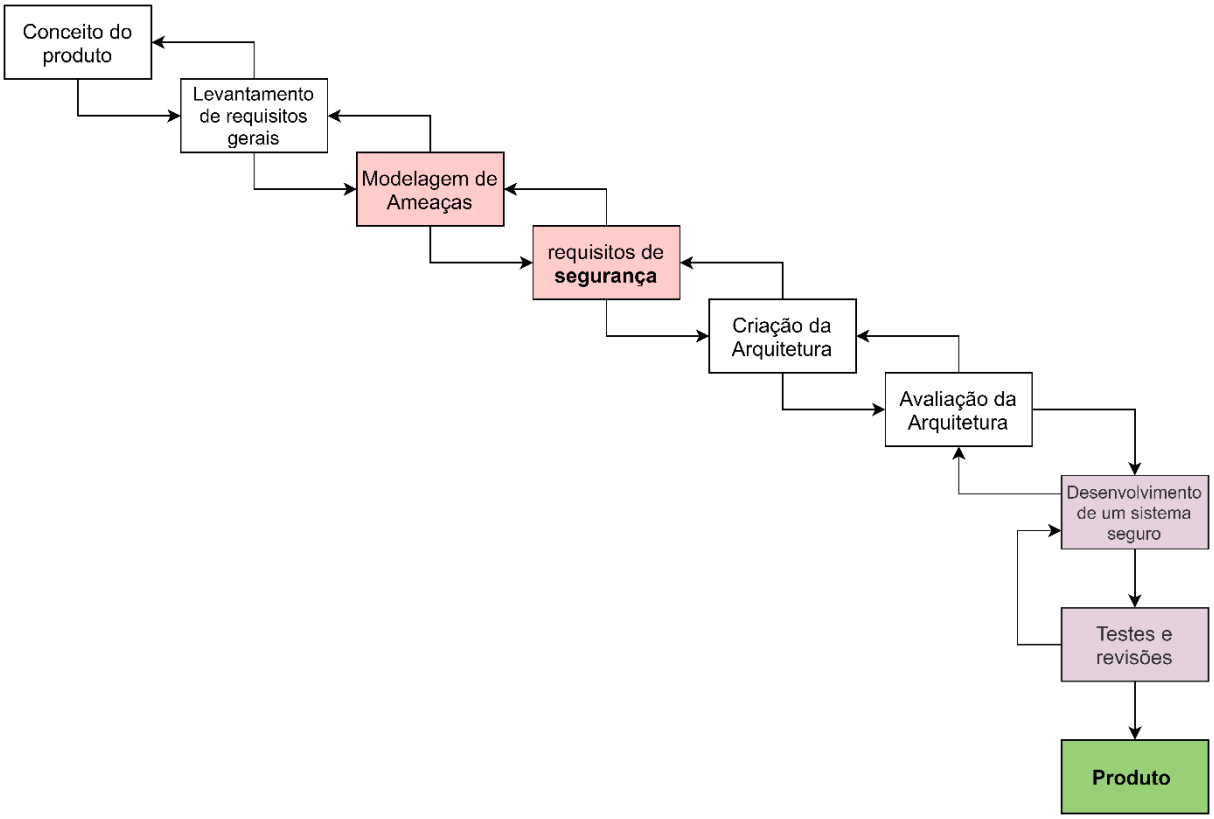
Os métodos convencionais de desenvolvimento não abordam de forma suficiente as medidas que devem ser aplicadas para sistemas em que segurança é um fator necessário. Logo, o desenvolvimento de sistemas seguros requer o uso de metodologias que diferem das convencionais.

Destacam-se duas principais abordagens possíveis para implementação de medidas de segurança em sistemas embarcados. Uma delas, popularizada principalmente devido aos avanços tecnológicos que permitem desenvolvimento rápido e simplificado de aplicações, é a “produto primeiro, segurança depois”. Este método geralmente resulta em vulnerabilidades e dívidas técnicas, tendo em vista que as alterações para aumento da segurança não foram

planejadas na concepção do sistema. Além disso, costumam não explorar o potencial das medidas de segurança por HW, haja vista sua menor mutabilidade.

Recomenda-se, porém, a segunda abordagem, que consiste em considerar os aspectos e as boas práticas de segurança desde a definição dos requisitos e da arquitetura do projeto (MENEGHELLO et al., 2019), amenizando a chance de necessidade de correções após a etapa de planejamento. De forma prática, adotam-se diretrizes de segurança em todo o processo e dar-se resposta às seguintes perguntas: "o que pode dar errado?" e "o que deve ser feito a respeito das coisas que podem dar errado?". A Figura 2 apresenta uma versão ajustada do ciclo de desenvolvimento disposto na Figura 1, onde se dá o devido enfoque às medidas de segurança.

Figura 2 – Ciclo de desenvolvimento adotado para implementação de um sistema embarcado seguro



Fonte: Noergaard (2005).

Nota: Adaptada pelo autor.

2.2.1 Diretrizes Gerais

Três diretrizes básicas de sistemas embarcados seguros se destacam para as fases iniciais de planejamento e foram aplicadas no desenvolvimento deste trabalho. São elas: a implementação

mínima, a arquitetura por componentes, e o método de privilégio mínimo (KLEIDERMACHER, KLEIDERMACHER, 2012, p. 95).

A implementação mínima é a diretriz que visa cumprir os objetivos do sistema da forma mais simples possível. Num sistema simples, as chances de existência de vulnerabilidades não consideradas ou não compreendidas é menor. No entanto, implementações mínimas exigem maior planejamento e normalmente maior tempo dedicado.

Uma arquitetura baseada em componentes (do inglês, *Component Architecture*) é formada pela divisão do sistema em componentes simples. Desta forma, é possível separar o sistema e impedir que falhas em componentes não críticos atinjam componentes críticos. Para implementação desta diretriz, deve-se estabelecer de forma criteriosa e imperativa a interface entre os componentes.

A diretriz de mínima autoridade (do inglês, *Least Privilege*) consiste em permitir que os componentes do sistema tenham apenas a autoridade que precisam ter. Por exemplo, se um componente do sistema não precisa se comunicar com outro, ou dar ordens, ele não deve ter a possibilidade. Recomenda-se que, durante o desenvolvimento do sistema todos os componentes sejam desenvolvidos inicialmente com capacidade nula e, que então, seja adicionada autoridade a eles quando se provar necessário.

2.2.2 Identificação de Ameaças

O processo de encontrar a resposta para a pergunta “o que pode dar errado?” é denominado identificação de ameaças e este procedimento faz parte da modelagem de ameaças (SHOSTACK, 2014, p. 4). Para a realização da modelagem existem diversos procedimentos indicados na literatura. Tais como a Árvore de Ataques (do inglês, *Attack Tree*) e o STRIDE (acrônimo em inglês para *Spoofing, Tampering, Repudiation, Information disclosure, Denial of Service,* e *Elevation of Privilege*) (SHOSTACK, 2014, p. 61), utilizado neste trabalho. Esses procedimentos têm o comum objetivo de estabelecer uma metodologia padronizada para listar de forma clara as ameaças, localizá-las no sistema e definir as formas de mitigá-las.

2.2.3 Medidas de Mitigação

As medidas de mitigação e suas propriedades constituem a política de segurança (KLEIDERMACHER, KLEIDERMACHER, 2012, p. 18). Seus principais componentes podem ser generalizados pelos termos apresentados a seguir:

a) **Isolamento** – estabelece os limites de interação entre partes componentes do sistema. Por exemplo, pode fazer sentido impedir que partes mais acessíveis do sistema consigam interagir com regiões críticas;
b) **Controle do fluxo de informações** – definição de regras de comunicação entre componentes do sistema. De forma prática, quais módulos poderão conversar entre si e qual deve ser a organização da comunicação;
c) **Proteção física** – diferente de aplicações de SW, sistemas embarcados devem prever proteções físicas. Estas podem ser contra ataques invasivos e não invasivos. Os invasivos podem ser, por exemplo, um acesso direto à memória *flash* do sistema para capturar informações de usuários. Já um exemplo de ataque não invasivo pode ser o monitoramento de canais de comunicação do sistema para coletar informações sigilosas;
d) **Políticas dependentes da aplicação** – englobam as demais possíveis políticas que fazem sentido para a aplicação específica.

Ao utilizar as ferramentas e diretrizes apresentadas nesta seção, torna-se possível realizar o desenvolvimento de um sistema embarcado seguro, ter ciência do que o sistema é capaz de fazer e de quais são suas limitações.

2.3 Controle Seguro de Travas Eletrônicas

Travas eletrônicas são dispositivos gerenciadores de acesso e possuem vasta aplicação na sociedade moderna. São exemplos: mecanismos de controle de acesso a ambientes (FAROOQ et al., 2014), compartilhamento de armários em agências bancárias (CHEONG; LING; TEH, 2014), controle de estacionamentos de veículos (HUANG; YANG; XIONG, 2012), dentre outros. Destaca-se que todas estas aplicações possuem características em comum. O processo de registro, autenticação e controle da trava. Sendo assim, tendo em vista que nos trabalhos analisados há omissão de detalhes suficientes sobre o desenvolvimento da estrutura das soluções e das políticas de segurança e que não é utilizado ou referenciado um método padrão para a solução do problema comum, será apresentada neste trabalho uma proposta de solução

geral para o problema de controle de travas eletrônicas, cobrindo requisitos funcionais e de segurança do sistema.

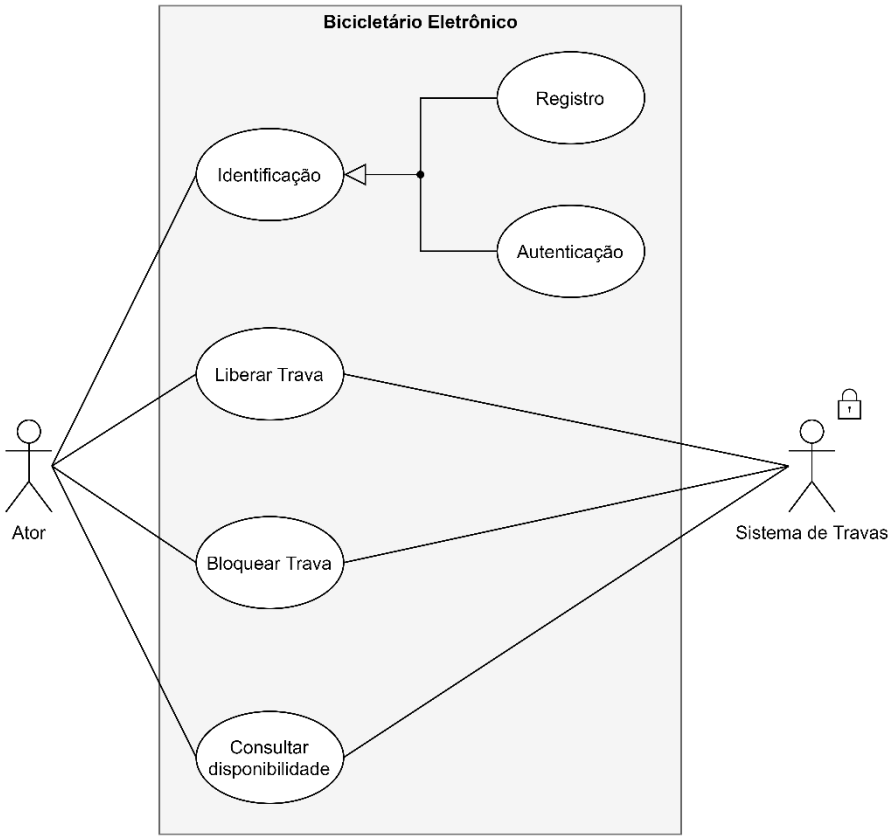
2.3.1 Requisitos Funcionais Convencionais de um Sistema de Controle de Travas

O sistema de controle de travas tem o objetivo de gerenciar acessos tendo como base requisitos estabelecidos. Sendo assim, por meio de análise convencional do sistema, podem ser destacados os requisitos gerais dispostos a seguir. Ressalta-se que aplicações específicas podem exigir novos requisitos.

a) **Identificar disponibilidade** – de forma geral, o sistema deve ser capaz de identificar se pode atuar. Um exemplo é para o caso de controle de acesso a armários. Caso todos os armários estejam ocupados, não há como permitir acesso a novos usuários;
b) **Registrar regras de atuação** – para que o sistema atue, as condições para atuação devem ser conhecidas e satisfeitas. Sendo assim, é necessária a possibilidade de registro de condições de atuação;
c) **Autenticar usuários –** conferir integridade do usuário a fim de comprovar o cumprimento de requisitos necessários para que o sistema atue;
d) **Liberar e bloquear travas** – o controlador precisa ser capaz de liberar ou bloquear as travas, visto que este é o objetivo do controle;
e) **Avisar o resultado da operação ao agente solicitante de acesso** – ao agente solicitante de acesso, seja um usuário ou outro sistema, devem ser dadas as informações necessárias para que este possa entender o resultado da operação. Por exemplo, avisá-lo que a autenticação falhou ou que foi liberado um acesso “x” sob condições “y”.

Os requisitos descritos acima são generalistas e de alguma forma, mesmo que não explicitamente, estão presentes em diversos trabalhos relacionados disponíveis na literatura. A partir deles é possível criar uma primeira versão de *Use Case,* apresentada na Figura 3. Neste primeiro diagrama foram adicionados alguns aspectos triviais relacionados à segurança que são a conferência de disponibilidade e a necessidade de autenticação.

Figura 3 – *Use Case* generalista para um sistema de controle de travas eletrônicas



Fonte: Produção do próprio autor.

2.3.2 Definição de Requisitos Mediante Análise de Segurança do Sistema

Após o estabelecimento inicial dos requisitos do sistema, devem ser analisadas as possíveis falhas de segurança que o modelo pode apresentar. A análise é realizada usando a abordagem focada em *software* – apontada por especialistas como a mais adequada para sistemas em que o *software* é o componente principal (SHOSTACK, 2014, p. 41) – em conjunto com a análise focada em recursos, tendo em vista a necessidade de cobrir as partes físicas do controle de travas eletrônicas que fogem do escopo tangível pelo *software*.

Para categorizar as ameaças será utilizado o método STRIDE, que é uma mnemônica para ameaças à segurança do sistema. Os elementos do acrônimo são explicados a seguir:

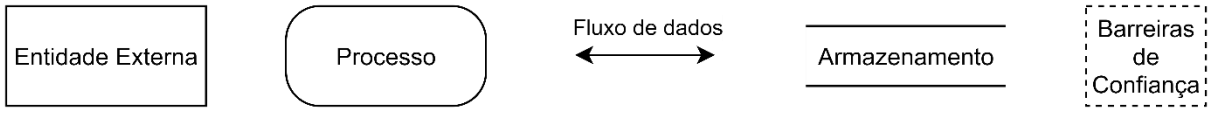
a) ***Spoofing*** – pode ser traduzido como o ato de enganar ou falsificar e consiste em fingir ser outra coisa ou pessoa;

b) ***Tampering* –** é a adulteração indevida de algo. Pode ser de pacotes de comunicação ou até mesmo de *bits* na memória;

c) ***Repudiation* –** pode ser traduzido como repúdio e significa, neste contexto, negar algum ato (tendo sido este realizado ou não);
d) ***Information Disclosure* –** se trata da exposição indevida de informações do sistema a agentes não autorizados;
e) ***Denial of Service* –** são as situações que impedem o sistema de realizar os serviços aos quais ele se propõe;
f) ***Elevation of Privilege* –** pode ser traduzido como elevação de privilégio e se define como a situação em que algum componente do sistema ou usuário passa a ser capaz de realizar operações que não deveria.

Para representar de forma clara os componentes do sistema e suas relações é utilizando um diagrama de fluxo de dados (DFD, do inglês *Data Flow Diagram*) construído com base nas diretrizes de implementação mínima, arquitetura de componentes e mínimo privilégio e utilizando metodologia espiral. O diagrama apresentado é utilizado como ferramenta para visualização do funcionamento interno do sistema a fim de facilitar e direcionar o processo de *brainstorm* para modelar as ameaças. Os elementos gerais de um DFD estão apresentados na Figura 4.

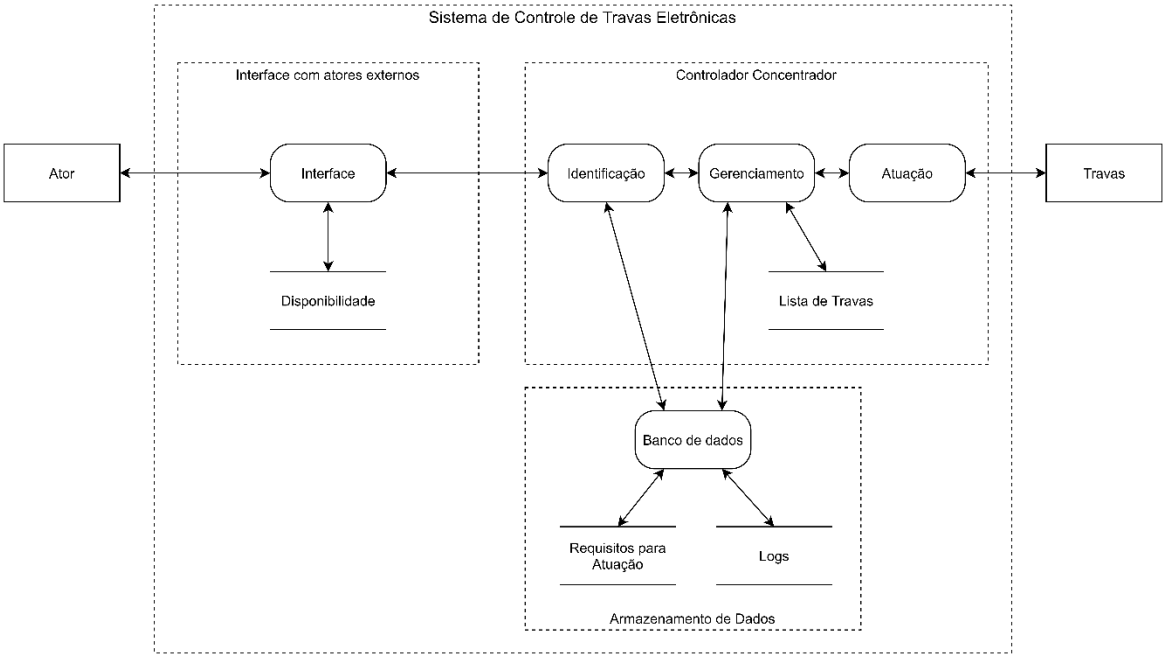
Figura 4 – Elementos principais de um diagrama de fluxo de dados



Fonte: Shostack (2014).
Nota: Adaptada pelo autor.

O diagrama de Fluxo de Dados proposto está disposto na Figura 5. Nele os usuários e as travas são componentes externos ao sistema. Internamente o sistema é dividido em três regiões de confiança: a interface com o usuário, o Controlador Concentrador (CC) e o armazenamento de dados, para os quais foi seguida uma premissa de não repetibilidade de informações a fim de diminuir a chance de inconsistências e tornar a implementação mais simples.

Figura 5 – Diagrama de fluxo de dados generalista



Nota: Produção do próprio autor.

Nota-se que a interface com atores externos, e somente ela, interage com os atores e armazena informações sobre disponibilidade do sistema. O CC armazena a lista de travas conectadas e se comunica com os demais componentes internos e com as travas. Já o banco de dados armazena os requisitos para atuação do CC e os *logs* do sistema.

A análise STRIDE desenvolvida com base no diagrama da Figura 5 está apresentada nos quadros 1 a 5. Por meio deles é possível elucidar quais tipos de ameaças cada um dos componentes pode estar exposto e quais são as possíveis ferramentas para mitigação das ameaças. A partir das possíveis ferramentas para mitigação são definidos novos requisitos funcionais que devem ser aplicados ao sistema a fim de atingir o nível de segurança desejado para a aplicação.

Quadro 1 – Análise STRIDE da Interface com os Atores Externos

Interface com Atores Externos			
Violação		**Definição da Ameaça**	**Possíveis formas de Mitigação**
S	Autenticação	ator externo fingir ser algo ou alguém que não é ou recebimento de informações de um CC falso.	estabelecimento de regras de comunicação rígidas e uso métodos de autenticação seguros
T	Integridade	alteração do dado inserido pelo ator	proteção mecânica da IHM e suas conexões
R	Responsabilização	ator alegar não ter feito alguma ação	*logs* das atividades realizadas com a identificação do agente e data
I	Confidencialidade	terceiros terem acesso a informações dadas pelos atores	proteção mecânica do CC e de suas conexões e utilização de criptografia em transações de dados confidenciais
D	Disponibilidade do serviço	falhas impedirem o funcionamento da interface com o ator	criação de projeto robusto eletronicamente e mecanicamente além da utilização de UPS (do inglês, *uninterruptible power supply*)
E	Autorização	permissão para realização de ações que não deveriam ser permitidas	definição clara das permissões dos atores e comunicação com detecção de adulterações

Fonte: Produção do próprio autor.

Quadro 2 – Análise STRIDE do armazenamento de disponibilidade

Disponibilidade			
Violação		**Definição da Ameaça**	**Possíveis formas de Mitigação**
S	Autenticação	recebimento de informações de um CC falso	estabelecimento de regras de comunicação rígidas e uso de métodos de autenticação seguros
T	Integridade	alteração dos valores de disponibilidade	regras para acesso à variável e criação de barreiras de segurança referentes a interação com os atores
R	Responsabilização	alterações de disponibilidade sem forma de comprovar o agente atuante	*logs* das atividades realizadas com a identificação do agente e data
I	Não afetado	-	-
D	Disponibilidade do serviço	sistema não representar a disponibilidade real	periodicidade da atualização do valor de disponibilidade e checagem de inconsistência
E	Autorização	algum outro componente do sistema atuando sobre a disponibilidade armazenada na IHM	isolamento dos componentes e criação de regras quanto ao escopo de atuação de cada um

Fonte: Produção do próprio autor.

Quadro 3 – Análise STRIDE do controlador concentrador (CC)

Controlador Concentrador			
Violação		**Definição da Ameaça**	**Possíveis formas de Mitigação**
S	Autenticação	recebimento de dados de identificação que são interpretados como ator indevido ou de uma IHM, banco de dados ou trava falsa	estabelecimento de regras de comunicação rígidas e uso de métodos de autenticação seguros
T	Integridade	adulteração de valores internos do CC	proteções físicas, conferência de integridade na comunicação e redundância de informações críticas
R	Responsabilização	ator ou sistema não se responsabilizar por ações sobre o CC	*logs* das atividades realizadas com a identificação do agente e data
I	Confidencialidade	terceiros terem acesso às informações de atores	proteção mecânica do CC e de suas conexões e utilização de criptografia em transações de dados confidenciais
D	Disponibilidade do serviço	falha na alimentação ou integridade física	proteção mecânica e utilização de UPS
E	Autorização	outros componentes do sistema interferindo em ações que não deveriam	isolamento dos componentes e criação de regras quanto ao escopo de atuação de cada um

Fonte: Produção do próprio autor.

Quadro 4 – Análise STRIDE do armazenamento de dados

Armazenamento de Dados			
Violação		**Definição da Ameaça**	**Possíveis formas de Mitigação**
S	Autenticação	CC falso se comunicando com o banco de dados	utilização de técnicas para conferir veracidade do transmissor e do receptor
T	Integridade	alteração de valores do banco de dados	uso de redundâncias e formas de conferir integridade
R	Responsabilização	usuário ou sistema não se responsabilizar por ações no sistema	*logs* das atividades realizadas com a identificação do agente e data
I	Confidencialidade	terceiros terem acesso a informações de atores	proteção mecânica do armazenamento de dados e de suas conexões e utilização de criptografia em transações de dados confidenciais
D	Disponibilidade do serviço	falha na alimentação ou integridade física	proteção mecânica e utilização de UPS
E	Autorização	outros componentes do sistema interferindo em ações que não deveriam	isolamento dos componentes e criação de regras quanto ao escopo de atuação de cada um

Fonte: Produção do próprio autor.

Quadro 5 – Análise STRIDE das Travas Eletrônicas

Travas Eletrônicas			
Violação		**Definição da Ameaça**	**Possíveis formas de Mitigação**
S	Autenticação	atuação sobre trava realizada por CC falso	estabelecimento de regras de comunicação rígidas e uso de métodos de autenticação seguros
T	Integridade	adulteração de estado da trava	utilização de seleção de estado não binária. Por exemplo, evitar algo como, nível lógico sendo o seletor do estado da trava.
R	Responsabilização	usuário ou sistema não se responsabilizar por ações no sistema	*logs* das atividades realizadas com a identificação do agente e data
I	não se aplica	-	-
D	Disponibilidade do serviço	falha na alimentação ou integridade física	proteção mecânica e utilização de UPS
E	Autorização	outros componentes do sistema interferindo em ações que não deveriam	isolamento dos componentes e criação de regras quanto ao escopo de atuação de cada um

Fonte: Produção do próprio autor.

Por meio da análise STRIDE realizada sobre os componentes do diagrama de fluxo de dados, destacam-se os seguintes novos requisitos para funcionamento seguro (para uma solução que desconsidera detalhes específicos de aplicação).

a) Utilizar método de autenticação confiável;
b) Implementar medidas de checagem de consistência de variáveis relevantes;
c) Utilizar regras de comunicação rígidas entre os componentes do sistema (tendo como base o mínimo privilégio);
d) Realizar *log* das ações relevantes para a aplicação;
e) Definir de forma clara as permissões de cada componente do sistema;
f) Utilizar métodos de criptografia em trocas de informações confidenciais entre módulos ou que estejam expostas de forma relevante a possíveis ataques;
g) Conferir periodicamente o funcionamento das partes componentes do sistema e das travas.

Além das características levantadas pela análise STRIDE (focada em *software*), precisam ser levantadas características relacionadas ao correto funcionamento “físico” da aplicação (eletrônica e mecanicamente). Sendo assim, são adicionadas as seguintes medidas de mitigação de danos baseadas em recursos:

a) Criar proteção mecânica robusta o suficiente para atender à aplicação de forma segura;
b) Providenciar formas de atuação sobre as travas em caso de falha do sistema embarcado (quando fizer sentido para aplicação);
c) Prover confirmação sensoriada de correta atuação;
d) Utilizar equipamento para processar atuação sobre a trava no mesmo encapsulamento da trava.

A união dos requisitos definidos nesta seção com os requisitos levantados pela análise convencional do sistema resulta nos requisitos funcionais de um sistema para controle seguro de travas eletrônicas, apresentados no Quadro 6.

Quadro 6 – Requisitos gerais de um sistema para controle seguro de travas eletrônicas

Requisitos Gerais - Travas Eletrônicas	
Classificação	**Requisito**
Convencionais	Autenticar usuários
	Avisar o resultado da operação ao agente solicitante de acesso
	Identificar disponibilidade
	Liberar e bloquear travas
	Registrar regras de atuação
De segurança	Conferir periodicamente o funcionamento das partes componentes do sistema e das travas
	Definir de forma clara as permissões de cada componente do sistema
	Implementar medidas de checagem de consistência de variáveis relevantes
	Realizar *log* das ações relevantes para a aplicação
	Utilizar equipamento para processar atuação sobre a trava no mesmo encapsulamento da trava
	Utilizar método de autenticação confiável
	Utilizar métodos de criptografia em trocas de informações confidenciais
	Utilizar regras de comunicação rígidas entre os componentes do sistema

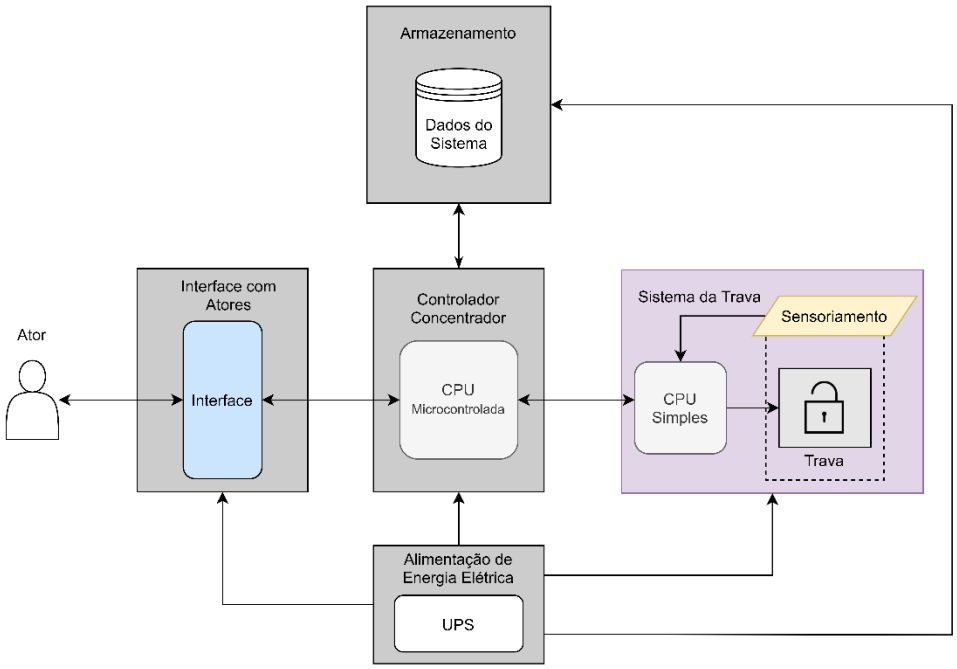
Fonte: Produção do próprio autor.

2.3.3 Sugestão de Arquitetura Generalista para um Sistema de Controle de Travas Eletrônicas

Nesta seção será apresentada uma sugestão de arquitetura de sistema embarcado para um sistema de controle de travas eletrônicas. Por se tratar de uma arquitetura generalista, modificações podem ser necessárias para que sejam atendidas especificidades das possíveis aplicações.

Tendo em vista as diretrizes de desenvolvimento de sistemas embarcados seguros e os requisitos para sua implementação sugere-se a utilização da arquitetura apresentada na Figura 6.

Figura 6 – Arquitetura generalista para controle seguro de travas eletrônicas



Fonte: Produção do próprio autor.

Semelhante ao digrama de fluxo de dados apresentado na Figura 5, a arquitetura contém uma interface com atores que isola a interação de partes críticas com o sistema, centraliza o gerenciamento e a autenticação no CC, armazena os dados num componente separado do sistema e que se comunica apenas com o CC e, por fim, isola a trava eletrônica e adiciona sensoriamento e capacidade de processamento para tornar possível comunicação criptografada e interpretação da leitura de sensores.

Ressalta-se também que grande maioria dos elementos presentes na arquitetura proposta se encontram dispostos de forma equivalente em trabalhos encontrados na literatura e já referenciados neste texto. Há diferença, porém, quanto a generalidade e adoção de parâmetros de segurança como sensoriamento e capacidade de processamento das travas e isolamento em componentes.

2.3.4 Sugestões Gerais quanto a Especificações de *Hardware*

O HW necessário para a implementação do sistema de controle de travas possui relação de relevante dependência com o tipo de aplicação intencionada. Por este motivo, nesta seção serão expostas algumas considerações gerais sobre a escolha do HW direcionado para o controle seguro de travas eletrônicas.

Recomenda-se nas fases iniciais o uso de diagramas de blocos para representar a função dos elementos do HW em maior nível de abstração. Desta forma, é possível visualizar o sistema antes de serem tomadas decisões quanto a componentes específicos.

De forma geral, um dos principais componentes do sistema será o microcontrolador e sendo assim, características como custo, número de pinos, periféricos disponíveis, capacidade memória e frequência de trabalho devem ser consideras prioritárias (BALL, 2002, p. 5). É possível também, a depender do caso, a implementação de HW com medidas nativas de segurança como impedimento de escrita na memória *flash* e opções de criptografia da memória RAM (do inglês, *Random Access Memory*) como é o caso do ESP32-S2 (ESPRESSIF, 2021) e da série *Kinetis* da NXP (NXP, 2021).

Quanto aos circuitos periféricos para sensoriamento, atuação e afins, recomenda-se o uso da literatura como forma de alcançar soluções mais confiáveis e evitar retrabalhos (MARWEDEL, 2018, p. 125). A leitura detalhada e atenciosa das folhas de dados de componentes também é extremamente recomendável (BENINGO, 2017, p. 46).

2.3.5 Sugestões Gerais quanto a Especificações de *Software*

Para a especificação de SW, apesar dos fatores dependentes da aplicação, é possível maior generalização por conta da possibilidade de utilização de APIs (do inglês, *Application Programming Interfaces*) e HALs (do inglês, *Hardware Abstraction Layers*). A utilização de ambas as ferramentas permite, como principais pontos, a reutilização de códigos e a modularização do sistema (fatores de grande relevância para segurança da aplicação).

A reutilização de código permite, além da diminuição de tempo de desenvolvimento, a possibilidade de aplicação de códigos testados em diversas situações. E este fator reduz

drasticamente a chance de introdução de *bugs* ao sistema. Não obstante, a utilização de firmware reutilizáveis necessita de uma boa definição da arquitetura do sistema e, para a maioria dos casos, maior capacidade de processamento – fator que tem perdido relevância para maioria das aplicações devido aos avanços tecnológicos (BENINGO, 2017, p. 31).

As HALs permitem que o *firmware* desenvolvido seja pouco dependente do HW. Fator que possibilita que mudanças de HW gerem menores custos e tempo de desenvolvimento, além disso, por ser reutilizada e em consequência testada em outras aplicações, ser mais confiável.

As APIs possibilitam que o SW desenvolvido seja pouco dependente dos detalhes dos componentes internos da arquitetura de SW. Ou seja, o SW pode ser desenvolvido considerando que um módulo possuirá entradas e saídas e suas configurações. O que possibilita definir as funções de interação com o SW e em paralelo desenvolver os detalhes do módulo.

Para o controlador de travas eletrônicas recomenda-se a utilização de uma HAL disponibilizada pelo fabricante do microcontrolador utilizado na aplicação ou disponível na literatura e a criação de APIs para cada um dos módulos componentes do sistema, sendo eles a interface com os atores, o CC, a memória do sistema e as travas eletrônicas.

3 PROJETO DA PROVA DE CONCEITO – BICICLETÁRIO ELETRÔNICO

Como prova de conceito da estrutura generalista proposta para a solução do problema de controle de travas eletrônicas, será realizada neste trabalho uma implementação mínima de um sistema de bicicletário eletrônico. Um bicicletário é composto por um conjunto de paraciclos (local utilizado para estacionar uma bicicleta) e se trata de um objeto muito presente no cotidiano das grandes cidades. No entanto, os bicicletários convencionais exigem que cada usuário carregue consigo uma trava mecânica – cuja confiabilidade costuma ser controversa.

Travas eletrônicas aplicadas a um bicicletário, por sua vez, têm o potencial de garantir maior segurança e comodidade aos ciclistas, visto que mitigam a necessidade do uso das travas mecânicas manuais e que possibilitam ações de compartilhamento de bicicletas. Por se tratar de um sistema com objetivo de guardar bens alheios, torna-se imprescindível a adoção de medidas que o tornem robusto quanto a segurança e confiabilidade. Ademais, vai ao encontro de medidas globais de incentivo à utilização de meios alternativos de transporte (DILL, 2009).

3.1 Definição de Requisitos e Arquitetura com Base na Solução Geral

Para a prova de conceito, realizou-se a implementação mínima do bicicletário eletrônico tendo como base as seguintes características:

a) Possui uma interface com o usuário, num módulo separado do controlador concentrador;
b) O armazenamento de dados se dá na memória interna do microcontrolador utilizado;
c) Não há conexão com a internet;
d) O sistema possui duas travas;
e) As travas estão dispostas com minimamente 30 cm entre si (distância para simular o espaçamento entre bicicletas);
f) Há uma trava por paraciclo;
g) Um único controlador concentrador deve gerenciar ambas as travas;
h) Cada usuário poderá ocupar apenas um paraciclo.

3.1.1 Adaptação dos Requisitos do Sistema

A aplicação intencionada mantém todos os requisitos generalistas apresentados no Quadro 6. No entanto, com base nas características específicas supracitadas, alguns novos requisitos podem ser definidos – para tal foram utilizados os métodos mencionados na Seção 2.2.

a) O protocolo de comunicação entre o CC e as travas deve suportar a utilização de barramentos e comunicação a distâncias superiores a 30 cm;
b) A trava deve possuir estados que representem estar ocupada, reservada ou livre. O estado reservado deve ter duração limitada, tendo em vista que, para o caso de um bicicletário, o usuário pode solicitar utilização do paraciclo, mas não realizar o uso.

Mediante os novos requisitos destacados, a lista de requisitos da aplicação está apresentada no Quadro 7 – requisitos específicos da aplicação estão destacados em cinza claro.

Quadro 7 – Requisitos do bicicletário eletrônico especificado

Requisitos Gerais – Bicicletário Eletrônico	
Classificação	**Requisito**
Convencionais	Autenticar usuários
	Avisar o resultado da operação ao agente solicitante de acesso
	Identificar disponibilidade
	Liberar e bloquear travas
	Registrar regras de atuação
	Implementar estado "reservado" para as travas eletrônicas a fim de suprir o período entre registro do usuário e efetiva utilização do paraciclo.
	Utilizar protocolo de comunicação entre CC e travas que suporte distâncias superiores a 30 cm
De segurança	Conferir periodicamente o funcionamento das partes componentes do sistema e das travas
	Definir de forma clara as permissões de cada componente do sistema
	Implementar medidas de checagem de consistência de variáveis relevantes
	Realizar *log* das ações relevantes para a aplicação
	Utilizar equipamento para processar atuação sobre a trava no mesmo encapsulamento da trava
	Utilizar método de autenticação confiável
	Utilizar métodos de criptografia em trocas de informações confidenciais
	Utilizar regras de comunicação rígidas entre os componentes do sistema

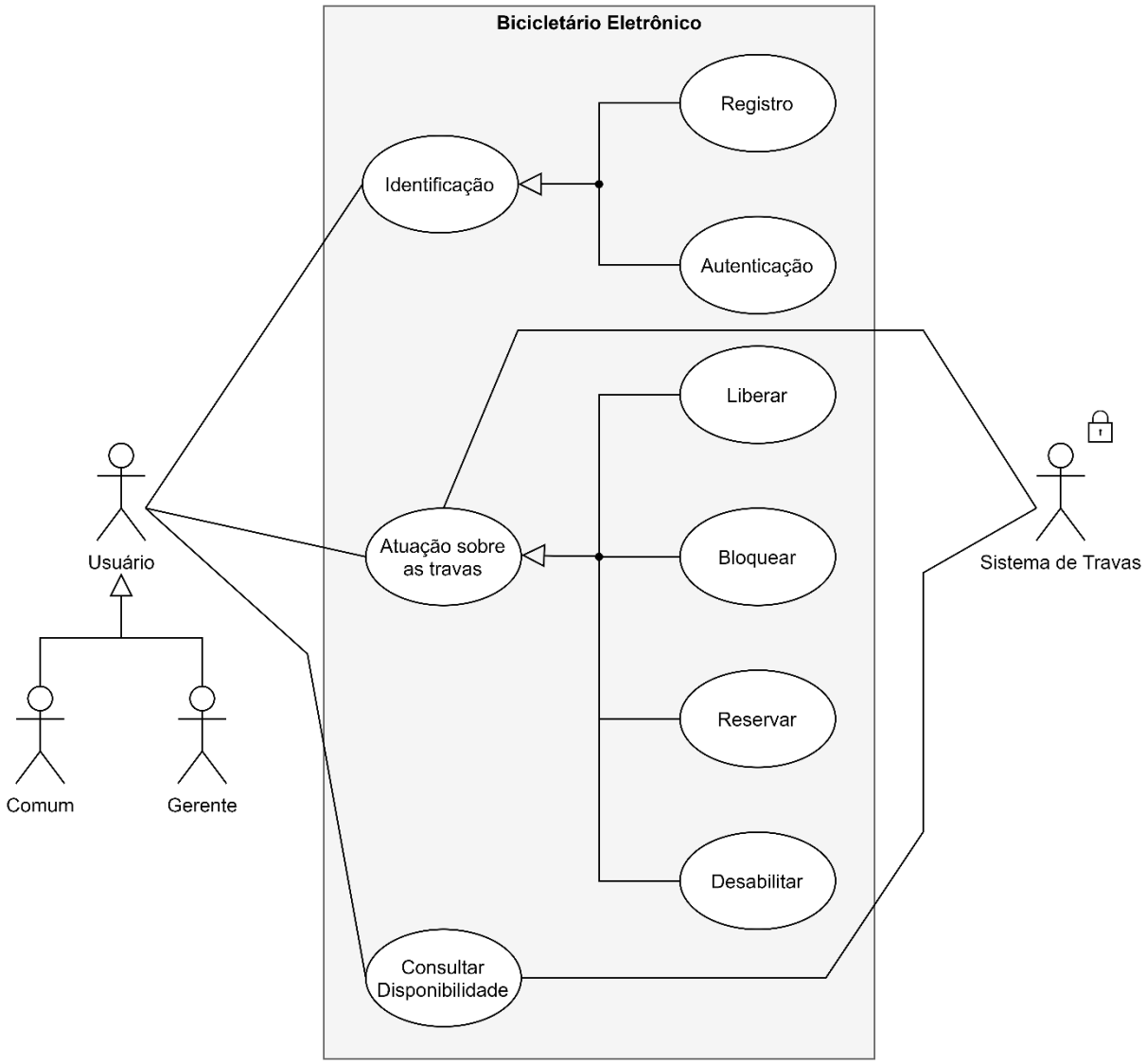
Fonte: Produção do próprio autor.

Apesar de não caracterizar um requisito, serão adotados dois tipos de usuários: comuns e gerentes. O sistema terá as seguintes características relacionadas aos gerentes:

a) Usuários gerentes somente podem ser adicionados por outros gerentes;
b) Devem ter autenticação diferente dos usuários comuns;
c) Podem desabilitar travas e desbloquear qualquer trava;
d) Podem declarar uma trava como desocupada;
e) Devem ser capazes de acessar os *logs* do bicicletário.

Um diagrama *Use Case* que representa o bicicletário eletrônico especificado nesta seção está apresentado na Figura 7.

Figura 7 – Diagrama *Use Case* para o bicicletário eletrônico especificado

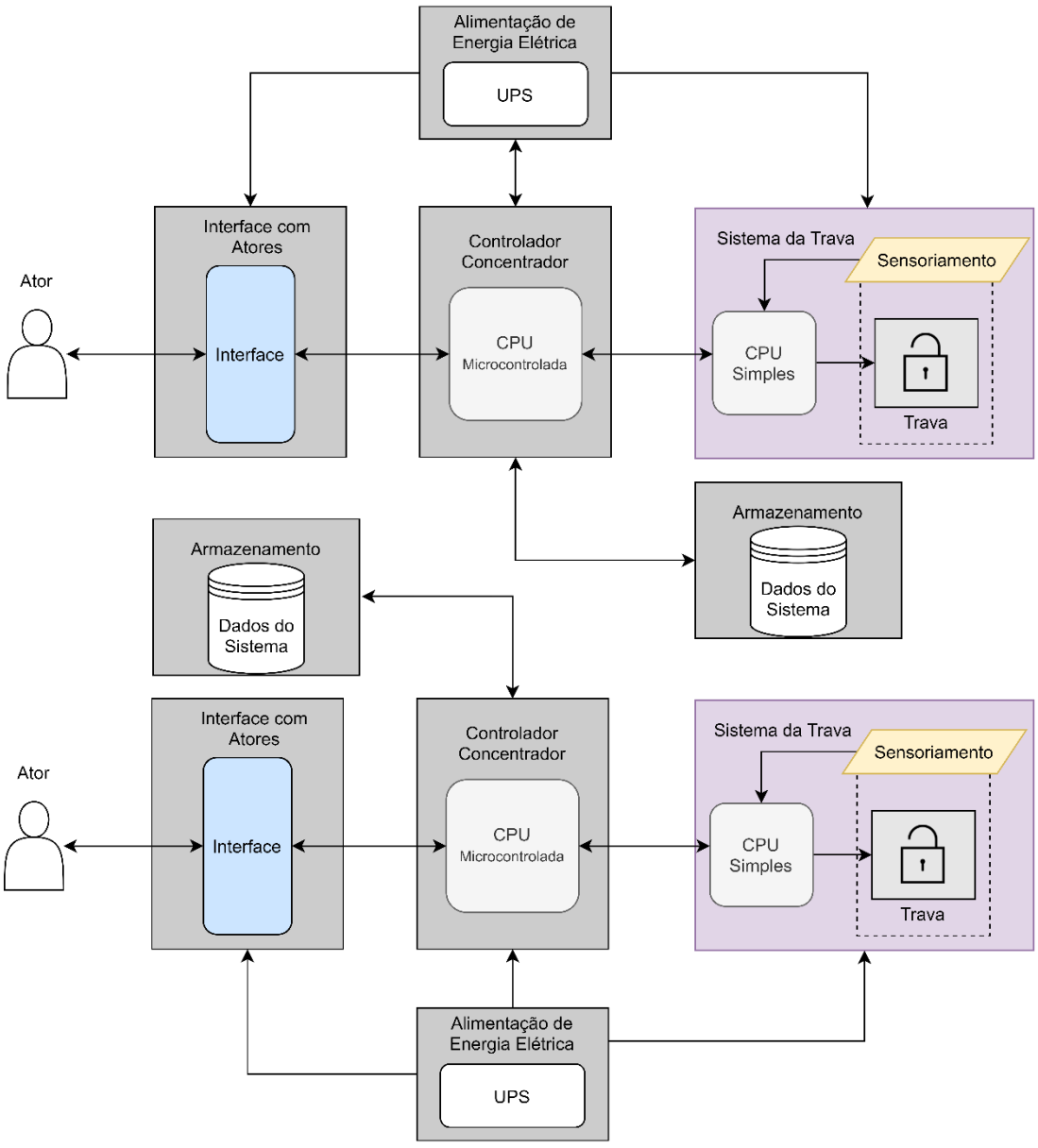


Fonte: Produção do próprio autor.

3.1.2 Aplicação da Arquitetura Proposta para o Bicicletário Eletrônico

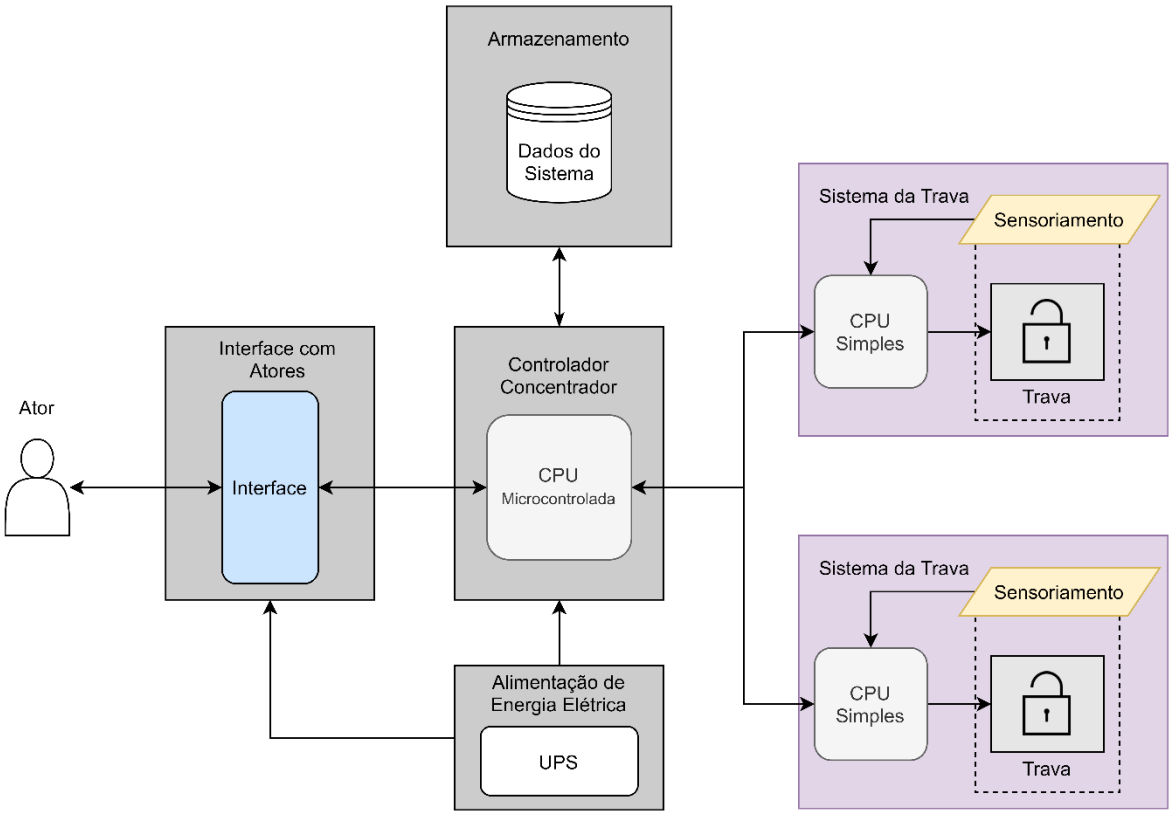
Para a implementação de um sistema com as características dadas, é possível utilizar a arquitetura generalista proposta de basicamente duas formas: distribuída ou centralizada – por simplicidade, não foram analisadas soluções híbridas. As arquiteturas distribuída e centralizada estão apresentadas na Figura 8 e na Figura 9, respectivamente.

Figura 8 – Arquitetura generalista distribuída para controle de duas travas eletrônicas



Fonte: Produção do próprio autor.

Figura 9 – Arquitetura generalista centralizada para controle de duas travas

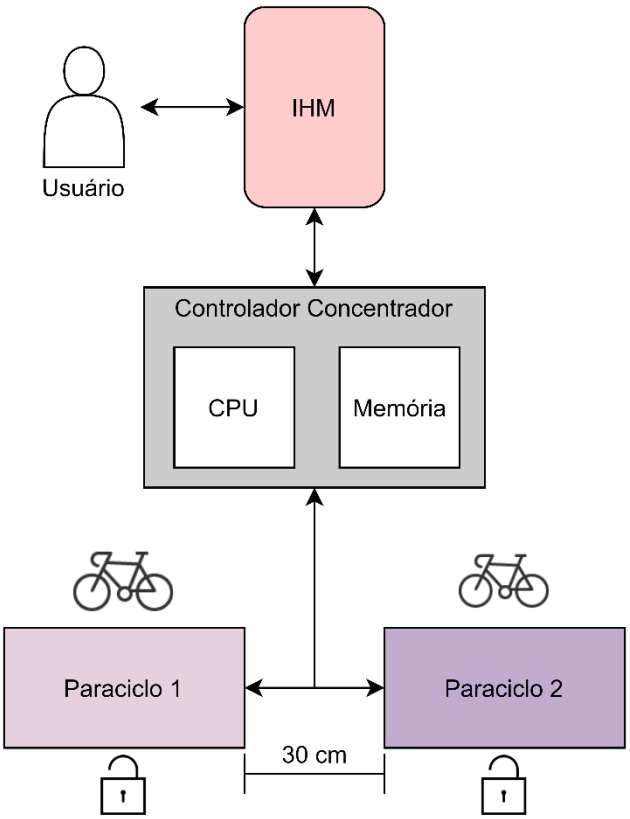


Fonte: Produção do próprio autor.

A arquitetura distribuída, apesar de exigir maior quantidade de componentes, permite o uso de microcontroladores com menor desempenho e maior simplicidade quanto ao desenvolvimento do SW. A centralizada, por sua vez, possibilita maior modularidade e simplicidade do ponto de vista do usuário. Para a prova de conceito, como foi especificado, será utilizada a arquitetura centralizada apresentada na Figura 9.

A Figura 10 apresenta graficamente a disposição dos componentes do bicicletário eletrônico mínimo especificado.

Figura 10 – Arquitetura do bicicletário eletrônico especificado



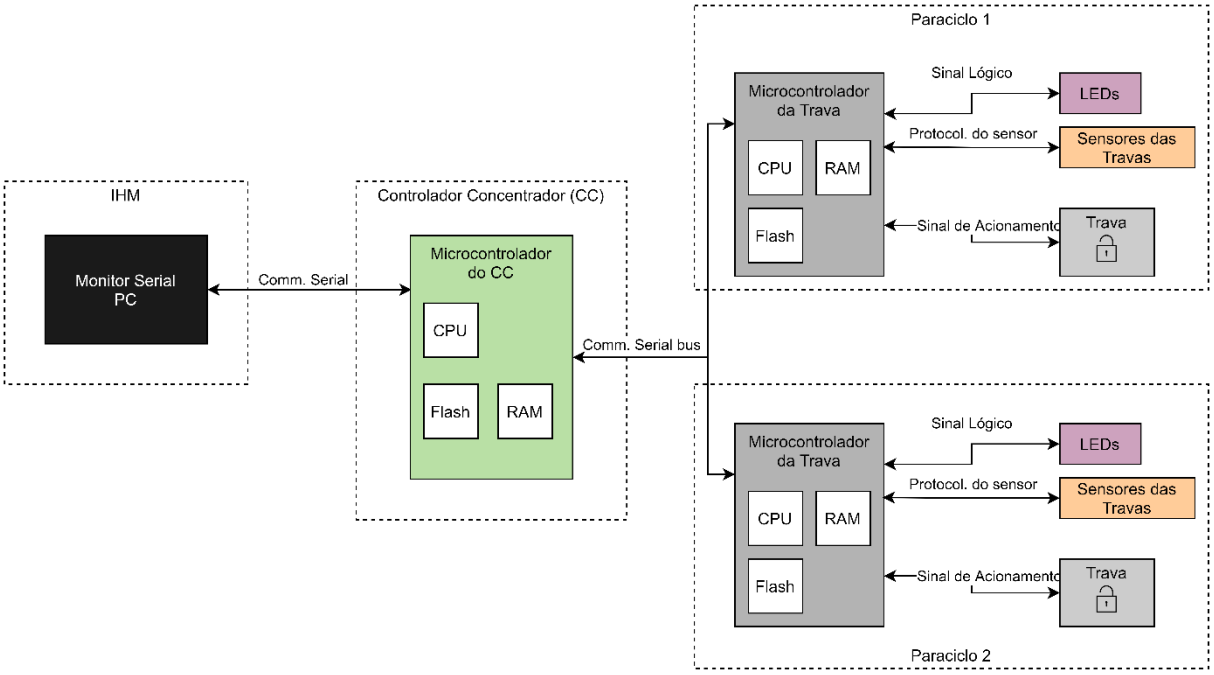
Fonte: Produção do próprio autor.

3.2 Especificação do *Hardware*

Utilizou-se como diretiva para especificação do HW da prova de conceito o custo, a facilidade de desenvolvimento (por se tratar de uma prova de conceito), a segurança e a exposição de nível técnico adequado. Vale ressaltar, porém, que uma das características desejadas para o sistema é a independência relativa em relação ao HW.

O diagrama de blocos apresentado na Figura 11 contempla uma abstração suficiente do HW implementado neste trabalho.

Figura 11 – Diagrama de especificação do *hardware* para o bicicletário eletrônico mínimo



Fonte: Produção do próprio autor.

Por simplicidade, define-se na especificação do *hardware* o uso da IHM por meio de um computador e seu monitor serial, com o papel de dispositivo de entrada e saída. Implementou-se o processamento da IHM no mesmo microcontrolador utilizado para o CC, no entanto. Por se tratar de uma prova de conceito, esta abordagem é considerada suficiente e oportuna. A comunicação com o monitor serial será realizada por meio do protocolo USB (do inglês, *Universal Serial Bus*) convertido para uma UART (do inglês, *Universal Asynchrounous Receiver/Transmiter*) no microcontrolador, por sua compatibilidade com a comunicação serial e pela presença do periférico na maioria dos microcontroladores disponíveis no mercado.

A comunicação entre o concentrador e as travas deve ser feita por meio de um protocolo que permita comunicação por meio de barramento e para distâncias maiores que 30 cm. Tendo em vista o atendimento dos requisitos, a presença do periférico na maioria dos microcontroladores atualmente em produção e facilidade de implementação, selecionou-se a utilização do periférico UART em conjunto com um protocolo de comunicação a ser especificado.

O uso do microcontrolador para as travas atende o requisito de utilização de processamento das informações (medida de segurança a fim de conferir complexidade aos comandos sobre as travas) enviadas pelo CC e permite a utilização dos sensores e LEDs (do inglês, *light- emitting*

diodes) processados no mesmo módulo. Os LEDs foram aplicados para possibilitar visualização do estado das travas por meio de cores. A definição dos microcontroladores depende mais diretamente da abordagem que será adotada pelo *software* e por este motivo será feita na seção de especificação do SW.

3.3 Especificação do *Software*

Serão implementados dois SW para o sistema. O SW do controlador concentrador e o das travas eletrônicas. Estes se diferem em relação a complexidade requerida, necessidade de periféricos e potencial de adição de funcionalidades em possíveis futuras versões.

Uma premissa adotada por simplicidade é que todos os componentes estarão previamente e corretamente registrados com seus endereços de comunicação e espécie (IHM, sistema de armazenamento e sistema de travas). Desta forma torna-se possível a abstração das complexidades que envolvem o registro de novos elementos no sistema, o que não é o foco do trabalho.

3.3.1 *Software* do Controlador Concentrador

Por generalidade, acréscimo de ferramentas de segurança e disponibilidade, escolheu-se a utilização de um sistema operacional de tempo real (RTOS do inglês, *Real Time Operating System*) para implementação do CC. O microcontrolador ESP32 está disponível em placas de desenvolvimento relativamente acessíveis, populares e de vasta documentação na literatura e que possuem os periféricos necessários para o cumprimento dos requisitos do projeto. Além disso, a fabricante (Espressif) disponibiliza um sistema operacional baseado no freeRTOS (freeRTOS, 2021), denominado ESP-IDF freeRTOS e uma HAL.

O uso de um RTOS permite que o comportamento quanto ao tempo seja previsível e adiciona ferramentas de conferência e divisão de tempo para cada tarefa do sistema (*Embedded Systems Design*). Ademais, proporciona uma solução confiável e em consequência redução tempo de desenvolvimento que seria dedicado à elaboração de um sistema de organização de tarefas.

A fim de possibilitar modularidade e, em consequência, mudanças de implementação sem necessidade de grandes alterações no SW, serão utilizadas APIs para os módulos que compõem

o sistema. Ademais, para conferir maior simplicidade, tendo em vista as complexidades que envolvem arquiteturas distribuídas, definiu-se que as interações serão do tipo *Parent/Worker*, onde há um responsável por demandar ações dos outros componentes do sistema (*Parent*), enquanto os demais respondem às demandas (*Workers*). As APIs apresentadas a seguir possuem esta premissa.

A IHM se relaciona – dentro do sistema – apenas com o CC e é mestre na relação. As possíveis interações do ponto de vista de quem interage com a IHM estão apresentadas no Quadro 8.

Quadro 8 – Definição das interações da API da IHM

Interface homem-máquina (IHM)		
Interface	**Interação**	**Definição**
CC	Resposta de atualização de disponibilidade	resposta à solicitação de atualização da disponibilidade do bicicletário
	Resposta a solicitação de sinal Keep Alive	sinal de confirmação do correto funcionamento do módulo
	Resposta a solicitação de registro	responde a solicitação de registro por parte de IHM
	Resposta a solicitação de autenticação	responde a solicitação de autenticação por parte de IHM

Fonte: Produção do próprio autor.

O Controlador Concentrador por sua vez, interage com todos os elementos internos do sistema. Definiu-se o CC como escravo na relação com a IHM e mestre em suas demais relações. Suas interfaces com esses elementos, do ponto de vista deles, estão apresentadas no Quadro 9.

Quadro 9 – Definição das interações da API do Controlador Concentrador

Controlador Concentrador (CC)		
Interface	**Interação**	**Definição**
IHM	Solicitação de atualização de estado das travas	solicitação de atualização da disponibilidade do bicicletário
	Solicitação de ocupação e registro do usuário	solicitação de registro de usuário comum ao sistema e em consequência ocupação de uma trava
	Solicitação de autenticação e liberação da trava	solicitação de autenticação de usuário comum e em consequência liberação da trava
	Solicitação de autenticação e entrada em modo gerente	solicitação de autenticação de usuário gerente, que leva o sistema ao modo administrador
	Solicitação de registro de gerente	registra um gerente (o sistema precisa estar em modo gerente)
	Solicitação de desabilitação de travas	desabilita uma trava (o sistema precisa estar em modo gerente)
	Solicitação de sinal *Keep Alive*	sinal de confirmação do correto funcionamento do módulo
Memória	Resposta a solicitação de sinal *Keep Alive*	sinal de confirmação do correto funcionamento do módulo
Travas	Resposta a solicitação de sinal *Keep Alive*	sinal de confirmação do correto funcionamento do módulo

Fonte: Produção do próprio autor.

O componente armazenamento de dados no sistema se relaciona apenas com o CC e é escravo. As possíveis interfaces do ponto de vista do CC estão apresentadas no Quadro 10.

Quadro 10 – Definição das interações da API do Armazenamento do Sistema

Armazenamento		
Interface	**Interação**	**Definição**
CC	Leitura de dado armazenado	utilizado para receber dados da memória
	Alteração ou inserção de dados armazenados	utilizado para alterar ou adicionar dados a memória
	Remoção de dados armazenados	utilizado para excluir um dado da memória

Fonte: Produção do próprio autor.

Por fim, o sistema de travas se relaciona apenas com o CC e é escravo da relação. Suas possíveis interfaces do ponto de vista do CC estão apresentadas no Quadro 11.

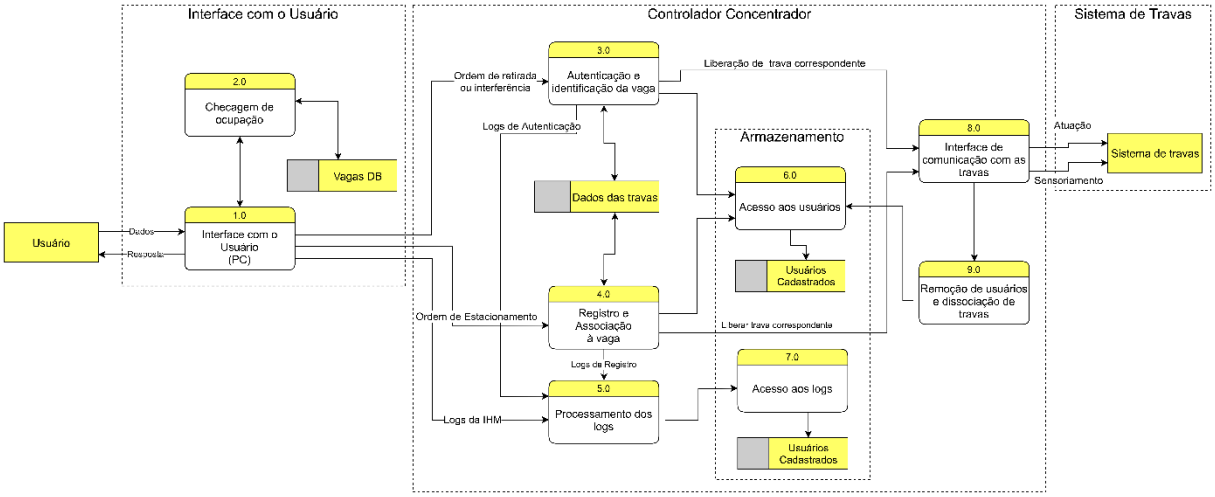
Quadro 11 – Definição das interações da API do Sistema de Travas

Sistema de Travas		
Interface	**Interação**	**Definição**
CC	Comando de atuação	utilizado para abrir ou fechar a trava
	Seleção de estado da trava	utilizado para mudar o estado entre (livre, ocupado ou reservado)
	Solicitação de leitura dos sensores	utilizado para receber os dados de leitura dos sensores das travas
	Solicitação de sinal de K*eep Alive*	sinal de confirmação do correto funcionamento do módulo

Fonte: Produção do próprio autor.

O diagrama de fluxo de dados do sistema de controle seguro do bicicletário eletrônico, baseado nas interações supracitadas, está apresentado na Figura 12.

Figura 12 – Diagrama de fluxo de dados do bicicletário eletrônico especificado



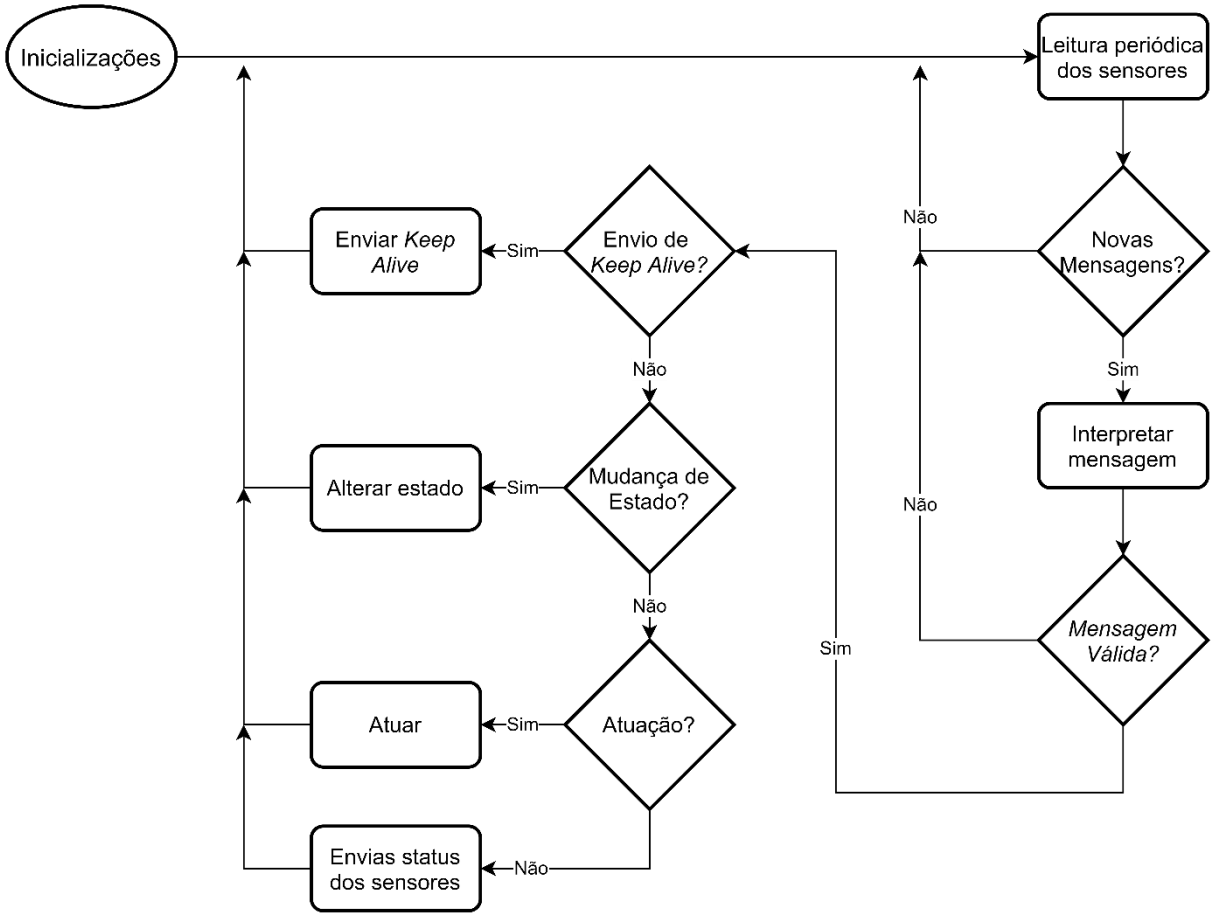
Fonte: Produção do próprio autor.

3.3.2 *Software* do Sistema de Travas

Tendo em vista a finalidade de prova de conceito e por se tratar de um sistema mais simples, desenvolveu-se o SW do sistema de travas utilizando uma arquitetura simples e trivial, denominada *super-loop* (OSHANA; KRAELING, 2013). Sendo assim, havendo liberdade para a escolha do microcontrolador, escolheu-se o mesmo utilizado para o CC.

De forma geral, o sistema de controle de travas está sempre aguardando comandos advindos do CC, de acordo com a interface apresentada no Quadro 11. Um fluxograma do *software* especificado está disposto na Figura 13.

Figura 13 – Fluxograma do *software* especificado para o Sistema de Travas



Fonte: Produção do próprio autor.

4 IMPLEMENTAÇÃO DA PROVA DE CONCEITO

Nesta seção serão explicitados os detalhes da implementação do bicicletário eletrônico mínimo quanto ao *hardware* e quanto ao *software,* tendo como base as definições e especificações realizadas nas seções anteriores.

4.1 Implementação do *Hardware*

A implementação do *hardware* da prova de conceito adotou a utilização de módulos comerciais de desenvolvimento a fim de eliminar dificuldades não relevantes para o cumprimento do papel de demonstrar as funcionalidades do trabalho realizado. Sendo assim, alguns componentes são abstraídos, tais como as travas mecânicas que são representadas como LED's e a UPS que é representada pela adição de um novo ramo para alimentação aos circuitos que compõem o sistema.

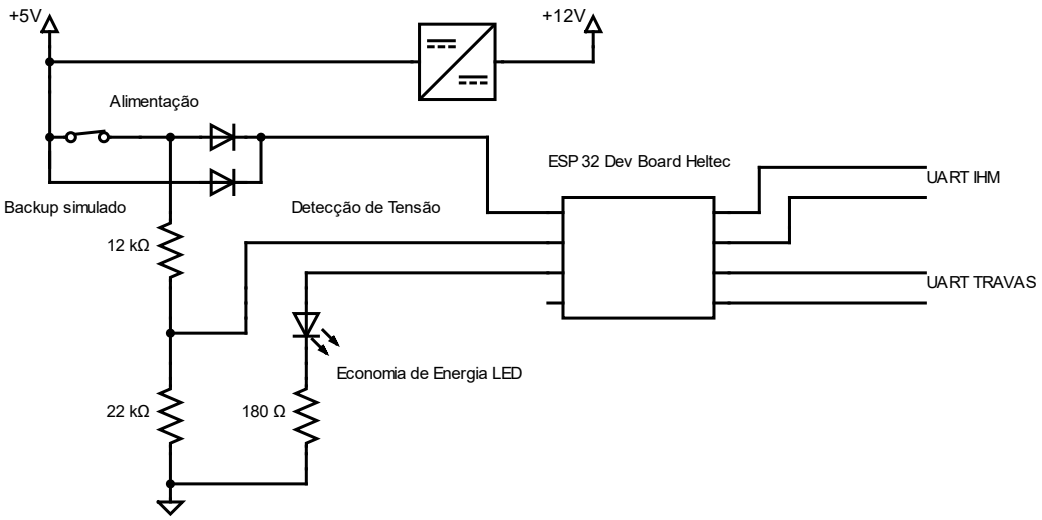
4.1.1 Interface com o Usuário

Como especificado, a interface com o usuário utiliza um PC (do inglês, *Personal Computer*) como dispositivo de entrada e saída e a comunicação entre este computador e o microcontrolador responsável por processar os sinais e se comunicar com o Controlador Concentrador é feita por meio do protocolo USB. Na placa de desenvolvimento do ESP32, o sinal USB é traduzido para UART e interpretado pelo microcontrolador. Portanto, o HW utilizado para implementação da interface com o usuário na prova de conceito está contido na placa de desenvolvimento HELTEC ESP32 WiFi LoRa V2 (HELTEC, 2021) utilizada no CC.

4.1.2 Controlador Concentrador

Assim como a IHM, boa parte das funções do Controlador Concentrador, quanto ao HW, foram englobadas na placa de desenvolvimento utilizada. O esquemático do HW está disposto na Figura 14.

Figura 14 – Esquemático do *hardware* do Controlador Concentrador



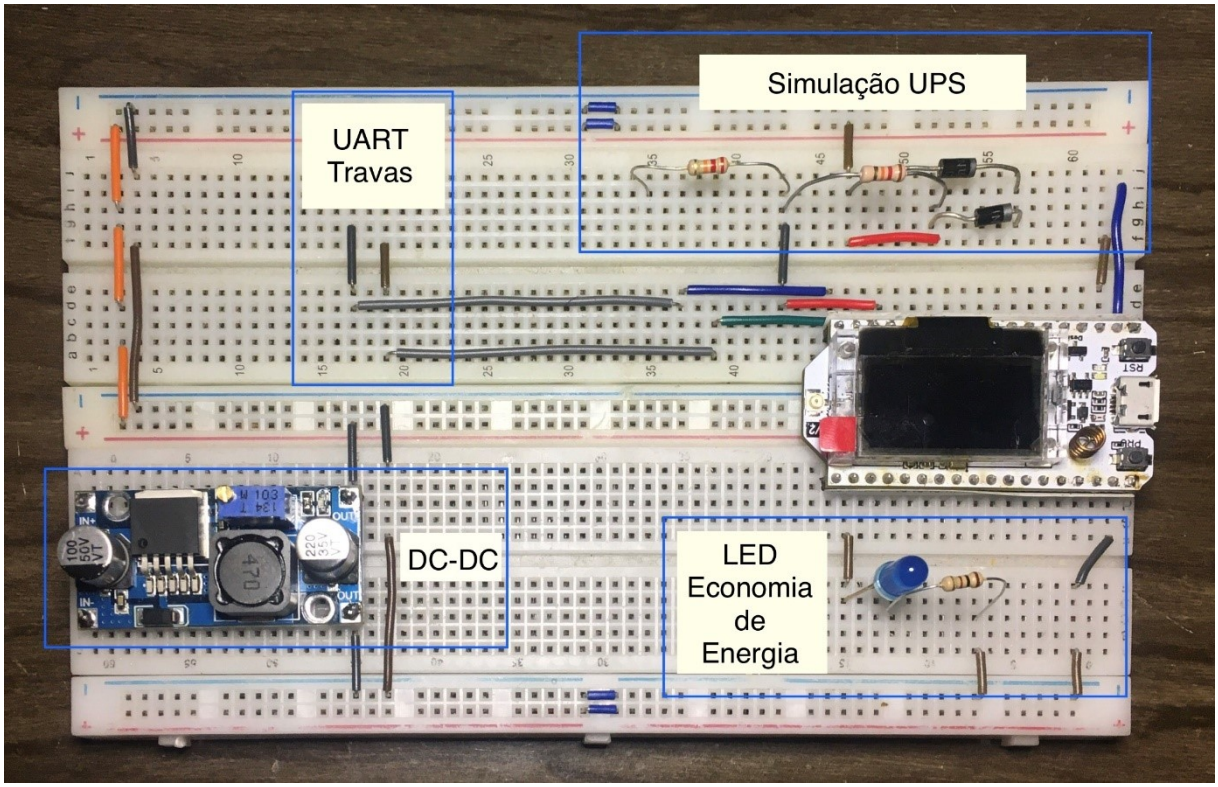
Fonte: Produção do próprio autor.

Na figura, os componentes externos à esquerda são um circuito para simulação de uma UPS, no qual há dois ramos para alimentação do microcontrolador e um dos ramos pode ser aberto por meio de uma chave. Neste ramo é realizada a medição do nível de tensão para que que seja possível ter conhecimento de alguma falha no ramo principal de alimentação e então ativar o modo de economia de energia – representado visualmente pelo LED de economia de energia.

Ademais, é representado um módulo de conversão de energia DC-DC *buck* (BARBI, 2006, p. 15) aplicado a fim de conformar o nível de tensão de alimentação aos valores adequados para o microcontrolador. Por fim, são explicitadas as saídas de conexão UART com a IHM e com as travas. Destaca-se que são utilizados periféricos UART diferentes no mesmo microcontrolador mantendo-se assim o isolamento entre a IHM e o sistema de travas.

A Figura 15 apresenta o circuito montado fisicamente e, a partir dela, é possível visualizar os elementos destacados no esquemático e a coerência das conexões especificadas.

Figura 15 – Montagem física do circuito do Controlador Concentrador



Fonte: Produção do próprio autor.

4.1.3 Sistema de Armazenamento

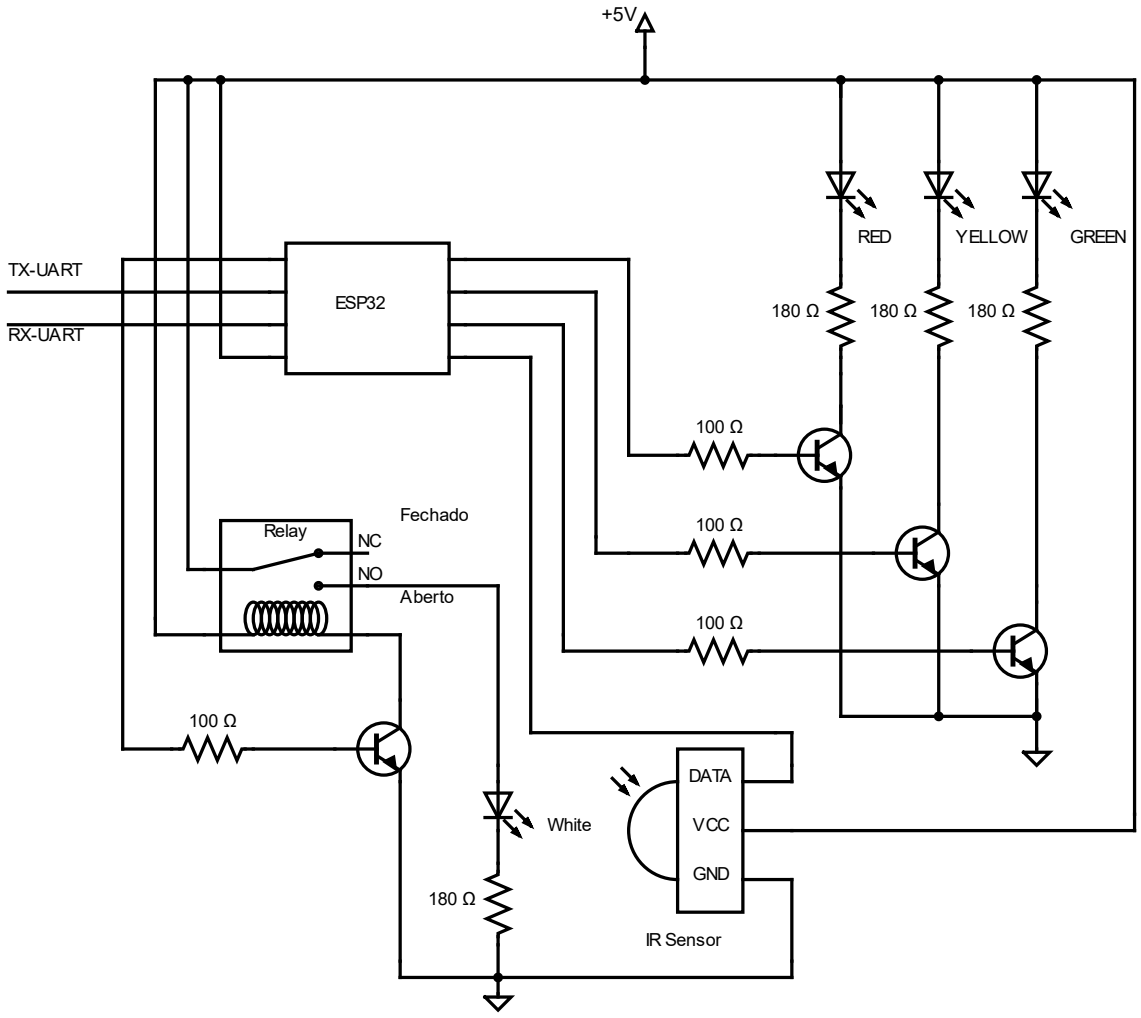
Semelhante à IHM, o sistema de armazenamento de dados da aplicação está contido na placa de desenvolvimento utilizada e compartilha a CPU com o CC. Não obstante, há implementação de isolamento por *software* e este será apresentado adiante neste texto.

4.1.4 Sistema de Travas

O sistema de controle de travas do bicicletário eletrônico mínimo, como especificado, possui LED's para sinalização de estado, sensoriamento para conferência de atuação do usuário e o acionamento das travas. Neste trabalho a trava é representada por um relé e seu estado de atuação (aberto ou fechado), por um LED.

É utilizado um sensor infravermelho convencional como maneira de simular detecção da inserção ou remoção da bicicleta no paraciclo, confirmando assim a atuação do usuário. A Figura 16 apresenta o esquemático do circuito implementado.

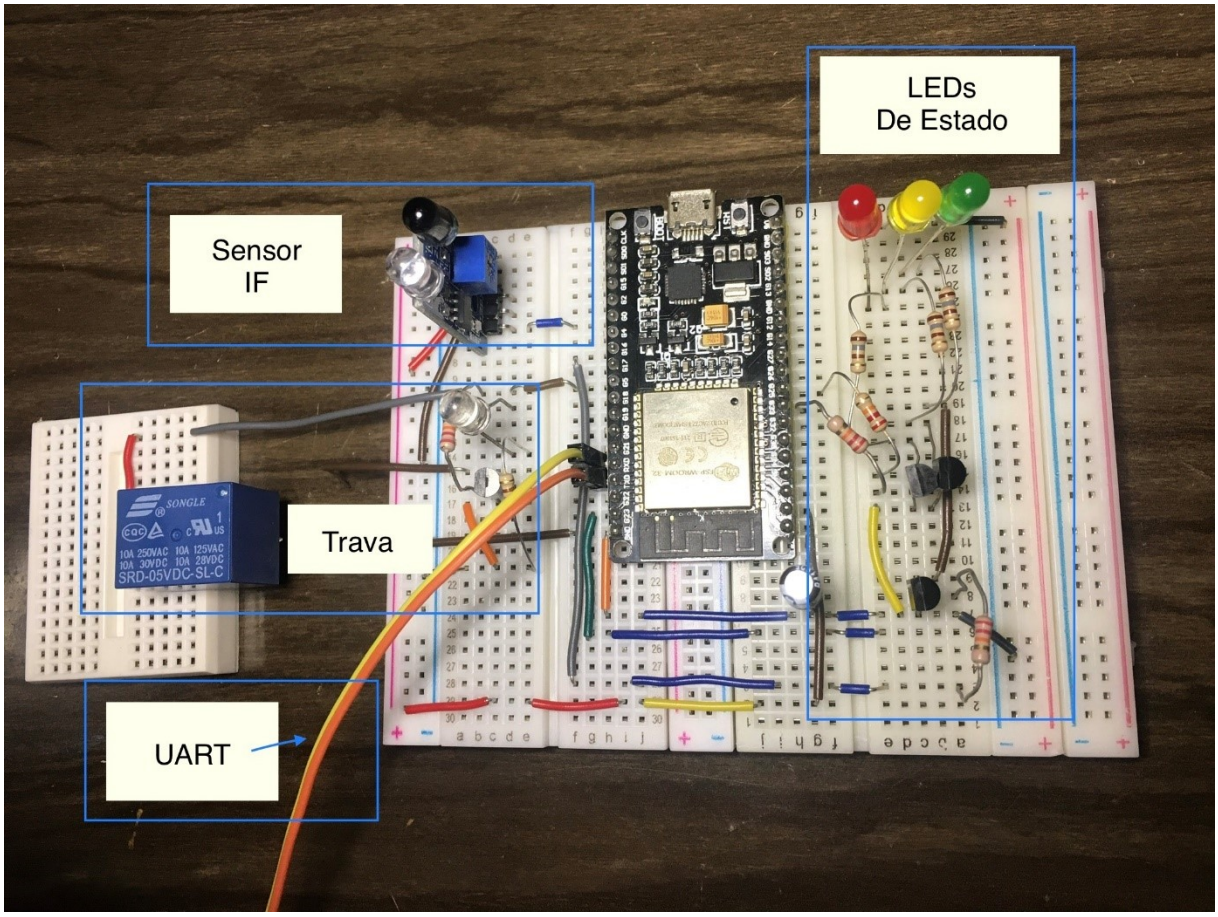
Figura 16 – Esquemático do *hardware* do sistema de travas



Fonte: Produção do próprio autor.

Os transistores implementados são de modelo BC547 e estão polarizados para trabalhar como chaves (corte e saturação). Para o caso do sistema de travas, foram implementadas fisicamente duas unidades equivalentes quanto aos componentes utilizados e baseadas no esquemático da Figura 16. É apresentada na Figura 17 a montagem física do sistema de travas.

Figura 17 – Montagem física do circuito do sistema de travas



Fonte: Produção do próprio autor.

4.2 Implementação do *Software*

Assim como a implementação do *hardware*, a implementação do *software* teve como base a especificação apresentada no Capítulo 3. Como especificado, adotou-se por simplicidade que os componentes do sistema são conhecidos entre si e que suas configurações e atributos estão corretos durante a execução do programa.

O código foi implementado utilizando a linguagem C e utilizando o ambiente de desenvolvimento PlatformIO (PLATFORMIO, 2021) em conjunto com o VSCode (VISUAL STUDIO CODE, 2021). Os conceitos de APIs e HALs foram aplicados.

Serão apresentados detalhes das decisões de implementação da IHM, do CC e do Sistema de Armazenamento. A especificação do *software* apresentada no Capítulo 3 em conjunto com as informações de implementação de *hardware,* linguagem de programação e ambiente de

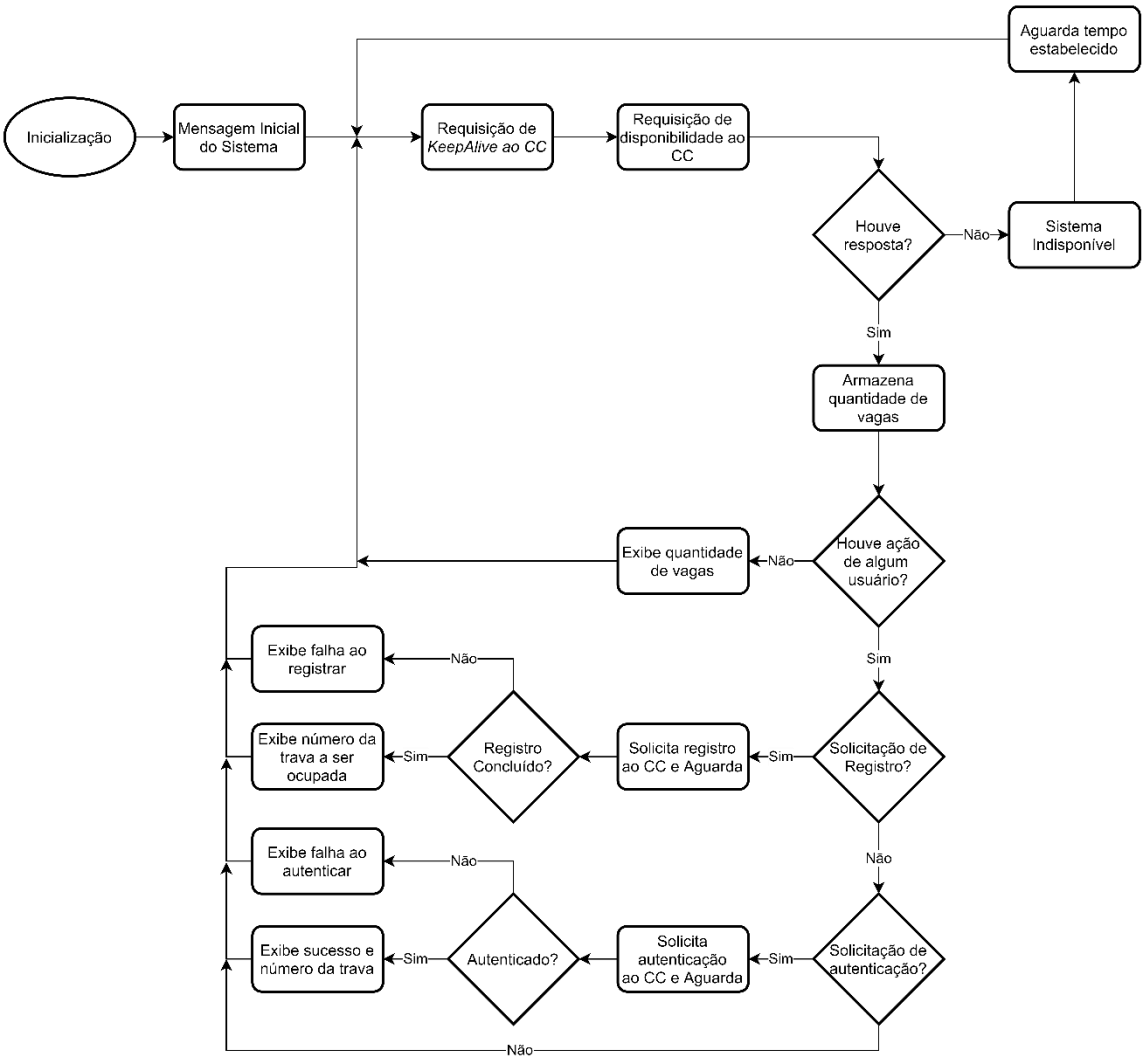
desenvolvimento utilizados, foram considerados suficientes como descrição da implementação do *software* do sistema de travas.

4.2.1 Interface com o Usuário

Apesar de ter sido utilizada a mesma CPU do CC para realização da interpretação dos dados inseridos pelo usuário por meio do PC, o isolamento entre as partes não foi violado tendo em vista a separação realizada em código. Para tal, foi criada uma tarefa exclusiva dentro do sistema operacional freeRTOS em conjunto com uma API da IHM a fim de limitar o privilégio de acesso às informações.

Foi definido um tipo abstrato de dado (TAD) para a IHM contendo seu número de identificação e tipo, considerando-se assim a possibilidade de mais de uma IHM e de tipos diferentes de implementação, como por exemplo, por leitura de impressão digital. Um fluxograma do funcionamento da IHM, com abstração de algumas características em relação ao comportamento temporal, é apresentado na Figura 18.

Figura 18 – Fluxograma de implementação da IHM



Fonte: Produção do próprio autor.

Por meio da implementação do fluxograma supracitado, cumprem-se as especificações de *software* para a IHM. Como forma de tornar o código mais simples, as funções de requisição aguardam resposta antes de retornar e o programa mudar de escopo. O Quadro 12 contém os principais métodos da IHM aplicados para implementação do fluxograma.

Quadro 12 – Principais métodos para implementação do fluxograma da IHM

Interface homem-máquina (IHM)	
Método	**Parâmetros**
uint8_t RequireKeepAlive(parâmetros)	Identificador da IHM e do CC ao qual está sendo requisitado o sinal de *KeepAlive*
uint8_t RequireVacancy(parâmetros)	Identificador da IHM e do CC ao qual está sendo requisitada a disponibilidade
void SetVacancy(parâmetro)	Disponibilidade recebida por meio da função RequireVacancy()
uint8_t RequireRegistration(parâmetro)	Identificador da IHM e do CC, dados do usuário
uint8_t RequireAuthenticaion(parâmetro)	Identificador da IHM e do CC, dados do usuário

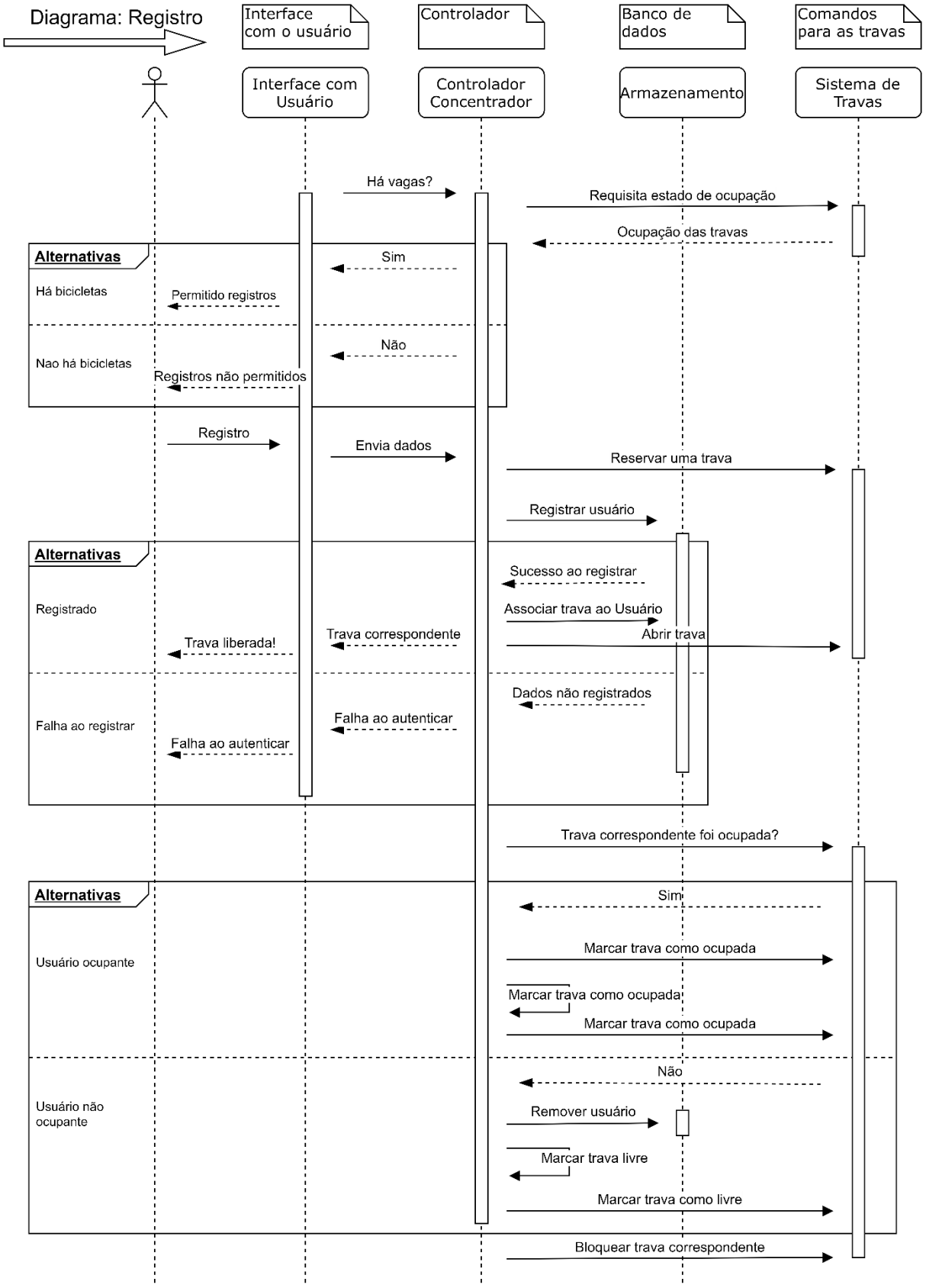
Fonte: Produção do próprio autor.

4.2.2 Controlador Concentrador

A implementação do *software* do Controlador Concentrador tem seu TAD com os parâmetros: identificação, ponteiro para o TAD de armazenamento de dados (será apresentado posteriormente no texto) e uma lista de um TAD de travas. Foi utilizada a atuação em tarefas possibilitada pelo sistema operacional com a finalidade de separar rotinas críticas quanto a segurança de rotinas não críticas.

Devido à complexidade e à quantidade de interações realizadas pelo Controlador Concentrador, considerou-se mais interessante a demonstração da implementação por meio de 2 diagramas sequenciais e de um fluxograma para representar os três principais estados de funcionamento do sistema – as ações de *log* foram omitidas visto que ocorrem em todas as interações do sistema. A Figura 19 apresenta o diagrama sequencial do sistema quando há solicitação de registro de usuário.

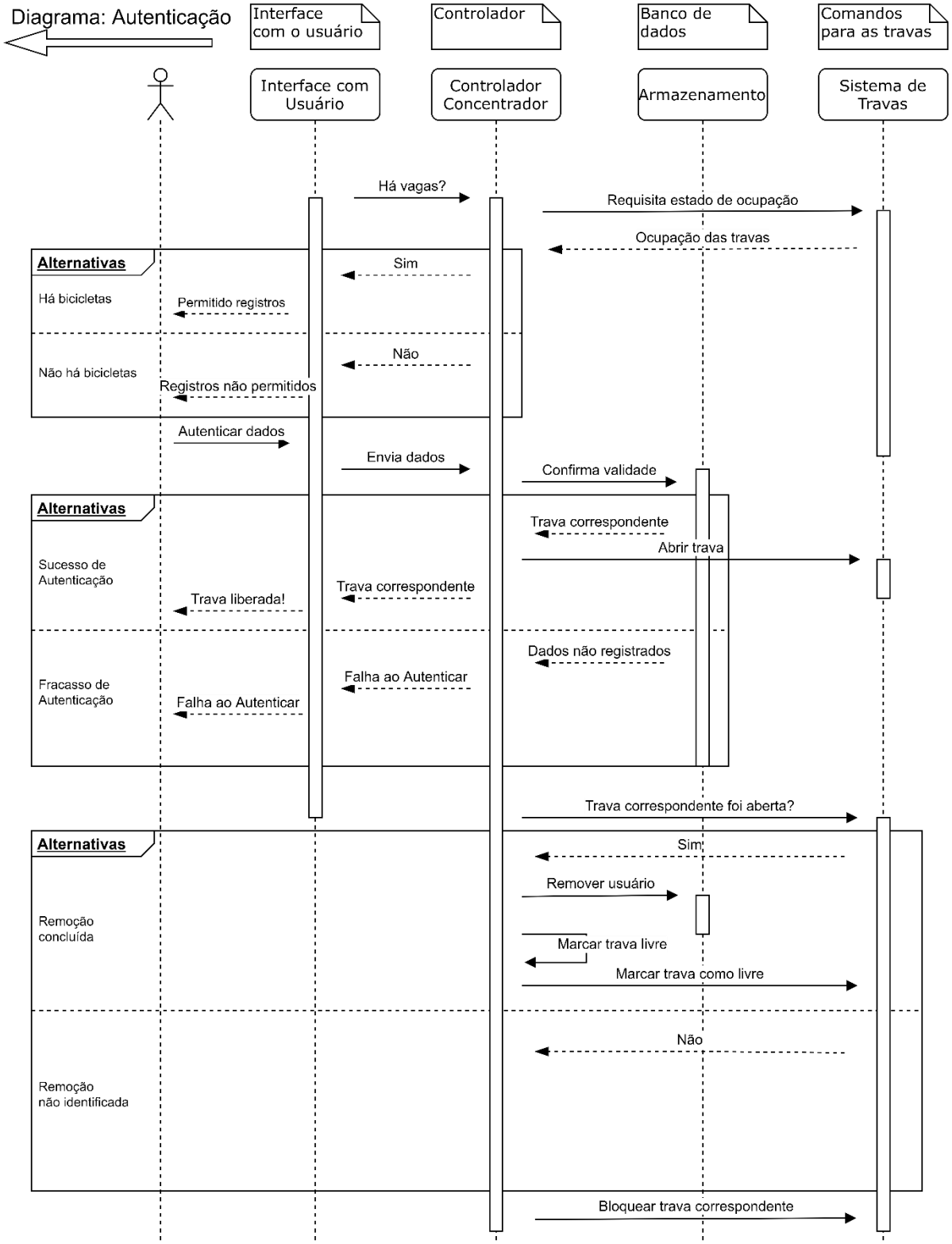
Figura 19 – Diagrama sequencial do registro do usuário



Fonte: Produção do próprio autor.

De forma semelhante ao diagrama de registro de usuário, o diagrama de autenticação está apresentado na Figura 20.

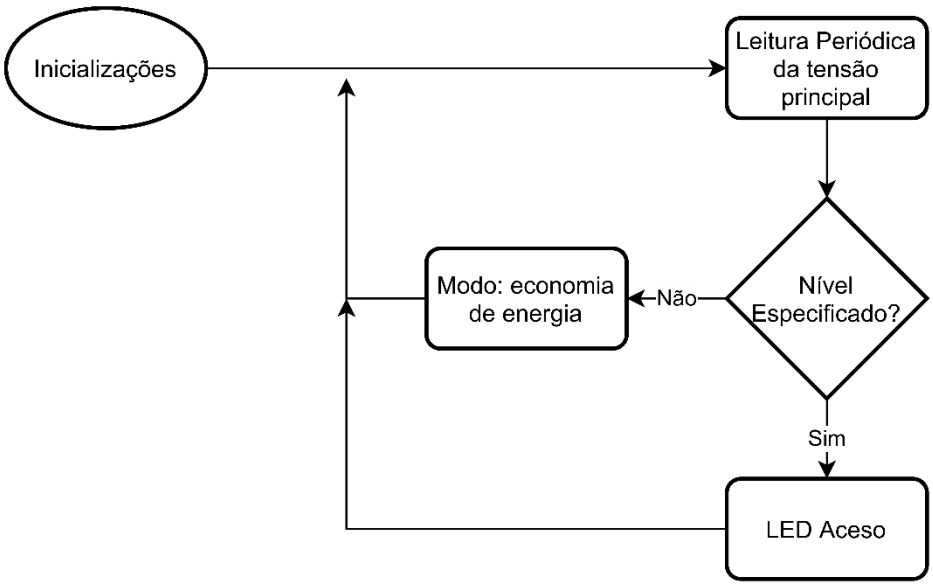
Figura 20 – Diagrama sequencial de autenticação do usuário



Fonte: Produção do próprio autor.

Por fim, apresenta-se na Figura 21 o fluxograma da ação do sistema em caso de detecção de falha na alimentação principal.

Figura 21 – Fluxograma de monitoramento da alimentação principal



Fonte: Produção do próprio autor.

4.2.3 Sistema de Armazenamento

O sistema de armazenamento utilizado possui as interfaces mencionadas anteriormente neste texto. A fim de conferir maior confiabilidade e simplicidade no desenvolvimento, levando também em consideração a diretriz de reutilização de *softwares*, utilizou-se uma biblioteca de alocação de *strings* disponibilizada na internet, denominada strmap e disponibilizada em (POKRISTENSSON, 2014).

4.3 Comunicação entre os Módulos

Para comunicação entre os módulos, realizada por meio do periférico UART, foi definido um protocolo de comunicação em ASCII (ASCIIDOC, 2021) – com a intenção de facilitar o processo de verificação de possíveis falhas. Por simplicidade, foi criado um protocolo básico apresentado nos quadros 13 a 15. É omitida dos quadros a identificação dos módulos, visto que ocorrem em todas as mensagens.

O elemento separador entre os parâmetros da mensagem é o caractere “;”, devido à sua comum utilização e à possibilidade de leitura das mensagens em “.csv”. As mensagens possuem o

formato “cabeçalho; parâmetro 1; parâmetro 2; ...; parâmetro N”. O Quadro 13 apresenta o protocolo de comunicação entre a IHM e o CC, o Quadro 14 entre o CC e o Sistema de Armazenamento e, por fim, o Quadro 15, entre o CC e o Sistema de Travas. Exemplos da aplicação do protocolo serão apresentados na seção de resultados.

Quadro 13 – Protocolo de comunicação entre a IHM e o CC

Participantes		**Permissão requerida**	**Cabeçalhos**	**Parâmetros**	
Mestre	Escravo			1	2
IHM	CC	*None*	*occupation*	resposta	
IHM	CC	*None*	*keepalive*	resposta	
IHM	CC	*None*	*register*	*user login*/ resultado	*user password*
IHM	CC	N/A	*authentication*	*user login*/ resultado	*user password*

Fonte: Produção do próprio autor.

Quadro 14 – Protocolo de comunicação entre o CC e o Sistema de Armazenamento

Participantes		**Permissão requerida**	**Cabeçalhos**	**Parametros**			
Mestre	Escravo			1	2	3	4
CC	*DataBase*	N/A	*PUT*	*user/ log*	*user login*/ gerador	*user password*/ mensagem	*user_locker*
CC	*DataBase*	N/A	*DELETE*	*user/ log*	*user login*	*user password*	
CC	*DataBase*	N/A	*GET*	*user/ log*	*user login*/ gerador	user password/ resposta	
CC	*DataBase*	N/A	*COUNT*	*user/ log*	gerador/ resposta	resposta	

Fonte: Produção do próprio autor.

Quadro 15 – Protocolo de comunicação entre o CC e o Sistema de Travas

Participantes		Permissão requerida	Cabeçalhos	Parametros	
Mestre	Escravo			1	2
CC	*Locker*	N/A	*locker_occupation*	*locker_address*	*free/ busy/ reserved/*
CC	*Locker*	N/A	*locker_sense*	*locker_address*	*sensed/ nothing*
CC	*Locker*	N/A	*locker_state*	*locker_address*	*unknown/ closed/ open*

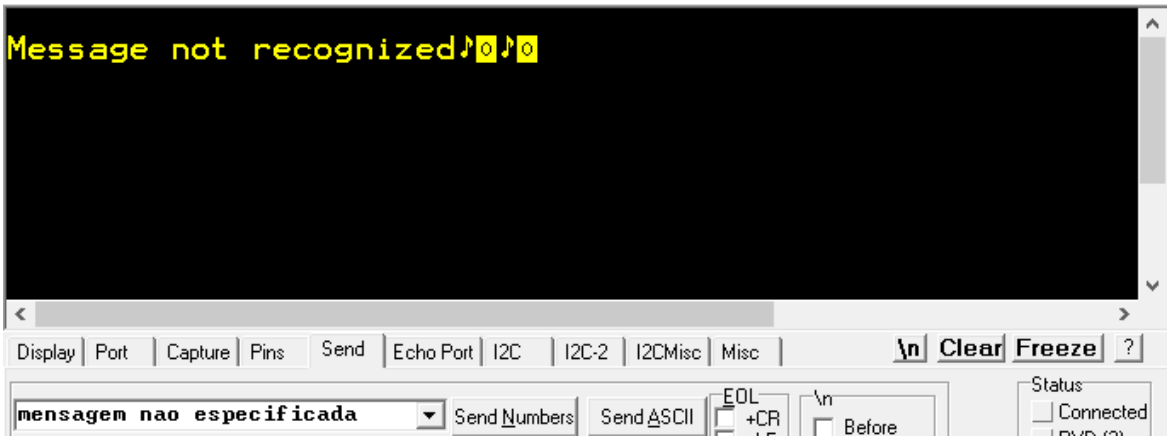
Fonte: Produção do próprio autor.

Os três quadros apresentados contêm a definição de todas as interações que previstas entre os módulos, reforçando a aplicação de implementação mínima, mínimo privilégio e isolação entre os componentes do sistema.

4.4 Resultados

Foi possível averiguar por meio dos testes realizados o funcionamento dos procedimentos e da arquitetura proposta para a implementação da prova de conceito. Uma das medidas tomadas e já descritas neste texto foi o controle do fluxo de informações por meio da definição das regras de comunicação. A Figura 22 mostra que, em caso de uma mensagem não especificada, o sistema retorna a mensagem "*message not recognized*" como alerta de não reconhecimento. O resultado apresentado é do módulo CC mas a mesma lógica é aplicada para os demais módulos.

Figura 22 – Alerta de mensagem não reconhecida no monitor serial



Fonte: Produção do próprio autor.

No código desenvolvido e apresentado na Figura 23, a função *ParseMessage()* é responsável pela identificação da presença de um *header* previamente definido. Em caso de inexistência, a mensagem recebida é então ignorada.

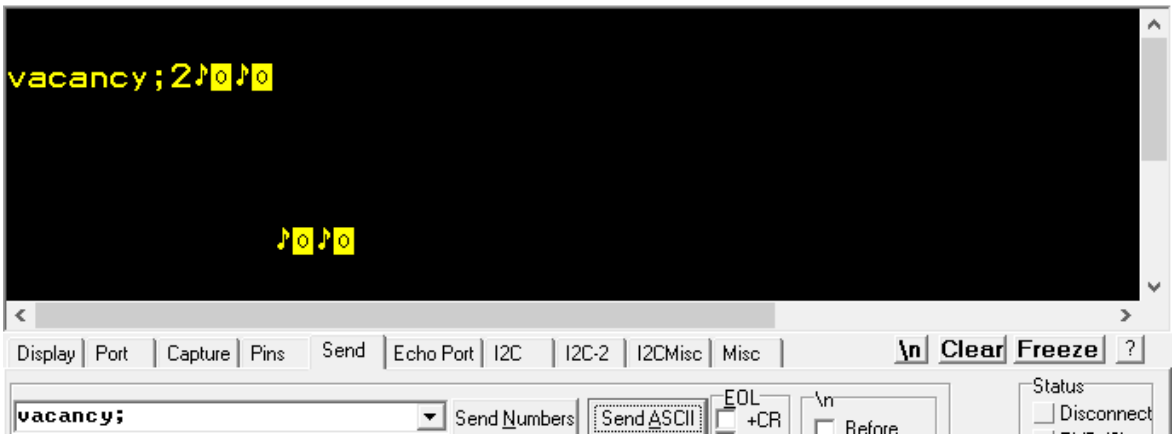
Figura 23 – Função utilizada para interpretação de mensagens no CC

```
void ParseMessage(uint8_t *buffer)
{
    char *message_buffer;
    const char delim[2] = MESSAGE_DELIM;
    char buffer_copy[BUF_SIZE];

    strcpy(buffer_copy, (const char*) buffer);
    message_buffer = strtok((char*) buffer_copy, delim);

    if(!strcmp( (const char*) message_buffer, HEADER_VACANCY))
    {
        ihm_comm_flags |= VACANCY_MASK;
    }
    else if(!strcmp( (const char*) message_buffer, HEADER_REG))
    {
        ihm_comm_flags |= IHM_ReadUserRegistrationRequest(message_buffer);
    }
    else if(!strcmp( (const char*) message_buffer, HEADER_KEEPALIVE))
    {
        ihm_comm_flags |= KEEPALIVE_MASK;
    }
    else if(!strcmp( (const char*) message_buffer, HEADER_AUTHENT))
    {
        ihm_comm_flags |= IHM_ReadUserAuthenticationRequest(message_buffer);
    }
    else
    {
        printf("\nMessage not recognized\n");
    }
}
```

Fonte: Produção do próprio autor.

Assim como a conferência básica de integridade da mensagem, a conferência de disponibilidade também foi implementada. De forma simples, o CC responde à IHM a quantidade de travas que estão disponíveis, como ilustra a Figura 24.

Figura 24 – Conferência de disponibilidade por parte de IHM utilizando o monitor serial

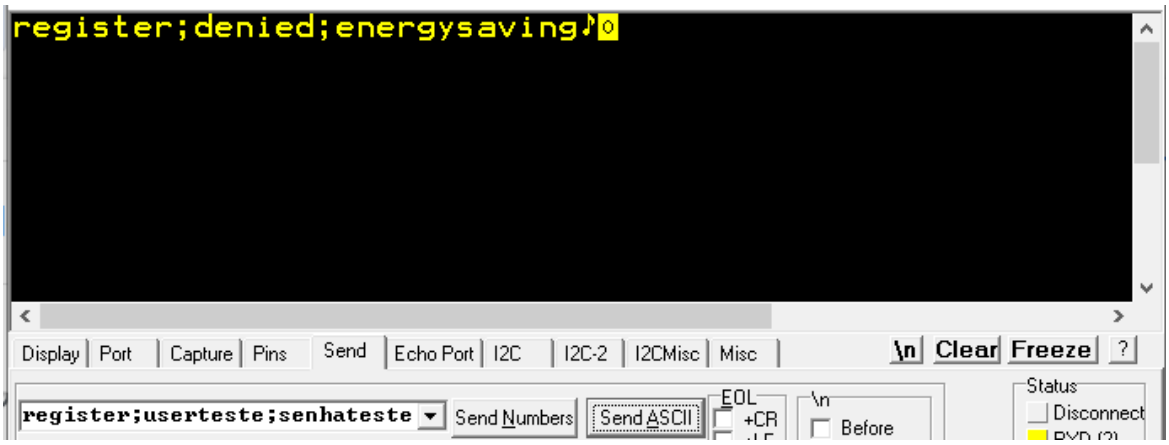


Fonte: Produção do próprio autor.

4.4.1 Monitoramento do Fornecimento de Energia

O suprimento ininterrupto de energia foi simulado em *hardware*, como especificado na Figura 16. Por meio da detecção do nível de tensão no "ramo principal" de alimentação, torna-se possível identificar se a energia elétrica do sistema está sendo fornecida pelo armazenamento de energia simulado. Caso seja detectada ausência de tensão no ramo principal, o sistema entra em economia de energia e passa a recusar novos registros (autenticações e outros processos continuam sendo possíveis). A Figura 25 apresenta a resposta do CC à solicitação de registro em estado de economia de energia. Neste estado, o LED de sinalização fica apagado.

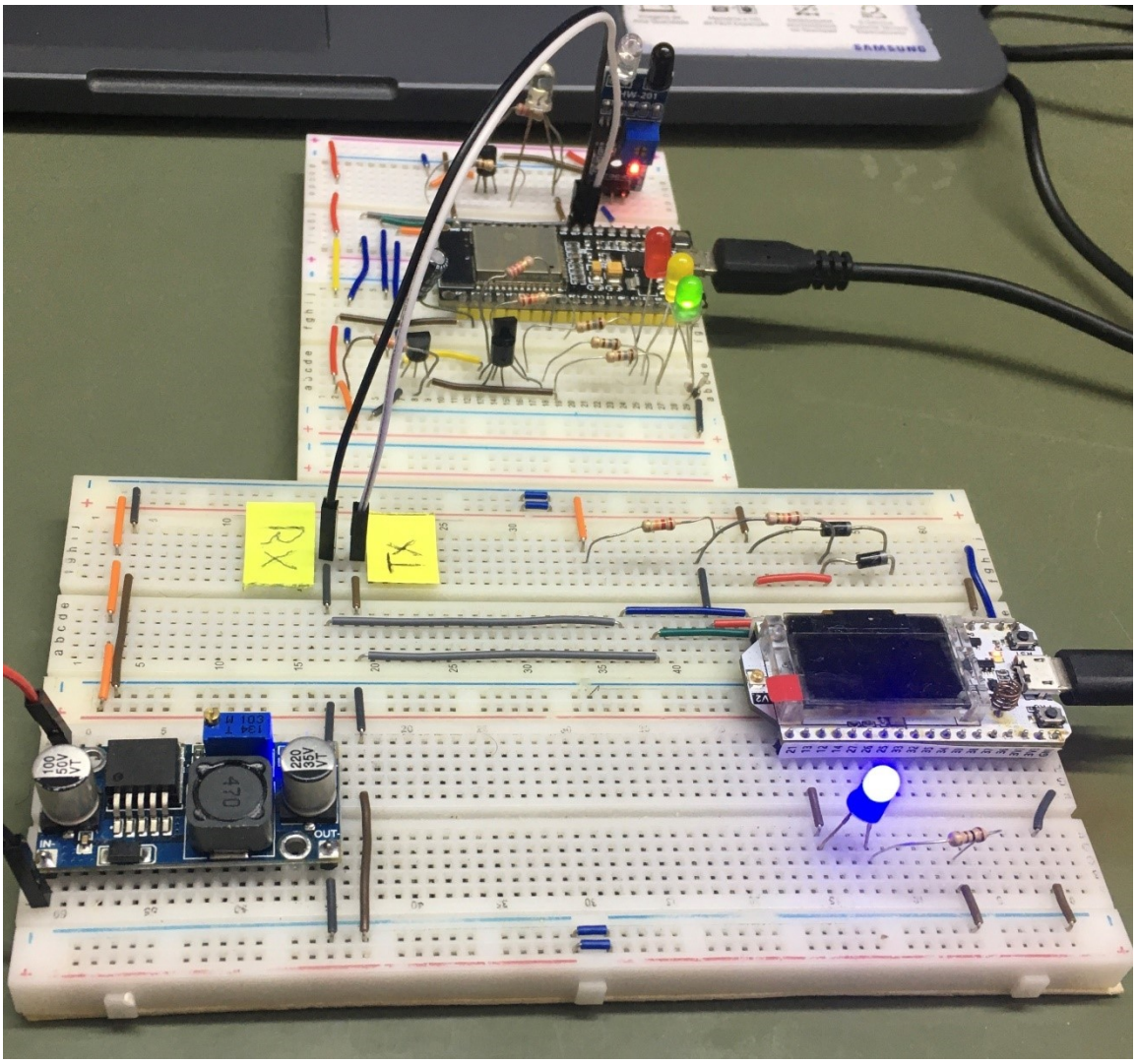
Figura 25 – Resposta por monitor serial à solicitação de registro em economia de energia



Fonte: Produção do próprio autor.

Quando o CC está sendo alimentado pelo ramo principal, ou seja, não está no modo de economia de energia, o LED sinalizador fica aceso. A Figura 26 ilustra esta situação.

Figura 26 – CC em estado normal de funcionamento

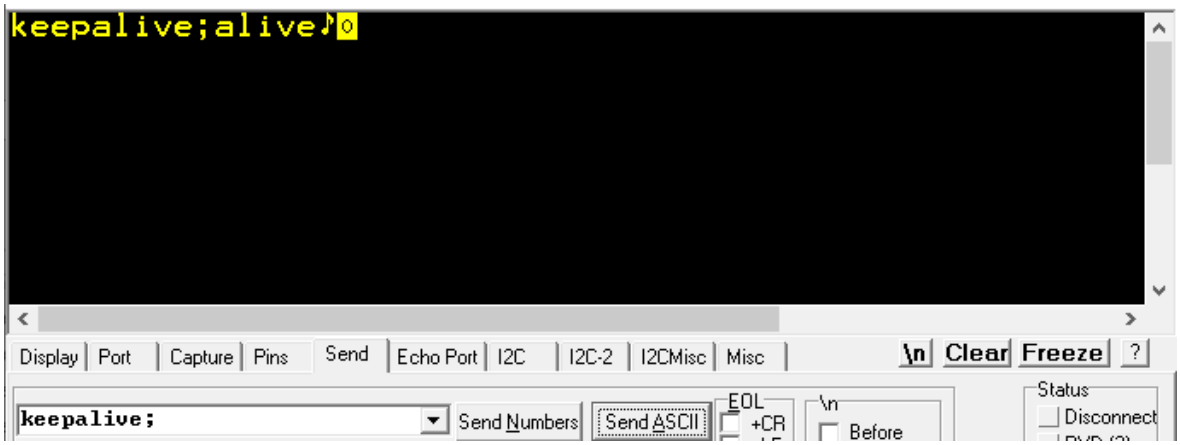


Fonte: Produção do próprio autor.

4.4.2 Conferência de Correto Funcionamento

Como especificado, para conferência de correto funcionamento é enviado aos módulos uma mensagem denominada *KeepAlive*. Ao receber esta mensagem, o módulo responde "*keepalive;alive;"* indicando assim seu correto funcionamento no que tange a interpretação das mensagens. A Figura 27 apresenta a solicitação de *KeepAlive* da IHM para o CC.

Figura 27 – Conferência de correto funcionamento do CC por monitor serial



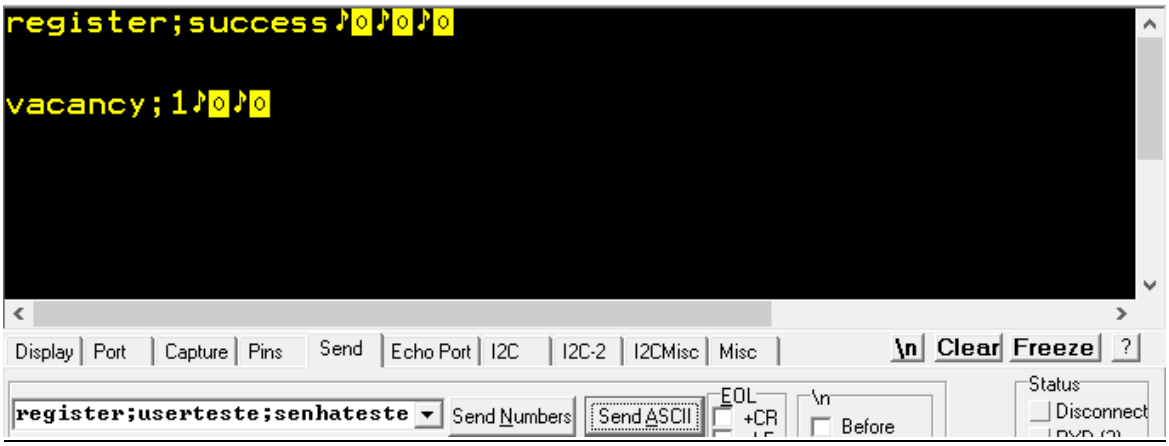
Fonte: Produção do próprio autor.

Em caso de ausência de resposta, o sistema passa a considerar que o módulo está em falha. Novas solicitações de *KeepAlive* são enviadas periodicamente.

4.4.3 Registro do Usuário

Como ilustrado na Figura 19, o registro do usuário é acompanhado da ocupação de uma trava (caso haja alguma disponível). Para tal, envia-se por meio do monitor serial de um PC a seguinte mensagem "*register*;<*login* do usuário>;<senha do usuário>" como mostra a Figura 28.

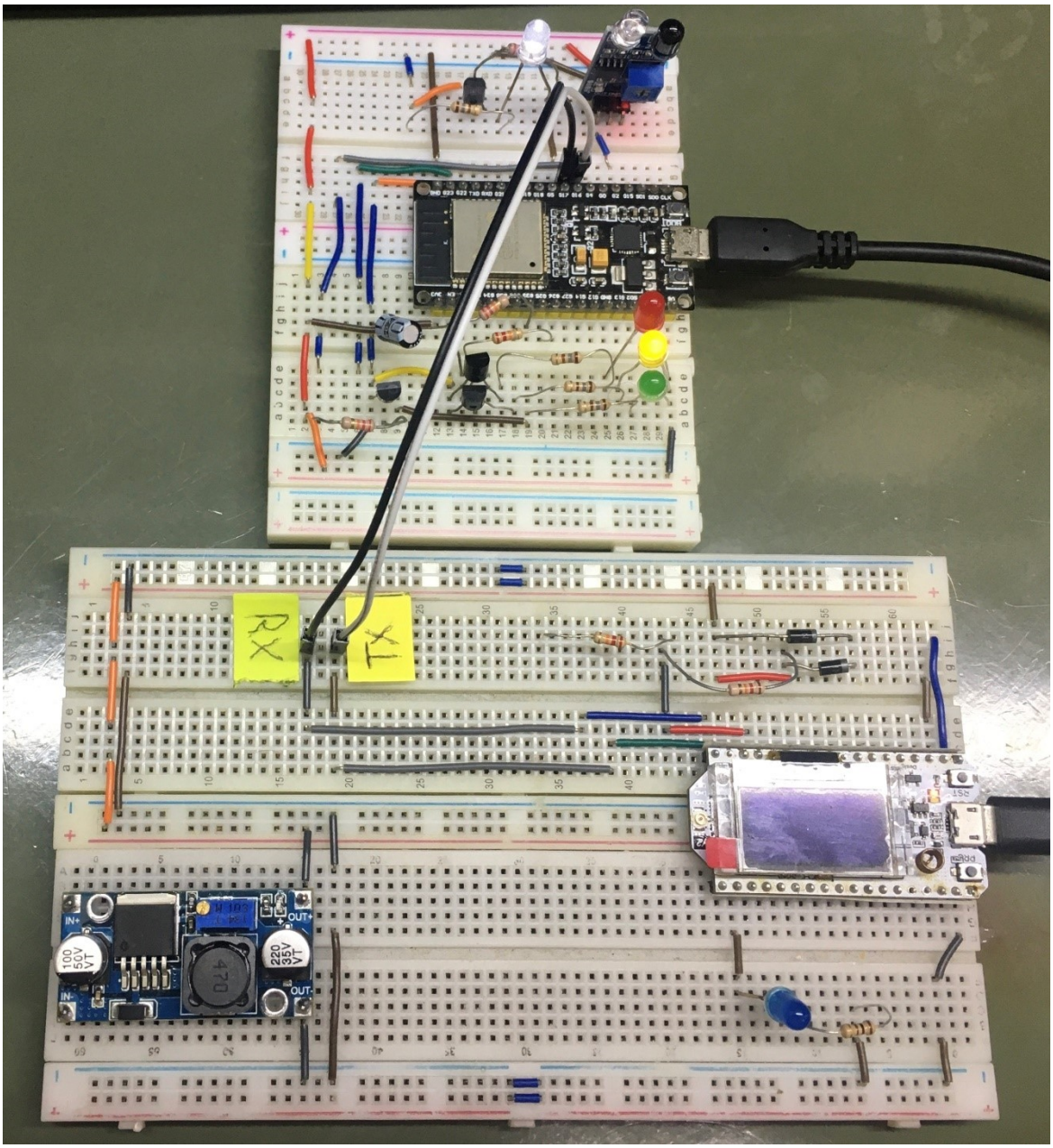
Figura 28 – Registro de usuário por monitor serial



Fonte: Produção do próprio autor.

Nota-se que, conforme estabelecido, após o recebimento da mensagem de registro, o número de vagas disponíveis é atualizado, tendo em vista que uma trava é automaticamente reservada. A Figura 29 apresenta o protótipo montado da trava eletrônica na situação de trava reservada.

Figura 29 – Trava reservada após solicitação de registro

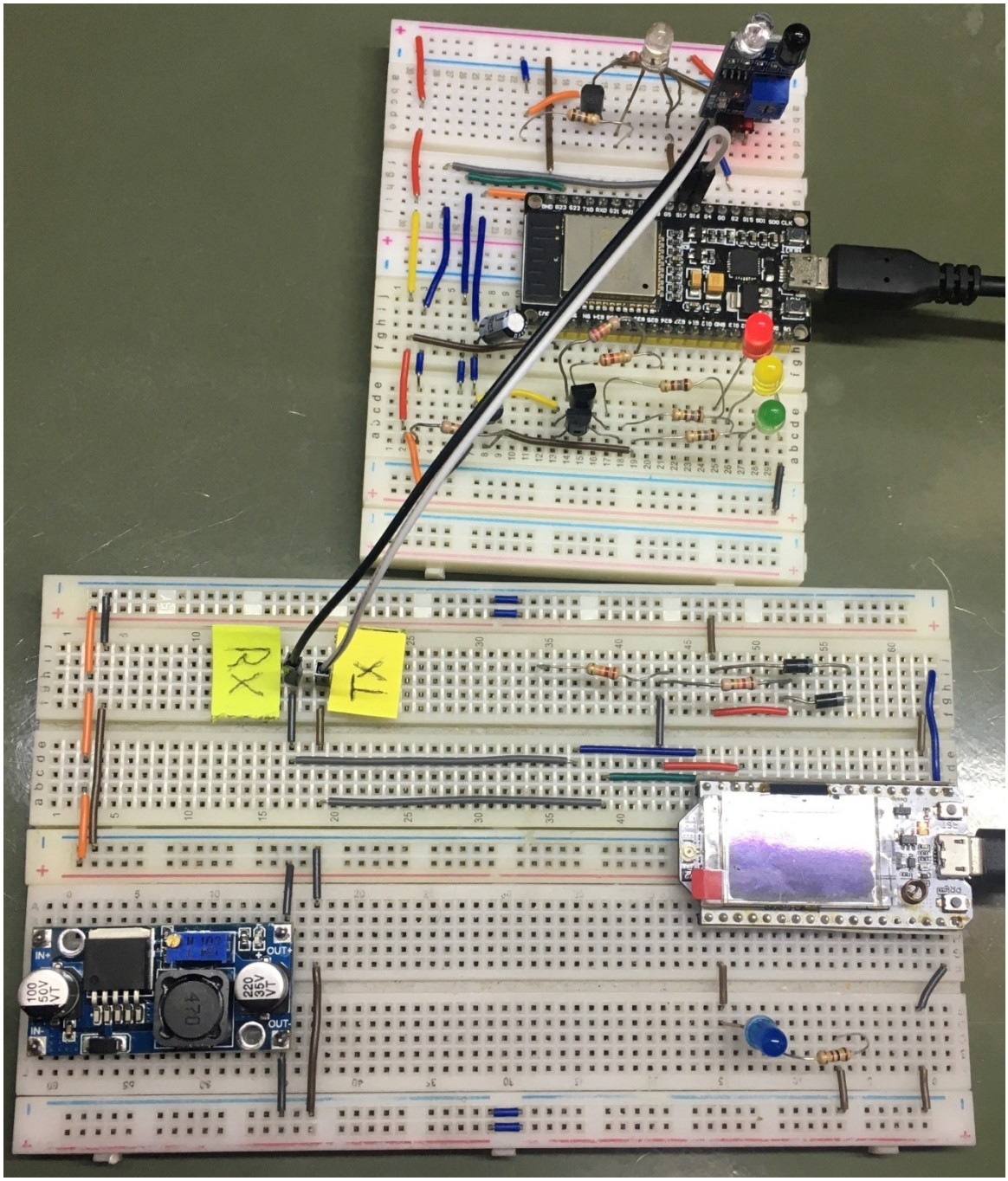


Fonte: Produção do próprio autor.

Na figura, o LED de cor amarela indica o estado de reservado e o LED branco indica que a trava está aberta. Este estado se mantém por intervalo de tempo determinado via *software* ou até que seja identificada a atuação do usuário sobre a trava. Caso não haja atuação, a trava é liberada.

Com a identificação de ocupação por parte do usuário sobre a trava, esta assume o estado de ocupado e a trava é então bloqueada, como mostrado na Figura 30. A partir de então, a trava passa a estar associada ao usuário e este usuário assume então o poder de desbloqueá-la. Desta forma, o processo de registro é concluído.

Figura 30 – Trava bloqueada após identificação de atuação do usuário

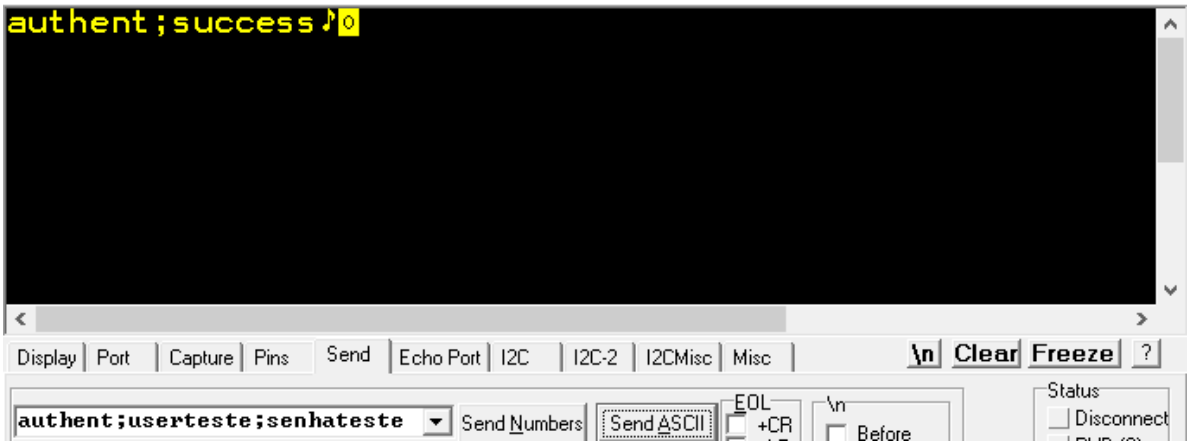


Fonte: Produção do próprio autor.

4.4.4 Autenticação do Usuário

O processo de autenticação do usuário é semelhante ao processo de registro, no entanto a mensagem enviada tem o formato "*authent*;<*login* do usuário>;<senha do usuário>" como mostrado na Figura 31. O diagrama sequencial deste processo está apresentado na Figura 20.

Figura 31 – Autenticação de usuário por monitor serial



Fonte: Produção do próprio autor.

Com a solicitação de autenticação por parte do usuário, o sistema averigua a presença dos dados fornecidos no sistema de armazenamento. O código de averiguação no banco de dados segue a API especificada no Quadro 14, um comando *GET(),* que está apresentado na Figura 32.

Figura 32 – Conferência de presença do usuário no sistema de armazenamento

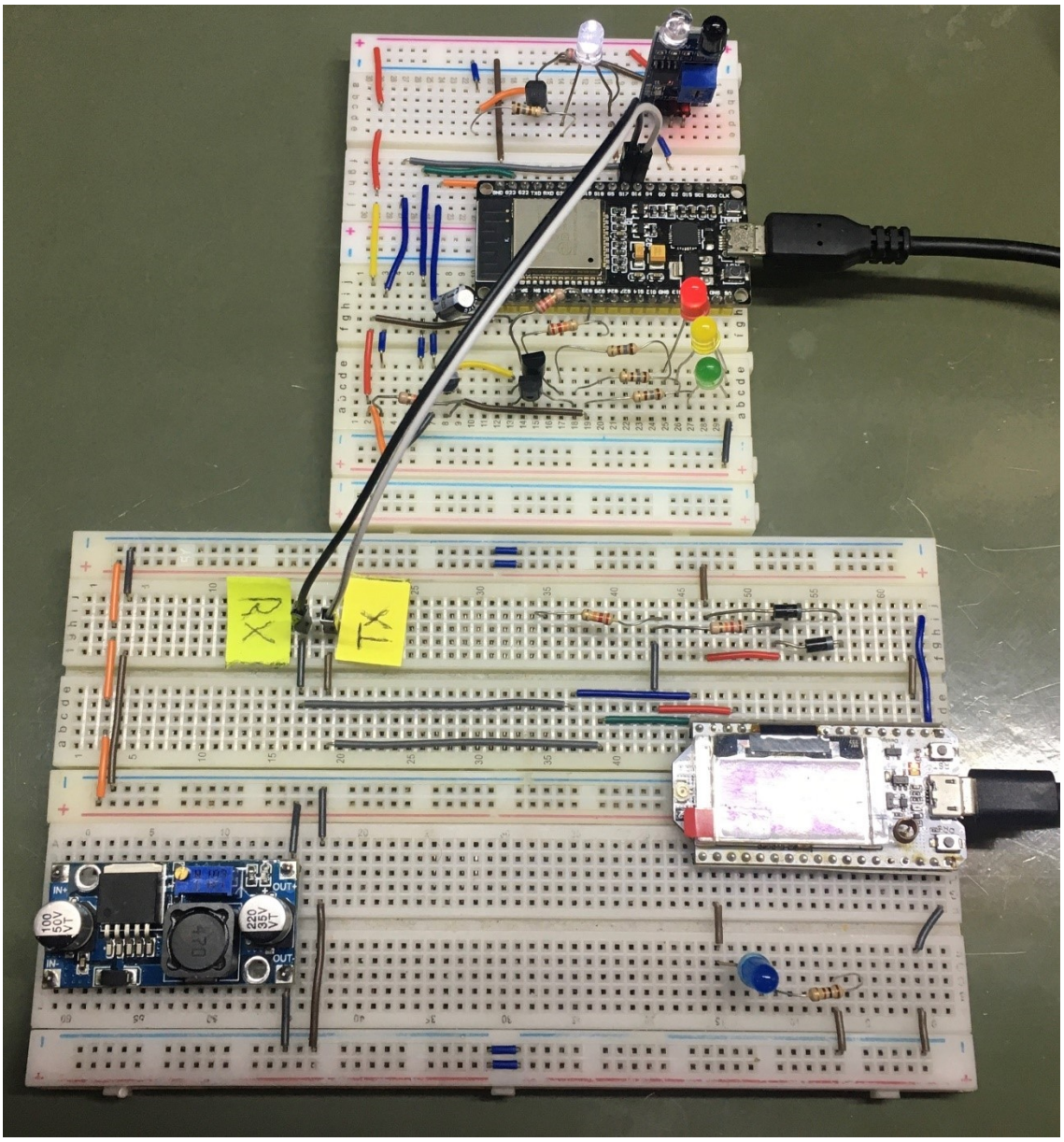
```
int CC_AuthenticateUser(CC *cc, char *login, char *password)
{
    char united[BUF_SIZE], hold_value[10];

    UniteMessageArguments(united, login, password);

    if(!DB_GET(cc->db, data_user, united, hold_value))
    {
        printf("authent;fail\n");
        return AUTHENT_CLEARMASK;
    }
```

Fonte: Produção do próprio autor.

Tendo sido realizado o processo de autenticação, a trava correspondente é aberta e inicia-se a contagem de tempo para a identificação de atuação do usuário. A Figura 33 apresenta o

protótipo sob o estado descrito, em que o LED vermelho aceso indica a trava ocupada e o LED branco aceso a trava aberta.

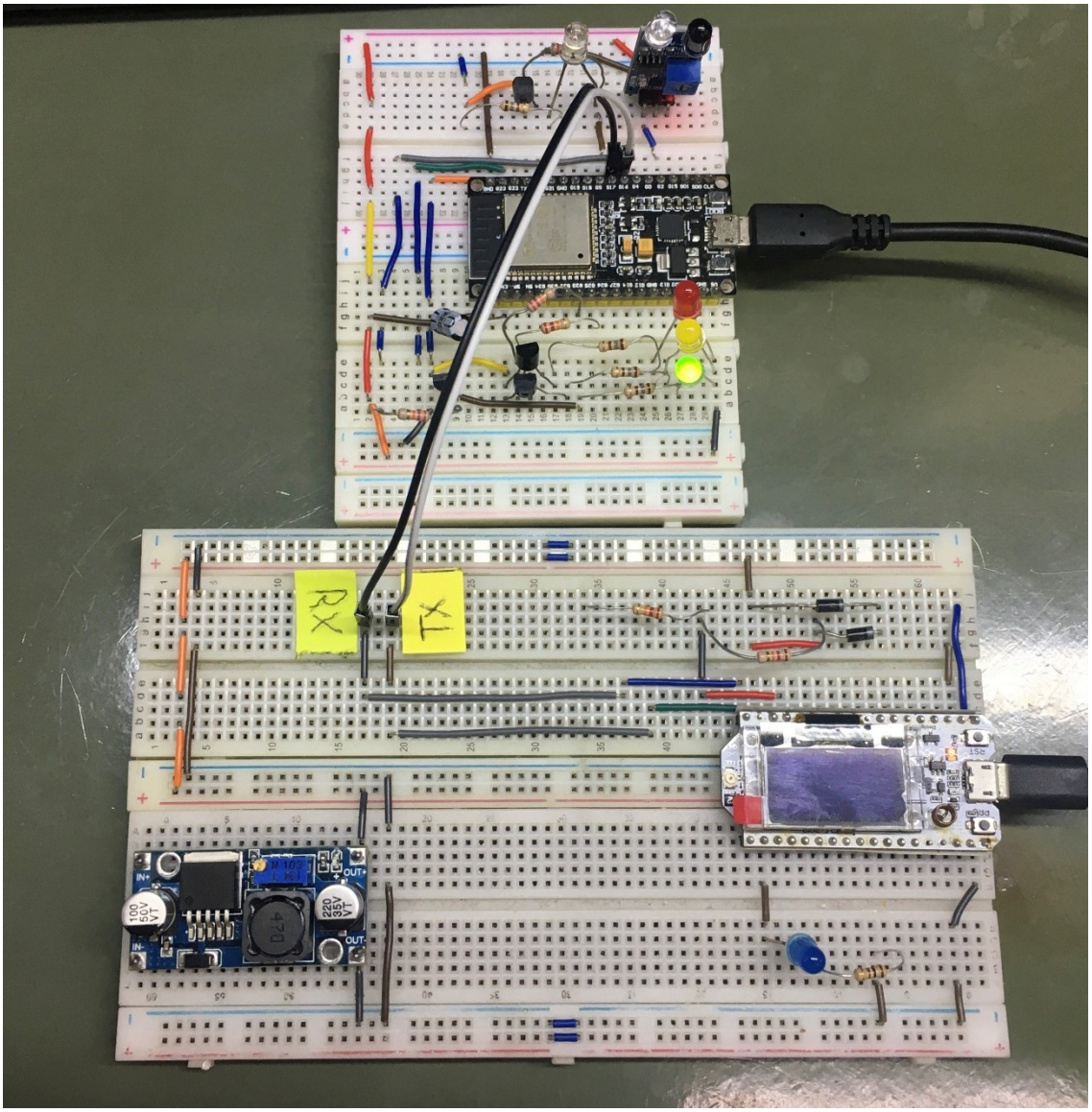
Figura 33 – Trava aberta e ocupada após solicitação de autenticação



Fonte: Produção do próprio autor.

Com a identificação da atuação do usuário sobre a trava, é realizada, no sistema de armazenamento, a dissociação entre trava e o usuário e a trava passa então a estar livre. A disponibilidade do sistema é também atualizada. A Figura 34 mostra o estado livre da trava após identificação de atuação do usuário.

Figura 34 – Trava liberada após autenticação



Fonte: Produção do próprio autor.

4.5 Discussões

A implementação da prova de conceito tem o potencial de possibilitar melhor entendimento da generalização proposta neste texto, tornando assim mais claras as características gerais e específicas de um sistema de controle de travas. Em comparação com outros trabalhos disponíveis na literatura e já citados, as negligências ou omissões relacionadas à segurança também se tornam explícitas.

Destaca-se, porém, que decisões diferentes de projeto quanto ao *software* e ao *hardware* e até mesmo quanto a arquitetura podem também ser viáveis e até mais eficientes para determinadas aplicações. Algumas das decisões tomadas são relativamente genéricas e não apresentam explicita vantagem em relação a outros métodos e por este motivo o critério de conveniência foi adotado.

Algumas simplificações e funcionalidades não implementadas também limitam a validação da proposta e devem ser mais bem exploradas. Dentre elas o monitoramento da alimentação de energia do sistema e a encriptação de mensagens críticas. Ambos são relevantes para a segurança do sistema e não foram implementados de forma completa por limitações de tempo.

A utilização de placas para desenvolvimento e módulos de *hardware*, como as placas HELTEC e WROOM, possibilitaram maior velocidade e confiabilidade durante o desenvolvimento, que foram de grande valia. O *software* implementado utilizando APIs e HALs provou ser mais eficiente em relação ao processo de encontrar erros e de entender o todas as etapas do código ainda durante o desenvolvimento. Esta constatação foi coerente com o que é apontado no livro *Reusable Firmware Development,* de Beningo (2017).

Não obstante, a implementação mínima apresentada neste trabalho não levou em consideração fatores como uso mínimo do *hardware* com foco em diminuição de custos, projeto de encapsulamentos mecânicos para os módulos do sistema e outros fatores. Desta forma, cumpre meramente o propósito de prova de conceito.

5 CONCLUSÃO

Com os avanços das tecnologias para implementação de sistemas embarcados e a participação desses sistemas em processos cotidianos, muitos problemas da sociedade foram resolvidos ou mitigados. O problema de controle de travas, por sua vez, ainda não possui soluções gerais fortemente estabelecidas e sistemas para controle de travas eletrônicas são ainda pouco populares, haja vista a necessidade de segurança e os custos associados.

Foi possível por meio deste trabalho demonstrar generalidades presentes em sistemas de controle de travas e trazer luz aos riscos e desperdícios de recursos apresentados em alguns trabalhos disponíveis na literatura e já citados neste texto, os quais podem ser reduzidos com a utilização de uma solução geral e devidamente validada.

Com a utilização de métodos para desenvolvimento de sistemas embarcados seguros aplicados ao problema de travas eletrônicas partindo de uma ótica generalista, foi possível a elaboração de uma arquitetura adequada a diversas aplicações e que leva em consideração aspectos de segurança e, demonstrar, por meio de uma prova de conceito, o correto funcionamento do que foi proposto.

A aplicação modelo de desenvolvimento descrito permitiu que a etapa de implementação do sistema fosse direta, tendo em vista que as decisões e processos já haviam sido definidos. Além disso, com a conferência prévia realizada, a lógica do sistema funcionou corretamente desde o primeiro teste.

Como trabalhos futuros, sugere-se a experiência de utilização da solução geral em aplicações específicas a fim de testar o resultado de um produto, além do que foi possível obter com a prova de conceito. A comparação quanto a segurança e tempo de desenvolvimento entre o desenvolvimento de um sistema para controle de travas eletrônicas utilizando o método convencional e utilizando a arquitetura generalista proposta também é um trabalho interessante para comparação dos métodos.

A realização de testes estruturados também é sugerida, visto que é um método de pôr à prova a estrutura proposta e encontrar pontos de melhoria. Além das sugestões quanto a estrutura

generalista, a implementação de um bicicletário eletrônico não mínimo e conectado à internet também é sugerida, visto que adiciona novos desafios e tem o potencial de resolver os problemas de falta de comodidade e segurança associados aos bicicletários mecânicos convencionais.

REFERÊNCIAS BIBLIOGRÁFICAS

AHELEROFF, S.; XU, X.; LU, Y.; ARISTIZABAL, M.; VELÁSQUEZ, J. P.; JOA, B.; VALENCIA, Y. IoT-enabled Smart Appliances Under Industry 4.0: A case study, **Advanced Engineering Informatics**, [*s. l.*], v. 43, fev. 2020.

ANSI C HASH TABLE. **POKRISTENSSON**, 2021. Disponível em: <http://pokristensson.com/strmap.html>. Acesso em: 5 de out. de 2021.

ASCIIDOC. **AsciiDoc**, 2021. Página Inicial. Disponível em: < https://tecnoblog.net/247956/referencia-site-abnt-artigos/>. Acesso em: 5 de out. de 2021.

BALL, S. R. **Embedded Microprocessor Systems**: real world design. 3rd. Ed. London: Newnes, 2002.

BARBI, I. Conversores CC-CC Básicos Não Isolados. 2nd. Ed. Florianópolis: Edição dos Autores, 2006.

CHEONG, S; LING, H; TEH, P. Secure Encrypted Steganography Graphical Password Scheme for Near Field Communication Smartphone Access Control System, **Expert Systems with Applications**, [*s. l.*], p. 3561-3568, v. 41, 2014.

CHIUCHISAN, I.; GEMAN, O. Trends in Embedded Systems for e-Health and Biomedical Applications, **2016 International Conference and Exposition on Electrical and Power Engineering (EPE)**, [*s. l.*], p. 304-308, v. 43, out. 2016.

DILL, J. Bicycling for Transportation and Health: the role of infrastructure. **Journal of Public Health Policy**, Portland, v. 30, p. 95-110, fev. 2009.

ESP32-S2 FEATURES. **Espressif**, 2021. Página inicial. Disponível em: < https://www.espressif.com/en/products/socs/esp32-s2 >. Acesso em: 5 de out. de 2021.

FAROOQ, U.; HASAN, M. U.; AMAR, M.; HANIF, A.; ASAD, M. U. RFID Based Security and Access Control System, **IACSIT International Journal of Engineering and Technology**, [*s. l.*], v. 6, p. 309-314, ago. 2014.

FREERTOS: real-time operating system for microcontrollers. **freeRTOS**, 2021. Disponível em: < https://www.freertos.org/>. Acesso em: 5 de out. de 2021.

WIFI LORA 32 (V2). **HELTEC**, 2021. Disponível em: < https://heltec.org/project/wifi-lora-32/ >. Acesso em: 5 de out. de 2021.

HTWE, K. M.; HTUN, Z. M. M; TUN, H. M. Design and Implementation of Bank Locker Security System Based on Fingerprint Sensing Circuit and RFID Reader, **International Journal of Scientific & Technology Research**, [*s. l.*], p. 6-10, v. 4, jul. 2015.

HUANG, Y.; YANG, Z.; XIONG, S. Design and Implementation of a Bicycle Parking System Based on Internet-of-things, **National Conference on Information Technology and Computer Science (CITCS)**, [*s. l.*], p. 897-900, v. 1, nov. 2012.

Kinetis K8x Secure Microcontrollers (MCUs) based on Arm Cortex-M4 Core. **NXP**, 2021. Disponível em: < https://www.nxp.com/products/processors-and-microcontrollers/arm-microcontrollers/general-purpose-mcus/k-series-cortex-m4/k8x-secure:K8X-SCALABLE-SECURE-MCU>. Acesso em: 5 de out. de 2021.

KLEIDERMACHER, K; KLEIDERMACHER, M. **Embedded Systems Security**: practical methods for safe and secure software and systems development. 1st. Ed. Massachusetts: Newnes, 2012.

LIEROP, D. V; GRIMSRUD, M.; EL-GENEIDY, A. Breaking into Bicycle Theft: insights from montreal, canada. **International Journal of Sustainable Transportation**, [*s. l.*], v. 9, n. 7, p. 490-501, fev. 2015.

MARTIN, R. C. **Clean Code**: a handbook of agile software craftsmanship. Upper Saddle River. 1st. Ed. New Jersey: Prentice Hall PTR, 2009.

MARWEDEL, P. **Embedded System Design**: embedded systems, foundations of cyber-physical systems, and the internet of things. 3rd. Ed. Gewerbestrasse: Springer Nature, 2018.

MATTERN, F.; FLOERKERMEIER, C. From the internet of computers to internet of things. **From Active Data Management to Event-Based Systems and More**, [*s. l.*], v. 6462, p. 242-259, nov. 2010.

MENEGHELLO, F.; CALOER, M.; ZUCCHETTO, D.; POLESE, M.; ZANELLA, A. IoT: Internet of Threats? a survey of practical security vulnerabilities in real iot devices, **IEEE Internet of Things Journal**, [*s. l.*], p. 8182-8201, v. 6, out. 2019.

NOERGAARD, T. Embedded Systems Architecture: a comprehensive guide for engineers and programmers. London: Newnes, 2005.

OSHANA, R; KRAELING, M. Software Engineering for Embedded Systems. *In*: OSHANA, R. **Software Engineering of Embedded and Real-Time Systems**. 1st. Ed. Washington: Newnes. 2013.

PLATFORMIO. **PlatformIO**, 2021. Página Inicial. Disponível em: <https://docs.platformio.org/en/latest/>, Acesso em: 5 de out. de 2021.

PEARSON, B; LUO, L; ZHANG, Y; DEY, R; LING, Z; BASSIOUNI, M; FU, X. On Misconception of Hardware and Cost in IoT Security and Privacy, **ICC 2019 – 2019 IEEE International Conference on Communications (ICC)**, p. 1-7, nov. 2019.

PONT, M, J. **The Engineering of Reliable Embedded Systems**. 2nd. Ed. Leicestershire: SafeTTy Systems Ltd, 2016.

SHOSTACK, A. **Threat Modeling: designing for security**. 1st. Ed. Indiana: Wiley Publishing, 2014.

VISUAL STUDIO CODE. **Visual Studio Code**, 2021, Docs. Disponível em: < https://code.visualstudio.com/docs>, Acesso em: 5 de out. de 2021.

WORTMANN, F; FLÜCHTER. Internet of Things – Technology and Value Added, **Business & Information Systems Engineering**, [*s. l.*], v. 57, p. 221-224, mar. 2015.

ZE-HONG, S; GUANG-YUAN, Z. Multi-functional Parcel Delivery Locker System, **2015 International Conference of Computer and Computational Sciences (ICCCS)**, [*s. l.*], v. 5, p. 207-210, dez. 2015.

ZIGANSHIN, E. G; NUMEROV, M. A; VYGOLOV, S. A. UWB BABY MONITOR, **International Conference Ultrawideband and Ultrashort Impulse Signals**, [*s. l.*], v. 5, p. 159-161, set. 2010.