Universidade Federal de Campina Grande

Centro de Engenharia Elétrica e Informática

Departamento de Engenharia Elétrica e Informática

Ícaro Modesto Granja Aguiar

Relatório de Estágio Supervisionado

Laboratório de Instrumentação e Metrologia Científicas - LIMC

Campina Grande, Paraíba

Outubro de 2024

Relatório de Estágio Supervisionado

Laboratório de Instrumentação e Metrologia Científicas - LIMC

Relatório de Estágio Supervisionado submetido à Unidade Acadêmica de Engenharia Elétrica da Universidade Federal de Campina Grande como parte dos requisitos necessários para a obtenção do grau de Bacharel em Ciências no Domínio da Engenharia Elétrica.

Ícaro Modesto Granja Aguiar

Orientando

Prof[a]. Dra. Georgina Karla de Freitas Serres

Orientadora

Campina Grande, Paraíba

Outubro de 2024

Dedico este trabalho a minha companheira, Yasmin.
que com todo seu amor, carinho e cuidado,
se tornou parte do meu corpo e da minha alma,
e estará presente em cada linha que eu hei de escrever.

AGRADECIMENTOS

Em primero lugar agradeço aos meus pais, Klaudy e Hermógenes, que abdicaram de tanto para que os meus sonhos se tornassem realidade, me sustetaram quando as minhas pernas falharam e me deram mais do que eu serei capaz de um dia retribuir.

Agradeço ao meu irmão, Iago, por ser um companheiro fiel e leal nas horas mais difíceis e a meu sobrinho, Davi, pelos sorrisos nos momentos em que estive mais triste e pelas noites de café e jogos.

Agradeço a João e Duda, que enquanto estive longe de casa, foram a minha família em Campina Grande.

Agradeço aos meus avós, Maria, Mário (*in memoriam*), Socorro e Juvany, pelo amor e pelos cuidados despendidos a mim durante toda a minha vida.

Agradeço ao meu tio e padrinho, Klayton, e meu tio Nelson (*in memoriam*) pelos conhecimentos passados, histórias engraçadas e músicas cantadas ao longo de todos esses anos.

Agradeço a todos os meus tios e tias, que são muitos para citar, pelo apoio incondicional e pelo carinho em todos esses anos.

Agradeço ao meu orientador, Raimundo Freire, por ter me recebido em seu laboratório e permitido a realização desse trabalho.

Agradeço a minha orientadora, Georgina Karla, pelo acolhimento e paciência ao longo do curso e desse estágio.

Agradeço a professora Alana Ramos, responsável pela minha última aula de graduação, pelos conselhos que ela me deu e por tornar o meu último período muito mais divertido, leve e tranquilo.

Agradeço aos meus amigos de laboratório, em especial, Henrique, Sávio, Sueldo, Thiago César e meus pupilos, José Henrique e Natanael, pela indispensável ajuda e inseparável companhia, sem as quais não seria possível a realização desse estágio.

E um agradecimento mais que especial a Duda, por todos os momentos divertidos, pudins e cortes de barba ao longo desse ano.

RESUMO

O relatório de estágio descreve a implementação de um Sistema de Controle de Acesso com tecnologia RFID no Laboratório de Instrumentação e Metrologia Científicas (LIMC) da Universidade Federal de Campina Grande. O estagiário realizou diversas tarefas, incluindo seleção de componentes, projeto de circuitos, programação e integração com banco de dados. O sistema, utilizando um ESP32, leitor RFID MFRC522 e Raspberry Pi como servidor, melhorou a segurança e o gerenciamento de acesso do laboratório. O projeto envolveu também a fabricação de PCBs e caixas de montagem. Este estágio proporcionou uma valiosa experiência prática em sistemas embarcados, eletrônica, processamento de sinais e programação.

Palavras-chave: Controle de acesso, RFID, Banco de dados, ESP32, Automação, Segurança.

ABSTRACT

The internship report describes the implementation of an Access Control System using RFID technology at the Laboratory of Instrumentation and Scientific Metrology (LIMC) of the Federal University of Campina Grande. The intern carried out various tasks, including component selection, circuit design, programming, and integration with a database. The system, utilizing an ESP32, an MFRC522 RFID reader, and a Raspberry Pi as the server, improved the security and access management of the laboratory. The project also involved the fabrication of PCBs and mounting enclosures. This internship provided valuable practical experience in embedded systems, electronics, signal processing, and programming.

Keywords: Access control, RFID, Database, ESP32, Automation, Security.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

A realização de estágios está presente na grade curricular do curso de engenharia elétrica e é um componente obrigatório na formação dos estudantes. Além disso, o estágio permite que os alunos apliquem a teoria aprendida em sala de aula em situações práticas e possibilita o desenvolvimento de habilidades práticas e técnicas que são essenciais para a profissão. Isso inclui a manipulação de equipamentos, a resolução de problemas técnicos e a implementação de projetos. A experiência do estágio também ajuda os estudantes a ajustarem suas expectativas e interesses profissionais, explorando diferentes áreas de atuação e identificando qual caminho desejam seguir após a graduação.

No decorrer desse relatório serão expostas as atividades realizadas pelo estagiário Ícaro Modesto Granja Aguiar sob orientação da Prof[a]. Dra. Georgina Karla de Freitas Serres e supervisão do Prof. Dr. Raimundo Carlos Silvério Freire. O estágio foi do tipo supervisionado e realizado no Laboratório de Instrumentação e Metrologia Científicas (LIMC), do Departamento de Engenharia Elétrica (DEE) na Universidade Federal de Campina Grande (UFCG). As atividades ocorreram entre os dias 02 de Julho de 2024 e 09 de Outubro de 2024, com carga horária semanal de 15 horas, totalizando uma carga horário de 225 horas.

O estágio foi focado na implementação de um Sistema de Controle de Acesso utilizando tecnologia RFID. A implementação de sistemas de controle de acesso tem se tornado cada vez mais importante para garantir a segurança e eficiência em diversos ambientes, desde corporações até instituições educacionais. Neste contexto, a tecnologia de Identificação por Radiofrequência (RFID) tem se demonstrando uma opção segura e eficaz [Munoz-Ausecha, Ruiz-Rosero e Ramirez-Gonzalez 2021, Anitha et al. 2023]. Ao longo do período de estágio, foram explorados os aspectos técnicos, desafios e benefícios da integração desta tecnologia em um ambiente real. O projeto visou não apenas aprimorar a segurança das instalações, mas também otimizar o fluxo de pessoas, demonstrando a aplicabilidade e a relevância da tecnologia RFID no cenário atual de gestão de acesso.

Durante o estágio foram abrangidos diversos aspectos, como design do sistema, programação de microcontroladores, fabricação de placas de circuito impresso (PCB) e integração com bancos de dados. Este projeto relacionou várias áreas de conhecimento, incluindo sistemas embarcados, eletrônica analógica e digital, processamento de sinais, redes

de comunicação e programação de sistemas. A experiência prática permitiu a aplicação de conhecimentos teóricos em áreas como eletromagnetismo (princípios de funcionamento do RFID), circuitos elétricos (design de interfaces de hardware), e sistemas de controle (lógica de acesso e automação). Além disso, o projeto envolveu aspectos de segurança da informação e gerenciamento de dados, demonstrando a natureza interdisciplinar da Engenharia Elétrica moderna.

2 OBJETIVOS

2.1 Objetivo Principal

Implementação de um Sistema de Controle de Acesso usando RFID no Laboratório de Instrumentação e Metrologias Científicas da UFCG.

2.2 Objetivos Específicos

- Instalação de Fechaduras Eletromecânicas;
- Instrumentação de Circuito de Controle e Acionamento de Fechadura Eletromecânica em Placa de Circuito Impresso;
- Fabricação de Peças para Montagem usando Impressora 3D;
- Programação de Microcontroladores (ESP32);
- Criação de Banco de Dados para Controle de Acesso ao Laboratório.

3 REVISÃO DA LITERATURA

A tecnologia de identificação por rádio frequência (RFID) é um componente essencial em diversos setores, particularmente em sistemas de segurança. A seguir, será apresentada uma revisão abrangente sobre os fundamentos do RFID, suas aplicações e a importância dessa tecnologia na segurança.

O RFID opera por meio da interação entre tags, que contêm informações eletrônicas, e leitores, que emitem sinais de rádio para comunicá-las. O sistema é composto por

três principais elementos: a tag RFID, o leitor e o middleware, que processa as informações recebidas. As tags podem ser passivas, ativas ou semi-ativas. As passivas não possuem bateria e dependem da energia do sinal emitido pelo leitor, enquanto as ativas possuem uma fonte de energia interna que permite maior alcance e capacidade de armazenamento de dados [Chocholac et al. 2021].

Os dados armazenados nas tags podem variar desde informações simples, como um número de identificação, até dados complexos que podem ser utilizados para acessar sistemas ou informações específicas. A comunicação entre o leitor e a tag é feita por meio de micro-ondas ou sinais de rádio, permitindo a identificação automática e sem contato, o que aumenta a eficiência e reduz o tempo de operação em diversas aplicações.

As aplicações do RFID são bastante amplas. No varejo, esta tecnologia é utilizada para gerenciar inventários de forma mais eficiente, possibilitando a rastreabilidade em tempo real dos produtos [Mishra e Mohapatro 2020, Tan e Sidhu 2022, Kumar, Prince e Shankar 2021]. Isso não só melhora o controle sobre o estoque, mas também minimiza perdas e furto de mercadorias.

Em ambientes corporativos, o RFID é empregado em controles de acesso, onde leitores são instalados em entradas e saídas para monitorar e registrar a passagem de funcionários e visitantes. Isso proporciona um nível elevado de segurança, pois permite o controle e a auditoria de quem está presente em uma determinada área [Su et al. 2023,Wan, Tanriover e Shah 2020]. Além disso, o uso de tags RFID em bens valiosos, como equipamentos eletrônicos ou documentos sensíveis, permite o rastreamento e a proteção contra furto ou extravio.

A importância do RFID em sistemas de segurança se destaca em sua capacidade de oferecer uma solução robusta contra diversas ameaças. A tecnologia facilita a identificação precisa e rápida de pessoas e objetos, aumentando a segurança em ambientes críticos, como bancos, aeroportos e instalações governamentais. Além disso, a utilização de RFID em sistemas de alarme pode ativar notificações imediatas em casos de acessos não autorizados ou movimentos suspeitos.

Outra vantagem significativa do RFID é a capacidade de coletar dados em tempo real [Haibi et al. 2023]. Isso permite que os gestores de segurança analisem o comportamento dos usuários e objetos em tempo real, possibilitando a tomada de decisões informadas e rápidas em resposta a potenciais incidentes. Sistemas avançados integrados

com RFID podem ainda ser conectados a redes inteligentes, ampliando o monitoramento e a resposta em situações de emergência [Pawłowicz, Salach e Trybus 2019].

Apesar dos múltiplos benefícios, a segurança dos sistemas que utilizam RFID é uma preocupação central. As vulnerabilidades incluem a possibilidade de clonagem de tags e a interceptação de dados transmitidos. A proteção das informações contidas nas tags e a implementação de protocolos de segurança são essenciais para mitigar esses riscos [Li e Liu 2021]. Medidas como a criptografia de dados e a utilização de protocolos de autenticação forte são recomendadas para proteger os sistemas RFID contra acessos não autorizados.

A literatura destaca que, à medida que a tecnologia evolui, também devem evoluir as estratégias de segurança. A integração do RFID com outras tecnologias de segurança, como biometria e sistemas de vídeo vigilância, pode criar uma solução mais robusta e eficaz. A criação de um ambiente seguro requer não apenas a implementação da tecnologia, mas um planejamento cuidadoso e uma avaliação contínua das ameaças e vulnerabilidades.

4 ATIVIDADES REALIZADAS

Nesse capítulo são apresentadas as atividades realizadas durante o estágio. As atividades se dividiram em: Escolha do Microcontrolador e Leitor RFID; Projeto, Simulação e Teste dos Circuitos de Controle e Alimentação; Programação; Integração com o Banco de Dados; Projeto e Fabricação das PCBs e Caixas para Montagem e Instalação Final.

4.1 Escolha do Microcontrolador e do Leitor RFID

A escolha do microcontrolador é um aspecto crucial no desenvolvimento de um sistema de acesso que utiliza RFID. Essa decisão impacta diretamente o desempenho, a eficiência e a confiabilidade do sistema como um todo.

Um microcontrolador adequado deve possuir capacidade de processamento suficiente para lidar com a leitura e interpretação dos dados RFID em tempo real. Isso inclui a decodificação dos sinais recebidos, a comparação com um banco de dados de usuários autorizados e a tomada de decisões rápidas sobre a concessão ou negação de acesso.

A escalabilidade do sistema também deve ser considerada na escolha do microcontrolador. Um dispositivo com recursos extras permite futuras expansões do sistema, como a adição de novos sensores, interfaces de usuário mais sofisticadas ou integração

com outros sistemas de segurança.

Por fim, a disponibilidade de ferramentas de desenvolvimento, bibliotecas e suporte técnico para o microcontrolador escolhido pode facilitar significativamente o processo de desenvolvimento e manutenção do sistema de acesso RFID.

Para a implementação do sistema foram considerados dois microcontroladores, o Arduino e a ESP32. Na Tabela 1 são apresentadas as principais características do Arduino e da ESP32.

Tabela 1: Principais Características dos Microcontroladores Avaliados.

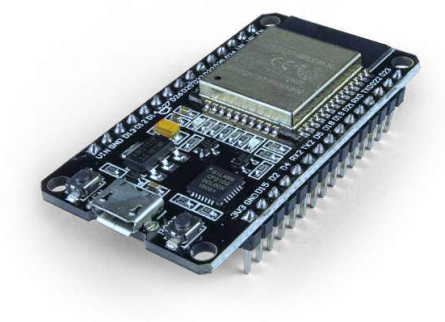
	Arduino Uno	**ESP32-WROOM**
Clock da CPU	16 MHz	240 MHz
Memória Flash	32 kb	16 Mb
Memória Ram	2 kb	512 kb
Arquitetura	8 bits	32 bits
Número de Núcleos	1	2
Módulo Wi-Fi	Não	Sim

Fonte: Autoria Própria, 2024.

O ESP32 foi escolhida devido a todas as vantagens apresentadas na Tabela 1, principalmente devido ao fato de possuir módulo Wi-Fi integrado, que é essencial para a comunicação com o banco de dados, evitando a necessidade de compra e da conexão de um outro componente, caso a escolha fosse o Arduino Uno.

O leitor escolhido foi o MFRC522, devido a grande disponibilidade no mercado. A frquência de operação desse leitor é de $13,56$ MHz. O ESP32, o leitor MFRC522 e a ligação entre os dois podem ser vistos na Figura 1, Figura 2 e Figura 3, respectivamente.

Figura 1: Microcontrolador ESP32.

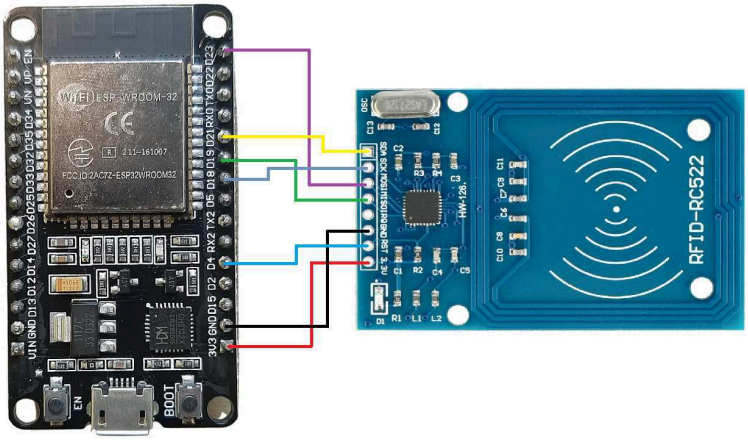


Fonte: RoboCore, 2024.

Figura 2: Leitor RFID MFRC522.

Fonte: Eletrogate, 2024.

Figura 3: Ligação entre o ESP32 e o leitor MFRC522.



Fonte: Autoria Própria, 2024.

4.2 Projeto, Simulação e Teste dos Circuitos de Controle e Alimentação

Essa etapa é muito importante no desenvolvimento do um sistema de acesso, pois garante a funcionalidade, confiabilidade e eficiência do sistema. Sua importância se manifesta em vários aspectos:

- Otimização do desempenho: O projeto adequado dos circuitos permite que o sistema opere de forma eficiente, minimizando o consumo de energia e maximizando a velocidade de resposta.
- Confiabilidade: A simulação e teste dos circuitos ajudam a identificar possíveis falhas ou pontos fracos antes da implementação, aumentando a confiabilidade do sistema.
- Redução de custos: A simulação e teste prévios podem evitar erros custosos na fase de produção, economizando recursos.

A alimentação do circuito, incluindo o microcontrolador, o leitor e a fechadura, foi feita usando uma fonte de alimentação de 12 V, como a representada na Figura 4. Como o ESP32 precisa ser alimentado com uma tensão de 5 V, foi usado o regulador de tensão LM7085 para baixar a tensão de entrade de 12 V para 5 V.

Para realizar o acionamento da fechadura é necessário uma chave que ligue ou desligue de acordo com um sinal recebido pelo microcontrolador. Foram considerados dois MOSFETs tipo N para a implementação da chave, o IRF530 e o IRF840. O IRF840 foi escolhido por possuir uma resistência maior entre dreno e fonte ($0,85\ \Omega$) em relação ao IRF530 ($0,18\ \Omega$), diminuindo o cunsumo de corrente do circuito.

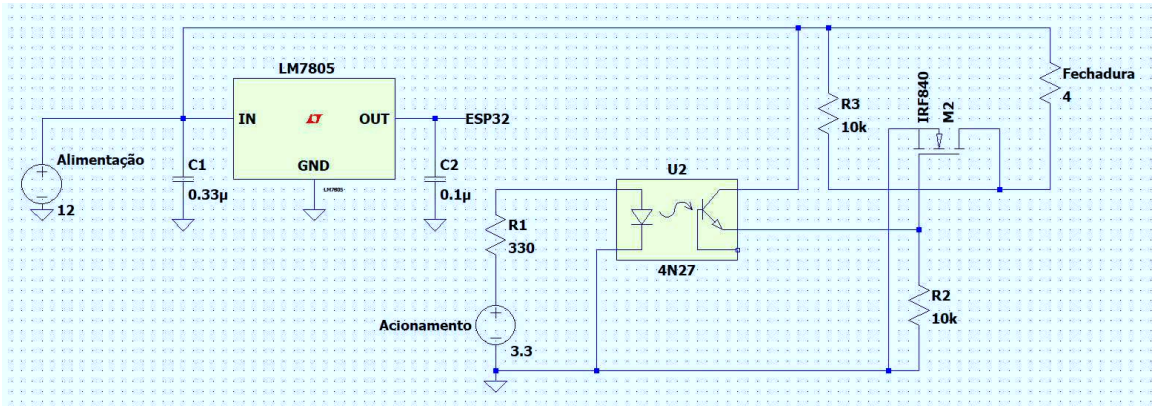
Figura 4: Fonte de Alimentação DC de 12V.

Fonte: Usinainfo, 2024.

Outra escolha importante no projeto do circuito foi o uso de um optoacoplador. O optoacoplador foi usado para fornecer um isolamento elétrico entra o sinal de controle, vindo das ESP32, e a chave. Além disso o optoacoplador protege os circuitos contra interferência elétricas, picos de tensão ou falhas no sistema. Ele também é necessário para reduzir o ruído e fazer a comunicação entre a parte digital (ESP32) e analógica (IRF840, LM7805, fonte de alimentação, resistores, etc.) do circuito. O modelo utilizado foi o 2N47, pela grande quantidade no labotatório, mas outros optoacopladores testados também apresentaram um ótimo desempenho.

Após a escolha dos principais componentes o circuito de alimentação e acionamento foi simulado no LTSpice. A simulação pode ser vista na Figura 5.

Figura 5: Simulação do Circuito de Alimentação e Acionamento no LTSpice.



Autoria Própria, 2024.

Abaixo é explicado, de forma sucinta o funcionamento do circuito.

- A fonte de alimentação é conectada ao LM7805, para forcencer a alimentação ao ESP32, a fechadura, tanto de forma direta como atráves do resistor R3, e ao optoacoplador;
- Os capacitores C1 e C2 estão conectados na entrada e saída do LM7805, servindo como desacoplamento para suavizar a tensão e minimizar ruídos no fornecimento de energia. Esses capacitores ajudam a manter a estabilidade do regulador e evitam oscilações;
- A fonte "Acionamento"representa o sinal de controle vindo da ESP32 e o resistor R1 é usado para diminuir a corrente que para do LED do optoacoplador;
- O optoacoplador possui um LED e um fototransistor. Quando o LED acende, devido ao sinal gerado pela ESP32, o fototransistor permite a conexão entre o coletor e o emissor;
- Quando a fonte "Acionamento"tem tensão zero, a porta do IRF840 fica conectada ao terra através do resistor R2 e não há conexão entre a fonte o dreno. Os dois terminais da fechadura estão conectadas a fonte de 12 V, então a sua diferença de tensão é de 0 V e ela permanece fechada;
- Quando a fonte "Acionamento"tem tensão de $3,3$ V(tensão máxima de saída do ESP32) o optoacoplador é acionado, e a porta do IRF840 é conectada a fonte de 12 V, essa tensão faz com que haja uma ligação entre a fonte e o dreno do MOSFET. A conexão entre a fonte e o dreno faz com que o terminal da fechadura que está ligada ao resistor R3 se conecte ao terra e a diferença de tensão entre os terminais da fechadura é de 12 V, realizando o acionamento da mesma.

O circuito que contém o ESP32 e o leitor não foi simulado, ele foi apenas montado em uma protoboard usando as ligações apresentadas na Figura 3. Esse circuito foi usado para dois propósitos diferentes, o primeiro foi para o cadastro de novos usuários e o segundo para gerar o sinal de acionamento da fechadura, a programação utliziada para ambos os casos é apresentada na seção Programação. Após o teste de todos os circuitos separadamente eles foram montados em uma protoboard para a realização dos testes iniciais.

4.3 Programação

O ESP32 precisou ser programado tanto para realizar o cadastro das pessoas que teriam acesso ao laboratório e também para acionar a fechadura. O primeiro passo para a programação foi a instalação dos drivers CP210x, para que a ESP32 fosse reconhecida pelo computador. A instalação dos drivers é feita de forma simples:

- Os drivers podem ser baixados em: https://www.silabs.com/developers/usb-to-uart-bridge-vcp-drivers?tab=downloads;
- Após o download o arquivo deve ser extraído;
- Deve ser realizada a conexão entre a ESP32 e o computador usando um cabo USB
- Depois é necessário abrir o Gerenciador de Dispositivos, selecionar o dispositivo CP210x USB to UART > Atualizar driver > Procurar drivers no meu computador > Selecionar a pasta que foi extraída, marcar a opção ”incluir subpasta”> Avançar e aguardar a configuração.

A programação do ESP32 foi feita usando o Arduino IDE, que pode ser baixado em https://www.arduino.cc/en/software. Com a instalação do IDE feito é preciso agora instalar os drivers do ESP32 no software. Na interface do Arduino IDE é preciso selecionar Arquivo > Preferência > No campo ”URLs do Gerenciador de Placas Adicionais”colar o link ”https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json”> Ok.

Após isso é escolhido a opção Ferramentas > Placas > Gerenciador de Placas > Pesquisar por ”esp32”> instalar a versão mais recente da bilioteca ”esp32 por Espressif Systems”. Agora o computador e o Arduino IDE estão configurados para a programação do ESP32

4.3.1 Cadastro

O código para o circuito de Cadastro é apresentado no Anexo A.

- A diretiva #include é usada para adicionar os arquivos de cabeçalho necessários;
- A diretiva #define é usada para configurar os pinos do ESP32 em que os pinos RST e SS do leitor estão conectados;

- Em ”Credenciais da sua rede”devem ser colocados o nome e senha da rede Wi-Fi, para permitir a conexão da ESP32 com a internet;
- Em ”Nome do Domínio e URL ou endereço de IP com o caminho para os arquivos”devem ser colocados o IP onde está o banco de dados e o caminho onde estão localizados os arquivos .php. O IP pode ser encontrado usando o comando ”ipconfig”no windows ou ”ifconfig”no Ubuntu ou Raspberry Pi OS e o caminho é o da pasta que contém os arquivos .php, que será explicada na seção Integração com o Banco de Dados;
- O comando ”MFRC522 mfrc522(SS_PIN, RST_PIN);”inicia o leitor MFRC522;
- Na função ”Setup”é iniciada a conexão com o Wi-Fi e a conexão entre o ESP e o MFRC522;
- A função ”http.begin(Linhas);”é usada para verificar se já existe um cartão Mestre cadastrado no bando de dados, se não houver, é feito um cadastro de um cartão. O cartão Mestra é necessário para realizar o cadastro de novos usuários;
- Após o cadastro de um cartão Mestre, toda vez que um cartão Mestre for aproximado do leitor é aberto um menu para realizar o cadastro de um novo Mestre ou de um novo Usuário;
- O menu mostra as opções disponíveis no programa e guia o responsável pelo sistema ao longo do cadastro;
- No sistema é possível escolher em quais sala cada usuário poderá acessar ou não.
- A função ”http.begin(Cadastro);”salva os novos dados de usuário no banco de dados.

4.3.2 Acesso

O código para o circuito de Acesso é apresentado no Anexo B. O código segue algumas funções parecidas com o Cadastro, e apresenta algumas novas.

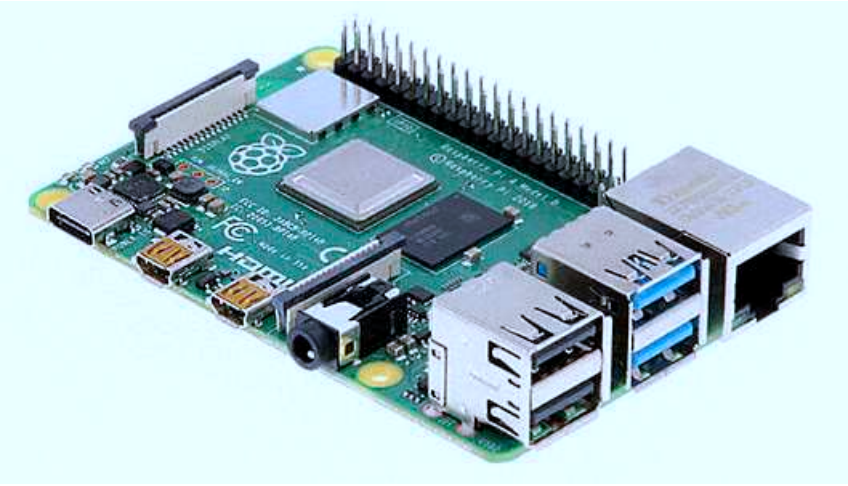
- A função ”#include <ArduinoOTA.h>”é usada para fazer a programação do ESP32 Over-the-Air (OTA), para que não seja necessário mexer na instalação todas as vezes que for preciso fazer uma atualização;

- A variável ”String Sala”contém o nome da sala em que o circito vai ser instalado, para poder fazer o controle com o banco de dados, se o usuário pode ou não acessar;
- Ao aproximar um cartão, é feita a leitura dos dados e comparada com o banco de dados para que seja tomada a decisão se o acesso é permitido ou não;
- A comparação é feita pela função ”http.begin(Acesso);”e ”http.POST(...);”
- A função ”respostaServidor = http.getString();”recebe a resposta do servidor para permitir o acesso ou bloqueio do usuário na sala.
- Se o acesso for permitido, os dados do usuário, a hora de acesso e a sala são salvos no banco de dados para futura conferência.

4.4 Integração com o Banco de Dados

Para a instalação do banco de dados foi escolhido um Raspberry Pi 4 (Figura 6), devido ao tamanho, poder de processamento e funcionalidades. O sistema operacional utilizado foi o Raspberry Pi OS Full (64-bits) que pode ser obtido gratuitamente com a ferramenta Raspberry Pi Imager.

Figura 6: Raspberry Pi 4.



DigiKey, 2024.

Depois da instalação do sistema operacional em um cartão microSD e da inicialização do Raspberry, foram necessário alguns passos para criar o banco de dados. Após abrir o Terminal do Raspberry Pi OS, os seguintes comandos devem ser executados:

- Atualizações necessarias: sudo apt update && sudo apt upgrade -y;
- Instalação do Apache: sudo apt install apache2 -y;
- Instalação do PHP:
  - sudo apt update && sudo apt install -y wget gnupg2 lsb-release;
  - sudo wget https://packages.sury.org/php/apt.gpg;
  - sudo apt-key add apt.gpg;
  - sudo echo ”deb https://packages.sury.org/php/ $(lsb_release -sc) main sudo tee /etc/apt/sources.list.d/php.list;
  - sudo apt update && sudo apt install -y php7.3 && sudo apt install -y php7.3-mysqli;
- Instalação do MariaDB Server:
  - sudo apt install mariadb-server php-mysql -y;
  - sudo service apache2 restart;
  - sudo mysql_secure_installation;
  - Após esse passos é necessário fazer a configuração do servidor:
    - Em ”Enter current password for root”pressione ENTER;
    - Y para escolher a nova senha e escolha a senha do banco de dados;
    - Em ”Remove anonymous users?”escolha ”Y”;
    - Em ”Disallow root login remotely?”escolha ”Y”;
    - Em ”Remove test database and access to it?”escolha ”Y”;
    - Em ”Reload privilege tables now?”escolha ”Y”;
- Instalaçao do phpMyAdmin:
  - cd /usr/share;
  - sudo wget https://files.phpmyadmin.net/phpMyAdmin/4.8.4/phpMyAdmin-4.8.4-all-languages.tar.gz;
  - tar -xzvf phpMyAdmin-4.8.4-all-languages.tar.gz;

– my phpMyAdmin-4.8.4-all-languages phpmyadmin;

– sudo ln -s /usr/share/phpmyadmin /var/www/html/phpmyadmin;

Após esses passos o Raspberry está pronto para ser utilizado como servidor de todo o sistema. No navegador dever ser acessado o link ”http://localhost/phpmyadmin/” e configurar um usuário para acessar o banco de dados. Após a configuração a tela de login deve ser como a apresentada na Figura 7. Após o login, na tela inicial, em ”Banco de Dados”é possível criar uma base de dados que irá conter todos os dados necessário para o funcionamento do sistema. E no novo banco de dados podem ser criadas as tabelas com as variáveis a serem utilizadas, um exemplo é mostrado na Figura 8.

Figura 7: Tela de Login do phpMyAdmin.

Autoria Própria, 2024.

Figura 8: Exemplo de uma base da dados, com as tabelas e variáveis utilizadas.

Autoria Própria, 2024.

Os arquivos .php devem ser colocados na pasta ”/var/www/html/”. Esses arquivos são os responsáveis pela comunicação entre o ESP32 e o Raspberry Pi. Nesses arquivos estão contidas todas as instruções para a requisição de dados necessário para que o ESP32 faça o controle do acesso ao laboratório.

Todos os códigos php utilizados estão disponibilizados nos Anexos C até G. O código em ”linhas.php” faz o teste se já existe um cartão mestre cadastrado, ele retorna o número de linhas da tabela ”Mestre”, se o número de linhas for zero, o ESP32 inicia o cadastro de um cartão Mestre.

O ”teste.php” é usado no cadastro de um novo usuário ou Mestre. Ele verifica se o cartão a ser cadastrado já está presente em algumas das tabelas do banco de dados.

O ”cadastro.php” realiza o cadastro de um novo usuário ou Mestre, se o ”teste.php” indicar que o novo cartão pode ser cadastrado. Ele adiciona o UID do cartão, nome do usuário, matrícula, telefone, cpf e email ao banco de dados na tabela selecionada no início do cadastro. Além disso, se o cadastro for de um usuário, é adicionado as salas que podem ser acessadas. Mestres podem acessar todas as salas sem a necessidade de nenhuma outra especificação.

O código ”acesso.php”checa se o cartão utilizado pode acessar um determinada sala, e se o acesso for permitido, os dados de quem está acessado, além da data e hora, são adicionados a tabela ”acesso”,

Em ”database.php” estão contidas algumas informações necessárias para o funcionamento dos outros códigos.

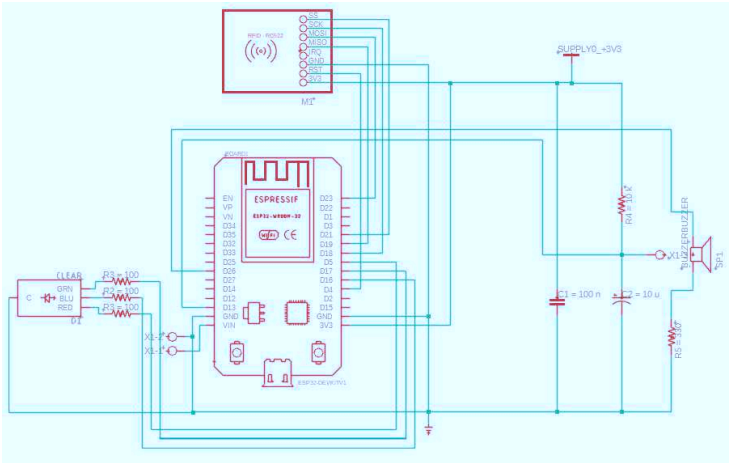
4.5 Projeto e Fabricação das PCBs e Caixas para Montagem

Para projetar tanto as placas de circuito impresso quanto as caixas necessárias para a montagem foi usado o Fusion 360, que pode ser obtido gratuitamente no site ”https://www.autodesk.com/products/fusion-360/personal” com o uso do e-mail institucional ”@ee.ufcg.edu.br”. O Fusion 360 foi escolhido tanto pela possibilidade de ser utilizado gratuitamente quanto por possibilitar o design de PCBs e de modelos para impressão 3D, evitando o uso de softwares diferentes para cada atividade.

4.5.1 Placas de Circuito Impresso

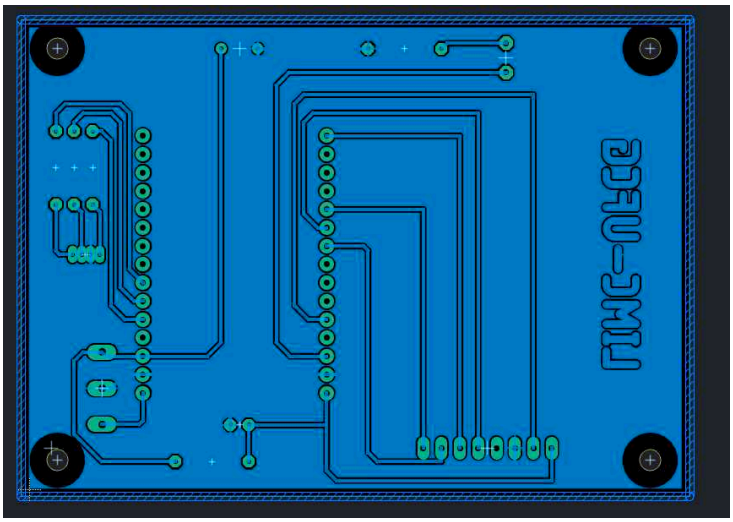
A primeira placa a ser apresentada é a de leitura da tag ou cartão RFID, que contém a ESP32 e o MFRC522. Foram adicionadas alguns outros componentes a placa, como um LED RGB, um buzzer, capacitores para minimizar os ruídos e resistores para limitar a corrente no circuito. O esquemático e o desenho podem ser vistos nas Figura 9 e Figura 10, respectivamente.

Figura 9: Esquemático da placa de leitura.



Autoria Própria, 2024.

Figura 10: Desenho da placa de leitura.



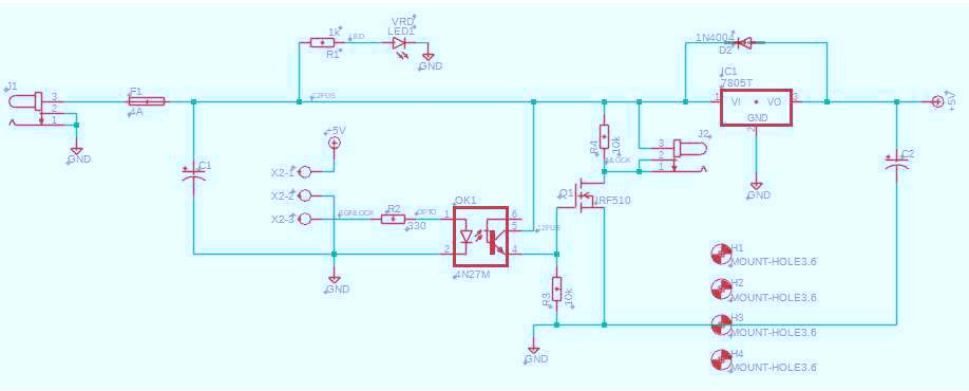
Autoria Própria, 2024.

As dimensões da placa são de $92,5934\ mm$ x $69,50\ mm$ e foi feita com um plano de terra para facilitar as ligações. As especificações do design da placa são listadas a seguir:

- Plano de terra:
  - Largura: $0,001\ mm$;
  - Isolamento: $0,254\ mm$;
  - Largura do Térmico: $0,254\ mm$;
  - Térmico: Sim;
  - Mostrar Preenchimento: Sim;
- Largura da Trilhas: $0,4064\ mm$;
- Os pads foram usados como nas biblioteca de cada componente;

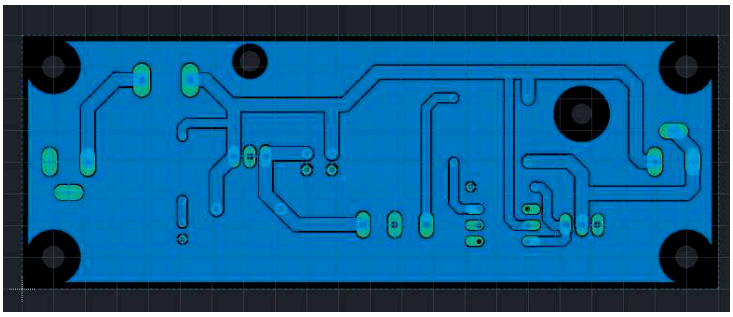
A placa de alimentação e acionamento e mostrada nas Figura 11 e Figura 12 e apresenta dimensões de $110,00\ mm$ x $40,00\ mm$. As configurações de design são as mesmas da placa de leitura, com exceção das trilhas. Foi usado largura de $2,0\ mm$ para as trilhas de alimentação e largura de $1,27\ mm$ para as demais.

Figura 11: Esquemático da placa de alimentação.



Autoria Própria, 2024.

Figura 12: Desenho da placa de alimentação.



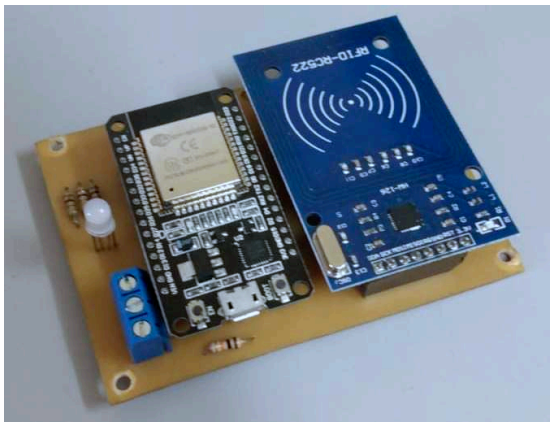
Autoria Própria, 2024.

Com os designs em mãos, as placas foram fabricadas usando uma prototipadora a laser, que pode ser vista na Figura 13. A prototipadora é controlada usando o software EZCAD2. As placas fabricadas podem ser vistas nas Figura 14 e Figura 15.

Figura 13: Prototipadora FIBER LASER.

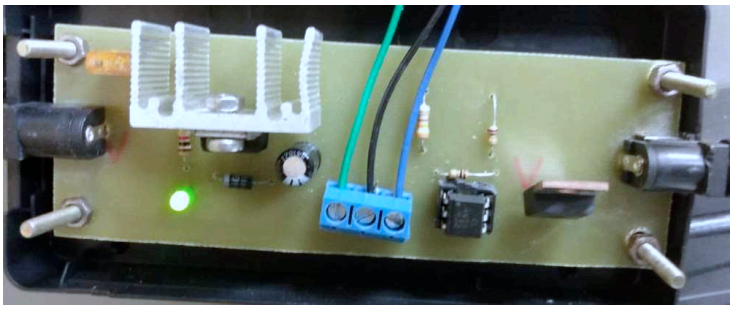
IRON LASER CNC, 2024.

Figura 14: Versão final da placa de leitura.



Autoria Própria, 2024.

Figura 15: Versão final da placa de alimentação e controle.



Autoria Própria, 2024.

Um detalhe importante na Figura 15 é que em apenas um dos MOSFETs foi colocado um dissipador, o LM7805. Esse escolha ocorreu porque apenas o LM7805 apresentou altas temperaturas durante os testes. O IRF840, devido a frequência muito baixa de chaveamento (o chaveamento ocorria apenas quando era feita a leitura de um cartão que possuia acesso), não esquentou.

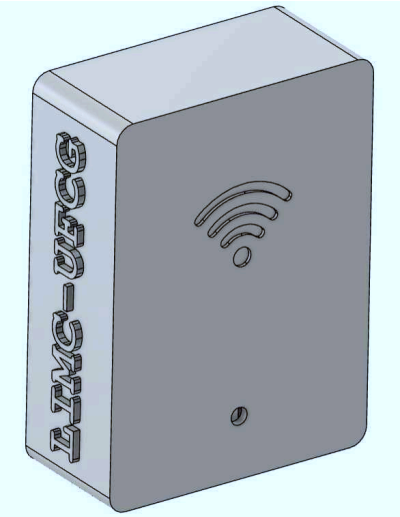
As configurações do software foram:

- Hatch:
  - Mark Contour;
  - Enable;
  - Follow edge once;
  - Cross hatch;
  - Line Space: $0,05\ mm$;
  - Average distribute line;
  - Loop discante:
- Laser:
  - Loop Count: 20;
  - Speed: 800 mm/s;
  - Power: 60%;
  - Frequency: 20 kHz;

4.5.2 Caixas de Montagem

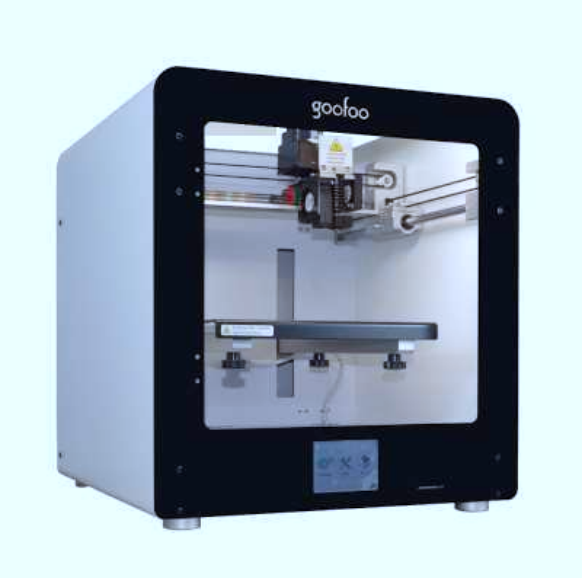
As caixas de montagem para o circuito da placa de leitura foram modeladas de acordo com o tamanho da PCB apresentada anteriormente. Foram feitas 6 caixas no total, uma para cada sala do laboratório. As caixas foram fabricadas com a impressora Goofoo Mini+. As caixas para o circuito de alimentação foram compradas já prontas. O modelo desenhado e a impressora podem ser vistos nas Figura 16 e Figura 17, respectivamente.

Figura 16: Modelo 3D da caixa.



Autoria Própria, 2024.

Figura 17: Impressora Goofoo Mini+.



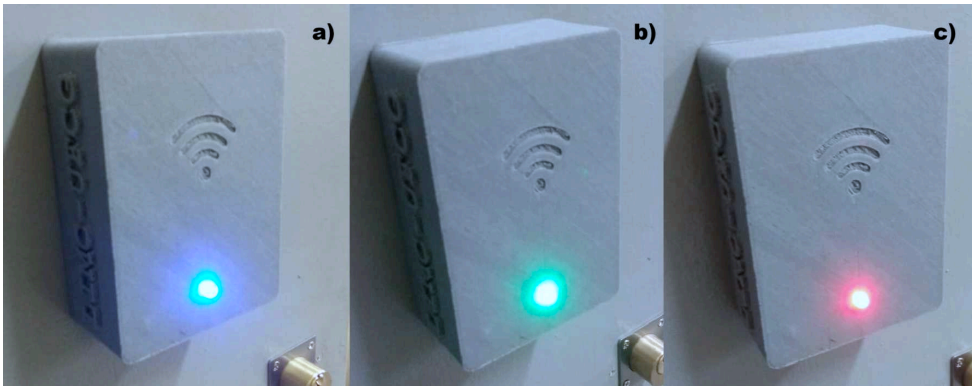
Simply Print, 2024.

4.6 Instalação Final

Com todas as placas feitas, caixas impressas, microcontroladores programados e banco de dados configurados, o passo final foi a instalação de todos os circuitos nas portas de cada sala do laboratório. Para a realização da instalação foram necessárias várias ferramentas, como furadeira, chave de fenda, martelo, retífica manual, prego e pararufos, lixas, entre outros. A montagem final pode ser vista na Figura 18.

Na Figura 18, a imagem a) indica que o sistema está funcionando corretamente. Na imagem b) a luz verde se acende mostrando que o usuário pode entrar na sala. E por fim, a imagem c) mostra um usuário tendo o seu acesso negado.

Figura 18: Instalação final do sistema: a) sistema em funcionamento, b) acesso permitido, c) acesso negado.



Autoria Própria, 2024.

5 CONSIDERAÇÕES FINAIS E TRABALHOS FUTUROS

A implementação de um Sistema de Controle de Acesso utilizando tecnologia RFID foi realizada com sucesso. Foram executadas diversas tarefas, incluindo a seleção de componentes adequados, como o microcontrolador ESP32 e o leitor RFID MFRC522, o projeto e simulação de circuitos de controle e alimentação, e a programação necessária para o funcionamento do sistema.

A integração do sistema com um banco de dados, utilizando um Raspberry Pi como servidor, permitiu um controle eficiente e seguro dos acessos ao laboratório. O projeto também envolveu a fabricação de placas de circuito impresso (PCBs) e caixas de montagem, demonstrando habilidades práticas em design e prototipagem.

O estágio proporcionou uma valiosa experiência prática, permitindo ao estudante

aplicar conhecimentos teóricos em áreas como sistemas embarcados, eletrônica, processamento de sinais e programação. Além disso, o projeto abordou aspectos importantes de segurança da informação e gerenciamento de dados.

A implementação bem-sucedida deste sistema de controle de acesso não apenas melhorou a segurança do laboratório, mas também ofereceu uma solução eficiente para o gerenciamento de acesso, demonstrando a relevância e aplicabilidade da tecnologia RFID em ambientes acadêmicos e profissionais.

Este estágio supervisionado cumpriu seu papel na formação do estudante, proporcionando uma experiência prática valiosa e preparando-o para os desafios do mercado de trabalho na área de Engenharia Elétrica.

Como trabalhos futuros podem ser feitas alterações e estudos com o objetivo de aumentar a segurança, eficiência e praticidade do sistema. Algumas sugestões são apresentadas a seguir:

- Integração com sistema de alarme: Conectar o sistema de controle de acesso a um sistema de alarme para maior segurança;
- Mudar a forma de comunicação: Realizar a comunicação entre a ESP32 e o banco de dados por outros meios, como LoRa ou Bluetooth, para permitir o funcionamento do sistema mesmo sem conexão internet;
- Instalação de baterias: Desenvolver um sistema de alimentação que mantenha o sistema funcionando mesmo em casos de falta de energia;
- Implementação de análise de dados: Desenvolver ferramentas para análise de padrões de acesso e geração de relatórios automáticos;
- Aprimoramento da interface do usuário: Melhorar a interface do sistema para facilitar o uso e a administração;
- Implementação de backup automático: Desenvolver um sistema de backup automático para garantir a integridade dos dados;
- Estudo de eficiência energética: Realizar uma análise do consumo de energia do sistema e implementar melhorias para redução do consumo;

- Implementação de protocolos de segurança avançados: Reforçar a segurança do sistema contra possíveis fraudes;
- Implementação de autenticação biométrica: Integrar um sistema de reconhecimento de impressões digitais ou facial para aumentar a segurança.

REFERÊNCIAS

ANITHA, G. et al. Efficient internet of things enabled smart healthcare monitoring system using rfid security scheme. In: *Intelligent Technologies for Sensors*. [S.l.]: Apple Academic Press, 2023. p. 125–143.

CHOCHOLAC, J. et al. Logistic technologies for tracking manufactured passenger cars on consolidation areas: Interpretative case study. *Transportation Research Procedia*, Elsevier, v. 53, p. 266–273, 2021.

HAIBI, A. et al. On the use of rfid middleware for real-time data stream processing. In: *Proceedings of the 6th International Conference on Networking, Intelligent Systems & Security*. [S.l.: s.n.], 2023. p. 1–5.

KUMAR, S. G.; PRINCE, S.; SHANKAR, B. M. Smart tracking and monitoring in supply chain systems using rfid and ble. In: IEEE. *2021 3rd International Conference on Signal Processing and Communication (ICPSC)*. [S.l.], 2021. p. 757–760.

LI, K.; LIU, J. A security protocol of rfid communication system based on password authenticated with provable security. *International Journal of Autonomous and Adaptive Communications Systems*, Inderscience Publishers (IEL), v. 14, n. 1-2, p. 64–82, 2021.

MISHRA, A.; MOHAPATRO, M. Real-time rfid-based item tracking using iot & efficient inventory management using machine learning. In: IEEE. *2020 IEEE 4th Conference on Information & Communication Technology (CICT)*. [S.l.], 2020. p. 1–6.

MUNOZ-AUSECHA, C.; RUIZ-ROSERO, J.; RAMIREZ-GONZALEZ, G. Rfid applications and security review. *Computation*, MDPI, v. 9, n. 6, p. 69, 2021.

PAWŁOWICZ, B.; SALACH, M.; TRYBUS, B. Smart city traffic monitoring system based on 5g cellular network, rfid and machine learning. In: SPRINGER. *Engineering Software Systems: Research and Praxis*. [S.l.], 2019. p. 151–165.

SU, Y.-H. et al. Tomoid: A scalable approach to device free indoor localization via rfid tomography. In: IEEE. *IEEE INFOCOM 2023-IEEE Conference on Computer Communications*. [S.l.], 2023. p. 1–10.

TAN, W. C.; SIDHU, M. S. Review of rfid and iot integration in supply chain management. *Operations Research Perspectives*, Elsevier, v. 9, p. 100229, 2022.

WAN, C.-Y.; TANRIOVER, C.; SHAH, R. C. Capturing customer browsing insights through rfid tag motion detection in high tag density environments. In: IEEE. *2020 IEEE International Conference on RFID (RFID)*. [S.l.], 2020. p. 1–8.

ANEXO A - CÓDIGO DE CADASTRO

```
#include <WiFi.h>
#include <HTTPClient.h>
#include <SPI.h>
#include <MFRC522.h>

#define RST_PIN      4
#define SS_PIN       21

// Credenciais da sua rede
const char* ssid = "";
const char* password = "";

// Nome do Domínio e URL ou endereço de IP com o caminho para os arquivos
const char* Cadastro = "http://meu_ip/cadastro.php";
const char* Linhas = "";
const char* Teste = "";

const String apiKeyValue = "";
int httpResponseCode;
String httpRequestData, TesteMestre, cartaouid, comando, TesteCartao;

MFRC522 mfrc522(SS_PIN, RST_PIN); // Iniciando o leitor

void setup() {

  Serial.begin(9600);
  Serial.println("\nLIMC-UFCG");
  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);
  Serial.println("Conectando");
  while(WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.print("Connected to WiFi network with IP Address: ");
  Serial.println(WiFi.localIP());
```

```
  SPI.begin();                  // Iniciando o bus SPI - Usado para comunicação entre
      o MFRC522 e a ESP

}

void loop() {

  HTTPClient http;
  http.begin(Linhas);
  http.addHeader("Content-Type", "application/x-www-form-urlencoded");
  http.GET();
  String linha = http.getString();
  http.end();

  cartaouid = "";
  TesteMestre = "";
  comando = "";
  httpRequestData = "";
  TesteCartao = "";
  for (int i = 1; i <= 6; i++)
  {
    sala[i] = "";
  }

  mfrc522.PCD_Init();
  MFRC522::MIFARE_Key key;
  MFRC522::StatusCode status;
  for (byte i = 0; i < 6; i++) key.keyByte[i] = 0xFF;

  if (linha == "0")
  {

    int erro = 0;
    String nada;

    Serial.println("Não há um Cartão Mestre\nColoque um cartão no leitor para
        cadastrar\n");
```

```
  while ( ! mfrc522.PICC_IsNewCardPresent()) {
  }

  while ( ! mfrc522.PICC_ReadCardSerial()) {
  }

  Serial.print(F("Card UID:"));
  for (byte i = 0; i < mfrc522.uid.size; i++) {
      cartaouid += String(mfrc522.uid.uidByte[i], HEX) + " ";
  }
  cartaouid.trim();
  cartaouid.toUpperCase();
  Serial.print("\n");
  Serial.println(cartaouid);

}
else

{

  Serial.println("Aproxime um cartão Mestre para abrir o menu.");

  while ( ! mfrc522.PICC_IsNewCardPresent()) {}

  while ( ! mfrc522.PICC_ReadCardSerial()) {}

  Serial.print(F("Card UID:")); //Dump UID
  for (byte i = 0; i < mfrc522.uid.size; i++) {
      cartaouid += String(mfrc522.uid.uidByte[i], HEX) + " ";
  }
  cartaouid.trim();
  cartaouid.toUpperCase();
  Serial.print("\n");
  Serial.println(cartaouid);

  http.begin(Teste);

  http.addHeader("Content-Type", "application/x-www-form-urlencoded");
```

```
http.POST("&UID=" + cartaouid + "&operacao=" + "1");
TesteMestre = http.getString();
TesteMestre.trim();

http.end();

if(cartaouid == TesteMestre)
{
  int erro = 0;
  Serial.print("\nM para cadastrar um novo Cartão Mestre\nU para cadastrar um novo
      Cartão de Usuário\nS para sair\n");
  while (Serial.available() == 0)
  {
    delay(500);
    erro ++;
    if (erro > 120)
    {
      Serial.println("\nTempo esgotado!\nFechando o menu...!\n");
      delay(2000);
      return;
    }
  }
  comando = Serial.readString();
  comando.trim();
  while (comando != "U" & comando != "S" & comando != "M")
  {
    Serial.println("Erro!!");
    while (Serial.available() == 0) {}
    comando = Serial.readString();
    comando.trim();
  }
  if(comando == "S")
  {
    Serial.println("Fechando o menu...\n");
    delay(2000);
    return;
  }
  Serial.print("\n");
  delay(1000);
```

```
  }
  else
  {
    Serial.println("Esse não é um cartão Mestre!!");
    Serial.println("Tente novamente");
    delay(2000);
    return;
  }

}

if (comando == "M" | comando == "U")
{
  cartaouid = "";
  int tempo1 = 0, tempo2 = 0;

  Serial.println("Coloque um novo cartão no leitor para cadastrar");

  while ( ! mfrc522.PICC_IsNewCardPresent())
  {
    delay(1000);
    tempo1++;
    if (tempo1 == 30)
    {
      Serial.println("\nTempo esgotado!\nEncerrando o cadastro!\n");
      delay(2000);
      return;
    }
  }

  while ( ! mfrc522.PICC_ReadCardSerial())
  {
    delay(1000);
    tempo2++;
    if (tempo2 == 30)
    {
      Serial.println("\nTempo esgotado!\nEncerrando o cadastro!\n");
      delay(2000);
      return;
```

```
    }
  }

  Serial.print("\n");
  Serial.print(F("Card UID:")); //Dump UID
  for (byte i = 0; i < mfrc522.uid.size; i++) {
      cartaouid += String(mfrc522.uid.uidByte[i], HEX) + " ";
  }
  cartaouid.trim();
  cartaouid.toUpperCase();
  Serial.print("\n");
  Serial.println(cartaouid);

  http.begin(Teste);

  http.addHeader("Content-Type", "application/x-www-form-urlencoded");

  http.POST("&UID=" + cartaouid + "&operacao=" + "2");
  TesteCartao = http.getString();
  TesteMestre.trim();

  http.end();

  if (cartaouid == TesteCartao)
  {
    Serial.println("Cartão já cadastrado");
    Serial.println("Encerrado o cadastro");
    delay(2000);
    return;
  }

}

////////////// NOME //////////////

String nome;
int tempo = 0;

Serial.println(F("Nome:"));
```

```
while (Serial.available() == 0)          // Esperando pela entrada
{
  delay(1000);
  tempo++;
  if (tempo == 30)
  {
    Serial.println("\nTempo esgotado!\nEncerrando o cadastro!");
    delay(2000);
    return;
  }
}
nome = Serial.readString();              // Lendo a entrada
Serial.println(nome);

////////////// MATRICULA //////////////

String matricula;
tempo = 0;

Serial.println(F("Matricula:"));
while (Serial.available() == 0)
{
  delay(1000);
  tempo++;
  if (tempo == 30)
  {
    Serial.println("\nTempo esgotado!\nEncerrando o cadastro!");
    delay(2000);
    return;
  }
}
matricula = Serial.readString();
Serial.println(matricula);

////////////// TELEFONE //////////////

String telefone;
tempo = 0;
```

```
Serial.println(F("Telefone:"));
while (Serial.available() == 0)
{
  delay(1000);
  tempo++;
  if (tempo == 30)
  {
    Serial.println("\nTempo esgotado!\nEncerrando o cadastro!");
    delay(2000);
    return;
  }
}
telefone = Serial.readString();
Serial.println(telefone);

////////////// CPF //////////////

String cpf;
tempo = 0;

Serial.println(F("CPF:"));
while (Serial.available() == 0)
{
  delay(1000);
  tempo++;
  if (tempo == 30)
  {
    Serial.println("\nTempo esgotado!\nEncerrando o cadastro!");
    delay(2000);
    return;
  }
}
cpf = Serial.readString();
Serial.println(cpf);

////////////// EMAIL //////////////

String email;
tempo = 0;
```

```
Serial.println(F("Email:"));
while (Serial.available() == 0)
{
  delay(1000);
  tempo++;
  if (tempo == 30)
  {
    Serial.println("\nTempo esgotado!\nEncerrando o cadastro!");
    delay(2000);
    return;
  }
}
email = Serial.readString();
Serial.println(email);

if (comando == "M" | linha == "0")
{

  String acesso = "limc123";
  byte bacesso[acesso.length()+1];
  acesso.getBytes(bacesso, acesso.length()+1);

  for (byte i = 7; i < 16; i++) bacesso[i] = ' '; // pad with spaces

  byte block;
  block = 60;

  status = mfrc522.PCD_Authenticate(MFRC522::PICC_CMD_MF_AUTH_KEY_A, block, &key,
      &(mfrc522.uid));
  if (status != MFRC522::STATUS_OK) {
    Serial.print(F("PCD_Authenticate() failed: "));
    Serial.println(mfrc522.GetStatusCodeName(status));
    return;
  }

  // Write block
  status = mfrc522.MIFARE_Write(block, bacesso, 16);
  if (status != MFRC522::STATUS_OK) {
```

```
    Serial.print(F("MIFARE_Write() failed: "));
    Serial.println(mfrc522.GetStatusCodeName(status));
    return;
  }
  else
  {
    Serial.println("Os dados foram gravados!");
  }

}
else if (comando == "U")
{

  ////////////// SALAS //////////////

  int indice = 1;
  tempo = 0;

  for (indice = 1; indice <= 6; indice++)
  {

    Serial.print("Conceder acesso a Sala ");
    Serial.print(indice);
    Serial.print("?(S/N)\n");

    while (Serial.available() == 0)
    {
      delay(1000);
      ++tempo;
      if (tempo == 30)
      {
        Serial.println("\nTempo esgotado!\nEncerrando o cadastro!");
        delay(2000);
        return;
      }
    }
    tempo = 0;
    sala[indice] = Serial.readString();
    sala[indice].trim();
```

```
    while (sala[indice] != "S" & sala[indice] != "N" & tempo < 30)
    {

      Serial.println("Comando inválido");
      while (Serial.available() == 0)
      {
        tempo++;
        delay(1000);
        if (tempo == 30)
        {
          Serial.println("\nTempo esgotado!\nEncerrando o cadastro!");
          delay(2000);
          return;
        }
      }
      sala[indice] = Serial.readString();
      sala[indice].trim();

    }

  }

}

http.begin(Cadastro);

http.addHeader("Content-Type", "application/x-www-form-urlencoded");

if (comando == "M")
{
  httpRequestData = "&UID=" + cartaouid + "&nome=" + nome + "&matricula=" +
      matricula + "&telefone="
                  + telefone + "&cpf=" + cpf + "&email=" + email + "&comando=" +
                      comando + "";
}
else if (linha == "0")
{
  httpRequestData = "&UID=" + cartaouid + "&nome=" + nome + "&matricula=" +
      matricula + "&telefone="
```

```
                    + telefone + "&cpf=" + cpf + "&email=" + email + "&comando=" +
                        linha + "";
  }
  else if (comando == "U")
  {
    httpRequestData = "&UID=" + cartaouid + "&nome=" + nome + "&matricula=" +
        matricula + "&telefone="
                    + telefone + "&cpf=" + cpf + "&email=" + email + "&comando=" +
                        comando
                    + "&sala1=" + sala[1] + "&sala2=" + sala[2] + "&sala3=" + sala[3]
                    + "&sala4=" + sala[4] + "&sala5=" + sala[5] + "&sala6=" + sala[6]
                        + "";
  }

  httpResponseCode = http.POST(httpRequestData);
  String respostaServidor = http.getString();
  http.end();

  if (httpResponseCode > 0)
  {
    Serial.println("Cadastro realizado com sucesso");
    delay(2000);
  }

}
```

ANEXO B - CÓDIGO DE ACESSO

```
#include <WiFi.h>
#include <ESPmDNS.h>
#include <WiFiUdp.h>
#include <ArduinoOTA.h>
#include <SPI.h>
#include <MFRC522.h>
#include <WiFiClient.h>
#include <HTTPClient.h>
#include "pitches.h"

#define RST_PIN      4
#define SS_PIN       21
#define GLED_PIN     27 // 14
#define RLED_PIN     14 // 12
#define BLED_PIN     26 // 27
#define RELAY_PIN    13
#define BUZ_PIN      15

#define LED_ON HIGH
#define LED_OFF LOW

// Credenciais da sua rede
const char* ssid = "";
const char* password = "";

// Nome do Domínio e URL ou endereço de IP com o caminho para os arquivos
//const char* Acesso = "";
const char* Acesso = "http://meu_ip/acesso.php";

String Sala = "Sala 1";           // Sala em que a ESP sera usada
String apiKeyValue = ""; // Código para permitir a conexão com o banco de dados
String codigo = "";         // Senha para permitir acesso de um cartão mestre mesmo
    com erro de conexão
String operacao = "1";
int tempoCon = 0;

MFRC522 mfrc522(SS_PIN, RST_PIN); // Iniciando o leitor
```

```
// Função para controlar o Buzzer
void beep(int note, int duration) {
  tone(BUZ_PIN, note, duration);
  delay(duration);
}

void setup() {

  //digitalWrite(BLED_PIN, LED_ON); // LED acende se a ESP estiver conectada a fonte
      de alimentação

  Serial.begin(9600);
  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);
  Serial.println("Conectando");
  while (WiFi.status() != WL_CONNECTED | tempoCon < 20) {
    delay(500);
    Serial.print(".");
    tempoCon ++;
  }

  ArduinoOTA
    .onStart([]() {
      String type;
      if (ArduinoOTA.getCommand() == U_FLASH)
        type = "sketch";
      else // U_SPIFFS
        type = "filesystem";

      Serial.println("Iniciando Upload " + type);
    })
    .onEnd([]() {
      Serial.println("\nAtualização Completa");
      Serial.println("\nReinicie o Dispositivo");
    })
    .onProgress([](unsigned int progress, unsigned int total) {
      Serial.printf("Progresso: %u%%\r", (progress / (total / 100)));
    })
```

```
    .onError([](ota_error_t error) {
      Serial.printf("Erro[%u]: ", error);
      if (error == OTA_AUTH_ERROR) Serial.println("Falha na Autenticação");
      else if (error == OTA_BEGIN_ERROR) Serial.println("Falha na Inicialização");
      else if (error == OTA_CONNECT_ERROR) Serial.println("Falha na Conexão");
      else if (error == OTA_RECEIVE_ERROR) Serial.println("Falha na Recepção");
      else if (error == OTA_END_ERROR) Serial.println("Falha na Finalização");
    });

  ArduinoOTA.begin();

  Serial.println("\nPronto");
  Serial.print("Conectado a rede Wi-Fi com o IP: ");
  Serial.println(WiFi.localIP());

  pinMode(GLED_PIN, OUTPUT);
  pinMode(RLED_PIN, OUTPUT);
  pinMode(BLED_PIN, OUTPUT);
  pinMode(RELAY_PIN, OUTPUT);
  pinMode(BUZ_PIN, OUTPUT);
  //digitalWrite(GLED_PIN, LED_OFF); // Garantindo que os LEDs estão apagados na
      inicialização da ESP
  //digitalWrite(RLED_PIN, LED_OFF);
  digitalWrite(RELAY_PIN, LOW); // Garantindo que a porta esteja fechada na
      inicialização
  setColor(0, 0, 255); // Blue

  SPI.begin();                // Iniciando o bus SPI - Usado para comunicação entre
      o MFRC522 e a ESP

}

void setColor(int vermelho, int verde, int azul){

  analogWrite(RLED_PIN, vermelho);
  analogWrite(GLED_PIN, verde);
  analogWrite(BLED_PIN, azul);
}
```

```
void loop() {
  ArduinoOTA.handle();

  //digitalWrite(BLED_PIN, LED_ON);
  setColor(0, 0, 255); // Azul

  tempoCon = 0;
  if (WiFi.status() != WL_CONNECTED)
  {
    WiFi.begin(ssid, password);
    while (WiFi.status() != WL_CONNECTED | tempoCon < 10)
    {
      delay(500);
      Serial.print(".");
      tempoCon ++;
    }
  }

  //digitalWrite(GLED_PIN, LED_OFF);
  //digitalWrite(RLED_PIN, LED_OFF);
  digitalWrite(RELAY_PIN, LOW);

  mfrc522.PCD_Init();                              // Iniciando o MFRC522

  MFRC522::MIFARE_Key key;
  for (byte i = 0; i < 6; i++) key.keyByte[i] = 0xFF;

  byte block;
  byte len;
  MFRC522::StatusCode status;

  // Reseta o loop se não houver um novo cartão no sensor.
  if ( ! mfrc522.PICC_IsNewCardPresent()) {
    return;
  }

  if ( ! mfrc522.PICC_ReadCardSerial()) {
    return;
  }
```

```
// Lê o UID do cartão

setColor(255, 255, 0); // Amarelo

String cartaouid;
Serial.print("\n");
Serial.print(F("Card UID:"));
for (byte i = 0; i < mfrc522.uid.size; i++) {
  cartaouid += String(mfrc522.uid.uidByte[i], HEX) + " ";
}
cartaouid.trim();
cartaouid.toUpperCase();
Serial.print("\n");
Serial.print(cartaouid);

// Ler a Senha no cartão

byte buffer[18];

block = 60;
len = 18;

status = mfrc522.PCD_Authenticate(MFRC522::PICC_CMD_MF_AUTH_KEY_A, block, &key,
    &(mfrc522.uid));
if (status != MFRC522::STATUS_OK) {
  Serial.print(F("Authentication failed: "));
  Serial.println(mfrc522.GetStatusCodeName(status));
  return;
}

status = mfrc522.MIFARE_Read(block, buffer, &len);
if (status != MFRC522::STATUS_OK) {
  Serial.print(F("Reading failed: "));
  Serial.println(mfrc522.GetStatusCodeName(status));
  return;
}

int s1 = 0;
```

```
String senha;
for (uint8_t i = 0; i < 16; i++)
{
  if (buffer[i] == 32 & buffer[i + 1] == 32)
  {
    s1++;
  }
  else if (s1 > 0) {}
  else
  {
    senha += (char)buffer[i];
  }
}
senha.trim();

Serial.print("\n");

String respostaServidor, a , b;
int httpCodigo = 0;
HTTPClient http;

if (WiFi.status() == WL_CONNECTED)
{

  // Testando se o UID do cartão tem permissão para o acesso

  http.begin(Acesso);

  http.addHeader("Content-Type", "application/x-www-form-urlencoded");

  httpCodigo = http.POST("&operacao=" + operacao + "&UID=" + cartaouid + "&sala=" +
      Sala); // Envia a requisição para o servidor
  respostaServidor = http.getString(); // Recebe a resposta do servidor

  http.end();

  Serial.println(respostaServidor);
  Serial.println(httpCodigo);
```

```
if (httpCodigo < 0)
{
  if (senha == codigo)
  {
    Serial.println("Acesso Permitido");
    digitalWrite(RELAY_PIN, HIGH);
    setColor(0, 255, 0); // Verde
    beep(NOTE_C7, 750);
    delay(2000);
    digitalWrite(RELAY_PIN, LOW);
    return;
  }
  else
  {
    Serial.println("Acesso Negado");
    setColor(255, 0, 0); // Vermelho
    beep(NOTE_A5, 200);
    delay(200);
    beep(NOTE_A5, 200);
    delay(200);
    beep(NOTE_A5, 200);
    delay(1000);
    return;
  }
}
else if (respostaServidor == "1")
{
  Serial.println("Acesso Permitido");
  digitalWrite(RELAY_PIN, HIGH);
  setColor(0, 255, 0); // Verde
  beep(NOTE_C7, 750);
  delay(2000);
  digitalWrite(RELAY_PIN, LOW);
  return;
}
else
{
  Serial.println("Acesso Negado");
  setColor(255, 0, 0); // Vermelho
```

```
      beep(NOTE_A5, 200);
      delay(200);
      beep(NOTE_A5, 200);
      delay(200);
      beep(NOTE_A5, 200);
      delay(1000);
      return;
    }

  }

}
```

ANEXO C - linhas.php

```php
<?php
include 'database.php';

// Subsituir por suas credenciais
$con = mysqli_connect("localhost", "usuário", "senha", "base de dados");

    $sql = "SELECT * from mestre";

    if ($result = mysqli_query($con, $sql)) {

    $rowcount = mysqli_num_rows( $result );
   echo $rowcount;
}

mysqli_close($con);

?>
```

ANEXO D - teste.php

```
<?php
include 'database.php';

   $uid= $uidm= $operacao= "";

   if (!empty($_POST)) {
      $uidm = $_POST["UID"];
      $uid = $_POST["UID"];
      $operacao = $_POST["operacao"];
      $pdo = Database::connect();
      $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
      $sql = "SELECT * FROM mestre WHERE UID = '$uidm'";

      $q = $pdo->prepare($sql);
      $q->execute(array($uidm));
      $data = $q->fetch(PDO::FETCH_ASSOC);

      $uidm = $data['UID'];
      if (!empty($uidm)){
         if ($operacao == "1" | $operacao == "2"){
            echo $uidm;
         }
      }
      else if ($operacao == "2"){
         $sql = "SELECT * FROM usuarios WHERE UID = '$uid'";

         $q = $pdo->prepare($sql);
         $q->execute(array($uid));
         $data = $q->fetch(PDO::FETCH_ASSOC);

         $uid = $data['UID'];
         if (!empty($uid)){
            echo $uid;
         }
      }
   }
?>
```

ANEXO E - cadastro.php

```
<?php
include 'database.php';

   $uid= $nome= $matricula= $telefone= $cpf= $email= $comando= "";
   $sala1= $sala2= $sala3= $sala4= $sala5= $sala6 = "";
   $uid = $_POST["UID"];
   $comando = $_POST["comando"];
   $nome = $_POST["nome"];
   $matricula = $_POST["matricula"];
   $telefone = $_POST["telefone"];
   $cpf = $_POST["cpf"];
   $email = $_POST["email"];

   if (!empty($_POST)) {
      $pdo = Database::connect();
      $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

      if ($comando == "0" | $comando == "M"){
         $sql = "INSERT INTO mestre (UID, Nome, Matricula, Telefone, CPF, Email)
         VALUES ('" . $uid . "', '" . $nome . "', '" . $matricula . "', '" . $telefone
             . "', '" . $cpf . "', '" . $email . "')";
         $pdo->query($sql);
         Database::disconnect();
         echo "Cartão Cadastrado";
      }
      else if ($comando == "U"){
         $sala1= $_POST["sala1"];
         $sala2= $_POST["sala2"];
         $sala3= $_POST["sala3"];
         $sala4= $_POST["sala4"];
         $sala5= $_POST["sala5"];
         $sala6= $_POST["sala6"];
         $sql = "INSERT INTO usuarios (UID, Nome, Matricula, Telefone, CPF, Email,
             Sala_1, Sala_2, Sala_3, Sala_4, Sala_5, Sala_6)
         VALUES ('" . $uid . "', '" . $nome . "', '" . $matricula . "', '" . $telefone
             . "', '" . $cpf . "', '" . $email . "', '" . $sala1 . "', '" . $sala2 .
             "', '" . $sala3 . "', '" . $sala4 . "', '" . $sala5 . "', '" . $sala6 .
```

```
            "')";
        $pdo->query($sql);
        Database::disconnect();
        echo "Cartão Cadastrado";
    }
  }
?>
```

ANEXO F - acesso.php

```
<?php
include 'database.php';

   $sala= $uid= $uidm= $nome= $matricula= $telefone= $cpf= $email= $operacao= "";
   $sala1= $sala2= $sala3= $sala4= $sala5= $sala6 = "";

   if (!empty($_POST)) {
      $uidm = $_POST["UID"];
      $uid = $_POST["UID"];
      $sala = $_POST["sala"];
      $operacao = $_POST["operacao"];
      $pdo = Database::connect();
      $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
      $sql = "SELECT * FROM mestre WHERE UID = '$uidm'";

      $q = $pdo->prepare($sql);
      $q->execute(array($uidm));
      $data = $q->fetch(PDO::FETCH_ASSOC);

      $uidm = $data['UID'];
      if (!empty($uidm)){
         $nome = $data['Nome'];
         $matricula = $data['Matricula'];
         $telefone = $data['Telefone'];
         $cpf = $data['CPF'];
         $email = $data['Email'];
         $sql = "INSERT INTO acessos (Sala, UID, Nome, Matricula, Telefone, CPF, Email)
         VALUES ('" . $sala . "','" . $uidm . "', '" . $nome . "', '" . $matricula .
             "', '" . $telefone . "', '" . $cpf . "', '" . $email . "')";
         $pdo->query($sql);
         Database::disconnect();
         if ($operacao == "1"){
         echo "1";
         }
         else if ($operacao == "2"){
         echo "1_", $nome;
         }
```

```
}
else{

   $sql = "SELECT * FROM usuarios WHERE UID = '$uid'";

   $q = $pdo->prepare($sql);
   $q->execute(array($uid));
   $data = $q->fetch(PDO::FETCH_ASSOC);

   $uid = $data['UID'];
   if (!empty($uid)){
      $nome = $data['Nome'];
      $matricula = $data['Matricula'];
      $telefone = $data['Telefone'];
      $cpf = $data['CPF'];
      $email = $data['Email'];
      $sala1 = $data['Sala_1'];
      $sala2 = $data['Sala_2'];
      $sala3 = $data['Sala_3'];
      $sala4 = $data['Sala_4'];
      $sala5 = $data['Sala_5'];
      $sala6 = $data['Sala_6'];
      if ($sala == "Porta 2"){
         $sql = "INSERT INTO acessos (Sala, UID, Nome, Matricula, Telefone, CPF,
             Email)
         VALUES ('" . $sala . "','" . $uid . "', '" . $nome . "', '" . $matricula
             . "', '" . $telefone . "', '" . $cpf . "', '" . $email . "')";
         $pdo->query($sql);
         Database::disconnect();
         echo "1_", $nome;
      }
      else if ($sala == "Sala 1" & $sala1 == "S"){
         $sql = "INSERT INTO acessos (Sala, UID, Nome, Matricula, Telefone, CPF,
             Email)
         VALUES ('" . $sala . "','" . $uid . "', '" . $nome . "', '" . $matricula
             . "', '" . $telefone . "', '" . $cpf . "', '" . $email . "')";
         $pdo->query($sql);
         Database::disconnect();
         echo "1";
```

```
}
else if($sala == "Sala 2" & $sala2 == "S"){
   $sql = "INSERT INTO acessos (Sala, UID, Nome, Matricula, Telefone, CPF,
       Email)
   VALUES ('" . $sala . "','" . $uid . "', '" . $nome . "', '" . $matricula
       . "', '" . $telefone . "', '" . $cpf . "', '" . $email . "')";
   $pdo->query($sql);
   Database::disconnect();
   echo "1";
}
else if($sala == "Sala 3" & $sala3 == "S"){
   $sql = "INSERT INTO acessos (Sala, UID, Nome, Matricula, Telefone, CPF,
       Email)
   VALUES ('" . $sala . "','" . $uid . "', '" . $nome . "', '" . $matricula
       . "', '" . $telefone . "', '" . $cpf . "', '" . $email . "')";
   $pdo->query($sql);
   Database::disconnect();
   echo "1";
}
else if($sala == "Sala 4" & $sala4 == "S"){
   $sql = "INSERT INTO acessos (Sala, UID, Nome, Matricula, Telefone, CPF,
       Email)
   VALUES ('" . $sala . "','" . $uid . "', '" . $nome . "', '" . $matricula
       . "', '" . $telefone . "', '" . $cpf . "', '" . $email . "')";
   $pdo->query($sql);
   Database::disconnect();
   echo "1";
}
else if($sala == "Sala 5" & $sala5 == "S"){
   $sql = "INSERT INTO acessos (Sala, UID, Nome, Matricula, Telefone, CPF,
       Email)
   VALUES ('" . $sala . "','" . $uid . "', '" . $nome . "', '" . $matricula
       . "', '" . $telefone . "', '" . $cpf . "', '" . $email . "')";
   $pdo->query($sql);
   Database::disconnect();
   echo "1";
}
else if($sala == "Sala 6" & $sala6 == "S"){
   $sql = "INSERT INTO acessos (Sala, UID, Nome, Matricula, Telefone, CPF,
```

```
                Email)
            VALUES ('" . $sala . "','" . $uid . "', '" . $nome . "', '" . $matricula
                . "', '" . $telefone . "', '" . $cpf . "', '" . $email . "')";
            $pdo->query($sql);
            Database::disconnect();
            echo "1";
          }
          else{
            Database::disconnect();
            echo "0";
          }
        }
        else{
          Database::disconnect();
          echo "0";
        }

    }
  }
?>
```

ANEXO G - database.php

```
<?php
   class Database {
      private static $dbName = '' ;
      private static $dbHost = '' ;
      private static $dbUsername = '';
      private static $dbUserPassword = '';

      private static $cont = null;

      public function __construct() {
         die('Init function is not allowed');
      }

      public static function connect() {
        // One connection through whole application
        if ( null == self::$cont ) {
        try {
          self::$cont = new PDO(
              "mysql:host=".self::$dbHost.";"."dbname=".self::$dbName,
              self::$dbUsername, self::$dbUserPassword);
        }
        catch(PDOException $e) {
          die($e->getMessage());
        }
        }
        return self::$cont;
      }

      public static function disconnect() {
         self::$cont = null;
      }
   }
?>
```