Título do Projeto de Ensino:

Dispositivo de Controle de Acesso aos Laboratórios

Equipe responsável:

Coordenador:

Prof. Maurício Khenaifes
Lotação: COELT-AP

Colaborador:

Prof. Thales Eugênio Portes de Almeida
Lotação: COELT-AP

Estudante Colaborador:

Silas Torres Rocha
RA: 2104644

Universidade Tecnológica Federal do Paraná
Apucarana - PR
Setembro de 2024

Sumário

1 Descrição da atividade desenvolvida **3**
1.1 Contextualização . . . 3
1.2 Objetivo . . . 3
1.3 Arquitetura do Sistema . . . 3
1.4 Hardware e Conexões . . . 9
1.5 Modelo 3D . . . 9
1.6 Funcionamento . . . 10

2 Número de participantes: **12**

3 Resultados do projeto: **12**
3.1 Funcionamento Geral . . . 12
3.2 Comunicação com a Planilha . . . 12
3.2.1 Usuários . . . 13
3.2.2 Entrada/Saída . . . 13
3.2.3 Entrada/Saída (offline) . . . 13
3.2.4 Horários . . . 13
3.2.5 INF . . . 13
3.3 Padrões de Retorno da Planilha . . . 14
3.4 Padrões de Dados da Planilha . . . 14
3.5 Interface Gráfica em Python . . . 16
3.5.1 Comunicação: . . . 16
3.5.2 Interface . . . 17
3.5.3 Usuários . . . 18
3.5.4 Entrada/Saída . . . 18
3.5.5 Horários . . . 18
3.6 Descrição Geral dos Resultados . . . 18
3.7 Análise de Desempenho . . . 19
3.8 Sincronização e Armazenamento de Dados . . . 19
3.9 Controle de Acesso . . . 19
3.10 Testes de Segurança . . . 19
3.11 Resumo dos Resultados . . . 19
3.12 Contribuições do Projeto . . . 21
3.13 Limitações do Projeto . . . 21
3.14 Trabalhos Futuros . . . 21
3.15 Considerações Finais . . . 22
3.16 Custo de fabricação estimado: . . . 22

4 Referências **23**

Resumo

Este projeto tem como objetivo desenvolver um sistema de controle de acesso que registra entradas e saídas, armazenando dados sobre quem transitou e os horários correspondentes. O sistema é implementado utilizando um ESP8266, que gerencia a abertura de uma porta através de um leitor de TAGs RFID e controla uma tranca eletrônica. Cada evento de acesso é registrado automaticamente em uma planilha do Google Sheets, que armazena os horários de entrada e saída, bem como a identificação dos usuários autorizados.

O dispositivo também foi projetado para funcionar de forma robusta, com um mecanismo de armazenamento local para registrar eventos quando a conexão com a internet não estiver disponível. Assim que a conectividade é restaurada, os dados armazenados localmente são enviados para a planilha. O projeto inclui ainda uma interface gráfica desenvolvida em Python, que facilita a gestão e edição dos dados armazenados na planilha, bem como um código App Script que serve como ponte entre o ESP8266 e o Google Sheets, garantindo a comunicação eficiente entre o hardware e o banco de dados na nuvem.

1 Descrição da atividade desenvolvida

1.1 Contextualização

Em ambientes acadêmicos, como universidades, a gestão do fluxo de pessoas em laboratórios é de extrema importância. Garantir que apenas alunos e funcionários autorizados tenham acesso a esses espaços é crucial para proteger equipamentos valiosos, materiais de pesquisa e garantir a segurança dos usuários. Além disso, o controle rigoroso e auditável das entradas e saídas permite monitorar a utilização dos laboratórios, identificar padrões de uso e melhorar a gestão do espaço. Com um sistema automatizado de controle de acesso, a universidade pode não apenas aumentar a segurança, mas também otimizar a utilização dos laboratórios, garantindo que os recursos estejam disponíveis para quem realmente precisa.

1.2 Objetivo

O objetivo deste projeto é melhorar a segurança e a gestão dos laboratórios da UTFPR através de um sistema automatizado de controle de acesso. Utilizando um dispositivo baseado no ESP8266, o projeto visa controlar a entrada e saída de pessoas no laboratório, permitindo o acesso apenas a indivíduos autorizados. Além disso, o sistema registra automaticamente cada evento em uma planilha do Google Sheets, possibilitando a auditoria completa do uso do laboratório. Com isso, a UTFPR pode assegurar que os laboratórios sejam utilizados de forma segura e eficiente, protegendo seus recursos e facilitando o monitoramento em tempo real e permitindo acesso nos horários livres e restringindo durante o período de aulas.

Posteriormente, será disponibilizado a todos os professores responsáveis pelos laboratórios. Para que isso seja possível, eles terão suporte e o passo-a-passo o qual deverão seguir.

1.3 Arquitetura do Sistema

- 1 Esp8266.
- 1 Placa Universal (3 x 7 cm).
- 1 Tranca Solenoide 12V.
- 2 Display LCD 1602.
- 2 Módulos I2C, baseados no PCF8574.

- 2 Módulos RFID, baseados no MFRC522.
- 1 Buzzer.
- 1 Microchave Push Button 6mm.
- 1 AMS1117 5V. *
- 1 AMS1117 3.3V.*
- 2 Transistor BC337.
- 2 Resistores 10KΩ.
- 1 Resistor 1.8KΩ.
- 1 Resistor 160Ω.
- 1 Resistor 250Ω.
- 1 Fonte 12V.

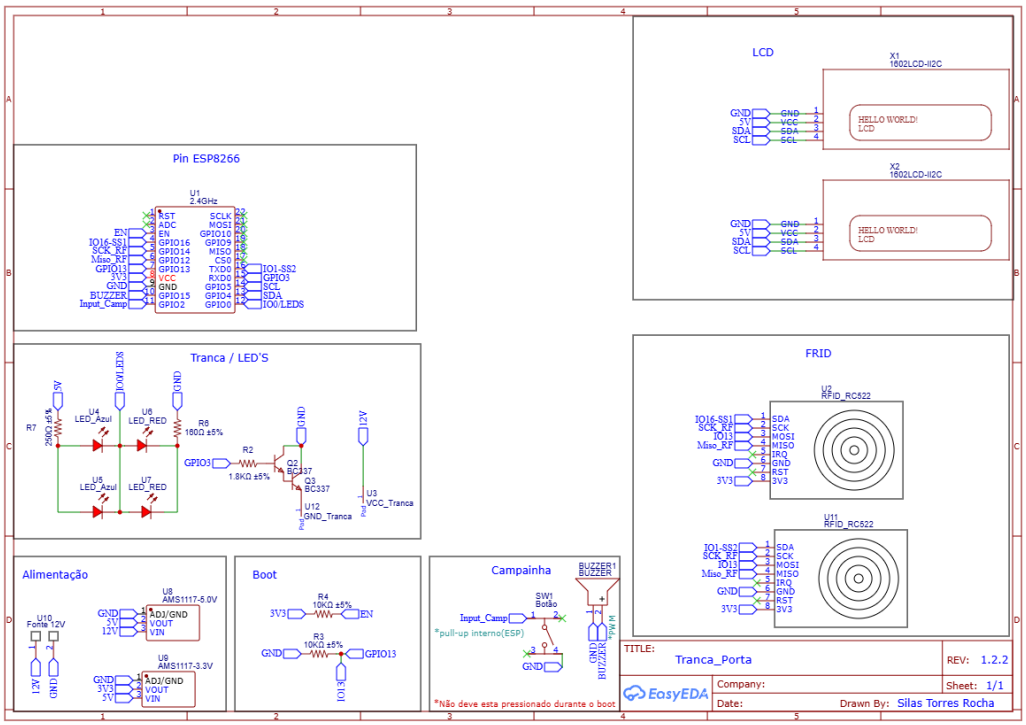


Figura 1: Esquema elétrico

*Não se utilizou um capacitor nos pinos de GND e input, o que pode resultar em maior ripple e ruído na saída.

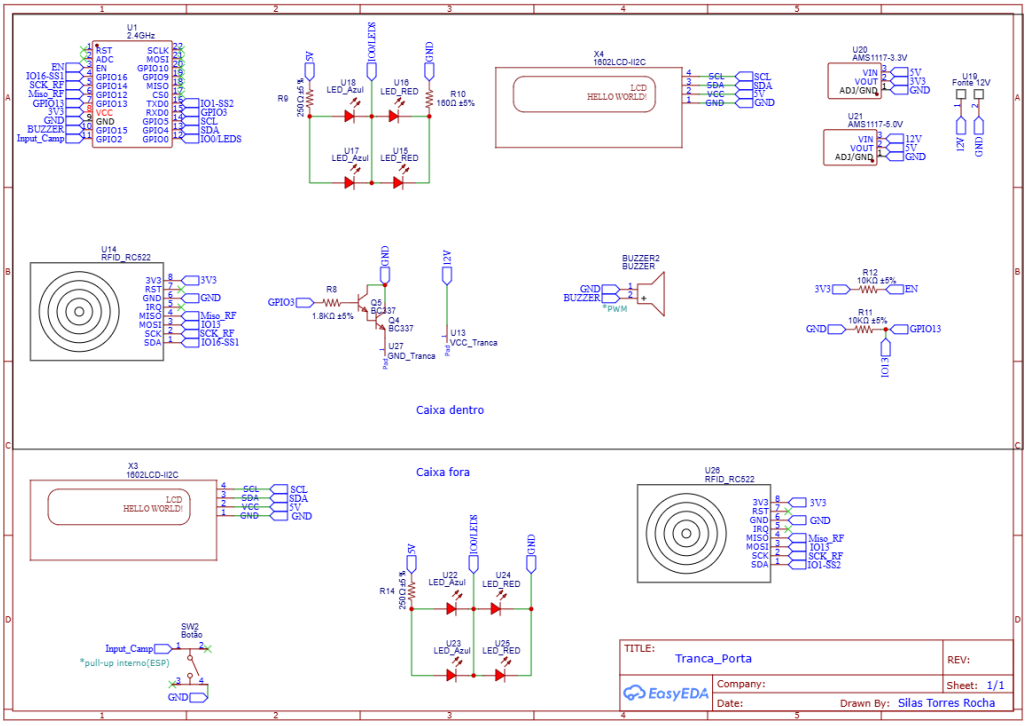


Figura 2: Esquema elétrico separado

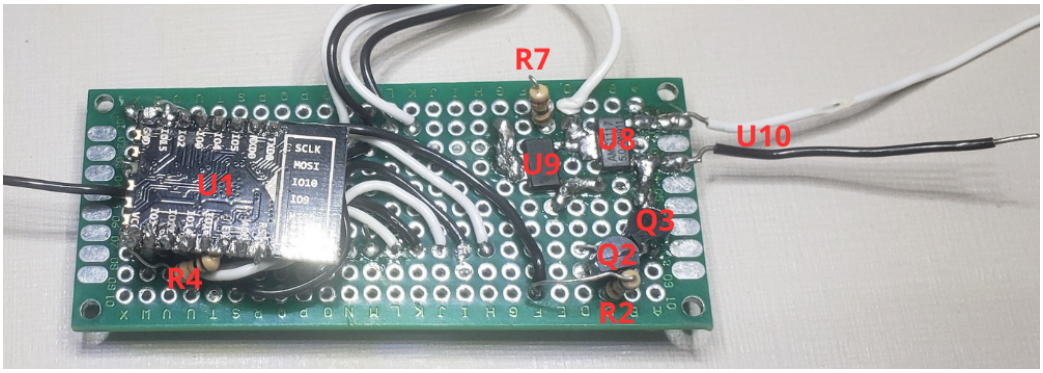


Figura 3: Foto com componentes posicionados

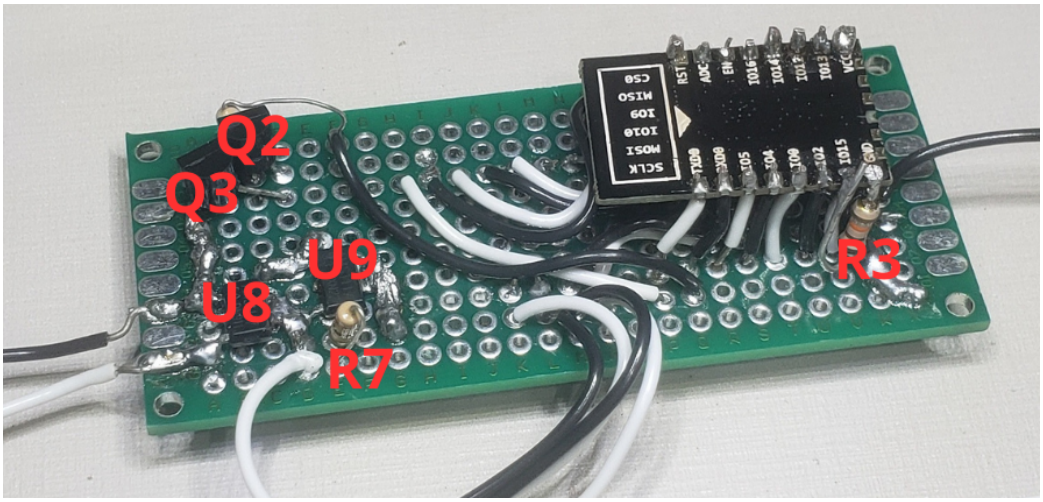


Figura 4: Foto com componentes posicionados

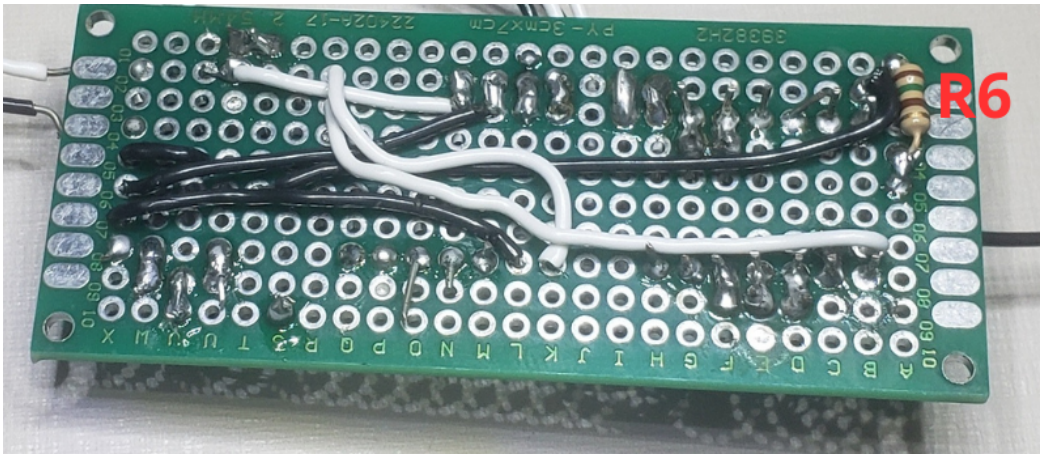


Figura 5: Foto com componentes posicionados, parte inferior

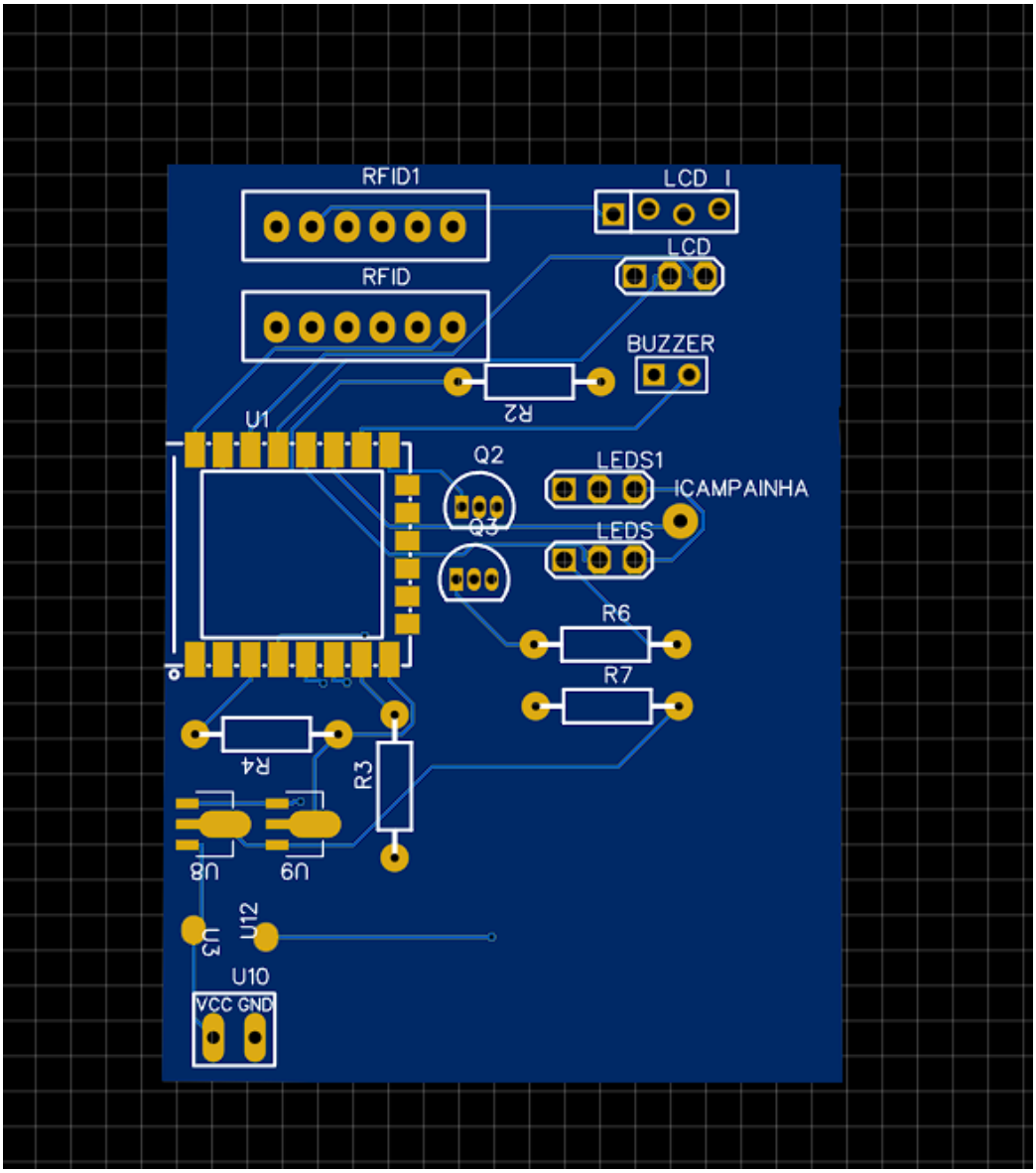


Figura 6: Imagem PCB 2D

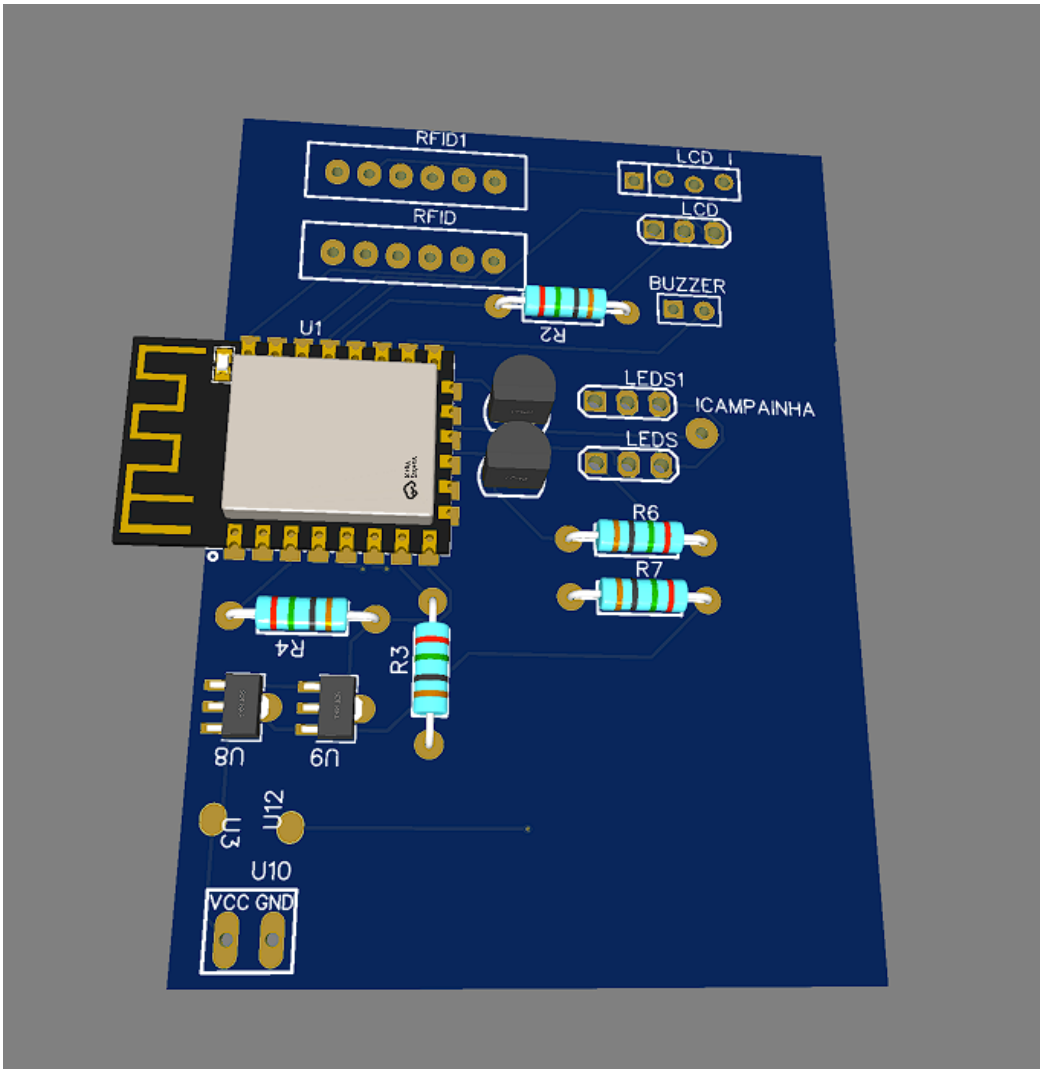


Figura 7: Imagem PCB 3D

1.4 Hardware e Conexões

Pino ESP	Pino componente	Característica
GPIO 0	"Comum"com os LED's	I/O
GPIO 1	Chip Select RFID Entrada	I/O
GPIO 2	Botão campainha	Pull-up interno
GPIO 3	Tranca	I/O
GPIO 4	SDA(LCD)	SDA
GPIO 5	SCL(LCD)	SCL
GPIO 12	MISO(RFID)	MISO
GPIO 13	MOSI(RFID)	MOSI
GPIO 14	SCK(RFID)	SCK
GPIO 15	+ Buzzer	PWM
GPIO 16	Chip Select RFID Saida	I/O

Tabela 1: Pinos ESP8266. [1]

É importante estar atento ao estado dos pinos abaixo durante o boot, pois eles definem o modo de inicialização do ESP8266.

Pino ESP	Modo bootloader	Modo execução
GPIO 0	LOW	HIGH
GPIO 2	HIGH	HIGH
GPIO 15	LOW	LOW

Tabela 2: Modos de inicialização. [2]
**O ESP possui outro modos de boot.*

A campainha não deve ser pressionada durante a inicialização; caso contrário, não entrará na inicialização correta do ESP.

1.5 Modelo 3D

Para acomodar o sistema, foi usado um modelo 3D que está disponível no Thingiverse sob a licença CC BY-NC. Este modelo foi criado por amamonem [3]. Devido à necessidade de mais espaço para acomodar o sistema inteiro, realizei uma modificação na tampa traseira. O arquivo modificado pode ser encontrado na seção "Remix"da página do modelo original.

1.6 Funcionamento

O processo de inicialização (boot) do dispositivo envolve a ativação dos LEDs [1], a configuração dos módulos, e a inicialização da memória interna SPIFFS. A SPIFFS é utilizada para armazenar dados localmente no formato .txt, permitindo a recuperação fácil dos arquivos caso seja necessário transformar o ESP8266 em um servidor de arquivos. Essa estratégia facilita o download dos arquivos para leitura sem dificuldades, especialmente quando o dispositivo está offline. O sistema é configurado para operar normalmente até que 90%[2] **da memória SPIFFS seja ocupada**; após esse limite, o dispositivo passa a permitir apenas saídas. Quando o dispositivo se conecta ou reconecta à rede, ele carrega os dados armazenados localmente para a planilha do Google Sheets e libera o espaço ocupado na memória SPIFFS.

Caso o ESP8266 não consiga se conectar à rede Wi-Fi configurada, ele cria uma rede própria, permitindo que um novo código seja carregado via OTA (Over-the-Air). Os campos `ssid_OTA` e `senha_wifi_OTA` podem ser alterados no arquivo `Credenciais.h`. A senha para a operação OTA também pode ser modificada nesse arquivo, ajustando o campo `#define SenhaOTA`. Se o ESP8266 inicializar sem acesso à rede, ele utilizará os dados salvos da última sincronização com a planilha do Google Sheets. Os arquivos essenciais incluem a lista de usuários autorizados e o status da porta (bloqueada ou não). Caso a porta esteja bloqueada e o ESP8266 reinicialize sem internet, ela permanecerá bloqueada. Na ausência desses arquivos essenciais, o dispositivo exibirá uma mensagem de erro crítico (Código: 11), indicando que não há conexão com a internet e os dados mínimos para operação offline não estão disponíveis.

Quando operando normalmente, ao ler uma nova TAG, o dispositivo verifica se a TAG corresponde a um usuário autorizado. Se não for, o acesso é negado; caso seja, o ESP8266 envia uma solicitação HTTP ao código App Script [3], conforme o seguinte esquema:

```
String url="https://script.google.com/macros/s/"+String
    (GOOGLE_SCRIPT_ID)+"/exec?acao="+dados+"&uidCard="+
    uidCard+"&nome="+nome+"&dt_hr="+hr_dt;
```

[1] O funcionamento dos LEDs baseia-se em um único sinal: quando está em nível alto, o LED vermelho acende e o azul apaga; quando está em nível baixo, o LED azul acende e o vermelho apaga, economizando assim GPIO.

[2] É possível modificar o limite de ocupação da SPIFFS no campo #define LimiteSPIFFS em Credenciais.h. O valor padrão é 90%, equivalente a aproximadamente 204,8 KB com uma configuração de 2MB para SPIFFS.

[3] Caso seja necessário alterar o ID do script Google, modifique o campo #define GOOGLE_SCRIPT_ID em Credenciais.h.

Os parâmetros enviados incluem: acao, uidCard, nome, e dt_hr. Embora todos esses campos sejam enviados em qualquer solicitação, nem todas as funções utilizam todos os parâmetros. Por exemplo, para obter os dados dos usuários autorizados, a solicitação é feita apenas com o campo acao:

```
https://script.google.com/macros/s/[ID DO SCRIPT]/exec?
    acao=bd&uidCard=&nome=&dt_hr=
```

No caso de gravar uma saída offline, a solicitação é enviada da seguinte forma:

```
https://script.google.com/macros/s/[ID DO SCRIPT]/exec?
    acao=saida_off&uidCard=12345&nome=JohnDoe&dt_hr
    =18:37:20-07/11/24
```

As respostas dessas solicitações variam. Por exemplo, ao gravar uma saída offline, o código retorna "OK"em caso de sucesso. Para obter a lista de usuários autorizados, o App Script retorna apenas o UID_Card e o nome dos usuários permitidos. O código HTTP retornado é sempre 200 para indicar sucesso. Se o ESP8266 não receber esse código, ele verifica se pode operar offline; caso possa, ele salva a entrada ou saída localmente.

Os valores suportados pelo campo acao estão definidos no arquivo Credenciais.h:

```
#define entrada "entrada"
#define saida "saida"
#define entrada_off "entrada_off"
#define saida_off "saida_off"
#define bd "bd"
#define status_ "status"
#define atualizar "atualizar"
#define horarios "horarios"
#define bloqueio "bloqueio"
```

O dispositivo executa rotinas periódicas para tentar se reconectar à rede ou verificar atualizações, conforme definido em #define intervaloFuncao2 no arquivo Credenciais.h. Se o dispositivo estiver offline, ele tentará reconectar; caso contrário, solicitará o status na aba INF da planilha para verificar se os dados estão atualizados. Durante o boot, todos os dados são sincronizados independentemente do status da aba INF. Se o valor de status for 1, nenhuma ação é necessária, mas se for 0, o dispositivo sincroniza os dados.

O termo "conectado"refere-se à conexão à rede WAN, que é verificada enviando uma solicitação para google.com. Esta função pode ser modificada no arquivo wifi.h em bool verificar_wan(). O dispositivo possui limites

para erros tolerados (#define acumulo_de_erros_MAX), tempo para conexão Wi-Fi (#define Timeout_WIFI), e tentativas de obtenção de dados via NTP (#define max_tentativas), todos configurados em Credenciais.h.

O dispositivo verifica constantemente se o acesso é permitido conforme os horários definidos, bloqueando o acesso durante as aulas. O status de bloqueio pode ser configurado para impedir o acesso a todos, mesmo aos usuários autorizados, até que o bloqueio seja desativado.

2 Número de participantes:

- Coordenador: Prof. Maurício Khenaifes
  Lotação: COELT-AP
- Colaborador: Prof. Thales Eugênio Portes de Almeida
  Lotação: COELT-AP
- Estudante Colaborador: Silas Torres Rocha
  RA: 2104644
- Total: 3 participantes

3 Resultados do projeto:

3.1 Funcionamento Geral

O ESP8266, em intervalos de tempo definidos, verifica se os dados locais estão sincronizados com a planilha, se está em um horário de aula, bloqueando o acesso ao laboratório e desativando a campainha durante as aulas. Como o ESP8266 não possui um RTC interno, ele monitora o tempo e, se necessário, faz uma nova solicitação NTP para manter a data e a hora sempre corretas.

O código do App Script associado à planilha age como um intermediário, disponibilizando os dados solicitados pelo ESP ou armazenando dados conforme necessário.

3.2 Comunicação com a Planilha

Toda a comunicação entre o ESP e a planilha é mediada pelo App Script. A interface Python utiliza a API do Google, que é detalhada na seção 3.5.1. A comunicação com o App Script é feita via HTTPS. Caso alguém tenha acesso ao ID do script e à estrutura dos parâmetros, poderá realizar operações como:

Operação	Abas
Gravação	Entradas, Saídas
Leitura	Usuários, Horários, INF

Tabela 3: Operações possíveis com o ID e estrutura corretos (apenas usuários com Status 1)

3.2.1 Usuários

A gravação na planilha só é possível através de uma conta Google (via API, interface gráfica ou acesso direto à planilha). A planilha retorna uma sequência de `UID_Card:Nome;` com os `UID_Card` ordenados. A princípio, uma busca binária foi considerada, mas devido ao baixo volume de registros, não foi necessário. A ação não retorna o Status, pois o ESP recebe apenas os registros com Status 1.

3.2.2 Entrada/Saída

A adição de registros de entrada ou saída segue processos quase idênticos, diferenciando-se apenas na aba da planilha. Se o dispositivo estiver online, o ESP envia o `UID_Card` e o nome; o horário é determinado pelo serviço Google, com fuso horário do Brasil.

3.2.3 Entrada/Saída (offline)

Para entradas ou saídas offline, o processo é semelhante ao online, porém o ESP envia o horário local (horário do ESP) e o App Script adiciona a flag que identifica o registro como offline.

3.2.4 Horários

Cada coluna da planilha representa um dia da semana com intervalos de horários. A resposta do App Script é formatada como: `Dia=Intervalo|Intervalo;`.

Exemplo: `"0=7:30-8:20|8:20-9:10;1=9:30-10:00|11:20-12:10"` indica que no Domingo (dia 0) há aulas das 7:30 às 8:20 e das 8:20 às 9:10, e na Segunda (dia 1), das 9:30 às 10:00 e das 11:20 às 12:10. O ESP armazena essa string e a processa para obter o intervalo de horários correspondente ao dia da semana.

3.2.5 INF

Devido à simplicidade da aba INF, com apenas duas células relevantes, as respostas são individuais. O campo `Status` retorna o valor da célula A1, uti-

lizado para controle de sincronismo com o ESP, enquanto o campo `Bloqueio` retorna o valor da célula B1.

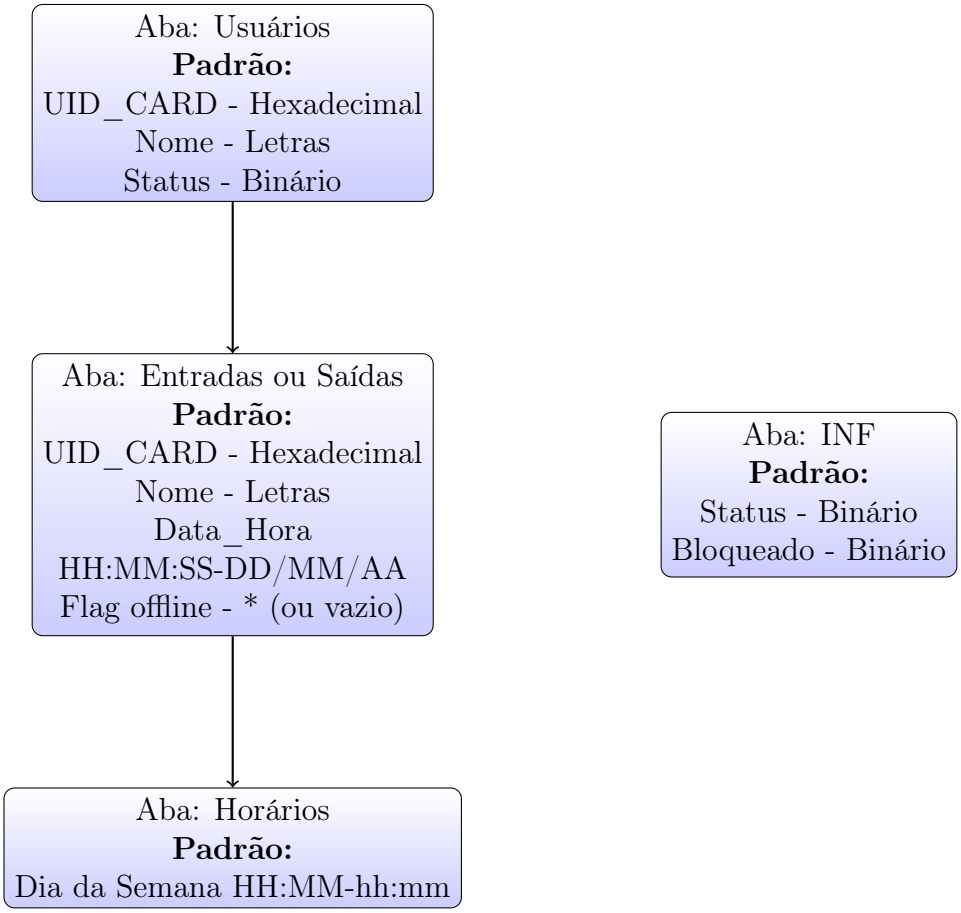
3.3 Padrões de Retorno da Planilha

As respostas às solicitações do ESP seguem os padrões apresentados na Tabela 4.

Operação	Abas	Resposta
Gravação	Entradas, Saídas	"ok"
Leitura	Usuários, Horários, INF	Informação solicitada

Tabela 4: Respostas às solicitações do ESP

3.4 Padrões de Dados da Planilha



	A	B	C	D
1	UID_CARD	Nome	Status	
2	6d0h34	Silas Torres	1	
3	9I0M7	Carlos Rocha	1	
4	AF2N1	Fernanda Peixoto	0	
5	BE3O2	Ricardo Araújo	1	
6	CF4P3	Aline Teixeira	0	
7	DG5Q4	Rodrigo Prado	1	
8	EH6R5	Sandra Monteiro	0	
9	FI7S6	Bruno Neves	1	
10	GJ8T7	Patricia Carvalho	0	
11	HK9U8	Marcos Macedo	1	
12	IL0V9	Claudia Santos	0	

Figura 8: Aba Usuários

H3O

	A	B	C	D
1	UID_CARD	Nome	Data_Hora_Entrada	flag_offline
2	g53udj4	Silas	11:51:18-26/10/23	
3	g53udj4	Silas	13:39:21-26/10/23	
4	g53udj4	Silas	13:49:41-26/10/23	
5	g53udj4	Silas	13:50:43-26/10/23	*
6	g53udj4	Silas	13:51:46-26/10/23	
7	12345	JohnDoe	12:49:03-29/10/23	
8	12345	JohnDoe	13:38:16-29/10/23	*
9	12345	JohnDoe	14:27:30-29/10/23	
10	G58DS42	SilasTorres	14:29:07-29/10/23	
11	W6H4G3	SilasTorresRocha	15:56:38-29/10/23	

Figura 9: Aba Entradas

	A	B	C	D	E	F	G
1	Domingo	Segunda	Terça	Quarta	Quinta	Sexta	Sábado
2	7:30-8:20			7:30-8:20			
3	8:20-9:10			8:20-8:40			
4	9:10-10:00						
5	10:20-11:10						
6	11:10-12:00						
7	12:00-12:50						
8	13:00-13:50					13:00-13:50	
9	13:50-14:40				13:50-14:40	13:50-14:40	
10	14:40-15:30					14:40-15:30	
11	15:50-16:40					15:50-16:40	
12	16:40-17:30				16:40-17:30		
13	17:30-18:20				17:30-18:20		
14	18:40-19:30				18:40-19:30		
15	19:30-20:20				19:30-20:20		
16	20:20-21:10						
17	21:20-22:10						
18	22:10-23:00						

Figura 10: Aba Horários

	A	B
1	1	0
2	Status	Bloqueado
3		

Figura 11: Aba INF

3.5 Interface Gráfica em Python

O uso da interface gráfica em Python visa tornar o sistema de controle de acesso mais amigável. Contudo, seu uso não é obrigatório, pois as modificações podem ser feitas diretamente na planilha, desde que sigam o padrão estabelecido.

3.5.1 Comunicação:

Para que o Python interaja com a planilha, realizando leituras e gravações, foi utilizada a biblioteca de cliente do Google para Python. As configurações necessárias foram realizadas conforme a documentação [4].

Resumidamente, são necessários:

- Python 3.10.7 ou superior;
- Pip;
- Um projeto no Google Cloud;
- Conta Google.

Configurações no Google Cloud:

- Ativar a API do Google Sheets;
- Configurar a tela de permissão OAuth;
- Autorizar credenciais para um aplicativo de computador.

Para permitir que a interface execute em máquinas sem Python instalado, foi gerado um arquivo executável (.exe) via "pyinstaller"[5]. O PyInstaller compila o programa Python e todas as suas bibliotecas necessárias em um único pacote autossuficiente.

3.5.2 Interface

A interface é dividida em seis seções independentes, cada uma responsável por um setor da planilha ou controle de acesso à mesma.

O arquivo inicial, chamado boot.py, verifica se a pasta raiz contém os arquivos necessários para acessar a planilha:

- ID.txt: armazena o ID da planilha.
- Credenciais.json: contém as credenciais baixadas do Google Cloud.

Caso esses arquivos não sejam encontrados, eles serão solicitados e criados no diretório do programa. Durante o acesso à planilha, será necessário autorizar o código com a conta Google associada, processo no qual será gerado um arquivo token.json para acesso, com validade limitada.

Ao clicar em "Abrir Editor", o arquivo principal main.py será aberto, enquanto boot.py será encerrado. O main.py é responsável pela aba "INF"da planilha e exibe o status de sincronização com o ESP8266: vermelho para desatualizado e verde para sincronizado. A interface principal contém sete botões:

O código de exemplo do Google utilizado neste projeto está licenciado sob a Licença Apache 2.0.

Três para controle da aba "INF":
Permitir acesso: Restringe o acesso, mesmo para usuários autorizados.
Recarregar: Recarrega os dados da aba "INF".
Submeter: Modifica o status na aba "INF"para informar ao ESP8266 que ele deve sincronizar com a planilha.
Quatro para controlar as demais abas da planilha:

3.5.3 Usuários

O arquivo usuario.py permite criar, buscar, apagar e modificar usuários. Cada usuário tem um UID_Card associado, que corresponde ao ID da tag RFID, além de um nome e status (1 para acesso permitido, 0 para negado).

3.5.4 Entrada/Saída

Essas abas possuem interfaces similares, permitindo filtrar por datas e horários, salvar cópias dos registros localmente e limpar o histórico de acesso da planilha. Registros destacados em amarelo indicam acessos durante o período offline do ESP8266, enquanto saídas offline sem horário registrado resultam em "ERRO"no campo Data_Hora_Saida.(Não e possível fazer entrada sem horário registrado.)

3.5.5 Horários

Nesta interface, é possível selecionar os horários de aula no laboratório para cada dia da semana. Durante esses intervalos, o acesso é bloqueado automaticamente.

Lembre-se de pressionar "Submeter"na Interface Principal para informar ao ESP que ele deve sincronizar os dados com a planilha. O ESP fará isso na próxima rotina de sincronização, configurada em Credenciais.h no campo #define intervaloFuncao2.

3.6 Descrição Geral dos Resultados

O sistema de controle de acesso ao laboratório desenvolvido demonstrou eficiência e confiabilidade na gestão das entradas e saídas dos usuários autorizados. A integração entre o ESP8266 e o Google Sheets, mediada pelo App Script, permitiu uma comunicação fluida e rápida, com tempo de resposta satisfatório, tanto em condições online quanto offline.

3.7 Análise de Desempenho

Em condições offline, o ESP8266 armazenou com sucesso os registros locais, que foram sincronizados automaticamente assim que a conexão foi restabelecida. Este mecanismo garantiu a integridade dos dados e a continuidade do sistema mesmo em situações de falha de rede.

3.8 Sincronização e Armazenamento de Dados

O sistema demonstrou robustez na sincronização de dados, com todos os registros offline sendo corretamente armazenados e enviados para a planilha ao restaurar a conexão. Não foram observadas perdas de dados durante os testes, e os dados armazenados localmente correspondiam fielmente aos dados enviados para o Google Sheets.

3.9 Controle de Acesso

Os testes realizados com diferentes UIDs mostraram que o sistema é capaz de identificar e diferenciar corretamente os usuários autorizados daqueles não autorizados. As entradas e saídas foram registradas corretamente, com o horário de cada operação sendo armazenado de acordo com as regras estabelecidas.

Não foram detectados erros ou inconsistências significativas no processo de registro, o que reforça a confiabilidade do sistema em cenários de uso real.

3.10 Testes de Segurança

Os testes de segurança realizados indicaram que o sistema é resistente a tentativas de acesso não autorizado. O controle baseado no UID do cartão RFID, aliado ao gerenciamento dos horários de bloqueio, forneceu uma camada adicional de segurança, impedindo acessos fora dos horários permitidos e o uso de cartões não registrados.

3.11 Resumo dos Resultados

O projeto desenvolvido atingiu com sucesso os objetivos propostos, resultando em um sistema robusto e confiável para o controle de acesso ao laboratório. A integração do ESP8266 com o Google Sheets, utilizando o App Script como intermediário, demonstrou-se eficiente tanto em cenários online quanto offline. O sistema foi capaz de gerenciar e armazenar dados de forma segura, garantindo a integridade e precisão das informações registradas.



Figura 12: Foto do dispositivo implementado.

3.12 Contribuições do Projeto

Este trabalho trouxe diversas contribuições, incluindo o desenvolvimento de uma solução acessível para controle de acesso utilizando componentes de hardware de baixo custo e uma integração eficaz com plataformas de software amplamente disponíveis, como o Google Sheets. A automação dos processos de registro e controle de acesso contribui significativamente para a segurança e a eficiência na gestão de ambientes restritos, como laboratórios acadêmicos.

3.13 Limitações do Projeto

Apesar dos resultados positivos, algumas limitações foram identificadas. O uso de um microcontrolador sem RTC (Relógio de Tempo Real) incorporado exige constantes verificações de horário via NTP, o que pode ser um ponto de falha em ambientes com conectividade instável. Além disso, a interface de gerenciamento baseada no Google Sheets, embora funcional, pode não ser a solução mais intuitiva.

3.14 Trabalhos Futuros

Para futuras melhorias, recomenda-se a implementação de um módulo RTC para melhorar a precisão temporal do sistema, eliminando a necessidade de constantes atualizações de horário via NTP. Além disso, o desenvolvimento de uma interface de usuário dedicada, possivelmente uma aplicação web ou móvel, poderia facilitar o gerenciamento dos horários de acesso e o monitoramento das entradas e saídas em tempo real.

Os dois displays atualmente recebem sempre a mesma tela, o que é um legado de versões anteriores do sistema, quando o protocolo I2C não era utilizado devido às limitações de número de pinos disponíveis. Um possível trabalho futuro seria explorar o uso pleno do I2C para permitir que os displays mostrem informações diferentes ou complementares, aproveitando melhor a capacidade dos displays e proporcionando mais flexibilidade na interface visual.

Outra melhoria seria a implementação de um sistema que salve a data e hora da última atualização e mostre na interface gráfica quando será a próxima solicitação do ESP aos dados da planilha. Isso ajudaria a monitorar melhor o estado de sincronização e a prever o próximo ciclo de atualização.

3.15 Considerações Finais

O sucesso deste projeto demonstra a viabilidade de soluções baseadas em microcontroladores para o controle de acesso em ambientes críticos. A abordagem adotada neste trabalho não apenas alcançou os objetivos iniciais, mas também abriu portas para novas possibilidades de pesquisa e desenvolvimento em sistemas de automação e segurança. A continuidade deste trabalho pode contribuir significativamente para o avanço de tecnologias acessíveis e eficientes em diversos setores.

3.16 Custo de fabricação estimado:

Item	**Valor (R$)**
1 ESP8266	18,00
1 Placa Universal (3 x 7 cm)	5,00
1 Tranca Solenoide 12V	45,00
2 Displays LCD 1602	40,00
2 Módulos I2C, baseados no PCF8574	20,00
2 Módulos RFID, baseados no MFRC522 (+ 2 Tag)	30,00
1 Buzzer	2,00
1 Microchave Push Button 6mm	1,00
1 AMS1117 5V	3,00
1 AMS1117 3.3V	3,00
2 Transistores BC337	2,00
2 Resistores 10KΩ	0,20
1 Resistor 1.8KΩ	0,10
1 Resistor 160Ω	0,10
1 Resistor 250Ω	0,10
1 Fonte 12V	20,00
Impressão 3D	15,00
Fios	3,00
4 TAG FRID	0,00
Valor total estimado	**207,50**

4 Referências

Referências

[1] Espressif Systems. *Technical Reference*. Acesso em: 2024. URL: <https://www.espressif.com/sites/default/files/documentation/esp8266-technical_reference_en.pdf>.

[2] Espressif Systems. *Boot Mode Selection*. Acesso em: 2024. URL: <https://docs.espressif.com/projects/esptool/en/latest/esp8266/advanced-topics/boot-mode-selection.html>.

[3] amamonem. *RFID-Arduino Door Lock System*. Acesso em: 2024. 2018. URL: <https://www.thingiverse.com/thing:3236652>.

[4] Google Developers. *Python quickstart*. Acesso em: 2024. URL: <https://developers.google.com/sheets/api/quickstart/python>.

[5] David Cortesi. *PyInstaller Manual*. Acesso em: 2024. URL: <https://pyinstaller.org/en/stable/index.html>.

[6] Martin Olejar. *MFRC522*. Acesso em: 2024. URL: <https://os.mbed.com/users/AtomX/code/MFRC522/docs/tip/classMFRC522.html>.

[7] Lucas Maziero. *LiquidCrystal_I2C*. Acesso em: 2024. URL: <https://github.com/lucasmaziero/LiquidCrystal_I2C>.