Projeto Fechadura Eletrônica

André Saibert Mattos & Thiago Oliveira Castro

IFMT – Octayde Jorge Da Silva

andrescmattos@hotmail.com & thiagoliveira_007@hotmail.com

Resumo - Este artigo tem como objeto criar um sistema de fechadura eletrônica, na qual, será controlada pelo microcontrolador ESP32, além do ESP32 a fechadura também conta com um teclado de membrana para a inserção de ID (identidade) e de senha. Os IDs e as senhas serão armazenados em um memória EEPROM I2c gerenciada pelo ESP e também em uma planilha google que possui conexão com a internet, assim criasse um banco de dados online, podendo ser editado de forma remota.

1. INTRODUÇÃO

Os microcontroladores têm desempenhado um importante papel na vida das pessoas, visto que estão integrados em diferentes tipos de equipamentos eletrônicos, eletrodomésticos, brinquedos, equipamentos médicos, linhas de montagem fabril, entre muitos outros. A integração dos microcontroladores tem crescido completo e novas aplicações do seu uso têm surgido. Grande parte desse crescimento decorre da aplicação dos microcontroladores em sistemas embarcados. A expressão embarcada (do inglês *embedded system*) decorre de o microcontrolador estar inserido nas aplicações (dispositivos) que executam funções específicas para as quais foram desenvolvidos.

Mesclando o mercado dos microcontroladores e da segurança eletrônica podemos nos referir a um equipamento bem útil para a sociedade, a fechadura eletrônica, pensando no controle de acesso de pessoas em quaisquer ambientes onde requer segurança é crucial que aja um controle de quem entra e quem sai do ambiente.

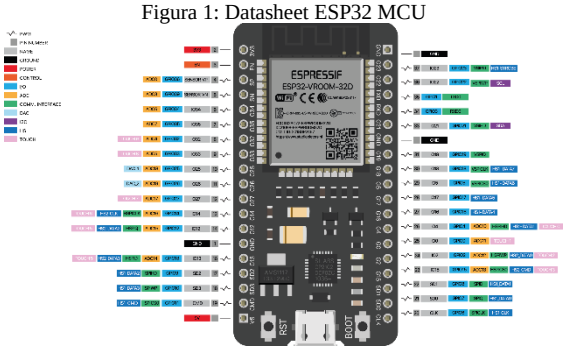
Desta forma este trabalha tem o intuito de criar uma fechadura eletrônica para sem implementada em qualquer ambiente onde é necessário.

2. Revisão Teórica

2.1 Micro Controlador ESP32

"ESP32 é uma série de microcontroladores de baixo custo e baixo consumo de energia. Também é um sistema-em-um-chip com microcontrolador integrado, Wi-Fi e Bluetooth. ESP32 foi criado e desenvolvido por *Espressif* Sistemas, uma empresa Chinesa com sede em Xangai e é fabricado pela TSMC usando seu processo de fabricação de 40 nm." (*Espressif*, 2021)

Figura 1: Datasheet ESP32 MCU



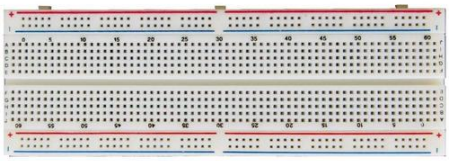
Fonte: https://www.studiopieters.nl/esp32-pinout/

2.2 Protoboard

"A Protoboard, também conhecida como placa de ensaio, matriz de contato ou *breadboard* (em inglês), é uma placa que permite a montagem e teste de

circuitos sem a necessidade de soldar, apenas “espetando” os componentes na placa. Com isso, é possível montar um circuito que não conhecemos muito bem seu comportamento e efetuar diversos testes, tendo a liberdade de substituir os componentes da forma que desejar e só soldar o circuito em uma placa definitiva quando tudo estiver testado e funcionando perfeitamente.” (ATHOS, 2021)

Figura 2: Protoboard



Fonte: https://produto.mercadolivre.com.br/MLB-1006852465-10-protoboard-breadboard-830-pontos-furos-pic-raspb-_JM

2.3 IDE – Arduino

“A IDE do Arduino é um ambiente de desenvolvimento integrado. Em outras palavras, é um espaço onde você tem tudo que precisa para programar sua placa baseada nessa plataforma escrevendo seus códigos de maneira satisfatória, rápida e eficiente” (Torres, 2013)

Figura 3: IDE - Arduino

Fonte: Autoria Própria

2.4 Bando de Dados

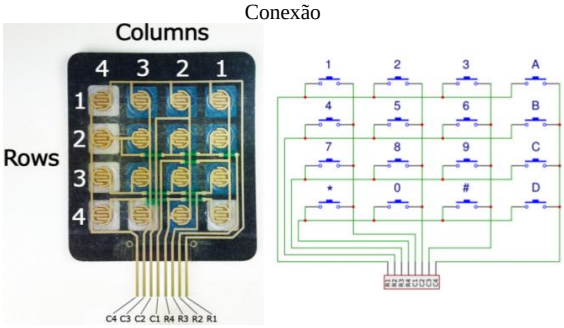
Um banco de dados é uma coleção organizada de informações - ou dados - estruturadas, normalmente armazenadas eletronicamente em um sistema de computador. Um banco de dados é geralmente controlado por um sistema de gerenciamento de banco de dados (DBMS). Juntos, os dados e o DBMS, juntamente com os aplicativos associados a eles, são chamados de sistema de banco de dados, geralmente abreviados para apenas banco de dados. Os dados nos tipos mais comuns de bancos de dados em operação atualmente são modelados em linhas e colunas em uma série de tabelas para tornar o processamento e a consulta de dados eficientes. Os dados podem ser facilmente acessados, gerenciados, modificados, atualizados, controlados e organizados. A maioria dos bancos de dados usa a linguagem de consulta estruturada (SQL) para escrever e consultar dados. (ORACLE, 2021)

2.5 Teclado de Membrana

Teclados de membrana são mais comuns e mais baratos que os mecânicos. A tal membrana é uma série de telas de silicone que ficam abaixo das teclas. Elas contam com contatos eletrônicos para cada umas teclas de seu teclado. Quando você pressiona uma delas, todas as camadas entram em contato, fechando um circuito elétrico e mandando um sinal para o controlador do teclado, que o interpreta de acordo com a tecla que você pressionou. (TechTudo, 2021)

Um exemplo é um teclado de 16 teclas, que estão dispostas em 4 linhas por 4 colunas, e ele possui 8 pinos para ligação. Embaixo de cada tecla há um interruptor de membrana. Cada interruptor em uma linha é conectado aos outros interruptores da mesma linha por um traço condutor sob o bloco, e da mesma forma são conectadas às colunas, onde todos os botões da coluna também estão conectados. Ou seja, todos os botões do teclado estão conectados a uma linha e a uma coluna, por isso que é chamado de teclado matricial. (Castro,2020)

Figura 4 Esquema de Ligação dos Botões e Terminais de Conexão



Fonte: https://www.circuitbasics.com/how-to-set-up-a-keypad-on-an-arduino/

2.6 LED

Na língua portuguesa, a palavra LED significa diodo emissor de luz. Trata-se de um componente eletrônico capaz de emitir luz visível transformando energia elétrica em energia luminosa. Esse processo é chamado de eletroluminescência.

O primeiro LED foi criado em 1963 pelo engenheiro e inventor Nick Holonyak e era capaz de emitir apenas a cor vermelha. Com o passar dos anos e com o desenvolvimento da tecnologia, novas cores de LED foram desenvolvidas. Diferentemente do laser, o LED não emite luz monocromática, mas sim uma faixa pequena de determinadas cores. Os LEDs são feitos de materiais semicondutores. Substituindo alguns dos seus átomos por outros em um processo chamado de dopagem, é possível controlar a cor emitida pelo dispositivo. (HELERBROCK,2021)

2.7 LCD 16x2 com I2C

Módulos de display LCD de caracteres alfanuméricos são interfaces de comunicação visual muito úteis e atraentes. Eles se encontram em quase todos os aparelhos domésticos, eletroeletrônicos, automóveis, instrumentos de medição etc. São dispositivos que possuem interfaces elétricas padronizadas e recursos internos gráficos e de software que permitem facilmente a permuta por outros de outros fabricantes, sem que seja necessário alterar o programa de aplicação. Por ser altamente padronizado seu custo é baixo. É um recurso antigo, deve ter uns vinte anos de idade ou mais, mas continua atual, com suas inúmeras formas, cores, tamanhos e preços. A tecnologia predominante continua sendo o LCD (Liquid Crystal Display), porém já se pode encontrar alguns baseados em LEDs orgânicos (OLED).(PUHLMANN,2015)

O Módulo Adaptador I2C para Display LCD foi desenvolvido com a finalidade de simplificar a conexão de display LCD ao microcontrolador. Para uma conexão de 4 bits entre o display LCD e o microcontrolador é necessário ao menos 6 cabos, logo, se o microcontrolador tiver poucas portas digitais isso poderá ser um problema. Com o Módulo Adaptador I2C para Display LCD são necessários apenas 2 cabos de comunicação entre o display LCD e o microcontrolador.

Em um projeto mais extenso e que é necessário à utilização de muitas portas digitais por parte de outros dispositivos, o Módulo Adaptador I2C para Display LCD pode ser a solução simples e prática para que você economize algumas portas digitais na ligação do seu display LCD. (Oliveira, 2018)

Figura 5 LDC com I2C

Fonte: https://solectroshop.com/pt/lcd/285-16x2-1602-tela-azul-iic-i2c-lcd.html

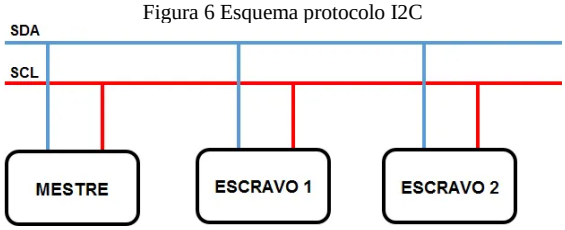
2.8 Memória EEPROM

A EEPROM está presente em todas as versões do Arduino, mas muitas vezes a falta de conhecimento de sua existência é contornada com a instalação de memórias EEPROM externas ou mesmo de um cartão de memória SD de vários *gibabytes* para armazenar algumas poucas variáveis ou dados. Uma EEPROM (de *Electrically-Erasable Programmable Read-Only Memory*) é um tipo de memória que pode armazenar valores que serão retidos mesmo quando a energia é desligada e pode ser programada e apagada várias vezes, eletricamente. Pode ser lida um número ilimitado de vezes, mas só pode ser apagada e programada um número limitado de vezes, que varia normalmente entre 100.000 e 1 milhão. (Garagem,2012)

2.9 Protocolo I2C

O modo de funcionamento do protocolo I2C é baseado na interação entre elementos seguindo a hierarquia mestre/escravo, ou seja, quando temos vários dispositivos se comunicando segundo esta premissa, pelo menos um destes deve atuar como mestre e os demais serão escravos. A função do mestre consiste em realizar a coordenação de toda a comunicação, pois, ele tem a capacidade de enviar e requisitar informações aos escravos existentes na estrutura de comunicação, os quais, devem responder às requisições citadas. A estrutura na qual o protocolo I2C atua é uma estrutura de barramento, que por sua vez, consiste em um arranjo em que todos os elementos se encontram conectados a um ramal principal. (Madeira, 2017)

Figura 6 Esquema protocolo I2C



Fonte: Madeira

3. Metodologia

Para a confecção do trabalho foi utilizado como base um programa inspiração passado pelo professor Dr. Alberto Willian Mascarenhas. A partir da análise deste projeto foi desenvolvido o fluxograma de lógica programada (anexo 1).

Invista que já temos uma lógica a ser programada foi esquematizado quais materiais necessitavam estar na montagem, são ele:

- Um microcontrolador: - Foi escolhido o ESP32.
- Um teclado de menbrana: - Foi escolhido um teclado reaproveitado de micro-ondas.
- Uma memória externa: - Foi escolhida a EEPROM 24C16
- Um visor responsavel por mostrar as informações: - Foi escolhido um LCD 16x2 com I2C

Após a escolha dos matérias, foi feita a etapa da escolha de como seria a montagem do equipamento, neste caso foi escolhido uma protoboard chamada placa furada, nela foi conectado os pinos para cada módulo a ser fixado, desta forma a placa obteve a configuração desta maneira:

Figura 7 Modulo Placa Montado

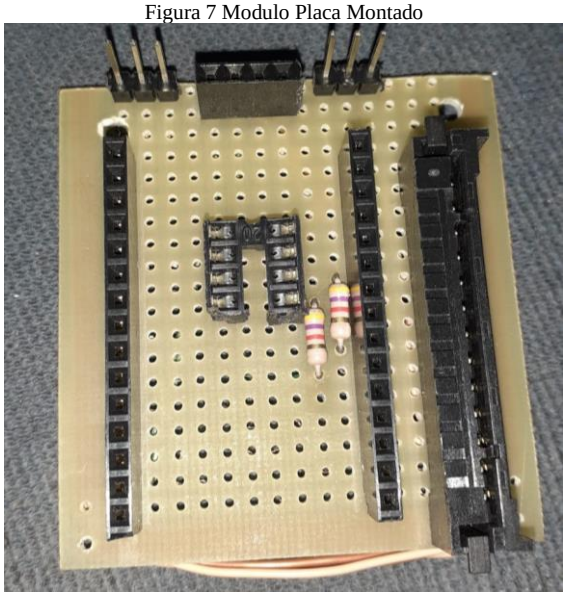
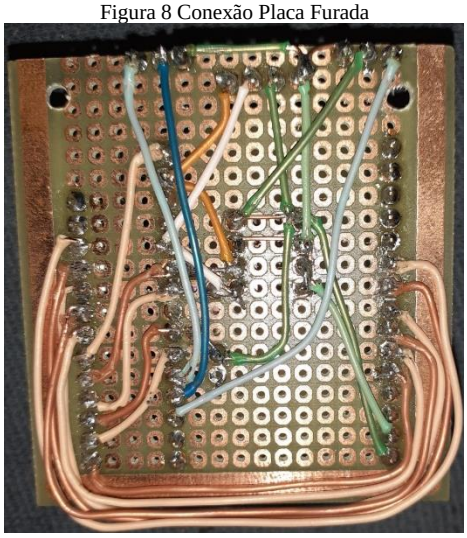
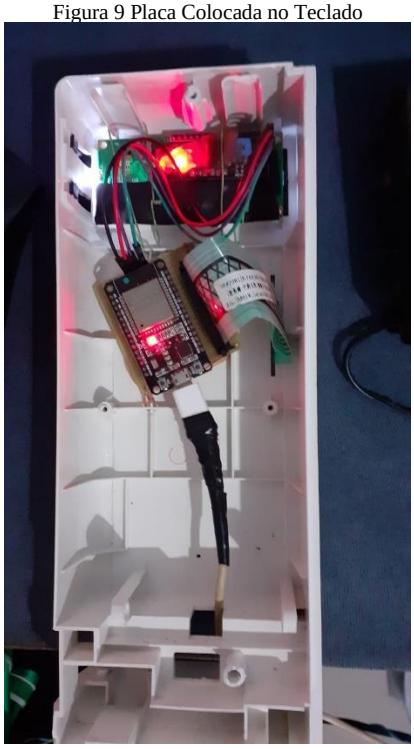


Figura 8 Conexão Placa Furada



Toda a board foi colocado no teclado de micro ondas para ficar escondido, funcionou perfeitamente.

Figura 9 Placa Colocada no Teclado



A partir deste momento começou um trabalho a ser de programação, estão começamos a pratica para fazer o teclado funcionar.

Para o teclado funcionar foi necessário um mapeamento de pinos, assim apertando as teclas e observando qual era a resposta nos pinos foi possível mapear qual era a matriz de cada botão.

A disposição de cada tecla ficou com a seguinte forma:

Figura 10 representação do teclado com coluna/linha

Figura 11 Tabela coluna x linha

	3	4	5	6	7
13	cancelar		relógio	memória	ligar
12	frango		0	7	2^
11	brigadeiro	carne^	9	pipoca	3^
10	potência	macarrão^	8	4	bovina
9	prato	arroz^	5	2	1
8	4^	legumes^	6	3	1^

Partindo do ponto do mapeamento surgiu o outro passso, a necessidade de leitura de botões precionados, assim foi criado um dado chamado *string teclado* (anexo 3) é o local onde verifica se existe botão apertado, essa *string* é composta de varias funções *if* e *else* que serviram pra entender o que é linha e o que é coluna.

Quando essa iformação é coletada, o seu valor é atribuido a uma variavel chamada tecla, tecla então entra em outra string chamada *String MatrizTeclado* essa por sua vez ira analisar linha e coluna e falar que valor sera atribuido para o botão precionado (anexo 2).

Alem deste esquema pra captação de teclas apertadas, tambem foi necessario a criação de um *timer* para que o codigo nao tenha que parar para

fazer a leitura do que esta acontecendo, desta maneira foi necessario fazer um pesquisa de como funciona o timer no *ESP32* (anexo 4).

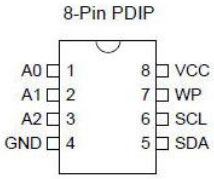
Após o teclado ser mapeado, configurado e ajustado o *timer*, foi a vez de entender como funciona a memória 24C16. Para entendermos como funciona foi necessario ler o datasheet[1] da mesma.

Desta forma foi compreendido que em uma memória de 2048 bytes quando se tem 8 endereços de entrada cada endereço tera uma parcela do total de bytes desta forma cada uma das entradas terão direito a alocar 256 bytes de informação.

Figura 12 Datasheet e Pinos Memória 24C16

Pin Configurations

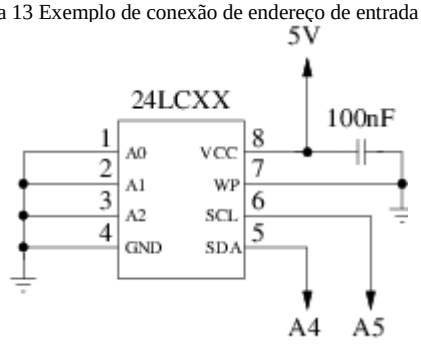
Pin Name	Function
A0 - A2	Address Inputs
SDA	Serial Data
SCL	Serial Clock Input
WP	Write Protect



Fonte: http://dami.azw.pt/eeprom-24c32/

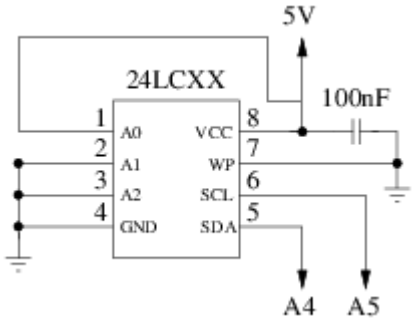
Para definir o endereço de entrada é de forma física, necessita que troque as conexões dos pinos exemplo:

Figura 13 Exemplo de conexão de endereço de entrada 0x50



Neste caso meu endereço de armazenamento é o 0x50 pois todos os endereços de entrada estão no GND.

Figura 14 Exemplo de conexão de endereço de entrada 0x51



Neste outro caso o endereço de entrada é o 0x51 pois o meu pino A0 está em alto. Desta forma quando cada pino é colocado em alto muda o endereço. Todos os endereços são:

$A_0A_1A_2 > 0x5x$

000 > 0x50
100 > 0x51
010 > 0x52
110 > 0x53
001 > 0x54
101 > 0x55
011 > 0x56
111 > 0x57

O endereço escolhido para fazer a alocação dos dados deste trabalho foi justamente o exemplo 2, o 0x51.

Para escreve na memória ultilizamos uma biblioteca chamada Wire.h, que é responsável por conter as funções necessárias para gerenciar a comunicação entre os dispositivos através do protocolo I2C.

Utilizamos a sentença Wire.begin() para inicializar a comunicação do ESP32 através do protocolo I2C, no entanto, desta vez, utilizamos o valor 0x51 como parâmetro, em outras palavras, quando fazemos isto, estamos dizendo que a placa ESP32 em questão estará presente no barramento como escravo e que o endereço do mesmo será 0x51.

Com a biblioteca Wire.h usamos as seguintes funções:

- *Wire.beginTransmission()* – Inicia a transmissão de dados como o dispositivo escravo.
- *Wire.write()* – Escreve a informação no dispositivo escravo.
- *Wire.endTransmission*() – encerra a transmissão de dados como o dispositivo escravo.
- *Wire.requestFrom*() - Ssolicita o retorno do que está escrito na posição indicada no dispositivo escravo.
- *Wire.read*() - Faz a leitura da mensagem

gravada no dispositivo escravo.

Encerrada a parte de memória iniciou então a adequação das escritas que saíram do monitor serial da IDE e passaram a ser no LDC, para fazer essa escrita também foi utilizado LiquidCrystal_I2C.h, que é responsável por conter as funções necessárias para gerenciar a escrita no display LDC.

Com a biblioteca LiquidCrystal_I2C.h usamos as seguintes funções:

- *lcd.init*() - Inicia o LCD
- *lcd.backlight*() – Acende a luz de fundo do LCD
- *lcd.clear*() – Limpa tudo que está escrito no display
- *lcd.setCursor*() – posiciona o curso do LDC no local que começa a escrever
- *lcd.print*() – Escreve a mensagem que deseja

Terminado o processo de inserção de bibliotecas começou a lógica de programação, essa parte não é complicada, é apenas trabalhosa, neste momento as funções foram criadas para cada necessidade, desta forma obtivemos certa de 23 funções, são elas:

1. void interrupcao_timer – Responsavel por cirar um timer para leitura de tecla;
2. void setup – Função resposavel por iniciar a configuração do microcontrolador;
3. void loop() – Responsavel por fazer um loop pedindo ao *USER* colocar o iD e a senha;
4. void Escrever_EEPROM – Responsavel por gravar os dados na memoria;
5. byte Ler_Byte – Responsável por fazer a leitura da memória byte a byte
6. EstruturaBancoDados Ler_EEPROM – faz a leitura da EEPROM por completo quando solicitada;
7. String checarCredenciais – Responsável por pesquisar se o usuário e existe e se a senha bate com o banco de dados;
8. void LiberarEntrada – Libera a entrada da pessoa que solicitou;
9. void cadastrar – Opção que Cria cadastros e o adiciona na memória;
10. void apagar – Quando solicitado apaga o iD desejado;
11. void mudarTipo – Edita o tipo do usuário pra 1 ou 0;
12. void mudarSenha() – Quando solicitada edita a senha do iD desejado;
13. void mudarID - Quando solicitada edita o iD desejado;
14. void editar – Esta função gera um menu pra opções de editar são elas: editar iD, editar senha e editar tipo;
15. void menuADM – Gera o menu do ADM;
16. void checarParametros – Esta função checa de o iD e a senha foram digitados certos;
17. void redirecionar – Limpa a memória EEPROM por completa;
18. void primeiraInicializacao – Quando a catraca não possui ao menos 1 ADM, ela inicia um cadastro de ADM.
19. String MatrizTeclado – Lê as linhas e colunas do teclado;
20. String teclado – Verifica a tecla pressionada;
21. void conectarWIFI – Função responsável por conectar o ESP na internet
22. void exportarBanco – Função que gera um dado e o manda para nuvem
23. void importarBanco -Função que gera um dado que armazena os dados presente na web, depois o grava na memória.

Após as funções colocadas no código foi a etapa de criar o banco de dados, para fazer isso, foi necessário fazer uma pesquisa na web sobre banco de dados, e como funciona as planilhas googles, foram cerca de 3 exemplos seguidos para conseguir fazer essa comunicação. Foram seguidos os exemplos do Sr. Ítalo Coelho, Sr. Cristiano Zanini Nazário e do blog ICHI.PRO.

Usando esses exemplos foi possível fazer a conexão entre o ESP e a planilha.

Neste contexto foi necessário o estudo de mais duas linguagens de programação Java script e HTML, assim foi gerado um código para a receber

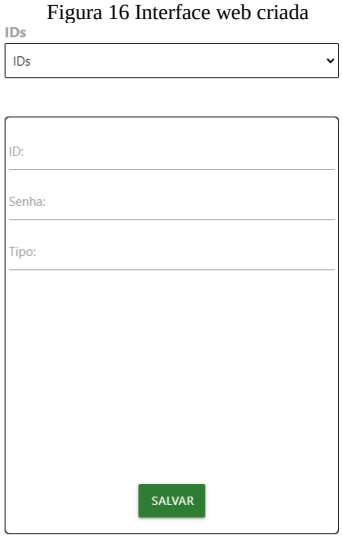
o banco, enviar o banco e editar, todos esses códigos foram escritos no APPS Script, um software online que o google disponibiliza para gerarmos códigos que automatizam o sistema de planilhas, documentos ou apresentação deles.

Figura 15 ambiente de trabalho apps script

Com o script pronto, agora o ESP consegue envia um vetor com os 3 parâmetros (iD, senha e tipo) para a planilha, lá conseguimos editar ou apagar as posições do vetor.

O ultimo passo a se fazer agora é criar uma interface de usuário web para que o ADM não abra apenas a planilha, então utilizando o HTML foi criado a seguinte interface.

Figura 16 Interface web criada



4. CONCLUSÕES

A fechadura eletrônica foi programada com sucesso, nela foi possível cadastrar vários usuários e administradores, apagar cadastros até mesmo editar os edita-los, além do principal que é liberar acesso por meio de iD e senha.

A fechadura funciona da seguinte maneira, na primeira inicialização a fechadura irá conferir se existe algum administrador cadastrado na memória, se não estiver cadastrado ele chama uma função para fazer o primeiro cadastro, caso já tenho o cadastro de ADM, a rotina se inicia.

A partir deste ponto a fechadura pede pra colocar o iD e a senha, se os dados estiverem incorretos ele pede pro usuário digitar novamente e acende um led vermelho, se estiver correto ela passa pro próximo e identifica se o iD é de usuário ou de ADM, se for de usuário a fechadura exibe uma mensagem de “bem vindos”, acende um led verde e libera a catraca, se for um iD de ADM ele abre um primeiro menu, nesse menu encontramos as seguintes opções: Liberar entrada, Menu e Sair.

Caso o ADM escolha a opção liberar entrada, a fechadura exibe uma mensagem de “bem vindos”, acende um led e libera a catraca. Caso o ADM escolha sair, a fechadura volta pra rotina inicial e pergunta qual o iD e a senha novamente.

No menu o ADM pode escolher outras opções como: cadastrar usuário, editar um usuário, apagar usuário, exportar banco de dados, importar banco de dados e novamente a opção sair para sair deste menu e ir pra tela anterior.

Na opção cadastrar usuário o ADM pode criar cadastros comum de user ou cadastro de ADM, ele digita um iD (um numero de 0 – 65534) uma senha de 4 dígitos e um tipo que pode ser 1 para ADM ou 0 para user.

Na opção editar usuário o ADM tem a opção de mudar o iD dos usuários cadastrados, a senha ou o tipo do usuário, tornando assim os cadastros editáveis.

A opção importar banco de dados, vai até a planilha googles, capta os dados presentes na planilha e o traz para memoria EEPROM.

A opção exportar banco de dados, cria um arquivo o envia pra planilha e a atualizada de acordo com os dados da memória EEPROM.

A fechadura faz todos esses processos, porem foi constatado que de vez enquanto o ESP trava e apresenta um algum erro, não se sabe ao

certo o porquê, de acordo com as pesquisas acreditamos que seja por conta do Watchdog nativo do ESP.

Para fazer essa correção foi feita uma pesquisa no site da empresa desenvolvedora do ESP a ESPRESSIF, lá obtivemos o comando "#define CONFIG_ESP_INT_WDT_TIMEOUT_MS 600" Desta forma foi solucionado parcialmente o erro de travar, ele ainda trava porem ocorre com menos frequência.

Assim chegamos a conclusão que apesar do problema a fechadura funciona e realiza as tarefas pedidas a ela.

5. REFERÊNCIAS

ATHOS ELECTRONICS (Brasil). **Protoboard – O que é – Simulador Online**. [*S. l.*], 2021. Disponível em: https://athoselectronics.com/protoboard-simulador-online/. Acesso em: 6 out. 2021.

BORSOI, Beatriz T.; **FAVARIM,** Fábio; CARNIEL, Guilherme; BRITO, Robison C. **Projeto e Desenvolvimento de Fechadura Eletrônica controlada pela Internet**. [*S. l.*], 2015. Disponível em: https://siaiap32.univali.br/seer/index.php/acotb/article/view/6989. Acesso em: 14 nov. 2021.

[1] **DATASHEET Memória 24C16**. [*S. l.*], 2003. Disponível em: http://dami.azw.pt/wp-content/uploads/2018/11/EEPROM_24C32_datasheet1.pdf. Acesso em: 10 nov. 2021.

CASTRO, Giovanni de; **CASSIOLI,** Matheus. **Usando o Teclado Matricial com Arduino.** [S. l.], 2020. Disponível em: https://www.robocore.net/tutoriais/usando-teclado-matricial-com-arduino. Acesso em: 14 nov. 2021.

COELHO, Ítalo. **Como enviar dados do ESP8266 para o Google Sheets**. [*S. l.*], 2021. Disponível em: https://www.filipeflop.com/blog/como-enviar-dados-do-esp8266-para-o-google-sheets/. Acesso em: 22 nov. 2021

ESP32: A feature-rich MCU with integrated Wi-Fi and Bluetooth connectivity for a wide-range of applications. [*S. l.*], 26 set. 2021. Disponível em: https://www.espressif.com/en/products/socs/esp32. Acesso em: 6 out. 2021.

ESPRESSIF. **Project Configuration**. [*S. l.*], 2016. Disponível em: https://docs.espressif.com/projects/esp-idf/en/latest/esp32s2/api-reference/kconfig.html#config-esp-int-wdt-timeout-ms. Acesso em: 22 nov. 2021.

GARAGEM, Laboratório de. **Artigo: Tutorial: Usando a EEPROM do Arduino para armazenar dados de forma permanente**. [*S. l.*], 2012. Disponível em: https://labdegaragem.com/profiles/blogs/tutorial-usando-a-eeprom-do-arduino-para-armazenar-dados-de-forma. Acesso em: 15 nov. 2021.

HELERBROCK, Rafael. **O que é LED?**; Brasil Escola. Disponível em: https://brasilescola.uol.com.br/o-que-e/fisica/o-que-e-led.htm. Acesso em 14 de novembro de 2021.

ORACLE. **BANCO de dados definido**. [*S. l.*], 2021. Disponível em: https://www.oracle.com/br/database/what-is-database/. Acesso em: 14 nov. 2021.

ICHI.PRO. **MicroPython em ESP32: envio de dados para Planilhas Google**. [*S. l.*], 2019. Disponível em: https://ichi.pro/pt/micropython-em-esp32-envio-de-dados-para-planilhas-google-169094141954363. Acesso em: 22 nov. 2021

MADEIRA, Daniel. **Protocolo I2C – Comunicação entre Arduinos**. [*S. l.*], 20 nov. 2021. Disponível em: https://portal.vidadesilicio.com.br/i2c-comunicacao-entre-arduinos/. Acesso em: 17 nov. 2021.

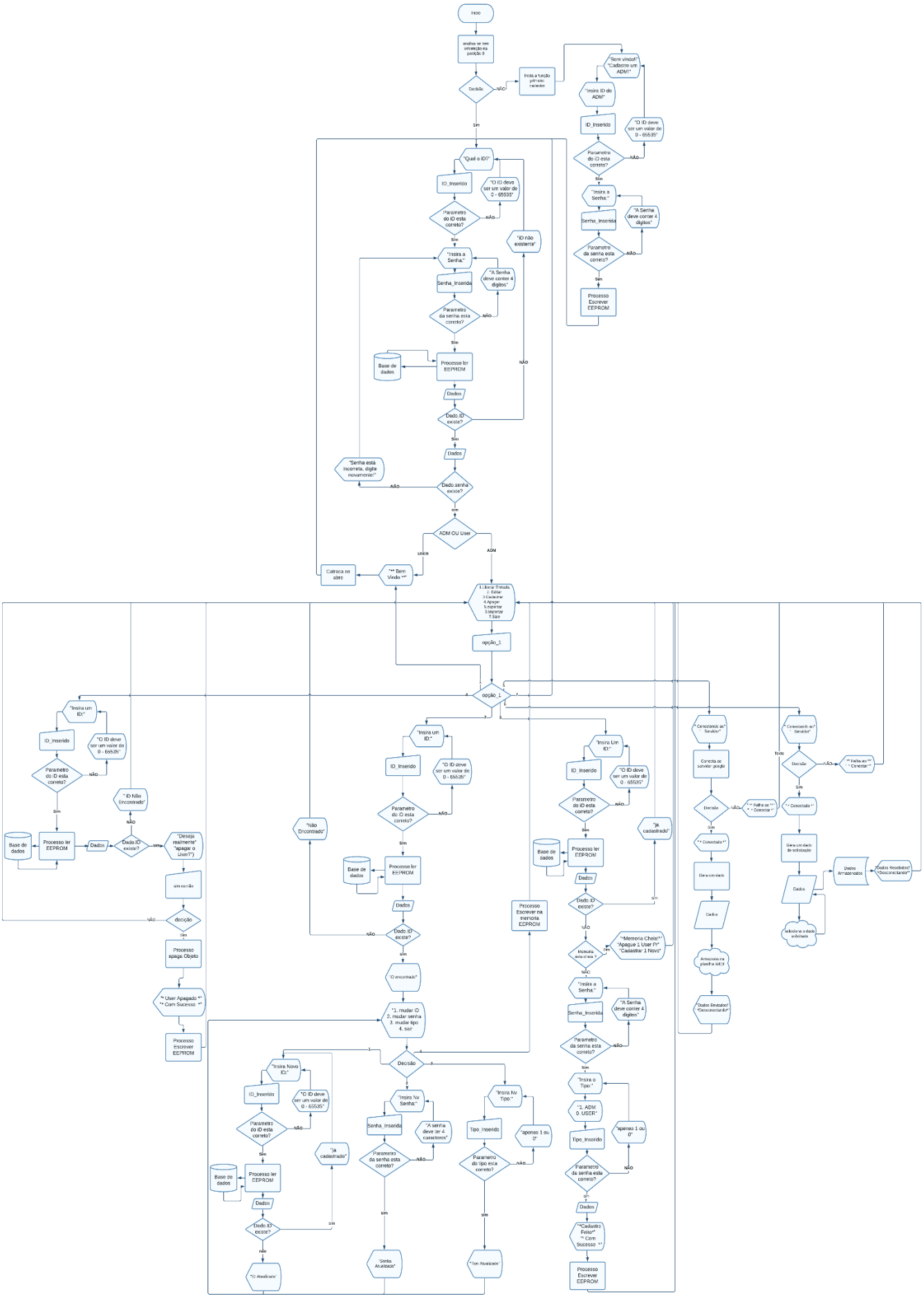
NAZÁRIO, Cristiano Zanini. **Aprenda como usar o ESP32 para publicar dados no Google Sheets**. [*S. l.*], 2021. Disponível em: https://www.crescerengenharia.com/post/aprenda-como-usar-o-esp32-para-publicar-dados-no-google-sheets. Acesso em: 22 nov. 2021.

OLIVEIRA, Euler. **Como usar com Arduino – Módulo Adaptador I2C para Display LCD (16X2 / 20X4)**. [*S. l.*], 2018. Disponível em: https://blogmasterwalkershop.com.br/arduino/como-usar-com-arduino-modulo-adaptador-i2c-para-display-lcd-16x2-20x4. Acesso em: 15 nov. 2021.

PUHLMANN, Henrique Frank Werner. **Módulo de Display LCD**. [*S. l.*], 2015. Disponível em: https://www.embarcados.com.br/modulo-de-display-lcd/. Acesso em: 15 nov. 2021.

TechTudo. **TECLADOS de membrana**: como funcionam e melhores modelos no Brasil: Saiba como funcionam os teclados membrana, que são mais baratos e fáceis de achar, e conheça alguns dos melhores modelos no Brasil. [S. l.], 2017. Disponível em: https://www.techtudo.com.br/noticias/2016/12/teclados-de-membrana-como-funcionam-e-melhores-modelos-no-brasil.ghtml. Acesso em: 14 nov. 2021.

Anexo 1


Início
analisa se tem informação na posição 0
Decisão
NÃO
Sim
Inicia a função primeiro cadastro
"Bem vindo!!"
"Cadastre um ADM!!"
"Insira ID do ADM!"
ID_Inserido
"O ID deve ser um valor de 0 - 65535"
Parametro do ID esta correto?
NÃO
Sim
"Insira a Senha:"
"A Senha deve conter 4 digitos"
Senha_Inserida
Parametro da senha esta correto?
NÃO
Sim
Processo Escrever EEPROM
"Qual o ID?"
ID_Inserido
"O ID deve ser um valor de 0 - 65535"
Parametro do ID esta correto?
NÃO
Sim
"ID não existente"
"Insira a Senha:"
"A Senha deve conter 4 digitos"
Senha_Inserida
Parametro da senha esta correto?
NÃO
Sim
Base de dados
Processo ler EEPROM
NÃO
Dados
Dado ID existe?
Sim
Dados
"Senha está incorreta, digite novamente!"
NÃO
Dado senha existe?
sim
ADM OU User
USER
ADM
Catraca se abre
"** Bem Vindo **"
1.Listar Ferrado
2. Editar
3.Cadastrar
4.Apagar
5.esperar
6.Imprimir
7.Sair
opção_1
opção_1
"Insira um ID:"
ID_Inserido
"O ID deve ser um valor de 0 - 65535"
Parametro do ID esta correto?
NÃO
Sim
" ID Não Encontrado"
NÃO
Base de dados
Processo ler EEPROM
Dados
Dado ID existe?
Sim
"Deseja realmente "apagar o User?")
sim ou não
decição
NÃO
Sim
Processo apaga Objeto
"* User Apagado *"
"* Com Sucesso *"
Processo Escrever EEPROM
"Insira um ID:"
ID_Inserido
"O ID deve ser um valor de 0 - 65535"
Parametro do ID esta correto?
NÃO
Sim
"Não Encontrado"
Base de dados
Processo ler EEPROM
Dados
Dado ID existe?
NÃO
sim
"ID encontrado"
"1. mudar ID
2. mudar senha
3. mudar tipo
4. sair"
Decisão
1
2
3
4
"Insira Novo ID:"
ID_Inserido
"O ID deve ser um valor de 0 - 65535"
Parametro do ID esta correto?
NÃO
Sim
"Já cadastrado"
Base de dados
Processo ler EEPROM
Dados
Dado ID existe?
Sim
não
"ID Atualizado"
"Insira Nv Senha:"
Senha_Inserida
"A senha deve ter 4 caracteres"
Parametro da senha esta correto?
NÃO
sim
"Senha Atualizada"
"Insira Nv Tipo:"
Tipo_Inserido
"apenas 1 ou 0"
Parametro do tipo esta correto?
NÃO
sim
"Tipo Atualizado"
"Insira Um ID:"
ID_Inserido
"O ID deve ser um valor de 0 - 65535"
Parametro do ID esta correto?
NÃO
Sim
"Já cadastrado"
Base de dados
Processo ler EEPROM
Dados
Dado ID existe?
sim
NÃO
Memoria esta cheia ?
Sim
"Memoria Cheia!**"
"Apague 1 User ?"
"Cadastrar 1 Novo"
NÃO
"Insira a Senha:"
Senha_Inserida
"A Senha deve conter 4 digitos"
Parametro da senha esta correto?
NÃO
Sim
"Insira o Tipo:"
"1. ADM
0. USER"
Tipo_Inserido
"apenas 1 ou 0"
Parametro da senha esta correto?
NÃO
sim
Dados
"*Cadastro Feito*"
"* Com Sucesso *"
Processo Escrever EEPROM
Processo Escrever na memoria EEPROM
" Conectando ao Servidor "
Conecta ao servidor (porta)
Decisão
NÃO
" ** Falha ao ** "
" * Conectar * "
Sim
" ** Conectado ** "
Gera um dado
Dados
Armazena na planilha (CSV)
"Dados Enviados!"
"Desconectando!"
" Conectando ao Servidor "
Decisão
NÃO
" ** Falha ao ** "
" * Conectar * "
Sim
" ** Conectado ** "
Gera um dado de solicitação
Dados
solicitarei o dado solicitado
Dados Armazenados
Dados Recebidos!
"Desconectando!"

Anexo 2

```
String MatrizTeclado(int linha, int coluna){
  String tecla;
  switch(linha){
    case 1:switch(coluna){
            case 1: tecla = "cancelar"; break;
            case 2: tecla = " ";        break;
            case 3: tecla = "relógio";  break;
            case 4: tecla = "memória";  break;
            case 5: tecla = "ligar";    break;
           }break;
    case 2:switch(coluna){
            case 1: tecla = "frango"; break;
            case 2: tecla = " ";      break;
            case 3: tecla = "0";      break;
            case 4: tecla = "7";      break;
            case 5: tecla = "2^";     break;
           }break;
    case 3:switch(coluna){
            case 1: tecla = "brigadeiro"; break;
            case 2: tecla = "carne^";     break;
            case 3: tecla = "9";          break;
            case 4: tecla = "pipoca";     break;
            case 5: tecla = "3^";         break;
           }break;
    case 4:switch(coluna){
            case 1: tecla = "potência";  break;
            case 2: tecla = "macarrão^"; break;
            case 3: tecla = "8";         break;
            case 4: tecla = "4";         break;
            case 5: tecla = "bovina";    break;
           }break;
    case 5:switch(coluna){
            case 1: tecla = "prato";  break;
            case 2: tecla = "arroz^"; break;
            case 3: tecla = "5";      break;
            case 4: tecla = "2";      break;
            case 5: tecla = "1";      break;
           }break;
    case 6:switch(coluna){
            case 1: tecla = "4^";       break;
            case 2: tecla = "legumes^"; break;
            case 3: tecla = "6";        break;
            case 4: tecla = "3";        break;
            case 5: tecla = "1^";       break;
           }break;
  }
  return tecla;
}
```

Anexo 3

```
String teclado(){
  String tecla;
  if(digitalRead(C1) && passador == 1){ passador = 2;digitalWrite(C1,  LOW); digitalWrite(C2, HIGH); digitalWrite(C3, HIGH);
digitalWrite(C4, HIGH); digitalWrite(C5, HIGH);
    // -- Testa qual tecla foi pressionada e armazena o valor --
    if     (!digitalRead(L1))  tecla = MatrizTeclado(1,1);
    else if(!digitalRead(L2))  tecla = MatrizTeclado(2,1);
    else if(!digitalRead(L3))  tecla = MatrizTeclado(3,1);
    else if(!digitalRead(L4))  tecla = MatrizTeclado(4,1);
    else if(!digitalRead(L5))  tecla = MatrizTeclado(5,1);
    else if(!digitalRead(L6))  tecla = MatrizTeclado(6,1);}else if(digitalRead(C2) && passador == 2){ passador = 3; digitalWrite(C1, HIGH);
digitalWrite(C2,  LOW); digitalWrite(C3, HIGH); digitalWrite(C4, HIGH); digitalWrite(C5, HIGH);
    // -- Testa qual tecla foi pressionada e armazena o valor --
    if     (!digitalRead(L1))  tecla = MatrizTeclado(1,2);
    else if(!digitalRead(L2))  tecla = MatrizTeclado(2,2);
    else if(!digitalRead(L3))  tecla = MatrizTeclado(3,2);
    else if(!digitalRead(L4))  tecla = MatrizTeclado(4,2);
    else if(!digitalRead(L5))  tecla = MatrizTeclado(5,2);
    else if(!digitalRead(L6))  tecla = MatrizTeclado(6,2);}else if(digitalRead(C3) && passador == 3){
    passador = 4; digitalWrite(C1, HIGH); digitalWrite(C2, HIGH); digitalWrite(C3,  LOW); digitalWrite(C4, HIGH); digitalWrite(C5,
HIGH);
    // -- Testa qual tecla foi pressionada e armazena o valor --
    if     (!digitalRead(L1))  tecla = MatrizTeclado(1,3);
    else if(!digitalRead(L2))  tecla = MatrizTeclado(2,3);
    else if(!digitalRead(L3))  tecla = MatrizTeclado(3,3);
    else if(!digitalRead(L4))  tecla = MatrizTeclado(4,3);
    else if(!digitalRead(L5))  tecla = MatrizTeclado(5,3);
    else if(!digitalRead(L6))  tecla = MatrizTeclado(6,3);  }else if(digitalRead(C4) && passador == 4){
    passador = 5;digitalWrite(C1, HIGH); digitalWrite(C2, HIGH); digitalWrite(C3, HIGH); digitalWrite(C4,  LOW); digitalWrite(C5,
HIGH);
    // -- Testa qual tecla foi pressionada e armazena o valor --
    if     (!digitalRead(L1))  tecla = MatrizTeclado(1,4);
    else if(!digitalRead(L2))  tecla = MatrizTeclado(2,4);
    else if(!digitalRead(L3))  tecla = MatrizTeclado(3,4);
    else if(!digitalRead(L4))  tecla = MatrizTeclado(4,4);
    else if(!digitalRead(L5))  tecla = MatrizTeclado(5,4);
    else if(!digitalRead(L6))  tecla = MatrizTeclado(6,4);}else if(digitalRead(C5) && passador == 5){passador = 1;digitalWrite(C1, HIGH);
digitalWrite(C2, HIGH); digitalWrite(C3, HIGH); digitalWrite(C4, HIGH); digitalWrite(C5, LOW);
 // -- Testa qual tecla foi pressionada e armazena o valor --
    if     (!digitalRead(L1))  tecla = MatrizTeclado(1,5);
    else if(!digitalRead(L2))  tecla = MatrizTeclado(2,5);
    else if(!digitalRead(L3))  tecla = MatrizTeclado(3,5);
    else if(!digitalRead(L4))  tecla = MatrizTeclado(4,5);
    else if(!digitalRead(L5))  tecla = MatrizTeclado(5,5);
    else if(!digitalRead(L6))  tecla = MatrizTeclado(6,5);
  }
  return tecla;
}
```

Anexo 4

```
// Configurando o Timer2
  hw_timer_t * timer = NULL;        /* Ponteiro usado para a configuração o cronômetro do Timer2    */
  timer = timerBegin(2, 80, true);  /* Inicializando o cronômetro, timerBegin(X, Y, Z)
                                    * X: indica o numero do temporizador que será usado (de 0 a 3).
                                    * Y: valor do prescaler, sabendo que a frequência do sinal de base usado pelo ESP32 é de 80MHz, iremos definir um prescaler de 80, para que obtenhamos um sinal com uma frequência de 1MHz que incrementará o contador do temporizador 1000000 vezes por segundo.
                                    * Z: bandeira indicando se o contador deve aumentar (true) ou regressar (false).
                                    */
  timerAttachInterrupt(timer, &interrupcao_timer, true);   /* Vincula ao timer uma função de manipulação, que será executada quando a interrupção for gerada
                                                           * timerAttachInterrupt(X, &Y, Z)
                                                           * X: ponteiro para o temporizador iniciado
                                                           * Y: endereço da função que será executa quando houver a interrupção
                                                           * Z: bandeira indicando se a interrupção a ser gerada é borda (true) ou nível (false)
                                                           */
  timerAlarmWrite(timer, 80000, true);  /* Especifica o valor do contador no qual a interrupção do timer será gerada.
                                        * timerAlarmWrite(X, Y, Z)
                                        * X: ponteiro para o temporizador iniciado
                                        * Y: o valor do contador em que a interrupção deve ser gerada, como definimos o prescaler para 80, teremos o contador em microssegundos, ou seja, 1000000us = 1s
                                        * Z: bandeira indicando se o temporizador deve recarregar automaticamente ao gerar a interrupção.
                                        */
  timerAlarmEnable(timer);  /* Habilitamos o Timer que acabamos de configurar  */
```