INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
DO RIO GRANDE DO SUL
CAMPUS RESTINGA

CRISTOPHER MARQUES DA SILVA

PROTÓTIPO DE UM ESTACIONAMENTO VERTICAL PARA A OTIMIZAÇÃO DE ESPAÇOS EM CONDOMÍNIOS RESIDENCIAIS URBANOS

Porto Alegre, abril de 2021.

CRISTOPHER MARQUES DA SILVA

PROTÓTIPO DE UM ESTACIONAMENTO VERTICAL PARA A OTIMIZAÇÃO DE ESPAÇOS EM CONDOMÍNIOS RESIDENCIAIS URBANOS

Monografia do Trabalho de Conclusão do Curso de Tecnologia em Eletrônica Industrial do Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Sul - *Campus* Restinga, como requisito parcial à obtenção do título de Tecnólogo em Eletrônica Industrial.

Orientador: Prof. Dr. Eng. Alexsandro Cristovão Bonatto

Porto Alegre, abril de 2021.

CRISTOPHER MARQUES DA SILVA

PROTÓTIPO DE UM ESTACIONAMENTO VERTICAL PARA A OTIMIZAÇÃO DE ESPAÇOS EM CONDOMÍNIOS RESIDENCIAIS URBANOS

Monografia do Trabalho de Conclusão do Curso de Tecnologia em Eletrônica Industrial do Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Sul - *Campus* Restinga, como requisito parcial à obtenção do título de Tecnólogo em Eletrônica Industrial.

Orientador: Prof. Dr. Eng. Alexsandro Cristovão Bonatto

BANCA EXAMINADORA:

_____

Professor Matheus Perin
Titulação Mestre em Engenharia Elétrica

_____

Professor Fausto Barbosa
Titulação Doutor em Ciências com ênfase em Astrofísica

_____

Coordenação do Curso:
Profª Elizandra Martinazzi

Dedico este trabalho à minha família, que deu apoio em minha trajetória acadêmica.

AGRADECIMENTOS

Agradeço aos meus familiares por me ajudarem em momentos de dificuldade, aos meus professores e professoras pelo grande apoio nos ensinamentos necessários para a conclusão desta graduação.

“O ignorante afirma, o sábio duvida, o sensato reflete.”
(Aristóteles)

“A mente que se abre a uma nova ideia jamais retorna ao seu tamanho original.”
(Albert Einstein)

RESUMO

Este projeto visa atender às necessidades de otimização de espaços em condomínios residenciais urbanos. Propõem-se implementar um sistema de estacionamento vertical na forma de carrossel rotativo em espaços condominiais. Tendo em vista que nas áreas urbanas os espaços estão em crescente ocupação, a redução da ocupação de área de solo usada para vagas de estacionamento de veículos traz como principais benefícios a otimização dos espaços em condomínios residenciais. Desta forma, proporciona a abertura de novos lugares tanto para a construção de mais prédios, quanto para a ampliação de áreas de lazer, conforme as necessidades que a gestão condominial priorizar frente às escolhas dos moradores. Este projeto visa desenvolver um protótipo de sistema de estacionamento vertical, para permitir uma avaliação do seu funcionamento do ponto de vista do controle eletrônico de acesso e da movimentação das vagas.

Palavras-chave: Estacionamento Vertical; Condomínios Residenciais Urbanos.

PROTOTYPE OF A VERTICAL PARKING FOR AREA OPTIMIZATION IN URBAN RESIDENTIAL CONDOMINIUMS

ABSTRACT

This project aims to answer the needs of space optimization in urban residential condominiums. It is proposed to implement a vertical parking system in the form of a rotating carousel in condominium spaces. Bearing in mind that spaces in urban areas are in increasing occupation, the reduction in the occupation of land area used for vehicle parking spaces brings as main benefits the optimization of spaces in residential condominiums. In this way, it provides the opening of new places both for the construction of more buildings, and for the expansion of leisure areas, according to the needs that the condominium management prioritize in the face of the residents' choices. This project aims to develop a prototype of a vertical parking system, to allow an evaluation of its functioning from the point of view of electronic access control and the movement of spaces.

Keywords*: Parking; Urban Residential Condominiums.*

LISTA DE FIGURAS

Figura 1 - Foto do elevador de carros da Engetax Engiquick Parking: a) carros estacionados de forma empilhados; b) entrada de um veículo na plataforma de movimentação veículos; c) plataforma de movimentação do veículo dentro do estacionamento **22**

Figura 2 - Torre de estacionamento vertical (Volkswagen AG stand parked - Edifício garagem automatizado) **23**

Figura 3 - Imagem do estacionamento vertical rotativo Jiu Hong, com dimensões de área de 6,5 m x 5,6 m de largura e alturas de 9,7 m, 11,58 m ou 13,51 m, dependendo do número de vagas de veículos **24**

Figura 4 - Imagem de estacionamento vertical rotativo da Empresa Roanpu, com dimensões de área largura de 15,5 m de altura e 18,5 m **24**

Figura 5 - Imagem de um estacionamento vertical rotativo da empresa Solid Park, com dimensões de largura de 6m a 8m e altura de 7m a 10m **25**

Figura 6 - a) Imagem de um estacionamento vertical rotativo da empresa Tada, com dimensões de largura de 6 metros e altura de 10 m; b) Figura estacionamento vertical multinível empilhado com 5 colunas Tada com uma altura de 9,2 m e uma largura 2,5 m por coluna **26**

Figura 7 - Componentes de um sistema RFID para registro de produtos **28**

Figura 8 - Tipos de *tags* RFID: a) *tag* RFID passivas; b) *tag* RFID ativas; e c) *tag* RFID semi-passivas **28**

Figura 9 - Imagem de um leitor RFID dos tipos (a) móvel e (b) fixo **29**

Figura 10 - Ilustração de diferentes modelos de equipamentos para leitura de etiquetas RFID dos tipos: (a) portal com especificações das partes operantes do equipamento enumeradas no centro do desenho; (b) portal para identificação de produtos etiquetados com *tags* RFID; e (c) portal para identificação de veículos **30**

Figura 11 - Imagem ilustra a representação um (a) túnel de leitura RFID em uma esteira de produção nomeando os principais elementos que seriam o objeto identificado, a antena e o leitor; e (b) foto real de um túnel de leitura RFID em uma esteira de produção que identifica produtos já embalados em caixa **30**

Figura 12 - Na imagem é mostrado um sistema utilizando um leitor RFID móvel para coletar dados de produtos estocados em prateleiras onde em cada seção da prateleira há uma tag instalada com referências do produto que está condicionado **31**

Figura 13 - Esta é uma imagem que demonstra um modelo de leitor RFID manual **31**

Figura 14 - Modelo de impressora que imprime etiqueta RFID **32**
Figura 15 - Imagem de uma Impressora Zebra Industrial e RFID **32**
Figura 16 - Ilustração do acoplamento indutivo usado nos sistemas RFID 13,56 MHz **33**
Figura 17 - Densidade da potência em função Rayleigh, Fresnel e Frauenhoffer **33**
Figura 18 - Região de campo próximo reativo, campo radioativo e campa distante **34**
Figura 19 - Imagem demonstrando uma antena com dipolo magnético ou também conhecido como laço retangular **34**
Figura 20 - Fotos do módulo de leitor RFID RC522; e (b) Chaveiro RFID 13,56 MHz **35**
Figura 21 - Imagens da Plataforma do Atmega 2560, com (a) o microcontrolador Atmega 2560; a (b) plataforma Arduino Mega 2560; e o (c) microcontrolador Atmega 16u2 **36**
Figura 22 - Circuito da comunicação serial do Arduino 2560 **37**
Figura 23 - Alimentação elétrica pode ser conectada na conexão da porta USB (conector superior da figura), quanto na fonte externa pelo plug jack p4 (conector inferior) **37**
Figura 24 - Circuito regulador de tensão do Arduino 2560 **38**
Figura 25 - Circuito de proteção da porta USB do Arduino 2560 **38**
Figura 26 - Circuito selecionador da fonte do Arduino 2560 **39**
Figura 27 - Conectores de alimentação para conexão de shields e módulos na placa Arduino Mega 2560 **40**
Figura 28 - Especificação completa dos terminais do Arduino Mega 2560 **41**
Figura 29 - Ilustração de um objeto metálico em frente a face do sensor indutivo **43**
Figura 30 - Ilustração sensor indutivo do tipo embutido **44**
Figura 31 - Ilustração sensor indutivo do tipo não embutido **44**
Figura 32 - Ilustração sensor indutivo do tipo semi embutido **45**
Figura 33 - a) Sensor indutivo NPN no arranjo normalmente aberto; b) Sensor indutivo NPN no arranjo normalmente fechado **45**
Figura 34 - a) Sensor indutivo PNP no arranjo normalmente aberto; b) Sensor indutivo PNP no arranjo normalmente fechado **46**
Figura 35 - Sensor indutivo PNP no arranjo normalmente aberto da marca TPCID e modelo Lj12A3-4Z/BY **46**
Figura 36 - Componentes de força entre dois dentes magneticamente permeáveis **47**
Figura 37 - Arranjo das bobinas de motores de passo unipolares na topologia com 6, 5 e 8 contatos **48**
Figura 38 - Arranjo das bobinas de motores de passo bipolar **48**
Figura 39 - Driver A4988 **49**

Figura 40 - Diagrama elétrico do driver A4988 **50**
Figura 41 - Motor de passos Stepperonline 17HS19-2004S1 **51**
Figura 42 - Dsservo Ds3218mg **52**
Figura 43 - Sensor de obstáculo infravermelho reflexivo **53**
Figura 44 - Ilustração da reflexão da luz infravermelha partindo do emissor refletindo em um obstáculo em direção ao receptor **53**
Figura 45 - Circuito do módulo do sensor de obstáculo reflexivo **54**
Figura 46 - Estrutura construída para o protótipo, apresentando as vistas: (a) frontal da estrutura; (b) lateral direita; (c) lateral esquerda; e (d) traseira. **56**
Figura 47 - Dianteira (a) e lateral (b) da plataforma da estrutura com seus pedestais **57**
Figura 48 - Foto rolamento com mancal para eixos de 8mm KFL08 **58**
Figura 49 - Rolamento com mancal para eixos de 8mm KFL08, instalado na estrutura **58**
Figura 50 - Fotografias lateral (a) e superior (b) das engrenagens inferiores instaladas com o eixo e rolamento; Fotografias lateral (c) e superior (d) das engrenagens superiores instaladas com o eixo e rolamento **59**
Figura 51 - Fotografia dianteira (a), traseira (b) e lateral (c) da plataforma de PVC instalada na estrutura **60**
Figura 52 - Foto motor de passos unido ao eixo inferior por um acoplamento flexível **61**
Figura 53 - Contra-porcas traseiras (a) e dianteiras (b) de uma das engrenagens inferiores **61**
Figura 54 - Furações feitas para permitir o ajuste de tensionamento da correia, apontada pela marcação amarela em forma de elipse: a) furação esquerda do rolamento traseiro; b) furação direita do rolamento traseiro; c) furação esquerda do rolamento dianteiro; d) furação direita do rolamento dianteiro **62**
Figura 55 - a) Foto demonstrando o parafuso para a fixação da plataforma do carro na correia após sacar o pino da correia; b) Foto demonstrando porcas para ajustar a plataforma fixada na correia **63**
Figura 56 - a) Servo motor instalado com as partes móveis do acionamento do freio; b) Superior do sistema de freio; c) Lateral esquerda do freio; d) Lateral direita do freio **64**
Figura 57 - Sensores indutivos posicionados para cada vaga nas plataformas **65**
Figura 58 - Sentido crescente da posição dos sensores frente do protótipo, de 1 ao 4 **65**
Figura 59 - Posição dos parafusos que passam a frente do sensor sendo: a) parafuso da vaga 1; b) parafuso da vaga 2; c) parafuso da vaga 3; e d) parafuso da vaga 4 **66**

Figura 60 - Posição dos sensores de obstáculo em lugares da estrutura onde não oferecem segurança: sensor na lateral esquerda (a), direita (b) e traseira (c) da estrutura em destaque com marcação em forma de bordas quadradas amarelas **67**
Figura 61 - Cancela do protótipo: a) lateral esquerda da cancela; b) lateral direita da cancela; c) traseira da cancela; d) dianteira da cancela **68**
Figura 62 - Servo motor Tower Pro Micro Servo 9g SG90 **68**
Figura 63 - a) Servo motor instalado na cancela; b) sensor de obstáculo instalado na cancela **69**
Figura 64 - Ilustração esquemática das ligações dos sensores e motores com o Arduíno, gerado pelo *software* Fritzing **70**
Figura 65 - Diagrama esquemático dos circuitos eletrônicos dos sensores, motores e a *shield* do Arduino, gerado pelo *software* Fritizing **71**
Figura 66 - Fluxograma de controle de acesso para usuários cadastrados no sistema **72**
Figura 67 - Algoritmo do cadastramento e descadastramento de *tags* RFID **74**
Figura 68 - Algoritmo do controle de acesso às vagas **75**
Figura 69 - Algoritmo para buscar uma nova vaga ou buscar um carro estacionado **76**
Figura 70. Algoritmo de controle de movimentação e parada do sistema de estacionamento vertical **78**
Figura 71. Mensagem de cadastro do cartão mestre **82**
Figura 72: Mensagem de entrada no modo de configuração **82**
Figura 73: Mensagem gerada quando é feito o cadastro de uma nova *tag* RFID **83**
Figura 74: Mensagem confirmando a validação da *tag* cadastrada **83**
Figura 75: Mensagem confirmando a saída do modo de configuração após passar a *tag* mestre no leitor **84**
Figura 76: Mensagem de exclusão de uma *tag* cadastrada no sistema **84**
Figura 77: Interface demonstrando o acesso liberado de uma *tag* cadastrada **85**
Figura 78: Interface demonstrando o acesso negado para uma *tag* não cadastrada **86**

LISTA DE TABELAS

Tabela 1: Quadro comparativo entre fabricantes e modelos de estacionamento vertical **26**

Tabela 2: Variação da distância que afeta o campo eletromagnético que emerge sobre a face sensora conforme o diâmetro do sensor e distância sensora nominal **45**

Tabela 3: Tabela verdade para configurar o modo de operação do tipo de passo do motor de passos **49**

Tabela 4: Especificações servo motor Dsservo Ds3218mg **52**

LISTA DE ABREVIATURAS E SIGLAS

ABS	*Acrylonitrile Butadiene Styrene*
AC	*Alternating Current*
DC	*Direct Current*
EEPROM	*Electrically Erasable Programmable Read-Only Memory*
EUA	Estados Unidos da América
FET	*Field Effect Transistor*
GND	*Graduated Neutral Density filter*
IBGE	*Instituto Brasileiro de Geografia e Estatística*
IDE	*Integrated Development Environment*
ICSP	*In Circuit Serial Programing*
LED	*Light Emitting Diode*
MISO	*Master In Slave Out*
MOSI	*Master Out Slave In*
MS	*Microresolution Step*
NC	*Normally Closed*
NO	*Normally Open*
PC	*Personal Computer*
PLC	*Programmable Logic Controller*
PVC	*Policloreto de Vinila*
PWM	*Pulse Width Modulation*
RFID	*Radio Frequency Identification*
SCL	*Serial Clock*
SDA	*Serial Data*
SPI	*Serial Peripheral Interface*
SUV	*Sport Utility Vehicle*
TWI	*Two Wire Interface*
UHF	*Ultra High Frequency*
USB	*Universal Serial Bus*
VCC	*Tensão de Corrente Contínua*
VIN	*Voltage Input*

LISTA DE SÍMBOLOS

VAC	*Tensão alternada*
COM	*Tensão Comum*
Tx	*Transmissor*
Rx	*Receptor*
Sn	*Distância sensora nominal*
λ	*Lambda, comprimento da onda*
AREF	*Tensão de referência para entradas analógicas*
NPN	*Negativo, Positivo, Negativo*
PNP	*Positivo, Negativo, Positivo*
n	*Nominal*
t	*Tangencial*
D	*Comprimento aproximado de uma antena*

SUMÁRIO

1. INTRODUÇÃO **18**
1.1 PROBLEMA E JUSTIFICATIVA 19
1.2 OBJETIVOS 21
2. REFERENCIAL TEÓRICO **22**
2.1 SISTEMAS DE ESTACIONAMENTO VERTICAL 22
2.2 IDENTIFICAÇÃO POR RADIOFREQUÊNCIA 27
2.2.1 TIPOS DE TAGS EXISTENTES 27
2.2.2 LEITOR RFID 28
2.2.3 IMPRESSORA DE ETIQUETAS RFID 31
2.2.4 ANTENA RFID 32
2.3 PLATAFORMA ARDUINO MEGA 2560 35
2.3.1 O MICROCONTROLADOR ATMEGA 2560 35
2.3.2 ALIMENTAÇÃO ELÉTRICA DO ATMEGA 2560 37
2.3.3 ESPECIFICAÇÕES DOS TERMINAIS DO ARDUINO 39
2.3.4 A MEMÓRIA NÃO VOLÁTIL DO ATMEGA 41
2.4 SENSOR INDUTIVO DE PROXIMIDADE 42
2.4.1 TIPOS DE SENSORES INDUTIVOS 43
2.4.2 CONFIGURAÇÃO ELÉTRICA DOS SENSORES 44
2.5 MOTOR DE PASSOS 46
2.5.1 POLARIZAÇÕES DE MOTORES DE PASSO 47
2.6 SERVO MOTORES 50
2.7 SENSOR DE BARREIRA INFRAVERMELHO REFLEXIVO 51
2.7.1 MÓDULO SENSOR DE OBSTÁCULO INFRAVERMELHO REFLEXIVO 52
3. DESENVOLVIMENTO DO PROJETO **54**
3.1 ESPECIFICAÇÕES DO PROTÓTIPO 54
3.2 MONTAGEM DA ESTRUTURA METÁLICA 54
3.3 MONTAGEM DO SISTEMA DE FREIO 62
3.4 INSTALAÇÃO MOTOR DE PASSO PARA ACIONAMENTO DO CARROSSEL DE VAGAS 63
3.5 INSTALAÇÃO DOS SENSORES INDUTIVOS 64
3.6 MONTAGEM DA CANCELA DE CONTROLE DE ACESSO 66

3.7 IMPLEMENTAÇÃO PARA O CONTROLE E O SENSORIAMENTO 68
3.8 PROGRAMAÇÃO DO SISTEMA 70
3.8.1 ALGORITMO DE CONTROLE DE ACESSO 70
3.8.2 ALGORITMO DE CADASTRAMENTO DE MORADORES 72
3.8.3 ALGORITMO DE CONTROLE DE ACESSO ÀS VAGAS 73
3.8.4 ALGORITMO PARA FAZER A BUSCA DE UMA VAGA 74
3.8.5 ALGORITMO DE CONTROLE DE MOVIMENTAÇÃO E PARADA 75
4. ANÁLISE DE RESULTADOS 79
4.1 CADASTRAMENTO, DESCADASTRAMENTO E LEITURA DE TAGS RFID 79
4.2 CONTROLE DE ACESSO E BUSCA DE VAGAS 83
5. CONCLUSÕES 85
5.1 TRABALHOS FUTUROS 86
REFERÊNCIAS 87
APÊNDICE A - CÓDIGO FONTE 91

1. INTRODUÇÃO

De acordo com dados do IBGE, o fenômeno da urbanização se intensificou a partir da década de 1960 frente a dois contextos: a mecanização da produção agrícola, que expulsa os trabalhadores rurais do campo para a cidade e; o avanço do processo de industrialização concentrado em centros urbanos (capitais e cidades) proporcionando maior acesso ao emprego e renda. Em 2015, 85% da população brasileira já vivia em áreas urbanas (IBGE, 2016). Com essa grande massa de pessoas e famílias vivendo em centros urbanos surge a necessidade de uma reorganização dos espaços urbanos e do tipo de moradias.

Nesse contexto, no Brasil, os condomínios residenciais (horizontais ou verticais, de classes populares ou médias e altas) passam a multiplicar-se e, com isso, a exigir que novas tecnologias sejam aplicadas à otimização de seus espaços, tendo em vista tanto a ampliação de prédios ou casas quanto dos espaços de lazer visando uma maior socialização entre moradores bem como a interação com a natureza. Contudo, a grande densidade populacional que reside em condomínios residenciais faz com que todo espaço disponível seja ocupado, estreitando a possibilidade de ampliar espaços internos tanto de prédios e casas (no caso de condomínios horizontais) bem como das áreas de lazer. Dessa forma, a tecnologia pode ser utilizada a favor de uma maior otimização dos espaços condominiais, com impactos visíveis na qualidade de vida dos moradores.

Motivado por estas configurações de caráter sociodemográfico e tecnológico, proponho um projeto tecnológico que tem por objetivo avaliar o funcionamento de um sistema de estacionamento vertical do tipo carrossel, que permite otimizar espaços em condomínios residenciais urbanos. Este tipo de estacionamento possui maior capacidade de vagas para os automóveis, ocupando espaço significativamente menor que estacionamentos convencionais (horizontais ou em espaços abertos delimitados para estacionamento).

Sistemas de estacionamento vertical são mecanismos automatizados de elevação e de armazenamento de veículos automotores. Estão presentes em diversas cidades com alta densidade demográfica e com necessidade de maiores espaços de estacionamento, conforme já destacado. No ano de 2012 estima-se que existiam 1,6 milhão de vagas no sistema de estacionamento automatizado (HAMELINK, 2011). Um mecanismo clássico de estacionamento automatizado é o modelo vertical, adotado neste trabalho, que faz o empilhamento de veículos através de um sistema mecânico para a elevação e transporte de veículos. Um dos primeiros sistemas de estacionamento automatizado foi construído em 1905 na cidade de Paris, em uma estrutura de concreto com múltiplos andares e um elevador para

transporte dos veículos (TR FÓRUM, 2021). Outro registro de uma das primeiras construções de sistemas de estacionamento vertical foi feito nos EUA, um sistema chamado de Kent Garagens de estacionamento automático, estes sistemas foram usados no final de 1920 até o início de 1960. Eram garagens públicas financiadas pelo Kent Garage Investing Corporation of New York. Um exemplo é a garagem de 28 andares na Quincy Street em Chicago, Illinois, com 1200 vagas de estacionamento (MECHANICAL PARKING SYSTHEMS, 2011). Um sistema autônomo para estacionamento vertical de veículos também foi construído na cidade de Washington no ano de 1951 (BEYOND DC, 2021).

A organização textual deste projeto tecnológico, após este capítulo introdutório, traz, no segundo capítulo, o referencial teórico. Neste capítulo é exposto, brevemente, uma argumentação sobre a relação entre o crescimento populacional nos centros urbanos, a tecnologia e a necessidade de otimizar espaços urbanos, nomeadamente, em condomínios residenciais. Serão abordados os dados de relevância dos dispositivos necessários para a construção da maquete focada na implementação do controle eletrônico de um estacionamento vertical tipo carrossel.

1.1 PROBLEMA E JUSTIFICATIVA

O momento atual é caracterizado por profundas transformações no espaço urbano das cidades. Isto se dá, dentre outros fatores, devido à constatação de que, pela primeira vez na história, a população urbana ultrapassa a população rural. Em pouco tempo, o mundo será, em sua maioria, urbanizado, tendo, no ano de 2015, 85% da população brasileira já residindo em áreas urbanas (LEVY, 2010; IBGE, 2016).

Fruto deste processo, surge a necessidade de repensar os tipos de moradias nos espaços urbanos de forma que se possa abrigar cada vez mais pessoas e famílias, consequentemente a expansão imobiliária que nasce junto com este processo (dimensão que não vou abordar). Observa-se, nesse sentido, o crescimento constante de moradias do tipo condomínios residenciais tanto para abrigar pessoas e famílias das classes populares quanto das classes médias e altas. Entendendo que a tecnologia se desenvolve juntamente com o processo de urbanização da população brasileira e mundial, o instrumento tecnológico pode ser utilizado em favor de uma maior otimização de espaços físicos urbanos.

Em condomínios residenciais urbanos com área de estacionamento lateral, o estacionamento convencional com vagas no solo posicionadas horizontalmente ocupam um grande espaço na construção desse tipo de moradia, normalmente, com áreas fechadas. A

utilização de um estacionamento vertical permite um reaproveitamento do espaço de área comum de um condomínio. Mas, em contrapartida, conforme a pesquisa de preços apresentado na Tabela 1, o custo de uma solução tecnológica vertical (para estacionamento de automóveis) pode ser maior do que o da área utilizada em um estacionamento convencional, uma vez que a construção de um novo edifício garagem pode possuir um custo elevado, dado a dificuldade de acesso aos materiais necessários para a sua construção no Brasil. A alternativa de construção de estacionamentos verticais permite a redução no espaço de construção do condomínio, podendo abrir locais para praças, jardim, canchas esportivas ou até construir outro prédio conforme as necessidades dos condôminos. À medida que esse tipo de estacionamento vertical se popularize, condomínios residenciais de diferentes níveis socioeconômicos podem vir a implementá-los mediante um projeto que ajuste o tipo de estacionamento vertical.

Os modelos pesquisados existentes no mercado foram do tipo carrossel de vagas, multiníveis, Engiquick Parking e o *Volkswagen AG stand parked* - Edifício garagem automatizado. Tais modelos são descritos em detalhes no Capítulo 2.

As vantagens de cada tipo de estacionamento vertical pesquisados são:

- **Estacionamento vertical tipo carrossel:** Este tipo de estacionamento ocupa em torno de três vagas de estacionamento no solo, mas pode ocupar mais vagas de forma empilhada. Além disso, tem flexibilidade para para ampliar o número de vagas devido a sua estrutura modular.
- **Estacionamento vertical tipo multinível:** As vagas nesse estacionamento podem ser movimentadas verticalmente e horizontalmente, estas movimentações agilizam o tempo em que o carro chega ao usuário.
- **Estacionamento vertical Engiquick Parking:** É um sistema de empilhamento multinível vertical de veículos que possui como vantagens a facilidade de inserir e de retirar o veículo. Além disso, permite criar estruturas de estacionamento com grande número de vagas verticais.
- ***Volkswagen AG stand parked* - Edifício garagem automatizado:** Este modelo consiste em uma estrutura cilíndrica onde cada nível vertical da estrutura contém um conjunto de vagas.

As desvantagens de cada tipo de estacionamento vertical pesquisados são:

- **Estacionamento vertical tipo carrossel:** Pelo fato das plataformas estarem instaladas nas correias, pode haver algum balanço na plataforma na hora da movimentação das vagas e o número de vagas menor comparado aos modelos pesquisados.

- **Estacionamento vertical tipo multinível:** Apesar de ser uma solução de forma vertical, pelo fato de ser multinível esta construção exige uma área horizontal maior para ser instalada.
- ***Volkswagen AG stand parked* - Edifício garagem automatizado e Estacionamento vertical Engiquick Parking:** A construção desta estrutura é complexa, utiliza uma área muito grande para ser construída.

Frente à justificativa exposta propõem-se elaborar um protótipo de um estacionamento vertical do tipo carrossel rotativo sustentado por uma estrutura metálica, sem a necessidade de uma construção predial em alvenaria. A hipótese deste trabalho é de validar o funcionamento deste tipo de estacionamento, que ocupa menor área de solo condominial se comparado ao modelo de estacionamento multinível. Esta escolha pode proporcionar a otimização dos espaços internos dos condomínios residenciais urbanos. Para tal, a seguir expõem-se os objetivos geral e específico que norteiam este projeto.

1.2 OBJETIVOS

O objetivo geral deste trabalho de conclusão de curso é elaborar um protótipo de um estacionamento vertical para avaliar o funcionamento de um sistema que permite otimizar espaços em condomínios residenciais urbanos através da implementação de estacionamento vertical do tipo carrossel.

São objetivos específicos:

a) Entender o contexto em que surge a necessidade de otimização de espaços em condomínios urbanos;
b) Realizar o levantamento bibliográfico sobre equipamentos/dispositivos tecnológicos para a construção de um estacionamento vertical do tipo carrossel;
c) Entender como funciona o estacionamento vertical no modo carrossel;
d) Construir um protótipo de estacionamento vertical do tipo carrossel.

2. REFERENCIAL TEÓRICO

Este capítulo destina-se a demonstrar o levantamento de referenciais das tecnologias que serão utilizadas para a execução do protótipo de estacionamento vertical tipo carrossel.

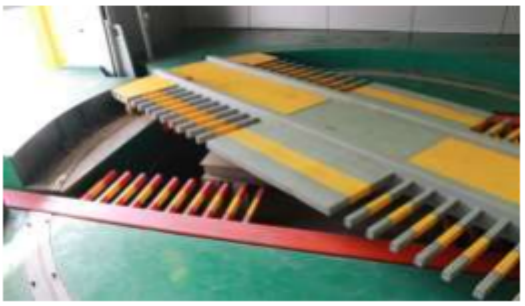
2.1 SISTEMAS DE ESTACIONAMENTO VERTICAL

Nesta seção, são apresentados alguns fabricantes e modelos de sistemas de estacionamento vertical. A empresa Engetax, com sede em São Paulo, possui uma fábrica instalada no Brasil na cidade de Campinas. Comercializa e fabrica sistemas de elevadores com aplicação para empresas e shoppings centers. Esta empresa está buscando inovar no ramo de estacionamento vertical e está produzindo uma linha nova de elevadores de carros, baseado no sistema de elevador que já produz e é mostrado na Figura 1 (ENGETAX, 2020).

Figura 1: Foto do elevador de carros da Engetax Engiquick Parking: a) carros estacionados de forma empilhados; b) entrada de um veículo na plataforma de movimentação veículos; c) plataforma de movimentação do veículo dentro do estacionamento.

a) b)

c)



Fonte: Adaptado de (ENGETAX, 2021).

A *Autostadt* (cidade dos automóveis em português) é um centro de visitantes localizado ao lado da fábrica da Volkswagen na cidade de Wolfsburg na Alemanha, com toda a estrutura principal focada em automóveis. Uma das maiores atrações deste centro são as duas torres de estacionamento vertical (*Volkswagen AG stand parked* - Edifício garagem

automatizado) conforme demonstrado na Figura 2, construídas em vidro e aço galvanizado, onde os carros são guardados automaticamente após a fabricação e depois entregues aos seus proprietários (GIGANTES DO MUNDO, 2021).

Figura 2: Torre de estacionamento vertical (*Volkswagen AG stand parked* - Edifício garagem automatizado).

Fonte: (GIGANTES DO MUNDO, 2021).

A empresa Jiu Road, Ltda, pertence ao grupo da indústria Jiu Hong. Esta empresa é uma fabricante de marcas de equipamentos eletromecânicos de estacionamentos inteligentes. Esta empresa fica localizada no condado de Gaotang, Liaocheng, província de Shandong na China (JIU ROAD PARKING,2021).

Na Figura 3 é mostrado o estacionamento vertical modelo PCX produzido pela empresa Jiu Road com dimensões de área de 6,5 m x 5,6 m de largura e alturas de 9,7 m, 11,58 m ou 13,51 m, dependendo do número de vagas de veículos, para tipos de carro sedan ou SUV, com as seguintes características: potência do motor trifásico AC de 380V de 7,5kW a 9,2kW (marca Alemanha Nord); modo de controle por PLC da marca Schneider Electric; e modo de operação por botões, ou cartão com *tag* RFID (*Radio Frequency Identification*) ou por aplicativo de telefone.

Figura 3. Imagem do estacionamento vertical rotativo da empresa Jiu Road.

Fonte: (ALIBABA , 2021a).

A empresa Roanpu Technologies é uma fornecedora de equipamentos de gerenciamento e soluções de segurança inteligente sediada em Shenzhen, na China. Esta empresa fabrica produtos e presta serviços de sistemas de gerenciamento inteligente de estacionamento, inteligência de gerenciamento de propriedades, construção de área para implementar gerenciamento inteligente de segurança (EUROPAGES, 2021).

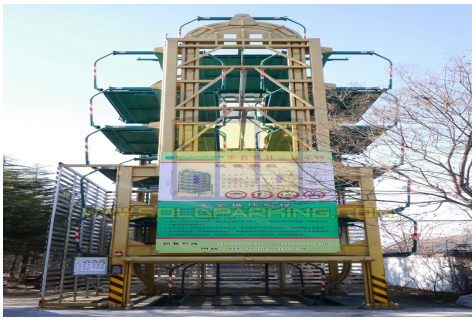
Na Figura 4 é mostrado o estacionamento vertical modelo RAP-01 com dimensões de área largura de 15,5 m de altura e 18,5 m, para tipos de carro sedan ou SUV, com as seguintes características: potência do motor trifásico AC de 380V de 5,4kW a 15kW (sem marca definida do motor no anúncio do fabricante); modo de controle por PLC (sem marca definida pelo anúncio do fabricante); e modo de operação painel com botões e cartão RFID.

Figura 4. Imagem de estacionamento vertical rotativo da Empresa Roanpu.

Fonte: (ALIBABA, 2021b).

A Solid Parking, sediada em Shandong na China, é uma empresa fornece seus elevadores estacionamento para empresas parceiras e usuários finais. Dentre seus produtos, estão elevadores de estacionamento, elevadores de vários níveis, sistema de transporte de veículos e sistemas de estacionamento automático (SOLID PARKING, 2021). Na Figura 5 é mostrado o estacionamento vertical rotativo de modelo RPS 3 com dimensões de largura de 6m a 8m e altura de 7m a 10m , para modelos de carro SUV e as seguintes características: Motor trifásico AC 380V com potência de 7,5kW (sem marca definida pelo fabricante); o modo de controle é com CLP (sem marca definida pelo fabricante) e o modo de operação é através de teclado com botões, cartão RFID e controle remoto.

Figura 5: Imagem de um estacionamento vertical rotativo da empresa Solid Park.



Fonte: (ALIBABA, 2021c).

A empresa Tada, sediada em Shandong na China, foi criada através de duas gigantes empresas de aço chinesas, Shan Steel Group e Shagang Steel Group, especializadas em pesquisa e desenvolvimento, projeto, fabricação, instalação e manutenção de equipamentos de estacionamento mecânico. A empresa Tada foi a primeira a conseguir uma licença de qualificação de fabricação no setor de estacionamento na China. Dentre seus produtos estão sistemas de estacionamento automatizado, sistema de estacionamento com empilhamento multinível conforme demonstrado na Figura 6.b), sistema de estacionamento tipo carrossel rotativo conforme demonstrado na Figura 6.a) (TADA PARKING, 2021)

Figura 6: a) Imagem de um estacionamento vertical rotativo da empresa Tada, com dimensões de largura de 6 metros e altura de 10 m ; **b)** Figura estacionamento vertical multinível empilhado com 5 colunas Tada com uma altura de 9,2 m e uma largura 2,5 m por coluna.

a) **b)**

Fonte: (ALIBABA,2021d; ALIBABA,2021e).

A Tabela 1 apresenta uma relação de valores de comercialização dos modelos de estacionamentos verticais apresentados neste trabalho, a fim de fornecer como referencial de comparação, a estimativa de valores totais ou por vaga de estacionamento.

Tabela 1: Quadro comparativo entre fabricantes e modelos de estacionamento vertical.

Fabricante	**Tipo**	**Valor de referência**	**Data da Cotação**
Engetax	Engequick Park - Estacionamento vertical	R$ 70.000,00 por vaga [1]	21 de julho de 2021
Volkswagen	Volkswagen AG stand parked - Edifício garagem automatizado	Não foi encontrado valor na referência [2]	27 de março de 2021
Jiu Hong	Estacionamento vertical rotativo	US $ 7032,83 [3]	27 de março de 2021
Roanpu	Estacionamento vertical rotativo	US $ 5525,80 [3]	27 de março de 2021
Solid Parking	Estacionamento vertical rotativo	US $ 5926,67 [3]	27 de março de 2021
Tada	Estacionamento vertical rotativo	US $ 6028,14 [3]	27 de março de 2021
Jiu Hong	Torre de soluções de gestão de estacionamento rotativo	US $ 90072,52 [3]	27 de março de 2021

Tada	Estacionamento vertical empilhado multinível	US $ 2793,14 [3]	27 de março de 2021

[1] Cotação obtida diretamente com a empresa Engetax via email.
[2] Não foi obtido valor estipulado.
[3] Cotação obtida pelo site Alibaba (2021).

2.2 IDENTIFICAÇÃO POR RADIOFREQUÊNCIA

Criado por volta dos anos 40 para a identificação de aviões militares, a identificação por radiofrequência (RFID) é uma tecnologia emergente pois permite a automação da tarefa de identificação com rapidez e segurança. Seu princípio de funcionamento é simples: um leitor envia o um sinal eletromagnético para uma etiqueta (*tag*) que, em seguida, retorna para seu número de identificação junto com outras informações, dependendo da aplicação (JUNIOR, 2010).

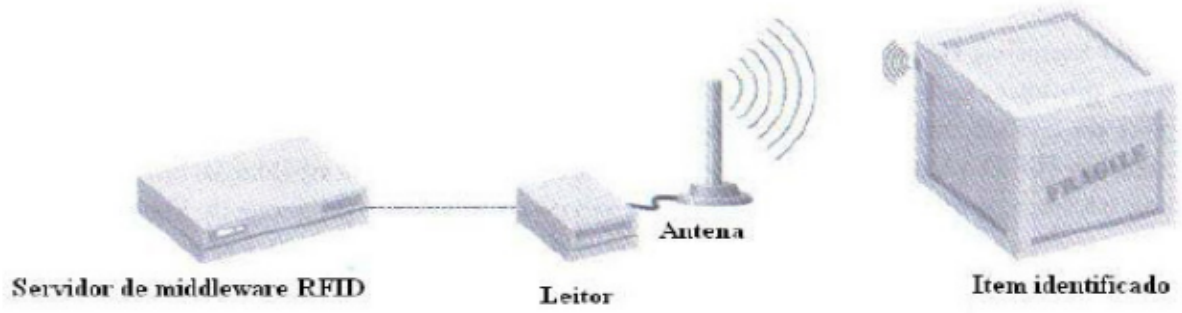
A leitura das informações contidas no dispositivo portátil é realizada por meio da utilização de um leitor (*reader*), e a comunicação entre os dispositivos acontece através de ondas de rádio frequência. Essa comunicação pode ocorrer em um ambiente onde não é necessário o contato visual nem físico entre os dispositivos.

O sistema de identificação por RFID utiliza os seguintes elementos básicos (PREDIGER; FREITAS; SILVEIRA, 2021):

- ***tag*:** é anexada a algum item ou confeccionado em cartão ou chaveiro para coletar informações;
- **leitor:** reconhece a *tag* e lê as informações contidas na mesma;
- **antena:** emite os sinais de radiofrequência para a ativação da *tag*.

Na Figura 7 será apresentado os elementos básicos de sistema com controle por RFID usados na identificação de produtos etiquetados com *tags*, qual as informações são mandadas a um servidor.

Figura 7: Componentes de um sistema RFID para registro de produtos



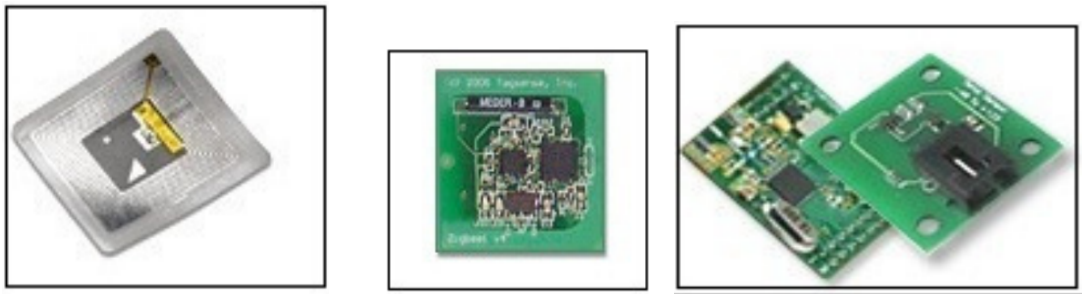
Fonte: (MALTA, 2009).

2.2.1 TIPOS DE TAGS EXISTENTES

É através das *tags* RFID que os dados são transmitidos, recebidos e enviados para o leitor RFID. As *tags* RFID são subdivididas em três principais topologias **(**IBM, 2014):

- ***tags* passivas:** uma etiqueta passiva não contém uma bateria, a energia é fornecida pelo leitor, sua imagem é demonstrada na Figura 8.a);
- ***tags* ativas:** uma etiqueta ativa é equipada com uma bateria que pode ser utilizada como uma fonte de energia parcial ou completa para o circuito da antena etiqueta, sua imagem é demonstrada na Figura 8.b);
- ***tags* semi-passivas:** não apenas contém uma bateria, mas também contém circuitos que lêem e transmitem diagnósticos de volta para seus sensores, sua imagem é demonstrada na Figura 8.c) (PREDIGER; FREITAS; SILVEIRA, 2021).

Figura 8: Tipos de *tags* RFID: a) *tag* RFID passivas; b) *tag* RFID ativas; e c) *tag* RFID semi passivas.



a) b) c)

Fonte: (IBM, 2014).

2.2.2 LEITOR RFID

Conforme Santini (2008), os leitores têm a função de comunicar-se com as *tags* RFID através de antenas, repassando as informações e em alguns casos, processando para outro sistemas. O mesmo autor afirma que todos os leitores, independente da capacidade, funcionalidade ou tipo, têm como dispositivo de entrada uma antena. Os leitores podem ser fixos ou móveis (Figura 9).

Figura 9: Imagem de um leitor RFID dos tipos (a) móvel e (b) fixo.

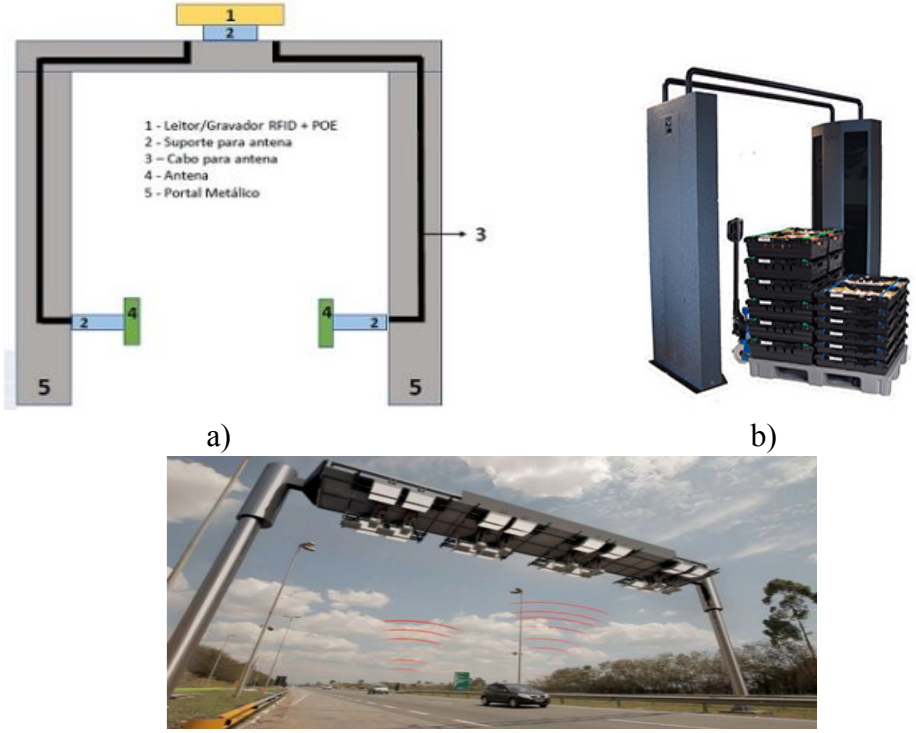
(a) (b)

Fonte: (MOTOROLA,2014).

Nos projetos envolvendo tecnologias RFID, há uma grande importância na forma em que os leitores estão sendo arranjados, pois a representação do leitor é essencial para o tipo de sistema RFID que será utilizado em um projeto. Assim, a utilização do leitor RFID irá variar conforme o tipo de sistema que o projeto irá utilizar. Portanto, haverá variações nas formas dos leitores, de tamanho e manuseio.

A seguir a representação da aplicação de alguns modelos de leitores RFID. No leitor RFID em forma de portal UHF, contendo duas antenas, as *tags* passam por um leitor RFID com antenas em forma de um portal. A Figura 10 apresenta algumas imagens deste modelo de leitor.

Figura 10: Ilustração de diferentes modelos de equipamentos para leitura de etiquetas RFID dos tipos: (a) portal com especificações das partes operantes do equipamento enumeradas no centro do desenho (BG SYSTEMS, 2020); (b) portal para identificação de produtos etiquetados com *tags* RFID (PEGASUS TECNOLOGIA, 2020); e (c) portal para identificação de veículos (BAUER, 2014 apud PREDIGER ; FREITAS; SILVEIRA, 2021).



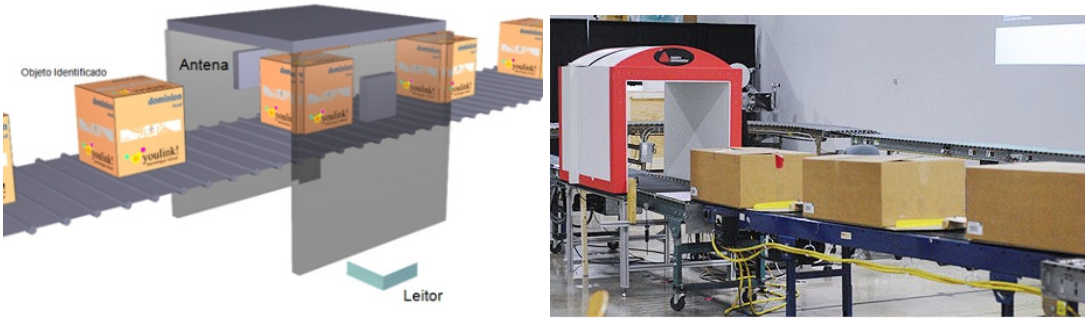
a) b)

c)

Fonte: (BG SYSTEMS,imagem a, 2021; PEGASUS TECHNOLOGY, imagem b, 2020; BAUER, 2014 apud PREDIGER ; FREITAS; SILVEIRA, imagem c, 2021).

O sistema de túnel para leitura RFID (Figuras 11a e 11b) oferece uma blindagem para a frequência de operação que será utilizada para as leituras das *tags* e o leitor.

Figura 11: Imagem ilustra a representação um (a) túnel de leitura RFID em uma esteira de produção nomeando os principais elementos que seriam o objeto identificado, a antena e o leitor; e (b) foto real de um túnel de leitura RFID em uma esteira de produção que identifica produtos já embalados em caixa.

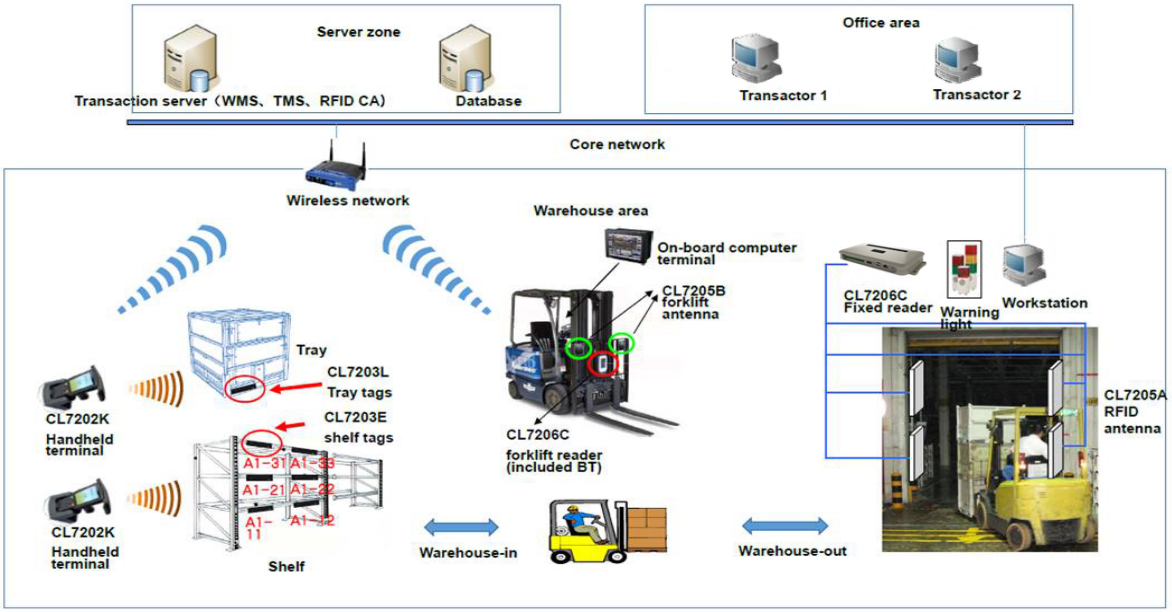


a) b)

Fonte: (OXXCODE,2014 apud PREDIGER ; FREITAS; SILVEIRA , imagem a, 2021; AVERYDENNISON, imagem b, 2020)

Os leitores de RFID portáteis (Figuras 12 e 13) são usados para fazer a leitura de *tags* usando equipamento móvel, onde o operador é uma pessoa que manuseia o equipamento manualmente para registrar as *tags* apontando para as mesmas.

Figura 12: Sistema utilizando um leitor RFID móvel para coletar dados de produtos estocados em prateleiras onde em cada seção da prateleira há uma *tag* instalada com referências do produto que está condicionado.



Fonte : (HOPELANDRFID,2021).

Figura 13: Modelo de leitor RFID manual.

Fonte : (ZEBRA, 2021).

2.2.3 IMPRESSORA DE ETIQUETAS RFID

As Figuras 14 e 15 apresentam impressoras de etiquetas RFID, que além de fazer leitura dos dados de uma etiqueta, também imprimem etiquetas RFID.

Figura 14: Modelo de impressora que imprime etiqueta RFID.

Fonte: (IMPRESSORA JATO,2021).

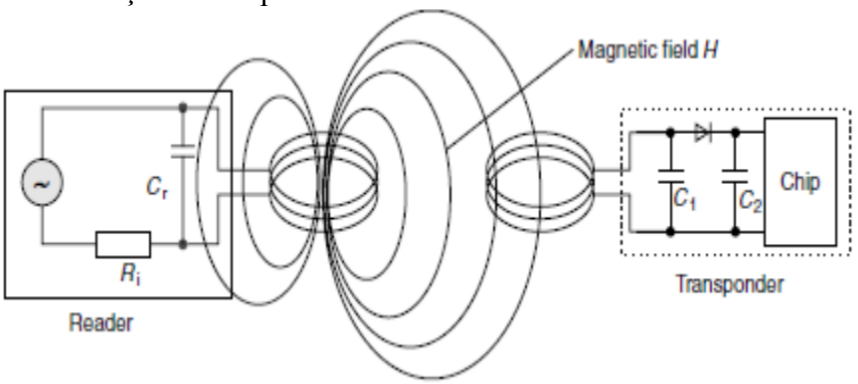
Figura 15: Imagem de uma Impressora Zebra Industrial e RFID.

Fonte: (TRADEWORK, 2020).

2.2.4 ANTENA RFID

Conforme Prediger; Freitas; Silveira, (2021) com base em Lahiri (2005), um leitor RFID se comunica com as etiquetas através de antenas do leitor. Alguns modelos de sistemas que utilizam a tecnologia RFID nos seus respectivos projetos, utilizam o leitor e antena no mesmo invólucro. Em um sistema RFID passivo com acoplamento indutivo, a energia é fornecida pela fonte de alimentação do leitor. O leitor que possui uma antena que gera um campo magnético, induz na bobina do *transponder* (*tag* RFID) uma corrente para ativar o microchip. Como o comprimento da onda (22,1 m na frequência 13,56 MHz) é, algumas vezes, maior que a distância entre o leitor e o transponder, o campo eletromagnético pode ser tratado como campo magnético simples alternado (FINKENZELLER, 2003 apud LEMAITRE 2018). Valores típicos de alcance de comunicação dependem do tamanho da etiqueta e variam de 10 cm a 70 cm (FINKENZELLER, 2003 apud LEMAITRE 2018). A Figura 16 demonstra esse sistema.

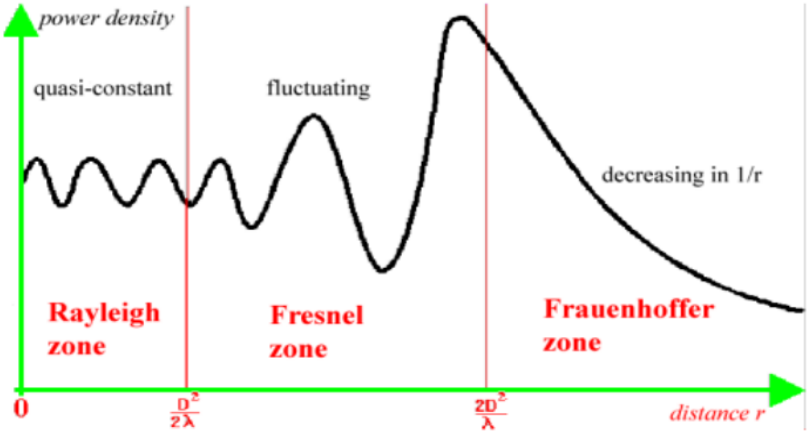
Figura 16: Ilustração do acoplamento indutivo usado nos sistemas RFID 13,56 MHz.



Fonte: (LEE, 2003 apud LEMAITRE, 2018).

A vizinhança de fonte de radiação por ondas eletromagnéticas, como no caso de antenas, pode ser dividido em 3 regiões, conforme mostrado na Figura 17 (DAOUT; SALLIN apud LEMAITRE, 2018): campo muito próximo (Zona de Rayleigh); campo próximo (Zona de Fresnel); e zona distante (Zona de Frauenhoffer). Este comparativo é utilizado para apresentar a diferença do comportamento da onda eletromagnética entre o campo próximo e o campo distante.

Figura 17: Densidade da potência em função Rayleigh, Fresnel e Frauenhoffer.



Fonte: (DAOUT; SALLIN apud LEMAITRE, 2018).

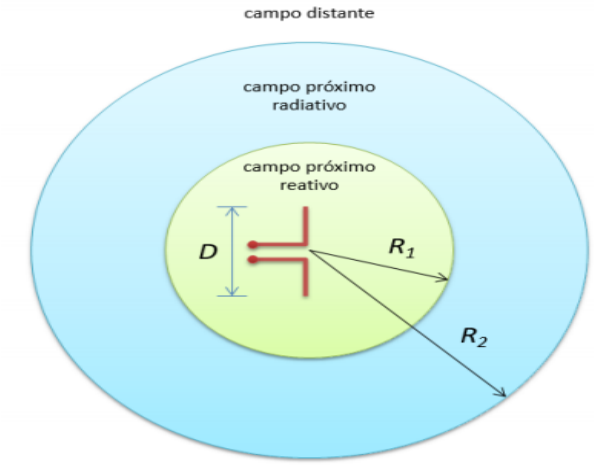
Uma zona intermediária é algumas vezes definida para descrever a transição entre os campos próximos e distantes. É possível fazer uma aproximação de raio de operação de cada função das dimensões das antenas e frequências de operação (BALANIS, 2005 apud LEMAITRE, 2018). A Figura 18 descreve os raios que delimitam cada região dada pelas equações 1 e 2.

$$R_1 = 0{,}62\sqrt{\frac{D^3}{\lambda}} \qquad R_1 = 0{,}62\sqrt{\frac{D^3}{\lambda}} \qquad (1)$$

$$R_2 = 2\frac{D^2}{\lambda} \qquad R_2 = 2\frac{D^2}{\lambda} \qquad (2)$$

Onde D é o comprimento aproximado da antena e λ o comprimento da onda.

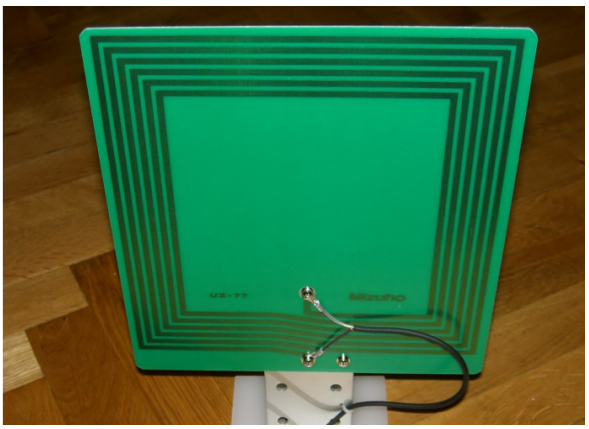
Figura 18: Região de campo próximo reativo, campo radioativo e campa distante.



Fonte: (BALANIS,2003 apud LEMAITRE, 2018).

Nos sistemas RFID de 13,56 MHz, são usados geralmente antenas *loop* que ressonam na frequência estabelecida. Esta antenas irradiam um campo magnético de campo próximo que decai r^3 e são chamadas de antenas dipolo magnética (LEE, 2003 apud LEMAITRE, 2018). A Figura 19 demonstra este sistema.

Figura 19: Antena com dipolo magnético (também conhecido como laço retangular).



Fonte: (LEE, 2003 apud LEMAITRE, 2018).

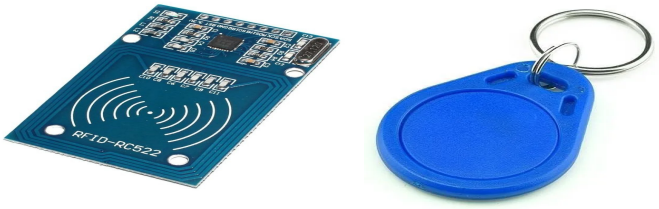
A tecnologia RFID utilizada neste projeto usará uma *tag* RFID que envia informações utilizando a tecnologia *wireless* (envio de informações sem fio) utilizando ondas de radiofrequência. A forma de comunicação do sistema é basicamente leitor e *tag* RFID que

estará contida em um chaveiro de ABS (Acrilonitrila Butadieno Estireno), ao aproximar a *tag* em forma de chaveiro do leitor, inicializará a leitura por *wireless* com uma frequência 13,56 MHz.

A Figura 20-a mostra o módulo leitor RFID RC522, o qual será utilizado neste projeto para coletar dados através de uma *tag* RFID (mostrada na Figura 20.b) que terá informações relativas à vaga do proprietário no estacionamento vertical. Estes dados serão gerenciados através da plataforma Arduino.

Figura 20: (a) módulo de leitor RFID RC522; e (b) Chaveiro RFID 13,56 MHz.

a) b)



Fonte: (MASTER WALKER,2021).

2.3 PLATAFORMA ARDUINO MEGA 2560

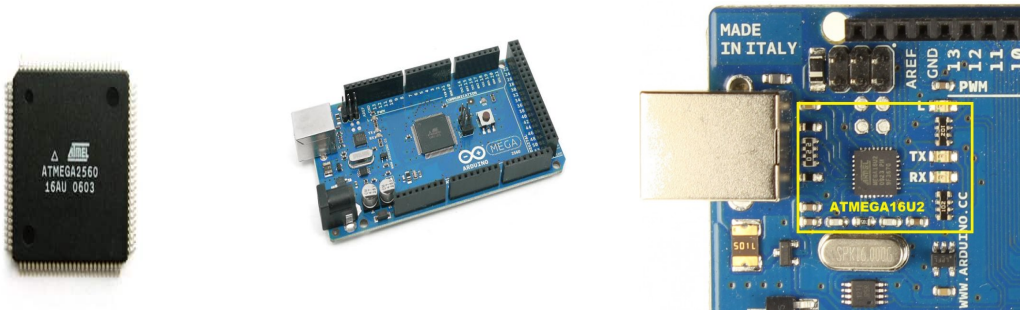
Esta seção apresenta a descrição da plataforma Arduino Mega 2560 (mostrado na Figura 21.b) e suas principais características construtivas.

2.3.1 O MICROCONTROLADOR ATMEGA 2560

O microcontrolador utilizado na plataforma Arduino MEGA 2560 é o Atmel ATMega 2560 (mostrado na Figura 21.a), um microcontrolador de 8 bits de arquitetura RISC avançada e um cristal externo de 16 MHz. Ele conta com 256 KB de memória Flash (mais 8 KB são utilizados para o *bootloader*), 8 kB de memória RAM e 4 kB de memória EEPROM. O processador do Atmega 2560, atinge uma taxa de processamento de 16 MIPS (*Mega Instructions Per Second*), operando na frequência de relógio de 16 MHz. O Atmega 2560 possui multiplicador por *hardware* e diversos periféricos que aumentam as possibilidades da plataforma Arduino baseada em Atmel ATMEGA, dentre as quais pode-se destacar 4 canais de comunicação serial, 16 entradas analógicas e 15 saídas PWM. Possui ainda comunicação SPI, I2C e 6 pinos de interrupções externas (EMBARCADOS, 2021).

Para a interface USB para a comunicação com o computador, há na placa um microcontrolador chamado de ATMega 16u2, conforme mostra a Figura 21c.

Figura 21: Imagens da Plataforma do ATMega 2560, com (a) o microcontrolador ATMega2560; a (b) plataforma Arduino Mega 2560; e o (c) microcontrolador ATMega 16u2.

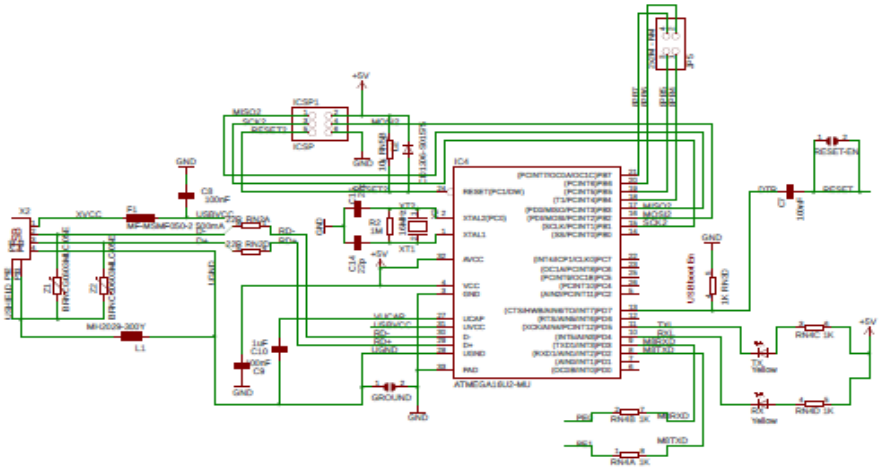


a) b) c)

Fonte: (EMBARCADOS, 2021).

O microcontrolador ATMega 16u2 possibilita fazer o *upload* do código binário gerado após a compilação do programa feito pelo usuário. Possui um conector ICSP para gravação de *firmware* através de um programador ATMEL, para atualizações futuras. Nele também estão conectados dois leds (TX, RX), controlados pelo *software* do microcontrolador, que indicam o envio e recepção de dados da placa para o computador. É interessante notar, que a conexão entre este microcontrolador com o Atmel ATMega 2560 é feita pelo canal serial (circuito demonstrado na Figura 22) desses microcontroladores. Outro ponto interessante que facilita o uso da placa Arduino é a conexão do pino 13 do ATMega 16U2 ao circuito de RESET do ATMega 2560, possibilitando a entrada no modo *bootloader* automaticamente quando é pressionado o botão *Upload* na IDE. Essa característica não estava presente nas primeiras placas Arduino onde era necessário pressionar o botão de RESET antes de fazer o Upload na IDE (EMBARCADOS, 2021).

Figura 22: Circuito da comunicação serial do Arduino 2560

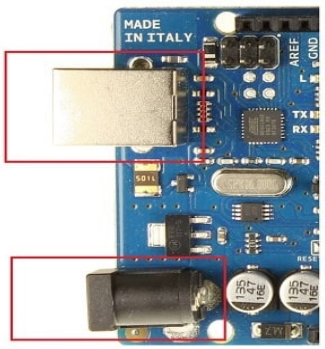


Fonte: (STORE ARDUINO, 2021).

2.3.2 ALIMENTAÇÃO ELÉTRICA DO ATMEGA 2560

A alimentação da placa Arduino Mega é feita tanto pela USB como por uma alimentação externa. Na Figura 23 são apresentados os conectores para alimentação.

Figura 23: Alimentação elétrica pode ser conectada na conexão da porta USB (conector superior da Figura), quanto na fonte externa pelo plug jack p4 (conector inferior).

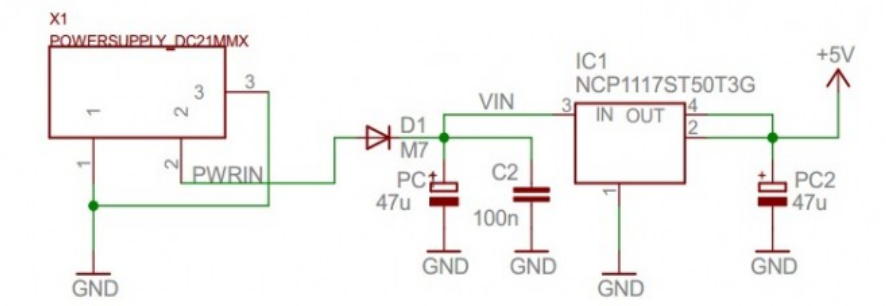


Fonte: (EMBARCADOS, 2021).

A alimentação externa é feita através do conector jack P4 com o terminal positivo no centro, onde o valor de tensão da fonte externa deve estar entre os limites 6 V a 20 V. Porém, se alimentada com uma tensão abaixo de 7 V, a tensão de funcionamento da placa, que no Arduino MEGA 2560 é de 5 V, pode ficar instável. Quando alimentada com tensão acima de 12V, o regulador de tensão da placa pode superaquecer e danificar a placa. Desta forma, é recomendado para tensões de fonte externa valores de 7 V a 12 V. O circuito integrado

responsável pela regulação de tensão é o OnSemi NCP1117, com esquemático mostrado na Figura 24 (EMBARCADOS, 2021).

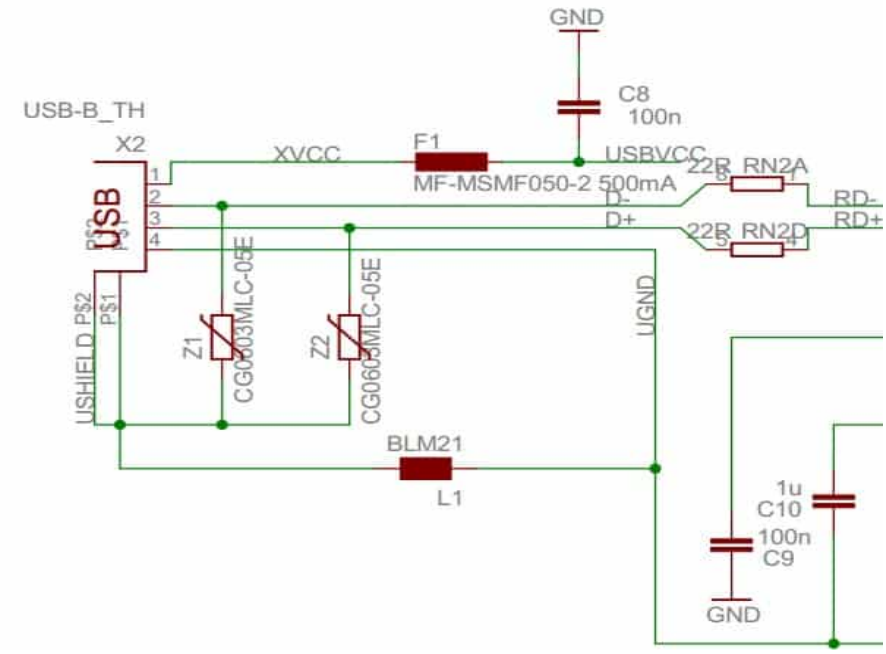
Figura 24: Circuito regulador de tensão do Arduino 2560



Fonte: (EMBARCADOS, 2021).

Quando o cabo USB é ligado a um PC, a tensão não precisa ser estabilizada pelo regulador de tensão, desta forma a placa é alimentada diretamente pela USB, como é mostrado na Figura 25. O circuito da USB apresenta alguns componentes que protegem a porta USB do computador em caso de alguma anormalidade. Na Figura 25 é exibido o circuito de proteção da USB da placa Arduino MEGA 2560 (EMBARCADOS, 2021).

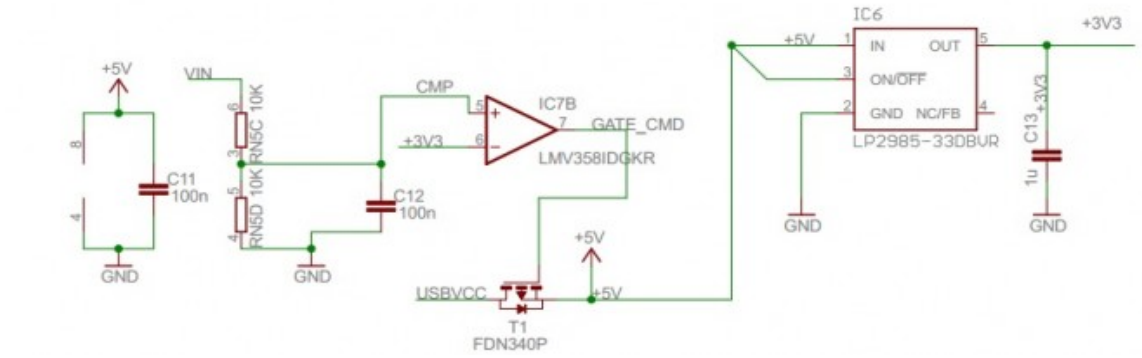
Figura 25: Circuito de proteção da porta USB do Arduino 2560



Fonte: (EMBARCADOS, 2021).

Na Figura 25, os dois varistores (Z1 e Z2) podem suportar picos elevados e potências elevadas de transientes. Os resistores de 22 ohms (RN2A e RN2D) limitam uma corrente resultante de alguma descarga elétrica eventual de um usuário em contato com o conector USB protegendo, desta forma, os pinos do microcontrolador. O fusível resetável (F1) de 500 mA impede que a porta USB do computador seja daficada caso ocorra algum problema de projeto ou uma falha no circuito e ultrapasse a corrente de 500 mA quando a placa estiver conectada ao PC. O indutor de ferrite L1 foi incluído no circuito para que ruídos da USB externa não entrem no circuito da placa Arduino através do terra. A placa também conta com um circuito para comutar a alimentação automaticamente entre a tensão da USB e a tensão da fonte externa. Esse circuito é apresentado na Figura 26. Caso haja uma tensão no conector DC e a porta USB é conectada, a tensão de 5V será proveniente da fonte externa e a porta USB servirá apenas para comunicação com o PC (EMBARCADOS, 2021).

Figura 26: Circuito selecionador da fonte do Arduino 2560



Fonte: (EMBARCADOS, 2021).

2.3.3 ESPECIFICAÇÕES DOS TERMINAIS DO ARDUINO

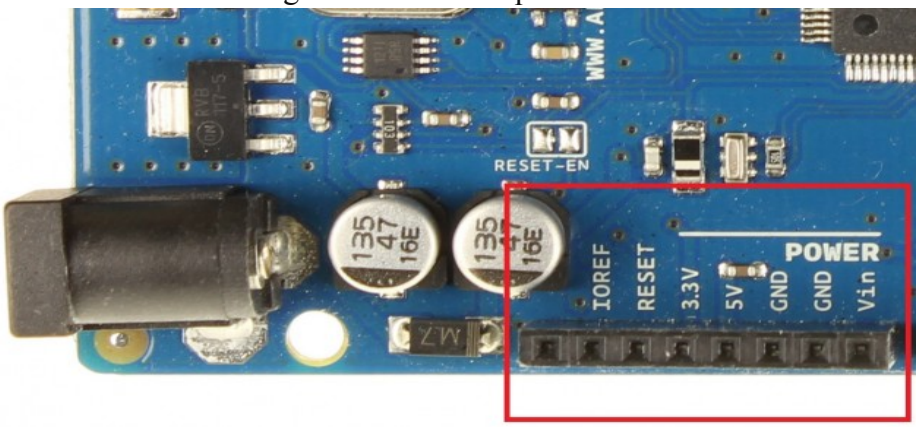
Nesta seção são apresentadas as especificações dos terminais do Arduino Mega 2560, que são utilizados para alimentação elétrica dos módulos e shields pela pela própria plataforma Arduíno Mega 2560 e especificação dos terminais analógicos e digitais com suas respectivas funções. As especificações dos terminais RESET, 3,3V, 5V, GND, VIN, IOREF, como pode ser visto no detalhe da Figura 27, são:

- **RESET** – pino conectado a pino de RESET do microcontrolador. Pode ser utilizado para um reset externo da placa Arduino;
- **3,3 V**. – fornece tensão de 3,3V. para alimentação de shield e módulos externos.

Corrente máxima de 50 mA;

- **5 V** – fornece tensão de 5 V para alimentação de shields e circuitos externos;
- **GND** – pinos de referência, ground, terra;
- **VIN** – pino para alimentar a placa através de shield ou bateria externa. Quando a placa é alimentada através do conector Jack, a tensão da fonte estará nesse pino;
- **IOREF** – fornece uma tensão de referência para que shields possam selecionar o tipo de interface apropriada. Dessa forma, *shields* que funcionam com a placas Arduino que são alimentadas com 3,3V podem ser adaptadas para ser utilizados em 5V e vice-versa

Figura 27: Conectores de alimentação para conexão de shields e módulos na placa Arduino Mega 2560 em destaque.



Fonte: (EMBARCADOS, 2021).

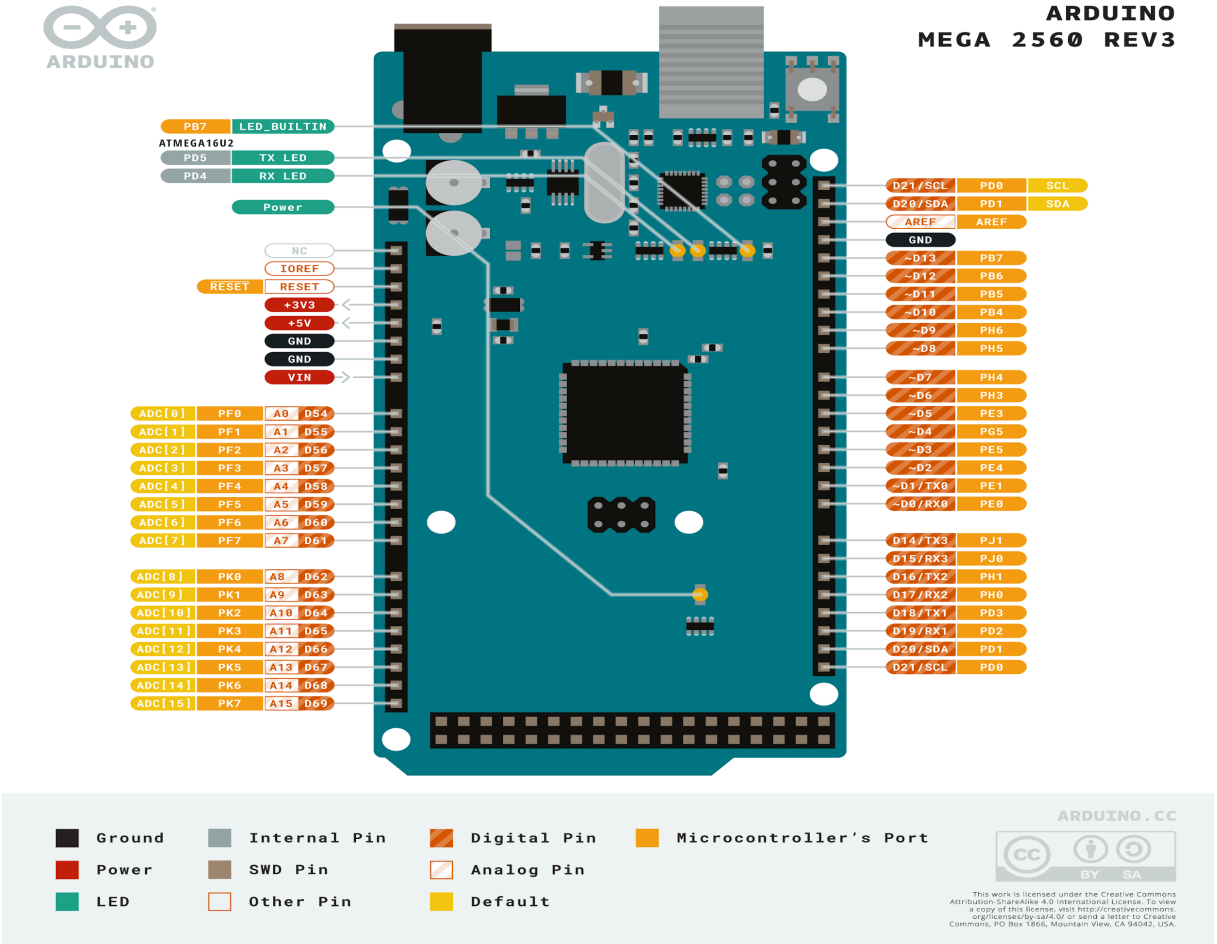
O Arduino Mega 2560 possui 54 pinos digitais que podem ser usados como entradas ou saídas, dependendo da necessidade do projeto, configurando no escopo do código do projeto que será executado. Os pinos operam com tensão máxima de 5 V e podem fornecer ou drenar até 40 mA. Cada pino possui resistor interno de *pull-up* que pode ser habilitado pelo software. Alguns desse pinos possuem funções especiais como descritos a seguir:

- Comunicação Serial – Serial 0 (RX) e 1 (TX); Serial 1: 19 (RX) e 18 (TX); Serial 2: 17 (RX) e 16 (TX); Serial 3: 15 (RX) e14 (TX). Os pinos 0 e 1 estão conectados aos pinos do ATmega16U2 responsável pela comunicação USB;
- Interrupções externas – 2 (interrupt 0), 3 (interrupt 1), 18 (interrupt 5), 19 (interrupt 4), 20 (interrupt 3), and 21 (interrupt 2). Estes pinos podem ser configurados para disparo da interrupção tanto na borda de subida ou descida, ou em níveis lógicos alto ou baixo, conforme a necessidade do projeto;
- PWM: os pinos 2 a 13 e 44 a 46 podem ser utilizados como saídas PWM. O sinal PWM possui 8 bits de resolução;

- Comunicação SPI: Pinos: 50 MISO (Master In Slave Out), 51 MOSI (Master Out Slave In), 52 SCK (Serial Clock), 53 SS (Slave Select). Estes pinos estão ligados ao conector ICSP;
- Comunicação I2C: TWI *(two-wired interface)*: pinos 20 SDA (serial data) e 21 SCL (Serial clock).

A plataforma Arduino Mega 2560 possui 16 entradas analógicas (pinos A0 a A15), onde pode ser feita a conversão com uma resolução de 10 bits, ou seja, o valor será convertido entre 0 e 1023. Por padrão, a tensão de referência é conectada a 5V. Porém é possível mudar o valor de referência através do pino AREF e no escopo do código do Arduino (EMBARCADOS, 2021). A especificação completa dos terminais do Arduino Mega 2560 pode ser vista na Figura 28.

Figura 28: Especificação completa dos terminais do Arduíno Mega 2560.



Fonte: (STORE ARDUINO, 2021).

Devido à características deste componente, que contém 54 pinos digitais e distribuídos nas conexões laterais da placa, utilizou-se este modelo para o protótipo de estacionamento

vertical (Arduino Mega 2560), pois possibilita instalar o número de ligações dos módulos e sensores necessários, uma vez que possui conexões na lateral da placa, possibilitando uma melhor distribuição de espaços entre as conexões na placa.

2.3.4 A MEMÓRIA NÃO VOLÁTIL DO ATMEGA

No projeto do estacionamento vertical automatizado, os códigos das *tags* de RFID que estão associados para acessar as vagas de estacionamento deverão permanecer armazenados em uma memória não volátil, para que a informação não se perca com a falta de energia elétrica. Uma forma de prevenir a perda desta informação é usar a memória EEPROM interna do microcontrolador ATmega, que tem uma tolerância de 100.000 ciclos de escrita e leitura e capacidades que variam de 1024 bytes, no ATmega328P, até 4096 bytes, no ATmega2560 (ARDUINO, 2020). Para usar o acesso à EEPROM, deve-se incluir o suporte para a biblioteca “EEPROM.h”, usando a declaração #include <EEPROM.h>. A escrita na EEPROM é feita usando o comando “EEPROM.write”, com as informações de endereço de escrita e o valor a ser armazenado no endereço, como é mostrado assim: **EEPROM.write(endereço, valor)**.

A variável “endereço” indica a posição onde o conteúdo da variável “valor” será armazenada. A variável “valor” deve estar compreendida entre o intervalo de 0 a 255, pois a EEPROM somente é capaz de armazenar *bytes*. O tempo necessário para realizar uma escrita na EEPROM é de 3,3 ms . Como forma de reduzir a necessidade de fazer ciclos de escrita na EEPROM, se pode utilizar a função “EEPROM.update”, que somente realiza a atualização da informação gravada em um determinado endereço, ou seja, faz escrita somente caso o dado armazenado seja diferente do dado a ser escrito. A declaração é mostrada assim: **EEPROM.update(endereço, valor)**.

A leitura de dados armazenados na EEPROM é feita usando o comando “EEPROM.read”, com o endereço a ser lido, como é mostrado desta forma: **EEPROM.read(endereço)**.

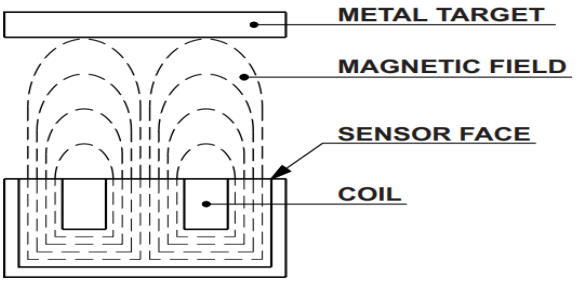
Quando é necessário trabalhar com tipos de dados diferentes de byte e que ocupem uma quantidade maior de posições da EEPROM para gravar a informação, deve-se utilizar as funções “EEPROM.put()” e “EEPROM.get()”. Tais funções permitem manipular na EEPROM diferentes tipos de dados, tais como *char* e *float* (EEPROM, 2020). A função “EEPROM.put()” aliada ao uso de estruturas de dados permite facilmente trabalhar com o

registro dos identificadores únicos das *tags* de RFID que estão associadas às vagas do estacionamento.

2.4 SENSOR INDUTIVO DE PROXIMIDADE

Os sensores indutivos são usados para detectar objetos metálicos. O princípio de funcionamento deste sensor na variação de campo magnético se dá quando um material metálico se aproxima da face deste sensor, este é o momento em que ocorre a variação do campo magnético. O princípio de funcionamento do sensor indutivo de proximidade baseia-se na geração de um campo eletromagnético de alta frequência, que é desenvolvido por uma bobina ressonante instalada na face sensora. A bobina faz parte de um circuito oscilador que em condição normal (desacionada) gera um sinal senoidal. Quando um metal aproxima-se do campo, por correntes de superfície, absorve a energia do campo, diminuindo a amplitude do sinal gerado no oscilador. A variação de amplitude deste sinal é convertida em uma variação contínua que comparada com um valor padrão, passa a atuar no estágio de saída. A Figura 29, demonstra um objeto metálico em frente à face do sensor indutivo (SENSE, 2021).

Figura 29: Ilustração de um objeto metálico em frente à face do sensor indutivo.

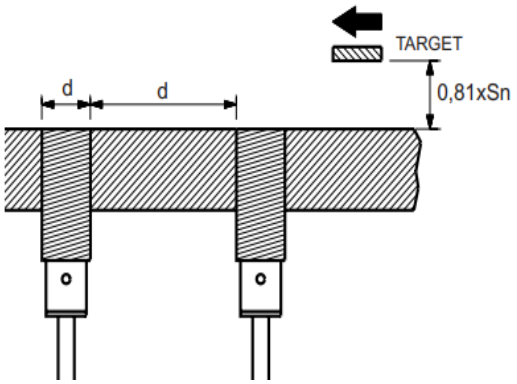


Fonte: (SENSE, 2021).

2.4.1 TIPOS DE SENSORES INDUTIVOS

Embutido: Este tipo de sensor tem o campo eletromagnético emergindo apenas na face sensora e permite que seja montado em uma superfície metálica conforme sua imagem é ilustrada na Figura 30.

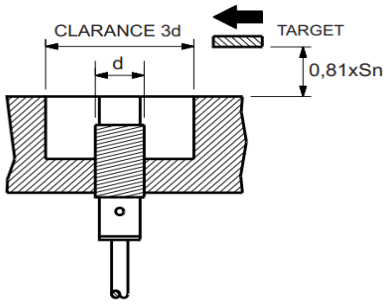
Figura 30: Ilustração sensor indutivo do tipo embutido.



Fonte: (SENSE, 2021).

Não embutido: Neste tipo o campo eletromagnético emerge também da superfície lateral da face sensora, sensível à presença de metal ao seu redor conforme sua imagem é ilustrada na Figura 31.

Figura 31: Ilustração sensor indutivo do tipo não embutido.



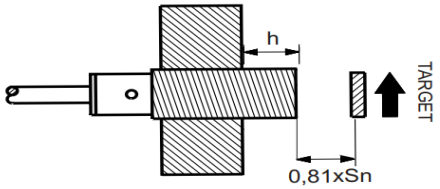
Fonte: (SENSE, 2021).

Semi embutido: Neste tipo o campo eletromagnético emerge também da superfície lateral da face sensora, sensível à presença de metal ao seu redor. O campo eletromagnético emerge somente na face sensora mas é afetado por metais próximos a sua face, podendo ser instalado em superfícies metálicas desde que obedeça uma distância livre a partir da superfície sensora, sua ilustração pode ser vista na Figura 32. Esta distância varia conforme mostrado na Tabela 2.

Tabela 2: Variação da distância que afeta o campo eletromagnético que emerge sobre a face sensora conforme o diâmetro do sensor e distância sensora nominal.

Sn (mm)	Diâmetro	Distância (h)
2	M8 x 1	0
4	M12 x 1	0,5
8	M18 x 1	2
15	M30 x 1,5	3

Figura 32: Ilustração sensor indutivo do tipo semi embutido.

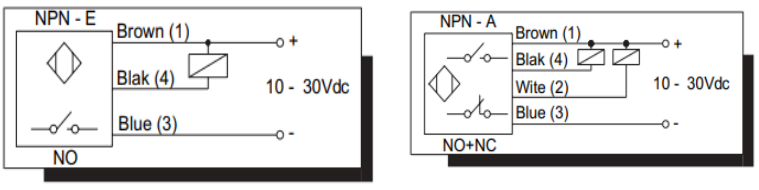


Fonte: (SENSE, 2021).

2.4.2 CONFIGURAÇÃO ELÉTRICA DOS SENSORES

Configuração NPN: São sensores que possuem no estágio de saída um transistor que tem função de chavear (ligar e desligar) o terminal negativo da fonte, que as configurações podem ser do arranjadas em normalmente aberto conforme ilustra a Figura 33.a) e normalmente fechado conforme ilustra a figura 33.b).

Figura 33: a) Sensor indutivo NPN no arranjo normalmente aberto; b) Sensor indutivo NPN no arranjo normalmente fechado.

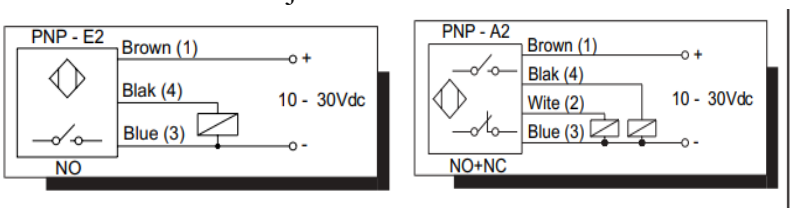


a) **b)**

Fonte: (SENSE, 2021).

Configuração PNP: São sensores que possuem no estágio de saída um transistor que tem função de chavear (ligar e desligar) o terminal positivo da fonte, que as configurações podem ser do arranjadas em normalmente aberto conforme ilustra a Figura 34.a) e normalmente fechado conforme ilustra a Figura 34.b).

Figura 34: a) Sensor indutivo PNP no arranjo normalmente aberto; b) Sensor indutivo PNP no arranjo normalmente fechado.



Fonte: (SENSE, 2021).

O modelo de sensor indutivo que é utilizado no projeto em questão é no arranjo pnp e normalmente aberto conforme mostra a Figura 35:

Figura 35: Sensor indutivo PNP no arranjo normalmente aberto da marca TPCID e modelo Lj12A3-4Z/BY.

Fonte: (MERCADO LIVRE, 2021).

2.5 MOTOR DE PASSOS

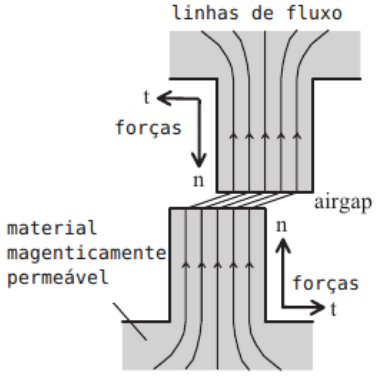
O motor de passo pode ser visto como um motor elétrico sem comutadores. Tipicamente, todos os enrolamentos são parte do estator, e o rotor é ou um ímã permanente, ou, no caso de motores de relutância variável, um bloco dentado de algum material magneticamente permeável. Toda a comutação deve ser feita externamente pelo controlador do motor, e tipicamente, os motores e controladores são projetados para que o motor fique fixo em uma posição ou rotacione em um sentido ou outro de maneira precisa com velocidades ou acelerações controladas (NETO, 2011).

A propriedade essencial do motor de passo é de transformar mudanças chaveadas da excitação em incrementos precisos da posição do rotor. Eles são categorizados como máquinas de dupla saliência, o que significa que eles têm dentes de material magneticamente permeáveis em ambas as partes estacionária e rotatória (NETO, 2011).

O fluxo magnético cruza o entreferro entre as saliências das duas partes do motor. De acordo com o tipo de motor, o fluxo pode ser proveniente de um imã permanente ou de uma bobina por onde passa uma corrente ou uma combinação dos dois. Todavia, o efeito é o

mesmo: os dentes sofrem forças iguais e opostas, que tentam alinhá-los e diminuírem o entreferro entre eles. A principal componente desta força, é a força normal (n), está tentando fechar o entreferro, mas para os motores elétricos, o componente de força mais útil é a força tangencial (t), que está tentando mover os dentes lateralmente entre si. A Figura 36, mostra os componentes de força entre dois dentes magneticamente permeáveis (NETO, 2011).

Figura 36: Componentes de força entre dois dentes magneticamente permeáveis.



Fonte: (NETO, 2011).

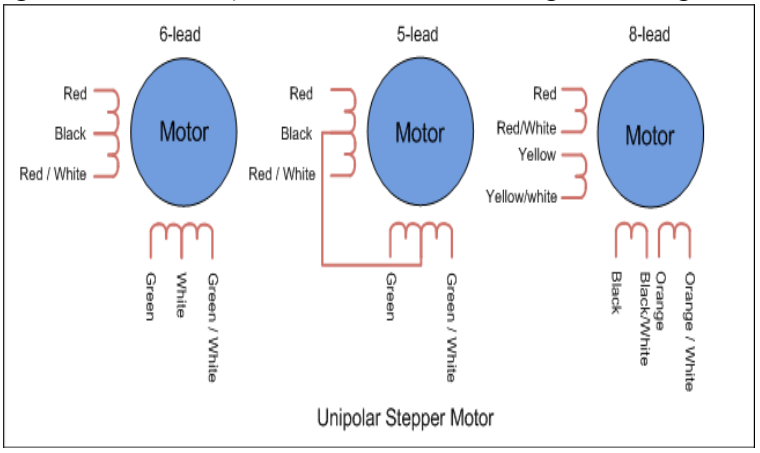
O motor de passo vai converter pulsos elétricos em movimento mecânico de rotação. A rotação do eixo do motor é caracterizada pelo ângulo incremental de passo para cada pulso de excitação. Esse ângulo incremental é repetido precisamente a cada pulso, gerado por um erro geralmente menor que 5 %, sendo este erro não acumulativo. O resultado é preciso e de movimento fixo, sendo que a cada pulso tem-se o movimento de um único ângulo incremental de passo, o que possibilita um eficiente controle de posição. Se o motor for corretamente dimensionado, obtém-se um motor que não depende da carga, desde que ela imponha um torque sempre menor que o torque do motor. O circuito excitador é constituído por um circuito sequencial (controlador) e um estágio amplificador de saída. O circuito sequencial pode ser projetado para que o motor gire com diferentes modos de acionamento (NETO, 2011).

2.5.1 POLARIZAÇÕES DE MOTORES DE PASSO

Monopolar: Nos motores de passo unipolares são usados dois enrolamentos por fase e costumam ter um contato em comum, resultando em cinco, seis ou oito conexões. Nos modelos onde a conexão comum dos dois pólos é separada, são seis conexões externas como demonstra na Figura 37.a e nos modelos onde a conexão comum é soldada internamente, são cinco conexões externas como demonstra na Figura 37.b. Os de oito conexões externas contêm a conexão em comum dos dois pólos separados facilitam a ligação em série ou

paralela das bobinas como demonstra a Figura 37.c. Eles são chamados de unipolares e facilitam o projeto por não necessitar de ligação reversa nos polos, conforme demonstra a Figura 37 (ENGINEERS GARAGE, 2021).

Figura 37: a) Enrolamento motor de passos unipolar de 6 fios; b) Enrolamento motor de passos unipolar de 5 fios; c) Enrolamento motor de passos unipolar de 8 fios.

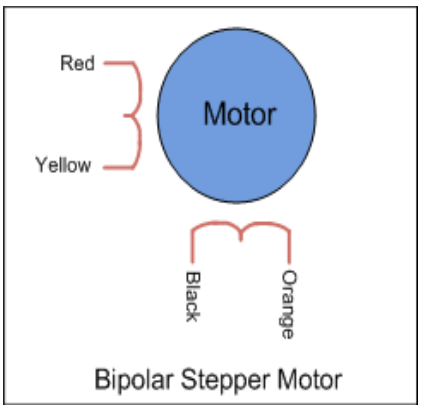


a) b) c)

Fonte: (ENGINEERS GARAGE, 2021).

Bipolar: Usam uma ligação por polo e necessitam que o circuito de controle possa reverter o sentido da corrente para acionar as bobinas de forma correta conforme demonstrado na Figura 38.

Figura 38: Arranjo das bobinas de motores de passo bipolar.



Fonte: (ENGINEERS GARAGE, 2021).

Para o controle suave do motor de passos deste projeto, será usado um *driver* que contém um chip A4988 no qual sua imagem é demonstrada na Figura 39. Uma característica deste driver é que ele foi fabricado para o controle de motor de passos bipolar, sendo que para o controle de motores unipolar este driver não se aplica. Para controlar motores bipolares, este *driver* utiliza transistores de efeito de campo (FET) DMOS arranjados na forma de 2 pontes

H. Com este *driver* é possível controlar o motor de passos bipolar em 5 modos de operação, que são configuráveis em passo completo (*full step*), meio passo (*half step*), um quarto de passo (*quarter step*), um oitavo de passo (*Eighth step*) e um dezesseis-avos de passo (*Sixteenth step*). Na Tabela 3, demonstra-se como funciona o nível lógico para cada um dos seus modos.

Figura 39: Imagem *driver* A4988

Fonte: (3DLAB, 2021).

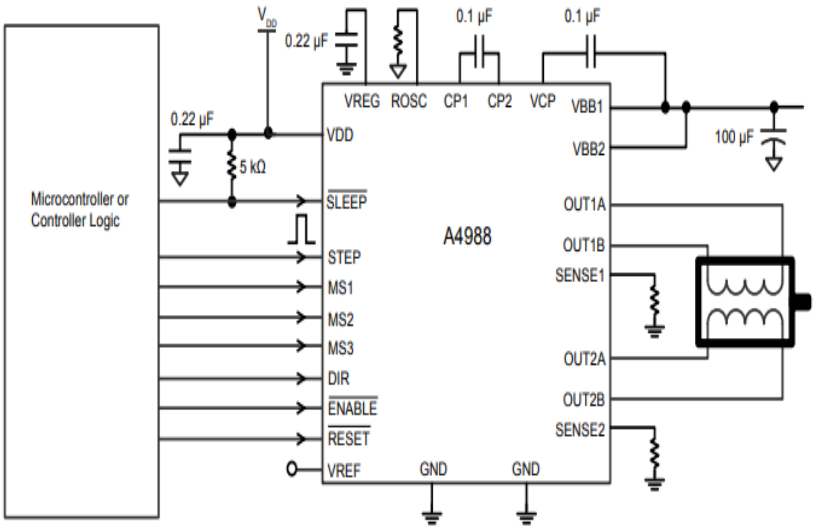
Tabela 3: Tabela verdade para configuração do modo de operação do tipo de passo do motor de passos.

MS1	MS2	MS3	Resolução do micro passo	Modo de excitação
L	L	L	Passo completo	2 Fase
H	L	L	Meio passo	1-2 Fase
L	H	L	Um quarto de passo	W1-2 Fase
H	H	L	Um oitavo de passo	2W1-2 Fase
H	H	H	Um dezesseis avos de passo	4W1-2 Fase

Fonte: (POLOLU, 2021).

Na Figura 40 é mostrado o diagrama elétrico do *driver* A4988:

Figura 40: Diagrama elétrico do *driver* A4988.



Fonte: (POLOLU, 2021).

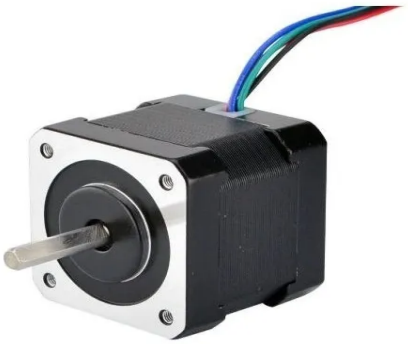
A função dos pinos de interface entre o microcontrolador e o A4988, está descrito abaixo:

- **STEP:** este terminal é responsável pelo avanço do passo do motor;
- **DIR:** este terminal é responsável pelo controle do sentido de rotação do motor de passos;
- **ENABLE:** este terminal é responsável por habilitar os *drivers* do motor, sendo que para um nível baixo os drivers são habilitados e para um nível alto os *drivers* são desabilitados;
- **RESET:** este terminal é responsável por resetar *drivers* do *chip* na saída, sendo que em nível baixo ele é habilitado e em nível alto ele é desabilitado;
- **SLEEP:** este terminal é responsável por limitar a habilitação de alguns circuitos internos para poupar energia do consumo da placa, sendo que para sua habilitação deverá receber um nível lógico baixo e para a sua desabilitação necessitará de um sinal de nível lógico baixo;
- **MS1, MS2 E MS3:** são os terminais responsáveis por controlar o modo de passos, quando arranjados em ligação lógica (POLOLU, 2021).

O motor de passos utilizado no protótipo desenvolvido neste projeto é o Stepperonline

17HS19-2004S1, mostrado na Figura 41.

Figura 41: Motor de passos Stepperonline 17HS19-2004S1.



Fonte: (MERCADO LIVRE, 2021)

Suas especificações elétricas e mecânicas são (OMC STEPPERONLINE, 2021):

- **número de fases:** 2;
- **corrente/fase:** 2 ampéres;
- **resistência por fase/ 25°:** 1,40 Ω ± ± 10%;
- **indutância por fase/1KHz:** 3 mH ± ± 10%;
- **torque de retenção:** 5,9 Kg.f/cm
- **ângulo de passo:** 1,8°;
- **inércia do rotor:** 82g/cm²;
- **Peso:** 0,40 kg.

2.6 SERVO MOTORES

Os servo motores são motores que podem ser controlados pelo ângulo da posição do seu eixo em uma faixa de 0° a 180° ou, em algumas literaturas, de 180° a 270° (EE, 2021). O controle do ângulo de posição do eixo deste motor é feito por sinal de largura de pulso PWM gerado por um microcontrolador ou algum circuito digital de controle. Este motor possui uma característica de movimentar o seu eixo até uma posição e mantê-la, mesmo quando sofre uma força contrária a direção controlada (FEIS, 2021).

O modelo de servo motor que será utilizado neste projeto é o Dsservo Ds3218mg conforme mostrado na Figura 42, suas características estão apresentadas na Tabela 4.

Figura 42: Dsservo Ds3218mg.

Fonte: (MERCADO LIVRE, 2021).

Tabela 4: Especificações servo motor Dsservo Ds3218mg.

Torque mínimo	Com 5V, 19 kg.f/cm²
Torque máximo	Com 6,8V, 21,5 kg.f/cm²
Banda morta	3µs
Velocidade mínima	0,16s/60° com 5V
Velocidade mínima	0,14s/60° com 6,8V
Operação de tensão elétrica	4,8V a 6,8V
Tipo de motor	CC
Transmissão	Ângulo de 270° ou ângulo de 180°
Tipo de engrenagem	Cobre e alumínio
Frequência de trabalho	1520µs/333Hz

Fonte: (MERCADO LIVRE, 2021).

2.7 SENSOR DE BARREIRA INFRAVERMELHO REFLEXIVO

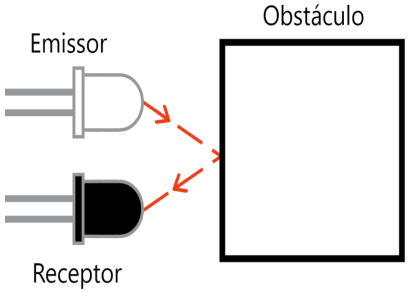
Este módulo que sua imagem é demonstrada na Figura 43, consiste na reflectância entre dois leds que utilizam a tecnologia infravermelho que estão instalados no módulo um ao lado do outro, sendo o de cor transparente o emissor e o de cor preta o receptor. Para que a luz infravermelha chegue ao receptor, é necessário que um obstáculo se aproxime dos dois leds

infravermelhos para que a luz seja refletida no objeto em direção ao receptor como mostrado na Figura 44; quanto mais distante o objeto, haverá menos reflexão, sendo que a cor do objeto também terá influência na intensidade da reflexão (MUNDO PROJETO, 2021).

Figura 43: Sensor de obstáculo infravermelho reflexivo.

Fonte: (BAÚ DA ELETRÔNICA, 2021).

Figura 44: Ilustração da reflexão da luz infravermelha partindo do emissor refletindo em um obstáculo em direção ao receptor.

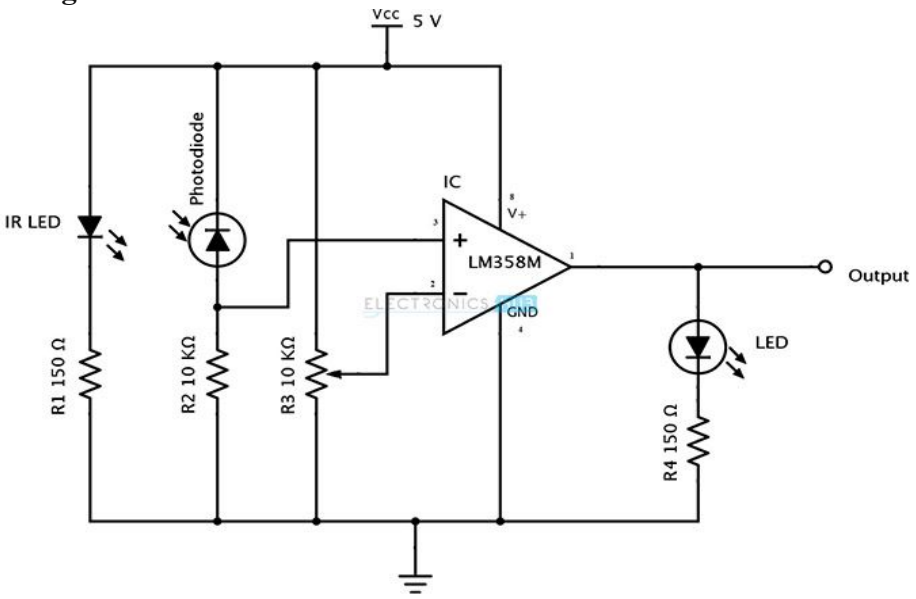


Fonte: (MUNDO PROJETO, 2021).

2.7.1 MÓDULO SENSOR DE OBSTÁCULO INFRAVERMELHO REFLEXIVO

Para ligar em um controlador digital, este módulo tem 3 terminais: VCC, GND e OUT. Este último é usado como sinal de controle. No circuito do módulo (Figura 45) há um comparador composto pelo circuito integrado LM393 que é ligado ao receptor e ligado a uma tensão de referência, ajustada por um potenciômetro *trimpot*. O terminal da saída indica nível alto para o caso do nível de tensão do receptor ser maior do que a tensão de referência e indica nível baixo para o caso do receptor tenha nível de tensão de referência (MUNDO PROJETO, 2021).

Figura 45: Circuito do módulo do sensor de obstáculo reflexivo.



Fonte: (ELETRONICS HUB, 2021).

Com base no referencial trazido no capítulo, pretendeu-se apontar os instrumentos e tecnologia necessária para o desenvolvimento de um projeto tecnológico em eletrônica industrial. A proposta do projeto em tela visa a equipar condomínios residenciais, especialmente em áreas urbanas, com garagens verticais, do tipo carrossel, a fim de garantir uma maior otimização de espaços internos em áreas condominiais urbanas. Para tal, o capítulo que segue apresenta como foi o desenvolvido o protótipo de base experimental.

3. DESENVOLVIMENTO DO PROJETO

Neste capítulo se apresenta o desenvolvimento de um protótipo de garagem vertical do tipo carrossel, a partir do detalhamento de seus componentes e de seu funcionamento prático. Com este experimento, objetiva-se entender e demonstrar o funcionamento de um sistema de estacionamento vertical. O método experimental pretende a realização de manipulações preestabelecidas observando os seus resultado/efeitos tendo em vista a descobrir conexões causais e atingir a sua demonstrabilidade (FACHIN,2017).

3.1 ESPECIFICAÇÕES DO PROTÓTIPO

Dentre os modelos de estacionamento vertical existentes no mercado, o modelo que ficou definido para este projeto foi o estacionamento vertical rotativo do tipo carrossel. Como o nicho para a implementação deste projeto ficou definido para condomínios residenciais, o modelo de estacionamento vertical rotativo tipo carrossel, é a melhor escolha pelo fato de sua estrutura ser construída com perfis metálicos sem a necessidade de construir paredes de concreto com a finalidade de erguer a estrutura, ganhando muito mais espaço e tempo para sua construção.

Para a construção do protótipo, definiu-se as seguintes etapas e materiais:

- Montagem de uma estrutura metálica, com suportes de sustentação para correntes e rodas dentadas, para comportar quatro vagas;
- Montagem de um sistema de freio;
- Instalação de um motor de passo para acionamento do carrossel de vagas;
- Instalação de sensores indutivos e de reflexão para controle de parada e segurança;
- Montagem do controle eletrônico de acionamento, usando Arduíno Mega;
- Programação e testes de funcionamento do sistema.

3.2 MONTAGEM DA ESTRUTURA METÁLICA

Nesta seção, é apresentada a descrição da montagem de uma estrutura metálica, com suportes de sustentação para correntes e rodas dentadas, para comportar quatro vagas. Para montar a maquete foram utilizados perfis quadrados de alumínio de largura (50,8 mm x 50,8

mm) e com a sua espessura de (1,58 mm). Para a montagem estrutural foram utilizados 4 barras de 650 mm, 2 barras de 285 mm e 4 barras de 190 mm. Estas barras foram unidas por 44 cantoneiras latonadas de largura 20,5 mm e 40 mm em cada face e também outras 8 para fixar a estrutura a plataforma. Para fixação das cantoneiras foram utilizado 176 parafusos de fenda de zinco com cabeça escareada com rosca de 3/16'' e comprimento de 3/4'', 4 parafusos de fenda de zinco com cabeça escareada com rosca de 3/16'' e comprimento de 2. 1/2'', 196 arruelas lisas de zinco de diâmetro de 10 mm e 196 porcas de zinco sextavadas de rosca 3/16''. Na Figura 46 é demonstrado a estrutura montada em 4 perspectivas.

Figura 46: Estrutura construída para o protótipo, apresentando as vistas: (a) frontal da estrutura; (b) lateral direita; (c) lateral esquerda; e (d) traseira.

(a)

(b)

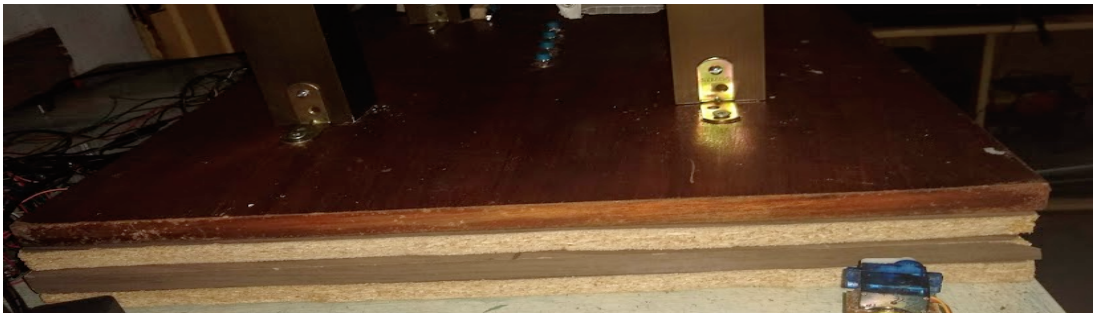
(c)

d)

Fonte: o autor.

A plataforma de estrutura foi feita com uma chapa de madeira retangular de 65cm x 47cm x 2cm e com 2 pedestais em baixo da plataforma composto por 3 chapas de madeira retangulares de 65cm x 10cm x 1,5cm. Este detalhe de construção pode ser visto na Figura 47.

Figura 47: Dianteira (a) e lateral (b) da plataforma da estrutura com seus pedestais.



(a)
Fonte: autor

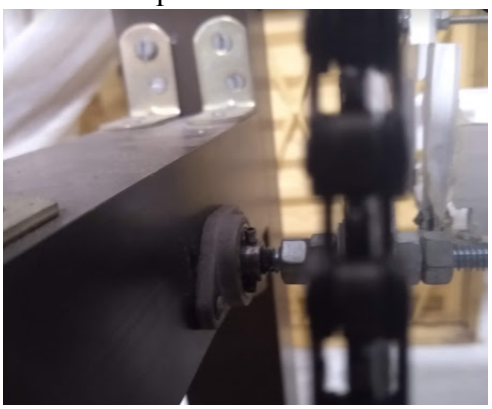
(b)
Fonte: o autor.

Para a montagem das partes móveis foram utilizados 4 mancais modelo KFL08 (Figuras 48 e 49) com rolamento para eixos de 8 mm com prisioneiro e 2 barras roscadas de rosca 5/16’’ com comprimento de 350 mm. Foram utilizadas 4 engrenagens do tipo estrela de 48 dentes e 19 cm de diâmetro de um dente ao outro, 2 correntes de bicicleta Tec C410 1/2x1/8 114 elos, 16 porcas sextavadas zincadas de rosca 5/16’’, 4 arruelas lisas de 38mm de diâmetro e 8 arruelas lisas de 24 mm. Esta montagem pode ser vista na Figura 50.

Figura 48: Foto rolamento com mancal para eixos de 8mm KFL08.

Fonte: (OLIMEX, 2021).

Figura 49: Rolamento com mancal para eixos de 8mm KFL08, instalado na estrutura.



Fonte: o autor.

Figura 50: Fotografias lateral (a) e superior (b) das engrenagens inferiores instaladas com o eixo e rolamento; Fotografias lateral (c) e superior (d) das engrenagens superiores instaladas com o eixo e rolamento.

(a) (b)

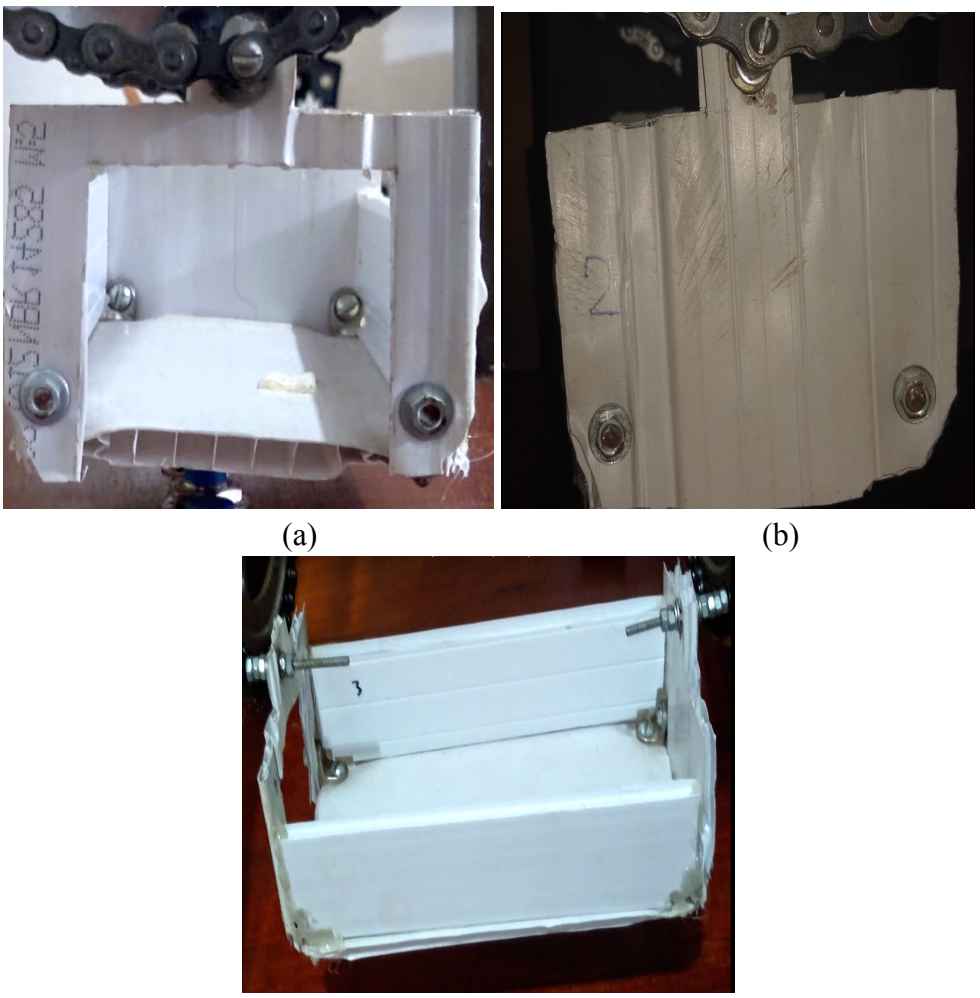
(c) (d)

Fonte: o autor.

Para esta maquete do projeto, foram utilizadas 4 plataformas montadas com chapas de PVC, mostradas na Figura 51. Para fixá-las na corrente de transmissão, foram utilizados 8 parafusos zincados de rosca 1/8" com comprimento de 2" e 24 porcas sextavadas zincadas de rosca de 1/8". As plataformas foram feitas com chapas de PVC e unidas com 24 cantoneiras

cromadas com largura de 11mm e comprimento de 14mm em cada face, com 48 parafusos fenda, cabeça escareadas zincados de rosca 3/16’’ e comprimento de 1/2’’, 98 arruelas lisas zincadas de diâmetro de 10mm. As laterais das plataformas de PVC foram unidas com cola quente.

Figura 51: Fotografia dianteira (a), traseira (b) e lateral (c) da plataforma de PVC instalada na estrutura.



(a) (b)

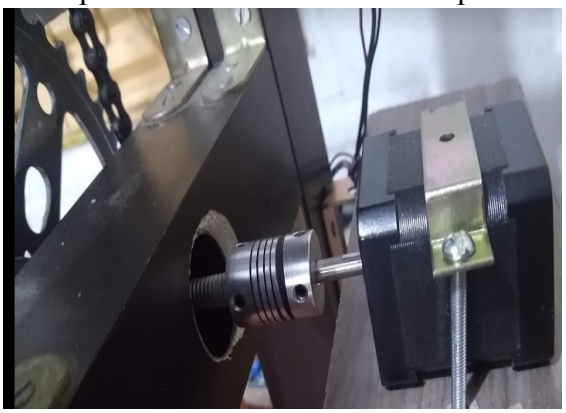
(c)

Fonte: o autor.

Para a montagem da estrutura em perfis quadrados de alumínio, foi unido os perfis quadrados parafusando cantoneiras com finalidade de conectar um perfil ao outro conforme mostrado na Figuras 46. Com a estrutura de perfis quadrados já montados, foi fixado os mancais com rolamentos através de 2 parafusos fenda de cabeça escareadas com rosca 3/16’’ e comprimento de 3/4’’ zincados nas extremidades de cada mancal nas estruturas. Nos mancais foram instalados as barras roscadas fixadas com prisioneiros em suas extremidades contidos nos próprios rolamentos conforme mostrado nas Figuras 49 e 50. Na barra roscada

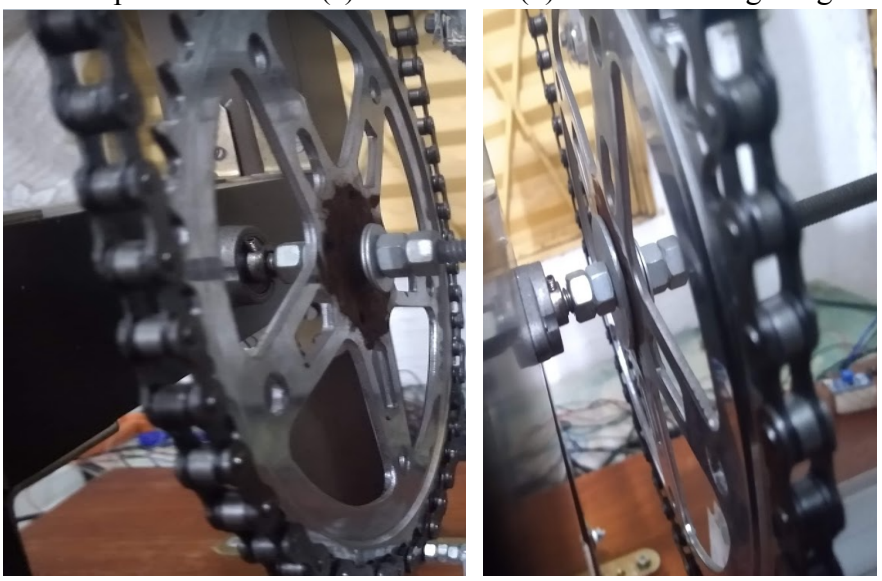
inferior estão instaladas as duas engrenagens inferiores que movimentam as plataformas e também, em um dos extremos da barra roscada, está acoplado a barra roscada com o eixo do motor de passos através de um acoplamento flexível de eixos de 8mm para 5mm. Para fixar estas engrenagens no eixo inferior, foi utilizado uma fixação com porca e contraporca com mais duas arruelas pressionando a engrenagem para garantir que a mesma fique alinhada e não se mova durante suas movimentações. Esta montagem é apresentada na Figura 52.

Figura 52: Foto motor de passos unido ao eixo inferior por um acoplamento flexível.



Fonte: o autor.

Figura 53: Contra-porcas traseiras (a) e dianteiras (b) de uma das engrenagens inferiores.



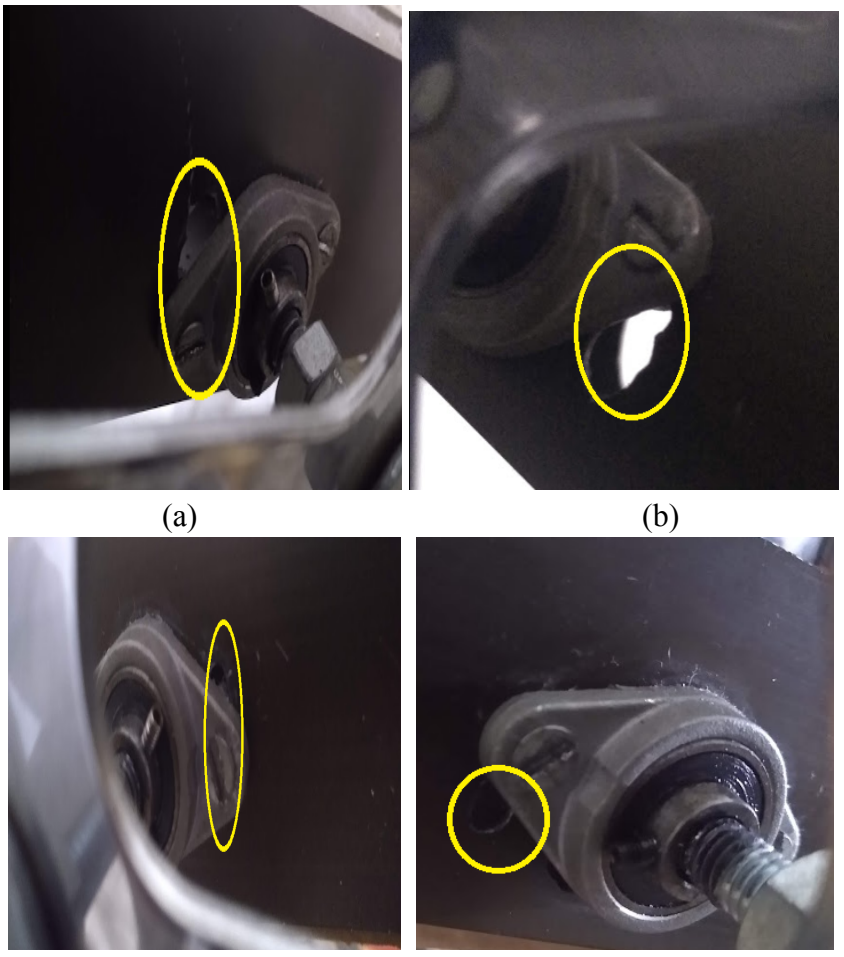
a) **b)**

Fonte: o autor.

Na parte superior foi fixada outra barra roscada e as outras duas engrenagens que movimentam as plataformas da mesma maneira que a barra roscada inferior e as engrenagens inferiores (Figura 53). Porém, na instalação dos mancais superiores que fixam partes móveis superiores, foi feito um orifício maior na fixação dos parafusos dos mancais superiores, que

atua como um ajuste das correias, pois, tendo um orifício para que possa movimentar para cima e para baixo os mancais superiores é possível fazer um ajuste das correias nas partes móveis do sistema, uma vez que o conjunto engrenagem e correia se encontram fixadas nos mancais. A Figura 54 apresenta imagens da furação feita para o ajuste de tensionamento da correia.

Figura 54: Furações feitas para permitir o ajuste de tensionamento da correia, apontada pela marcação amarela em forma de elipse: a) furação esquerda do rolamento traseiro; b) furação direita do rolamento traseiro; c) furação esquerda do rolamento dianteiro; d) furação direita do rolamento dianteiro.



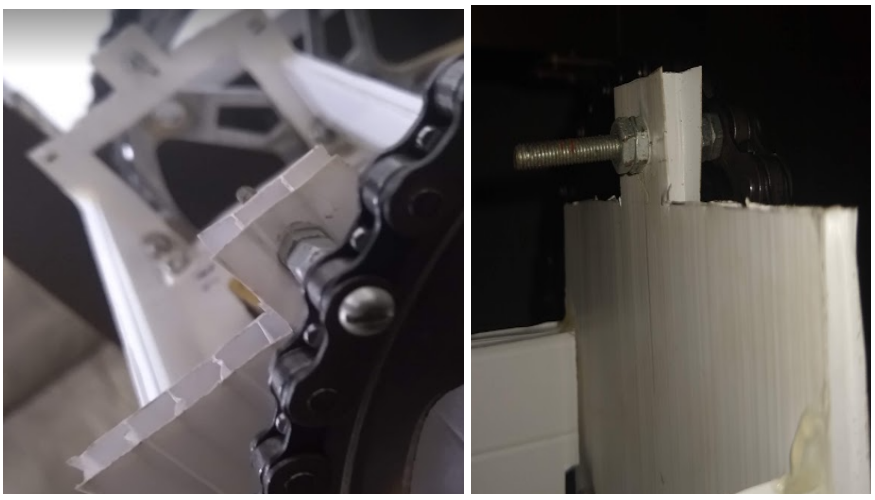
(a) (b)

(c) (d)

Fonte: o autor.

Para fixar as plataformas nas correias foi necessário remover o pino da correia com a finalidade de passar os parafusos nos orifícios dos pinos retirados, para fixar no corpo destes parafusos as abas das plataformas que estão vazadas para a passagem dos parafusos fixando-as com porca, conforme é demonstrado na Figura 55.

Figura 55: a) Foto demonstrando o parafuso para a fixação da plataforma do carro na correia após sacar o pino da correia; b) Foto demonstrando porcas para ajustar a plataforma fixada na correia.



(a) (b)

Fonte: o autor.

3.3 MONTAGEM DO SISTEMA DE FREIO

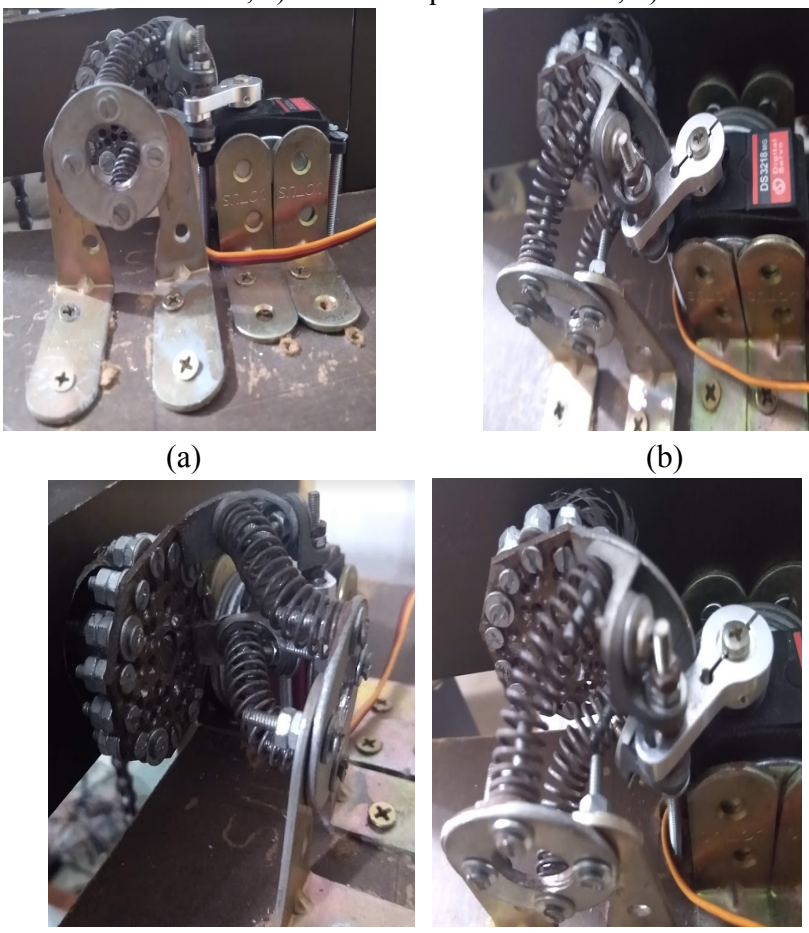
Para o carrossel giratório parar, é necessário um sistema de frenagem. O sistema de frenagem adotado para este protótipo consiste na utilização de um servo motor com capacidade de 20 kg.f/cm² para atuar com uma frenagem eletromecânica microcontrolada pelo Arduino. Esta frenagem consiste em uma instalação de um disco de freio na ponta do eixo superior com parafusos fixados no disco, com a finalidade de formar arestas para o atrito no momento da atuação da frenagem. Este sistema eletromecânico e suas partes são mostradas na Figura 56.

O servo motor do freio atuará sobre o disco de freio com uma alavanca instalada no eixo do servo motor, que em sua extremidade possui uma pastilha metálica em forma de “U”, a qual está posicionada em torno de 135° para avançar em direção ao disco de freio. O controle do avanço da pastilha consiste na pressão exercida por 2 molas helicoidais sobre a pastilha, que terá o avanço da alavanca do servo motor controlada através de controle de posição em graus pelo Arduino, sendo a posição freada de 30° e de freio liberado de 40°.

A alavanca freia mecanicamente com a força exercida pelas molas helicoidais, caso haja ausência de alimentação elétrica no motor.

Com motor alimentado com uma fonte adequada e o sinal de controle do Arduino, o acionamento do freio tem a parte mecânica microcontrolada.

Figura 56: a) Servo motor instalado com as partes móveis do acionamento do freio; b) Superior do sistema de freio; c) Lateral esquerda do freio; d) Lateral direita do freio.



(a) (b)

(c) (d)

Fonte: o autor.

3.4 INSTALAÇÃO MOTOR DE PASSO PARA ACIONAMENTO DO CARROSSEL DE VAGAS

O dispositivo que é usado para movimentar uma determinada vaga até o solo é um motor de passo bipolar Stepperonline 17HS19-2004S1, conforme sua imagem é mostrado na Figura 41. O motor de passo foi instalado na parte inferior do protótipo onde o seu eixo é unido há uma barra roscada através de um acoplamento flexível de eixos de 8mm para 5mm, com a finalidade de transmitir o movimento para o carrossel de vagas, conforme mostrado na Figura 52. O motor de passo é acionado quando a *tag* lida pelo leitor RFID conter o código cadastrado no algoritmo do sistema, gerando um sinal de PWM para *driver* A4988 (mostrado na Figura 39) e dando início a movimentação do carrosel. Sua atuação é interrompida quando

o sensor indutivo envia um sinal em nível alto para o Arduino, conforme mostra a posição dos sensores no sistema da Figura 57.

3.5 INSTALAÇÃO DOS SENSORES INDUTIVOS

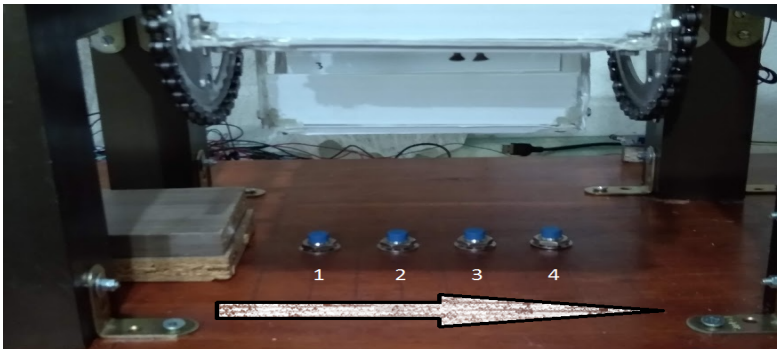
Após as partes móveis estarem montadas corretamente, para controlar as paradas das vagas serão utilizados 4 sensores indutivos, que terão gravados no código fonte do Arduino sua respectiva vaga por sensor. O controle de parada será feito através de um pulso emitidos cada vez que uma peça metálica instalada abaixo das plataformas passar em frente a face do sensor indutivo, que serão enviados ao Arduino e o mesmo irá reconhecer qual o sinal de uma vaga específica que deverá para de acionar o motor de passo e servo do freio do motor.

Figura 57: Sensores indutivos posicionados para cada vaga nas plataformas.

Fonte: o autor.

O algoritmo de controle das paradas das plataformas consiste no sinal de 4 sensores indutivos, que atuam em forma de uma chave sem contato mecânico. Os sensores estão enfileirados de forma que fiquem arranjados em uma sequência de 1, 2, 3 e 4 para o sentido da frente do protótipo para trás do mesmo, para facilitar a inspeção visual dos sensores equivalente a sua respectiva vaga (Figura 58).

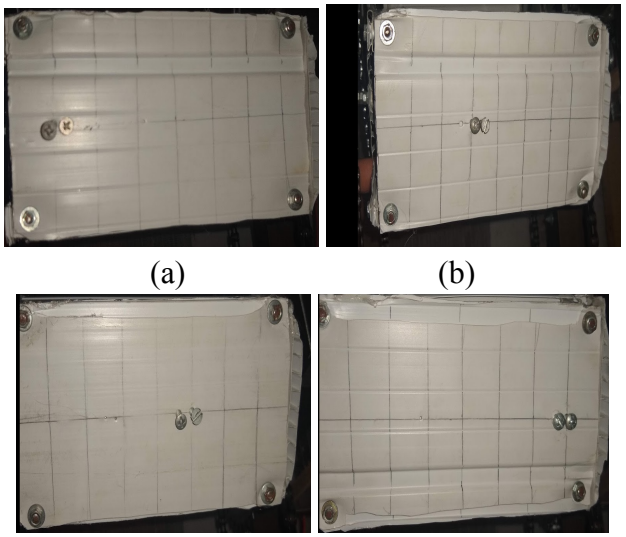
Figura 58: Sentido crescente da posição dos sensores frente do protótipo, de 1 ao 4.



Fonte: o autor.

Sua atuação como chave acontecerá quando o parafuso instalado na parte inferior da plataforma do carro passar sobre o sensor indutivo, que geram um sinal em nível alto para o Arduíno. Desta forma, o microcontrolador irá interpretar este sinal como uma condição para ocorrer a parada do sistema, ou seja, a vaga está posicionada no lugar. As posições dos parafusos em cada uma das vagas estão representadas na Figura 59.

Figura 59: Posição dos parafusos que passam a frente do sensor sendo: a) parafuso da vaga 1; b) parafuso da vaga 2; c) parafuso da vaga 3; e d) parafuso da vaga 4.



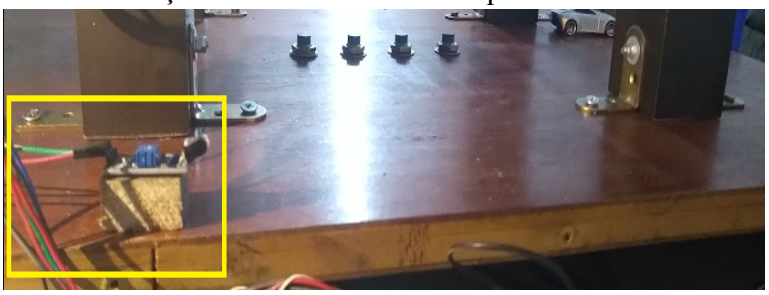
(a) (b)

(c) (d)

Fonte: o autor.

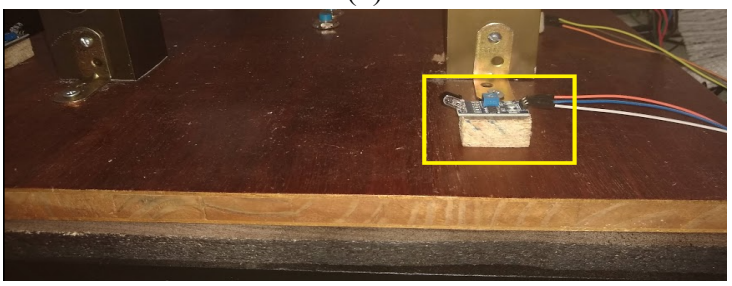
Para a parte de segurança, onde as pessoas podem passar em lugares de risco enquanto a plataforma se movimenta, foi implementado sensor de obstáculo de tecnologia com LEDs infravermelho por reflexão nas 3 laterais da estrutura (mostrado na Figura 60) que complementariam a restrição de pessoas ao acesso de partes móveis que poderiam causar acidentes estando após as grades de restrição do acesso aos arredores do estacionamento vertical na lateral esquerda, direita e traseira. Quando um obstáculo passa em frente de algum destes sensores, o sensor enviará um sinal ao microcontrolador em nível baixo que fará com que o motor pare e o freio seja acionado. Para que a plataforma se movimente novamente, será necessário passar a *tag* no leitor RFID novamente.

Figura 60: Posição dos sensores de obstáculo em lugares da estrutura onde não oferecem segurança: sensor na lateral esquerda (a), direita (b) e traseira (c) da estrutura em destaque com marcação em forma de bordas quadradas amarelas.



(a)

(b)



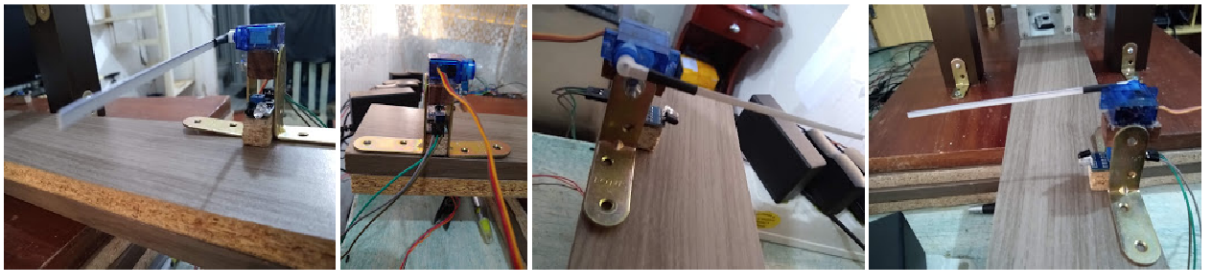
(c)

Fonte: o autor.

3.6 MONTAGEM DA CANCELA DE CONTROLE DE ACESSO

Para o controle de acesso do carro ao estacionamento, há uma cancela para restringir a passagem do carro do dono da vaga. Como mostra a Figura 61, a cancela deste protótipo é feita com 2 cantoneiras latonadas de largura de 20 mm por face e altura de 48 mm por face, que contém um servo motor de 2,5 kgf.cm instalado em suas extremidades em uma chapa retangular de madeira de medidas 15mm x 31mm x 12mm e uma haste feita de PVC fixada no eixo do servo motor.

Figura 61: Cancela do protótipo: a) lateral esquerda da cancela; b) lateral direita da cancela; c) traseira da cancela; d) dianteira da cancela.



(a) (b) (c) (d)

Fonte: o autor.

O controle de fechamento da cancela é feito através de um sinal do Arduino que controla a posição do eixo do servo motor programada por ângulos de 0° a 180°. No projeto proposto, o servo motor (mostrado na Figura 62) está programado para 90° aberto e 0° fechado.

Figura 62: Servo motor Tower Pro Micro Servo 9g SG90

Fonte: (EE, 2021).

O controle da cancela na posição aberta (90°), é programado para quando passar a *tag* no leitor RFID e se o código da *tag*, que estiver cadastrado, for válido. A cancela será fechada quando o carro passar à frente de um sensor de obstáculo, que se encontra instalado abaixo do servo motor da cancela. Este sensor controla a atuação do servo motor em 0° enviando um sinal em nível baixo para o Arduino enviar o sinal de 0° para baixar a cancela. A montagem do servomotor da cancela e do sensor de obstáculo é mostrada na Figura 63.

Figura 63: a) Servo motor instalado na cancela; b) sensor de obstáculo instalado na cancela.

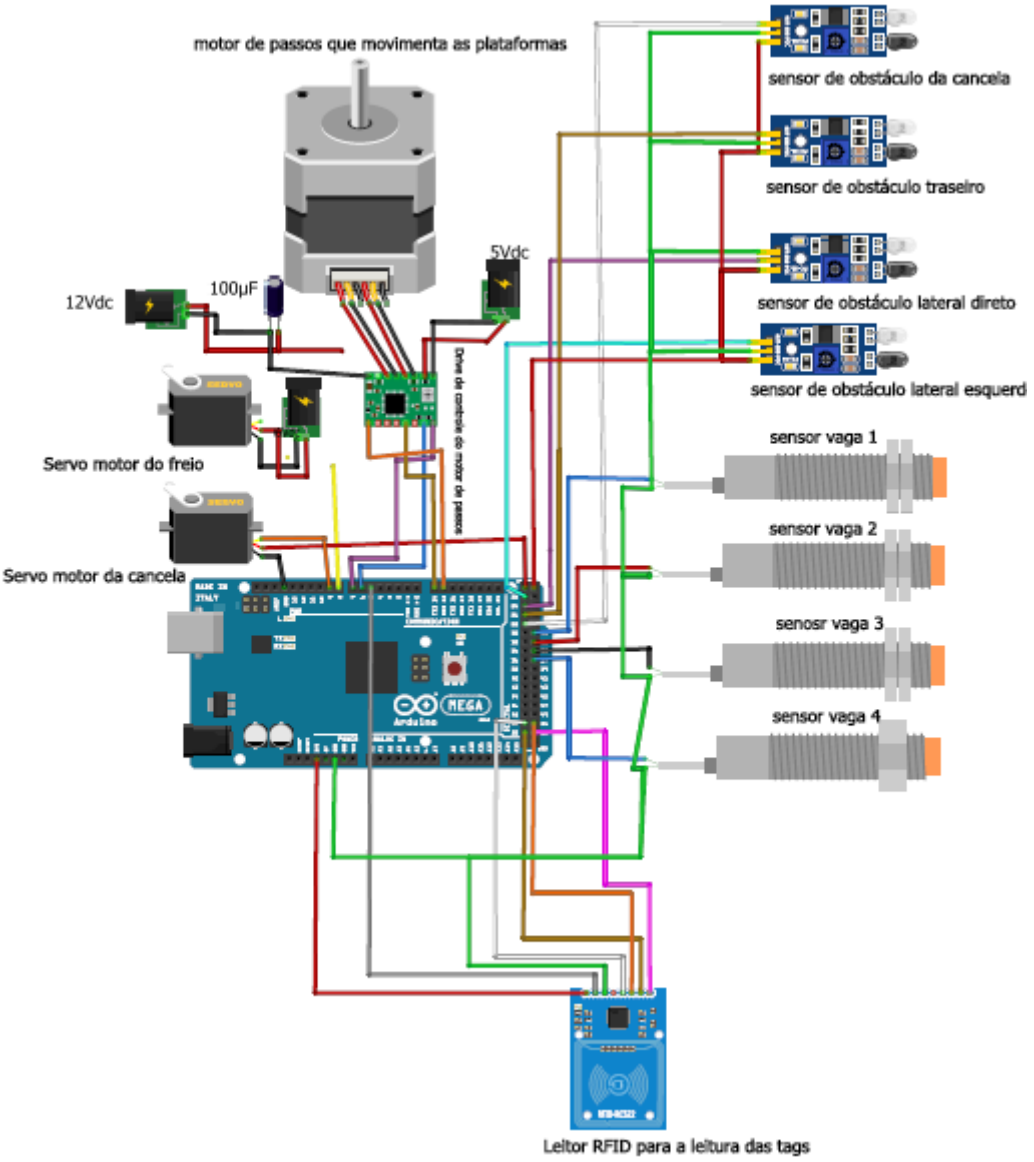
(a) (b)

Fonte: o autor.

3.7 IMPLEMENTAÇÃO PARA O CONTROLE E O SENSORIAMENTO

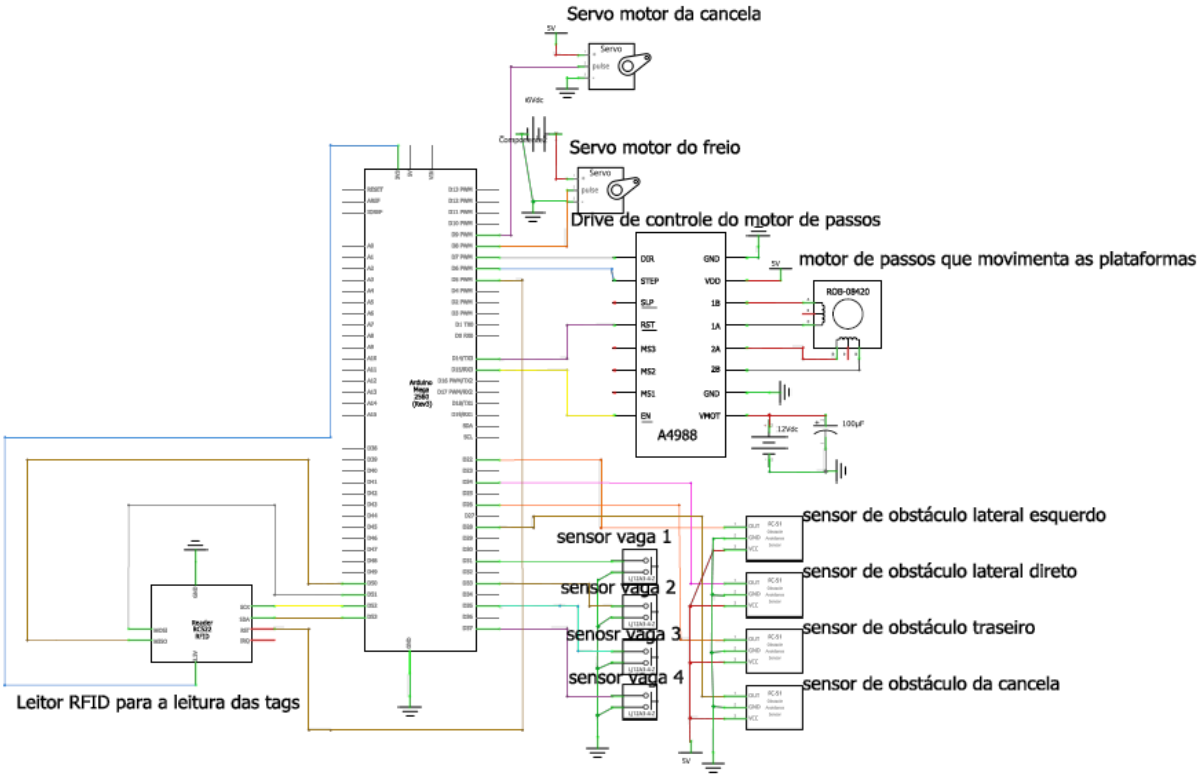
Os componentes eletrônicos necessários para a implementação completa deste projeto são apresentados nos diagramas das Figuras 64 e 65. Na Figura 65 é apresentada ilustração do diagrama esquemático dos circuitos eletrônicos dos sensores, motores e a *shield* do Arduino, gerado pelo *software* Fritizing. Na Figura 64 é mostrada a ilustração do diagrama esquemático com as ligações elétricas dos circuitos eletrônicos dos sensores, motores e a *shield* do Arduino.

Figura 64: Ilustração esquemática das ligações dos sensores e motores com o Arduíno, gerado pelo *software* Fritzing.



Fonte: o autor.

Figura 65: Diagrama esquemático dos circuitos eletrônicos dos sensores, motores e a *shield* do Arduino, gerado pelo *software* Fritizing.



Fonte: o autor**.**

Para a interface com o usuário será representada pelo monitor serial da IDE do Arduíno, sendo que ao passar a *tag* no leitor RFID o mesmo irá demonstrar na tela do monitor informações a respeito do proprietário da vaga e posição da vaga.

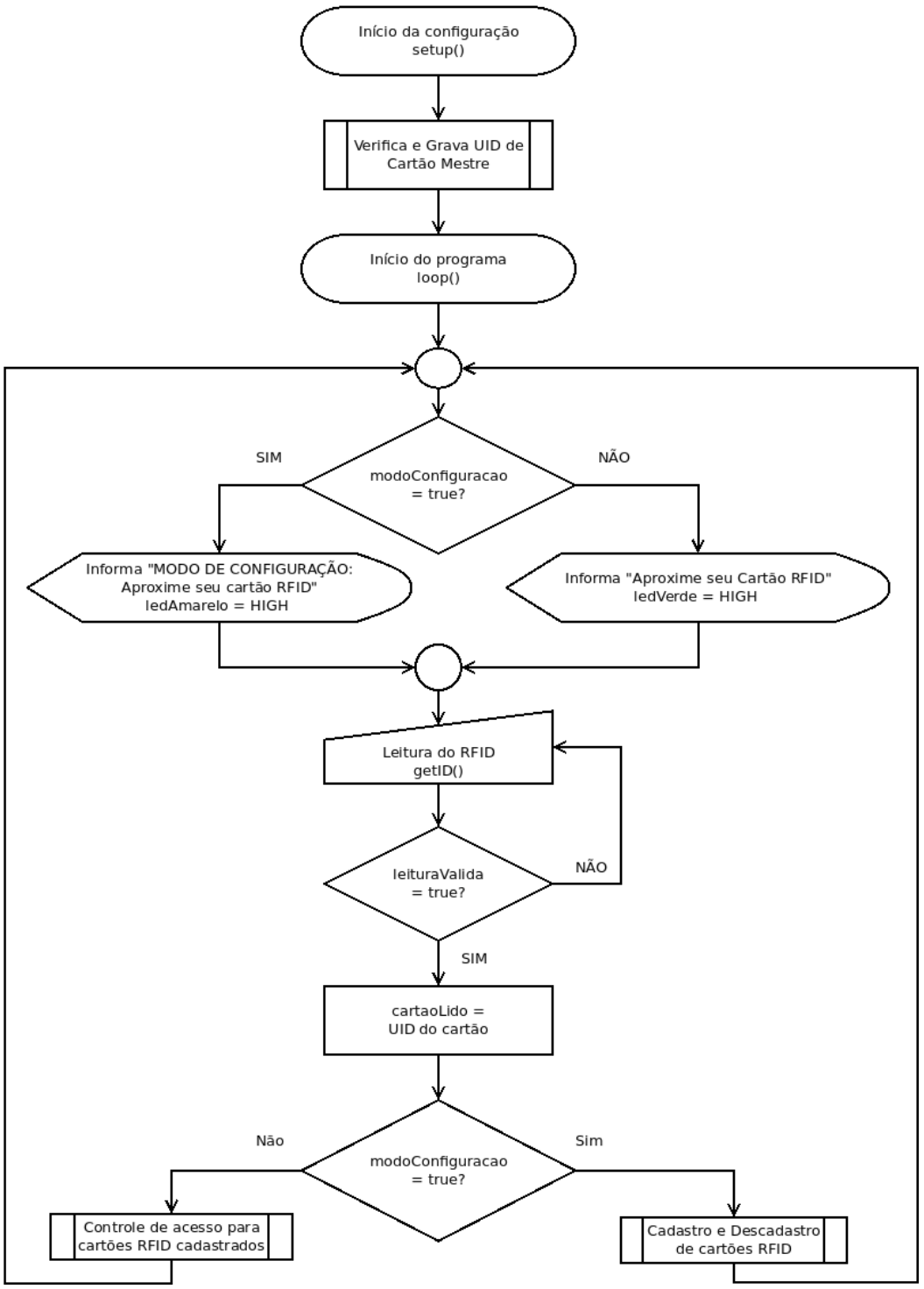
3.8 PROGRAMAÇÃO DO SISTEMA

Nesta seção serão apresentadas as técnicas utilizadas no algoritmo da programação utilizadas no sistema do protótipo.

3.8.1 ALGORITMO DE CONTROLE DE ACESSO

O algoritmo geral de controle de acesso, contendo as etapas de inicialização do sistema, é mostrado no fluxograma da Figura 66.

Figura 66: Fluxograma de controle de acesso para usuários cadastrados no sistema



Fonte: o autor.

No início deste algoritmo é feita a verificação de registro de um UID para gravar o cartão mestre. Caso não esteja gravado o cartão mestre, a ação que deverá ser feita é de passar uma *tag* no leitor RFID para cadastrar o cartão mestre. Após isto, o programa irá dar início a função *loop()* do código do Arduino. Na função *loop()*, a primeira ação feita é verificar se a

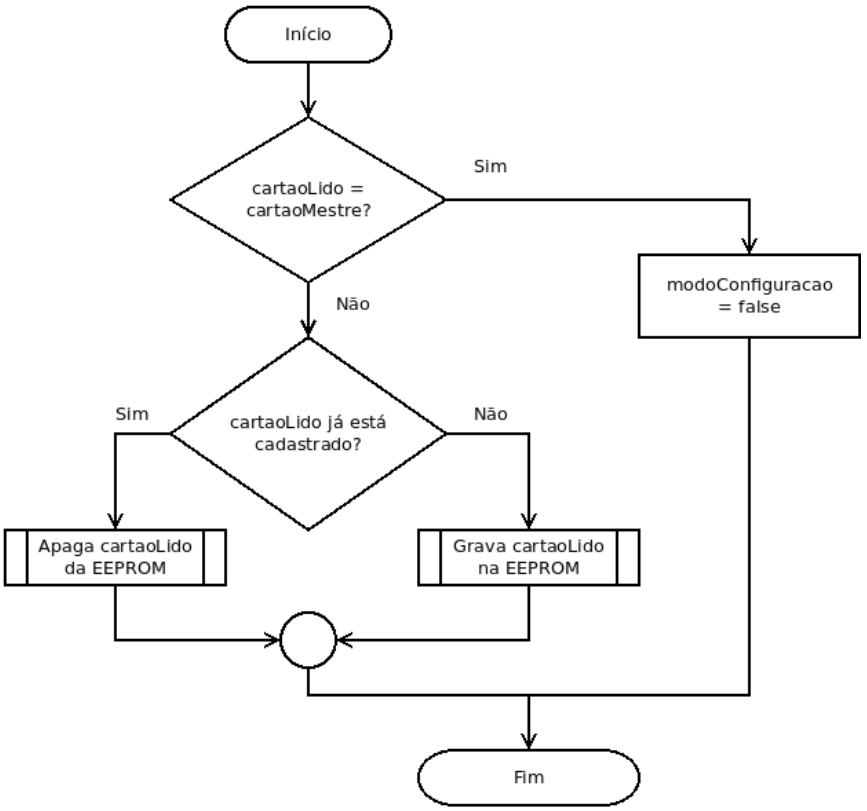
variável modoConfiguracao, usada para entrar no modo configuração, é verdadeira ou não. Sendo que, para torná-la verdadeira, é necessário haver a leitura do cartão mestre uma vez no leitor RFID e para torná-la falsa deverá ser feita novamente a leitura do cartão mestre.

Sendo verdadeira esta condição, o modo de configuração será ativado e o LED amarelo irá acender para demonstrar que o modo configuração foi iniciado. Neste momento, o sistema irá pedir para efetuar a leitura de uma *tag* para cadastrar ou descadastrar um usuário. Após cadastrar um morador no modo configuração, o mesmo deve ser desativado. Sendo falsa esta condição o modo de configuração será desativado. Sendo encerrado o modo configuração o sistema aguarda a leitura de uma *tag* para conferir se a *tag* está cadastrada no sistema. Se a *tag* lida não estiver cadastrada, o sistema irá continuar aguardando a leitura de uma *tag* cadastrada. Se a *tag* estiver cadastrada haverá uma nova avaliação para condição do modo configuração. Se o modo configuração for falso, o sistema irá dar início ao processo de buscar a vaga, porém se for verdadeiro, o sistema irá começar o sistema para cadastro e descadastro das *tags*.

3.8.2 ALGORITMO DE CADASTRAMENTO DE MORADORES

O algoritmo de cadastramento e descadastramento de *tags* RFID é mostrado na Figura 67. Sendo o modo de configuração verdadeiro através da validação do cartão mestre, o sistema entrará em modo para cadastro e descadastro das *tags*. Ao efetuar a leitura de uma *tag* que não seja a do cartão mestre, será feito uma avaliação se esta *tag* está cadastrada no sistema. Caso a *tag* esteja cadastrada no sistema seu descadastro será realizado; caso contrário, a *tag* terá seu cadastro efetuado na memória EEPROM do Arduíno.

Figura 67: Algoritmo do cadastramento e descadastramento de tags RFID



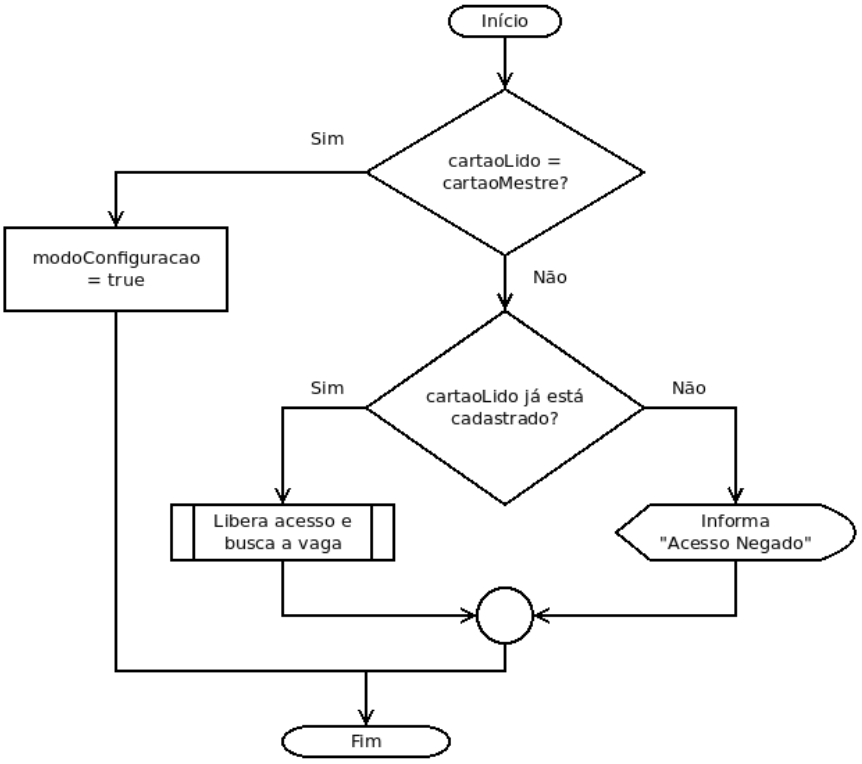
Fonte: o autor.

3.8.3 ALGORITMO DE CONTROLE DE ACESSO ÀS VAGAS

A Figura 68 apresenta o algoritmo de controle de acesso às vagas de estacionamento, para as *tags* RFID já cadastradas. Para permitir o acesso a uma vaga cadastrada, o algoritmo irá avaliar primeiramente se a *tag* que foi lida contém o UID do cartão mestre ou não. Não sendo o cartão mestre, o sistema irá avaliar se a *tag* que está tentando o acesso está cadastrada no sistema. Caso a *tag* esteja cadastrada no sistema, começará o processo para liberação e busca de vaga. Se a *tag* não estiver cadastrada, o sistema acusará que o acesso é negado.

Sendo verdadeira a leitura do cartão mestre, o sistema entra no modo de configuração, fazendo modoConfiguracao = true.

Figura 68 : Algoritmo do controle de acesso às vagas



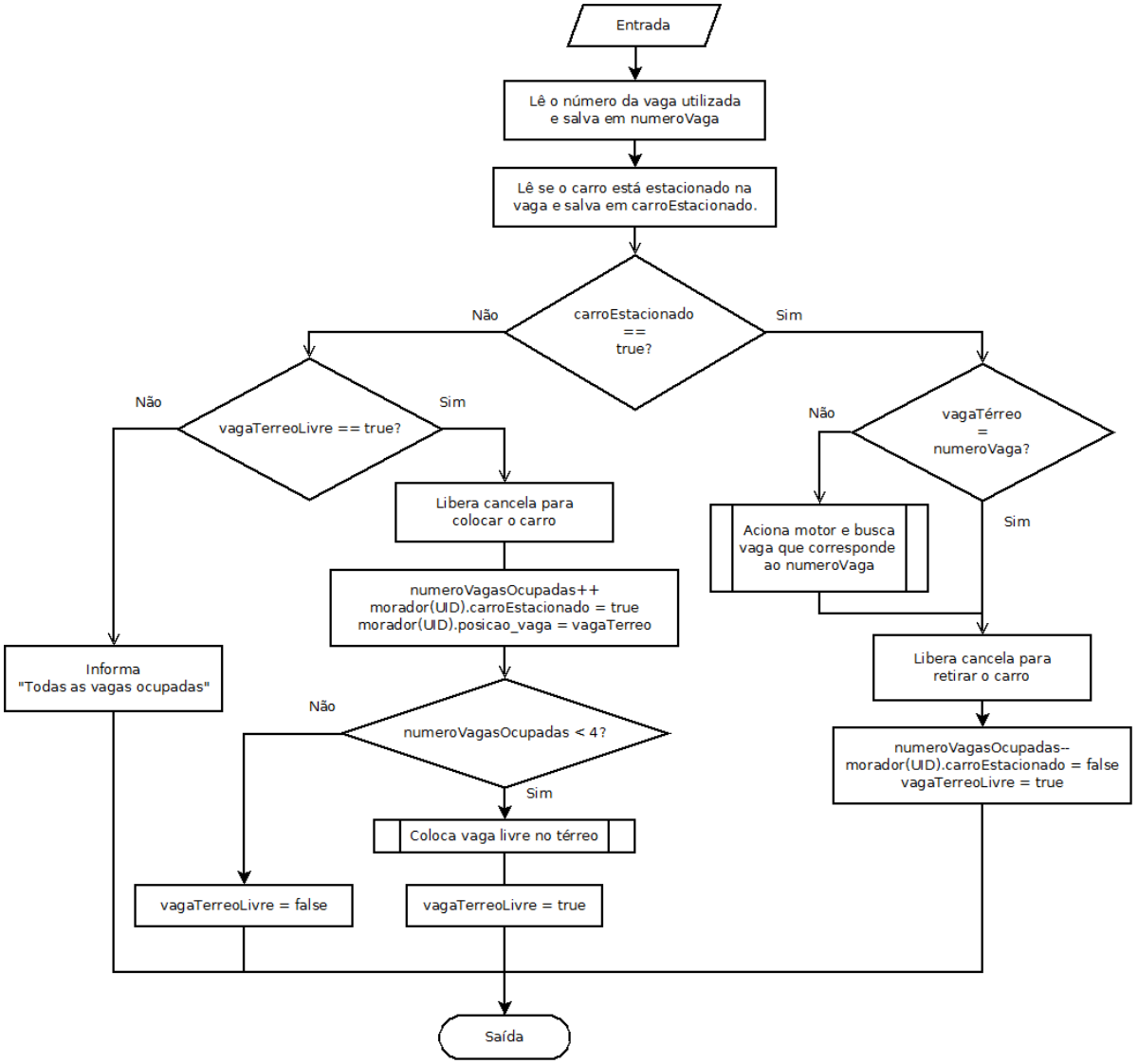
Fonte: o autor.

3.8.4 ALGORITMO PARA FAZER A BUSCA DE UMA VAGA

O algoritmo apresentado na Figura 69 é usado para buscar uma vaga para uma *tag* RFID, com ou sem o carro estacionado. Neste algoritmo é demonstrado como ocorre a busca da vaga quando o sistema lê uma *tag* que está cadastrada. O usuário que contém uma *tag* cadastrada no sistema terá que salvar sua *tag* associada ao número de uma vaga. Se caso o carro estiver estacionado na vaga, haverá uma variável no sistema para salvar o carro que estiver estacionado. Se o carro estiver estacionado no térreo, haverá uma condição que avalia se carro estacionado é igual o número da vaga associada a *tag*. Se a vaga no térreo não for a vaga associada à *tag* lida, então o sistema acionará o motor que buscará a vaga solicitada até o plano térreo. Se a vaga que estiver no térreo for a vaga solicitada, então o sistema abrirá cancela para retirada do carro. Após isso, haverá a liberação de uma vaga e o sistema reconhecerá a vaga do solo como vaga livre. Caso o carro não esteja estacionado, haverá uma condição que avaliará se a vaga do térreo está livre e se a vaga do térreo não estiver

disponível o sistema acusará que todas as vagas estão ocupadas. Caso a vaga do térreo estiver disponível, haverá um comando para liberar a cancela para colocar o carro na vaga do térreo. Uma vaga será registrada como ocupada para o sistema no térreo. Como neste protótipo o número máximo de vagas é quatro, o sistema avaliará se o número de vagas ocupadas é menor que quatro. Se o número de vagas não for menor que quatro vagas, o sistema reconhece que a vaga no térreo não está disponível, se a vaga for menor 4 vagas o sistema coloca a vaga livre no solo.

Figura 69: Algoritmo para buscar uma nova vaga ou buscar um carro estacionado.



Fonte: o autor.

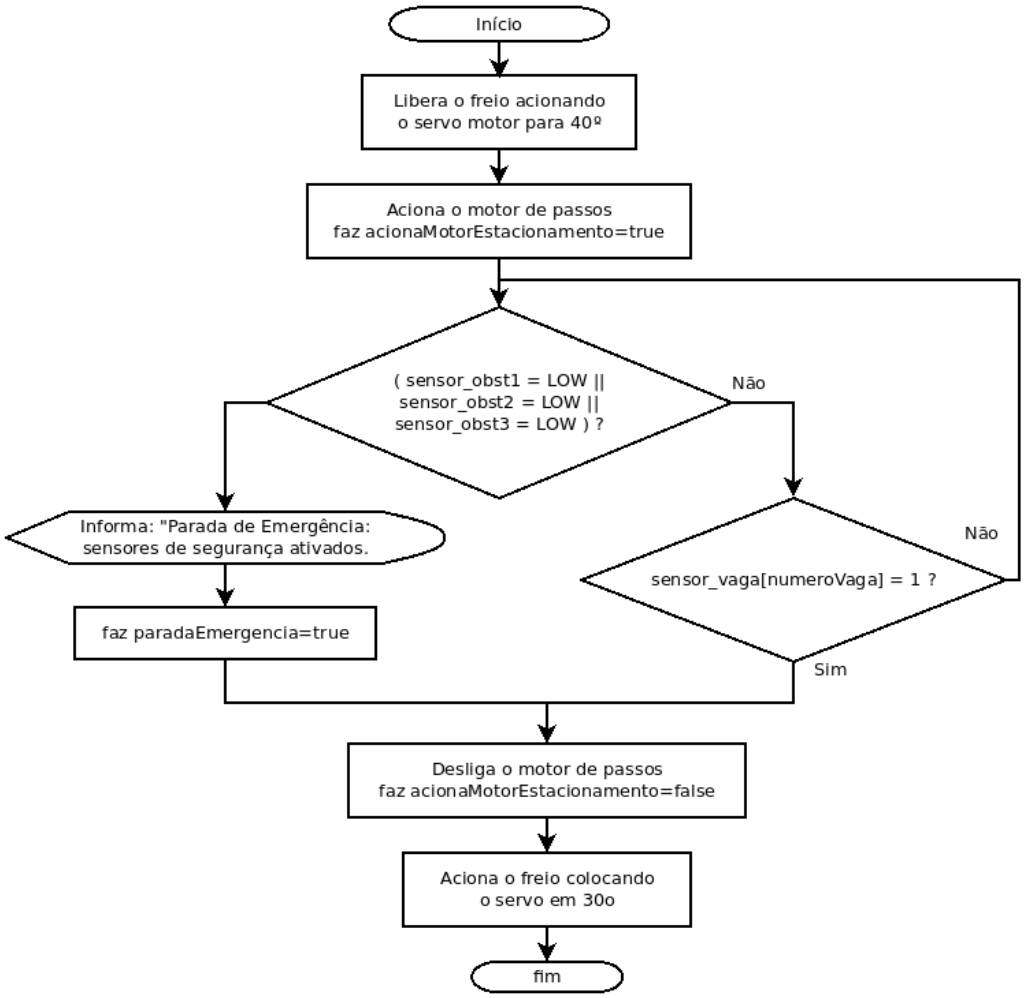
3.8.5 ALGORITMO DE CONTROLE DE MOVIMENTAÇÃO E PARADA

O algoritmo de controle de movimentação do carrossel de vagas e de parada é usado para controlar o funcionamento do sistema de freio eletromecânico, controlar o movimento do carrossel e monitorar os sensores de posição das vagas e sensores de parada de emergência.

Esta rotina de código monitora o estado dos sensores indutivos para fazer o posicionamento correto da vaga de estacionamento que é solicitada pelo usuário, para retirar um carro que está estacionado. Além disso, durante o movimento do carrossel, sensores ópticos de barreira laterais e traseiras são monitorados para fazer uma parada de emergência, caso alguém invada o perímetro do plano térreo do carrossel durante a sua movimentação.

A Figura 70 apresenta o algoritmo de controle de movimentação e parada do sistema de estacionamento vertical proposto, implementado através da rotina “operacoesVaga( )”. Este algoritmo foi elaborado seguindo as seguintes etapas: primeiramente, libera-se o freio eletromecânico para permitir o movimento da estrutura de carrossel; em seguida, aciona-se o motor de passos para movimentar a estrutura, no sentido horário; enquanto o motor estiver acionado, o microcontrolador permanece em *pooling* monitorando o estado do sensor indutivo correspondente à vaga de estacionamento solicitada; por último, quando a vaga chega ao térreo, desliga-se o motor e aciona-se o freio eletromecânico.

Figura 70: Algoritmo de controle de movimentação e parada do sistema de estacionamento vertical.



Fonte: o autor.

O algoritmo apresentado na Figura 70 através da função chamada de operacoesVaga( ) é listado exposto abaixo.

```
void operacoesVaga(uint8_t numeroVaga)
{
 bool acionaMotorEstacionamento = true;

 sfreio.write(40); // libera o freio movimentoando o servo para 40o
 digitalWrite(enable, LOW); // habilita driver do servo motor
 digitalWrite(reset, HIGH);
 digitalWrite(dirPin, LOW);

 while (acionaMotorEstacionamento)
 {
        // Verifica sensores de barreira laterais e traseiro
        if ( (digitalRead(sensor_obst1) == LOW) ||
          (digitalRead(sensor_obst2) == LOW) ||
          (digitalRead(sensor_obst3) == LOW) ){
          acionaMotorEstacionamento = false;
          paradaEmergencia = true; // sinaliza na variável global
          digitalWrite(stepPin, LOW);
```

```
            digitalWrite(reset, LOW);
            digitalWrite(enable, HIGH); // desabilita driver do servo motor
            sfreio.write(30); // aciona o freio eletromecânico
            Serial.println("Parada de Emergência: Sensores de segurança ativados.");
          } else if (digitalRead(sensor_vaga[numeroVaga]) == HIGH) {
            // Verifica se o estacionamento já está com a vaga  no solo
            acionaMotorEstacionamento = false;
            digitalWrite(stepPin, LOW);
            digitalWrite(reset, LOW);
            digitalWrite(enable, HIGH); // desabilita driver do servo motor
            sfreio.write(30); // aciona o freio eletromecânico
            Serial.println("Vaga pronta!");
          } else {
          // aciona motor de passos enquanto o sensor de posição da vaga não for acionado
            digitalWrite(stepPin, HIGH);
            delayMicroseconds(MOTOR_PASSO_TON);
            digitalWrite(stepPin, LOW);
            delayMicroseconds(MOTOR_PASSO_TOFF);
            delay(delay_ajuste);
    }
  }
}
```

Este trecho do código fonte contém o comando de parada com acionamento do freio e a desabilitação dos pinos de controle do drive do motor de passos que movimentam o carrossel. A seguir demonstra-se os trechos do código em que o sensor indutivo envia um sinal de nível alto para o acionamento do freio e a desabilitação dos terminais do drive do motor de passos

Os comandos digitalWrite(stepPin, LOW), digitalWrite(reset, LOW) e digitalWrite(enable, HIGH) são usados para desabilitar o *driver* do motor de passos. O comando sfreio.write(30) permite acionar o freio eletromecânico, enquanto que o comando sfreio.write(40) permite liberar o freio eletromecânico.

Os seguintes comandos para início do giro do carrossel.

```
sfreio.write(40); // libera o freio movimentando o servo para 40o
digitalWrite(enable, LOW); // habilita driver do servo motor
digitalWrite(reset, HIGH);
digitalWrite(dirPin, LOW);
```

O comando sfreio.write(40) permite liberar o freio eletromecânico, enquanto que os comandos digitalWrite(enable, LOW), digitalWrite(reset, HIGH) e digitalWrite(dirPin, LOW), habilitam do *driver* do motor de passos.

O sensoriamento de segurança está arranjado nas laterais esquerda, direita e traseiro da estrutura, conforme está representado nas Figuras 60 a, b e c. Um sinal em nível baixo LOW está programado no algoritmo do código para que seja interpretado que um obstáculo passou à frente de um local de risco e assim acionando os freios do sistema, bem como desabilitando o drive do motor de passos.

```
if ( (digitalRead(sensor_obst1) == LOW) ||
            (digitalRead(sensor_obst2) == LOW) ||
            (digitalRead(sensor_obst3) == LOW) ){
            acionaMotorEstacionamento = false;
            digitalWrite(stepPin, LOW);
            digitalWrite(reset, LOW);
            digitalWrite(enable, HIGH); // desabilita driver do servo motor
            sfreio.write(30); // aciona o freio eletromecânico
            Serial.println("Alerta: Sensor de segurança ativado!");
}
```

São os comandos digitalRead(sensor_obst1) == LOW, digitalRead(sensor_obst2) == LOW e digitalRead(sensor_obst3) == LOW que analisam se o sinal do sensor está em nível baixo para a condição de parada do sistema.

A seguir expõem-se a análise dos resultados deste experimento de garagem vertical do tipo carrossel.

4. ANÁLISE DE RESULTADOS

Este capítulo apresenta os resultados obtidos da implementação deste projeto prático de Trabalho de Conclusão de Curso. São apresentados imagens de captura de tela do terminal serial da IDE do Arduino, com o registro das informações apresentadas pelo sistema para o usuário ou administrador do sistema.

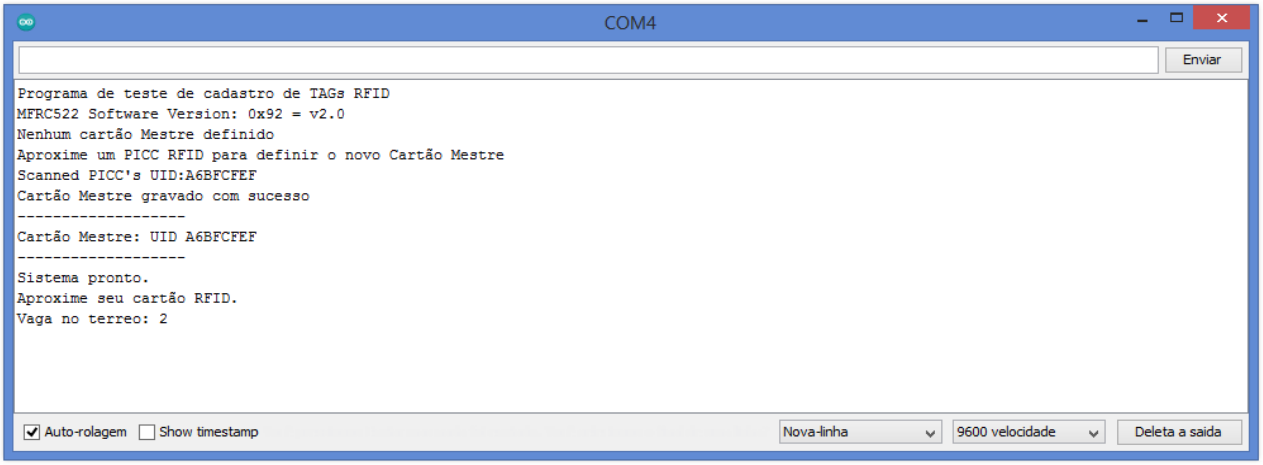
4.1 CADASTRAMENTO, DESCADASTRAMENTO E LEITURA DE TAGS RFID

Para demonstrar o funcionamento do sistema, é apresentada nesta seção um conjunto de imagens com as mensagens geradas pelo sistema através do monitor serial do Arduino durante a etapa de leitura de *tags* RFID, para fazer o controle de acesso, o cadastramento e o descadastramento. O cadastro do usuário será feito através de uma *tag* administradora de cadastros que, ao passá-la no leitor RFID, irá habilitar o modo de administração de cadastros.
A operação para o cadastro será feito da seguinte forma:

- Ao passar o cartão administrativo o modo para cadastrar será ativado;
- Haverá uma mensagem solicitando que passe uma *tag*;
- Se a *tag* estiver cadastrada seu código será descadastrado, porém, se a *tag* não estiver cadastrada a mesma será cadastrada;
- Após passar a *tag* o cartão administrativo deve ser passado novamente para encerrar o cadastro.

A Figura 71 apresenta o registro de mensagens do terminal serial do Arduino onde é feito o pedido para cadastrar o cartão mestre, após, a gravação da *tag* mestre, o sistema aguarda leitura de uma *tag* para verificar se a mesma está cadastrada no sistema e informando que a vaga 2 está no térreo.

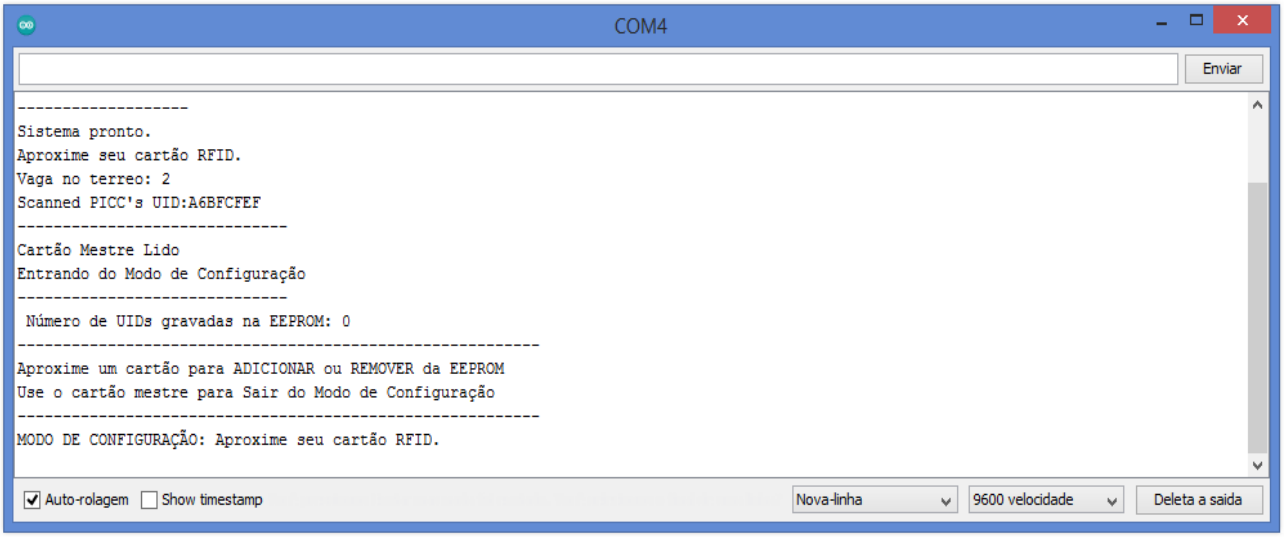
Figura 71: Mensagem de cadastro do cartão mestre.



Fonte: o autor

A Figura 72 apresenta a mensagem gerada pelo terminal serial do Arduino quando ocorre a inserção do cartão mestre no leitor RFID, para se entrar no modo de configuração. A partir disso, o sistema permite passar uma *tag* no leitor para cadastrar, caso haja ausência do seu cadastro no sistema, ou descadastrar, se houver o cadastro da *tag* registrado no sistema. Desta forma, ao passar duas vezes a mesma *tag* durante o modo de configuração, o descadastro será realizado.

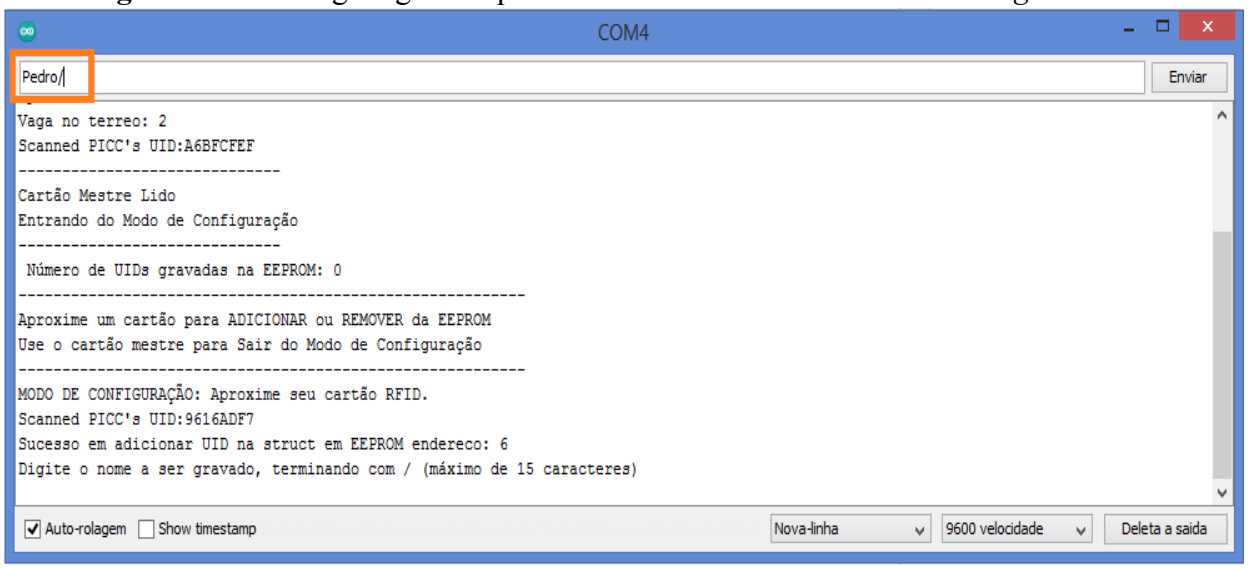
Figura 72: Mensagem de entrada no modo de configuração.



Fonte: o autor

Na Figura 73 é mostrado o registro de mensagens quando uma nova *tag* é cadastrada no sistema. O sistema efetua a leitura de uma *tag* que não estava cadastrada, após, o sistema está pedindo para digitar um nome de usuário para a *tag* na barra superior do monitor serial conforme destacado em no retângulo vermelho (a validação do nome tem que ser realizada pela tecla enter do computador).

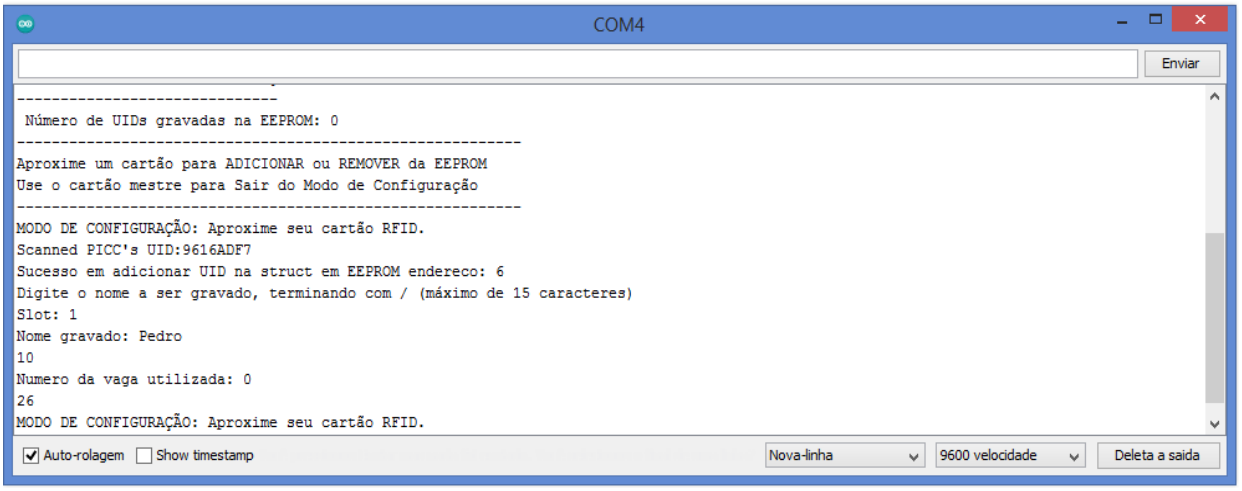
Figura 73: Mensagem gerada quando é feito o cadastro de uma nova *tag* RFID.



Fonte: o autor.

A Figura 74 apresenta a mensagem gerada pela interface do sistema confirmando a validação da *tag* que está cadastrada no sistema, apresentando o nome do usuário registrado e número da vaga utilizada (caso esteja com o veículo estacionado).

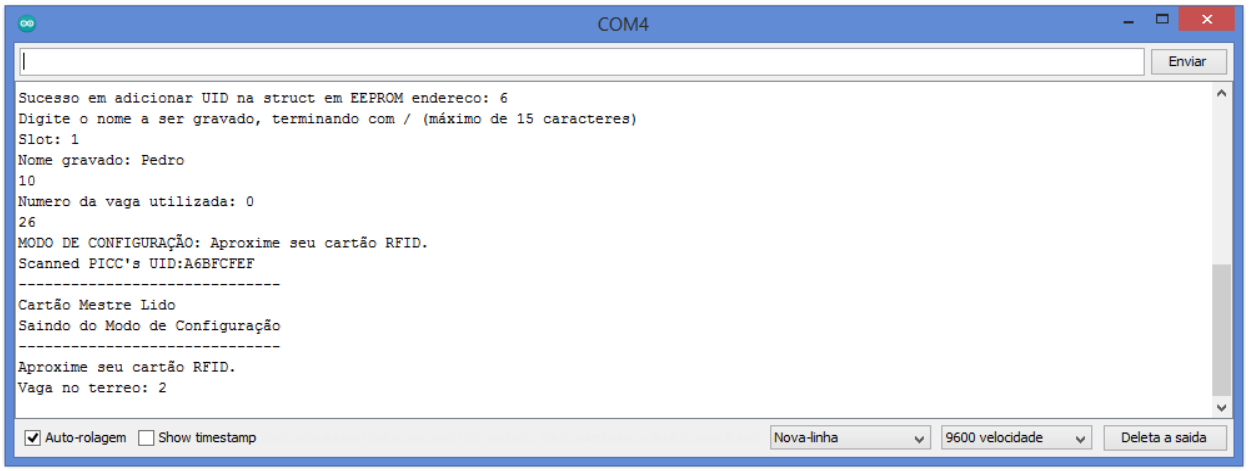
Figura 74: Mensagem confirmando a validação da *tag* cadastrada.



Fonte: o autor.

A Figura 75 apresenta a sequência de mensagens geradas pelo sistema confirmando a saída do modo configuração, após passar a *tag* mestre no leitor. Também informa que o sistema está esperando para fazer a leitura de uma *tag* do usuário, para buscar uma vaga, e informa qual é a vaga que está no térreo.

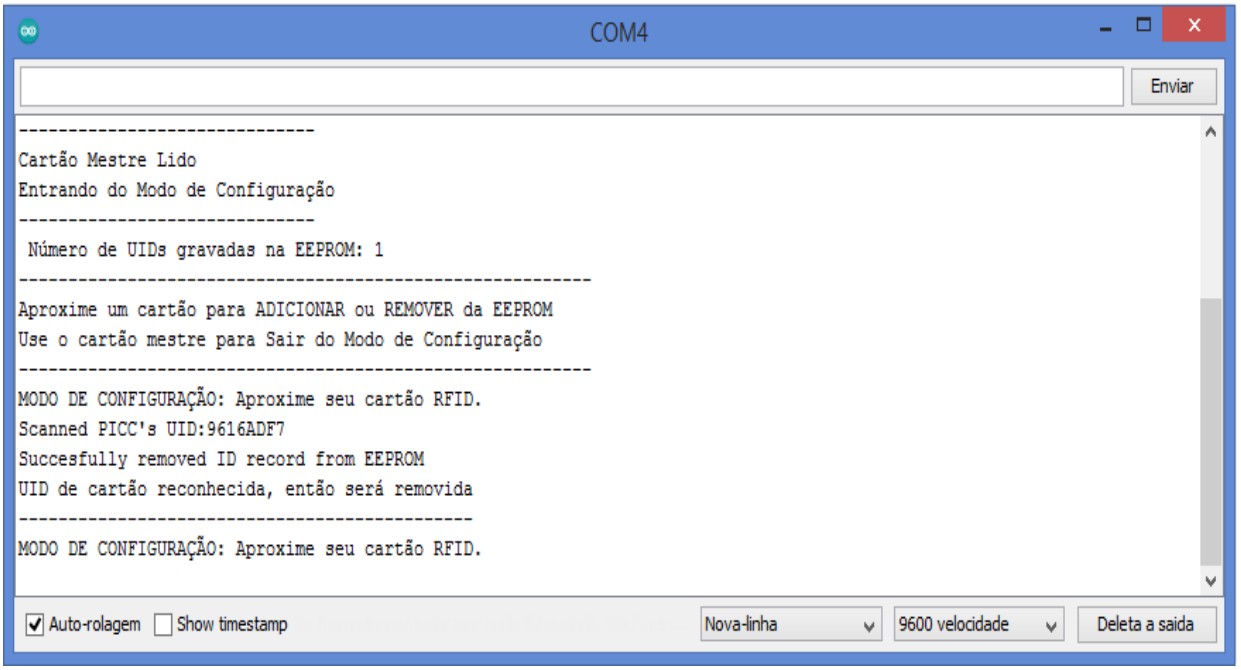
Figura 75: Mensagem confirmando a saída do modo de configuração após passar *tag* mestre no leitor.



Fonte: o autor.

A Figura 76 apresenta a sequência de mensagens gerada pela interface informando a entrada no modo configuração, após ser registrada a *tag* mestre, e o processo de exclusão de uma *tag* que está cadastrada no sistema.

Figura 76: Mensagem de exclusão de uma tag cadastrada no sistema .

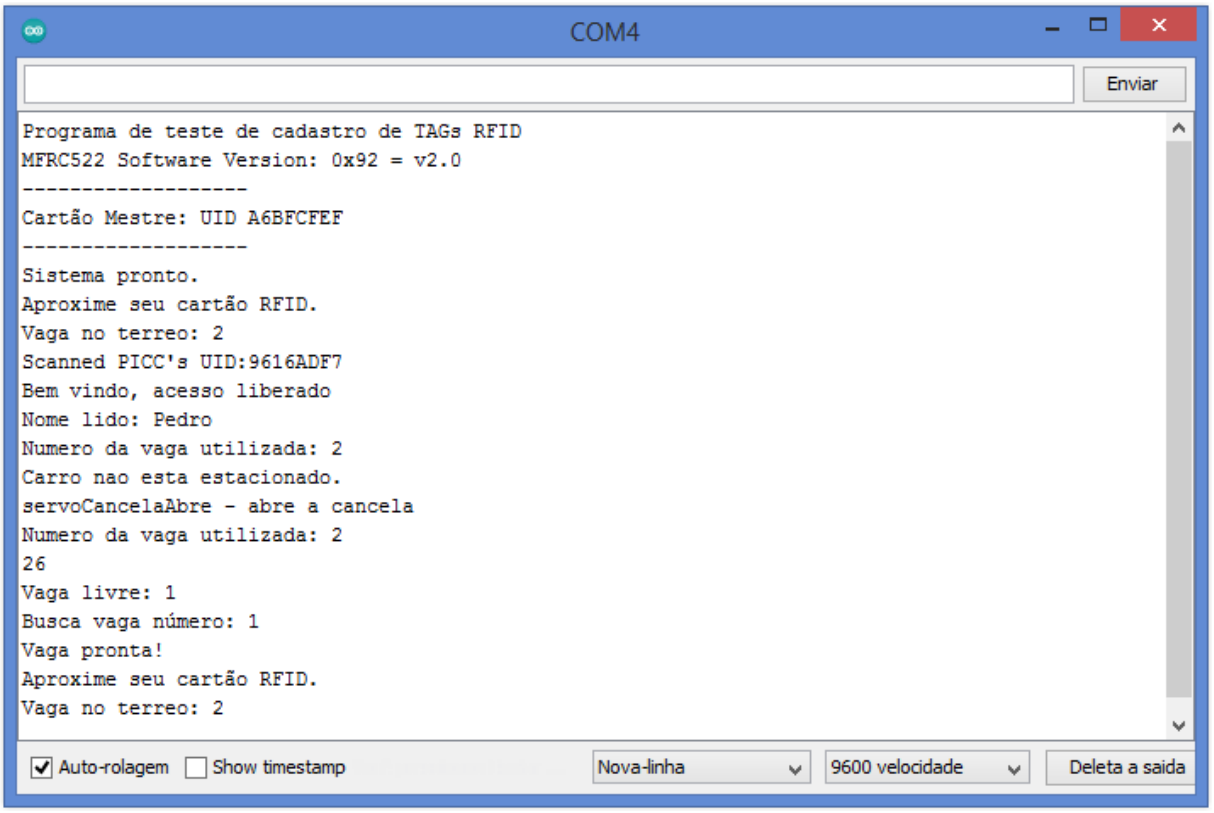


Fonte: o autor.

4.2 CONTROLE DE ACESSO E BUSCA DE VAGAS

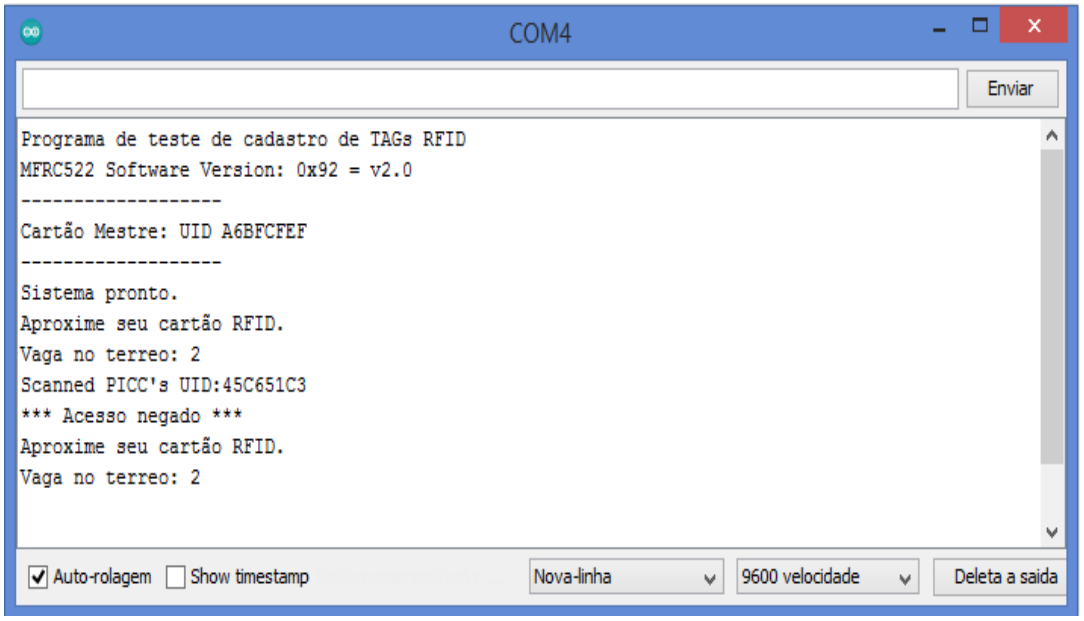
Nesta seção serão apresentados exemplos de funcionamento do controle de acesso e busca de vagas a partir da detecção de *tags* RFID. Na Figura 77 é apresentada a liberação de acesso para o morador colocar ou retirar um veículo. Na figura 78 é apresentada a sequência de mensagens demonstrando o acesso negado para uma *tag* que não está cadastrada no sistema.

Figura 77: Interface demonstrando o acesso liberado de uma *tag* cadastrada.



Fonte: o autor.

Figura 78: Interface demonstrando o acesso negado para uma *tag* não cadastrada.



Fonte: o autor.

De forma exploratória, apresentou-se os resultados obtidos com a observação do protótipo construído tendo em vista compreender tanto a identificar o modo de funcionamento de estacionamentos verticais, do tipo de carrossel, quanto visibilizar e entender a aplicação da tecnologia envolvida em um modelo para observação. Com isso se pode prospectar a possibilidade de implementação dos mesmos em condomínios residenciais urbanos.

5. CONCLUSÕES

Nestas considerações finais, são retomados pontos importantes do projeto que propõe a construção de estacionamentos verticais, com a finalidade de proporcionar aos condomínios residenciais urbanos uma maior otimização de seus espaços internos. Dentre os modelos de estacionamento vertical existentes no mercado, conforme demonstrado na seção 2.1, o modelo que ficou definido para este projeto foi o estacionamento vertical rotativo do tipo carrossel.

O protótipo montado neste trabalho apresenta uma demonstração de funcionamento de um sistema de estacionamento vertical com controle de acesso e com um sistema de segurança pessoal. Como esta proposta visa a aplicação em condomínios residenciais urbanos, haverá circulação de pessoas nas proximidades das partes móveis, portanto para não haver risco de acidente, foram implementados sensores de segurança em pontos estratégicos do estacionamento. Os ensaios realizados na execução deste protótipo permitiu observar o funcionamento do sistema de segurança em ação. Demonstrou-se, ainda, que, como há um sistema eletrônico de controle de acesso, o morador poderá ter uma vaga personalizada no sistema do estacionamento vertical.

O experimento tecnológico indicou que a implementação de estacionamentos verticais, pode, de fato, otimizar espaços em condomínios residências, em contraponto aqueles que utilizam estacionamentos horizontais, comprovando a hipótese de que estacionamentos verticais do tipo carrossel otimizam espaços condominiais em áreas urbanas.

Porém, conforme o levantamento de preços realizado nesta pesquisa, exposto na Tabela 1, este tipo de estacionamento ainda tem um custo bastante elevado. Assim, considerada a atual conjuntura socioeconômica do Brasil, ele é mais viável para condomínios cuja taxa condominial corresponda a de famílias com renda mais alta. Contudo, à medida que a tecnologia utilizada para a construção de estacionamentos verticais se popularize no Brasil, o seu uso para otimizar espaços pode ser viável e adequado para os diversos níveis socioeconômicos de famílias que residam em condomínios residenciais fechados, especialmente em centros urbanos.

Finalmente se pode inferir, que os resultados deste Trabalho de Conclusão de Curso e seus objetivos, que levaram à construção de um protótipo de sistema de estacionamento vertical, foram atingidos conforme o esperado; pois permitiu avaliar o funcionamento deste tipo de sistema, considerando: os aspectos de controle de acesso de pessoas e de veículos; a eletrônica necessária para fazer o gerenciamento de acesso usando a tecnologia RFID a partir de um sistema microcontrolador; e a avaliação de mecanismos de segurança.

5.1 TRABALHOS FUTUROS

Para o protótipo implementado poderiam ser realizados estudos e aplicações com melhorias, tais como: a otimização dos sensores indutivos de controle de parada de quatro para apenas um único sensor, através de um sistema de contagem; a reavaliação da aplicação de outra tecnologia de sensores como por exemplo os sensores ópticos; o uso da tecnologia já usada para o controle de acesso neste protótipo, que é a tecnologia com leitores RFID com etiquetas (*tags*), poderia ser implementada também para o controle de paradas posicionando um leitor abaixo das plataformas com etiquetas abaixo das plataformas.

Além destas, poderia ser implementado um sistema para controlar a suavidade do giro do motor através da variação da carga do peso exercida pelo número de carros que o estacionamento está movimentando, dentre as tecnologias para esta aplicação poderiam ser utilizados *encoders* ópticos. A inclusão de um sistema de controle de velocidade permitiria limitar a velocidade máxima de movimentação, a fim de controlar o arranque e parada de forma suave.

Outra melhoria seria o aumento do número de vagas neste protótipo, para 6 vagas. Para projetos futuros envolvendo tecnologia com estacionamento vertical, poderia ser implementado um sistema com aplicativo de celular onde a vaga do morador poderia ser posicionada de maneira remota por este aplicativo e o controle de acesso ao estacionamento poderia ser feito através de QR CODE ao invés de *tags*.

REFERÊNCIAS

3DLAB. **Soluções em Impressão 3D**. Minas Gerais, BH. 2021. Disponível em: http://3dlab.com.br/produto/driver-a4988-c dissipador/?gclid=CjwKCAiAsaOBBhA4EiwAo0_AnKuS2fjEB1ELMPJ5YJ1Ao2iweh7gG__951ugLal4el6JKEDTpAXx-xoC3zwQAvD_BwE . Acessado em 28 mar. 2021.

ARDUINO language reference. EEPROM Library.. 2021. Disponível em: https://www.arduino.cc/en/Reference/EEPROM. Acessado em 30 jul. 2020.

ALIBABA.COM. **Garagem automática rotary Carrossel vertical soluções de estacionamento de automóveis.** ALIBABA.COM, 2021a. Disponível em: https://portuguese.alibaba.com/product-detail/Garage-automatic-rotary-vertical-Carousel-car-62065346596.html?spm=a2700.8699010.2017203.1.72525fc0liJhno&s=p . Acessado em 27 de mar. de 2021.

ALIBABA.COM. **Auto vertical elevador do carro do estacionamento, multi-nível de rotação Plataforma Pit Parking System**. ALIBABA.COM., 2021b. Disponível em: https://portuguese.alibaba.com/product-detail/auto-vertical-car-parking-lift-multilevel-rotating-platform-pit-parking-system 60689193780.html?spm=a2700.8699010.normalList.19.72525fc0liJhno . Acessado em 27 mar. 2021.

ALIBABA.COM. **Product-detail/solid-smart-parking-equipment**. ALIBABA.COM, 2021c. Disponível em: https://portuguese.alibaba.com/product-detail/solid-smart-parking-equipment-for-6-16-suv-vertical-rotary-car-lift-parking-62190789849.html . Acessado 27 mar. 2021.

ALIBABA.COM. **Rotação Vertical equipamentos do sistema de estacionamento elevador do carro.** Imagem a. ALIBABA.COM., 2021d. Disponível em: https://portuguese.alibaba.com/product-detail/vertical-rotating-car-parking-system-equipment-lift-60802417124.html?spm=a2700.8699010.normalList.55.22415fc0ePChMF. Acessado em 27 mar. 2021.

ALIBABA.COM. **4 post quad pilha sistema de estacionamento de elevação vertical equipamentos de elevação de estacionamento de veículos.** Imagem.b. ALIBABA.COM., 2021e. Disponível em: https://portuguese.alibaba.com/product-detail/4-post-quad-stack-parking-system vertical-lifting-equipment-vehicle-lifting-parking-60778847983.html?spm=a2700.galleryofferlist.0.0.564476188OvQ35 . Acessado em 28 mar. 2021.

BAÚ DA ELETRÔNICA. **Módulo sensor de obstáculo infravermelho IR**. São Paulo: BAÚ DA ELETRÔNICA, 2021. Disponível em: https://www.baudaeletronica.com.br. Acessado em 28 mar. 2021.

BEYOND DC. **Auto parking automatic**. BEYOND DC, 2021. Disponível em: http://beyonddc.com/log/?p=8733. Acessado em 28 mar. 2021.

BG SYSTHEM**. Soluções RFID para o controle e rastreabilidade de Ativos Patrimoniais, Inventário de Ativos Fixos, tanto para produtos metálicos e não metálicos.** BG SYSTHEM, 2021. Disponível em: https://bgsystems.com.br. Acessado em 28 mar.2021.

EE.IC.AC.UK. **Servo motor SG90**. EE.IC.AC.UK, 2021. Disponível em: http://www.ee.ic.ac.uk/pcheung/teaching/DE1_EE/stores/sg90_datasheet.pdf . Acessado em 28 mar. 2021.

EMBARCADOS. **Arduino Mega 2560**. EMBARCADOS, 2021. Disponível em: **https://www.embarcados.com.br/arduino-mega-2560/**. Acessado em 28 mar. 2021.

ENGINEERS GARAGE. **Stepper motor: types and working**. ENGINEERS GARAGE, 2021. Disponível em: https://www.engineersgarage.com/article_page/stepper-motor-basics-types-and-working. Acessado em 28 mar. 2021.

ENGETAX ELEVADORES**. Segurança e inovação em movimento**. ENGETAX ELEVADORES, 2021. Disponível em: https://engetax.com.br. Acessado em 25 mar. 2021.

ELETRONICS HUB. **IR (infrared) obstacle detection sensor circuit**. ELETRONICS HUB, 2021. Disponível em: ElectronicsHub.org. Acessado em 28 mar. 2021.

EUROPAGES. **Shenzhen roanpu technology CO**. EUROPAGES, 2021. Disponível em: https://www.europages.co.uk/SHENZHEN-ROANPU-TECHNOLOGY-CO/0 0000003713399-112520001.html . Acessado em 28 mar. 2021.

FACHIN, Odília. **Fundamentos de metodologia**: noções básicas em pesquisa científica. São Paulo: Saraiva, 2017.

FEIS.UNESP.BR. **Servo motor.** FEIS.UNESP.BR, 2021. Disponível em: https://www.feis.unesp.br/Home/departamentos/engenhariaeletrica/aula-4---servo-motor-13-03-2013-final.pdf **.** Acessado em 1 maio 2021.

GIGANTE DO MUNDO. **Maior estacionamento vertical do mundo**. GIGANTE DO MUNDO, 2021. Disponível em: https://gigantesdomundo.blogspot.com/2014/12/maior-estacionamento-vertical-do-mundo.html. Acessado em 27 mar. 2021.

HAMELINK. Leon J. **The mechanical parking Guide 2011**- increase the number of parking places on de value of your real estate Project. Free Chapter, v. 2, n. 2, 2011. (ISBN 1-466-43786-3). Disponível em: https://www.mechanical-parking-systems.com/wp-content/uploads/Free-Chapter-v2.2.pdf. Acessado em 03 abr. 2021

HOPELAND. **Intelligent UHF RFID hardware system provider**. HOPELAND, 2021. Disponível em: https://www.hopelandRFID.com/. Acessado em 28 mar. 2021.

IMPRESSORA A JATO. **Impressora de etiquetas Brother QL-810W Wireless.** Rio de Janeiro: IMPRESSORA A JATO, 2021. Disponível em: https://www.impressorajato.com.br/rotulador-brother-ql-810w-wireless. Acessado em 28 de março de 2021.

IBGE. Brasil. **Arranjos populacionais e concentrações urbanas no pais**. 2ª ed. – Rio de Janeiro: IBGE, 2016. Disponível em:https://www.ibge.gov.br/apps/arranjos_populacionais/2015/pdf/publicacao.pdfAcessado em 23 mar de 2016.

JIU ROAD PARKING. **Shandong Jiuroad Parking Equipment Co.,LTDA**. Disponível em: https://jiuroadparking.en.china.cn/ . Acessado em 27 mar. 2021

JUNIOR, Alberto Wiltgn. **Projeto do processamento digital de tag RFID à norma ISO LIEC 1800-2.** Universidade Federal do Rio Grande do Sul. Instituto de Informática (Trabalho de conclusão de curso).Porto Alegre, 2010.

LEMAITRE, Fernando Justiniano. **Estudo de um sistema de RFID HF: Aumento do alcance de leitura dos transponders.** Universidade Federal de São Carlos. Escola de Engenharia. (Monografia apresentada no curso de engenharia elétrica com ênfase em eletrônica).São Paulo, 2018.

LEVY, Dan Rodrigues. Condomínios residenciais fechados e a recontextualização do exercício da cidadania nos espaços urbanos. **Revista Ponto e Vírgula**, n. 7, p. 95 -108, 2010.

MASTERWALKER**. Master walker eletronic shop**. MASTERWALKER, 2021. Disponível em: https://www.masterwalkershop.com.br. Acessado em 28 mar. 2021.

MUNDO PROJETADO. **Sensor obstáculo infravermelho.** MUNDO PROJETADO, 2021. Disponível em: http://mundoprojetado.com.br/sensor-de-obstaculo-infravermelho/. Acessado em 28 mar. 2021.

NETO, Pedro Baumgartner. **Driver de motor de passo controlado por CLP.** Universidade Federal do Rio do Sul. Instituto de Engenharia. (Trabalho de conclusão de curso em engenharia elétrica). Porto Alegre, 2011. Acesso em 28 de março de 2021.

OLIMEX. **Pillow block bearing KFL08.** OLIMEX, 2021. Disponível em: https://www.olimex.com/Products/Robot-CNC-Parts/Bearings/KFL08/**.** Acessado em 28 mar. 2021.

PEGASUS THECNOLOGY. **Soluções de análise de vídeo**. PEGASUS THECNOLOGY, 2021. Disponível em: https://www.pegasustec.com.br . Acessado em 28 mar. 2021.

POLOLU. **DMOS Microstepping Driver with Translator And Overcurrent Protection .** POLOLU.2021. Disponível em: https://www.omc-stepperonline.com/download/17HS19-2004S1.pdf. Acessado em 28 mar. 2021.

PREDIGER, Daniel; FREITAS, Edson Pignato; SILVEIRA, Sidnei Renato. **Modelo de Aplicabilidade de Sistema RFID para Rastreabilidade na Indústria Alimentícia.** UFSM. Texto digitalizado. Data de acesso 28 de março 2021. Link:https://www.ufsm.br/app/uploads/sites/333/2018/11/ModelodeAplicabilidadedeSistemaRFIDparaRastreabilidadena_IndústriaAlimentícia.pdf .

SENSE.COM.BR. **Manual de Instruções**- sensores indutivos. SENSE.COM.BR, 2021. Disponível em: https://www.sense.com.br/app/webroot/idiomas/pt_BR/arquivos/produtos/arq2/Indutivos.pdf. Acessado em 28 mar. 2021.

SOLID PARKING. **Mechanical parking solution expert**. Disponível em: http://www.solidparking.com/about-us/. Acessado em 27 mar. 2021.

STEPPERONLINE. **Stepper Motor 17HS19-2004S1**. STEPPERONLINE, 2021. Disponível em: https://www.omc-stepperonline.com/download/17HS19-2004S1.pdf. Acessado em 28 mar. 2021.

STORE ARDUINO. **Arduino Mega 2560 REV3**. Disponível em: https://store.arduino.cc/usa/mega-2560-r3. Acessado em 28 mar.2021.

TADA. **Tada auto parking**. TADA, 2021. Disponível em: http://www.tadaparking.com/els/about.html . Acessado em 27 mar. 2021.

TRADEWORK. **Impressora Zebra industrial e RFID S4M (S4M00-200A-0100T)**. TRADEWORK, 2021. Disponível em: https://www.tradework.com.br/impressora-zebra-industrial-rfid-s4m00200a0100t-p-3998.html . Acessado em 28 mar. 2021.

TRFORUM ORG**. Transportation Research Forum**. TRFORUM ORG, 2021. Disponível em: *ttp://www.trforum.org/*. Acessado em 28 mar. 2021.

ZEBRA. **Leitores de RFID e scanners habilitados para RFID de mão**. ZEBRA, 2021. Disponível em: https://www.zebra.com/. Acessado em 28 mar. 2021.

APÊNDICE A - CÓDIGO FONTE

```
/*
  Programa de cadastramento de TAGs RFID

  Cristopher Marques da Silva
  e-mail: cmdsilva@restinga.ifrs.edu.br

  Prof. Alexsandro C. Bonatto
  e-mail: alexsandro.bonatto@restinga.ifrs.edu.br

  Versão 5: 13/04/2021

  Descrição da pinagem do módulo MFRC522:
  -----------------------------------------------------------------------------------------
          MFRC522         Arduino         Arduino   Arduino         Arduino          Arduino
          Reader/PCD   Uno/101    Mega    Nano v3 Leonardo/Micro   Pro Micro
  Signal          Pin      Pin       Pin       Pin       Pin              Pin
  -----------------------------------------------------------------------------------------
  RST/Reset   RST         9         5         D9        RESET/ICSP-5   RST
  SPI SS          SDA(SS)         10        53        D10       10              10
  SPI MOSI        MOSI            11 / ICSP-4   51            D11       ICSP-4          16
  SPI MISO        MISO            12 / ICSP-1   50            D12       ICSP-1          14
  SPI SCK         SCK     13 / ICSP-3   52            D13       ICSP-3          15
*/

#include <EEPROM.h>     // We are going to read and write PICC's UIDs from/to EEPROM
#include <SPI.h>        // RC522 Module uses SPI ###protocol
#include <MFRC522.h>    // Library for Mifare RC522 Devices
#include <Servo.h>      // Biblioteca para controle do servomotor
#include <Wire.h> //Defino os pinos das portas SDA e SDL para comunicação serial

/*
  LEDs para sinalização das operações realizadas de acesso pela TAG mestre, cadastro e descadastro de TAGs.
*/
#define ledVerde  18   // Indica que o programa está ativo e pisca ao liberar acesso para uma TAG cadastrada
#define ledAmarelo  19   // TAG mestre e modo de cadastro de vagas (descadastro: pisca 5x; cadastro: pisca 2x);
#define ledVermelho 20   // Informa que a TAG não está cadastrada.

#define MOTOR_PASSO_TON 10000 // ms
#define MOTOR_PASSO_TOFF 10000 // ms
#define delay_ajuste 50 // ms

// Pinos usados para o motor de passos e o drive
const int stepPin = 6 ;
const int dirPin = 7 ;
const int reset = 14 ;
const int enable = 15 ;

/*
  Variáveis usadas pela leitura de TAGs RFID
*/
byte cartaoGravado[4];   // Variável que armazena um UID de cartão RFID que foi gravado na EEPROM
byte cartaoLido[4];      // Variável que armazena um UID de cartão RFID que foi lido pelo módulo de RFID
byte cartaoMestre[4];    // Variável que armazena um UID de cartão RFID mestre.

uint8_t leituraValida;   // Variável que indica se a leitura do UID de um cartão RFID foi bem sucedida

bool modoConfiguracao = false;  // indica se o programa está no modo de configuração de cartões RFID
```

```
const int numeroVagas = 4; // Define o número de vagas do estacionamento

// Cria um array de structs
typedef struct {
  byte cartaoRfid[4];          // grava o UID do cartao RFID do morador
  char nome[16];               // nome da pessoa
  uint8_t posicao_vaga;   // usada sequencia de contagem
  bool carroEstacionado;  // indica se o carro está estacionado (True) ou não está (False)
} dadosMorador;

dadosMorador morador[numeroVagas];

bool vagaTerreoLivre = true;
uint8_t numeroVaga;
uint8_t vagaTerreo = 0;
uint8_t numeroVagasOcupadas = 0; // numero total de vagas ocupadas
bool carroEstacionado;
const uint8_t VAGA_0 = 0;

// Pinos dos sensores indutivos
const int sensor_vaga[4] = {31, 33, 35, 37}; //  para o Arduino Mega
//const int sensor_vaga[4] = {5, 4, 3, 2}; //  para o Arduino Uno

// Create MFRC522 instance.
#define SS_PIN 53 //PINO SDA // MEGA
#define RST_PIN 5 //PINO DE RESET // MEGA

//#define SS_PIN 10 // UNO
//#define RST_PIN 9 // UNO
MFRC522 mfrc522(SS_PIN, RST_PIN);

#define SERVO 8 // Porta Digital 8 servo freio
#define SERVO_CANC 9 // Porta Digital 9 servo cancela // MEGA
//#define SERVO_CANC 1 // Porta Digital 1 servo cancela // UNO
Servo sc; // Variável Servo CANCELA
Servo sfreio; // Variável Servo FREIO

int sensor_obst1 = 22;
int sensor_obst2 = 24;
int sensor_obst3 = 28;
int sensor_obst4 = 30;

bool paradaEmergencia = false;

/*
   Configuração
*/
void setup() {
  pinMode ( sensor_obst4, INPUT ) ;
  //Definição dos pinos para o drive do motor de passos
  pinMode ( stepPin, OUTPUT ) ;
  pinMode ( dirPin, OUTPUT ) ;
  pinMode ( reset, OUTPUT ) ;
  pinMode ( enable, OUTPUT ) ;

  //Comando inicial para os pinos do motor de passos, para que o motor não movimente ao iniciar o Arduino
  digitalWrite(reset, LOW);
  digitalWrite(enable, HIGH);
```

```
 // Configuração dos pinos do Arduino
 pinMode(ledVerde, OUTPUT);
 pinMode(ledAmarelo, OUTPUT);
 pinMode(ledVermelho, OUTPUT);

 digitalWrite(ledVerde, LOW);
 digitalWrite(ledAmarelo, LOW);
 digitalWrite(ledVermelho, LOW);

 //Definição dos pinos dos sensores indutivos
 pinMode ( sensor_vaga[1], INPUT ) ;
 pinMode ( sensor_vaga[2], INPUT ) ;
 pinMode ( sensor_vaga[3], INPUT ) ;
 pinMode ( sensor_vaga[4], INPUT ) ;

 sc.attach(SERVO_CANC);//Inicia o servo motor cancela
 sc.write(0);

 sfreio.attach(SERVO);//Inicia o servo do freio
 sfreio.write(30);

 // Protocolos de comunicação
 Serial.begin(9600);  // Initialize serial communications with PC
 Wire.begin(); //INICIALIZA A BIBLIOTECA WIRE
 SPI.begin();       // MFRC522 Hardware uses SPI protocol
 mfrc522.PCD_Init();       // Initialize MFRC522 Hardware

 //If you set Antenna Gain to Max it will increase reading distance
 //mfrc522.PCD_SetAntennaGain(mfrc522.RxGain_max);

 Serial.println(F("Programa de teste de cadastro de TAGs RFID"));
 ShowReaderDetails();  // Testa acesso e mostra detalhes da placa MFRC522

 // Verifica se o cartão Mestre já está definido, se não, usuário deverá aproximar um cartão Mestre
 // Esta rotina também é útil para redefinir o cartão mestre, trocando o "número mágico"
 // Para redefinir o cartão mestre, basta trocar o número mágico, que fica gravado no endereço 1 da EEPROM
 uint8_t numeroMagico = 22; // número mágico, entre 0 e 255.
 if (EEPROM.read(1) != numeroMagico) {
Serial.println(F("Nenhum cartão Mestre definido"));
Serial.println(F("Aproxime um PICC RFID para definir o novo Cartão Mestre"));
do {
        leituraValida = getID();     // grava leituraValida=1 quando acontecer uma leitura, caso contrário =0
        digitalWrite(ledVerde, HIGH);       // Mostra que o sistema está esperando para programar o cartão
mestre
        delay(200);
        digitalWrite(ledVerde, LOW);
        delay(200);
}
while (!leituraValida);                  // o programa ficará neste laço enquanto não houver uma leitura válida

for ( uint8_t j = 0; j < 4; j++ ) {
        EEPROM.write( 2 + j, cartaoLido[j] );   // Grava na EEPROM o UID do cartão lido
}

EEPROM.write(1, numeroMagico); // Grava na EEPROM o número mágico, indicando que o cartão mestre foi
gravado.
Serial.println(F("Cartão Mestre gravado com sucesso"));
 }
 Serial.println(F("--------------------"));
 Serial.print(F("Cartão Mestre: UID "));
```

```
  for ( uint8_t i = 0; i < 4; i++ ) {      // Lê o UID do cartão mestre gravado na EEPROM
 cartaoMestre[i] = EEPROM.read(2 + i); // Grava o UID na variável masterCard
 Serial.print(cartaoMestre[i], HEX);
  }
  Serial.println("");
  Serial.println(F("--------------------"));
  Serial.println(F("Sistema pronto."));
  piscaLeds();       // pisca todos os leds indicando que o sistema está pronto
}


/* ----------------------------------------------------------------------------
   Programa principal
  ---------------------------------------------------------------------------- */
void loop () {
  if ((digitalRead(sensor_obst4) == LOW))
  {
   // pos_canc = 0;
 sc.write(0);
  }

  // Sinaliza nos LEDs e terminal
  if (!modoConfiguracao) {
 Serial.println(F("Aproxime seu cartão RFID."));
 digitalWrite(ledVerde, HIGH);  // Liga LED verde
 digitalWrite(ledVermelho, LOW);  // Garante LED vermelho desligado
 digitalWrite(ledAmarelo, LOW);  // Garante LED amarelo desligado

 // Informa qual vaga está no térreo
 encontraVagaTerreo();
  }
  else {
 Serial.println(F("MODO DE CONFIGURAÇÃO: Aproxime seu cartão RFID."));
 digitalWrite(ledVerde, LOW);  // Liga LED verde
 digitalWrite(ledVermelho, LOW);  // Garante LED vermelho desligado
 digitalWrite(ledAmarelo, HIGH);  // Garante LED amarelo desligado
  }

  // Escuta aproximação de cartão RFID e segura o programa aqui
  do {
 leituraValida = getID();     // grava leituraValida=1 quando acontecer uma leitura, caso contrário =0
  }
  while (!leituraValida);   // o programa ficará neste laço enquanto não houver uma leitura válida

  // Após ler cartão, verifica se o modo de configuração está ativado
  if (modoConfiguracao) {
 // Está no Modo de Configuração de cartões RFID
 if ( isMaster(cartaoLido) ) { // Testa se é o cartão mestre
            Serial.println(F("-------------------------------"));
            Serial.println(F("Cartão Mestre Lido"));
            Serial.println(F("Saindo do Modo de Configuração"));
            Serial.println(F("-------------------------------"));
            modoConfiguracao = false;
            return;
 }
 else {
            if ( findID(cartaoLido) ) { // Se o cartão lido é conhecido, então: apagar este cartão
            deleteIDinStruct(cartaoLido);
            Serial.println(F("UID de cartão reconhecida, então será removida"));
            Serial.println(F("-----------------------------------------------"));
```

```
            }
            else {                  // Se o cartão lido não é conhecido, então: gravar este cartão
            writeIDinStruct(cartaoLido);
            //Serial.println(F("------------------------------"));
            //Serial.println(F("UID de cartão desconhecida, então será gravada"));
            writePersonalDatainStruct(cartaoLido); // Grava nome do morador
            writeCarroEstacionadoinStruct(cartaoLido, false); // Limpa registro de vaga na struct do morador
            writeVagaUtilizadainStruct(cartaoLido, VAGA_0); // grava vaga 0, ou seja, sem vaga atribuida
            }
}
 }
 else {
if ( isMaster(cartaoLido) ) { // Testa se é o cartão mestre, então entra no Modo de Configuração
            modoConfiguracao = true;
            uint8_t count = EEPROM.read(0); // Lê o primeiro byte da EEPROM que grava o número de UIDs
gravadas na EEPROM
            Serial.println(F("------------------------------"));
            Serial.println(F("Cartão Mestre Lido"));
            Serial.println(F("Entrando do Modo de Configuração"));
            Serial.println(F("------------------------------"));
            Serial.print(F(" Número de UIDs gravadas na EEPROM: "));
            Serial.println(count);
            Serial.println(F("------------------------------------------------------------"));
            Serial.println(F("Aproxime um cartão para ADICIONAR ou REMOVER da EEPROM"));
            Serial.println(F("Use o cartão mestre para Sair do Modo de Configuração"));
            Serial.println(F("------------------------------------------------------------"));
}
else {
            if ( findID(cartaoLido) ) { // Verifica se a UID está cadastrada na EEPROM
            // Libera acesso para colocar ou retirar um veiculo
            Serial.println(F("Bem vindo, acesso liberado"));
            readPersonalDatafromStruct(cartaoLido); // Le os dados pessoais associados a este cartao
            acessoLiberadoVaga(300);
            numeroVaga = readVagaUtilizadafromStruct(cartaoLido); // Le qual vaga foi utilizada pelo morador
            carroEstacionado = readCarroEstacionadofromStruct(cartaoLido); // Lê se o morador deixou ou não um
carro estacionado
            // Verifica se o morador está com o carro estacionado ou não
            if ( carroEstacionado ) {
            // Testa se a vaga do morador já está no térreo
            if ( numeroVaga != vagaTerreo ) {
            // Busca vaga do morador
            operacoesVaga(numeroVaga);
            }
            // Libera a cancela , aciona servo durante 5 s e fecha
            servoCancelaAbre(5000);
            numeroVagasOcupadas--; // Decrementa o contador de vagas ocupadas
            writeCarroEstacionadoinStruct(cartaoLido, false); // Limpa registro de vaga na struct do morador
            } else {
            // Morador não está com o carro estacionado, então abre a vaga do térreo (se disponível)
            if ( vagaTerreoLivre ) { // verifica se a vaga do terreo está livre para estacionar um carro
            servoCancelaAbre(5000); // Libera a cancela , aciona servo durante 5 s e fecha
            writeCarroEstacionadoinStruct(cartaoLido, true); // grava que morador deixou um carro
            writeVagaUtilizadainStruct(cartaoLido, vagaTerreo); // grava vaga utilizada
            numeroVagasOcupadas++; // incrementa contador de vagas utilizadas
            if ( numeroVagasOcupadas < 4 ) { // Testa se existem vagas disponíveis
            numeroVaga = buscaVagaLivre(); // busca pela vaga livre e coloca no terreo
            // aguarda morador sinalizar que saiu do estacionamento, pela tag rfid
            operacoesVaga(numeroVaga); // traz vaga livre para o térreo
            vagaTerreoLivre = true; // sinaliza que a vaga do terreo está livre
            } else {
```

```
            vagaTerreoLivre = false; // sinaliza que a vaga do terreo está ocupada, não existem vagas livres
            }
            } else {
            // Estacionamento está cheio, então informa e sinaliza com LEDs
            Serial.println("Lamento, mas o estacionamento está cheio.");
            acessoNegado(300);
            }
            }
            }
            else {     // Se a UID não está cadastrada, não libera acesso
            Serial.println(F("*** Acesso negado ***"));
            acessoNegado(300);
            }
}
  }
}


/* ---------------------------------------------------------------------------------------
 *  buscaVagaLivre - função que busca por uma vaga livre do estacionamento.
 --------------------------------------------------------------------------------------- */
uint8_t buscaVagaLivre( void ){
  uint8_t vagaLivre=1;
  // busca nas structs dos moradores quais vagas estão ocupadas

  for (uint8_t j=1; j<=4; j++) {
 for (uint8_t i=1; i<=4; i++) {
            if ( morador[i].posicao_vaga == j ){
            vagaLivre++;
            }
}
 if (vagaLivre == j) {
            break;
}
  }
  Serial.print("Vaga livre: ");
  Serial.println(vagaLivre);

  return vagaLivre;
}

/* ---------------------------------------------------------------------------------------
 * operacoesVaga - função para movimentar o estacionamento e buscar a vaga solicitada
 *                      para o térreo, fazendo: (1) liberar o freio; (2) acionar o servomotor
 *                      no sentido horário; (3) verificar sensor de posicionamento correspondente
 *                      a vaga solicitada.
 --------------------------------------------------------------------------------------- */
void operacoesVaga(uint8_t numeroVaga) {
  bool acionaMotorEstacionamento = true;

  Serial.print("Busca vaga número: ");
  Serial.println(numeroVaga);
  sfreio.write(40); // libera o freio movimentoando o servo para 40o
  digitalWrite(enable, LOW); // habilita driver do servo motor
  digitalWrite(reset, HIGH);
  digitalWrite(dirPin, LOW);

  while (acionaMotorEstacionamento)
  {
 // Verifica sensores de barreira laterais e traseiro
```

```
if ( (digitalRead(sensor_obst1) == LOW) ||
        (digitalRead(sensor_obst2) == LOW) ||
        (digitalRead(sensor_obst3) == LOW) ){
        acionaMotorEstacionamento = false;
        paradaEmergencia = true; // sinaliza na variável global
        digitalWrite(stepPin, LOW);
        digitalWrite(reset, LOW);
        digitalWrite(enable, HIGH); // desabilita driver do servo motor
        sfreio.write(30); // aciona o freio eletromecânico
        Serial.println("Alerta: Movimento interrompido!");
} else if (digitalRead(sensor_vaga[numeroVaga]) == HIGH) {
        // Verifica se o estacionamento já está com a vaga  no solo
        acionaMotorEstacionamento = false;
        digitalWrite(stepPin, LOW);
        digitalWrite(reset, LOW);
        digitalWrite(enable, HIGH); // desabilita driver do servo motor
        sfreio.write(30); // aciona o freio eletromecânico
        Serial.println("Vaga pronta!");
} else {
        // aciona motor de passos enquanto o sensor de posição da vaga não for acionado
        digitalWrite(stepPin, HIGH);
        delayMicroseconds(MOTOR_PASSO_TON);
        digitalWrite(stepPin, LOW);
        delayMicroseconds(MOTOR_PASSO_TOFF);
        delay(delay_ajuste);
}
 }
}

/* ---------------------------------------------------------------------------------
  encontraVagaTerreo - descobre qual vaga está estacionada no térreo do elevador,
                       testando um por um os sensores indutivos. Supondo que uma das
                       vagas está parada no solo, caso contrário, deveria acionar o
                       motor para deslocar as vagas.
 --------------------------------------------------------------------------------- */
void encontraVagaTerreo() {
 for (uint8_t i = 0; i < 4; i++) {
if ( digitalRead(sensor_vaga[i]) == HIGH ) {
        vagaTerreo = i + 1; // grava na variável global qual vaga está no térreo
}
 }
 Serial.print("Vaga no terreo: ");
 Serial.println(vagaTerreo);
}

/* ---------------------------------------------------------------------------------
  servoCancelaAbre - libera a cancela pelo tempo informado
 --------------------------------------------------------------------------------- */
void servoCancelaAbre( int tempo ) {
 Serial.println("servoCancelaAbre - abre a cancela");
 sc.write(90); // Abre a cancela
 delay(tempo); // Espera pelo tempo em us
 while ( digitalRead(sensor_obst4) == LOW ){ // trava aqui esperando liberar o sensor
Serial.println("servoCancelaAbre - libere a cancela");
delay(1000);
 }
 sc.write(0); // Fecha a cancela
}

/* ---------------------------------------------------------------------------------
```

```
  Acesso liberado - Busca a vaga de estacionamento
  ------------------------------------------------------------------------------------------ */
void acessoLiberadoVaga ( uint16_t setDelay) {
  digitalWrite(ledVerde, LOW);  // Garante LED verde desligado
  digitalWrite(ledVermelho, LOW);  // Garante LED vermelho desligado
  for (uint8_t i = 0; i < 5; i++) { // pisca o led amarelo 5 vezes
 digitalWrite(ledAmarelo, HIGH);  // Liga LED amarelo
 delay(setDelay);
 digitalWrite(ledAmarelo, LOW);  // Garante LED amarelo desligado
 delay(setDelay);
  }
  delay(1000);
}

/* ------------------------------------------------------------------------------------------
  Acesso negado
  ------------------------------------------------------------------------------------------ */
void acessoNegado(uint16_t setDelay) {
  digitalWrite(ledAmarelo, LOW);  // Garante LED verde desligado
  digitalWrite(ledVerde, LOW);  // Garante LED vermelho desligado
  for (uint8_t i = 0; i < 5; i++) { // pisca o led amarelo 5 vezes
 digitalWrite(ledVermelho, HIGH);  // Liga LED amarelo
 delay(setDelay);
 digitalWrite(ledVermelho, LOW);  // Garante LED amarelo desligado
 delay(setDelay);
  }
  delay(1000);
}


/* ------------------------------------------------------------------------------------------
  Faz a leitura do código UID de um PICC (Proximity Integrated Circuit Card)
  ------------------------------------------------------------------------------------------ */
uint8_t getID() {
  // Getting ready for Reading PICCs
  if ( ! mfrc522.PICC_IsNewCardPresent()) { //If a new PICC placed to RFID reader continue
 return 0;
  }
  if ( ! mfrc522.PICC_ReadCardSerial()) {   //Since a PICC placed get Serial and continue
 return 0;
  }
  // There are Mifare PICCs which have 4 byte or 7 byte UID care if you use 7 byte PICC
  // I think we should assume every PICC as they have 4 byte UID
  // Until we support 7 byte PICCs
  Serial.print(F("Scanned PICC's UID:"));
  for ( uint8_t i = 0; i < 4; i++) {  //
 cartaoLido[i] = mfrc522.uid.uidByte[i];
 Serial.print(cartaoLido[i], HEX);
  }
  Serial.println("");
  mfrc522.PICC_HaltA(); // Stop reading
  return 1;
}

/* ------------------------------------------------------------------------------------------
  Apresenta detalhes do MFRC522 e testa comunicação com a placa
  ------------------------------------------------------------------------------------------ */
void ShowReaderDetails() {
  // Get the MFRC522 software version
  byte v = mfrc522.PCD_ReadRegister(mfrc522.VersionReg);
```

```
  Serial.print(F("MFRC522 Software Version: 0x"));
  Serial.print(v, HEX);
  if (v == 0x91)
 Serial.print(F(" = v1.0"));
  else if (v == 0x92)
 Serial.print(F(" = v2.0"));
  else
 Serial.print(F(" (unknown),probably a chinese clone?"));
  Serial.println("");
  // When 0x00 or 0xFF is returned, communication probably failed
  if ((v == 0x00) || (v == 0xFF)) {
 Serial.println(F("WARNING: Communication failure, is the MFRC522 properly connected?"));
 Serial.println(F("SYSTEM HALTED: Check connections."));
 // Visualize system is halted
 digitalWrite(ledVerde, LOW);  // Garante LED verde desligado
 digitalWrite(ledVermelho, HIGH);  // Liga LED vermelho
 digitalWrite(ledAmarelo, LOW);  // Garante LED amarelo desligado
 while (true); // do not go further
  }
}

/* ---------------------------------------------------------------------------------------
   Pisca Leds
  ---------------------------------------------------------------------------------------- */
void piscaLeds() {
  digitalWrite(ledVerde, HIGH);  // Liga LED verde
  digitalWrite(ledVermelho, LOW);  // Garante LED vermelho desligado
  digitalWrite(ledAmarelo, LOW);  // Garante LED amarelo desligado
  delay(200);
  digitalWrite(ledVerde, LOW);  // Garante LED verde desligado
  digitalWrite(ledVermelho, HIGH);  // Liga LED vermelho
  digitalWrite(ledAmarelo, LOW);  // Garante LED amarelo desligado
  delay(200);
  digitalWrite(ledVerde, LOW);  // Garante LED verde desligado
  digitalWrite(ledVermelho, LOW);  // Garante LED vermelho desligado
  digitalWrite(ledAmarelo, HIGH);  // Liga LED amarelo
  delay(200);
}


/* ---------------------------------------------------------------------------------------
   readIDinStruct - Busca um UID na EEPROM
  ---------------------------------------------------------------------------------------- */
void readIDinStruct( uint8_t number ) {
  uint8_t start = (number * 4 ) + 2;   // Figure out starting position
  for ( uint8_t i = 0; i < 4; i++ ) {     // Loop 4 times to get the 4 Bytes
 cartaoGravado[i] = EEPROM.read(start + i);   // Assign values read from EEPROM to array
  }
}

/* ---------------------------------------------------------------------------------------
   writeIDinStruct - grava novo cartão em espaço de struct na EEPROM
   ---------------------------------------------------------------------------------------*/
void writeIDinStruct( byte a[] ) {
  if ( !findID( a ) ) {           // Before we write to the EEPROM, check to see if we have seen this card before!
 uint8_t num = EEPROM.read(0);    // Get the numer of used spaces, position 0 stores the number of ID cards
 uint8_t start = ( num * 22 ) + 6;  // Calcula o início do novo bloco de struct
 num++;                // Incrementa o contador de 1
 EEPROM.write( 0, num );              // escreve o novo contador na EEPROM
 for ( uint8_t j = 0; j < 4; j++ ) {   // Loop 4 times
```

```
          EEPROM.write( start + j, a[j] );  // Grava o código UID nos primeiros 4 bytes do bloco da struct
}
//successWrite();
Serial.print(F("Sucesso em adicionar UID na struct em EEPROM endereco: "));
Serial.println(start);
 }
 else {
//failedWrite();
Serial.println(F("Failed! There is something wrong with ID or bad EEPROM"));
 }
}

/* ---------------------------------------------------------------------------------------------
   writePersonalDatainStruct - grava dados pessoais no em espaço de struct na EEPROM
   ---------------------------------------------------------------------------------------------*/
void writePersonalDatainStruct(byte uid[]) {
 char buffer[16];
 byte len;
 uint8_t start;

 Serial.setTimeout(60000L) ;          // wait until 20 seconds for input from serial

 // Pede para digitar o nome que será gravado
 Serial.println(F("Digite o nome a ser gravado, terminando com / (máximo de 15 caracteres)"));
 len = Serial.readBytesUntil('/', buffer, 16) ; // read family name from serial
 for (byte i = len; i < 16; i++) buffer[i] = ' ';  // pad with spaces

 uint8_t slot = findIDSLOT(uid); // encontra o slot de informacoes na EEPROM
 Serial.print("Slot: ");
 Serial.println(slot);

 // Grava nome na struct
 strncpy(morador[slot].nome, buffer, 16);
 Serial.print("Nome gravado: ");
 Serial.println(morador[slot].nome);

 // Grava o nome na EEPROM
 start =  10 + ( (slot-1) * 22 );  // Calcula o início do novo bloco de struct
 Serial.println(start);
 for ( uint8_t j = 0; j < 16; j++ ) {   // Loop 16 times
EEPROM.write( start + j, buffer[j] );  // Grava o nome do morador nos 16 bytes do bloco da struct
 }
}

/* ---------------------------------------------------------------------------------------------
   readPersonalDatafromStruct - lê os dados pessoais gravados no espaço de struct na EEPROM
   ---------------------------------------------------------------------------------------------*/
void readPersonalDatafromStruct(byte uid[]) {
 char buffer[16];
 uint8_t slot = findIDSLOT(uid); // encontra o slot de informacoes na EEPROM
 uint8_t start =  10 + ( (slot-1) * 22 );  // Calcula o início do novo bloco de struct

 // Le o nome que está na EEPROM
 for ( uint8_t j = 0; j < 16; j++ ) {   // Loop 16 times
buffer[j] = EEPROM.read( start + j );  // Lê o nome do morador nos 16 bytes da EEPROM
 }

 strncpy(morador[slot].nome, buffer, 16);
 Serial.print("Nome lido: ");
 Serial.println(morador[slot].nome);
```

```
}

/* ------------------------------------------------------------------------------------------------
   writeVagaUtilizadainStruct - escreve qual vaga o morador está utilizando
   ------------------------------------------------------------------------------------------------*/
void writeVagaUtilizadainStruct(byte uid[], uint8_t numeroVaga) {
  uint8_t slot = findIDSLOT(uid); // encontra o slot de informacoes na EEPROM

  // Escreve o numero da vaga na struct
  morador[slot].posicao_vaga = numeroVaga;
  Serial.print("Numero da vaga utilizada: ");
  Serial.println(morador[slot].posicao_vaga);

  // Escreve o numero da vaga na EEPROM
  uint8_t start =  26 + ( (slot-1) * 22 );  // Calcula o início do novo bloco de struct
  Serial.println(start);
  EEPROM.write(start, numeroVaga );  // Grava o numero da vaga
}


/* ------------------------------------------------------------------------------------------------
   readVagaUtilizadafromStruct - lê qual vaga o morador está utilizando, se 0, então não há carro
                                 estacionado
   ------------------------------------------------------------------------------------------------*/
uint8_t readVagaUtilizadafromStruct(byte uid[]) {
  uint8_t numeroVaga;
  uint8_t slot = findIDSLOT(uid); // encontra o slot de informacoes na EEPROM
  uint8_t start =  26 + ( (slot-1) * 22 );  // Calcula o início do novo bloco de struct

  // Lê o numero da vaga da EEPROM
  numeroVaga = EEPROM.read(start);  // Lê o numero da vaga, se 0, então não há carro estacionado

  // Grava na struct
  morador[slot].posicao_vaga = numeroVaga;
  Serial.print("Numero da vaga utilizada: ");
  Serial.println(morador[slot].posicao_vaga);

  return numeroVaga;
}

/* ------------------------------------------------------------------------------------------------
   writeCarroEstacionadoinStruct - escrevea se o morador deixou ou retirou um carro estacionado
   ------------------------------------------------------------------------------------------------*/
void writeCarroEstacionadoinStruct(byte uid[], bool carroEstacionado) {
  uint8_t slot = findIDSLOT(uid); // encontra o slot de informacoes na EEPROM
  uint8_t start =  27 + ( (slot-1) * 22 );  // Calcula o início do novo bloco de struct

  // Escreve o estado do carro: estacionado (true) ou não estacionado (false)
  EEPROM.write(start, carroEstacionado );  // Grava na eeprom
  morador[slot].carroEstacionado = carroEstacionado;
}

/* ------------------------------------------------------------------------------------------------
   readCarroEstacionadofromStruct - lê se o morador deixou um carro estacionado
   ------------------------------------------------------------------------------------------------*/
bool readCarroEstacionadofromStruct(byte uid[]) {
  bool carroEstacionado;
  uint8_t slot = findIDSLOT(uid); // encontra o slot de informacoes na EEPROM
  uint8_t start =  27 + ( (slot-1) * 22 );  // Calcula o início do novo bloco de struct
```

```
  // Lê da EEPROM se o carro esta estacionado
  carroEstacionado = EEPROM.read(start);  // Lê da eeprom

  morador[slot].carroEstacionado = carroEstacionado;
  if (carroEstacionado) {
 Serial.println("Carro esta estacionado.");
  } else {
 Serial.println("Carro nao esta estacionado.");
  }

  return carroEstacionado;
}

//////////////////////////////////////// Add ID to EEPROM   ///////////////////////////////////
//void writeID( byte a[] ) {
//  if ( !findID( a ) ) {          // Before we write to the EEPROM, check to see if we have seen this card before!
//          uint8_t num = EEPROM.read(0);    // Get the numer of used spaces, position 0 stores the number of ID
cards
//          uint8_t start = ( num * 4 ) + 6;  // Figure out where the next slot starts
//          num++;                // Increment the counter by one
//          EEPROM.write( 0, num );              // Write the new count to the counter
//          for ( uint8_t j = 0; j < 4; j++ ) {   // Loop 4 times
//          EEPROM.write( start + j, a[j] );  // Write the array values to EEPROM in the right position
//          }
//          successWrite();
//          Serial.println(F("Succesfully added ID record to EEPROM"));
//  }
//  else {
//          failedWrite();
//          Serial.println(F("Failed! There is something wrong with ID or bad EEPROM"));
//  }
//}

/* ---------------------------------------------------------------------------------------------
 deleteIDinStruct - Apaga um UID da EEPROM
  ---------------------------------------------------------------------------------------------- */
void deleteIDinStruct( byte a[] ) {
  if ( !findID( a ) ) {          // Before we delete from the EEPROM, check to see if we have this card!
 //failedWrite();    // If not
 Serial.println(F("Failed! There is something wrong with ID or bad EEPROM"));
  }
  else {
 uint8_t num = EEPROM.read(0);   // Get the numer of used spaces, position 0 stores the number of ID cards
 uint8_t slot;        // Figure out the slot number of the card
 uint8_t start;       // = ( num * 22 ) + 6; // Figure out where the next slot starts
 uint8_t looping;   // The number of times the loop repeats
 uint8_t j;
 uint8_t count = EEPROM.read(0); // Read the first Byte of EEPROM that stores number of cards
 slot = findIDSLOT( a );   // Figure out the slot number of the card to delete
 start = ((slot-1) * 22) + 6;  // reserva 6 endereços para UID do mestre, count e número mágico
 looping = ((num - slot) * 22);
 num--;   // Decrement the counter by one
 EEPROM.write( 0, num );   // Write the new count to the counter
 for ( j = 0; j < looping; j++ ) {         // Loop the card shift times
          EEPROM.write( start + j, EEPROM.read(start + 22 + j));   // Shift the array values to 4 places earlier in
the EEPROM
 }
 for ( uint8_t k = 0; k < 22; k++ ) {               // Shifting loop
          EEPROM.write( start + j + k, 0);
 }
```

```
//successDelete();
Serial.println(F("Succesfully removed ID record from EEPROM"));
  }
}


////////////////////////////////////// Remove ID from EEPROM   ////////////////////////////////
//void deleteID( byte a[] ) {
//  if ( !findID( a ) ) {         // Before we delete from the EEPROM, check to see if we have this card!
//          failedWrite();      // If not
//          Serial.println(F("Failed! There is something wrong with ID or bad EEPROM"));
//  }
//  else {
//          uint8_t num = EEPROM.read(0);   // Get the numer of used spaces, position 0 stores the number of ID
cards
//          uint8_t slot;         // Figure out the slot number of the card
//          uint8_t start;        // = ( num * 4 ) + 6; // Figure out where the next slot starts
//          uint8_t looping;   // The number of times the loop repeats
//          uint8_t j;
//          uint8_t count = EEPROM.read(0); // Read the first Byte of EEPROM that stores number of cards
//          slot = findIDSLOT( a );   // Figure out the slot number of the card to delete
//          start = (slot * 4) + 2;
//          looping = ((num - slot) * 4);
//          num--;    // Decrement the counter by one
//          EEPROM.write( 0, num );   // Write the new count to the counter
//          for ( j = 0; j < looping; j++ ) {         // Loop the card shift times
//          EEPROM.write( start + j, EEPROM.read(start + 4 + j));   // Shift the array values to 4 places earlier in
the EEPROM
//          }
//          for ( uint8_t k = 0; k < 4; k++ ) {                 // Shifting loop
//          EEPROM.write( start + j + k, 0);
//          }
//          successDelete();
//          Serial.println(F("Succesfully removed ID record from EEPROM"));
//  }
//}

////////////////////////////////////// Check Bytes   ////////////////////////////////
bool checkTwo ( byte a[], byte b[] ) {
  for ( uint8_t k = 0; k < 4; k++ ) {   // Loop 4 times
 if ( a[k] != b[k] ) {          // IF a != b then false, because: one fails, all fail
          return false;
 }
  }
  return true;
}

////////////////////////////////////// Find Slot   ////////////////////////////////
uint8_t findIDSLOT( byte find[] ) {
  uint8_t count = EEPROM.read(0);             // Read the first Byte of EEPROM that
  for ( uint8_t i = 1; i <= count; i++ ) {          // Loop once for each EEPROM entry
 readIDinStruct(i);             // Read an ID from EEPROM, it is stored in cartaoGravado[4]
 if ( checkTwo( find, cartaoGravado ) ) {   // Check to see if the cartaoGravado read from EEPROM
          // is the same as the find[] ID card passed
          return i;             // The slot number of the card
 }
  }
}

////////////////////////////////////// Find ID From EEPROM   ////////////////////////////////
```

```
bool findID( byte find[] ) {
  uint8_t count = EEPROM.read(0);              // Read the first Byte of EEPROM that
  for ( uint8_t i = 1; i <= count; i++ ) {          // Loop once for each EEPROM entry
 readIDinStruct(i);              // Read an ID from EEPROM, it is stored in cartaoGravado[4]
 if ( checkTwo( find, cartaoGravado ) ) {   // Check to see if the cartaoGravado read from EEPROM
            return true;
 }
 else {     // If not, return false
 }
  }
  return false;
}

////////////////////// Check readCard IF is masterCard   ///////////////////////////////////
// Check to see if the ID passed is the master programing card
bool isMaster( byte test[] ) {
  return checkTwo(test, cartaoMestre);
}
```