CENTRO UNIVERSITÁRIO UNIFACVEST
CURSO CIÊNCIA DA COMPUTAÇÃO
ALEXSANDRO ECKHARDT

AUTOMACAR:
UMA TECNOLOGIA PARA CARROS POPULARES

LAGES
2020

CENTRO UNIVERSITÁRIO UNIFACVEST
CURSO CIÊNCIA DA COMPUTAÇÃO
ALEXSANDRO ECKHARDT

Projeto apresentado à Banca Examinadora do Trabalho de Conclusão de Curso II de Ciência da Computação para análise e aprovação.

LAGES
2020

CENTRO UNIVERSITÁRIO UNIFACVEST
CURSO CIÊNCIA DA COMPUTAÇÃO
ALEXSANDRO ECKHARDT

Trabalho de conclusão de curso apresentado
à Banca Examinadora da Unifacvest como
parte dos requisitos para obtenção do título de
Bacharel em Ciência da Computação.
Orientador: Me. Igor Muzeka Coorientadores:
Me: Willen LeolattoCarneiro
Me. Márcio José Sembay

Lages, SC __/__/2020.

Nota _____

_____
Coordenador do curso de Ciência da Computação

LAGES
2020

Este trabalho dedico a minha família,
em especial aos meus pais, irmãos e esposa,
que apesar das dificuldades que tive
nessa caminhada sempre acreditaram em mim.

“Crescer custa, demora, esfola, mas compensa.
É uma vitória secreta, sem testemunhas.
O adversário somos nós mesmos.”
(Martha Medeiros)

SUMÁRIO

RESUMO

A tecnologia da informação vem sendo aplicada mais no dia-a-dia da população afim de prover tecnologias que facilitam e tragam comodidade em suas tarefas diárias, através de plataformas embarcadas, como Arduino por exemplo. Este projeto tem por objetivo automatizar o sistema de um automóvel, eliminando a necessidade do uso de chaves para abrir as portas. Será analisada a possibilidade de acessar um automóvel utilizando um *smartphone*, travar e destravar as portas de forma onde somente pessoas autorizadas possam realizá-lo. Serão utilizados os métodos dedutivos e quantitativos, onde serão buscados através de livros e artigos obter conclusões dedutivas e através de testes com o Arduino UNO R3, alcançar os métodos quantitativos. Será utilizada uma IDE disponível no site da Arduino, a qual faz a compilação do código no computador, para posteriormente sem nenhum erro ser transferido para o Arduino Uno R3 para execução. Será utilizado um módulo com *chip* ESP8266 para estabelecer a conexão Wi-Fi com a *smartphone*, como este *chip* necessita de uma carga maior de energia, será colocado um capacitor para suprir caso tenha necessidade de uma carga extra durante o funcionamento do *chip* ESP8266.

Palavras-chave: Arduino, *smartphone*, automóvel, Wi-Fi

INTRODUÇÃO

A tecnologia da informação vem sendo empregada para a comodidade e facilidade no dia a dia do cidadão. Tecnologias essas, que são desenvolvidas sob plataformas embarcadas, onde é possível desenvolver diversos sistemas para facilitar tarefas diárias. Como exemplos de aplicações embarcadas encontramos o controle de temperatura interna do veículo de forma automatizada, acendimento automático dos faróis, controle das fechaduras das portas, entre vários outros, através de um Smartphone, via Wi-Fi, ou ligado na Internet, remotamente.

Com a intenção de facilitar algumas interações com o automóvel, este projeto tem como objetivo desenvolver uma forma simples de acessar os recursos e periféricos de um veículo sem o uso de uma chave. Será garantida a segurança para a realização da conexão com *Smartphone*, estabelecendo um meio seguro para realizar a comunicação entre o sistema embarcado Arduino e o *Smartphone.*

O projeto proposto, irá enviar comandos para abrir e fechar as travas elétricas das portas e implementar sensores dentro do veículo para monitoramento remoto. Comandos estes, que serão emitidos através de um *Smartphone*, utilizando um aplicativo. O *Smartphone* será conectado via Wi-Fi com o Arduino através de uma interface WI-FI, na aplicação estarão disponíveis os botões para as respectivas funções desejadas, e também a temperatura interna e externa do automóvel.

Este trabalho está estruturado em três capítulos. No Capítulo 1, estão descritos os elementos do projeto. No Capítulo 2, está disposta a fundamentação teórica, onde foram colocadas as referências de diversos autores renomados sobre o assunto em estudo. No Capítulo 3, será apresentado de forma descritiva a análise dos estudos realizados para corroborar ou refutar as hipóteses levantadas.

ASPECTOS METODOLÓGICOS

Este capítulo contém os caminhos metodológicos necessários para uma melhor compreensão e a realização do estudo.

1.1 TEMA

Controle automotivo utilizando *smartphone* e plataforma Arduino, programado para abrir e fechar as portas, visando mais comodidade e segurança de acesso.

1.1.1 Delimitação do tema

Buscar através de conceitos estudados em sistemas embarcados e distribuídos, desenvolver uma aplicação para acessar um automóvel através de um aplicativo instalado em um *smartphone*, pelo qual serão enviados comandos para abrir e/ou fechar as travas elétricas das portas, tendo como princípio a segurança para o acesso restrito aos usuários autorizados.

Este trabalho foi realizado pelo acadêmico Alexsandro Eckhardt, da oitava fase do Curso de Ciência da Computação do Centro Universitário Unifacvest, de Junho a Novembro de 2020.

1.2 OBJETIVOS

Os objetivos que foram elencados para a pesquisa, são os que seguem.

1.2.1 Objetivo geral

Realizar um projeto que traz facilidade ao acessar e obter a localização do seu veículo de uma forma segura utilizando um *smartphone.*

1.2.2 Objetivos específicos

- Implantar um microcontrolador, Arduino uno R3, ligado a uma interface *wireless* ESP 8266, para comunicar-se com um *smartphone.*

- Realizar os testes de modo simulado e posteriormente na prática em um automóvel.

- Analisar os resultados da pesquisa e dos testes e descrever no Capítulo 3 deste projeto.

1.3 JUSTIFICATIVA

Este trabalho aborda as facilidades trazidas pelos sistemas embarcados, tanto em tarefas cotidianas que podem ser automatizadas até implementado uma ferramenta de funcional importância para o usuário final, buscando abranger um leque de possibilidades, desde monitoramento através de sensores de temperatura, umidade, velocidade; até o controle automático do sistema de freios, climatização, piloto automático de um automóvel.

Buscando elaborar uma aplicação que trabalhe de forma em que uma ou mais aplicações trabalhem de uma forma única, sistemas distribuídos máscara e traz a sensação de que o sistema é único, simplificando para o usuário milhões de compartilhamentos de áreas da memória, sincronizações, trocas de mensagens, e também favorecendo com que inúmeros recursos distintos trabalham em conformidade mesmo em situações de processamento diferentes. Afim de garantir com que toda esta informação seja acessada somente por pessoas ou sistemas autorizados, a segurança da informação tem por objetivo prover formas de autenticação seguras com facilidade de acesso barrando usuários não desejáveis.

1.4 PROBLEMA DE PESQUISA

Utilizando um *smartphone*, é possível acessar um automóvel, travando e destravando as portas, de maneira com que somente pessoas autorizadas possam realizá-lo?

1.5 HIPÓTESES

As hipóteses elencadas para esse trabalho foram:

- É possível comunicar o sistema do Arduino com um *smartphone* de forma segura.

- É possível realizar o travamento das portas de um veículo usando um Arduino e um Smartphone.

1.6 METODOLOGIA

Os pesquisadores chegam a seus postulados através da exploração de métodos buscados no ramo da filosofia da ciência denominado metodologia.

1.6.1 Métodos de abordagem

Para este trabalho foram utilizados os métodos dedutivo e quantitativo.

1.6.1.1 Dedutivo

O método dedutivo, busca resultados menores, partidos do meio abstrato, vindo de um conceito ou lei abrangente ou também de alguma hipótese elencada temporariamente. Algumas conclusões foram tiradas através de livros e artigos renomados, assim sendo utilizados o método dedutivo.

1.6.1.2 Quantitativa

Esta forma de abordagem é obtida através de estatísticas e formulações de problemas que buscam orientar ações entre variáveis.

1.6.1 Métodos de procedimento

Utilizou-se da pesquisa experimental e bibliográfica.

1.6.1. Pesquisa bibliográfica

Buscou-se contemplar os principais autores de maior renome referente aos assuntos abordados neste trabalho, tais como, Andrew S.Tanenbaum, Charles Pfleeger entre outros.

1.6.1.2 Pesquisa experimental

Utilizando um Arduino UNO R3 ligado a uma interface *wireless* ESP 8266, que se comunicará com um s*martphone*. Este por sua vez envia comandos para a placa Arduino para travar ou destravar as travas elétricas das portas do automóvel.

1.6.1.3 Estudo de caso

Este estudo visa adquirir o conhecimento profundo na realidade delimitada. Seu foco está voltado a juntar o meio tecnológico e automobilístico. Desenvolvendo uma aplicação que conectando com a placa Arduino no veículo, irá controlar através da conexão pelo s*martphone*, as travas elétricas do automóvel. O grupo analisará e aplicará uma forma de como estabelecer uma conexão segura entres estes meios.

1.6.2 Técnicas

Utilizou-se da técnica prática e de observação.

1.6.2.1 Observação

A observação baseia-se em examinar todos os fenômenos ou fatos que se desejam analisar, tanto na estrutura física quanto na elétrica do automóvel, afim de entender o funcionamento do mesmo. Para posteriormente ligar o Arduino no sistema das travas elétricas.

1.6.2.2 Prática

A prática é um experimento que deve ser organizada por uma série de experimentos, os quais devem ser anotados afim de chegar ao objetivo.

1.7 ORÇAMENTO

Neste item serão apresentadas as informações referentes aos custos obtidos no decorrer do desenvolvimento do trabalho.

Tabela 1 - Orçamento

Descricão	Valor Unitário	Quantidade	Frete	Total
Arduino Uno R3	R$ 30,00	1	R$ 23,00	R$ 53,00
ESP8266	R$ 15,00	1	R$ 21,50	R$ 36,50
Protoboard	R$ 11,50	1	R$ 19,22	R$ 30,72
Jumpers	R$ 16,50	1	R$ 18,00	R$ 34,50
Placa GPRS	R$ 35,00	1	RS15,00	R$50,00
TOTAL				**R$ 204,72**

1.8 CRONOGRAMA

O cronograma de um projeto define-se em um quadro, que tem como objetivo mostrar as atividades realizadas em um determinado período de tempo.

Tabela 2 - Cronograma

ATIVIDADES	Março	Junho	Setembro	Novembro
Escolha do Tema				
Elaboração do Capítulo1				
Compra dos equipamentos para testes				
Realização do Capítulo2				
Tabulação e Análise dos Dados				
Redação do Relatório Final				
Implantação e desenvolvimento da prática				

Finalizado Em andamento A realizar

1.9 RECURSOS HUMANOS

Os recursos humanos são a disponibilidade dos professores orientadores.

1.9.1 Recursos institucionais

Os recursos institucionais utilizados para a realização desse trabalho, foi a biblioteca.

1.9.2 Recursos Materiais

Os recursos materiais utilizados serão livros, folhas de ofício, *notebook*, Arduino, Interface *wireless, Protoboard, Jumpers* entre outros.

2 REFERENCIAL TEÓRICO

Nesse capítulo estará disposta toda a parte teórica, esse conteúdo foi coletado de livros retirados em bibliotecas e versões digitais.

Um sistema embarcado é definido como um sistema que é utilizado para executar tarefas, atingindo um determinado produto ou resultado. Esse tipo de implementação é muito comum, e pode ser encontrado principalmente em dispositivos que executam tarefas especificas, como por exemplo, controlar a abertura e fechamento de portões de garagens, automóveis, receber e processar dados de sensores, que podem ser de temperatura, umidade, velocidade, ultrassônico, entre vários outros.

As principais características de um sistema embarcado são ter uma funcionalidade única, ou resumindo, deve executar um *loop* infinito executando várias vezes a mesma função; outra característica seria, possuir um processamento com excelente velocidade na realização de cálculos, usando pouca energia; a terceira característica imprescindível de um sistema embarcado é a resposta em tempo real, pois vários sistemas necessitam respostas imediatas, como frenagem automática de um automóvel por exemplo.

2.1.1 Redes de Computadores

As redes de computadores são responsáveis pela conexão de computadores e vários outros equipamentos, como impressoras, *smartphones*, *notebook*, servidores, etc. As redes pessoais PANs (*Personal Area Network*), são responsáveis pela comunicação aos computadores de pequeno porte, normalmente usadas por uma pessoa. As redes locais LANs (*Local Area Network*) são as redes responsáveis por operar dentro de pequenos locais, como prédios, fábricas, residências.

> Segundo Tanenbaum (2001, pag 163), as redes metropolitanas MANs (*Metropolitan Area Network*) são responsáveis pela abrangência dentro de uma cidade. As redes de longas distâncias WAN (*Wide Area Network*)

> atuam em áreas de grande abrangência geográfica, conectando cidades, estado, países e continentes. Para que esta conexão aconteça, existem vários meios de propagar estes dados, podendo ser por cabos de rede de par trançado, normalmente usados dentro de residências, até redes de fibra óptica e sinal de rádio, normalmente usados para interligar cidades.

Outros tipos de rede são a sem fio ou Wi-Fi, via satélite e por par de cobre. Por haver vários tipos de tecnologias de comunicação, foram projetadas um conjunto de regras para padronizar esta tecnologia, estas regras são chamadas de protocolo, que controla o formato e o significado dos pacotes trocados através de mensagens. O modelo que referência OSI (*Open Systems Interconnection*) possui sete camadas para controlar esta padronização, a camada física, enlace de dados, camada de rede, camada de transporte, sessão, apresentação e aplicação. Outro modelo de referência é TCP/IP, constituído pela camada de transporte, aplicação e *host*/rede.

Segundo Tanenbaum (2001), existem vários tipos de topologia de rede, a topologia em barramento atua sobre as redes locais, onde todos os dispositivos conectados são compartilhados através do mesmo barramento, o problema é que todos os computadores ligados a esta topologia receberão o pacote destinado a somente um computador, mas somente o comutador alvo poderá abrir este pacote. A topologia em anel consiste em interligar todos os dispositivos em um círculo, trafegando os dados em um único sentido, o problema está quando o rompimento desta rede acontece, comprometendo a rede.

> Apesar do autor Tanenbaum (2001, pag 274), não considerar ponto a ponto como topologia, o autor Tude (2016), define enlace ponto a ponto como a mais comum, onde é estabelecido um *link* entre dois pontos, conectado *links* de telecomunicações e principalmente *links* de radiofrequência. A topologia em estrela funciona através de nós centralizados, normalmente um *switch* comutador, comprometendo toda a rede caso este concentrador para de funcionar. A tecnologia em árvore trabalha de forma mista, mesclando propriedades do barramento estrela e outras redes, utilizado frequentemente em cascateamento de *switchs*. As redes em malha (*Mesh*) trocam dados aleatoriamente entre todos os dispositivos conectados, utilizando um algoritmo de custo, alta disponibilidade e geralmente utilizada em Wi-Fi.

2.2.1 Conceito de um Sistema Embarcado

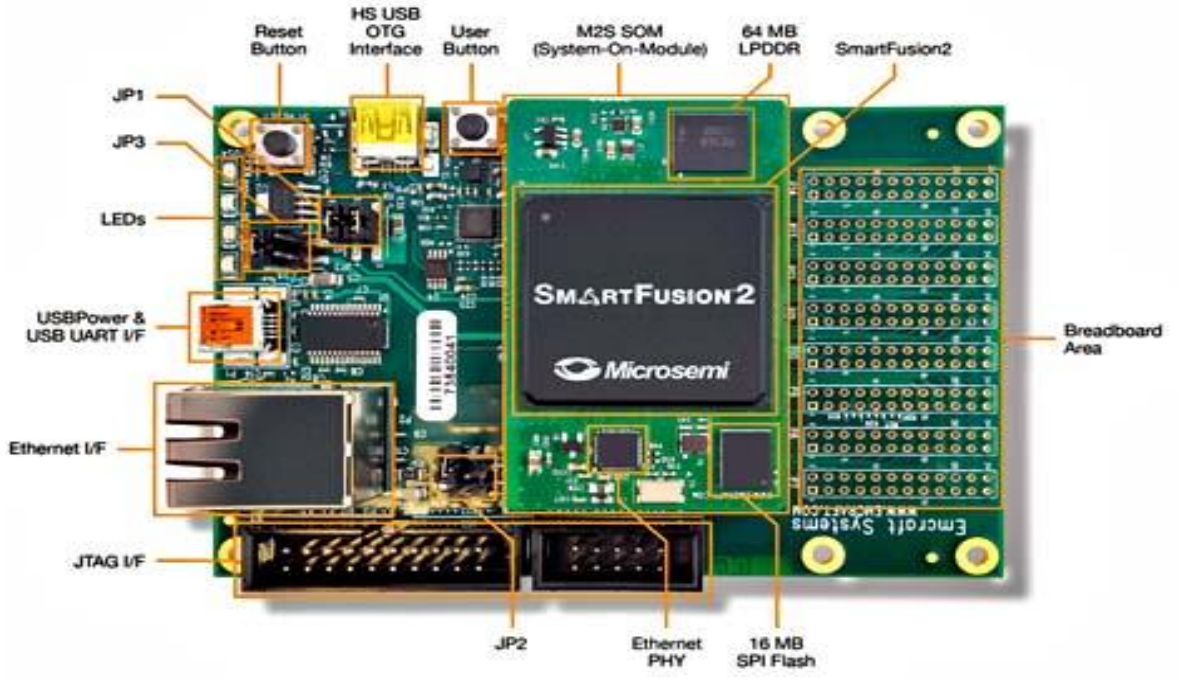
Segundo Prado (2013), um sistema embarcado é definido por um computador construído afim de atender uma única aplicação, assim personalizando sua construção de acordo com suas necessidades computacionais. Antes de 2007,

sistemas operacionais embarcados eram tratados como eletrônicos de forma geral, mas como sua complexidade e poderes computacionais foram cada vez mais diferenciando-se dos eletrônicos sem capacidade computacional tronou-se necessária esta divisão.

2.1.2 SoC (*System on-chip*)

Segundo Graves (2011), SoC (*System on Chip*) nada mais é do todo um sistema dentro de um chip. As características deste sistema consistem em ter memória, microprocessador e seus periféricos. Seu processador pode ser personalizado ou específico para uma única atividade, onde pode haver múltiplos processadores ou ainda controladores DMA. Os controladores DMA se distinguem dos processadores pela sua particularidade em não haver busca as instruções. Os processadores interligam-se de várias formas, através de memórias compartilhadas etc. A figura1 ilustra um exemplo de SoC, o *Microsemi Announces Smartfusion*2.

Figura 1 Exemplo SoC



Fonte: Open Systems Media

2.1.3 Arduino

Segundo Arduino (2016), o Arduino é uma plataforma física, que possui código aberto e é baseado em uma simples placa micro controladora, e um ambiente de desenvolvimento para criar o código, e enviar para a placa. A placa do Arduino

pode ser usada para o desenvolvimento de objetos independentes, ou também para ser conectada a um computador. Uma placa Arduino padrão é composta por um controlador, linhas de E/S digital e analógica, uma serial ou USB, para poder interligar-se ao hospedeiro, que é normalmente usada para programá-la e interagir com a mesma em tempo real. Essa placa em si não possui nenhum recurso de rede mas muito simples e comum de combinar um ou mais Arduinos, usando ferramentas apropriadas, chamadas de *shields*.

O Arduino usado neste projeto será o modelo UNO R3, a alimentação da placa pode ser feita através de conexão USB e também por uma fonte de alimentação. Para fonte externa é necessária respeitar os limites definidos entre 7V e 12V para seu correto funcionamento, o conector deve ser Jack, onde o pino central será o positivo.

> Segundo Souza (2013, pag 413), o Arduino possui conectores de alimentação para módulos e *shields.* O conector *Reset* quando acionado reseta as configurações do Arduino, de forma analógica, sem necessidade de comandos, caso o placa tenha travada, esta é uma alternativa. As saídas 3,3V alimentam as *shields* e módulos externos, porém deve ser respeitada a corrente máxima de 50 mA. O IOREF tem por função principal referenciar qual o tipo de interface mais apropriada para que as *shields* funcionem no Arduino mesmo não sendo alimentadas pela Voltagem de 3,3V, então sendo adaptadas para utilização em 5V, sendo aplicada de forma contrária também.

O conector GND funciona como o terra no circuito. O pino de 5V alimenta as *shields* e demais circuitos com esta voltagem. O pino VIN alimenta a placa através da bateria externa, quando conectado ao conector Jack.

A comunicação USB com o computador utiliza um microcontrolador ATMEL ATMEGA16U2, o qual é responsável pelo *upload* do código binário compilado pelo programa feito pelo usuário. Também podem ser feitas atualizações de *firmware* através do conector ICSP.

Os LEDs TX e RX, que são responsáveis para mostrar o envio e recebimento de dados da placa para o computador, quem faz o controle dos LEDs é o *software* do próprio microcontrolador.

O ATMEL ATMEGA328 é o microcontrolador principal da placa Arduino UNO, o qual possui 8 bits, desenvolvido em arquitetura RISC e encapsulado por DIP28, possui 32 KB de memória Flash, onde 512 Bytes são usados para o *bootloader*, 1 KB para memória EEPROM e 2 KB de memória RAM. A velocidade de processamento pode alcançar no máximo 20 MHz. Possuindo 28 pinos ao total, o microcontrolador

ATMEL ATMEGA328 utiliza 23 para entrada e saída, podendo operar com baixa voltagem, como forma de poupar energia.

Os dispositivos de entrada e saída da placa Arduino UNO são formados por 14 pinos digitas operantes em 5V a 40 mA máximos. Os pinos PWM, que são os 3, 5, 6, 9, 10 e 11 tem 8bits e são acessados através do comando analogWrite() no software. Como comunicação serial os pinos 0 e 1 devem ser conectados ao micro controlador que faz a comunicação USB com o Computador. Os pinos 2 e 3 servem para configurar a interrupção externa através do comando attachInterrupt().

As entradas analógicas são compostas por 6 pinos, trabalhando em 10 bits, ligados em 5V, quando ligada nesta voltagem o pino assume o valor analógico digital de 1023. Por não possuir um botão de liga/desliga, para desligar a placa, deve-se desconectar o cabo de alimentação, USB ou fonte alimentadora.

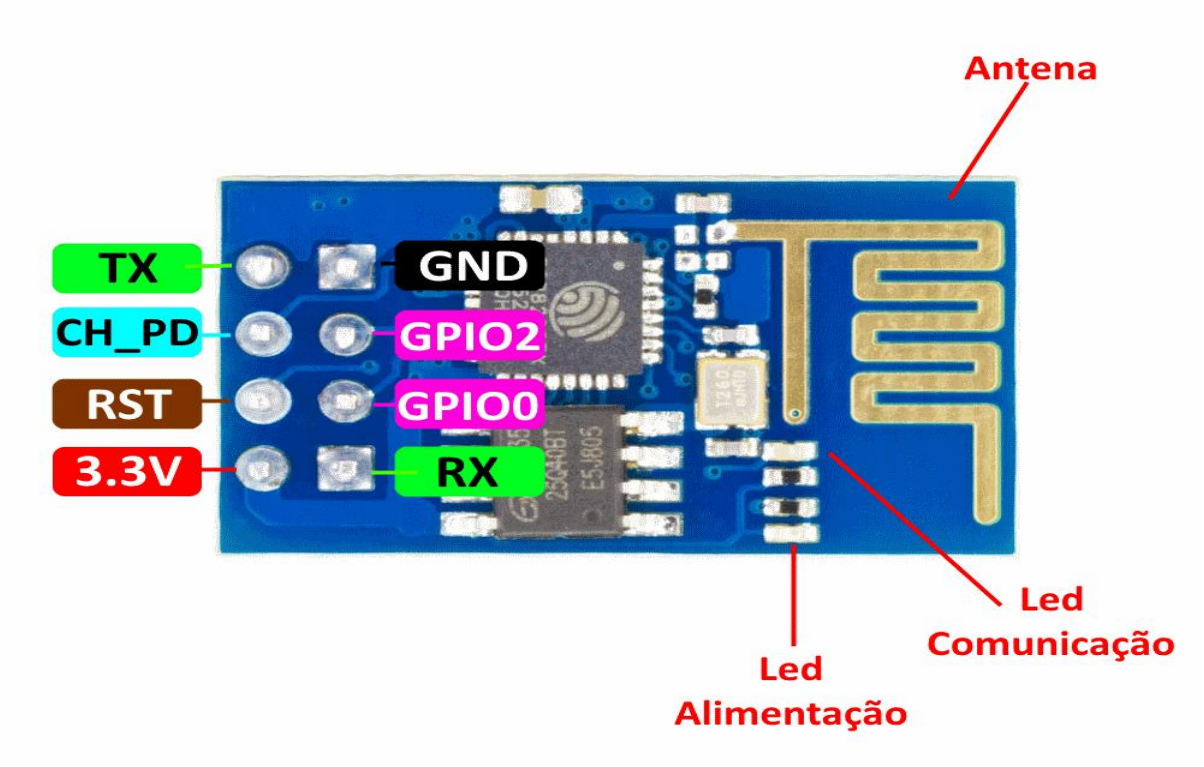
A programação é realizada através do protocolo STK500, o microcontrolador já vem programado com o *bootloader*, onde está comunicação é feita via serial. Outra forma de programação é feita através do ICSP, só deve ser feita através de um programador ATMEL.

2.1.4 Introdução a ESP8266

Os módulos com *chip* ESP8266 estão se popularizando cada vez mais no uso do desenvolvimento da Internet das coisas, pela sua praticidade de uso, tamanho reduzido e baixo consumo de energia. Essa placa envia ou recebe sinal via-rádio a uma distância de até no máximo 90 metros. Sua programação pode ser realizada de diversas maneiras, em nosso trabalho vamos enviar comandos, chamados de AT, comandos esses enviados via porta serial pela IDE do próprio Arduino. Na figura a seguir estão dispostos os dois equipamentos utilizados para a elaboração desse projeto: O Arduino Uno R3 e a Placa Wi-Fi ESP8266.

O ESP8266 foi utilizado nesse trabalho porque consiste em uma placa *Wi-Fi,* que possui um consumo de energia bem reduzido. Essa placa é ligada a uma voltagem de 3,3 V, e seu alcance pode chegar a 90 metros de distância. O preço dessa placa é muito acessível e possui uma ampla diversidade de configurações. Na figura 9 está disposta a imagem do ESP8266, com a mostra descrita dos pinos, utilizado para realizar a comunicação com o Arduino.

Figura 2 ESP8266 com a identificação de seus pinos



Fonte: FILIPEFLOP

2.1.5 Módulo de relés para Arduino

Como os motores das travas elétricas do automóvel trabalham com uma voltagem de 12V, foi necessária a aquisição de um módulo de relé específico para Arduino. Os relés são alimentados com a energia 12V da bateria do próprio automóvel. O que o Arduino faz é acionar os relés conforme a necessidade ou programado. Esse módulo de relés possui uma proteção de corrente, para que não retorne uma corrente maior para o Arduino, correndo assim o risco da queima do mesmo. Esses relés também, diferentes dos comuns, são acionados tranquilamente com uma tensão 5V, que é a fornecida pelo Arduino. A Figura 3 mostra o módulo dos relés com seus respectivos pinos para acionamento dos mesmos.

Figura 3 Módulo Relé de 4 saídas para Arduino

Fonte: Mercado Livre Adaptado

2.1.6 RISC e CISC

Atualmente o mercado dispões processadores com arquitetura hibrida, na sua essência são processadores CISC mas com alguns recursos dos processadores RISC.

> Segundo Marimoto (2007, pag 312), o computador que contém um processador com a arquitetura CISC (*Complex Instrution Set Computer*), ou computador com um conjunto complexo de instruções, é capaz de abranger um leque maior de instruções diversificada, sendo muito versátil. Na década de 80 a principal ideia era produzir chips com instruções cada vez mais complexas, porém, houve fabricantes que fizeram exatamente o contrário, assim desenvolvendo a arquitetura RISC (*Reduced Instruction Set Computer*) ou computador com um conjunto reduzido de instruções. Como o próprio nome já diz, esta arquitetura trabalha com a capacidade de executar somente algumas poucas instruções simples, assim barateando esta tecnologia, outra vantagem, é por ter um número menor de circuitos internos, pode-se trabalhar em frequências mais altas.

Figura 4 Exemplos de Arquitetura

Fonte: Emily Blem, Jaikrishana Menos and Karthikeyan Sankaralingan

2.1.7 Arquitetura ARM

Segundo Marimoto (2007), esta arquitetura é comumente usada na fabricação de celulares e sistemas de automação, este processador é produzido por diversas fabricantes, ele é um *chip* RISC de 32 bits, e demandam um bom desempenho, se o compararmos com um processador x86. Utilizando este processador com 4 MB de memória, pode-se até rodar o sistema operacional Linux, abrindo um leque de possibilidades de ferramentas de desenvolvimento. Por seu baixo consumo de energia, este processador é utilizado em cerca de 75% dos sistemas embarcados.

Existindo vários tipos de *chips* e controladores, cada um tem seu próprio *kit* de ferramentas de desenvolvimentos (SDK), tendo seus próprios compiladores, *debuggers* etc, sendo muitas vezes distribuídos gratuitamente, mas em alguns casos há de ser comprado ou licenciado. Esta ferramenta de desenvolvimento normalmente é utilizada em um PC, transferindo seus dados apenas no estágio final de desenvolvimento para o sistema embarcado afim de testes. Essa transferência

geralmente é feita por porta USB ou serial, mas em outros modelos se faz necessário gravar em um *Chip* EPROM ou memória *flash* e transferir o *ship* para o sistema embarcado.

2.1.8 Eletrônica Básica para Prototipação

Os principais componentes que serão usados para este projeto são resistores, que por sua vez, tem a função de limitar a corrente elétrica, afim de não regular a corrente para que trabalhe na tensão correta, não queimando ou danificando partes do circuito. Não possui polos, podendo ser ligado sem determinação de positivo ou negativo, dissipando sua energia em calor, sua resistência é determinada pelas cores impressas em seu corpo.

> Segundo Filho (2012, pag 54), fundamenta-se a prototipação em três conceitos, resistência, corrente e tensão. A tensão é a energia vinda do armazenamento de pilhas ou baterias que alimentam circuitos fechados entres seus polos de maior e menor potencial, onde o positivo e o negativo indicam a direção que a corrente irá fluir, partindo do positivo para o negativo.

Os capacitores trabalham armazenando energia, afim de corrigir oscilações da corrente e protegendo também o circuito. Podendo ser polarizados ou não, e sua unidade de medida é dada por *Farad*(F).

Já o diodo é um semicondutor que permite a passagem da corrente elétrica apenas em um sentido, evitando que alguma corrente reversa possa atingir e danificar o circuito. Outro exemplo de diodo é o condutor de luz (LED), quando a energia passa pelo diodo, ele acende.

Os transistores vieram afim de substituir as válvulas, trabalhando com chave eletrônica e amplificador de sinal. Possuindo três terminais, denominados base, coletor e emissor, devendo ser consultado cada modelo individualmente para especificações técnicas.

Os circuitos integrados aglomeram vários outros componentes eletrônicos de forma miniaturizada em um único *chip*.

A *protoboard* facilita a prototipação, usada principalmente para restes provisórios para montagem de circuitos eletrônicos, permitindo assim, a reutilização

de componentes. Os pinos dos componentes conectam-se em linha diferentes, e conexões entre componentes acontecem na mesma linha.

2.1.9 SDKs para Sistemas Embarcados

Segundo Arduino (2016), SDKs para Sistemas Embarcados facilita a vida dos iniciantes, os projetos com micro controladores, trouxeram para baixo nível o seu nível de abstração. Trazendo de forma prática até a leitura de sinal digital no Arduino, não precisando interpretar e manipular a corrente elétrica. A portabilidade de código facilita seu desenvolvimento, trazendo para mais perto da nossa linguagem a abstração também de *hardware*. O ponto negativo é o tamanho que o código se desenvolve. Estas bibliotecas são desenvolvidas em C++. O EEPROM faz a leitura e escrita permanente, a GSM faz a conexão a uma rede GPRS ou GSM com GSM, o *LiquidCrystal* controla as telas LCD.

2.2 SISTEMAS DISTRIBUIDOS

Sua definição consiste em computadores autônomos, muitas vezes estando ocultas ao usuário, assim tendo a sensação de estar trabalhando em um único sistema. Outra característica é a interação consistente e uniforme entre usuário e sistema, não dependendo de onde isto ocorra. O sistema distribuído está sempre em funcionamento, mesmo que algumas partes estejam *off-line*, o mesmo vale em caso de avarias e manutenções, as quais não devem afetar o usuário.

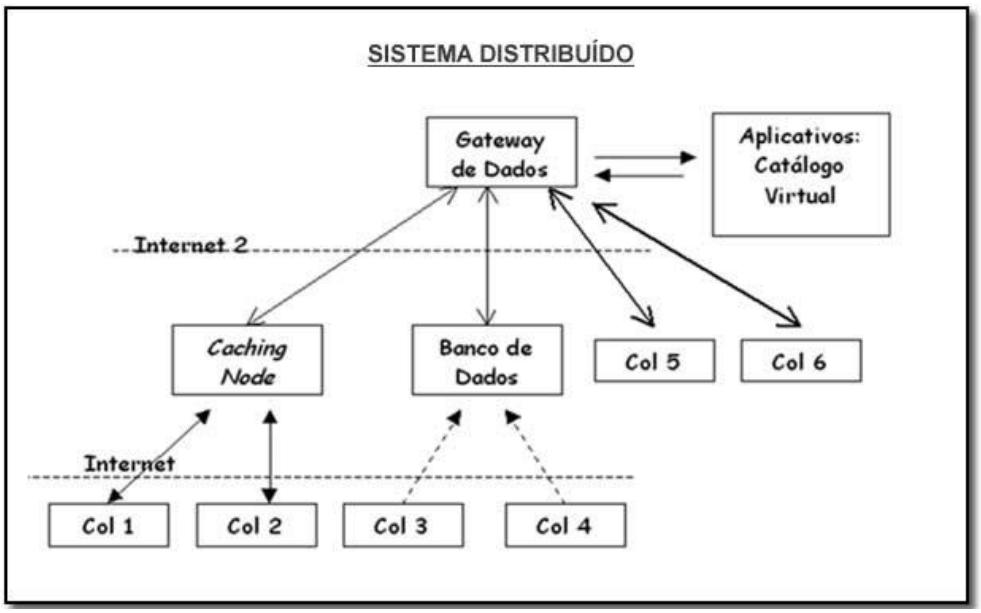
A organização dos sistemas distribuídos se faz através de camada de *software* localizada na camada de mais alto nível, composta por aplicações e usuários, e por outra camada composta por sistemas operacionais e funcionalidades de comunicação.

> De acordo com Tanenbaum (2007, pag 112), pode-se dividir sistemas distribuídos em três elementos, onde os sistemas distribuídos pervasivos são aplicados domesticamente, em sistemas eletrônicos para intervenção médica e também redes de sensores. Já o sistema de informação distribuído trabalha com de propósitos empresariais e processamento de transações. Os sistemas de computação distribuídos constituem-se em computação em grade e clusters.

Detalhando sobre o assunto de redes de sensores, sua aplicação basicamente processa informações, formando uma rede em malha, ou conjunto de nós fixos comunicando-se em rede sem fio. Afim de organizar um banco de dados de

sensoriamento, existem dois exemplos. Primeiramente há o exemplo onde não há cooperação dos sensores, simplesmente eles enviam os dados a um servidor central, basicamente algum site pertencente ao administrador. No segundo exemplo, os dados são repassados aos sensores mais relevantes, permitindo o processamento individual onde cada um irá gerar uma resposta, requerendo de forma sensata as respostas envolvidas por parte do operador.

Figura 5 Exemplo de Sistemas Distribuídos



Mariana Martinez

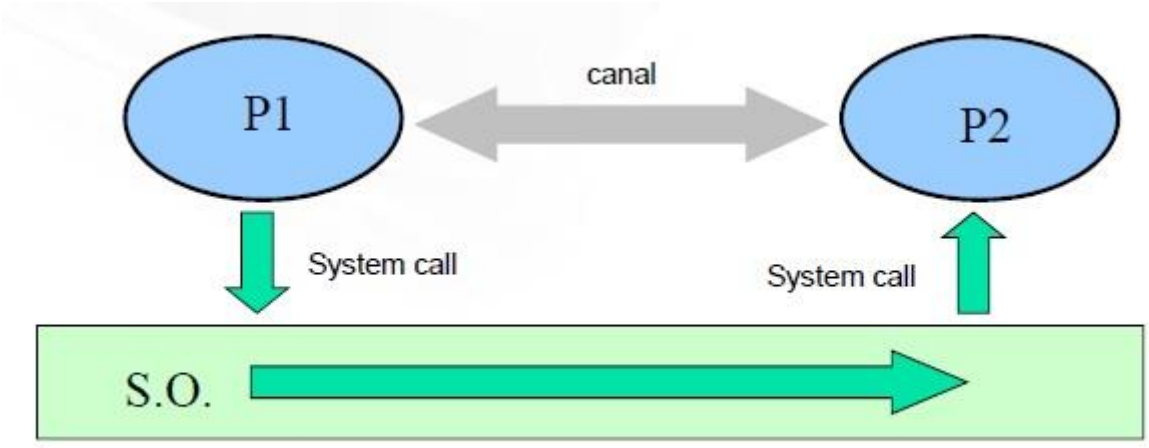
2.2.1 Comunicação Interprocessos IPC

Como o *hardware* protege a memória, nenhum processo interfere no endereçamento da memória do outro. Já os processos interagem na execução de tarefas, pela necessidade da divisão de tarefas assim contribuindo para o aumento da capacidade de processo da rede, e de forma controlada atendendo requisições simultâneas. O mecanismo que faz a troca de informações entre processos e múltiplas *threads* tem por nome de IPC (*Inter-Process Communication*), este processo tem a necessidade de sincronização afim de coordenar seus recursos.

> Segundo Gomes (2016, pag 345), algumas características relevantes na comunicação entre processos fundamentam-se na independência que assegura com que os sistemas operacionais implementem mecanismos entre processos, onde são executadas separadamente em cápsulas autônomas, porém sua execução não afeta outros processos.

O funcionamento do mecanismo IPC acontece quando o Sistema Operacional fornece mecanismos IPC implementando canais de comunicação, sendo eles implícitos ou explícitos entre processos.

Figura 6 Exemplo de Comunicação entre Processos



Fonte: Tanenbaum

As características que se esperam da comunicação interprocessos IPC, são que este modelo funcione de forma igualitária em sistemas distribuídos, seu modelo de sincronização deve ser bem definido. Porém há uma preocupação com a sincronização, onde deve haver a permissão do *sender* para que o mesmo indique quando um dado for transmitido, permitir com que o *receiver* saiba quando estes dados estarão disponíveis e também ambos devem saber quando podem realizar uma nova IPC.

Existem duas abordagens para os mecanismos IPC, a primeira é suportar alguma forma de espaço de endereçamento compartilhado e a segunda é tornar o Sistema Operacional responsável pela comunicação transportando os dados de um processo a outro através do seu núcleo. A vantagem da comunicação na memória compartilhada é a eficiência, pois não exige a cópia de dados para estrutura do núcleo, o ponto negativo são problemas com a sincronização.

Segundo Gomes (2016), a comunicação de um núcleo tem a vantagem de poder ser realizada em várias CPUs através da sincronização implícita, porém é mais complexa e demorada por haver a necessidade de recursos adicionais do núcleo.

Existem vários modelos de comunicação utilizados pelo IPC, por *broadcast* ou difusão, onde há o envio da mesma mensagem para todos os receptores sem saber

que é e quantos são. A comunicação produtor-consumidor é de forma direta e unidirecional fixa, sendo *unicast*.

O modelo *client-server* trabalha de forma bidirecional fixa. O modelo *Peer-to-Peer*, cada nó executa a função tanto de servidor quanto de cliente simultaneamente.

O modelo *Mailbox* a comunicação entre produtor e consumidor é feita de forma indireta através da caixa de correio, onde não há opção de escolha por parte do consumidor sobre o produto que escreveu a mensagem.

O modelo comunicante cria um canal dedicado onde também é criado um servidor dedicado para cada cliente, quando a sessão se termina o canal é desligado.

2.2.2 Estilo Arquitetônicos

Este componente pode ser substituído desde que se considere suas interfaces. Outro mecanismo é o conector, que media a comunicação entre componentes, usando estes componentes pode-se configurar vários estilos arquitetônicos.

> O estilo arquitetônico tem sua formulação em termos de componentes, pela forma como estes estão conectados uns com os outros, como os dados são trocados entre componentes e pela maneira de como são configurados para trabalhar de forma conjunta para um sistema. "Um componente é uma unidade modular com interfaces requeridas e fornecidas bem definidas que é substituível dentro de seu ambiente (OMG, 2004b). (Tanenbaum, 2007, p.20)."

A arquitetura em camadas é simples, onde seus componentes são organizados em camadas, cada camada tem permissão de chamar sua camada subjacente, as requisições fluem através de requisições hierárquicas, porém não pode ser feito o caminho inverso, sendo este modelo periodicamente usado pela comunidade de redes.

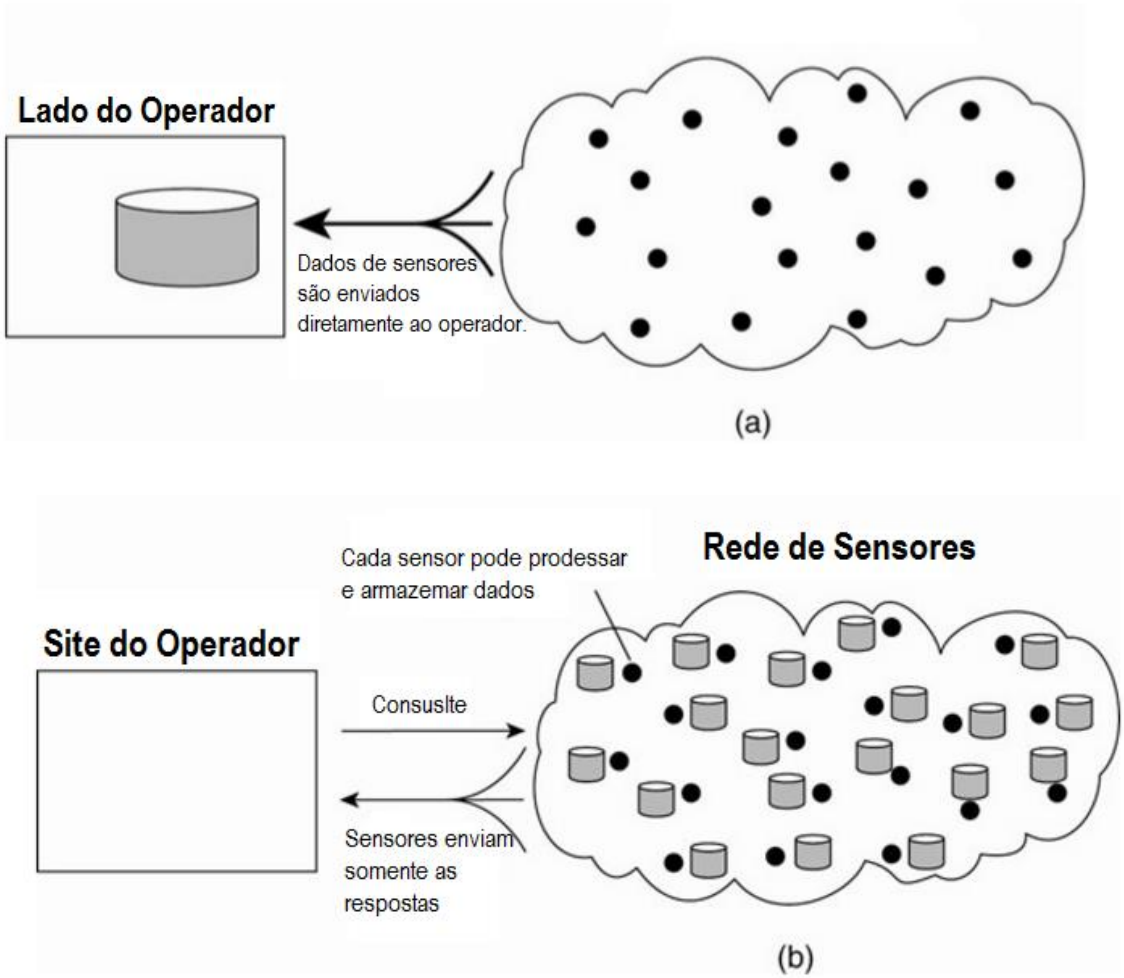
> Segundo Tanenbaum (2007, pag 20), a arquitetura baseada em objetos é de forma mais maleável, cada componente é conectado por meio de uma chamada de procedimento remota. Esta arquitetura juntamente com a arquitetura de chamada é a mais importante para sistemas de grande porte. A arquitetura centrada em dados se baseia em processos que se comunicam através de um repositório comum, sendo este repositório passivo ou ativo. A arquitetura baseada em eventos, como o próprio nome diz, é propagada por meio de eventos, podem também transportar dados. Voltada para sistemas distribuídos, esta arquitetura associa-se com a denominação de sistemas publica/subscrever. "A ideia básica é que processos publiquem eventos após os quais o *middleware* assegura que

> somente os processos que se subscreveram para esses eventos os receberão. (Tanenbaum, 2007, p.21)".

Como resultado da combinação de arquitetura centrada em dados e eventos, obtém-se espaços compartilhados de dados, onde não é necessário ambos estarem ativos para que a comunicação ocorra.

As arquiteturas citadas buscam a transparência de distribuição, visando a melhor alternativa entre tolerância a falha, desempenho, facilidade de programação entre outros requisitos.

Figura 7 Rede de Sensores



Fonte: Tanenbaum

2.2.3 Metas de um sistema distribuído

Como meta do sistema distribuído está em facilitar o acesso a recursos remotos de maneira compartilhada e controlada e eficiente pelos usuários e

aplicações. Porém, como o compartilhamento e a conectividade crescem, se torna cada vez mais importante a segurança.

2.2.3.1 Transparência na distribuição

Para que um sistema seja denominado transparente, ele deve ser capaz de apresentar ao usuário e aplicações como se fosse um único computador. A transparência de acesso oculta as diferenças de representações de dados e o modo com que os usuários acessam estes recursos. Na transparência de localização o usuário não saberá dizer qual a localização física de quaisquer recursos do sistema, uma técnica usada para este recurso é atribuir nomes lógicos ao recursos, não estando codificado no nome a localização do recurso.

Segundo Tanenbaum (2007), com a transparência de migração os recursos são movimentados sem afetar o acesso ou modo de como estes acessos são feitos.

Na transparência de relocação os recursos são recolocados no mesmo momento que estão sendo acessados sem que a aplicação ou usuário percebam quaisquer coisas. A transparência de replicação consiste em colocar uma cópia perto do lugar em que ele é acessado, ocultando o fato da existência de várias cópias deste determinado recurso. Na transparência de concorrência dois ou mais usuários armazenam seus arquivos no mesmo servidor sem que um saiba que o outro está utilizando o mesmo recurso. O sistema de transparência à falha define-se em fazer com que o usuário não perceba quando um determinado recurso parou de funcionar, e posteriormente, o sistema se recuperou da falha.

O grau de transparência nem sempre é uma boa ideia, principalmente quando se trata de um sistema distribuído de longa distância, onde acaba acarretando em uma demora devido à distância, onde levara algumas centenas de milissegundos e também o processamento de comutadores intermediários. Tentar mascarar uma falha transitória de servidores antes da tentativa em outro servidor, reduz a velocidade do sistema, então, permitir com que o usuário cancele as tentativas para fazer outro contato seria uma alternativa, onde seria adotado este alto grau de transparência.

2.2.3.2 Sistema distribuído aberto

A Interoperabilidade é a característica que delimita até quando duas implementações de sistemas ou componentes diferentes devem trabalhar em conjunto, já a portabilidade define até que ponto um sistema pode executar dentro de outro sistema sem modificar nem a sua interface, nem a interface do outro sistema.

> De acordo com Tanenbaum (2007, pag 114), um sistema distribuído aberto é uma meta que visa oferecer serviços regrados padronizadamente onde será descrito a semântica e a sintaxe desses serviços. Estas 'regras' são especificadas através de interfaces, escritas em IDL (*Interface Definition Language*). Esta linguagem quase sempre especifica os nomes das funções disponíveis, s tipos de parâmetros, valores de retorno e possíveis exceções que poderão surgir, e assim por diante.

A facilidade de configurar um sistema distribuído aberto com componentes em bases diferentes é outra meta importante a ser cumprida. A substituição de componentes e a adição de novos não devem afetar em nada seu funcionamento, sendo assim, extensível.

Visando a flexibilidade do sistema distribuído, deve-se organizar em pequenos conjuntos de componentes, facilitando assim a adaptação e a substituição. A necessidade de se alterar um sistema distribuído vem devido ao não fornecimento da política para uma aplicação ou usuário específico.

2.2.4 Problemas e Técnicas de Escalabilidade

Vários problemas são encontrados na escalabilidade, o primeiro é em relação ao tamanho, tratando-se de serviços centralizados, os problemas aparecem quando é preciso suportar mais usuários ou recursos. Neste caso, próprio servidor centralizado pode atuar como um gargalo, devido ao grande volume de usuários e recursos.

> De acordo com Tanenbaum (2007, pag 135), a escalabilidade mede-se por três fatores diferente, o primeiro é em relação ao seu tamanho, facilitando a adição de mais usuários e recurso ao sistema. O segundo mede-se em fatores geográficos, onde os usuários e recursos poderão estar distantes uns dos outros. Por fim, o terceiro é escalável em termos administrativos, facilitando a sua gerencia, mesmo que abrangendo organizações administrativas diferentes.

A escalabilidade geográfica enfrenta problemas nos casos onde sistemas distribuídos que foram projetados para trabalhar em redes locais são baseados em

comunicação síncrona. Esta comunicação funciona somente quando a mensagem que o cliente enviou seja enviada de volta pelo servidor, funciona bem em LANs, mas em redes geográficas o tempo desta troca de comunicação triplica.

Técnicas de escalabilidade vem em busca de solucionar problemas de desempenho nos servidores e rede. Ocultar latências de comunicação é fundamental para solucionar os problemas causados pela escalabilidade geográfica, este processo funciona tentando evitar ao máximo a espera por respostas a requisições remotas de serviços.

Como alternativa para esta espera, adota-se a execução de outro trabalho útil no lado do requisitante, adotando uma aplicação que só use comunicação assíncrona. Quando esta resposta chega, esta aplicação é interrompida e um manipulador especial concluí a requisição executada anteriormente. Contudo, tem aplicações que não podem fazer uso da comunicação assíncrona, por exemplo, as aplicações interativas, onde o usuário não faz nada além do que esperar a resposta de seu comando, a melhor solução é reduzir a comunicação global, transferindo a computação que normalmente é executada no servidor para o processo cliente.

Outra técnica importante é a de distribuição, onde um componente é subdivido em partes menores e espalhado pelo sistema, trabalhando de forma parecida com a do DNS. Outra ideia a ser adotada é a replicação dos componentes de um sistema distribuído, aumentando a disponibilidade e equilibrando a carga entre componentes.

De acordo com Tanenbaum (2007), este processo é uma alternativa para escalabilidade geográfica, pois sempre haverá uma cópia perto do requisitante, reduzindo os problemas com latência de comunicação. Um exemplo deste sistema é o armazenamento em *cache*, porém nesta aplicação, a decisão de replicação é definida no lado do cliente, e não do servidor. Os problemas em usar *cache* e replicação está em suas várias cópias de recurso, se uma delas for modificada, ficará diferente das outras, resultando em inconsistência.

2.2.5 Sistemas de Informação Distribuídos

Então este cliente ao realizar o envio de uma requisição ao servidor, executa determinada operação e posteriormente recebe uma resposta. Trabalhando de forma mais centralizada, a integração acontecia com várias requisições, as quais eram

empacotadas, podendo até mesmo ser de diferentes servidores, todas em uma única requisição maior, enviando em uma única vez através de uma transação distribuída. Afim de executar ou não executar todas as requisições.

> De acordo com Tanenbaum (2007, pag 192), mostrando-se uma classe importante encontrada em organizações cuja profusões das aplicações na rede passaram pelo problema de interoperabilidade, soluções com *middleware* resultando da combinação de infraestrutura onde integrar aplicações com o sistema de informações em nível empresarial se mostrou de fácil gestão. Diferenciando os níveis de integração, a aplicação em rede nada mais é do que um servidor executando determinada aplicação, com banco de dados, disponibilizando este serviço para programas remotos, chamados clientes.

Buscando a integração entre aplicações de banco de dados e componentes de processamento desenvolveu-se a integração de aplicações empresarias (EAI).

Os sistemas de processamento de transações são operações realizadas em banco de dados. Para realizar as programações primitivas especiais devem ser fornecidos pelo sistema distribuídos subjacente ou pela linguagem de programações em execução.

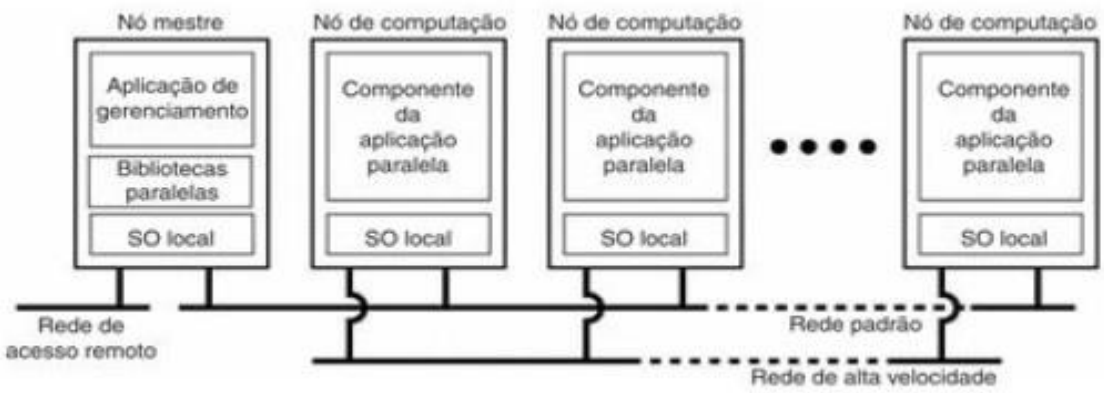
Segundo Tanenbaum (2007), nas integrações de aplicações empresariais as aplicações desenvolvidas para bandos de dados quando passam a se desvincular do banco de dados necessitam de aplicações que devem deixar cada vez mais este serviço independente. Os componentes de aplicação devem ter a capacidade de se comunicar diretamente sem a característica requisição/resposta. O sistema de chamada de procedimento remoto (RPC), envia uma requisição de chamada de procedimento a outro componente da aplicação, empacotando a requisição como mensagem e enviando-o para o chamador. Da mesma forma o resultado é enviado de volta a aplicação.

Com a popularização das chamadas a objetos remotos, desenvolveu-se as invocações de método remoto (RMI), tem o mesmo funcionamento da RPC, o que muda é a utilização de objetos ao contrário da utilização de aplicações. A desvantagem destas duas aplicações é que o chamado e o chamado devem estar em pleno funcionamento na hora da troca de comunicação.

2.2.6 Tipos de Sistemas Distribuídos

Começando pelo sistema de computação distribuído, sendo uma ferramenta de alto desempenho. A computação em *cluster* é um conjunto de computadores ou estações trabalho conectados através de uma rede local de alto rendimento, onde todo o nó executa um único sistema operacional. Já a computação em grade trabalha na forma de federação de computadores, onde cada sistema pode ficar sob o domínio de um administrador diferente, esta diferença sendo tecnologia empregada na rede, de *software* e de *hardware*.

Figura 8 Exemplo de Cluster



Fonte: Tanenbaum

> De acordo com Tanenbaum (2007, pag 201), aprofundando sobre a computação em *cluster*, esta implementação é uma boa alternativa de custo benefício, pois ela é implementada através de vários computadores relativamente simples e uma rede de alta velocidade, sendo de notável diferença financeira em relação á supercomputadores. Atuando com programação paralela onde um único programa é usado em todas as máquinas.

Seu funcionamento se dá por um nó mestre executando o *middleware* necessário para o gerenciamento do *cluster* e execução dos programas. Este *middleware* é constituído por bibliotecas necessárias para a execução dos programas paralelos, as quais são implementadas com facilidades de comunicação por mensagem, não fazendo nenhuma outra função.

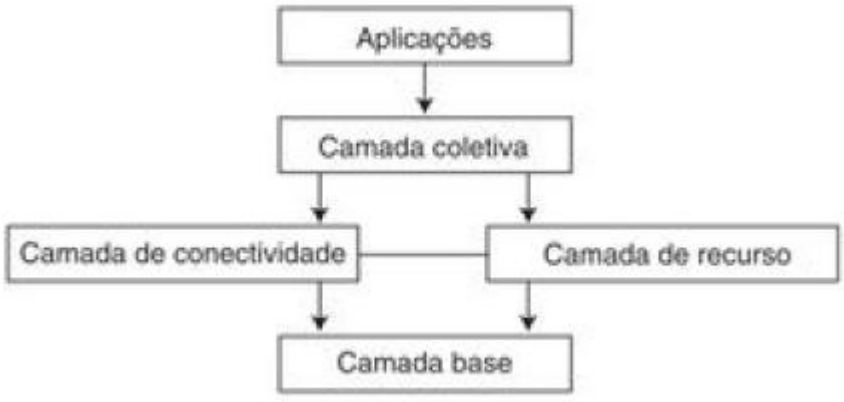
Os sistemas de computação em *grids* são heterogêneos, onde nenhuma premissa é adotada em relação as políticas d segurança, redes, domínios, SO, *hardware* e assim por diante. Outra característica de computação em *grid* é a colaboração de diferentes organizações reunidas afim de beneficiar um grupo de

pessoas ou instituições, esta arquitetura organizada de forma virtual, onde as pessoas da mesma organização virtual têm direitos de acessar os recursos fornecidos para aquela organização.

A finalidade do *software* desta arquitetura é prover acesso em diferentes domínios administrativos onde só os usuários e aplicações pertencentes a esta organização virtual específica tem acesso. Sua arquitetura subdivide-se em quatro camadas. A primeira delas, camada-base, gera interfaces para recursos locais através de um site específico, permitindo compartilhar recursos de uma organização virtual, tendo por finalidade consultar o estado e capacidade dos recursos. A segunda camada, a de conectividade é formada por protocolos de comunicação, segurança de autenticação de usuários e recursos.

Com a terceira camada, a de recursos gerencia-se um único recurso, utilizando funções que são fornecidas com a camada de conectividade fazendo a chamada das interfaces disponibilizadas pela camada-base. A quarta e última camada é a chamada coletiva, responsável pela manipulação do acesso a múltiplos recursos. A camada de aplicação faz uso do ambiente da computação em grade através dos aplicativos que funcionam através desta camada.

Figura 9 Divisão da camadas para sistema de computação em grade



Fonte: Tanenbaum

2.2.7 Automação veicular

A automação veicular busca auxiliar o motorista a tomar decisões em caso de situações críticas de possíveis acidentes, lhe informando, ou até efetuando manobras

relacionadas a dirigibilidade. Outra razão da indústria automotiva automatizar o sistema automotivo é a questão de conforto, regulando de forma automática o ar-condicionado, piloto automático de direção, sistema de estacionamento automático, acendimento automático dos faróis etc.

Para que isto aconteça, esta tecnologia conta com um grande conjunto de sensores, atuadores, transdutores, sistemas de comunicação interprocessos, facilitando a tomada de decisões cada vez mais complexos de forma segura, buscando substituir soluções mecânicas por soluções eletromecânicas.

2.3 OBJETIVOS DA SEGURANÇA

Outro atributo é a integridade, onde só os usuários autorizados podem acessar de forma autorizada determinados ativos. Partindo do exemplo anterior, referente a folha de pagamento, busca-se afirmar a integridade dos salários verificando se os modificados estão dentro dos parâmetros normais da empresa, se esta modificação foi feita pela parte contábil da empresa ou pelo sistema ERP, e comparando se os salários aplicados a cargos e funções podem ser equiparados.

> Segundo PFLEEGER (2006, pag 166), existem três atributos básicos para segurança da informação, a confidencialidade é primeira delas, onde só os usuários que tem autorização podem acessar determinado ativo. Um exemplo claro é quando um funcionário que faz gestão da TI invade e acessa a folha de pagamento de seus colegas, sendo que este acesso está autorizado somente a parte contábil da empresa. Este funcionário acessou o ativo, que é a folha de pagamento, onde a parte, que é o funcionário da TI não tem acesso nenhum, quebrando a segurança, podendo comprometer a confidencialidade deste ativo.

Por último, a disponibilidade traz por objetivo liberar o acesso à quem tem autorização em momentos pré-definidos. Este atributo pode ser afetado por vários fatores, desde ações não intencionais, como quedas de luz, interrupção do serviço de Internet, ou ações maliciosas, como ataques de negação de serviços (DoS).

Outros dois atributos, devido a sua importância, também devem ser citados, a irrefutabilidade e a autenticidade.

Segundo Anderson (2008), irrefutabilidade prova, que determinada transação, realmente aconteceu. Um exemplo claro é quando um atacante realiza um saque bancário na conta de alguma vítima, e esta vítima, posteriormente consegue provar que não foi ele quem realizou este saque. A autenticidade é responsável pela

validação de determinada transação ou mensagem, provando que determinada mensagem não foi alterada durante o seu envio por algum atacante.

2.3.1 Criptografia

A criptografia tem por fim garantir confidencialidade, integridade, autenticidade e irrefutabilidade, mas, ela é só uma pequena parte para garantir a correta segurança que necessitamos.

> Segundo PFLEEGER (2006, pag 219), algoritmos criptográficos são fundamentados na matemática, utilizando da estatística, complexidade computacional, probabilidade etc. A definição de um criptossistema consiste em um sistema que faz a encriptação e decriptação de mensagens. A cifragem ou encriptação codifica uma determinada mensagem garantindo com que seu significado não fique óbvio, utilizando um algoritmo de cifragem.

Seu funcionamento utiliza o texto em claro, que neste caso é a mensagem em seu formato original, onde qualquer pessoa que possua esta mensagem possa a ler, e o texto cifrado, é o resultado obtido através da cifragem deste texto em claro. Esta mensagem pode ser qualquer massa de dados, códigos, aplicações etc.

Já a decifragem ou decriptação faz justamente o processo inverso da cifragem, trazendo a mensagem criptografada de volta a ser um texto em claro. Para este processo também é necessário um algoritmo, mas de decifragem. Para que isto aconteça, se faz necessário o uso de chaves, as quais são utilizadas para criptografar e descriptografar.

Segundo SCHNEIER (2006), a criptoanálise estuda as formas de se obter o texto em claro a partir do texto cifrado sem que se tenha o conhecimento da chave criptográfica. Normalmente para que este processo se concretize, é usado um ataque de força bruta, onde são testados todas as possibilidades de uso da chave através do método de tentativa e erro. Para que um algoritmo seja considerado bom, o tamanho da chave interfere no grau de segurança da criptografia, mas isto não é motivo de garantia.

Outra forma de se conseguir a chave é a dedução, onde, se por muito tempo, for usada a mesma chave para troca de mensagens. As fraquezas ao implementar o algoritmo podem ser exploradas pelo atacante ou criptoanalista. Também inserindo

informações no texto cifrado pode-se deduzir algumas partes criptogradas. Para evitar ataques e resistir a criptoanálise o algoritmo deve ser confuso e difuso.

2.3.2 Criptografia WEP

Para calcular o *checksun* das mensagens, o WEP utiliza o CRC-32, que por sua vez é inserido no pacote, garantindo a integridade dos dados. Quando o pacote chega no seu destino, o *checksum* volta a calcular a mensagem para verificar se não foi alterada. Este meio de criptografia garante a segurança da rede no com a mesmo nível que é utilizada em redes cabeadas. O padrão WEP utiliza dois padrões, o 64 bits e o 128 bits, porém, o padrão 128 bits não são suportados em todos os equipamentos, para seu uso, todos os equipamentos de suportar este padrão, caso contrário os equipamentos que usam o padrão 64 bits não conseguirão usar esta rede.

> Segudno Rufino (2005, pag 114), a criptografia WEP (*Wred Equivalency Privacy*) é referenciada como um método de criptografia usada em redes wireless do patrão 802.11. Operando na camada de enlace de dados atuando criptografando a parte entre cliente e o *Access Point*. O WEP possui sua criptografia baseada no RC4 (*Route Coloniale* 4) da RSA, através do seu vetor de iniciação de 24 bits e uma chave secreta compartilhada que possui entre 40 e 104 bits. A mescla do vetor de iniciação, e esta chave forma uma única chave de 64 ou 128 bits que faz a parte da criptografia dos dados.

Como o WEP usa um vetor de inicialização afim de controlar a frequência com que as senhas são repetidas, o atacante pode analisar estatisticamente os quadros que serão cifrados pela mesma chave, pelo fato do vetor usar somente 24 bits.

2.3.3 Criptografia WPA

Segundo Rufino (2005), com a implementação do padrão 802.11i, a tecnologia de criptografia WEP mostrou-se muito frágil, então foi implementada outra criptografia, a WPA (*Wi-Fi Protected Access*), como forma provisória a esta fraqueza. Assim as chaves fracas foram deixadas de lado, assim como o Protocolo Temporal de Integridade de Chaves e a autenticação EAP.

O WPA possui vários modelos de segurança, podendo se adaptado ao uso que se deseja ser explorado, sendo comumente usados em redes domésticas ou pequenos escritórios, usando a criptografia WPA-PSK que faz o reconhecimento do dispositivo conectado. Outra criptografia é chamada de infraestrutura, que adiciona o

servidor RADIUS (*Remote AutHentications Dial-In User Server*), responsável pela autenticação, ainda podendo incrementar o uso de chaves públicas, onde são utilizados certificados digitas afim de autenticar usuários. Com o sistema de troca de chaves é feito manualmente, não é uma boa ideia usar este método em redes de grande porte, por ser necessário que os participantes estejam acessíveis a maior parte do tempo. O protocolo que faz o gerenciamento das chaves é o TKIP(*Temporal Key Integrity Protocol*), trabalhando em 48 bits. Este protocolo pode ser programado afim de mudar o vetor de iniciação a cada pacote, período ou sessão.

2.3.4 Criptografia WPA2

Em 2004, a Wi-Fi Alliance levou ao público a segunda geração do WPA, o WPA2, com promessas de ser mais seguro que o protocolo anterior, baseado no padrão IEEE 802.11i, atualizando o método criptográfico utilizado, mudando para a versão com AES (*Advanced Encyptation Standart*) e TKIP com 256 bits. Ainda utilizando a autenticação IEEE 302.1x/EAP.

> AES é um bloco cifra, um tipo de cifra de chave simétrica que usa grupos de bits de comprimento fixo - chamados blocos. Uma cifra fundamental simétrica é uma cifra que usa a mesma chave para encriptação e descriptação.(Duarte, 2010)

As vulnerabilidades encontradas no WPA2 são exploradas com ataques de negação de serviço (DoS), pelo fato de não haver proteção para os quadros de gerenciamento e controle. Segundo especialistas da AirTight *Networks* (empresa focada no desenvolvimento de soluções para os segmentos de Redes e Segurança da Informação), há uma falha no protocolo de rede sem fio que foi denominada Buraco 196, fazendo citação ao manual de padrões IEEE, mais especificadamente na página 196. Para que esta falha seja explorada, o atacante deve estar logado na rede ou tenha acesso a chave WPA2, tornando o ataque mais difícil. A chave GTK (*Group Temporal Key*) que é usada para fazer broadcast não consegue detectar quando o endereço MAC é forjado, então quando é gerado pacotes arbitrários de *broadcast*, afim de aguardar a resposta de cliente com informações de suas PTKs (*Pairwise Transient Key*) secretas, podendo ser interceptadas e decodificada pelo atacante. O atacante simula no seu computador um MAC de *Acess Point* passando-se pelo *gateway*. Este ataque usa a técnica ARP *Spoofing,* que é a técnica mais conhecida de executar um ataque por *Man-In-The-Middle.*

2.3.5 Ataque de Replay

Para efetuar o ataque de *replay*, o atacante deve interceptar e salvar os pacotes transmitidos na comunicação entre *hosts* e depois utilizá-los na tentativa de forjada de estabelecer uma nova comunicação. Até mesmo conteúdos cifrados podem sofrer este ataque, mesmo sem conhecer seu conteúdo, o atacante pode ter conhecimento que se trata da abertura de alguma sessão ou até mesmo dados de alguma senha ou *login*. Este ataque se torna mais fácil quando utilizado o protocolo UDP, por não utilizar nenhuma forma de sequência para o envio e recebimento de pacotes.

Figura 10 Attack on Replay

Fonte: Dr. A. Q. Khan

2.4 PROGRAMAÇÃO MOBILE

Cada vez mais os aplicativos para aparelhos móveis vem sendo utilizados para facilitar o nosso dia-a-dia. São utilizados para uma infinidade de tarefas, lazer, redes sociais. Existe uma gama muito diversa de linguagens e ferramentas para desenvolver esses aplicativos.

2.4.1 Android Studio

Para desenvolver a aplicação desse projeto foi utilizado o Android Studio, uma ferramenta muito robusta e completa

Atualmente essa ferramenta pode ser utilizada para programar na linguagem JAVA, que foi a utilizada nesse trabalho, e a linguagem KOTLIN, uma linguagem mais recente. As duas linguagens podem ser utilizadas de forma conjunta para conseguir desfrutar das mais diversas opções de desenvolvimento.

> O Android Studio é o ambiente de desenvolvimento integrado (IDE) oficial para o desenvolvimento de aplicativos para o sistema operacional Android. O software conta com um editor de código e ferramentas de desenvolvedor do IntelliJ (uma ferramenta integrada a IDE que visa dar assistência ao programador durante o desenvolvimento do código). Outro recurso importante do Android Studio é o emulador nativo, que permite testar rapidamente os aplicativos em um dispositivo virtual (ANDROID, 2020a).

3 RESULTADOS OBTIDOS

Neste Capítulo serão relatados todos os resultados obtidos, a partir dos estudos e experimentos realizados com o Arduino, a ESP8266 (placa Wi-Fi) e o aplicativo *mobile*. Serão relatados todos os procedimentos, problemas enfrentados e os sucessos na realização do objetivo.

3.1 PESQUISA DE MERCADO

Antes de dar início a realização desse trabalho, foi realizada uma pesquisa de mercado para verificar se a população em geral aprovaria e utilizaria esse projeto no seu dia-a-dia. Utilizando-se de uma ferramenta virtual, obteve-se respostas de várias pessoas com idades diferentes. As perguntas realizadas e suas respectivas respostas são as que seguem no apêndice desse projeto.

3.2 UTILIZAÇÃO DO ARDUINO

Para a programação do Arduino, foi utilizada a IDE própria do Arduino, disponível na Internet para *download*. Na Figura 11 está disposto o modelo do Arduino utilizado na realização do trabalho, que especificamente é um Arduino UNO R3.

Figura 11 Arduino Uno R3

Fonte: Comphaus

3.2 APLICAÇÃO *MOBILE*

Para poder enviar os comandos para o Arduino, de uma forma segura foi pensada na criação de um aplicativo *Mobile*. Utilizando o *software* Android *Studio*, foi criado um aplicativo bem simples e intuitivo para fazer a interação do usuário com o Arduino. A linguagem de programação utilizada foi java. Criando um Socket Client/Server foi possível enviar e receber as informações necessárias para realizar as operações.

3.2.1 Comunicação entre Aplicação e Arduino

A comunicação entre os dois dispositivos foi realizada através de uma conexão TCP/IP de forma bem simples. Após ter realizada a conexão do smartphone com a rede wi-fi estabelecida pelo ESP-8266 o aplicativo poderá ser usado para enviar e receber os comandos.

Para enviar e receber os dados entre as duas partes, foi criado um protocolo de comunicação, que será detalhado na imagem a seguir:

Figura 12 Protocolo de Comunicação

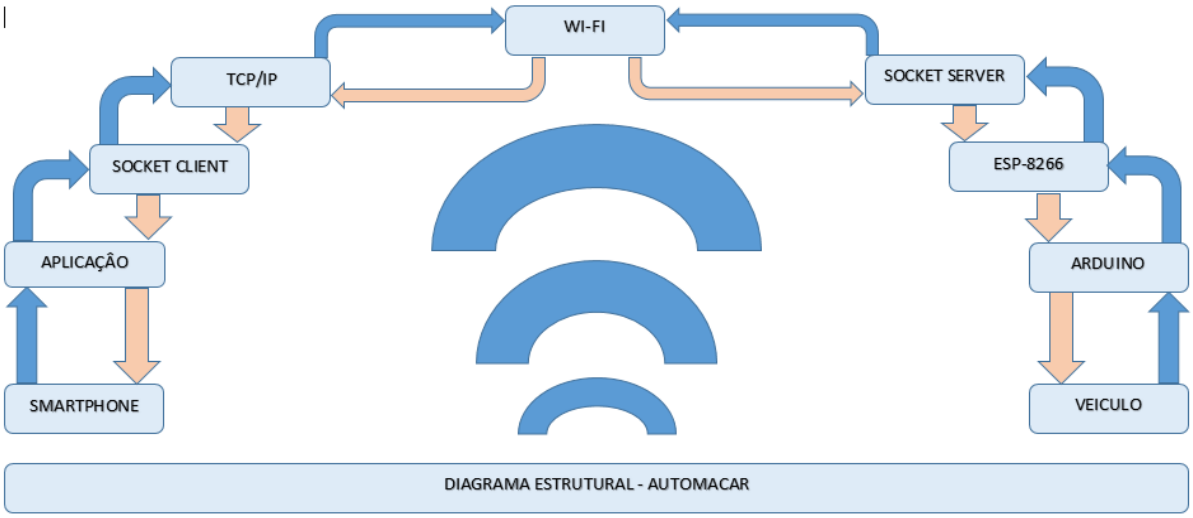
Fonte:Eckhardt

As telas da aplicação estão dispostas no apêndice desse trabalho. O aplicativo encontra-se disponível para download na *Play Store* e de momento só disponível para sistema operacional android.

3.2.2 Diagrama de conexão do projeto

Para a realização desse esquema de conexão, como será abordado também posteriormente, foi elaborado um diagrama, elencando o processo de conexão e comunicação entre as partes envolvidas. A seguir está disposto o diagrama de conexão e comunicação desse projeto:

Figura 13: Diagrama de conexão e comunicação do projeto



Fonte: Eckhardt

3.3 LIGAÇÃO FÍSICA ENTRE AS FERRAMENTAS

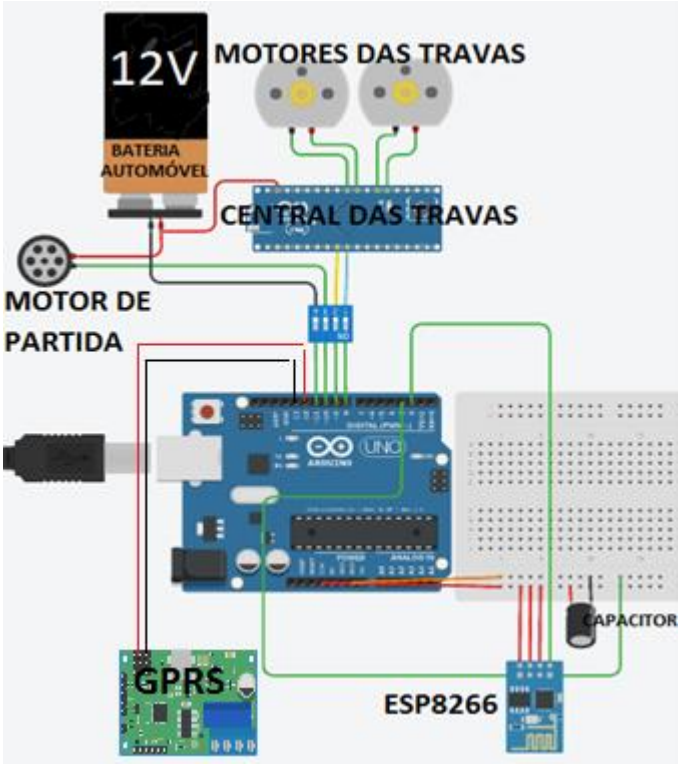
A ligação física dos equipamentos é muito simples, para facilitar ainda mais foi utilizada uma *protoboard*. Como o ESP8266 deve ser alimentado a uma energia de 3,3 V, foi muito fácil a instalação, pois o Arduino já possui uma saída dessa voltagem. Os pinos RX e TX da placa ESP foram ligados nas portas TX e RX respectivamente, para realizar a comunicação serial. Já os pinos 3.3V, RST, e CH_PD foram ligados na porta 3.3V do Arduino, com a utilização da *protoboard.* O pino GND do ESP8266 foi ligado em uma das portas terra ou GND do Arduino.

Os pinos 8, 9, 10 e 11 do Arduino foram utilizados para acionar os relés. Cada saída foi programada para uma função como veremos mais adiante. Para a realização desse trabalho somente foram utilizados três relés, sobrando um para o caso de querer realizar mais alguma função como desligar o carro, buzinar, acender os faróis, entre várias outras.

Na Figura 12 está disposta uma simulação, incluindo todas as placas e as ferramentas utilizadas, desde o Arduino até os motores acionados. Ressaltando que na *protoboard* foi colocado um capacitor nas ligações da placa ESP8266, entre o 3,3V e o GND, afim de evitar quedas ou falhas no *upload* do código ou comandos, pois em vários momentos a placa necessita de energia maior do que a capacidade dos *jumpers* ou o Arduino podem fornecer. Este capacitor tem a função de

armazenar uma energia e usá-la em caso de consumo maior, evitando assim possíveis falhas e problemas.

Figura 14 Simulação do projeto com todas as ferramentas



Fonte: Eckhardt

3.4 PROGRAMAÇÃO DO ARDUINO

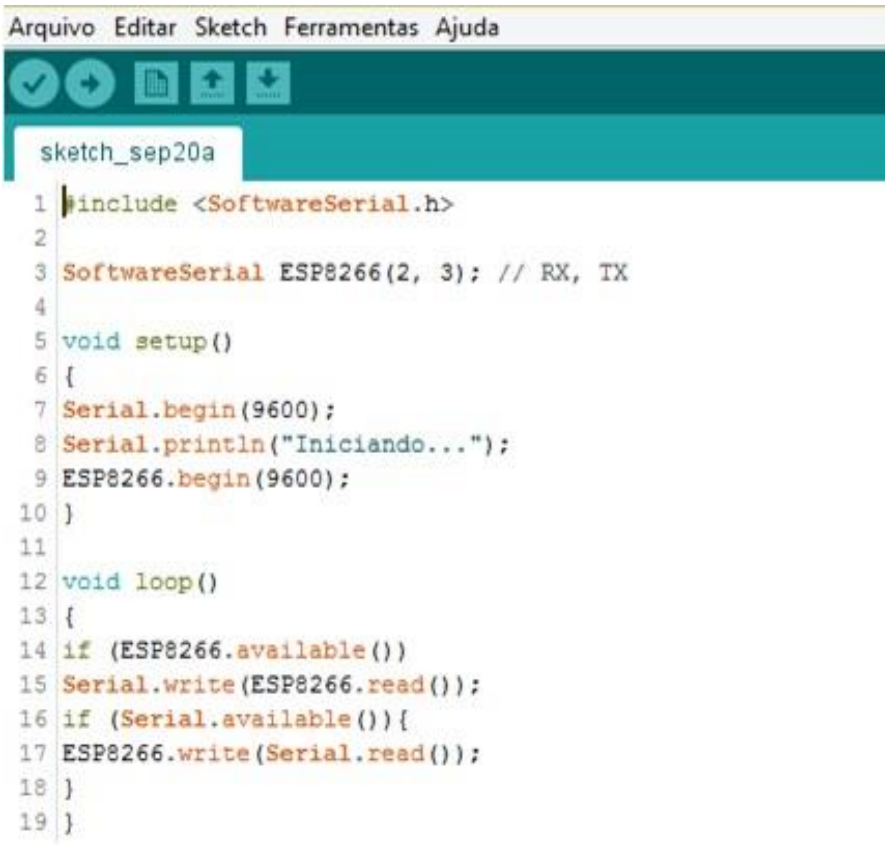
Realizada a ligação física do Arduino, placa ESP8266, *protoboard* e o módulo dos relés, partiremos para programação dos mesmos. Primeiramente deve-se realizar o *upload* do código para o Arduino afim de programar a ESP8266 via serial. Iniciando o código deve-se importar a biblioteca *softwareSerial,* que é responsável pela comunicação via serial entre o Arduino e a ESP8266.

Após isso foram definidas as portas do Arduino que irão se comunicar com o ESP8266. Ressaltando que as ligações RX e TX são invertidas do ESP8266 para o Arduino, ligando TX do Arduino no RX do ESP e vice-versa. No código da Figura 12 os pinos do Arduino utilizados foram 2 e 3 para TX e RX respectivamente.

No *Void Setup* foi colocado toda a parte do código que deverá ser executado logo quando o Arduino será ligado. Na figura 13, foi definida a velocidade de comunicação entre o Arduino e a ESP8266 em 9600. No Void Loop é colocado toda a parte do código que será executado em forma de loop, ou seja, ele é executado

infinitamente. Nesse código o Arduino vai ler todos os comandos digitados no monitor serial da IDE e enviará para o ESP, que por sua vez retornará os resultados do comando, que podem variar conforme o comando.

Figura 15 Código a ser carregado para o Arduino para a realização da comunicação com o ESP8266



Fonte: Eckhardt

3.4.1 Programação do ESP8266

Para programar o ESP são enviados comandos, os chamados comandos AT. Com esses comandos podem ser realizadas inúmeras configurações dependendo da necessidade. Neste trabalho só serão mostrados os comandos que foram utilizados para a realização deste projeto. Para verificar se o seu ESP está respondendo, basta enviar um AT e este deve retornar um OK. Feito isso ele estará pronto para receber os comandos de configuração. Na tabela 3 estarão listados todos os comandos utilizados para a configuração do ESP8266, com suas respectivas finalidades.

Tabela 3 – Comandos para configurar o ESP8266

AT+RST	Reinicia a placa ESP8266
AT+GMR	Informa a versão do *firmware*
AT+RESTORE	Reseta a placa para as configurações originais
AT+CWMODE_DEF=3	Torna o ESP um ponto de acesso em modo default
AT+UART_DEF=9600,8,1,0,0	Define a velocidade da comunicação serial
AT+CWDHCP_DEF=1,1	Ativa a função DHCP
AT+CWSAP_DEF="SSID","senha",5,3	Define criptografia wpa2, SSID e senha
AT+CIPMUX=1	Ativa as conexões simultâneas, neste caso somente uma
AT+CIPSERVER=1,80	Configura como servidor TCP, porta 80
AT+CWDHCPS_DEF=1,3,”192.168.99.99”, ”192.168.99.100”	Define os range de IP utilizados, primeiro e último da rede

3.4.2 Código utilizado no Arduino

Para melhor entender o que o código vai fazer, será explicado em partes. Primeiramente deve se incluir a biblioteca serial para realizar a comunicação via porta serial. Logo após devem ser declaradas as variáveis e as respectivas portas para a ligação nos relés. Depois o número das portas seriais TX e RX do Arduino para realizar a comunicação serial com a ESP8266.

Figura 15 Declaração da biblioteca e variáveis

projeto_final

```
#include <SoftwareSerial.h>
int RELE1 = 11;
int RELE2 = 10;
int RELE3 = 9;
int RELE4 = 8;
SoftwareSerial ESP8266(2, 3); // RX, TX

```

Fonte: Eckhardt

Após declaradas as variáveis dos relés e as portas seriais, foram necessárias mais duas variáveis, que serão utilizadas para a captura do comando enviado do aplicativo *mobile*. Onde o b é referente a cada *byte* recebido, e *buffer* é a quantidade de *bytes* que são recebidos a cada comando que for enviado. Ou seja, no caso do nosso código sempre chegarão doze *bytes*.

Figura 16 Definição de variáveis

```

int b = 0;
char buffer[12]; // IPD,0,2:2\r\n\0

```

Fonte: Eckhardt

Dentro do *Void Setup* foram declarados todos os pinos utilizados para acionar os relés em modo de saída, ou seja, OUTPUT. Conforme mostrado na imagem a seguir.

Figura 17 Definição de saída de sinal

```
void setup()
{
  pinMode( RELE1, OUTPUT );
  pinMode( RELE2, OUTPUT );
  pinMode( RELE3, OUTPUT );
  pinMode( RELE4, OUTPUT );
```

Fonte: Eckhardt

Como já vimos, no Setup devemos inserir todo o código que precisa ser executado quando o Arduino for ligado. Por isso faz se necessário definir a velocidade de comunicação do Arduino com a ESP8266, configurando os dois na mesma velocidade, no nosso caso 9600. Se for definida uma velocidade diferente dessa, por

exemplo a 115200 que vem configurada de fábrica do ESP8266, a comunicação não tem um funcionamento satisfatório.

Figura 18 Configuração do ESP8266

```
21
22  Serial.begin(9600);
23  Serial.println("A iniciar");
24  ESP8266.begin(9600);
25
26  delay(3000);
27  ESP8266.write("AT+CIPMUX=1\r\n");
28  delay(2000);
29  Serial.write(ESP8266.read());
30  ESP8266.write("AT+CIPSERVER=1,80\r\n");
31  delay(2000);
32  Serial.write(ESP8266.read());
```

Fonte: Eckhardt

No *Void Loop* então passamos a colocar o código que estará esperando pelo comando enviado. Uma característica do ESP8266, é que toda vez que ele recebe um comando o primeiro *byte* recebido sempre é um '+'. Com isso precisamos fazer uma *string* que esteja pronto para captar esses *bytes*. Com um IF comparamos se o primeiro byte for um '+', armazena os doze primeiros *bytes* no *buffer*.

Figura 19 Captura dos comandos

```
40 void loop() {
41
42  if (ESP8266.available()) {
43    b = ESP8266.read();
44
45    if (b == '+') {
46      Serial.write("debug: recebido +\r\n");
47      delay(250);
48      ESP8266.readBytes(buffer, 11);
```

Fonte: Eckhardt

Com as observações feitas pode ser notado que todo o comando recebido do aplicativo mobile, sempre o oitavo byte era o enviado. Então com um switch case foi pego o oitavo byte do buffer, e com os case comparando se era 1, 2, 3 ou 4. Se por exemplo, o oitavo byte for 1 ele entraria no primeiro case acionaria o relé, com o *delay* ele esperaria um segundo, que nessa programação é definida em milissegundos, por isso o 1000 e depois voltando a desligar. No código usado para o nosso trabalho foi necessário fazer uma adaptação, pois toda vez que o Arduino era ligado os relés já começavam ligados. Então com testes feitos com os comandos LOW, eles

continuavam ligados. Por isso foi necessária a adição das linhas a seguir no código, invertendo assim todos os outros comandos LOW e HIGH.

Figura 2012 Adaptação das saídas de sinais

```
17  digitalWrite( RELE1, HIGH );
18  digitalWrite( RELE2, HIGH );
19  digitalWrite( RELE3, HIGH );
20  digitalWrite( RELE4, HIGH );
```

Fonte: Eckhardt

Figura 21 *Switch case* - Função dos comandos

```
49
50      switch (buffer[8]) {
51        case '1':
52          digitalWrite( RELE1, LOW );
53          delay(1000);
54          digitalWrite( RELE1, HIGH);
55          break;
56        case '2':
57          digitalWrite( RELE2, LOW  );
58          delay(1000);
59          digitalWrite( RELE2, HIGH );
60          break;
```

Fonte: Eckhardt

No final do código foi adicionado um *default*, que pegaria todo o conteúdo que não entrou nos case, jogando os para o *buffer*. Encerrando, o *else* imprime todos os bytes na serial.

Figura 22 Regras default do código

```
71        default:
72          Serial.write("debug: ");
73          Serial.write(buffer);
74      }
75    } else {
76      Serial.write(b);
77    }
78
```

Fonte: Eckhardt

3.5 SEGURANÇA EMPREGADA NO SISTEMA

Levando em consideração a segurança do sistema, foi adotado o modo de criptografia WPA2. Para reforçar a segurança, foi utilizado uma senha forte, contendo letras maiúsculas, minúsculas, números e caracteres especiais. Dificultando assim

possíveis ataques de força bruta. Além da criptografia WPA2, outra forma de reforçar a segurança é limitar o acesso aos aparelhos conhecidos através do MAC ADDRESS

3.6 CONSIDERAÇÕES FINAIS

Durante a elaboração desse projeto, inúmeras vezes precisei mudar a forma de realizar o processo das conexões e elaboração da aplicação, por encontrar uma maneira mais prática e funcional de realiza-la.

Várias dúvidas e dificuldades no processo, que consegui resolver, assistindo vídeos, realizando cursos online, lendo artigos, entre outros. Apesar de várias dificuldades, tive um grande aprendizado, além do que aprendi em sala de aula.

Observando ainda que o projeto ficou de fácil adaptação para um possível upgrade de funções, ou ser utilizado para desempenhar outras funções de automação.

REFERÊNCIAS

4LINUX, 2014. **O que é Monitoramento**. Disponível em: <https://www.4linux.com.br/o-que-e-monitoramento>. Acesso em: 29/03/2020.

ARDUINO .2016. **What is Arduino?** Disponível em:< https://www.arduino.cc/en /Guide/Introduction>. Acesso em: 25/04/2020.

ARDUINO,2016.**Libraries**.Disponívelem:<https://www.arduino.cc/en/Reference/ Libraries>.Acesso em:04/03/2020.

Barros, Wagner Rocha. 2012. **Sistema de Automação Veicular com Arduino e Android.** Disponível em:<http://docplayer.com.br/1645426-Sistema-de-automacao-veicular-com-arduino-e-android.html>. Acesso em: 01/04/2020.

Blem, Emily. Menon, Jaikrishnan. Sankarialingam, Karthikeyan. 2016. **Power Struggles: Revisiting the RISC vs CISC Debate on Contemporary ARM and x86 Architectures.** Disponível em: < http://www.slideserve.com/kaveri/risc>. Acesso em: 16/03/2020.

FERGUSON, Niels. Schneier, Bruce. 2003. **Practical Cryptography.** 1ª ed. Wilei. Indianapolis, Indiana: ISBN: 978-0-471-22357-3.

FILHO, Onildo Henrique B. 2012. **Componentes eletrônicos e unidades de medida, conceitos básicos.** Disponível em:<http://www.hardware.com.br/tutoriais/ componentes-eletronicos-unidades-medida-conceitos-basicos/componentes-eletronicos.html>. Acesso em 28/04/2020.

GOMES, Roberta Lima. 2016. **Inter-process Communication (IPC).** Disponível em: <http://www.inf.ufes.br/~rgomes/so_fichiers/aula14.pdf>. Acesso em: 21/04/2020.

GREAVES, David J. 2011. **System on Chip Design and Modelling.** Disponível em: <http://www.cl.cam.ac.uk/teaching/1011/SysOnChip/socdam-notes1011.pdf>. Acesso em 22/03/2020.

Inovação Tecnológica. 2016. **Quebrando sistema de segurança das redes Wi-Fi.** Disponível em:< http://www.inovacaotecnologica.com.br/noticias/noticia.php?artigo= quebrado-protocolo-seguranca-wpa2-redes-sem-fio>. Acesso em: 22/04/2020

Khan, Dr. A. Q. 2013. **Network Security.** Disponível em: < http://www.slideshare.net/ lineking/cns-13flec01-overview>. Acesso em 16/03/2020.

LOVATO, Adalberto. 2013. **Metodologia da Pesquisa**, SETREM. Três de Maio. ISBN: 978-85-99020-05-0.

MARIMOTO, Carlos E. 2007. **Processadores RISC X Processadores SISC.** Disponível em: < http://www.hardware.com.br/artigos/risc-cisc/>. Acesso em: 25/04/2020.

MORIMOTO, Carlos E. 2007. **ARM.** Disponível em: < http://www.hardware.com.br/ termos/arm>. Acesso em 24/02/2020.

OETIKER, Tobias. 2013. **What is MRTG?.** Disponível em: <http://oss.oetiker.ch/mrtg /doc/mrtg.en.html>. Acesso em 30/04/2020.

OPEN SYSTEMS MEDIA, 2013. **System on Chip.** Disponível em: < http://fpgablog. com/posts/tag/system-on-chip/>. Acesso em 16/04/2020.

PFLEEGER, Charles. 2011. **Security in Computing.** 4ª ed. Pearson. Massaschusetts, United States: ISBN 0-13-239077-9.

PRADO, Sergio. 2013. **Sistema Embarcado - O que é? Qual a sua importância?.** Disponível em: <http://www.embarcados.com.br/sistema-embarcado/>. Acesso em: 23/04/2020.

RUFINO, Nelson Murilo de Oliveira. 2011. **Segurança em Redes sem Fio.** 3ª ed. Novatec. São Paulo: ISBN 978-85-7522-243-0.

SOUSA, Linderberg Barros. 1999**. Redes de Computadores, Dados, Voz e Imagem**.8ª ed. Editora Érica. São Paulo: ISBN:85-7194-590-x.

SOUZA, Fabio. 2014. **Criando suas Próprias Bibliotecas Para Arduino**. Disponível em: < http://www.embarcados.com.br/criando-suas-proprias-bibliotecas -par- arduino />.Acesso em: 04/03/2020.

SOUZA, Fabio. 2014. **Criando suas Próprias Bibliotecas Para Arduino.** Disponível em: < http://www.embarcados.com.br/criando-suas-proprias-bibliotecas -par-arduino/>.Acesso em: 04/04/2020.

TANENBAUM, Andrew s. 2001. **Redes de computadores**. 4ª ed. Editora Campus. Rio de Janeiro: ISBN:.85-352-1185.3.

TANENBAUM, S. Andrew, Wetherall, David. 2011. **Redes de Computadores**, 5ª ed.Pearson. São Paulo: ISBN 978-85-7605-924-0.

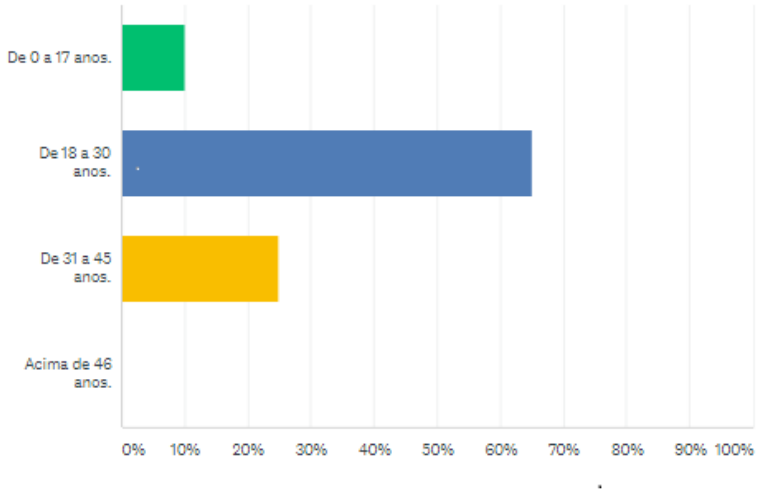
TUDE, Eduardo. 2016. **Rádio Digital: O que é?** Disponível em: < http://www.teleco. com.br/tutoriais/tutorialrdig/pagina_1.asp>. Acesso em: 04/04/2020.

APENDICES

Abaixo estarei dispondo o resultado da pesquisa de mercado que foi realizada no período de 01/03/2020 a 15/03/2020.

Qual a sua faixa etária?

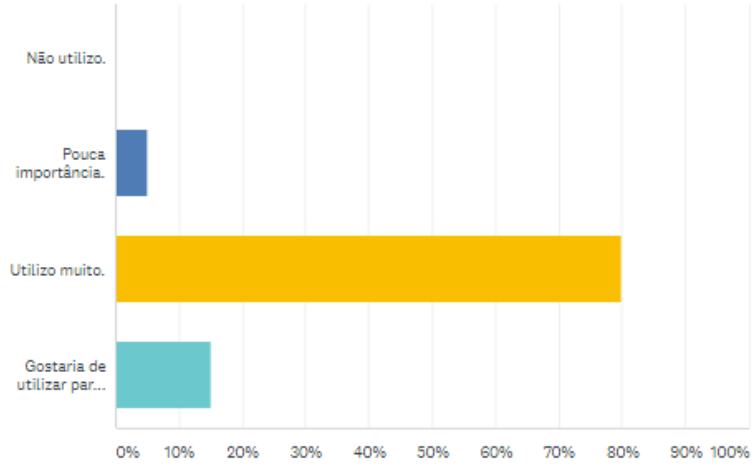
Answered: 20 Skipped: 0



OPÇÕES DE RESPOSTA	RESPOSTAS	
De 0 a 17 anos.	10,00%	2
De 18 a 30 anos.	65,00%	13
De 31 a 45 anos.	25,00%	5
Acima de 46 anos.	0,00%	0
TOTAL		20

Qual a importância de um Smartphone no seu cotidiano?

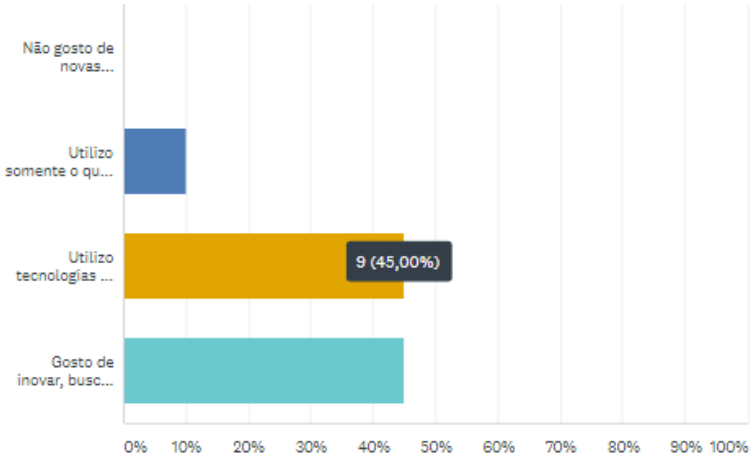
Answered: 20 Skipped: 0



OPÇÕES DE RESPOSTA	RESPOSTAS	
Não utilizo.	0,00%	0
Pouca importância.	5,00%	1
Utilizo muito.	80,00%	16
Gostaria de utilizar para mais tarefas.	15,00%	3
TOTAL		20

Como se considera tecnologicamente?

Answered: 20 Skipped: 0

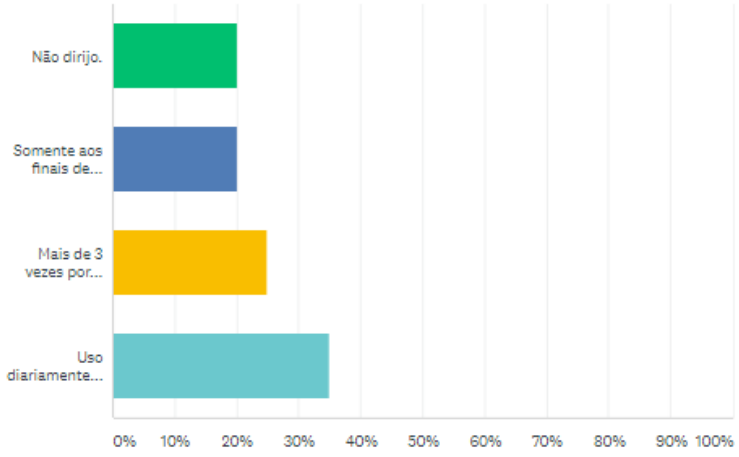


OPÇÕES DE RESPOSTA	RESPOSTAS	
Não gosto de novas tecnologias.	0,00%	0
Utilizo somente o que preciso.	10,00%	2
Utilizo tecnologias que me atraem.	45,00%	9
Gosto de inovar, buscar tecnologias novas.	45,00%	9
TOTAL		20

Comentários (0)

Qual a frequência que dirige um carro?

Answered: 20 Skipped: 0

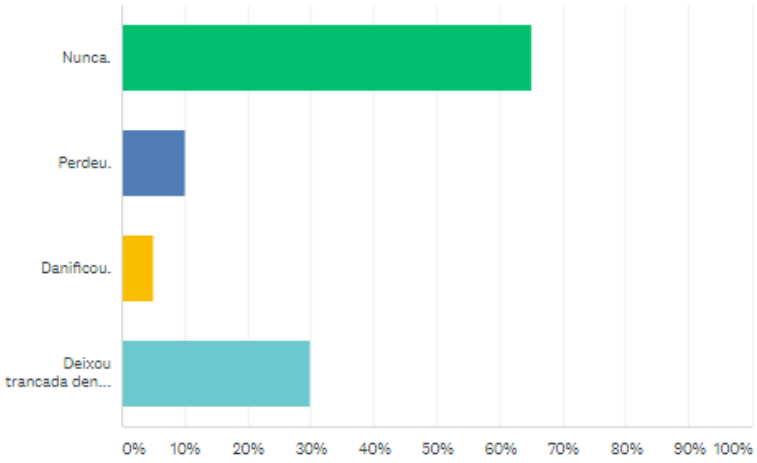


OPÇÕES DE RESPOSTA	RESPOSTAS	
Não dirijo.	20,00%	4
Somente aos finais de semana.	20,00%	4
Mais de 3 vezes por semana.	25,00%	5
Uso diariamente para trabalho e lazer.	35,00%	7
TOTAL		20

Comentários (0)

Alguma vez já perdeu, danificou ou deixou a chave dentro do veículo?

Answered: 20 Skipped: 0



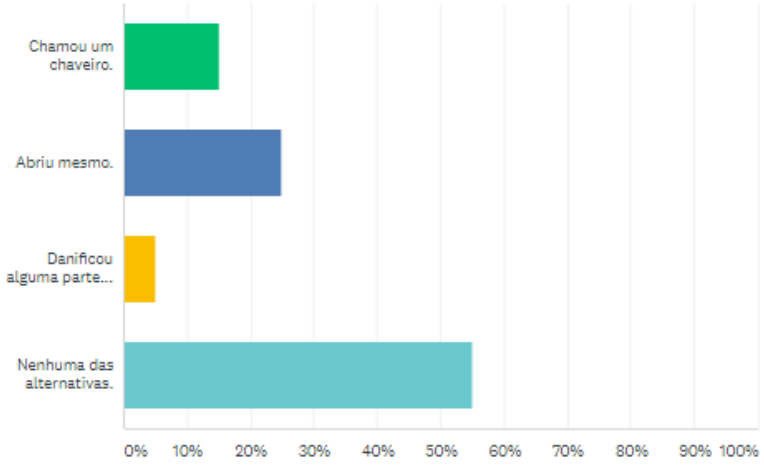
OPÇÕES DE RESPOSTA	RESPOSTAS	
Nunca.	65,00%	13
Perdeu.	10,00%	2
Danificou.	5,00%	1
Deixou trancada dentro do carro.	30,00%	6
Total de respondentes: 20		

Comentários (0)

Quando teve problema com a chave, o que fez?

Quando teve problema com a chave, o que fez?

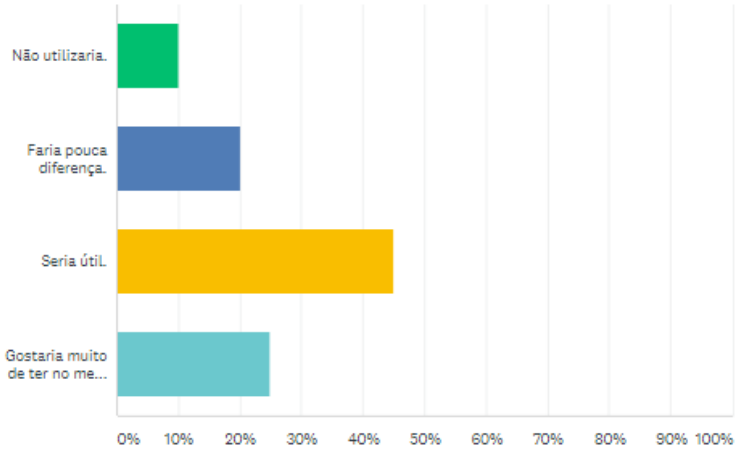
Answered: 20 Skipped: 0



OPÇÕES DE RESPOSTA	RESPOSTAS	
Chamou um chaveiro.	15,00%	3
Abriu mesmo.	25,00%	5
Danificou alguma parte do carro para poder usar/acessar ele.	5,00%	1
Nenhuma das alternativas.	55,00%	11
TOTAL		20

Qual sua opinião em utilizar o seu próprio Smartphone para acessar seu carro sem tirar do próprio bolso?

Answered: 20 Skipped: 0

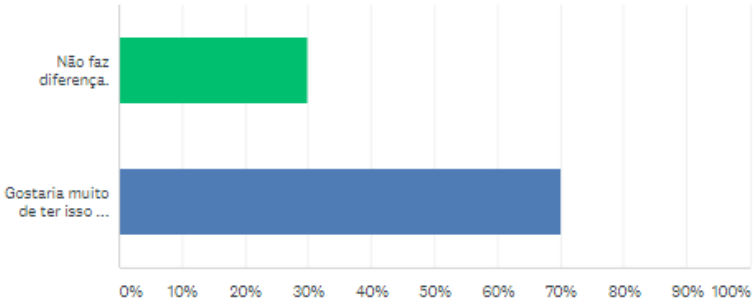


OPÇÕES DE RESPOSTA	RESPOSTAS	
Não utilizaria.	10,00%	2
Faria pouca diferença.	20,00%	4
Seria útil.	45,00%	9
Gostaria muito de ter no meu carro.	25,00%	5
TOTAL		20

Comentários (0)

Acha interessante conseguir monitorar a localização do seu carro em tempo real pelo seu Smartphone?

Answered: 20 Skipped: 0

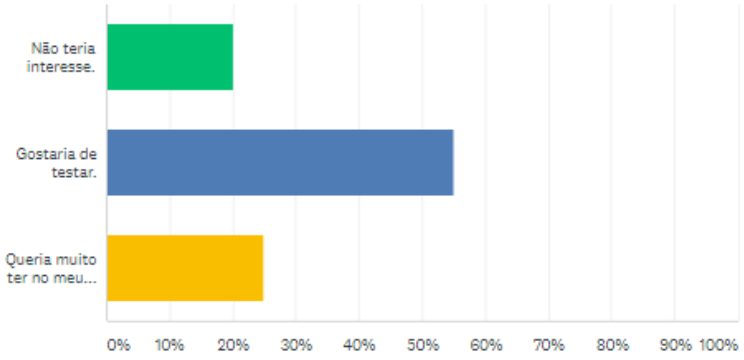


OPÇÕES DE RESPOSTA	RESPOSTAS	
Não faz diferença.	30,00%	6
Gostaria muito de ter isso no meu carro.	70,00%	14
TOTAL		20

Comentários (0)

Teria interesse em automatizar seu carro e ter essas e outras funções disponíveis, independente do modelo, por um valor acessível?

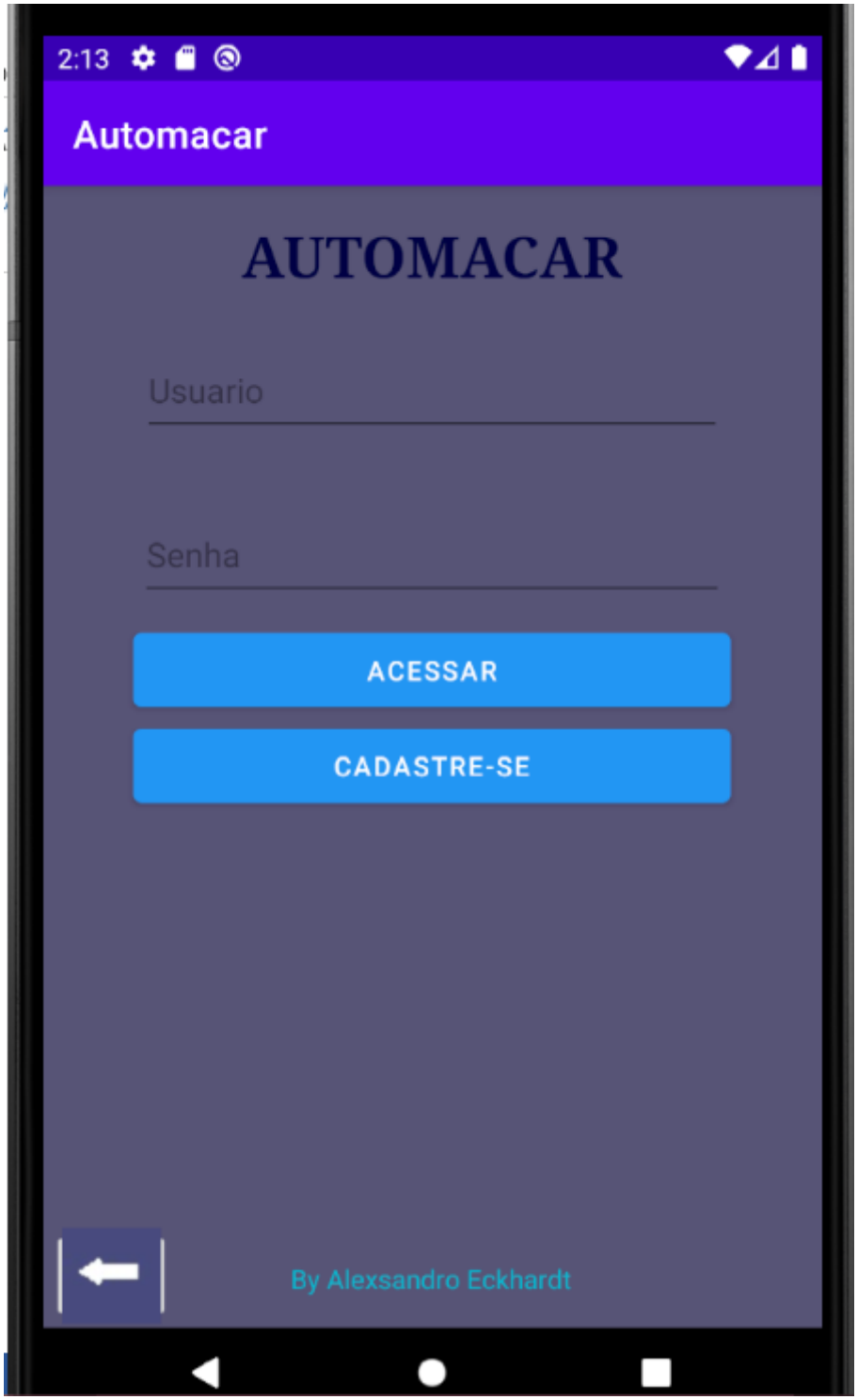
Answered: 20 Skipped: 0

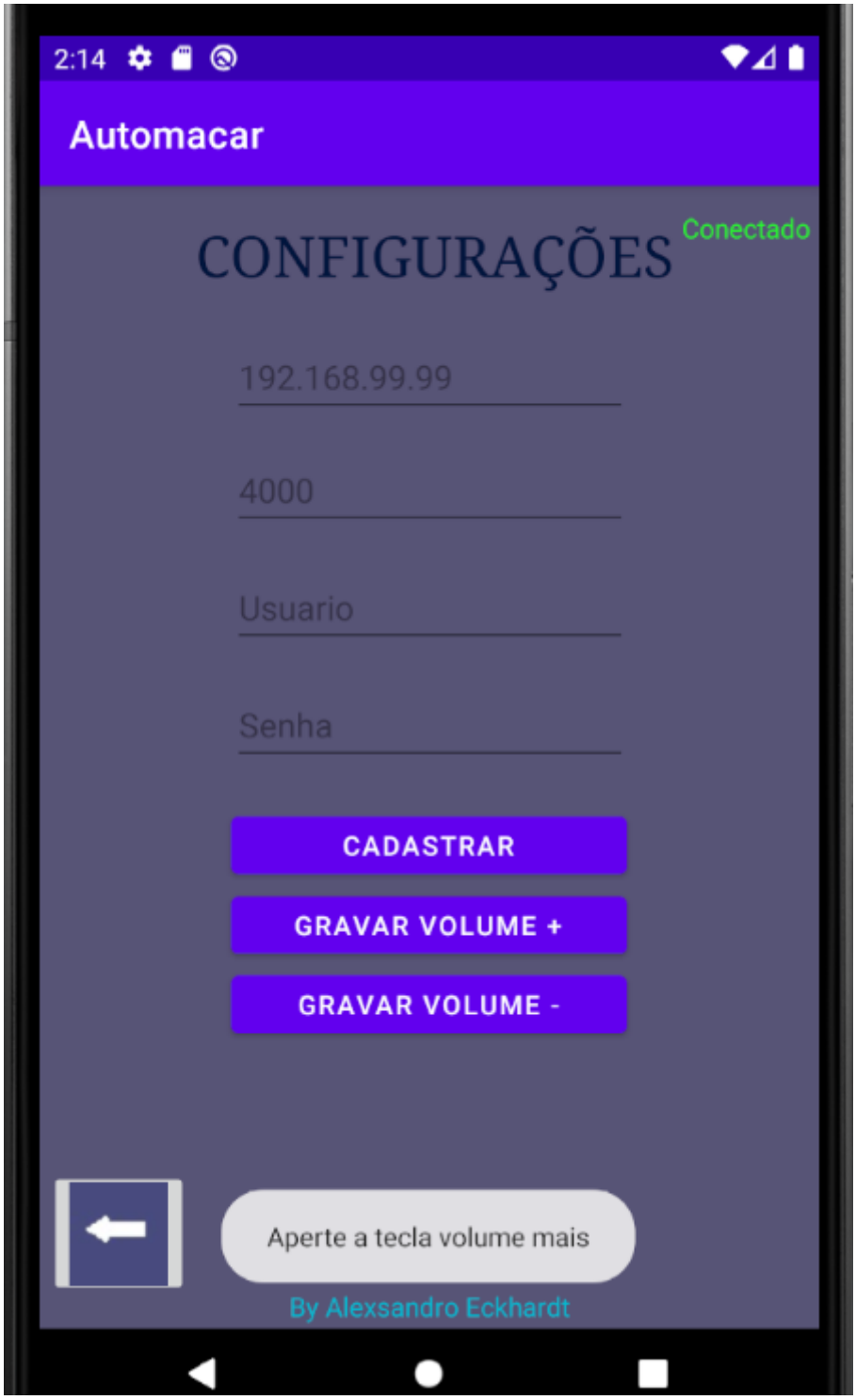


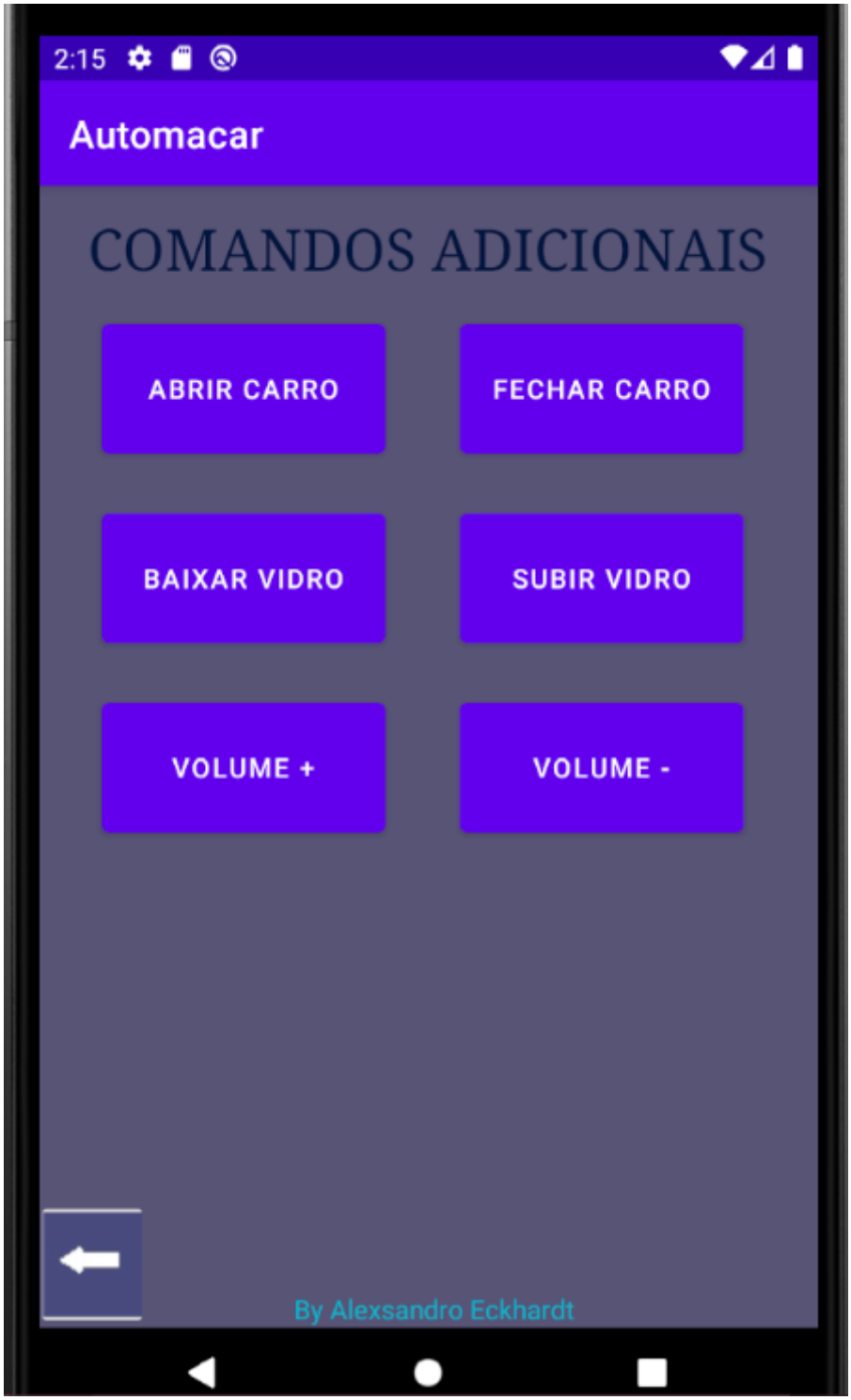
OPÇÕES DE RESPOSTA	RESPOSTAS	
Não teria interesse.	20,00%	4
Gostaria de testar.	55,00%	11
Queria muito ter no meu carro.	25,00%	5
TOTAL		20

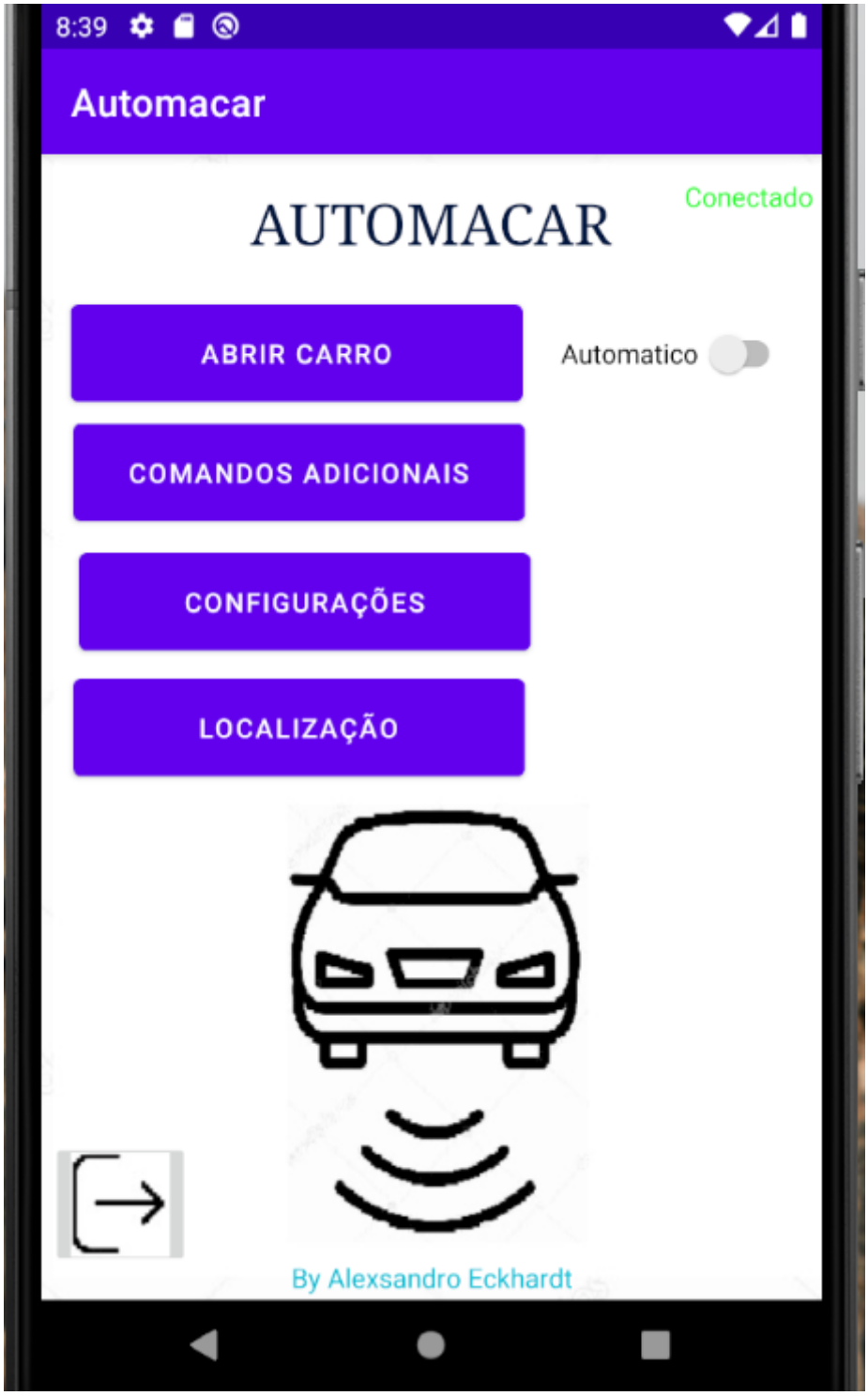
Comentários (0)

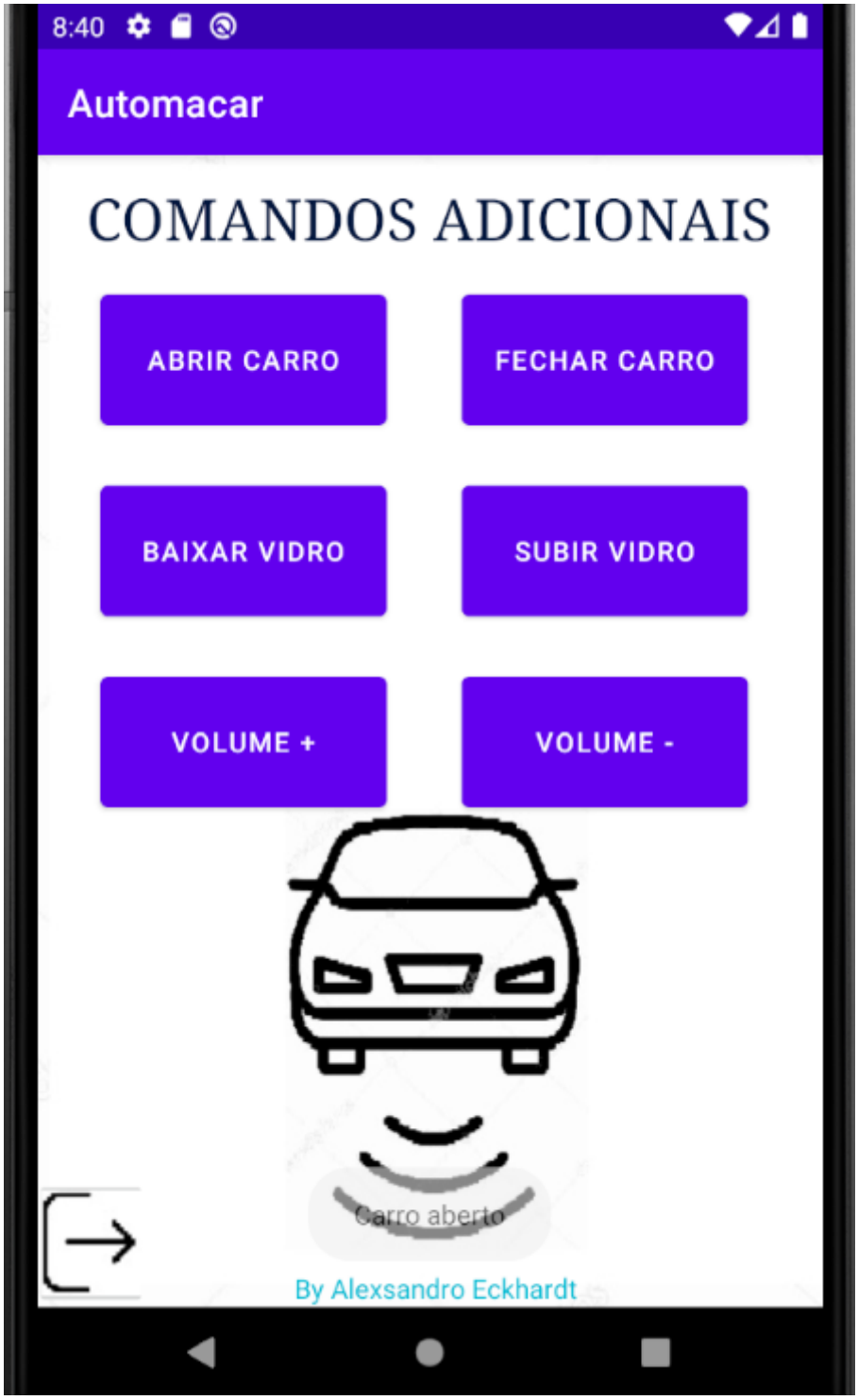
Ainda no apêndice estão dispostos todas as telas do aplicativo mobile.


2:13
Automacar
AUTOMACAR
Usuario
Senha
ACESSAR
CADASTRE-SE
By Alexsandro Eckhardt


2:14
Automacar
CONFIGURAÇÕES
Conectado
192.168.99.99
4000
Usuario
Senha
CADASTRAR
GRAVAR VOLUME +
GRAVAR VOLUME -
Aperte a tecla volume mais
By Alexsandro Eckhardt


2:15
Automacar
COMANDOS ADICIONAIS
ABRIR CARRO
FECHAR CARRO
BAIXAR VIDRO
SUBIR VIDRO
VOLUME +
VOLUME -
By Alexsandro Eckhardt


8:39
Automacar
AUTOMACAR
Conectado
ABRIR CARRO
Automatico
COMANDOS ADICIONAIS
CONFIGURAÇÕES
LOCALIZAÇÃO
By Alexsandro Eckhardt


8:40
Automacar
COMANDOS ADICIONAIS
ABRIR CARRO
FECHAR CARRO
BAIXAR VIDRO
SUBIR VIDRO
VOLUME +
VOLUME -
Carro aberto
By Alexsandro Eckhardt


8:40
Buscando localização...
Google