

**Controle do Documento**

**Histórico de revisões**

|  |  |  |  |
| --- | --- | --- | --- |
| **Data** | **Autor** | **Versão** | **Resumo da atividade** |
| 17/11 | Luiz Granville | 3.1 | Atualização da seção 1.1 e 1.2 |
| 21/11 | Luísa Leite | 3.2 | Atualização da seção 2 |
|  |  |  |  |





# Índice

1. [Introdução 3](#_TOC_250004)
   1. Solução 3
   2. Arquitetura da Solução 3
2. Componentes e Recursos 4
   1. [Componentes de hardware 4](#_TOC_250003)
   2. Componentes externos 4
   3. [Requisitos de conectividade 4](#_TOC_250002)
3. Guia de Montagem 5
4. Guia de Instalação 6
5. Guia de Conﬁguração 7
6. [Guia de Operação 8](#_TOC_250001)
7. [Troubleshooting 9](#_TOC_250000)
8. **Créditos 10**

# Introdução

* 1. **Solução (sprint 3)**

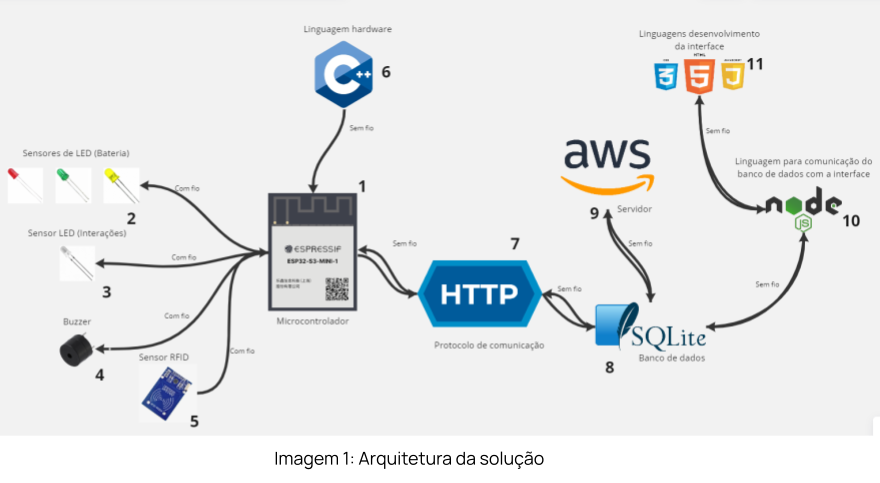
O dispositivo apresentado se trata de uma solução IOT integrada ao Prisma Estapar, tem como objetivo realizar o monitoramento e controle dos veículos em estacionamentos, para acompanhamento do processo de valet, veriﬁcando o estágio do valet bem como seu tempo estimado para conclusão. Outro importante objetivo é a associação do prisma ao manobrista, para que seja possível controlar e entender melhor o trabalho de cada um dos manobristas, criando novos KPIs de desempenho.

A solução em questão depende de 2 tecnologias essenciais, o Wi-Fi e o RFID, o Wi-Fi é necessário para que haja intercomunicação entre os prismas e o sistema do estacionamento, bem como para gerar a localização do veículo. Já a tecnologia RFID é necessária para que seja possível a autenticação e associação do manobrista ao prisma, servindo como chave de acesso para executar os processos do valet. Dessa maneira, o Prisma inteligente permite à Estapar agregar valor ao seu cliente, diminuindo o tempo de espera e dando mais autonomia ao usuário.





* 1. **Arquitetura da Solução (sprint 3)**







# Componentes e Recursos

|  |  |  |  |
| --- | --- | --- | --- |
| 2 | Sensores de LED (bateria) |  | Os três leds neste projeto irão representar o nível da bateria do dispositivo, a cor verde representará nível de carregamento alto, amarelo o nível médio e o vermelho nível baixo. |

**(sprint 3)**

## Componentes de hardware

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | Microcontrolador (ESP32 S3) |  | Será necessário dois ou mais microcontroladores  ESP32-S3, para realizar a geolocalização deles no projeto . Ele é um pequeno computador num único circuito integrado o qual contém um núcleo de processador, memória e periféricos programáveis de entrada e saída. Neste projeto ele é responsável por localizar o carro, processar as informações e se comunicar com o banco de dados. |

|  |  |  |  |
| --- | --- | --- | --- |
| 3 | Sensores de LED  (interações) |  | Esse led em especíﬁco será usado para indicar quando o cartão do manobrista encostou o cartão no RFID |
| 4 | Buzzer |  | O buzzer indica também quando o cartão é encostado no RFID, ele emite um sinal sonoro junto com o led branco |





|  |  |  |  |
| --- | --- | --- | --- |
| 5 | Sensor RFID |  | O sensor RFID acompanha junto o cartão e um chip para fazer a  leitura. Esse sensor será o principal autenticador e leitor do dispositivo, e servirá essencialmente para avançar entre os estados de manobra do veículo. |
| 6 | Jumper Macho/Mach o e Jumper Macho/Fêmea |  | Os Cabos Jumper são ﬁos de ligação elétrica, ideais e indispensáveis para quem faz montagens com Protoboard. |

|  |  |  |  |
| --- | --- | --- | --- |
| 7 | Protoboard |  |  |
|  | 830 Pontos |  |
|  |  | Protoboard, também conhecida como Breadboard, Placa de Ensaio ou Matriz de Contato, é uma placa com furos e conexões pré-deﬁnidas, que visa auxiliar a montagem de teste de circuitos eletrônicos experimentais de forma simples e ágil. |
| 8 | Carregador Portátil Bateria Power Bank |  | Um carregador portátil como auxiliar para carregar o outro ESP32-S3 que ﬁcará longe para calcular a distância de um esp para outro. |



8- Node JS: Um conjunto de bibliotecas que funciona como um interpretador de JavaScript.

|  |  |  |  |
| --- | --- | --- | --- |
| 9 | Resistor |  | Resistores para controlar a passagem de corrente elétrica  dos leds, para que não queimem. |



## Componentes externos

1. Toten: Para visualizar o tempo de espera para chegar o carro até o dono;
2. Computador: Para visualizar a produtividade do manobrista; 3- MQTT (MQ Telemetry Transport): Protocolo de comunicação;
3. IDE Arduino: Se conecta às placas Arduino para carregar programas e se comunicar com eles;
4. AWS: Plataforma de serviços de computação em nuvem; 6- SQLite: Biblioteca em linguagem C;

7- VSCode: Editor de código;

## Requisitos de conectividade

Rede Wi-ﬁ: O principal requisito para nossa IoT funcionar é uma rede de Wi-ﬁ que funcione bem, pois é por ela que conseguiremos saber a localização de cada carro.

HTTP: HTTP é um protocolo de transferência que possibilita que as pessoas que inserem a URL do seu site na Web possam ver os conteúdos e dados que nele existem. A sigla vem do inglês Hypertext Transfer Protocol.

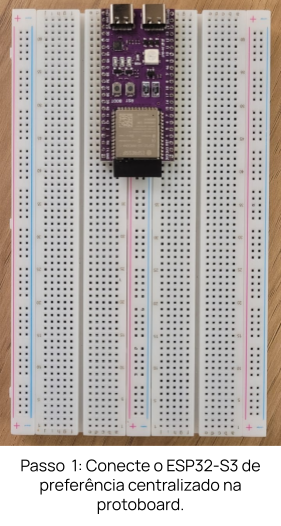
SQLite: SQLite é uma biblioteca em linguagem C que implementa um banco de dados SQL embutido. Será utilizado para armazenar e gerenciar o Banco de Dados.

AWS: O Servidor AWS é uma plataforma de serviços de computação em nuvem, que formam uma plataforma de computação na nuvem oferecida pela Amazon. Será utilizado para hospedar o serviço na nuvem.

FTM: O protocolo de medição de tempo ﬁno Wi-Fi (FTM) especiﬁcado no padrão IEEE 802.11-2016, fornece uma abordagem de alcance bidirecional para aprimorar a capacidade de posicionamento. Por meio dele conseguimos medir a distância de ESP32 de outro.

I2C: Signiﬁca o controlador integrado. Este é um protocolo de comunicação serial que pode conectar dispositivos de baixa

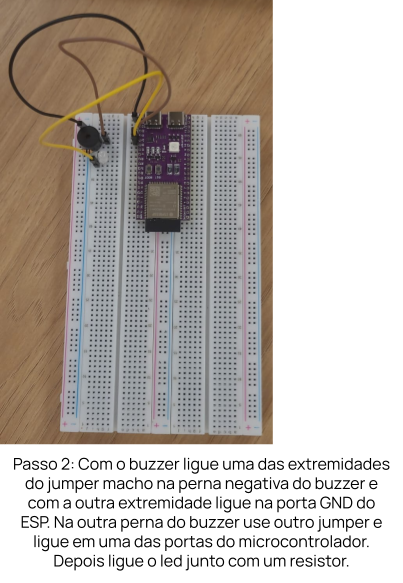
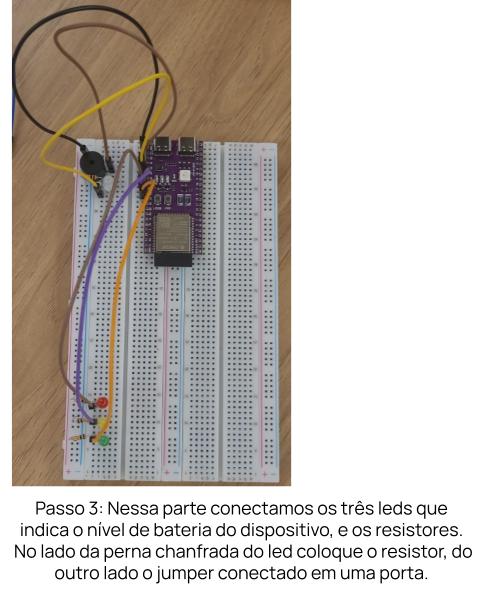


velocidade . É uma comunicação mestre-escravo na qual podemos conectar e controlar vários escravos a partir de um único mestre. Neste, cada dispositivo escravo possui um endereço especíﬁco.

# Guia de Montagem

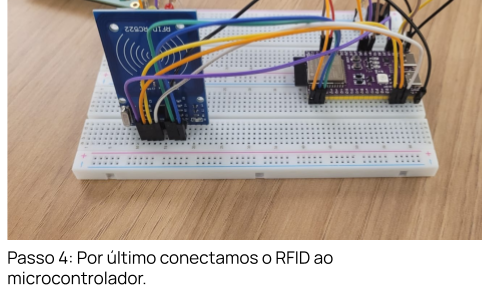




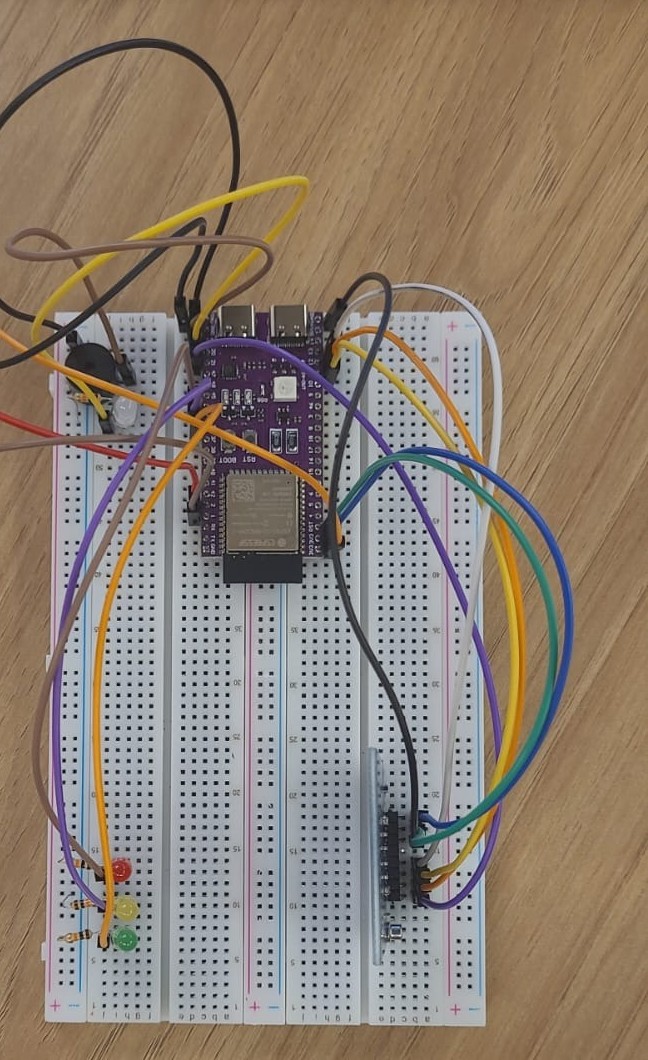
|  |  |
| --- | --- |
| SDA | Pino 18 |
| SCK | Pino 3 |
| MISO | Pino 10 |
| MOSI | Pino 8 |

Tabela de referência para conexão de cada pino do RFID na porta do ESP:

|  |  |
| --- | --- |
| **RFID** | **ESP32-S3** |
| GND | GND |
| VCC | 3V3 |
| RST | Pino 17 |





Por ﬁm o seu Hardware se parecerá com isso: **4. Guia de Instalação**

**(sprint 4)**

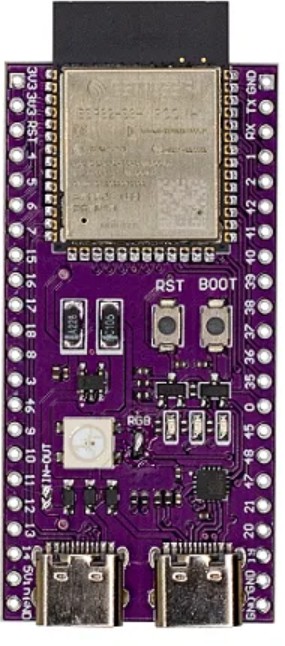
O primeiro passo para poder usar nossa solução IoT é baixar o software próprio para rodar o código, o arduino IDE. Acesse esse link ([hps://www.arduino.cc/en/software](https://www.arduino.cc/en/software)).

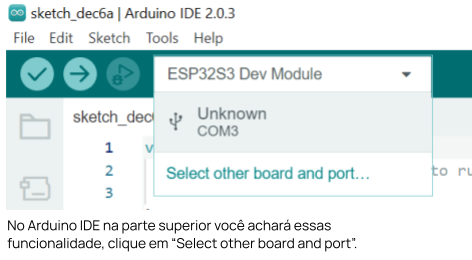


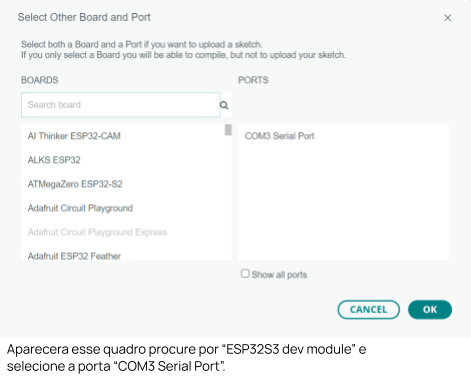
Após isso será baixado um arquivo compactado, vá até a área de trabalho e descompacte o arquivo, e seu Arduino IDE poderá ser usado.



Depois conecte o esp ao computador por meio de um cabo USB.





Para instalar o IoT precisamos também de um sinal de WiFi no estacionamento, para conectarmos eles na rede e assim conseguir sabermos a distância de cada um deles. Como testado pelo grupo, o alcance de um esp para outro é de 62 metros.

Em relação a localização do ESP ele estará em vários pontos do estacionamento, mas principalmente um ESP incorporado no carro



e outro na recepção. Como um ESP estará dentro de um recipiente ele terá que estar perto do carro.

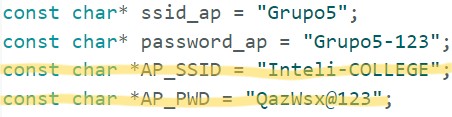
Nesse projeto temos dois roteadores um do WiFi que precisa estar num ponto estratégico conforme a arquitetura do estacionamento, para que todos os ESPs consigam ter um bom sinal. O os roteadores dos ESPs espalhados pelo estacionamento.

Outra parte importante do projeto é o totem que estará do lado do cliente para ele visualizar o tempo estimado da chegada do seu carro.

# Guia de Conﬁguração

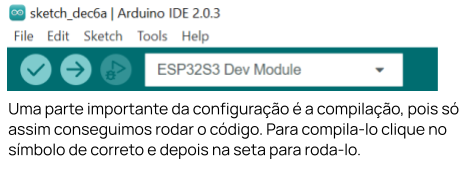
**(sprint 4)**

Para conectar o IoT no WiFi, é preciso mudar no código as variáveis “AP\_SSID” e “AP\_PWD” para o nome da rede e a senha do estacionamento.



Também é preciso instalar as bibliotecas Wire.h, MFRC522.h, SPI.h, WiFi.h, time.h, string.h, Arduino\_JSON.h e HTTPClient.h.



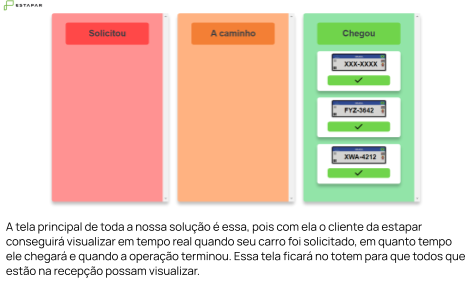






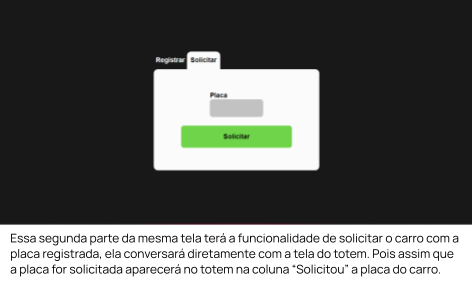
# Guia de Operação

**(sprint 5)**











# Troubleshooting

**(sprint 5)**

Liste as situações de falha mais comuns da sua solução (tais como falta de conectividade, falta de bateria, componente inoperante etc.) e indique ações para solução desses problemas.

|  |  |  |
| --- | --- | --- |
| **#** | **Problema** | **Possível solução** |
| 1 | Descarregamento muito | Checar se a bateria não está |





|  |  |  |
| --- | --- | --- |
|  | rápido do dispositivo | viciada e se está sendo carregado corretamente |
| 2 | Não conexão com a rede WiFi | Veriﬁcar o funcionamento do roteador, e se o nome da rede e a Senha do WiFi estão corretos no código |
| 3 | Tempo estimado da chegada do carro está errado | Examinar se existe algum problema com os ESPs e sua conexão com WiFi e banco de dados |
| 4 | O esquecimento de passar o crachá no RFID | Inserir manualmente no banco de dados, para que os dados ﬁquem corretos e de continuidade no processo |





# Créditos

**(sprint 5)**

Renato Machado - Código fonte do sistema; Luis Grainville - Montagem do Dispositivo; Luísa Leite - Front-end;

Luca Sarhan Giberti - Documentação. Giovani Andreussi - Documentação;