inteli

Controle do Documento

Histórico de revisões

Data	Autor	Versão	Resumo da atividade
‹17/11/2022›	‹Vitor Zeferino›	‹2.2›	‹Preenchimento dos componentes externos utilizados›
‹17/11/2022›	‹Melyssa Rojas›	‹2.1›	‹Preenchimento dos componentes de hardware utilizados na placa›
‹17/11/2022›	‹Melyssa Rojas›	‹2.3›	‹Preenchimento das tecnologias utilizadas no backend›
‹18/11/2022›	‹Júlia Togni›	‹3›	‹Preenchimento do guia de montagem›
‹18/11/2022›	‹Ariel›	‹1.1›	‹Preenchimento da solução›
‹18/11/2022›	‹Ariel›	‹1.2›	‹Preenchimento da arquitetura›

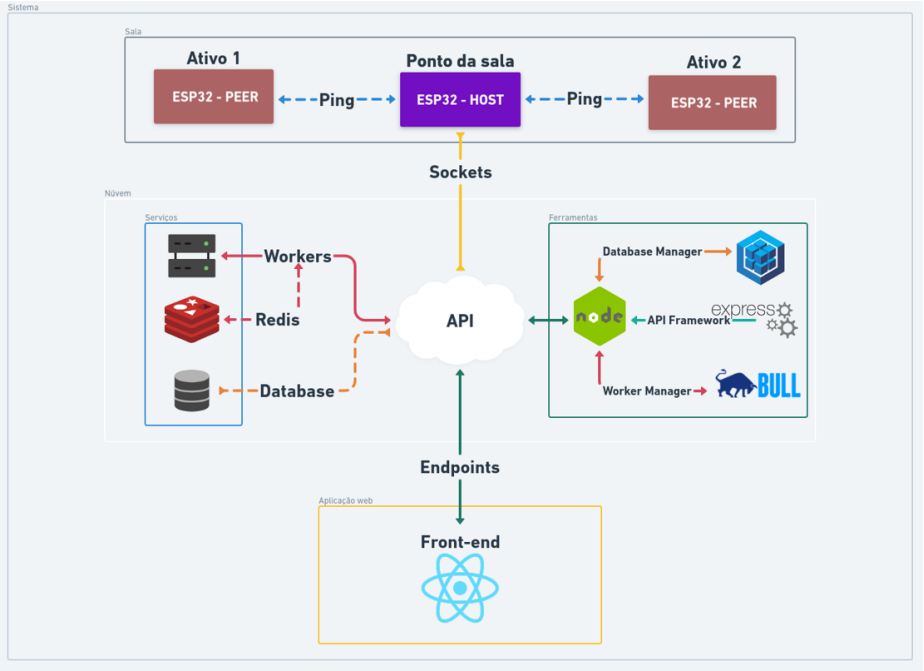
Índice

1. Introdução

1.1. Solução (sprint 3)

A nossa proposta de solução é implementar o sistema de rastreamento para que quando algum funcionário da controladoria for buscar por um ativo ele consiga, em tempo real, rastrear a sala em que o objeto está. Além disso, caso o aparelho saia do campus do IPT, esse funcionário é prontamente notificado para questionar ao responsável sobre a saída do equipamento sem aviso prévio ou qualquer outro tipo de problema que possa ter ocorrido.

1.2. Arquitetura da Solução (sprint 3)



git

ESP32 - HOST:
Responsável por receber as solicitações por *sockets web* e retornar as informações de posições dos equipamentos de sua área.
O *Host* estará diretamente conectado a uma rede wifi para se comunicar com a *API* e sempre estar a espera de novos comandos / solicitações. Exemplo: Recebe solicitação de posição dos equipamentos mais próximos, assim, o mesmo retorna as informações.

ESP32 - PEER:
Subordinado que estará sempre 'dormindo' durante inatividade, ao menos que seja solicitado a identificação para que o *Host* também tenha informação da sua distância atual do mesmo.
O *Peer* tem como seu único objetivo ser uma *Tag* que armazena um *ID* único, que estará atrelado a um equipamento (pelo sistema de *API*).

Serviços:
São os auxiliáreis da *API*, onde pode ser encontrado o banco de dados de longo prazo (como por exemplo *SQLITE*), o banco de dados de curto prazo (Redis) junto ao seu "sistema" de *workers*. O banco de dados de longo prazo é utilizado para armazenar dados como: O *ID* de tal dispositivo representa o equipamento X, gerenciamento de usuários e acessos...
Agora sobre os *workers*, são essencialmente tarefas realizadas em segundo plano, não atrapalhando os processos principais, como o serviço de rotas. (O *Redis* nessa situação é um armazém de dados temporários sobre as tarefas realizadas). Em prol do projeto, seria utilizado para monitoria / salvamento das posições dos equipamentos, assim gerando o histórico de posições do mesmo.

Front-End:
Para a criação do *Front-end* e consumo dos *endpoints* da *API*, utilizaremos em base o *React Native*(uma *framework* de desenvolvimento *web*, que facilita processos, sendo gerado a maioria do código de *front*, em próprio *JavaScript*).
Com essa tecnologia, será possível entregar de maneira mais eficaz a solução, principalmente que é facilmente portável para dispositivos móveis, abrindo portas para múltiplas plataformas de utilização do projeto.

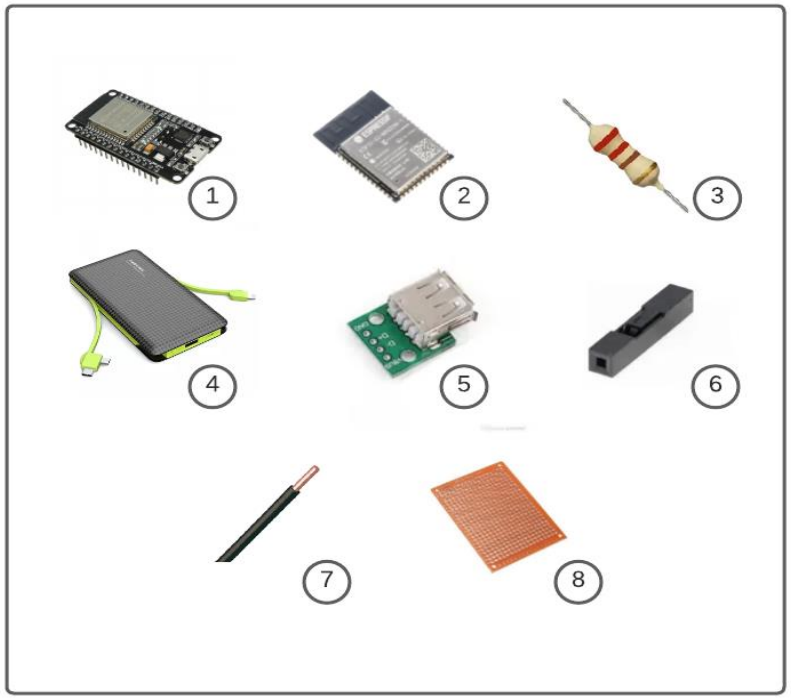
ESP32 Host para Esp32 Peer (*Ping*)
O *ping* que é citado como comunicação entre os microcontroladores, são nada mais que rotas de resposta do protocolo FTM, que faz o calculo de medida de distância entre os microcontroladores com base na conexão *wifi*.

API:
É o serviço a ser consumido pelos microcontroladores e *front-end*(aplicação web). Com a *API* será possível ter acesso a conexão por *sockets* e *endpoints* de rotas, sendo respectivamente, para a comunicação constante com os microcontroladores, pois os dados de posições / atualizações devem ser enviados em tempo real, assim o *sockets* teria um ótimo desempenho, reduzindo a necessidade de chamada repetitivas de rota em *"heath beat"* , já para as rotas de *endpoints* web, seria para consumo da aplicação que o usuário final utilizará, ou seja, rotas que requisita informações de posições de algum equipamento, histórico do mesmo...

Ferramentas:
As ferramentas são as tecnologias a serem utilizadas na *API*, sendo citado, NodeJS, que por sua vez, tem como pilares do projeto, as bibliotecas, *Sequelize*(faz o gerenciamento do banco de dados por ORM), *Express*(Framework para criar os endpoints e sockets) e *JsBull*(Que gerencia de forma fácil as tarefas de segundo plano, os *workers*).

2. Componentes e Recursos

2.1. Componentes de hardware



Nome (modelo e marca)	Especificação técnica	Funcionalidade na solução
devkit ESP-32-S3 **(1)**;	Integração do software no hardware;	Funciona como recebedor de posicionamento de outro esp , manda dados para o servidor via websockets, e ainda, dependendo de sua função, cria uma rede própria;
ESP32-S3-WROOM-1/1U **(2)**;	Processa dados e executa funções;	Processa dados e executa funções da aplicação;
2 resistores (resistência 220 Ohms) **(3)**;	Diminuidor da tensão;	Ele diminui a tensão para haver a leitura da bateria;
Carregador Portátil Power Bank Pineng 10000 Mah V8 **(4)**;	Bateria;	Serve para energizar o dispositivo Iot;
Módulo Adaptador USB Fêmea 2.0 **(5)**;	Conector de entrada USB;	Serve para conectar o esp32 ;

Conector Jumper fêmea **(6)**;	Auxilia na conexão dos cabos;	Ele diminui a tensão para haver a leitura da bateria;
Cabos sólidos **(7)**;	São cabos que possuem apenas um fio de cobre;	São usados para a conexão dos componentes da placa ilhada;
Placa Ilhada **(8)**;	Placa utilizada para auxiliar na conexão entre os componentes;	Essa placa permite a soldagem dos fios e dos componentes que vão ser usados;

2.2. Componentes externos

Liste aqui componentes como computadores, tablets e/ou celulares que deverão fazer parte da sua solução, bem como eventuais serviços em nuvem, softwares de edição de código ou outras aplicações utilizadas.

Nome	**Especificação técnica**	**Utilidade na solução**
API		É o serviço a ser consumido pelos microcontroladores. Acesso a conexão por sockets e endpoints de rotas para comunicação constante com os microcontroladores.
SQLITE		Banco de dados de longo prazo para guardar informações como ID.
Redis		Banco de dados de curto prazo para armazenar informações temporárias como localização do ESP.
NodeJS e suas bibliotecas		Gerenciamento do banco de dados, criar endpoints e gerenciar as tarefas de segundo plano.
React Native		Criação do Front-

		End e consumo dos endpoints da API.
Dispositivo	Conexão com a internet	Necessário para utilizar o dashboard da aplicação.

2.3. Requisitos de conectividade

Nome (tipo)	Especificação técnica
Node (ambiente de execução);	Plataforma utilizada para executar e criar códigos typescript fora do navegador;
Prisma (library);	Ferramenta que utiliza ORM (Object–relational mapping) para a criação das tabelas;
Express (framework);	Ferramenta utilizada para a criação de rotas, endpoints e da criação do servidor;
Socket.io (library);	Ferramenta utilizada para a criação da comunicação via websocket;
Typescript (linguagem de programação);	Linguagem utilizada na criação do backend;
npm (package manager)	Um gerenciador de pacotes que permite instalar, desinstalar e atualizar dependências do projeto;
WiFi (library)	Uma biblioteca que permite que os dispositivos inteligentes possam usufruir de funções da própria biblioteca, funções essas que permitem a conexão a uma rede local, criar uma própria rede como ponto de acesso;
WebSocketsClient (library)	Uma biblioteca que permite o cliente participar da troca de mensagens entre ele próprio e o servidor, por meio da inicialização da comunicação via ip e da porta do servidor;
Plataform.io	Editor de código para sistema embarcado com as bibliotecas necessárias instaladas;

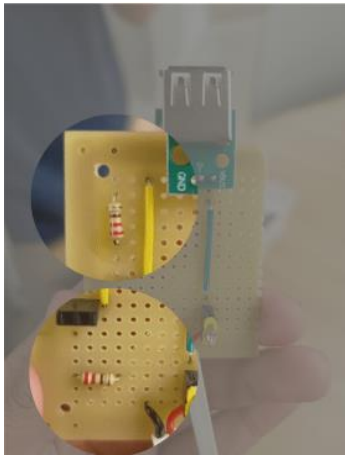
3. Guia de Montagem

Separe todos os componentes listados na seção 2.1 e utilize também um ferro de solda (cuidado).

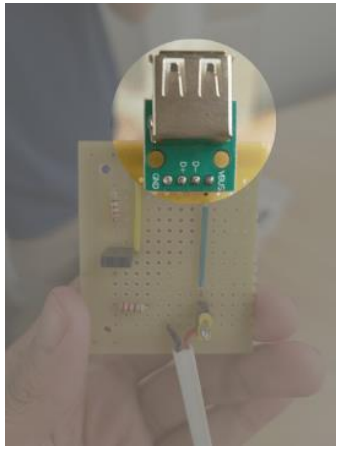
1. Para a montagem da placa de medidor de bateria do

esp:

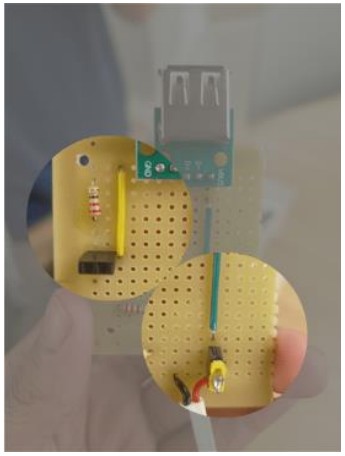
2. Na placa ilhada, adicione os dois resistores (componentes de número 3) e- imagem esquematizada abaixo



3. Adicione o componente 6 na placa



4. Conecte o componente 5 ao componente 6 e conecte ao componente 7 (o fio)



4. Guia de Instalação

(sprint 4)

Descreva passo-a-passo como instalar os dispositivos IoT no espaço físico adequado, conectando-os à rede, de acordo com o que foi levantado com seu parceiro de negócios.

Não deixe de especificar propriedades, limites e alcances dos dispositivos em relação ao espaço destinado.

Especifique também como instalar softwares nos dispositivos.

Utilize fotografias, prints de tela e/ou desenhos técnicos para ilustrar o processo de instalação.

5. Guia de Configuração

(sprint 4)

Descreva passo-a-passo como configurar os dispositivos IoT utilizando os equipamentos devidos (ex. smartphone/computador acessando o servidor embarcado ou a página na nuvem).

Utilize fotografias, prints de tela e/ou desenhos técnicos para ilustrar o processo de configuração.

6. Guia de Operação

(sprint 5)

Descreva os fluxos de operação entre interface e dispositivos IoT. Indique o funcionamento das telas, como fazer leituras dos dados dos sensores, como disparar ações através dos atuadores, como reconhecer estados do sistema.

Indique também informações relacionadas à imprecisão das eventuais localizações, e como o usuário deve contornar tais situações.

Utilize fotografias, prints de tela e/ou desenhos técnicos para ilustrar os processos de operação.

7. 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		
2		
3		
4		
5		

8. Créditos

(sprint 5)

Seção livre para você atribuir créditos à sua equipe e respectivas responsabilidades