inteli

Manual de Instruções

TECHNOTURTLES
Beacon School

Controle do Documento

Histórico de revisões

Data	Autor	Versão	Resumo da atividade
16/11/2022	Emanuele Morais	Preenchimento do tópico 1 - Introdução	Inserção do tópico 1.1 - Solução e 1.2 Arquitetura do solução
04/12/2022	Emanuele Morais, Gabriela Moares, Lucas Sales		
05/12/2022	Filipi Kikuchi	Preenchimento dos tópicos4 e 5	Inserção dos itens 4.1, 4.2 e 5.1 e subitens, 5.2 e 5.3

Índice

1. Introdução

1.1. Solução (sprint 3)

1.1.1. Problema

O colégio possui diversos aparelhos eletrônicos que auxiliam os alunos e colaboradores em seu aprendizado e, de acordo com a necessidade de cada aluno, pode disponibilizar esses equipamentos por um período de tempo. A problemática apontada pela Beacon School é que há uma grande dificuldade de localizar os equipamentos eletrônicos emprestados dentro do campus causando excesso de tempo gasto à procura dos itens emprestados e possíveis perdas.

1.1.2. Objetivos

O objetivo geral da solução proposta neste documento é uma solução em IoT (do inglês, “Internet of things“ e em portugues "Internet das coisas”) para a localização e rastreamento dos aparelhos eletrônicos que são patrimônio da escola. O resultado da implementação dessa solução será positiva pois reduzirá custos de operação, aumentará a segurança dos aparelhos em questão e o controle deles, sabendo onde eles estão localizados.

1.1.3. Planejamento Geral da Solução

O problema apresentado pelo parceiro se trata da dificuldade de gerenciamento e localização de dispositivos eletrônicos emprestados à comunidade escolar. A solução será composta por um sistema de localização de itens escolares em um produto IoT (“Internet Of Things”) que mapeia a escola e detecta onde os bens materiais estão ou se saíram de dentro do campus do colégio. Para resolução deste problema, a Beacon School disponibilizou acesso ao banco de dados que possui a relação dos dispositivos, incluindo os computadores e tablets pertencentes ao cliente. Além disso, também temos disponibilizada a planta baixa da unidade Campus da Beacon School para mapeamento do local. O projeto trata-se da instalação de microcontroladores nas áreas do colégio e nos dispositivos para que eles se comuniquem e, por meio de um software, os equipamentos sejam localizados. Os benefícios envolvem a redução dos custos operacionais e da perda de aparelhos, além disso, irá melhorar a segurança da informação e a gestão dos aparelhos tecnológicos. Para a definição de sucesso da solução serão avaliados critérios qualitativos e quantitativos, sendo eles, respectivamente, a melhoria da gestão de recursos, como tempo, dinheiro e qualidade de vida, após a implantação e a relação de dispositivos encontrados por período de tempo.

1.2. Arquitetura da Solução (sprint 3)

A arquitetura da Solução representa como será o funcionamento do ecossistema do projeto. Dessa maneira,na imagem abaixo, através das setas e números melhor descritos nas tabela é perceptível as etapas do funcionamento da arquitetura, além disso, na tabela abaixo é descrito os passos do funcionamento da solução.

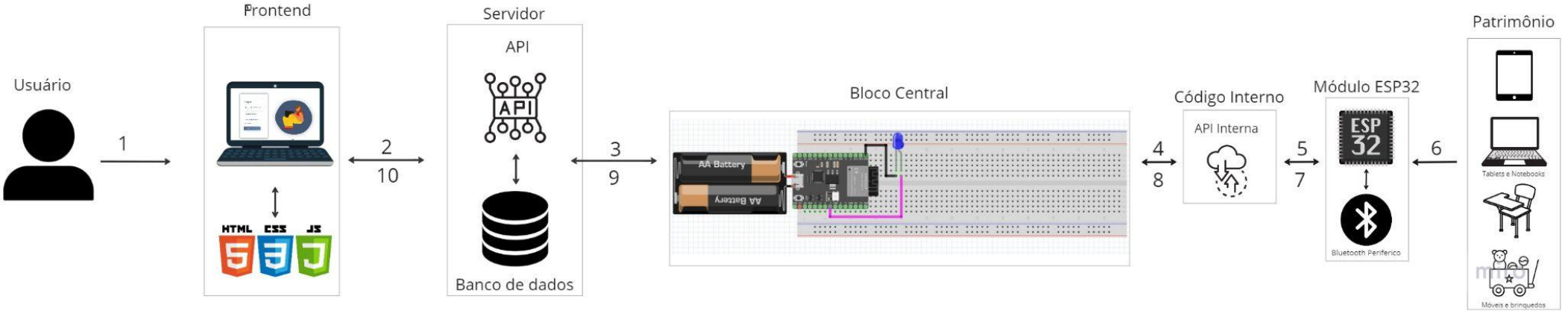


Tabela com descrição das etapas da arquitetura:

Conexão	Descrição da função
1	Usuário consulta local do dispositivo através de uma interface gráfica.
2	Interface faz requisições à API e ao banco de dados para quaisquer alterações na aplicação web.
3	A API que está no servidor faz uma requisição à API interna do microcontrolador, que verifica os status dos dispositivos próximos ou o próprio microcontrolador envia informações de perda e localização para o servidor.
4	O ESP32 será equipado com sensores que permitem a visualização de status (com um LED RGB) e tela LCD que transmite ao usuário da solução feedbacks da busca.
5	O microcontrolador se comunica com móveis e brinquedos por meio de etiquetas RFID.
6	Cada móvel e brinquedo possuirá uma etiqueta RFID e será requisitado, recorrentemente, sua localização e data de extração da informação.
7	É possível transformar o ESP32 em um ponto de acesso wi-fi, possibilitando o recebimento constante de informações do próprio dispositivo eletrônico.
8	É possível transformar o ESP32 em um scanner bluetooth, possibilitando o recebimento constante de informações do próprio dispositivo eletrônico.
9	Com as etapas descritas em 6 e 7 os equipamentos eletrônicos estarão conectados o tempo todo e em constante envio de informações.
10	Após as buscas a API interna do microcontrolador irá devolver a informação num Json para o servidor.
11	Por fim o servidor devolve as informações necessárias ao frontend e assim o usuário consegue ter as informações necessárias.

Fonte: Autoria própria

2. Componentes e Recursos

(sprint 3)

2.1. Componentes de hardware

Para a montagem do bloco central de hardware, precisamos dos seguintes componentes:

- 1x Microcontrolador ESP32
- 1x LED Difuso 5mm
- 2x Cabos jumper Macho/Macho
- 1x Bateria 3.3V
- 1x Protoboard (Opcional)

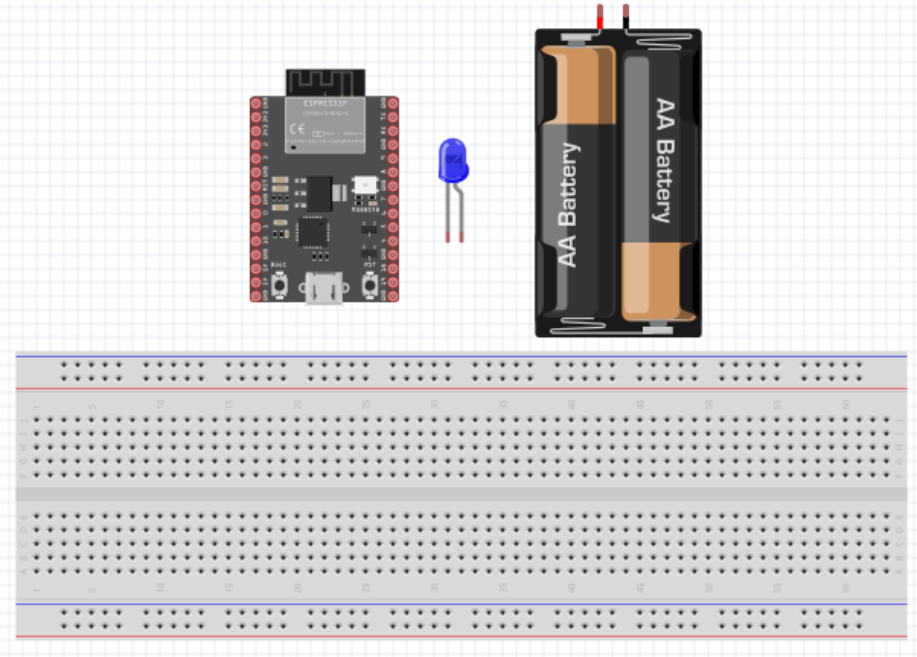


Figura 1: use sempre uma legenda e mencione o número da figura no corpo do texto. Cuidado para que detalhes da imagem não fiquem ilegíveis, como na imagem.

2.2. Componentes externos

Para utilizar a solução por completo, é recomendado o uso de uma rede de internet sem fio (Wi-Fi) e um dispositivo que possua acesso à ela (notebook, tablet, celular).

2.3. Requisitos de conectividade

Para o funcionamento correto da solução, as rotas que acionam os processos de leitura do microcontrolador devem estar ativas e hospedadas em um servidor (local ou na nuvem) que possui conexão com a internet. Para servidores locais, recomendamos servir a aplicação em Node.js. Da mesma forma, o ESP 32 deve estar conectado na mesma rede de internet para acessar o servidor e ser acessado para consultar o status dos dispositivos. Além disso, ESP 32 periféricos devem estar com suas funções bluetooth ativadas.

3. Guia de Montagem

(sprint 3)

Para a montagem do bloco central de hardware, sugerimos a seguinte sequência de passos, os quais todos necessitam do microcontrolador ESP32:

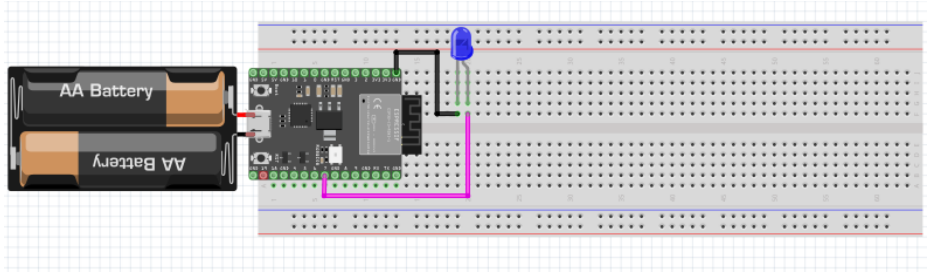
1. Conexão do LED

3.1. Montagem do LED

Nesta etapa, utiliza-se:

- 2x Cabos Jumpers Macho/Macho
- 1x LED Difuso 5mm

Conecte o cátodo (terminal maior do LED, cuja base é arredondada) à porta X do microcontrolador e o anodo ao GND, conforme a imagem abaixo:



4. Guia de Instalação

(sprint 4)

Para este guia de Instalação, separaremos a definição dos ESP's em duas maneiras:

- ESP32 Beacon, que deverá estar em todas as salas do campus, contendo o código que identifique exatamente o nome da sala que ele representa, ele servirá de identificador de cada sala e lerá os ESP's acoplados aos equipamentos na sua respectiva sala de modo a enviar os dados via wifi para o ESP32 Master;
- ESP32 peripheral, que serão acoplados a todos os dispositivos da escola com o respectivo código que o identifique como aquele dispositivo. Quando em um espaço com um ESP32 local, emitirá, via BLE, seus dados para o local e será contabilizado no banco de dados.

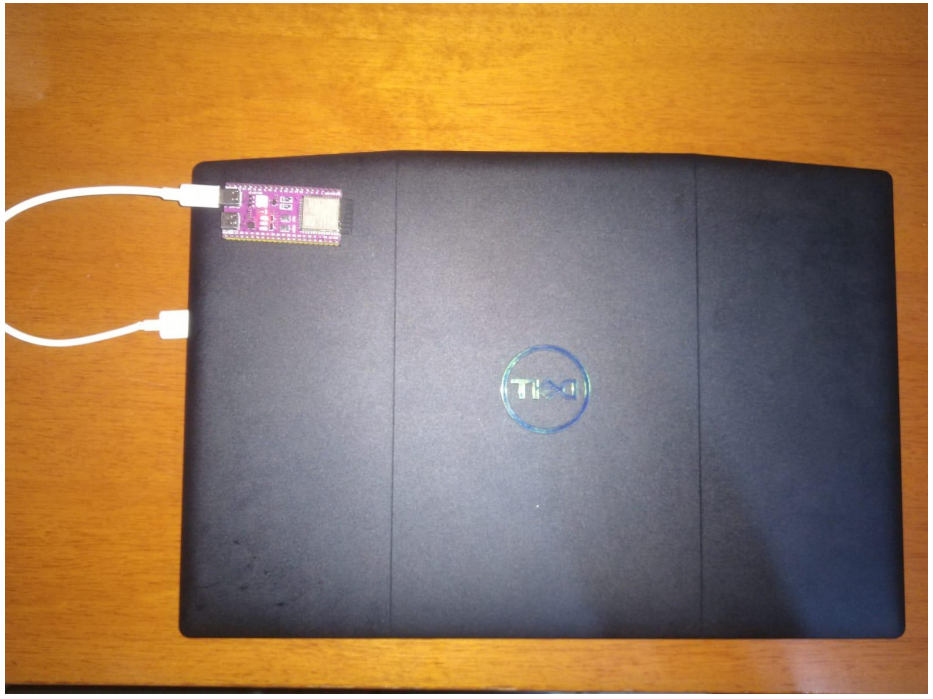
Para a instalação dos dispositivos na Beacon School, será necessário que todos os dispositivos eletrônicos estejam acoplados com um ESP32-S3. Ademais, haverá, aproximadamente, 2 ESP's por sala, além dos ESP's que serão espalhados nos demais espaços da escola de acordo com o tópico 5 do IoTDocument. Os ESP's deverão ser colados ou pregados no centro do teto das salas para melhor aproveitamento de seu alcance de leitura Bluetooth. Para os outros espaços, eles precisam estar a, aproximadamente, 10 metros de distância um do outro.

4.1. Instalação do ESP32 Beacon

Para a instalação do ESP32 Beacon, basta adesivá-lo num local que possua boa conexão com internet. Conectá-lo diretamente à tomada é recomendável para que não haja custos extras com baterias. Porém, se não for possível colocá-lo próximo à uma fonte de alimentação do prédio, deve-se acoplar uma bateria ao circuito. Segue uma foto do sistema instalado sem a presença de uma bateria:

4.2. Instalação do ESP32 Peripheral

Para a instalação do ESP32 Peripheral, basta adesivá-lo ao dispositivo. É recomendado que o acoplamento seja feito em superfícies lisas que não sejam bases de apoio. A alimentação pode ser tanto externa quanto provinda do próprio dispositivo. A seguir, uma foto ilustra um dispositivo com o ESP32 já implementado.



5. Guia de Configuração

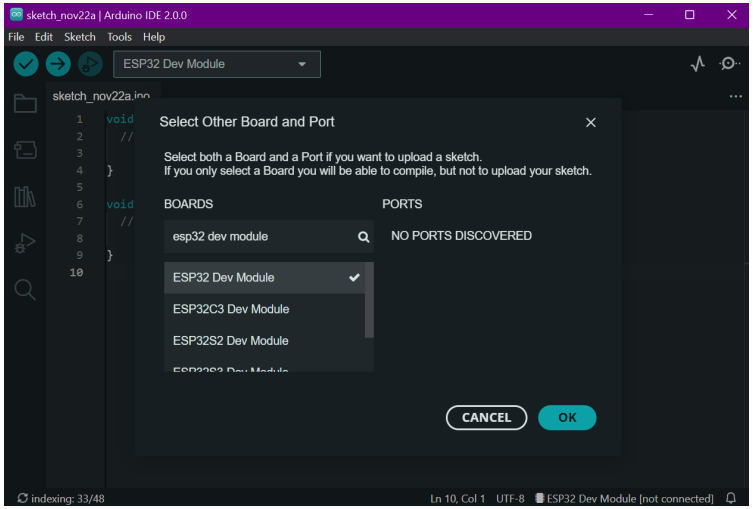
(sprint 4)

5.1. Ambientes de programação

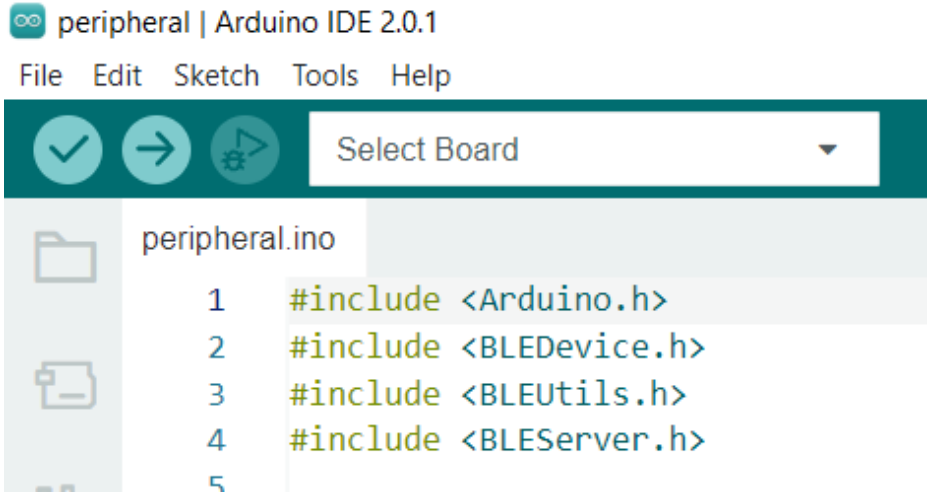
5.1.1. Arduino IDE

Para iniciar a configuração, faça o download da Arduino IDE (escolha versões 2.0.0 ou mais recentes e siga as instruções de instalação do próprio site da IDE.)

Depois de instalado, conecte o ESP32 ao computador com o auxílio de um cabo USB-C; em seguida abra a IDE. Dentro da aba indicada na imagem, clique em “Select Board” → ”Select other board and Ports”, selecione “ESP32S3 Dev Module” (em BOARDS) e escolha a porta COM que pertence à entrada USB do Esp32 (em ports). Depois de selecionadas, clique em “ok”.

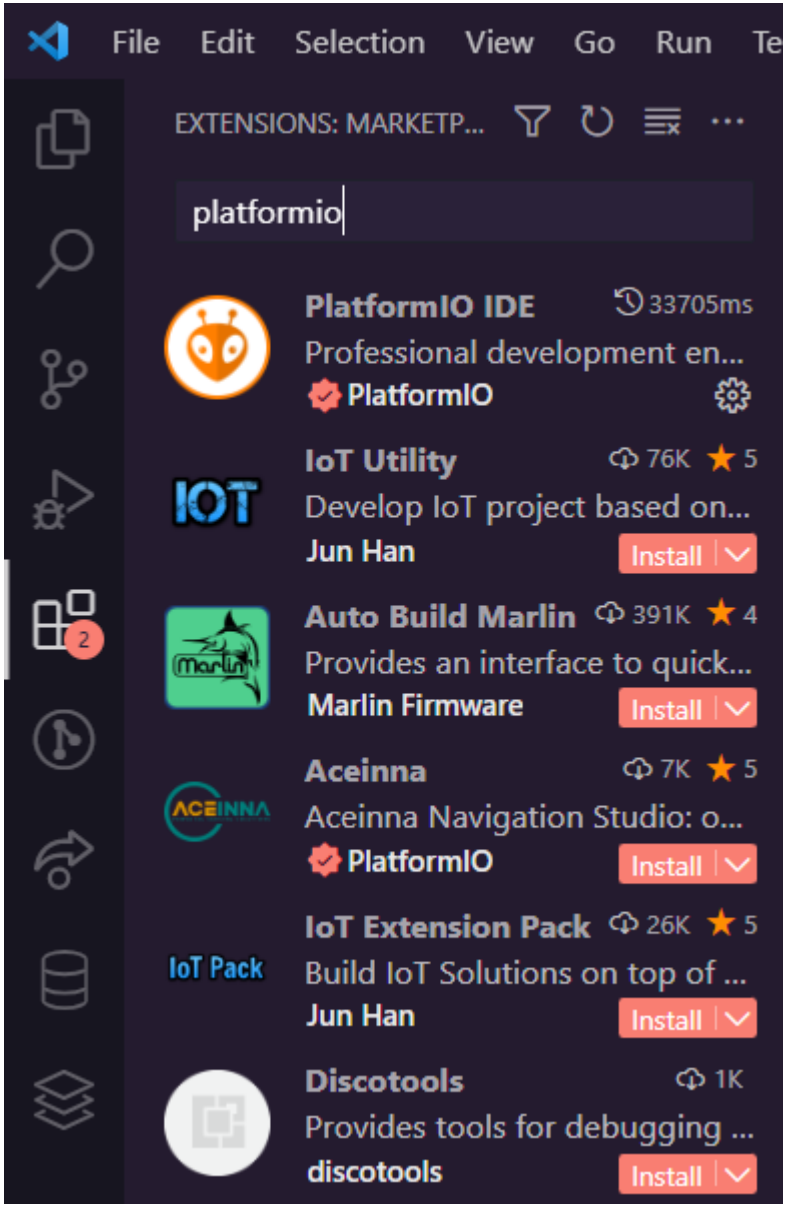


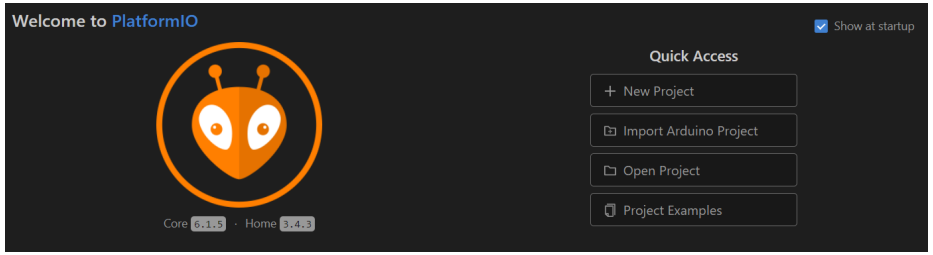
Uma vez que o arquivo foi aberto e as devidas modificações forem feitas, carregue o arquivo à memória do microcontrolador pressionando o botão “Upload”, representada por uma seta apontada para a direita.

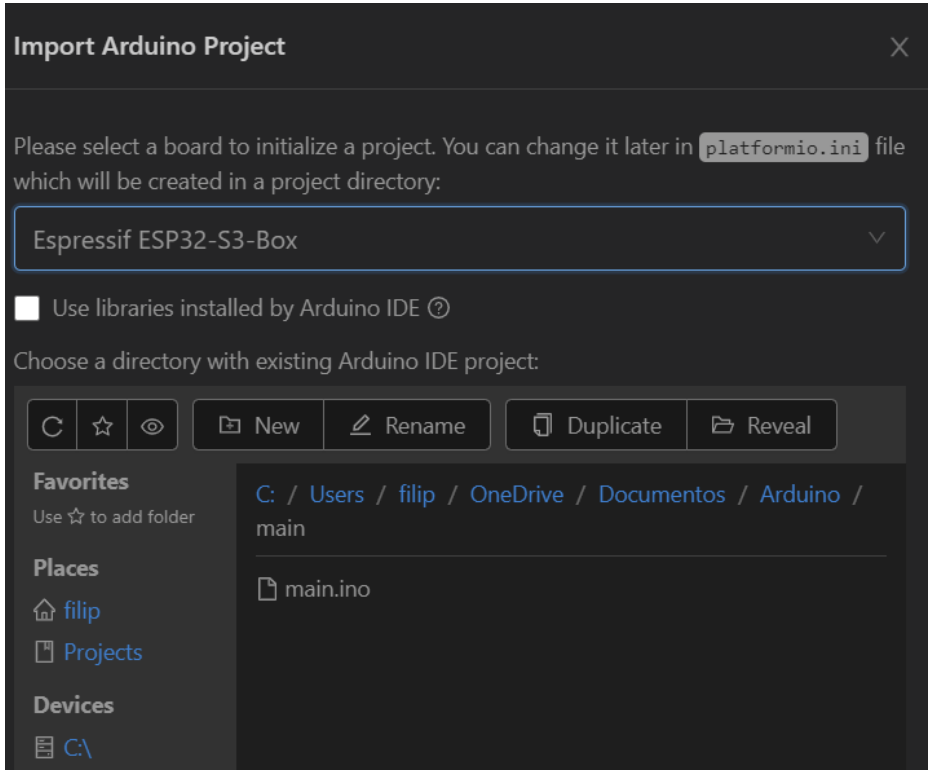


5.1.2. PlatformIO

Caso queira utilizar o VSCode, é necessário instalar a extensão “PlatformIO”. Com o plataforma aberta, vá para ‘’Extensions” e na barra de pesquisa, digite “platformIO” e clique em “install”. Após isso, reinicie o aplicativo e um ícone aparecerá na barra de ferramentas à esquerda. Selecione-o. Para abrir os arquivos necessários para configurar os dispositivos, clique em “Import Arduino Project” e procure os arquivos que serão carregados. Escolha a placa “Espressif ESP32-S3-Box” e clique em “Import”.







Uma vez que o arquivo foi aberto e as devidas modificações forem feitas, carregue o arquivo à memória do microcontrolador pressionando o botão “Upload”, na parte inferior esquerda.

5.2. Configuração do ESP32 Beacon

Para carregar o código com a função de Beacon para o microcontrolador, primeiramente deve-se abrir o arquivo “main.ino”. As alterações que devem ser feitas dizem respeito à rede de internet local e o endereço onde o servidor está hospedado.

1. Altere o conteúdo do ponteiro “ssid” para o nome da rede de Wi-Fi local.
2. Altere o conteúdo do ponteiro “password” para a senha da rede de Wi-Fi local.
3. Altere o conteúdo da variável “serverName” para a url onde está hospedada a API do servidor.
4. Carregue o código para o microcontrolador.

```
#include <Arduino.h>
#include <WiFi.h>
#include <HTTPClient.h>
#include <AsyncTCP.h>
#include <ESPAsyncWebServer.h>

#include <BLEDevice.h>
#include <BLEUtils.h>
#include <BLEScan.h>
#include <BLEAdvertisedDevice.h>

#include <ArduinoJson.h>

const char *ssid = "COLOQUE O NOME DA REDE WI-FI AQUI!"; // Exemplo: "Meu Wi-Fi"
const char *password = "COLOQUE A SENHA DA REDE WI-FI AQUI!"; //Exemplo "MinhaSenha123"

String serverName = "COLOQUE A URL DO SERVIDOR AQUI SEGUIDO DE '/test_device'"; //Exemplo "http://servidor.com/test_device"
String request;
```

5.3. Configuração do ESP32 Peripheral

Para carregar o código com a função de Peripheral (dispositivo que deve ser encontrado) para o microcontrolador, primeiramente deve-se abrir o arquivo “peripheral.ino”. As alterações que devem ser feitas dizem respeito ao nome pelo qual o dispositivo será reconhecido.

1. Escolha um nome para o dispositivo no campo “BLEDevice::init(“”)”.
2. Gere um UUID (Universally Unique Identifier) na plataforma geradora à sua escolha e troque os campos “SERVICE_UUID” e “CHARACTERISTIC_UUID”.
3. Carregue o código para o microcontrolador.

```
#include <Arduino.h>
#include <BLEDevice.h>
#include <BLEUtils.h>
#include <BLEServer.h>

// See the following for generating UUIDs:
// https://www.uuidgenerator.net/

#define SERVICE_UUID        "COLOQUE UM UUID AQUI!" //Exemplo "4fafc201-1fb5-459e-8fcc-c5c9c331914b"
#define CHARACTERISTIC_UUID "COLOQUE OUTRO UUID AQUI!" //Exemplo "beb5483e-36e1-4688-b7f5-ea07361b26a8"

void setup() {
  Serial.begin(115200);
  Serial.println("Starting BLE work!");

  BLEDevice::init("Escolha o nome do dispositivo aqui!");
  BLEServer *pServer = BLEDevice::createServer();
  BLEService *pService = pServer->createService(SERVICE_UUID);
  BLECharacteristic *pCharacteristic = pService->createCharacteristic(
                                         CHARACTERISTIC_UUID,
                                         BLECharacteristic::PROPERTY_READ |
                                         BLECharacteristic::PROPERTY_WRITE
                                       );
```

5.4. Configuração da API

Para enviar as requisições de maneira correta, é preciso saber o endereço de IP dos microcontroladores em modo Beacon. Quando o código é passado para o ESP32, é recomendado verificar o monitor Serial e verificar o endereço fornecido pelo dispositivo quando conectado à rede. Após isso, no arquivo “app.js”, deve-se trocar o valor de IP na rota da API que faz a requisição ao dispositivo em questão (Exemplificando, para configurar o beacon de nº2, deve-se trocar na rota “app.get(/test_device2)”a URL dentro da função “fetch(‘http:// … /list_devices’)”). Salve as alterações.

```
app.get('/test_device', async (req, res) => {
    // Exemplo "http://128.196.1.200/list_devices"
    const devices = await fetch('http://IP_DO_DISPOSITIVO_1!/list_devices').then(data => {
        return data.json();
    });

    console.log(devices);
    for (let i = 0; i <= 50; i++) {
        console.log(devices[`${i}`]);
        updateStatusfromAddr(devices[`${i}`]);
        insertLog(obj[i]);
    }
    res.json({
        "statusCode": 200
    });
});

app.get('/test_device2', async (req, res) => {
    // Exemplo "http://128.196.1.201/list_devices"
    const devices = await fetch('http://IP_DO_DISPOSITIVO_2!/list_devices').then(data => {
        return data.json();
    });
    for (let i = 0; i <= 50; i++) {
        console.log(devices[`${i}`]);
        updateStatusfromAddr2(devices[`${i}`]);
```

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