Projeto para localização de dispositivos bluetooth BLE indoor

Josimar de Andrade Silva, Rafael Gomes de Paula,
Wanderson Thiago da Silva Pagani

24/03/2020

Resumo – A proposta do projeto é criar um sistema de localizaçao em tempo real de dispositivos bluetooth dentro de um local fechado. O hardware consiste na criação de um dispositivo utilizando o ESP32, capaz de localizar outros dispositivos bluetooth, como IBeacons, através da triangulação destes dispositivos que denominamos "estações de rastreamento". As estações disponibilizaram as informações dos dispositivos encontrados através da internet, utilizando o protocolo MQTT e a plataforma Node-Red. O Software consiste em um website, construído utilizando Node.js como servidor aplicacional, capaz de se conectar ao Node-Red e fazer a interpretação dos dados. A plataforma é responsável por receber os dados e determinar qual é o dispositivo mais proximo dentro do raio da triangulação das estações.

Palavras-chave – IoT; Internet das Coisas; Comunicação; Localização; Sinal; Rádio Frequência; Alarmes; Indoor

Abstract – The project proposal is to create a real-time location system for bluetooth devices within an enclosed location. The hardware consists of creating a device using ESP32, capable of locating other bluetooth devices, such as IBeacons, through the triangulation of these devices that we call "tracking stations". The stations made available the information of the devices found through the internet, using the MQTT protocol and the Node-Red platform. The Software consists of a website, built using Node.js as an application server, capable of connecting to Node-Red and interpreting the data. The platform is responsible for receiving the data and determining which is the closest device within the radius of the triangulation of the stations

Keywords – IoT; Internet of Things; Comunication; Location; Signal; Radio Frequency; Alarm; Indoor

1 Introdução

De acordo com LIMA (2014) o primeiro sistema de identificação por radiofrequência (RFID) surgiu em 1937 com a invenção do primeiro radar, liderado pelo escocês Sir Robert Alexander Watson-Watt da United States Naval Research Laboratory. Após a invenção do primeiro radar, a tecnologia RFID tem sido usada para diversos tipos de aplicação, como controle de segurança, monitoramento, controle de temperatura, rastreabilidade, entre outras diversas situações. Segundo OMEGA (2020) os sensores sem fio utilizados em aplicações de localização são ferramentas de medição padrão equipadas com transmissores utilizados para converter sinais de instrumentos de controle de processo em uma transmissão de rádio. O sinal de rádio é interpretado por um receptor que converte o sinal sem fio em uma saída específica, tal como uma corrente analógica ou uma análise de dados feita por um software. LOUREIRO (2003) exemplifica que cada sensor possui características particulares que variam de acordo com o ambiente e objetivo do uso, sendo alguns desses sensores: com endereçamento, para mobilidade, limitação de energia e outros. A pesquisa apresentada neste trabalho se refere aos sensores RFID, aplicados para simplificação de atividades humanas em locais indoor. Um exemplo disso são os sistemas de localização e cálculo de distância, que utilizam as propriedades da rádio frequência como um recurso para calcular a posição ou a longitude entre dois objetos. O objetivo principal deste trabalho é demonstrar uma das técnicas baseada no sinal Bluetooth de Baixa Energia (BLE) mais utilizadas nos sensores de localização indoor, a Indicação de Intensidade do Sinal de Rádio (RSSI), que utiliza, respectivamente, o tempo de chegada do sinal a uma ERB (Estação Base), o ângulo de chegada do sinal em relação às Estações de Rádio Base (ERB's) e a intensidade do sinal, para calcular a posição de um objeto. Este trabalho, portanto, orientar-se-á no sentido de analisar o processo de localização de pessoas em um ambiente indoor de modo a utilizar a técnica RSSI, podendo ajudar a diminuir o tempo de atuação envolvendo problemas em sistemas de missão crítica.

2 Pesquisa Bibliográfica

Ao desenvolver as pesquisas foi utilizado o método de revisão literária, sendo esse um método qualitativo, apoiando-se em técnicas de pesquisa de dados. Realizado pela análise de conteúdos e artigos de sites, artigos técnicos, dissertações de mestrado, pesquisas bibliográficas, manuais e normas técnicas, publicados nos períodos dos últimos 20 anos, sobre o tema em questão. Ainda é um grande desafio para as tecnologias de localização em ambientes internos. O Sistema de Posicionamento Global (GPS), por exemplo, é uma das descobertas mais eficientes, tratando-se de tecnologias baseadas nos sensores RFID. No entanto, os metros quadrados de um ambiente indoor estão fora do alcance de seus 28 satélites (LIMA, 2001). Na realidade, estes ambientes não são mapeados para que um sistema de GPS possa localizá-los. Desta forma, o foco de pesquisas associadas à localização de pessoas e objetos têm se baseado na busca por tecnologias apropriadas a ambientes internos. Muitas das dificuldades estão relacionadas, já que estes locais apresentam uma estrutura que não se vê externamente, como COLEMAN e (WESTCOTT, 2009):

1. Alta atenuação e difusão do sinal, devido aos inúmeros obstáculos;
2. Mudanças temporais relacionadas à movimentação de pessoas e abertura de portas;
3. Multipath causado pela reflexão das paredes e móveis.

Por outro lado, os ambientes internos oferecem algumas facilidades, já que não sofrem interferências de fatores climáticos; podem ser facilmente mapeados e possuem melhor infraestrutura

(acesso a internet e energia elétrica). Portanto, neste trabalho, o estudo da localização baseada na leitura bidirecional do RSSI será destinado a ambientes indoor.

3 Estudo de Caso

3.1 Objetivo

O objetivo do estudo de caso referente a este trabalho é demonstrar a necessidade que os data centers tem em resolver problemas de infraestrutura em um menor tempo possível e o que impacta com um problema não resolvido, influenciando na sociedade. Para demonstrar a utilização do projeto, foi escolhido o Data Center Prodesp, considerado ambiente de missão crítica, onde em uma situação de crise na infraestrutura, o tempo de atuação dos técnicos deve ser o mais rápido possível

3.2 Justificativa

De acordo com ICOR (2020), existem muitos fatores de risco de paralisação nos Data Centers, como falhas naturais, humanas e de origem dos hardwares e sistemas instalados. Onde o maior percentual de causas da paralisação dos data centers são por falhas humanas. Devido a planta do Data Center Prodesp ser de grande escala em relação ao numero de técnicos para manter a energia elétrica constante alimentando os equipamentos de Tecnologia da Informação (TI) e equipamentos de climatização (Ar-condicionado de precisão), foi possível uma projeção para melhorar o tempo de atuação dos técnicos em um ocorrência (alarmes) com o uso de um localizador de pessoas indoor por meio da tecnologia RSSI. Considerado que a equipe de manutenção residente do Data Center Prodesp leva em média 5 minutos para localizar uma ocorrência através de um sistema supervisório Controle de Supervisão e Aquisição de Dados (SCADA), com a ajuda do localizador esse tempo pode ser menor, de acordo com a proximidade entre o técnico atuante e o equipamento em alarme.

3.3 Relevância para a Sociedade

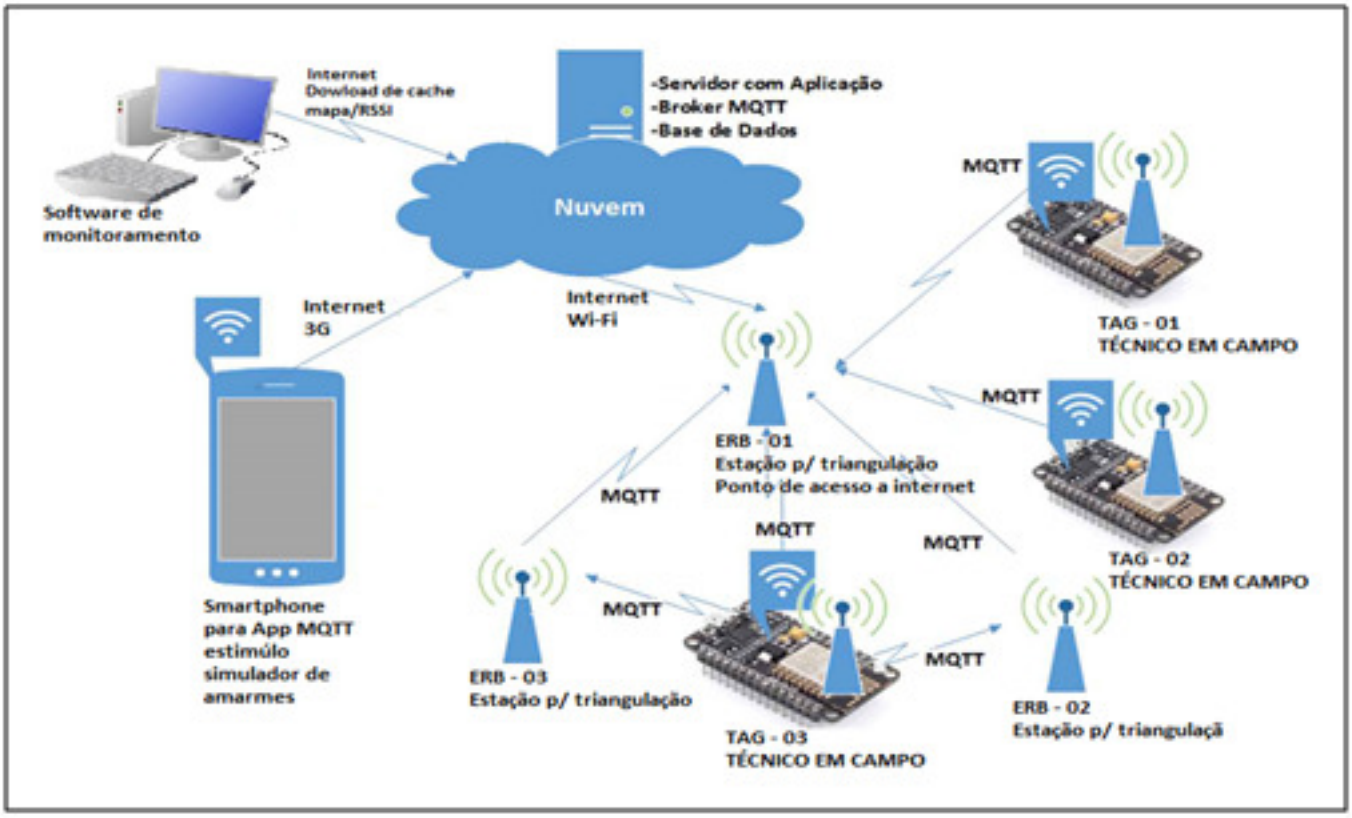
Segundo PENSO (2020), os data centers são importantes para realizar atividades simples do cotidiano, sendo desde a Internet, que você acessa para se divertir com jogos e vídeos, até os pagamentos feitos no aplicativo do seu banco, consultas de saldo e extrato, transferências ou aplicações.

4 Definição do Projeto

4.1 Topologia

A internet das coisas (IOT) proporcionou o uso do protocolo Transporte de Telemetria do Serviço de Enfileiramento de Mensagens (MQTT) para comunicação entre os hardwares de forma harmoniosa sendo possível troca de informações entre as TAG's, ERB's, aplicativo para celular (APP) de alarmes, software de monitoramento e servidor em nuvem. Segundo Zhu (2010) a origem do protocolo MQTT surgiu no laboratório Massachusetts Institute of Technology (MIT) com a pesquisa no campo de localização e identificação usando sensores sem fio, em 1999, onde deu inicio a área de estudos da Internet das Coisas.

Figura 1 – Topologia MQTT



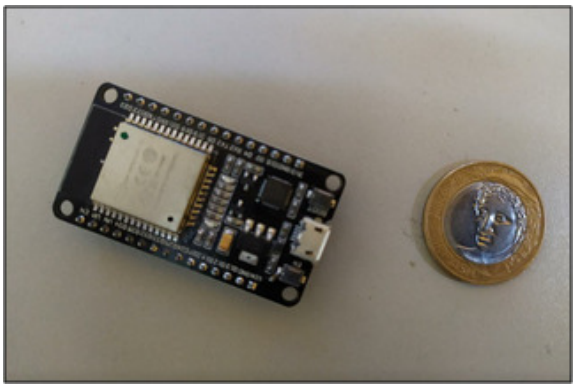
Fonte: Própria

Na figura 01 pode-se ver o fluxo de dados utilizando o protocolo MQTT para este trabalho. O servidor com aplicação MQTT Node-RED da International Business Machines (IBM) recebe os estímulos de um APP que simula um possível alarme e equipamento instalado em um ambiente indoor. As ERB's verificam a intensidade do sinal RFID vindas das TAG's IBeacons, que também são informadas sobre o alarme mais próximo.

4.2 Hardware

A definição do hardware foi definido da seguinte forma, por placas ESP-32 para as ERBs devido a praticidade de programação e conjuntura de tecnologias RFID, por TAGs do tipo beacon paraos localizadores devido a facilidade de pessoas portarem no bolso e a tecnologia Bluetooth de Baixa Energia (BLE).

Figura 2 – TAG Beacon



Fonte: Própria

Figura 3 – ERB ESP32

Fonte: Própria

4.3 Programa

Dividimos o projeto em duas partes, projeto de hardware e software, surgindo assim, dois programas fundamentais. Um programa definido para as ERBs e o outro para o visualizador web (software). As TAGs foram isoladas de programação devido às limitações do próprio fabricante dos beacon.

4.3.1 Firmware

```
/*  SENAI - PROJETO ESTA  O SCANNER BLUETOOTH PARA LOCALIZA  O
   INDOOR UTILIZANDO ESP32 */
//Biblioteca String para manipula  o de variavel texto
#include <string>
//Biblioteca Wifi para gerenciamento de redes sem fio
#include <WiFi.h>
//Biblioteca PubSubClient para gerenciamento do protocolo MQTT
#include <PubSubClient.h>
//Bibliotecas BLE para gerenciamento do bluetooth
#include <BLEDevice.h>
#include <BLEUtils.h>
#include <BLEScan.h>
#include <BLEAdvertisedDevice.h>
```

```
13 //Constante que define a quantidade maxima que a esta ao poder
   ler
14 #define MAX_BEACONS_BUFFER 50
15 //Constante que define o tempo de busca do bluetooth, em segundos
16 const int beaconScanTime = 4;
17 //Constante que o nome do dispositivo bluetooth da esta  o
18 const char *stationName = "Station 1";
19 //Constante que define o nome da rede WiFi
20 const char *ssid = "[YOUR_SSID]";
21 //Constante que define a senha da rede Wifi
22 const char *password = "[YOUR_PASSWORD]";
23 //Constante que define o nome do servidor Broker MQTT
24 const char *mqttServer = "[BROKER_HOST]";
25 //Constante que define a porta do Broker MQTT
26 const int mqttPort = [BROKER_PORT];
27 //Constante que define o usu rio do Broker MQTT
28 const char *mqttUser = "[BROKER_USER]";
29 //Constante que define a senha do Broker MQTT
30 const char *mqttPassword = "[BROKER_PASS]";
31 //Constante que define os t picos de subscri  o para o MQTT
32 const char *subTopics[1] = {"/stations/command"};
33 //Constante que define os t picos de publica  o para o MQTT
34 const char *pubTopics[1] = {"/stations/beacons/get"};
35
36 //Estrutura de dados para armazenamento dos beacons
37 typedef struct
38 {
39   //Endere o MAC do dispositivo bluetooth encontrado
40   char address[17];
41   //Potencia do sinal do dispositivo bluetooth encontrado
42   int rssi;
43   //Nome do dispositivo bluetooth encontrado
44   char *bName;
45
46 } BeaconData; //Nome da estrutura de dados
47
48 //Objeto Wifi Client para gerenciamento da rede Sem fio
49 WiFiClient espClient;
50 //Objeto PubSubCliente para gerenciamento do protocolo MQTT
51 PubSubClient client(espClient);
52 //Array de Objetos que armazenara os dados dos beacons encontrados
53 BeaconData beacons[MAX_BEACONS_BUFFER];
54 //Variavel para localiza  o de um beacon dentro do array "beacons
   "
55 uint8_t beaconIndex = 0;
56 //Buffer para envio dos beacons no formato JSON
57 uint8_t message_char_buffer[MQTT_MAX_PACKET_SIZE];
58
59 //Classe personalizada que herda os m todos
   BLEAdvertisedDeviceCallbacks para subrecarga do m todo onResult
```

```
60 class MyAdvertisedDeviceCallbacks : public
      BLEAdvertisedDeviceCallbacks
61 {
62 public:
63   //Sobrecarga do m todo onResult para tratamento dos dados
        recebidos pela busca bluetooth
64   void onResult(BLEAdvertisedDevice advertisedDevice)
65   {
66     //Varivel relacionada a beaconIndex externa.
67     extern uint8_t beaconIndex;
68     //Varivel relacionada a beacons externo.
69     extern BeaconData beacons[];
70
71     //Valida a quantidade de beacons encontrados na busca
72     if (beaconIndex >= MAX_BEACONS_BUFFER) return;
73     //Valida a potencia de sinal do dispositivo encontrado
74     if (advertisedDevice.haveRSSI()) beacons[beaconIndex].rssi =
          advertisedDevice.getRSSI();
75     //Caso n o possua sinal,    definido 0 para esse dispositivo
76     else beacons[beaconIndex].rssi = 0;
77
78     //Atribui o endere o MAC do dispositivo a variavel beacons[
          beaconIndex].address
79     strcpy(beacons[beaconIndex].address, advertisedDevice.
          getAddress().toString().c_str());
80
81     //Atribui o nome do dispositivo a variavel beacons[beaconIndex
          ].bName
82     std::string str = advertisedDevice.getName();
83     beacons[beaconIndex].bName = new char[str.length() + 1];
84     strcpy(beacons[beaconIndex].bName, str.c_str());
85
86     //Incrementa mais um ao contador de dispositivos
87     beaconIndex++;
88   }
89 };
90 //M todo de configura  o do ESP
91 void setup()
92 {
93   //M todo de in cio a porta serial na velocidade 115200
94   Serial.begin(115200);
95   //M todo de in cio ao dispositivo bluetooth
96   BLEDevice::init(stationName);
97 }
98 //M todo de configura  o WiFi
99 void connectWiFi()
100 {
101   //Valida se o Wifi esta conectado a rede, caso nao esteja, inicia
102   if (WiFi.status() != WL_CONNECTED) WiFi.begin(ssid, password);
103
```

```
104   //Aguarda at  que o status do wifi seja conectado
105   while (WiFi.status() != WL_CONNECTED)
106   {
107     delay(2000);
108     Serial.println("Connecting␣to␣WiFi...");
109   }
110 }
111 //M todo de configura  o do protocolo MQTT
112 void connectMQTT()
113 {
114   //Executa enquanto o protocolo est  desconectado
115   while (!client.connected())
116   {
117     //Define as credenciais do Broker MQTT
118     client.setServer(mqttServer, mqttPort);
119     //Define o m todo de callback para os t picos de subscri  o
120     client.setCallback(mqttCallback);
121 
122     Serial.println("Connecting␣to␣MQTT...");
123     //Tenta conex o e Valida se a conx o com o Broker funcionou
124     if (client.connect("ESP32Client", mqttUser, mqttPassword))
125     {
126       Serial.println("Client␣Connected");
127       Serial.println("Subscribing␣to␣topic:");
128       boolean result;
129 
130       Serial.print(subTopics[0]);
131       //Faz a subscri  o no t pico definido em subTopics[0]
132       result = client.subscribe(subTopics[0]);
133       Serial.print(".....");
134       Serial.println(result);
135     }
136     else
137     {
138       Serial.print("failed␣with␣state:␣");
139       Serial.print(client.state());
140       delay(2000);
141     }
142   }
143   //Coloca o protocolo MQTT em loop para novas mensagens
144   client.loop();
145 }
146 //M todo que escaneia os beacons e armazena os dados encontrados
147 void scanBeacons()
148 {
149   delay(1000);
150   //Objeto que define o scanner bluetooth
151   BLEScan *pBLEScan = BLEDevice::getScan();
152   //Objeto que define o m todo de callback
153   MyAdvertisedDeviceCallbacks cb;
```

```
154   //Configura o scanner para o callback definido
155   pBLEScan->setAdvertisedDeviceCallbacks(&cb);
156   //Configura o scanner como ativo
157   pBLEScan->setActiveScan(true);
158   //Objeto que receber  os dispositivos encontrados pelo scanner
159   BLEScanResults foundDevices = pBLEScan->start(beaconScanTime);
160   //Configura o scanner como inativo
161   pBLEScan->stop();
162   //Aguarda 1 segundo
163   delay(1000);
164 }
165 //M todo que trata as mensagens recebidas pelo protocolo MQTT
166 void mqttCallback(char *topic, byte *payload, unsigned int length)
167 {
168   //Objeto para convers o da mensagem em String
169   String strPayload = "";
170
171   //Convers o da mensagem em String
172   for (int i = 0; i < length; i++) strPayload += (char)payload[i];
173
174   Serial.print("Message␣arrived␣in␣topic:␣");
175   Serial.print(topic);
176   Serial.print("␣=>␣");
177   Serial.println(strPayload);
178
179   //Valida ao da mensagem para comando de busca
180   if (String(topic) == String(subTopics[0]) && strPayload == "find"
        ) scanBeacons();
181   //Valida ao da mensagem para comando de envio de dados no
        formato CSV
182   else if (String(topic) == String(subTopics[0]) && strPayload == "
        sendCSV") sendBeaconsCSV();
183   //Valida ao da mensagem para comando de envio de dados no
        formato JSON
184   else if (String(topic) == String(subTopics[0]) && strPayload == "
        sendJSON") sendBeaconsJSON();
185   //Valida ao da mensagem para comando de busca e envio de dados
        no formato CSV
186   else if (String(topic) == String(subTopics[0]) && strPayload == "
        findAndSendCSV") {
187     //Busca os dispositivos
188     scanBeacons();
189     //Envia os dispositivos encontrados em formato CSV
190     sendBeaconsCSV();
191   }
192   //Valida ao da mensagem para comando de busca e envio de dados
        no formato JSON
193   else if (String(topic) == String(subTopics[0]) && strPayload == "
        findAndSendJSON")
194   {
```

```
195     //Busca os dispositivos
196     scanBeacons();
197     //Envia os dispositivos encontrados em formato JSON
198     sendBeaconsJSON();
199   }
200 }
201 //M todo de envio das informa  es em formato JSON
202 void sendBeaconsJSON()
203 {
204   //Varival de resultado da publica  o no Broker MQTT
205   boolean result;
206   //Variavel de mensagem payload
207   String payload = "{\"e\":[";
208
209   for (uint8_t i = 0; i < beaconIndex; i++)
210   {
211     //Incremento do endere o
212     payload += "{\"m\":\"";
213     payload += String(beacons[i].address);
214     //Incremento da for a do sinal
215     payload += "\",\"r\":\"";
216     payload += String(beacons[i].rssi);
217     //Incremento do nome
218     payload += "\",\"n\":\"";
219     payload += String(beacons[i].bName);
220     payload += "\"}";
221     if (i < beaconIndex - 1) payload += ',';
222   }
223   //Incremento do endere o da esta  o
224   payload += "],\"stMac\":\"";
225   payload += String(WiFi.macAddress());
226   //Incremento do nome da esta  o
227   payload += "\",\"stNome\":\"";
228   payload += stationName;
229   payload += "\"}";
230
231   //Publica  o da mensagem no Broker
232   Serial.print("Publishing: ");
233   Serial.print(payload);
234   payload.getBytes(message_char_buffer, payload.length() + 1);
235   result = client.publish_P(pubTopics[0], message_char_buffer,
        payload.length(), false);
236   Serial.print(" -> Result: ");
237   Serial.println(result);
238
239   //Reseta a variavel para posi  o inicial
240   beaconIndex = 0;
241 }
242 void sendBeaconsCSV()
243 {
```

```
244   //Varival de resultado da publica  o no Broker MQTT
245   boolean result;
246   for (uint8_t i = 0; i < beaconIndex; i++)
247   {
248     //Variavel de mensagem payload e Incremento do endere o da
          esta  o
249     String payload = String(WiFi.macAddress());
250     payload += ";";
251     //Incremento do nome da esta  o
252     payload += stationName;
253     payload += ";";
254     //Incremento do nome
255     payload += String(beacons[i].bName);
256     payload += ";";
257     //Incremento do endere o
258     payload += String(beacons[i].address);
259     payload += ";";
260     //Incremento da potencia do sinal
261     payload += String(beacons[i].rssi);
262
263     //Publica  o da mensagem no Broker
264     Serial.print("Publishing:␣");
265     Serial.print(payload);
266     result = client.publish(pubTopics[0], (char *)payload.c_str());
267     Serial.print("␣->␣Result:␣");
268     Serial.println(result);
269   }
270   //Reseta a variavel para posi  o inicial
271   beaconIndex = 0;
272 }
273 void loop()
274 {
275   //M todo que valida a conex o do Wifi
276   connectWiFi();
277   //M todo que valida a conexao com Broker MQTT
278   connectMQTT();
279   //Aguarda 0.5 segundos
280   delay(500);
281 }
```

4.3.2 Software

Devido a complexidade do software ficará disponível o link para acesso ao conteúdo.

https://github.com/rafargp/Projeto_Senai

5 Validação

5.1 Testes

Os primeiros testes foram feitos entre uma ERB utilizando uma placa ESP-32 e uma TAG utilizando um beacon, chegamos aos resultados mostrados na tabela 01

Figura 4 – Intensidade de Sinal - BLE

Distância real (metros)	Experimento 1 (metros)	Experimento 2 (metros)	Experimento 3 (metros)
5	3,9	5,89	4,78
10	12,88	7,24	9,66
15	23,99	9,77	15,31
20	25,70	16,22	20,41
25	26,30	24,55	25,40
30	35,48	25,41	29,85

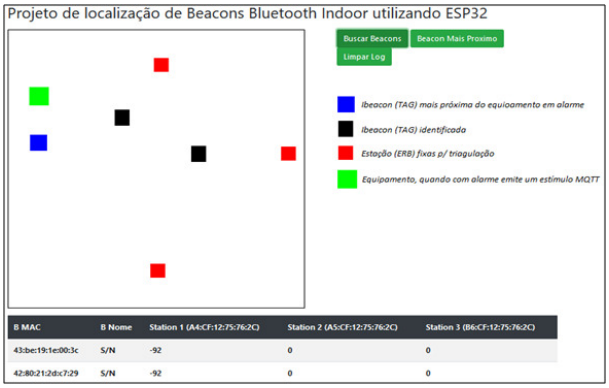
Fonte: Própria

Através dos testes de distâncias foi possível desenvolver o programa para as ERBs.

5.2 Acertos

A tabela foi transformada em uma tela de localização, onde houve o desenvolvimento de um software para localização. Utilizamos a programação web Hypertext Markup Language (HTML) conforme ilustrado na figura 02 para demonstrar de forma intuitiva a tela de lçocalização

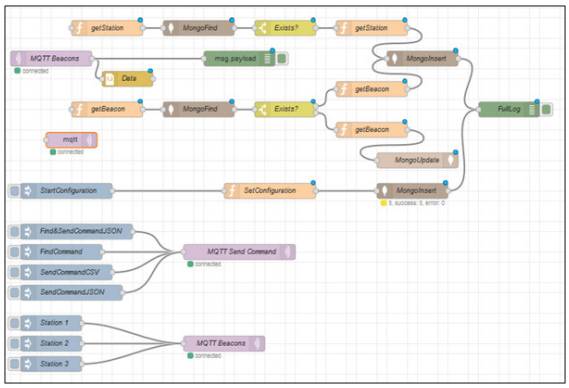
Figura 5 – Layout Software



Fonte: Própria

De acordo com a figura 05 o aplicativo para celular (APP) de alarmes foi modificado por uma programação na aplicação Node-RED da IBM para simulação das ERBs e equipamento com e sem alarmes.

Figura 6 – Aplicação Node-Red

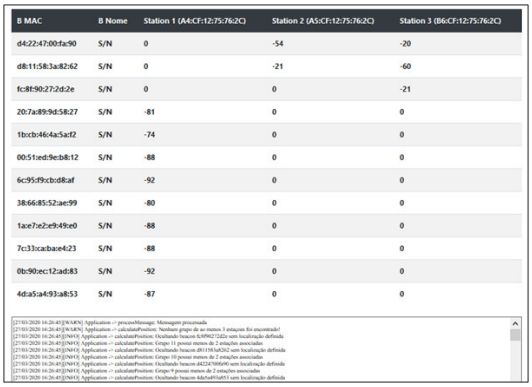


Fonte: Própria

6 Resultados

Os resultados esperados foram alcançados, em destaque o software desenvolvido, onde é possível localizar pessoas em tempo real e monitorar a intensidade do sinal através da tabela de monitoramento conforme a figura 06.

Figura 7 – Software de Monitoramento



B MAC	B Nome	Station 1 (A4:CF:12:75:76:2C)	Station 2 (A5:CF:12:75:76:2C)	Station 3 (B6:CF:12:75:76:2C)
d4:22:47:00:fa:90	S/N	0	-54	-20
d8:11:58:3a:82:62	S/N	0	-21	-60
fc:8f:90:27:2d:2e	S/N	0	0	-21
20:7a:89:9d:58:27	S/N	-81	0	0
1b:cb:46:4a:5a:f2	S/N	-74	0	0
00:51:ed:9e:b8:12	S/N	-88	0	0
6c:95:f9:cb:d8:af	S/N	-92	0	0
38:66:85:52:ae:99	S/N	-80	0	0
1a:e7:e2:e9:49:e0	S/N	-88	0	0
7c:33:ca:ba:e4:23	S/N	-88	0	0
0b:90:ec:12:ad:83	S/N	-92	0	0
4d:a5:a4:93:a8:53	S/N	-87	0	0

Fonte: Própria

7 Conclusão

Em vista que, foi possível demonstrar como funciona e como se monitora pessoas por sistema RSSI, pode se afirmar que foi alcançado o objetivo deste trabalho. Os objetivos foram atingidos de forma, apresentado o principal tipo e modelo de localizador indoor e a sua utilização em sistema de missão crítica. Verificado o principal método de localização indoor, RSSI. Através dos ensaios e testes foi possível notar a importância de um bom programa (software) a manter um nível de confiabilidade de localização, através de algoritmos podendo corrigir erros causados por ruídos. Para trabalhos futuros sugiro que analisem outros métodos de localização com tecnologia de rádio frequência e a viabilidade econômica em manter a localização indoor para grande alcance. Esse trabalho seria de grande valia também, pois demonstra a base do estudo de um sinal RFID, para avaliação da intensidade de sinal em um ambiente indoor. Pode se concluir com este trabalho que através de ensaios e testes com componentes RSSI é possível

garantir a localização de pessoas, diminuindo o risco de falhas e interrupções de funcionalidade em sistemas de missão crítica, garantindo um melhor aproveitamento dos técnicos que atuam para resolver problemas em data centers.

Agradecimentos

Dedicamos este trabalho aos amigos e professores que nos apoiaram para nossa formação acadêmica.

Referências

PENSO, International Consortium for Organizational Resilience. Disponível em: https://www.penso.com.br/ . Acesso em: 16 mar. 2020.

ICOR, Infraestrutura de TI. Disponível em: https://www.build-resilience.org/ . Acesso em: 15 mar. 2020.

LIMA, Leonardo, Como a tecnologia pode ajudar na sua operação logística. Disponível em: http://www.cabtecgti.com.br/blog/ . Acesso em: 10 mar. 2020.

LIMA, E. A. (2001) Sistemas para Localização de Pessoas e Objetos em Ambientes Indoor. Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, RJ, Novembro.

OMEGA, Introdução aos sensores sem fio. Disponível em: https://br.omega.com/ . Acesso em: 16 mar. 2020.

LOUREIRO, A. A. F; NOGUEIRA, J. M. S; RUIZ, L. B; MINI, R. A. F; NAKAMURA, E. F; FIGUEIREDO, C. M. S. (2003) Redes de Sensores Sem Fio. . In: XXI Simpósio Brasileiro de Redes de Computadores (SBRC'03), Anais... Natal, RN, Brasil. Tutorial, p. 179-226.

ZHU, Q. et al. Iot gateway: Bridgingwireless sensor networks into internet of things. In:IEEE. Embedded and Ubiquitous Computing (EUC), 2010 IEEE/IFIP 8th InternationalConference on. [S.l.], 2010. p. 347–352.

Rafael Gomes de Paula nasceu no Brasil, em 07 de julho de 1993. É formado em Técnico de Informatica pelo Colégio Cruzeiro do Sul em 2010 e em Ciência da Computação pela Universidade Cruzeiro do Sul em 2018. Ele trabalha na AXA Seguros do Brasil desde 2018, onde atualmente exerce a função de analista de sistemas. Ele tem muito interesse em aplicações voltadas para o Agronegócio.

Wanderson Thiago da Silva Pagani nasceu no Brasil, em 05 de abril de 1988. É formado em Eletrotécnica pela Escola SENAI "Antônio Devisate"em 2004, em Engenharia Elétrica pela Universidade Anhanguera - CL em 2019 e atualmente cursando Pós-graduação na Escola SENAI "Mariano Ferraz". Ele trabalha na PRODESP Companhia de Processamento de Dados do Estado de São Paulo desde 2013, onde atualmente exerce a função de Supervisor de Manutenção no Sistema de Missão Crítica. Ele tem muito interesse em aplicações voltadas para data centers.

Josimar de Andrade Silva, nascido em São Paulo em 14 de dezembro de 1986. Técnico em Eletroeletrônica formado no Senai “Roberto Simonsen ” em 2003 e Engenharia de Controle e Automação pela Universidade Anhanguera em 2012. Atualmente desempregado e em busca de novas oportunidades, vinha atuando na Hyundai Rotem, empresa coreana fabricante de uma das novas frotas de trens de São Paulo, realizando o comissionamento estático e dinâmico dos trens buscando a validação de segurança e desempenho, para que os mesmos possam entrar em operação nas linhas da CPTM.