

## Formulário de acompanhamento de trabalho de Iniciação Científica

Título do projeto: Accessible Campus

Bolsista (s):

Aluno1: Pedro Pereira Guimarães

Orientador(a): Luma Rissati Borges do Prado

Data: 06/2023

### ATIVIDADES CUMPRIDAS NO PERÍODO

- O código do projeto está no GitHub: <https://github.com/PedroPereiraGuimaraes/IC07>.
- Uma delas foi a integração com o realtime database da Firebase.
- Foram adicionadas funcionalidades na captação de sinal de RSSI pelo módulo ESP8266.
- Exemplo da conexão com o Firebase foi usado para a conexão do ESP com o banco. Link do passo a passo para fazer a conexão: <https://douglasgaspar.wordpress.com/2021/04/23/utilizando-esp8266-nodemcu-com-realtime-database-do-firebase/>.
- Após a conexão foi definido os caminhos para os dados, divididos em 2 grupos, de treinamento e de uso.

O caminho para os dados de treinamento é: training/nome\_do\_local/MAC.

```
for (int i = 0; i < numNetworks; i++) {
    FirebaseJson json;
    float sumRssi = 0;
    int numValues = networks[i].numValues;
    int startIndex = numValues > MAX_RSSI_VALUES ? numValues - MAX_RSSI_VALUES : 0;
    for (int j = startIndex; j < numValues; j++) {
        sumRssi += networks[i].rssiValues[j];
    }
    if ((numValues - startIndex) > 0) {
        networks[i].avgRssi = sumRssi / (numValues - startIndex);
        Serial.print("MAC: " + networks[i].macAddress);
        Serial.print("\nNome da Rede: " + networks[i].bssid + "\nRssi: ");
        Serial.println(networks[i].avgRssi);
    }
    if (Firebase.ready() || sendDataPrevMillis == 0) {
        sendDataPrevMillis = millis();
        String time = timeClient.getFormattedTime();
        String local = "biblioteca";

        String mackey_add = "training/" + String(local) + "/" + networks[i].macAddress + "/mac";
        String mackey_bssid = "training/" + String(local) + "/" + networks[i].macAddress + "/bssid";
        String mackey_rssi = "training/" + String(local) + "/" + networks[i].macAddress + "/rssi";

        Serial.printf("SET MAC. %s\n", Firebase.RTDB.setString(&fbdo, mackey_add.c_str(), networks[i].macAddress) ? "ok" : fbdo.errorReason().c_str());
        Serial.printf("SET BSSID. %s\n", Firebase.RTDB.setString(&fbdo, mackey_bssid.c_str(), networks[i].bssid) ? "ok" : fbdo.errorReason().c_str());
        Serial.printf("SET AVG RSSI. %s\n", Firebase.RTDB.setFloat(&fbdo, mackey_rssi.c_str(), networks[i].avgRssi) ? "ok" : fbdo.errorReason().c_str());
    }
}
```

O caminho para os dados de uso é: networks/ano/mês/dia/hora/MAC.

```
for (int i = 0; i < numNetworks; i++) {
    FirebaseJson json;
    float sumRssi = 0;
    int numValues = networks[i].numValues;
    int startIndex = numValues > MAX_RSSI_VALUES ? numValues - MAX_RSSI_VALUES : 0;
    for (int j = startIndex; j < numValues; j++) {
        sumRssi += networks[i].rssiValues[j];
    }
    if ((numValues - startIndex) > 0) {
        networks[i].avgRssi = sumRssi / (numValues - startIndex);
        Serial.print("MAC: " + networks[i].macAddress);
        Serial.print("\nNome da Rede: " + networks[i].bssid + "\nRssi: ");
        Serial.println(networks[i].avgRssi);
    }
}

if (Firebase.ready() || sendDataPrevMillis == 0) {
    sendDataPrevMillis = millis();
    String time = timeClient.getFormattedTime();

    String mackey_add = "networks/" + String(year) + "/" + String(month) + "/" + String(day) + "/" + String(hour) + "/" + networks[i].macAddress + "/mac";
    String mackey_bssid = "networks/" + String(year) + "/" + String(month) + "/" + String(day) + "/" + String(hour) + "/" + networks[i].macAddress + "/bssid";
    String mackey_rssi = "networks/" + String(year) + "/" + String(month) + "/" + String(day) + "/" + String(hour) + "/" + networks[i].macAddress + "/rssi";
    String mackey_hour = "networks/" + String(year) + "/" + String(month) + "/" + String(day) + "/" + String(hour) + "/" + networks[i].macAddress + "/hour";

    Serial.printf("SET MAC. %s\n", Firebase.RTDB.setString(&fbdo, mackey_add.c_str(), networks[i].macAddress) ? "OK" : fbdo.errorReason().c_str());
    Serial.printf("SET BSSID. %s\n", Firebase.RTDB.setString(&fbdo, mackey_bssid.c_str(), networks[i].bssid) ? "OK" : fbdo.errorReason().c_str());
    Serial.printf("SET AVG RSSI. %s\n", Firebase.RTDB.setFloat(&fbdo, mackey_rssi.c_str(), networks[i].avgRssi) ? "OK" : fbdo.errorReason().c_str());
    Serial.printf("SET TIME. %s\n", Firebase.RTDB.setString(&fbdo, mackey_hour.c_str(), time) ? "OK" : fbdo.errorReason().c_str());
}
}
```

Os dados ficaram separados assim:

Cada endereço MAC dos dados de uso possui seus dados de BSSID(nome da rede), a hora que foi captado o ultimo dado de RSSI, e o último valor de RSSI.

Cada endereço MAC de treinamento possui seus dados de BSSID(nome da rede) e o último valor de RSSI.



Além disso foi feito um código em Python que acessa as informações do Firebase, calcula distância a partir do RSSI, e ordena os valores de menor a maior.

```
1     import firebase_admin
2     from firebase_admin import credentials
3     from firebase_admin import db
4
5     cred = credentials.Certificate("esp8266.json")
6     firebase_admin.initialize_app(cred, {
7         'databaseURL': 'https://esp8266-2dca6-default-rtdb.firebaseio.com/'
8     })
9
10    def rssiParaDistancia(rssi):
11        # Rssi por um metro
12        a = -45
13        # Rssi - Rssi/metro dividido pelo PathLoss
14        w = (rssi - a) / -40
15        # Calculo do Log(distancia)
16        distancia = 10 ** w
17
18        return distancia
19
20    local = input("Local:")
21    ref = db.reference(f'/training/{local}')
22    data = ref.get()
23
24    redes = []
25    for mac, info in data.items():
26        rssi = info.get('rssi')
27        if rssi is not None:
28            redes.append({'mac': info['mac'], 'rssi': rssi})
29
30    redes.sort(key=lambda x: x['rssi'], reverse=True)
31
32    for rede in redes[:5]:
33        print(f"MAC: {rede['mac']}")
34        print(f"RSSI: {rede['rssi']}")
35        print(f"DISTANCE: {round(rssiParaDistancia(rede['rssi']),2)} metros")
36        print()
37
```

Como atividades a serem cumpridas está a criação de um algoritmo de machine learning que defina o local do ESP em relação aos valores dos nós de treinamento.