### Import Libraries

Qweqeqeqe

|  |
| --- |
| #include <BLEDevice.h>  #include <BLEServer.h>  #include <BLEUtils.h>  #include <BLE2902.h>  #include <ArduinoJson.h>  #include <WiFi.h> |

Gfdgfghfhf

### Global Variables

Qweqeqeqe

|  |
| --- |
| BLEServer \*pServer = NULL;  BLECharacteristic \*pSensorCharacteristic = NULL;  BLEService \*pService = NULL; // Declare pService globally  BLECharacteristic \*pWifiScanCharacteristic = NULL;  bool deviceConnected = false;  bool oldDeviceConnected = false; |

BLE

### UUID Definitions

Qweqeqeqe

|  |
| --- |
| #define SERVICE\_UUID "19b10000-e8f2-537e-4f6c-d104768a1214"  #define SENSOR\_CHARACTERISTIC\_UUID "19b10001-e8f2-537e-4f6c-d104768a1214"  #define WIFI\_SCAN\_CHARACTERISTIC\_UUID "19b10003-e8f2-537e-4f6c-d104768a1214" |

BLE

### Callbacks Class for BLE Server

Qweqeqeqe

|  |
| --- |
| class MyServerCallbacks : public BLEServerCallbacks  {  void onConnect(BLEServer \*pServer)  {  deviceConnected = true;  };  void onDisconnect(BLEServer \*pServer)  {  deviceConnected = false;  }  }; |

BLE

### Callbacks Class for BLE Characteristic

Qweqeqeqe

|  |
| --- |
| class MyCharacteristicCallbacks : public BLECharacteristicCallbacks  {  void onWrite(BLECharacteristic \*pCharacteristic)  {  String value = pCharacteristic->getValue().c\_str();  JsonDocument doc; // JSON buffer size  deserializeJson(doc, value);  String ssid = doc["ssid"];  String password = doc["password"];  Serial.print("SSID: ");  Serial.println(ssid);  Serial.print("Password: ");  Serial.println(password);  // Perform actions based on received parameters  }  }; |

BLE

### Setup Function

Qweqeqeqe

|  |
| --- |
| void setup()  {  Serial.begin(115200);  BLEDevice::init("ESP32");  pServer = BLEDevice::createServer();  pServer->setCallbacks(new MyServerCallbacks());  pService = pServer->createService(SERVICE\_UUID);  pSensorCharacteristic = pService->createCharacteristic(  SENSOR\_CHARACTERISTIC\_UUID,  BLECharacteristic::PROPERTY\_READ |  BLECharacteristic::PROPERTY\_WRITE |  BLECharacteristic::PROPERTY\_NOTIFY |  BLECharacteristic::PROPERTY\_INDICATE);  pWifiScanCharacteristic = pService->createCharacteristic(  WIFI\_SCAN\_CHARACTERISTIC\_UUID,  BLECharacteristic::PROPERTY\_READ |  BLECharacteristic::PROPERTY\_NOTIFY);  pWifiScanCharacteristic->addDescriptor(new BLE2902());  pService->start();  BLEAdvertising \*pAdvertising = pServer->getAdvertising();  pAdvertising->addServiceUUID(SERVICE\_UUID);  pAdvertising->setScanResponse(false);  pAdvertising->setMinPreferred(0x0);  BLEDevice::startAdvertising();  Serial.println("Waiting for a client connection to notify...");  } |

BLE

### Loop Function

Qweqeqeqe

|  |
| --- |
| void loop()  {  if (!deviceConnected && oldDeviceConnected)  {  Serial.println("Device disconnected.");  delay(500);  pServer->startAdvertising();  Serial.println("Start advertising");  oldDeviceConnected = deviceConnected;  }  if (deviceConnected && !oldDeviceConnected)  {  oldDeviceConnected = deviceConnected;  Serial.println("Device Connected");  BLECharacteristic \*pCharacteristic = pService->getCharacteristic(WIFI\_SCAN\_CHARACTERISTIC\_UUID);  JsonDocument doc;  int networksFound = WiFi.scanNetworks();  if (networksFound == 0)  {  Serial.println("No networks found");  return;  }  Serial.print("Found ");  Serial.print(networksFound);  Serial.println(" networks");  doc.clear();  for (int i = 0; i < networksFound; ++i)  {  JsonObject wifiInfo = doc.add<JsonObject>();  wifiInfo["ssid"] = WiFi.SSID(i);  wifiInfo["rssi"] = WiFi.RSSI(i);  }  String jsonStr;  serializeJson(doc, jsonStr);  pCharacteristic->setValue(jsonStr.c\_str());  pCharacteristic->notify();  Serial.println("WiFi scan results sent to client");  }  delay(2000);  } |

BLE