## Лабораторная работа 9: Работа с Bluetooth на ESP32

### Цель

* Изучить основы работы с Bluetooth модулем ESP32.
* Научиться подключаться к устройствам по Bluetooth (SPP, BLE).
* Написать программу для передачи данных между ESP32 и мобильным устройством через Bluetooth.
* Создать простое приложение для управления функциями ESP32 через Bluetooth.

### Теоретическая часть

#### Введение в Bluetooth

ESP32 имеет встроенный Bluetooth модуль, который позволяет подключаться к другим устройствам по беспроводной связи. Bluetooth поддерживает различные профили, которые определяют тип соединения и формат данных.

* **SPP (Serial Port Profile)**: используется для последовательной передачи данных, как через обычный COM-порт.
* **BLE (Bluetooth Low Energy)**: энергоэффективный профиль, suitable for low-power devices and applications that require low latency.

#### Подключение к устройствам по Bluetooth

Для подключения к устройствам по Bluetooth необходимо использовать соответствующие библиотеки и функции.

* **SPP:** Библиотека SerialBluetooth.h позволяет подключаться к устройствам, поддерживающим профиль SPP.
* **BLE:** Библиотека BLE2900.h или BLE2902.h (в зависимости от версии ESP32) позволяет подключаться к устройствам, поддерживающим BLE.

### Практическая часть

#### 1. Подключение к Android-устройству по SPP

**Схема подключения:**

* ESP32 и Android-устройство соединены через Bluetooth.

**Код:**

#include <SerialBluetooth.h>  
  
void setup() {  
 Serial.begin(115200);  
  
 // Инициализация Bluetooth  
 SerialBluetooth.begin("ESP32");  
 Serial.println("Waiting for Bluetooth connection...");  
  
 // Ожидание подключения  
 while (!SerialBluetooth.connected()) {  
 delay(500);  
 }  
  
 Serial.println("Connected!");  
}  
  
void loop() {  
 // Проверка наличия данных от Bluetooth  
 if (SerialBluetooth.available()) {  
 String message = SerialBluetooth.readStringUntil('\n');  
 Serial.print("Received: ");  
 Serial.println(message);  
  
 // Отправка ответа на Bluetooth  
 SerialBluetooth.print("Reply: ");  
 SerialBluetooth.println(message);  
 }  
}

**Описание:**

* В коде определено имя Bluetooth-устройства ESP32.
* В функции setup() происходит инициализация Bluetooth и ожидание подключения от Android-устройства.
* В цикле loop() происходит проверка наличия данных от Bluetooth.
* Если данные есть, они читаются и выводятся на Serial Monitor.
* Ответ на полученные данные отправляется на Bluetooth.

**Приложение на Android:**

1. Установите на Android-устройство приложение для работы с Bluetooth (например, Bluetooth Terminal).
2. Подключитесь к Bluetooth-устройству ESP32.
3. Отправляйте сообщения из приложения, они будут отображаться на Serial Monitor ESP32.
4. ESP32 будет отправлять ответы на ваши сообщения.

#### 2. Создание приложения для управления ESP32 по BLE

**Схема подключения:**

* ESP32 и Android-устройство соединены через BLE.

**Код:**

#include <BLE2902.h>  
  
// UUID сервиса  
const char\* serviceUUID = "123e4567-e89b-12d3-a456-426655440000";  
  
// UUID характеристики для чтения данных  
const char\* characteristicUUID\_read = "123e4567-e89b-12d3-a456-426655440001";  
  
// UUID характеристики для записи данных  
const char\* characteristicUUID\_write = "123e4567-e89b-12d3-a456-426655440002";  
  
BLEServer server; // Сервер BLE  
BLECharacteristic\* pCharacteristic\_read; // Характеристика для чтения  
BLECharacteristic\* pCharacteristic\_write; // Характеристика для записи  
  
void setup() {  
 Serial.begin(115200);  
  
 // Инициализация BLE  
 BLE.begin();  
 BLE.name("ESP32 BLE");  
  
 // Создание сервиса BLE  
 BLEService\* pService = server.createService(serviceUUID);  
  
// Создание характеристики для чтения  
pCharacteristic\_read = pService->createCharacteristic(characteristicUUID\_read, BLE\_PROPERTY\_READ | BLE\_PROPERTY\_NOTIFY);  
pCharacteristic\_read->setValue("Initial value");  
pCharacteristic\_read->addNotifyCallback([&](BLERemoteCharacteristic\* pCharacteristic, uint8\_t data[], uint16\_t len) {  
 Serial.print("Read: ");  
 for (int i = 0; i < len; i++) {  
 Serial.print(data[i]);  
 }  
 Serial.println();  
});  
  
// Создание характеристики для записи  
pCharacteristic\_write = pService->createCharacteristic(characteristicUUID\_write, BLE\_PROPERTY\_WRITE);  
pCharacteristic\_write->writeValue("");  
pCharacteristic\_write->writeValueCallback([&](BLERemoteCharacteristic\* pCharacteristic, uint8\_t data[], uint16\_t len) {  
 Serial.print("Write: ");  
 for (int i = 0; i < len; i++) {  
 Serial.print(data[i]);  
 }  
 Serial.println();  
  
 // Обработка полученных данных  
 String message = String((char\*)data);  
 if (message.equals("LED\_ON")) {  
 digitalWrite(ledPin, HIGH);  
 } else if (message.equals("LED\_OFF")) {  
 digitalWrite(ledPin, LOW);  
 }  
});  
  
// Добавление характеристик в сервис  
pService->addCharacteristic(pCharacteristic\_read);  
pService->addCharacteristic(pCharacteristic\_write);  
  
// Добавление сервиса в сервер  
server.addService(pService);  
  
// Запуск сервера BLE  
server.start();  
BLE.advertise();  
  
 Serial.println("BLE Server started.");  
}  
  
void loop() {  
 // ... (остальной код loop() остается неизменным)  
}

**Описание:**

* В коде определены UUID сервиса, характеристик для чтения и записи.
* В функции setup() происходит инициализация BLE, создание Bluetooth-сервера, сервиса BLE, характеристик для чтения и записи.
* Для характеристики чтения устанавливается начальное значение (“Initial value”) и назначается callback-функция, которая будет вызываться при чтении данных из характеристики.
* Для характеристики записи назначается callback-функция, которая будет вызываться при записи данных в характеристику.
* В callback-функции записи происходит обработка полученных данных (в данном случае, включение/выключение светодиода по команде “LED\_ON” или “LED\_OFF”).
* Сервис добавляется в сервер BLE, а сервер запускается.
* Функция BLE.advertise() объявляет о доступности устройства другим устройствам BLE.

**Приложение на Android:**

1. Установите на Android-устройство приложение для работы с BLE (например, nRF Connect for Mobile).
2. Подключитесь к BLE-устройству ESP32.
3. Найдите сервис “123e4567-e89b-12d3-a456-426655440000” и характеристики “123e4567-e89b-12d3-a456-426655440001” и “123e4567-e89b-12d3-a456-426655440002”.
4. Для характеристики “123e4567-e89b-12d3-a456-426655440001” можно посмотреть значение, которое будет обновляться при отправке данных на ESP32.
5. Для характеристики “123e4567-e89b-12d3-a456-426655440002” можно отправлять команды “LED\_ON” и “LED\_OFF” для управления светодиодом на ESP32.

## Задачи:

**Задачи:**

**1. Управление RGB-светодиодом по BLE:**

* Добавьте в код функции управления RGB-светодиодом по BLE.
* Реализуйте возможность отправки команд с Android-устройства для установки цвета (красный, зеленый, синий) светодиода.
* Добавьте возможность отображения текущего цвета светодиода на Android-устройстве.

**2. Обмен данными с датчиком по BLE:**

* Подключите к ESP32 датчик (например, температуры, влажности, освещенности).
* Считывайте данные с датчика.
* Отправляйте данные с датчика на Android-устройство по BLE.
* Отображайте данные с датчика на Android-устройстве в виде графика или таблицы.

## Примеры кода для лабораторной работы :

### 1. Управление RGB-светодиодом по BLE

**Код:**

#include <BLE2902.h>  
  
// ... (остальной код из предыдущего примера)  
  
void setup() {  
 // ... (остальной код setup() из предыдущего примера)  
  
 // Создание характеристики для записи данных RGB  
 pCharacteristic\_rgb = pService->createCharacteristic(characteristicUUID\_rgb, BLE\_PROPERTY\_WRITE);  
 pCharacteristic\_rgb->setValue("");  
 pCharacteristic\_rgb->writeValueCallback([&](BLERemoteCharacteristic\* pCharacteristic, uint8\_t data[], uint16\_t len) {  
 Serial.print("Write RGB: ");  
 for (int i = 0; i < len; i++) {  
 Serial.print(data[i]);  
 }  
 Serial.println();  
  
 // Обработка полученных данных RGB  
 if (len == 3) {  
 int red = data[0];  
 int green = data[1];  
 int blue = data[2];  
  
 // Установка цвета RGB-светодиода  
 leds.setColor(0, leds.Color(red, green, blue));  
 leds.update();  
 }  
 });  
  
 // Добавление характеристики RGB в сервис  
 pService->addCharacteristic(pCharacteristic\_rgb);  
  
 // ... (остальной код setup() из предыдущего примера)  
}  
  
void loop() {  
 // ... (остальной код loop() из предыдущего примера)  
}

**Описание:**

* В коде добавлена константа characteristicUUID\_rgb для UUID характеристики RGB.
* В функции setup() создается характеристика pCharacteristic\_rgb для записи данных RGB.
* В callback-функции записи происходит обработка полученных данных RGB (3 байта: красный, зеленый, синий).
* Значения цветов используются для установки цвета RGB-светодиода с помощью библиотеки FastLED.

### 2. Обмен данными с датчиком по BLE

**Код:**

#include <BLE2902.h>  
#include <Wire.h>  
#include <Adafruit\_BME280.h>  
  
// ... (остальной код из предыдущего примера)  
  
void setup() {  
 // ... (остальной код setup() из предыдущего примера)  
  
 // Инициализация датчика BME280  
 if (bme280.begin(0x76)) {  
 Serial.println("Sensor connected.");  
 } else {  
 Serial.println("Sensor not found.");  
 while (1);  
 }  
  
 // Создание характеристики для чтения данных датчика  
 pCharacteristic\_data = pService->createCharacteristic(characteristicUUID\_data, BLE\_PROPERTY\_READ | BLE\_PROPERTY\_NOTIFY);  
 pCharacteristic\_data->setValue("");  
  
 // Callback-функция для уведомления об изменении данных датчика  
 bme280.setTemperatureCallback([&](float temperature) {  
 updateSensorData();  
 });  
  
 bme280.setHumidityCallback([&](float humidity) {  
 updateSensorData();  
 });  
  
 bme280.setPressureCallback([&](float pressure) {  
 updateSensorData();  
 });  
  
 // ... (остальной код setup() из предыдущего примера)  
}  
  
void loop() {  
 // ... (остальной код loop() из предыдущего примера)  
}  
  
void updateSensorData() {  
 // Считывание данных с датчика  
 float temperature = bme280.readTemperatureC();  
 float humidity = bme280.readHumidity();  
 float pressure = bme280.readPressurePa();  
  
 // Формирование JSON-строки с данными датчика  
 String jsonData = "{\"temperature\": " + String(temperature) + ", \"humidity\": " + String(humidity) + ", \"pressure\": " + String(pressure) + "}";  
  
 // Обновление значения характеристики BLE  
 pCharacteristic\_data->setValue(jsonData.c\_str());  
 // Уведомление подключенных устройств об изменении данных  
 pCharacteristic\_data->notify();  
}

**Описание:**

* В код добавлены библиотеки Wire.h и Adafruit\_BME280 для работы с датчиком BME280.
* В функции setup() происходит инициализация датчика.
* Создается характеристика pCharacteristic\_data для чтения данных датчика.
* Устанавливаются callback-функции для уведомления об изменении температуры, влажности и давления.
* Функция updateSensorData() считывает данные с датчика и формирует JSON-строку с этими данными.
* Значение характеристики pCharacteristic\_data обновляется JSON-строкой.
* Вызывается pCharacteristic\_data->notify(), чтобы уведомить подключенные устройства об изменении данных.

**Приложение на Android:**

* Приложение должно уметь подключаться к BLE-устройству ESP32.
* Приложение должно отображать сервис с характеристикой characteristicUUID\_data.
* Приложение должно читать данные из характеристики и парсить JSON-строку, чтобы получить значения температуры, влажности и давления.
* Приложение должно отображать полученные данные на экране в виде текста или графика.