Politechnika Wrocławska

Wydział Informatyki i Telekomunikacji

Urządzenia peryferyjne

Bluetooth Low Energy

Ćwiczenie nr 14

Prowadzący:

dr inż. Jan Nikodem

Grupa:

Piątek 13:15 TP

Autorzy:

Daniel Glazer, 252743

Paweł Helisz, 252779

Wrocław 26.11.2021

1 Cel ćwiczenia

Celem ćwiczenia było podłączenie ESP32 do komputera w celu zaprogramowania układu działającego w technologii BLE (Bluetooth Low Energy). Układ ESP32 miał wysyłać za pomocą tej technologii zbiór danych do smartfona.

2 BLE

Technologia Bluetooth Low Energy (BLE) znana także jako Bluetooth Smart jest bezprzewodową siecią o zasięgu osobistym (WPAN). Została zaprezentowana w 2010r. aby umożliwić szybszy rozwój aplikacji związanych z terminem IoT (Internet rzeczy). BLE umożliwia wysyłanie serii danych z przerwami między połączeniami rozłożonymi od siebie, tak aby oszczędzać pobór energii.

2.1 Specyfikacja techniczna

W poniższej tabeli porównano technologię BLE z klasycznym Bluetooth:

Specyfikacja techniczna	Bluetooth	Bluetooth Low Energy
Dystans (teoretyczne maksimum)	100 m (330 ft)	<100 m (<330 ft)
Szybkość transmisji danych	1–3 Mbit/s	125 kbit/s – 500 kbit/s – 1 Mbit/s – 2 Mbit/s
Przepustowość aplikacji	0.7–2.1 Mbit/s	0.27-1.37 Mbit/s
Liczba klientów	7	niezdefiniowane, zależne od implementacji
Bezpieczeństwo	56/128-bit, zdefiniowany użytkownik w warstwie aplikacji	128-bit, zdefiniowany użytkownik w warstwie aplikacji
Opóźnienie	100ms	6ms
Minimalny czas na wysłanie danych	0.625ms	3ms
Topologia sieci	Scatternet	Scatternet
Pobór energii	1 W	0.01–0.50 W
Szczytowe zużycie prądu	<30 mA	<15 mA

2.2 Architektura BLE

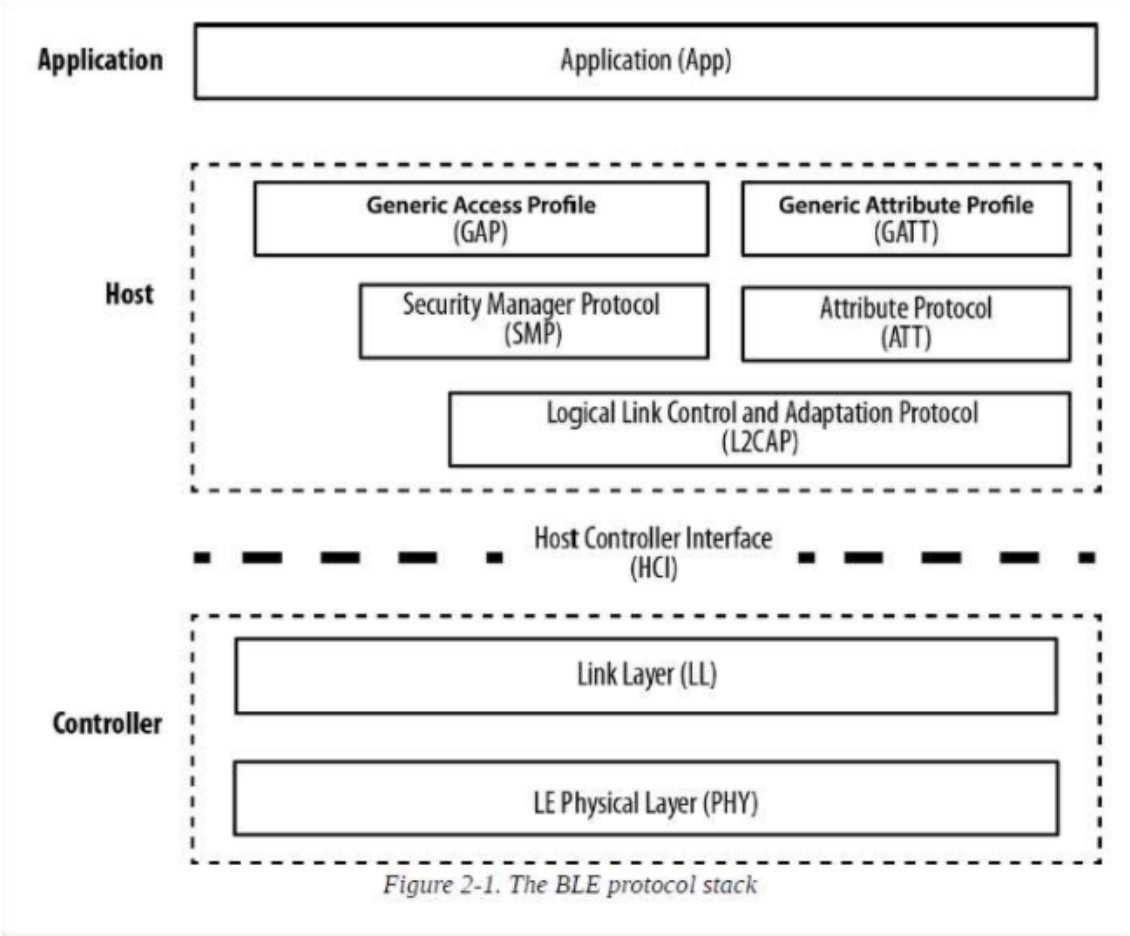


Figure 2-1. The BLE protocol stack

2.3 Warstwa fizyczna (PHY)

Pierwszą warstwą na warstwie protokołów jest warstwa fizyczna. Zawiera analogowe obwody komunikacyjne. Odpowiada za przesyłanie danych drogą radiową. BLE działa w paśmie ISM 2,4 GHz, które jest przemysłowym pasmem naukowym i medycznym. Jest to pasmo wolne od licencji, które jest używane do zastosowań bliskiego zasięgu. Pasmo jest podzielone na 40 kanałów o szerokości pasma 2 MHz, trzy są wykorzystywane przez urządzenia peryferyjne do szukania centrali - rozgłaszanie (advertising), 37 jako kanały danych.

2.4 Warstwa łącza (LL)

Warstwa łącza jest zwykle implementowana jako połączenie sprzętu i oprogramowania. Odpowiada za rozgłaszanie oraz tworzenie i utrzymywanie połączeń. Działa na strukturach pakietów.

2.5 Interfejs kontrolera hosta (HCI)

Warstwa ta umożliwia współdziałanie między hostem a kontrolerem.

2.6 Protokół kontroli i adaptacji łącza logicznego (L2CAP)

L2CAP oznacza logiczną kontrolę łącza i protokół adaptacyjny. Zapewnia usługi enkapsulacji danych dla wyższych warstw. Jest to warstwa odpowiedzialna za integralność danych. Zatem w przypadku, gdy pakiet nie dociera do miejsca docelowego, ta warstwa zapewnia retransmisję.

2.7 Protokół menedżera bezpieczeństwa (SMP)

Warstwa protokołu menedżera bezpieczeństwa zapewnia usługi innym warstwom w dwóch przypadkach: bezpiecznego połączenia i bezpiecznej wymiany danych między dwoma urządzeniami BLE.

2.8 Ogólny profil dostępu (GAP)

Ogólny profil dostępu zapewnia dostęp do operacji warstwy łącza. Obejmuje to określenie roli urządzenia Bluetooth (BLE), obsługi rozgłaszania, nawiązywania połączenia i bezpieczeństwa. GAP definiuje role dla urządzeń. Urządzenie posiada jedną z czterech ról:

1. Broadcaster - nadawca, przekazuje swoje dane poprzez rozgłaszanie
2. Observer - obserwator, jest to rola komplementarna lub przeciwna do nadawcy, obserwuje dostępne urządzenia. Nie nawiązuje dalszych połączeń.
3. Central - centrala, tak jak obserwator skanuje w poszukiwaniu urządzeń, ale może wchodzić w interakcje z urządzeniem i zainicjować proces łączenia. Rola ta określana jest jako Master - może łączyć się z wieloma urządzeniami peryferyjnymi
4. Peripheral - urządzenie peryferyjne, rozgłasza swoją obecność oraz oczekuje na komunikat z cenrali wskazujący na połączenie. Po nawiązaniu połączenia wysyła dane

2.9 Ogólny profil atrybutów (GATT)

GATT zajmuje się wymianą danych w BLE, a zatem jest najwyższą warstwą danych w BLE. Obsługuje urządzenia, które przeszły już proces rozgłaszania, którym zarządza GAP.

2.10 Protokół atrybutów (ATT)

Protokół GATT używa protokołu ATT do przesyłania danych. ATT służy do przechowywania usług, charakterystyk i powiązanych danych, które są nazywane atrybutami. Usługa to zbiór fragmentów danych zwanych charakterystykami. Każda usługa zawiera UUID (uniwersalny unikalny identyfikator), który odróżnia ją od innych usług.

2.11 Warstwa aplikacji

Warstwa aplikacji obsługuje komunikację między aplikacją użytkownika a stosem protokołów BLE.

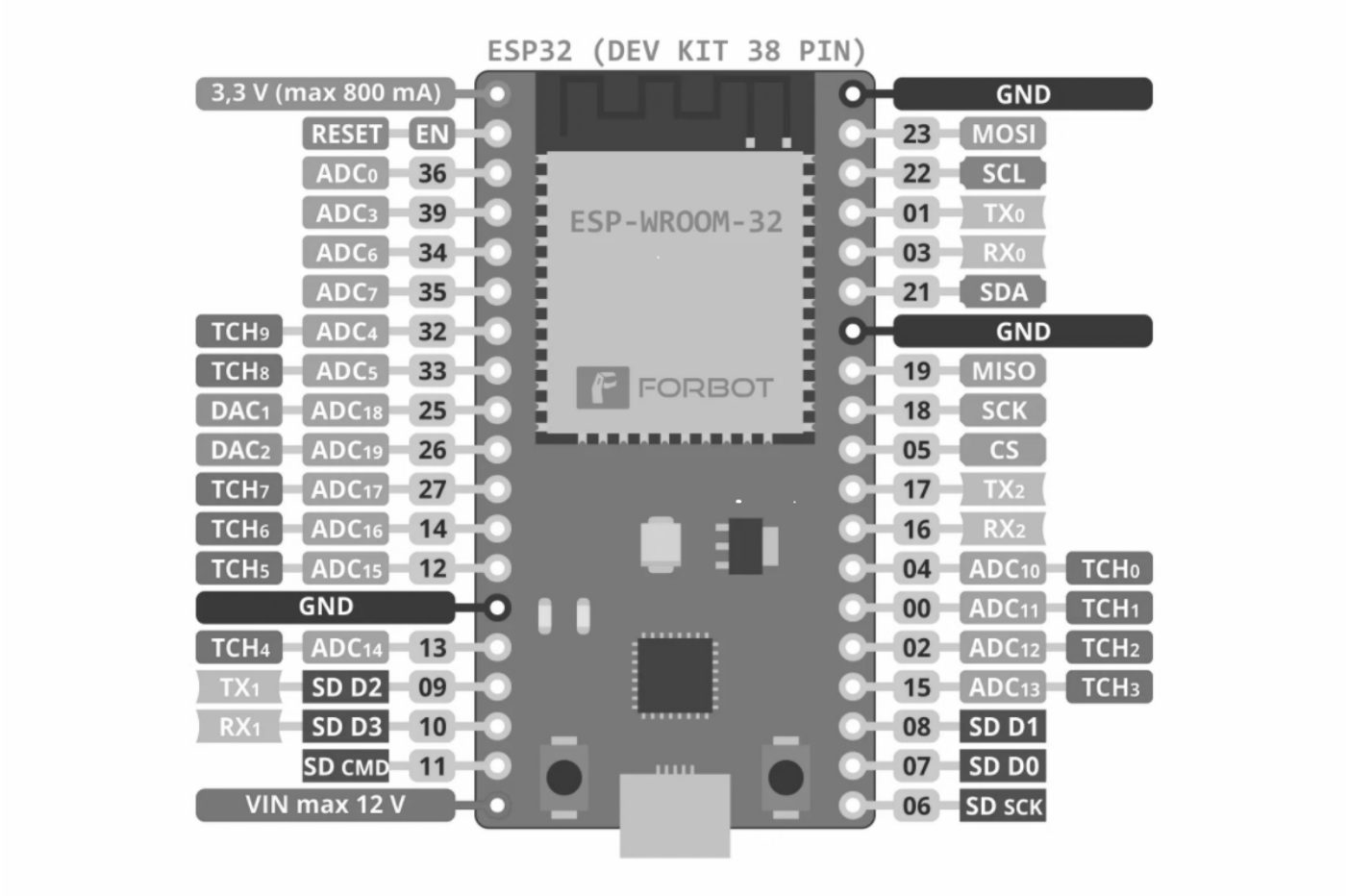
3 ESP32

ESP32 jest układem typu SoC (ang. System-on-a-chip) produkowanym przez chińską firmę Espressif Systems. Układ ten integruje w sobie niezbędne elementy do komunikacji przez WiFi (802.11 b/g/n) oraz przez Bluetooth (klasyczny 4.2 i BLE), dzięki czemu idealnie nadaje się do budowy energooszczędnych urządzeń Internetu Rzeczy (IoT).

3.1 Specyfikacja płytki ESP32 DevKit

1. Komunikacja WiFi:
   (a) standard 802.11 b/g/n 2,4 GHz
   (b) prędkość transmisji do 150 Mb/s
   (c) zabezpieczenia WiFi: WEP, WPA/WPA2, PSK/Enterprise, AES / SHA2 / Elliptical Curve Cryptography / RSA-4096
2. Komunikacja Bluetooth:
   (a) BLE
   (b) Bluetooth Classic 4.2
3. Zasilanie:
   (a) napięcie pracy: 2,3 – 3,6 V
   (b) napięcie zasilania: 4,8 – 12 V
   (c) maksymalny pobór prądu: 800 mA
4. Aktualizacja oprogramowania:
   (a) UART
   (b) OTA
5. CPU:
   (a) Dual Core Tensilica LX6 240 MHz
   (b) obudowa: QFN48-pin (6 mm × 6 mm)
   (c) interfejsy: UART/SDIO/SPI/I2C/I2S,
   (d) dostępne 30 GPIO,
   (e) 12 kanałowy ADC,
   (f) 2 kanałowy DAC.
6. Konwerter USB-TTL (UART)
7. Raster wyprowadzeń: 2,54 mm
8. Wymiary modułu: 55 × 28 mm

3.2 Opis wyprowadzeń modułu ESP32 DevKit:



4 Konfiguracja *Arduino IDE*

Przed pracą na układzie ESP32 należało skonfigurować środowisko, w którym będzie możliwość kompilowania i wgrywania programów na układ. Dlatego w tym celu skonfigurowano program *Arduino IDE*. W preferencjach tego środowiska dodano odnośnik `https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json` do repozytorium zawierającego biblioteki obsługujące płytkę ESP32. Po poprawnym dołączeniu hiperłącza pobrano repozytorium. Następnie w zakładce urządzenia wybrano model płytki - *ESP32 Dev Module* i port, do którego został podłączony układ, który w naszym wypadku był portem COM5.

5 Program w Arduino

5.1 Struktura kodu

Pierwszym etapem pisania kodu na układ ESP32 było zawarcie bibliotek odpowiedzialnych za obsługę *Bluetooth Low Energy* na tym układzie.

```
#include <BLEDevice.h>
#include <BLEServer.h>
#include <BLEUtils.h>
#include <BLE2902.h>
```

Następnie zadeklarowano UUID oraz zmienne, które zostały wysłane za pomocą płytki.

```
#define serviceID BLEUUID((uint16_t)0x1801)
char charArr[]="Tablica znakow 1";
String str = "Tablica znakow 2";
int integer = 3;
float decimalNumber = 4.00000;
```

Kolejnym krokiem było skonfigurowanie serwera BLE, usług jakie ma realizować oraz samego rozgłaszania.

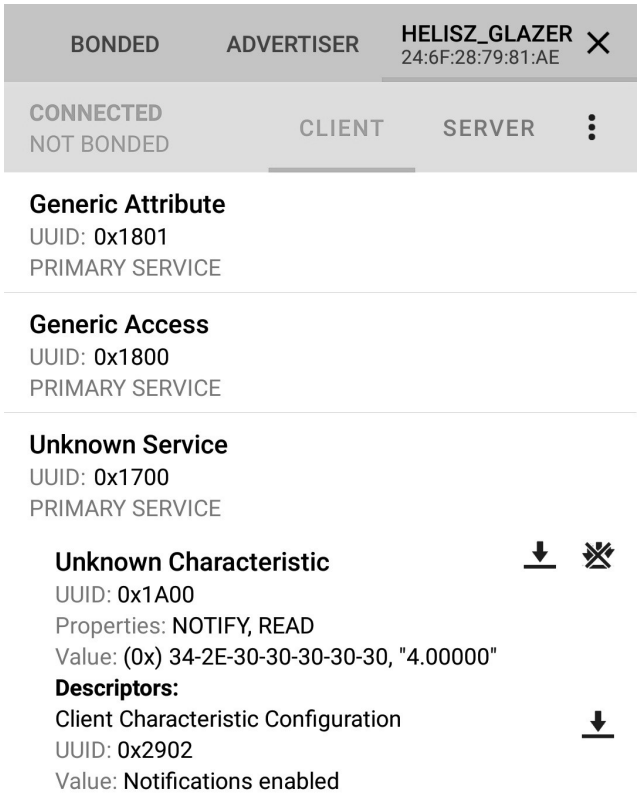
```
//Ustawienie nazwy
BLEDevice::init("Helisz_GLazer");

//Stworzenie serwera BLE
BLEServer *MyServer = BLEDevice::createServer();
MyServer->setCallbacks(new ServerCallbacks());

// Utworzenie uslugi BLE
BLEService *customService = MyServer->createService(serviceID);
// Scharakteryzowanie uslugi
customService->addCharacteristic(&customCharacteristic);
// Utworzenie descryptora
customCharacteristic.addDescriptor(new BLE2902());
// Skonfigurowanie rozglaszania
MyServer->getAdvertising()->addServiceUUID(serviceID);

// Uruchomienie uslugi
customService->start();
// Uruchomienie rozglaszania
MyServer->getAdvertising()->start();
```

Na sam koniec należało zdefiniować w usłudze wartości jakie mają być wysyłane za pomocą powiadomień.

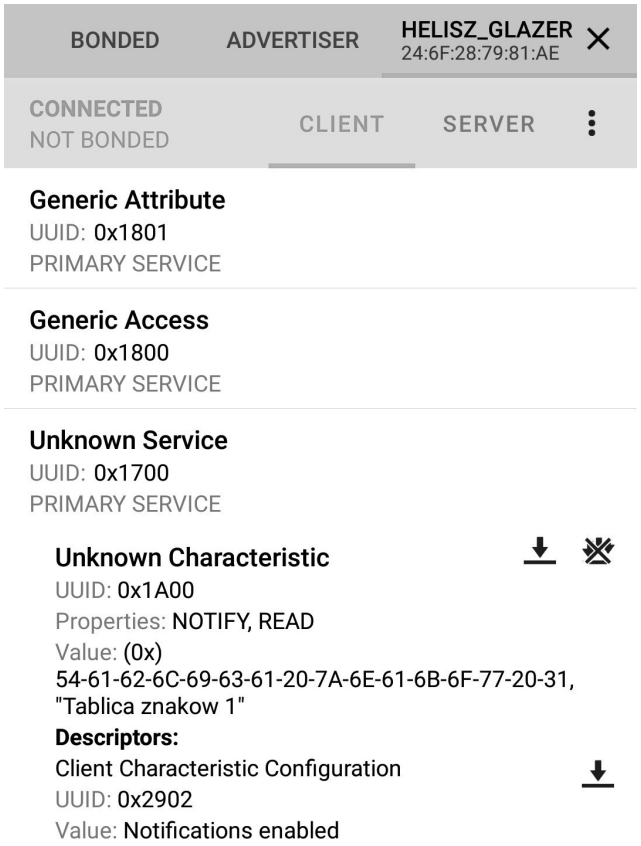
```
//Wysylanie tablicy znakow
    customCharacteristic.setValue((char*)&charArr);
    customCharacteristic.notify();

//Wysylanie liczby zmiennoprzecinkowe
    char buffer[20];
    dtostrf(decimalNumber,1,5,buffer);
    customCharacteristic.setValue((char*)&buffer);
    customCharacteristic.notify();
```

5.2 Przesłanie zbioru

Aby przesłać dane z układu do telefonu przez *BLE* wykorzystano w tym celu aplikacje *nRF Connect* ze sklepu Play. Po połączeniu aplikacji z płytką, telefon odbierał dane jak na załączonych zdjęciach poniżej.



(a) Przesłanie liczby zmiennoprzecinkowej



(b) Przesłanie tablicy znaków