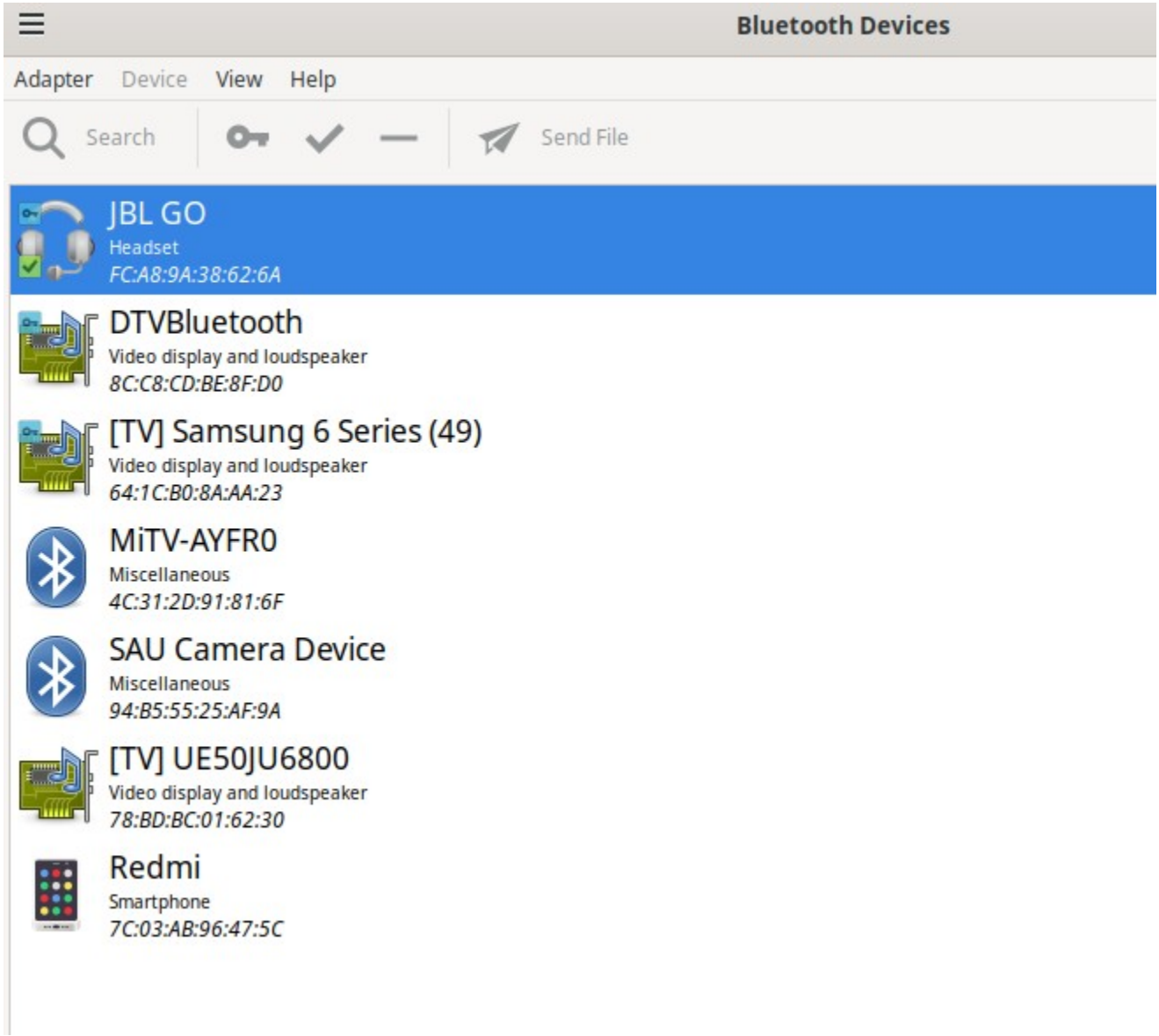
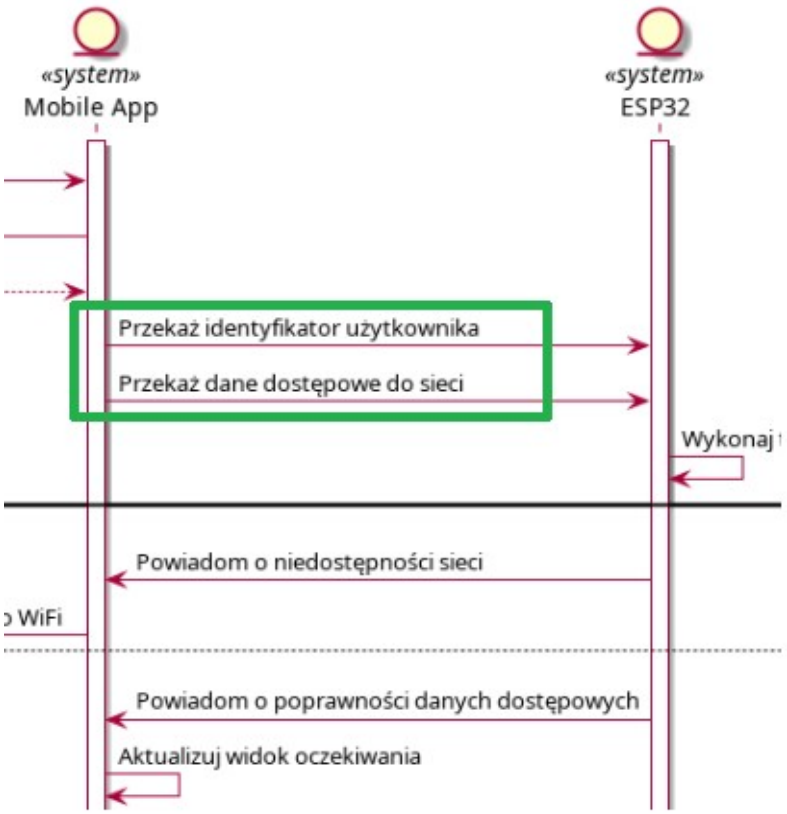
Interfejs komunikacyjny ESP32 (BLE) - Aplikacja Mobilna

ESP32 widoczne jako urządzenie peryferyjne BLE w managerze bluetooth Blueman. Dla tego konkretnego urządzenia widocznego w listingu - Bluetooth MAC = 94:b5:55:25:af:9a (zmienne w zależności od urządzenia). Jako stałą przyjmujemy widoczną nazwę urządzenia - “SAU Camera Device”.



s

O której części komunikacji tutaj mówimy?



s

ESP32 pełni rolę GAP adveritiser, aplikacja mobilna wymaga sparowania z urządzeniem poprzez GAP device discovery, w celu otrzymania pary GAP peripheral (ESP32) + GAP central (aplikacja mobilna).
Jakie usługi są dostępne w ramach profilu GATT? Jakie charakterystyki są dostępne w ramach usług? Co nas interesuje?

```
mundus@pc6:~$ gatttool -b 94:B5:55:25:AF:9A -I
[94:B5:55:25:AF:9A][LE]> connect
Attempting to connect to 94:B5:55:25:AF:9A
Connection successful
[94:B5:55:25:AF:9A][LE]> help
help                                        Show this help
exit                                        Exit interactive mode
quit                                        Exit interactive mode
connect         [address [address type]]    Connect to a remote device
disconnect                                  Disconnect from a remote device
primary         [UUID]                      Primary Service Discovery
included        [start hnd [end hnd]]       Find Included Services
characteristics [start hnd [end hnd [UUID]]] Characteristics Discovery
char-desc       [start hnd] [end hnd]       Characteristics Descriptor Discovery
char-read-hnd   <handle>                    Characteristics Value/Descriptor Read by handle
char-read-uuid  <UUID> [start hnd] [end hnd] Characteristics Value/Descriptor Read by UUID
char-write-req  <handle> <new value>        Characteristic Value Write (Write Request)
char-write-cmd  <handle> <new value>        Characteristic Value Write (No response)
sec-level       [low | medium | high]       Set security level. Default: low
mtu             <value>                     Exchange MTU for GATT/ATT
[94:B5:55:25:AF:9A][LE]> primary
attr handle: 0x0001, end grp handle: 0x0005 uuid: 00001800-0000-1000-8000-00805f9b34fb
attr handle: 0x0006, end grp handle: 0x0009 uuid: 00001801-0000-1000-8000-00805f9b34fb
attr handle: 0x000a, end grp handle: 0xffff uuid: 9c65cb67-faa6-098a-c14d-1211a5051082
[94:B5:55:25:AF:9A][LE]> characteristics
handle: 0x0002, char properties: 0x02, char value handle: 0x0003, uuid: 00002a00-0000-1000-8000-00805f9b34fb
handle: 0x0004, char properties: 0x02, char value handle: 0x0005, uuid: 00002a01-0000-1000-8000-00805f9b34fb
handle: 0x0007, char properties: 0x20, char value handle: 0x0008, uuid: 00002a05-0000-1000-8000-00805f9b34fb
handle: 0x000b, char properties: 0x08, char value handle: 0x000c, uuid: c4cdddf0-bcf3-1a85-2348-bea2aa91c9d8
handle: 0x000d, char properties: 0x08, char value handle: 0x000e, uuid: 4e44eb46-7080-9d83-ac4a-e9f1b82b13d8
handle: 0x000f, char properties: 0x08, char value handle: 0x0010, uuid: b8b40435-be4f-55a5-a447-9c14ddcd20fc
[94:B5:55:25:AF:9A][LE]> char-write-req 0x000c 4d6f6a6557696669
Characteristic value was written successfully
[94:B5:55:25:AF:9A][LE]> char-write-req 0x000e 4d6f6a654861736c6f
Characteristic value was written successfully
[94:B5:55:25:AF:9A][LE]> char-write-req 0x0010 416a616a616a616a616a616a31323334
Characteristic value was written successfully
[94:B5:55:25:AF:9A][LE]> _
```

Z powyższego listingu interesuje nas usługa posiadająca UUID=9c65cb67-faa6-098a-c14d-1211a5051082 (umawiamy się na takie UUID).

Do usługi tej należą trzy interesujące nas charakterystyki o UUID oraz uchwytach wartości charakterystyki równych odpowiednio:

UUID	**char value handle**	**uprawnienia**	**znaczenie w kontekście warstwy aplikacji**
c4cdddf0-bcf3-1a85-2348-bea2aa91c9d8	0x000c	write (ch. prop=0x08)	Nazwa sieci wifi_ssid
4e44eb46-7080-9d83-ac4a-e9f1b82b13d8	0x000e	write (ch. prop=0x08)	Hasło do sieci wifi_psk
b8b40435-be4f-55a5-a447-9c14ddcd20fc	0x0010	write (ch. prop=0x08)	Identyfikator użytkownika user_id

W celu przekazania wifi_ssid, wifi_psk oraz user_id przez interfejs komunikacyjny BLE, konieczne jest więc wykonanie przez aplikację mobilną operacji zapisu na każdej z 3 charakterystyk widocznych w zamieszczonej wcześniej tabeli.

W zamieszczonym wyżej listingu dokonano tego z wykorzystaniem komendy char-write-req z przekazanym uchwytem wartości charakterystyki oraz ciągiem bajtów zapisanym z użyciem notacji szesnastkowej.

Jak w efekcie reaguje na komendy wydane we wcześniejszym listingu software zaimplementowany na ESP32? Tak jak poniżej:

```
I (756791) registration: Preparing to start GAP advertising...
I (756801) registration: Starting GAP advertising...
I (756801) NimBLE: GAP procedure initiated: advertise;
I (756811) NimBLE: disc_mode=2
I (756811) NimBLE:  adv_channel_map=0 own_addr_type=0 adv_filter_policy=0 adv_itvl_min=0 adv_
itvl_max=0
I (756821) NimBLE:

I (756831) registration: Successfully started GAP advertising!
I (823061) registration: In on_ble_gap_adv_start callback.
I (823061) registration: Event --- BLE_GAP_EVENT_CONNECT
I (823061) registration: It's a successful connection
I (956911) registration: In GATT wifi_ssid characteristic access callback stub
I (956911) registration: Successfully written characteristic value
I (956921) registration: Provided wifi_ssid = {MojeWifi}
I (1057401) registration: In GATT wifi_psk characteristic access callback stub
I (1057401) registration: Successfully written characteristic value
I (1057401) registration: Provided wifi_psk = {MojeHaslo}
I (1167651) registration: In GATT user_id characteristic access callback stub
I (1167661) registration: Successfully written characteristic value
I (1167661) registration: Provided user_id = {Ajajajajajaj1234}
```

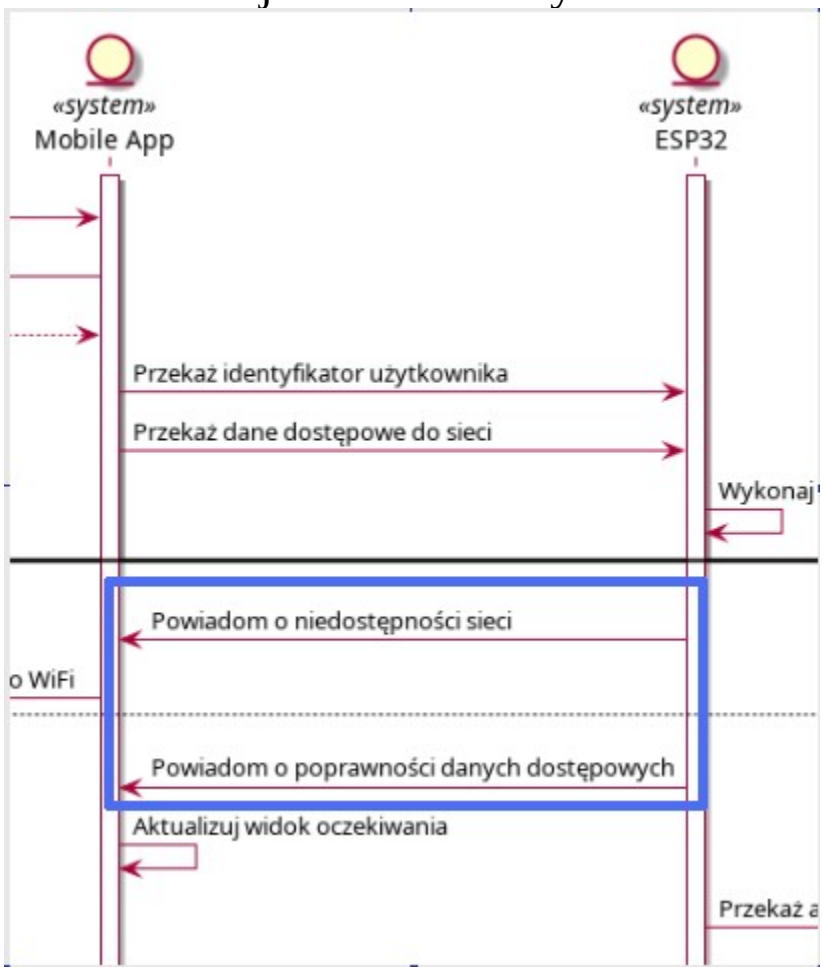
Warto wspomnieć że obowiązują następujące ograniczenia w przekazywanych łańcuchach znaków

```
#define MIN_WIFI_SSID_LENGTH 1
#define MAX_WIFI_SSID_LENGTH 20
#define MIN_WIFI_PSK_LENGTH 1
#define MAX_WIFI_PSK_LENGTH 40
#define MIN_USER_ID_LENGTH 16
#define MAX_USER_ID_LENGTH 16
```

Ciąg dalszy (dotyczący przesyłania powiadomienia o pomyślnym połączeniu się urządzenia lub o błędzie połączenia z siecią WiFi) nastąpi ASAP - planowane wykorzystanie dodatkowej charakterystyki z własnością *Notify* w ramach tej samej usługi

Interfejs komunikacyjny ESP32 (BLE) - Aplikacja Mobilna - c.d.

O której części komunikacji teraz mówimy?



s

Jak zmieniła się lista charakterystyk dla usługi z UUID= 9c65cb67-faa6-098a-c14d-1211a5051082 ?

Odpowiedź: W ramach tej usługi pojawiła się dodatkowa charakterystyka o UUID=dd0a6b3f-86d8-68a9-4141-3ed99c4c2ac7

```
[3C:E9:0E:86:77:16][LE]> primary
attr handle: 0x0001, end grp handle: 0x0005 uuid: 00001800-0000-1000-8000-00805f9b34fb
attr handle: 0x0006, end grp handle: 0x0009 uuid: 00001801-0000-1000-8000-00805f9b34fb
attr handle: 0x000a, end grp handle: 0xffff uuid: 9c65cb67-faa6-098a-c14d-1211a5051082
[3C:E9:0E:86:77:16][LE]> characteristics  0x000a
handle: 0x000b, char properties: 0x08, char value handle: 0x000c, uuid: c4cdddf0-bcf3-1a85-2348-bea2aa91c9d8
handle: 0x000d, char properties: 0x08, char value handle: 0x000e, uuid: 4e44eb46-7080-9d83-ac4a-e9f1b82b13d8
handle: 0x000f, char properties: 0x08, char value handle: 0x0010, uuid: b8b40435-be4f-55a5-a447-9c14ddcd20fc
handle: 0x0011, char properties: 0x10, char value handle: 0x0012, uuid: dd0a6b3f-86d8-68a9-4141-3ed99c4c2ac7
[3C:E9:0E:86:77:16][LE]> _
```

Zaktualizowana tabela z charakterystykami:

UUID	char value handle	Uprawnienia/char props	Znaczenie w kontekście warstwy aplikacji
c4cdddf0-bcf3-1a85-2348-bea2aa91c9d8	0x000c	write (ch. prop=0x08)	Nazwa sieci wifi_ssid
4e44eb46-7080-9d83-ac4a-e9f1b82b13d8	0x000e	write (ch. prop=0x08)	Hasło do sieci wifi_psk
b8b40435-be4f-55a5-a447-9c14ddcd20fc	0x0010	write (ch. prop=0x08)	Identyfikator użytkownika user_id
dd0a6b3f-86d8-68a9-4141-3ed99c4c2ac7	***0x0012***	***notify (ch. prop=0x10)***	***Stan połączenia z siecią WiFi network_state***

s

Wartość charakterystyki z UUID=dd0a6b3f-86d8-68a9-4141-3ed99c4c2ac7 to NETWORK_STATE_WIFI_DISCONNECTED = 0x00 lub NETWORK_STATE_WIFI_CONNECTED=0x01

```
#define NETWORK_STATE_WIFI_DISCONNECTED 0U
#define NETWORK_STATE_WIFI_CONNECTED 1U
```

```
.....
[3C:E9:0E:86:77:16][LE]> char-write-req 0x0010 626262626262626261616161616161
Characteristic value was written successfully
.....
[Upłynięcie czasu, w którym ESP32 sprawdza połączenie z siecią].....
.....
Notification handle = 0x0012 value: 01
```

Jeśli otrzymano w powiadomieniu wartość 01 (tak jak w powyższym przykładowym listingu), to przyjmujemy, że ESP32 pomyślnie połączyło się z siecią, zatem wifi_ssid oraz wifi_psk były prawidłowe i prawdopodobnie w aktywnym połączeniu aplikacji z serwerem, lada moment pojawi się pakiet TCP z serwera informujący o pomyślnym zarajestrowaniu urządzenia. Jeśli jednak otrzymano 00, to należy ponownie poprosić użytkownika o wprowadzenie danych / reaktywować przycisk zatwierdzania danych w aplikacji w celu wykonania ponownego testu połączenia z siecią przez ESP32.