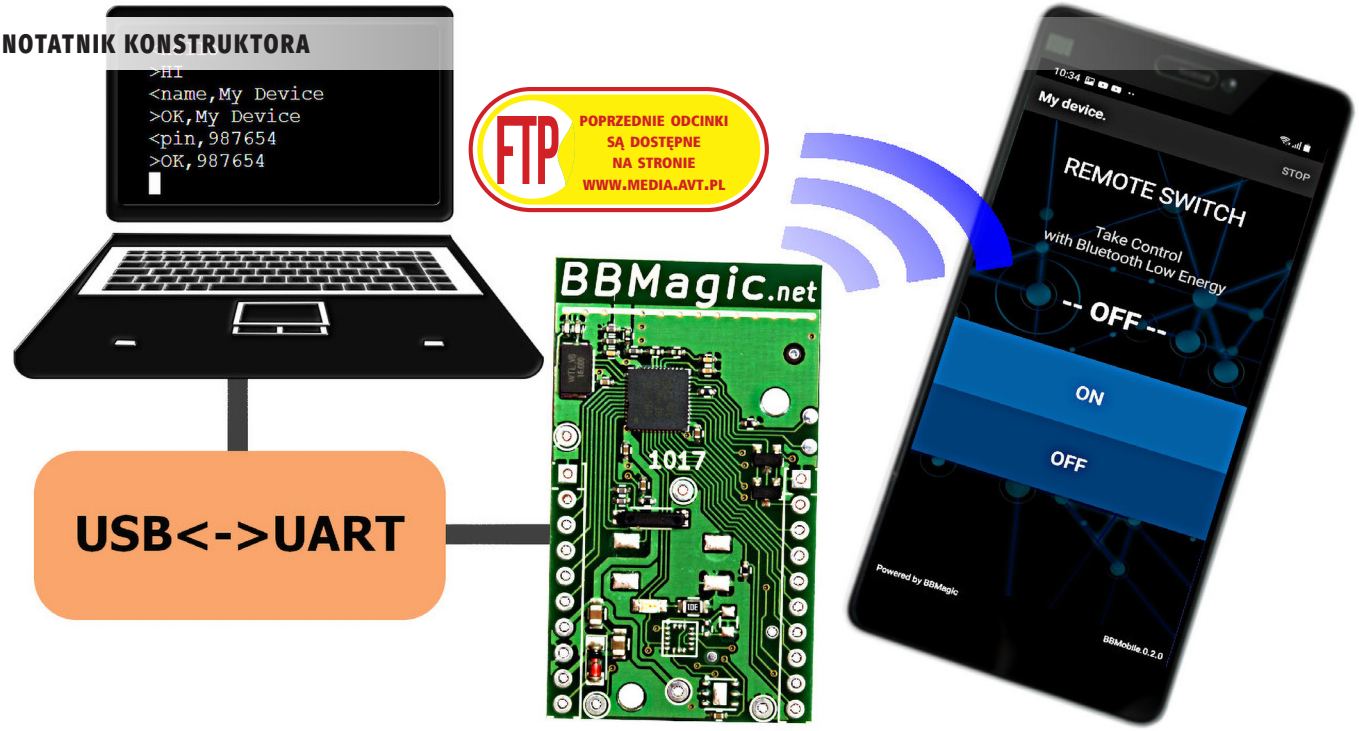


Jak napisać aplikację mobilną poprzez UART (3)

W trakcie procesu tworzenia urządzeń elektronicznych projektanci napotykają szereg, zdawałoby się, nierozwiązywalnych problemów, z którymi muszą sobie sprawnie poradzić. Wracającym niezwykle często, niczym bumerang, kłopotem jest zorganizowanie interfejsu użytkownika HMI (Human Machine Interface). Wymagania są niejednokrotnie sprzeczne: wysoki poziom estetyki, ergonomia użytkowania i elastyczność funkcjonalna w opozycji do minimalnych kosztów oraz krótkiego czasu wdrożenia. Skłania to do poszukiwania nowych rozwiązań będących optymalnym kompromisem rozcinającym ten gordyjski węzeł.

W drugiej części artykułu uruchomiliśmy komunikację z modułem BBMobile i poznaliśmy komendy, które pozwalają sterować jego pracą. Pora na szczegółowe przedstawienie kontrolek oraz omówienie procesu projektowania i testowania aplikacji mobilnych budowanych poprzez UART. Zrobimy to, przechodząc pełny proces tworzenia projektu o roboczej nazwie REMOTE SWITCH. Finalnie będzie on umożliwiał bezprzewodowe sterowanie dowolnym urządzeniem elektrycznym z dowolnego urządzenia mobilnego z wykorzystaniem z technologii BLE.

Kontrolki

Kontrolki są podstawowym budulcem aplikacji. Po wyświetleniu na ekranie interfejsu przesłanego w postaci kodu JSON pozwalają prezentować informacje przesłane do portu UART oraz przyjmować i przekazywać do mikrokontrolera dane wprowadzane przez użytkownika. Podstawowe typy kontrolek wraz z ich opisem i sposobem definiowania zestawiono w **tabeli 1**. **Tabele 2** i **3** zawierają natomiast cechy kontrolek, których użyjemy w tworzonym projekcie. Wszystkie dostępne kontrolki opisane są szczegółowo w dokumentacji modułu BBMobile dostępnej w Internecie.

Tabela 1. Podstawowe typy kontrolek budujących interfejs aplikacji

Kontrolka	Opis	Definicja JSON
Button	Przycisk monostabilny	„ty”: „Button”
TogButton	Przełącznik dwustanowy	„ty”: „TogButton”
TextView	Kontrolka wyświetlająca tekst	„ty”: „TextView”
EditText	Kontrolka służąca wprowadzaniu danych liczbowych lub alfanumerycznych	„ty”: „EditText”
Spinner	Pole wyboru wielokrotnego	„ty”: „Spinner”
ProgBar	Pasek postępu	„ty”: „ProgBar”

Stwórzmy teraz opis (w języku JSON) wszystkich kontrolek, które budować będą interfejs użytkownika projektowanej aplikacji 'REMOTE SWITCH'. Skorzystanie z zaprezentowanych tabel 2 i 3 bardzo ułatwi to zadanie. Zacznijmy od kontrolki będącej tytułem aplikacji. Kod JSON został pokazany na **listingu 1**. To nieedytowalne pole tekstowe – kontrolka typu TextView – co definiuje linia 2. Ponieważ w trakcie działania aplikacji nie zamierzamy zmieniać żadnego z parametrów tej kontrolki, pominiemy nadawanie jej nazwy, aby skrócić finalną długość kodu. W linii 3 zdefiniowany został tytułowy tekst aplikacji, a linia 4 określa składowe RGB jego koloru – biały. Przed tekstem ‘REMOTE SWITCH’ wstawiona została na ekranie dodatkowa

Listing 1. Kod opisujący kontrolkę typu TextView będącej tytułem aplikacji

```
1  {
2  „ty”:”TextView”,
3  „te”:”\nREMOTE SWITCH”,
4  „tc”:”255,255,255”,
5  „ts”:”30”,
6  „w”:”1”
7  }
```

Tabela 2. Cechy kontrolki typu przycisk monostabilny – Button

Kontrolka: Button		„ty”:”Button”
NAZWA POLA	**MNEMONIK**	**OPIS**
Type	ty	Określa typ kontrolki. W tym przypadku jest to ‘Button’ – przycisk jednostanowy
Name	n	Umożliwia nadanie unikalnej nazwy kontrolce służącej do jej identyfikacji w aplikacji, np. ”b1”
Text	te	Ustawia wyświetlany na guziku tekst, np. ”OK”
Text Color	tc	Definiuje składowe RGB koloru tekstu, np. ”255.255.255” określa kolor biały
Text Size	ts	Określa rozmiar tekstu, np. ”30”
Text Style	tl	Ustawia styl wyświetlanego tekstu, np. ”bold”, ”italic”, ”bolditalic”
Background	bg	Definiuje składowe RGB koloru tła kontrolki, np. ”255.0.0” określa kolor czerwony
Weight	w	Definiuje wielkość kontrolki jako ułamkową część całości widoku, gdzie licznik ułamka to wartość parametru ‘w’, a mianownik to suma wartości parametrów ‘w’ wszystkich kontrolek znajdujących się w layoucie
EnabLED	en	Wartość 0 – kontrolka nieaktywna (nie reaguje na dotknięcie), wartość 1 – kontrolka aktywna

Tabela 3. Cechy kontrolki nieedytowalnego pola tekstowego – TextView

Kontrolka: TextView		„ty”:”TextView”
NAZWA POLA	**MNEMONIK**	**OPIS**
Type	ty	Określa typ kontrolki. W tym przypadku jest to ‘TextView’ – nieedytowalne pole tekstowe
Name	n	Umożliwia nadanie unikalnej nazwy kontrolce służącej do jej identyfikacji w aplikacji, np. ”t1”
Text	te	Ustawia wyświetlany w polu tekst, np. ”Hello World”
Text Color	tc	Definiuje składowe RGB koloru tekstu, np. ”255.255.255” określa kolor biały
Text Size	ts	Określa rozmiar tekstu, np. ”30”
Text Style	tl	Ustawia styl wyświetlanego tekstu, np. ”bold”, ”italic”, ”bolditalic”
Background	bg	Definiuje składowe RGB koloru tła kontrolki, np. ”255.0.0” określa kolor czerwony
Weight	w	Definiuje wielkość kontrolki jako ułamkową część całości widoku, gdzie licznik ułamka to wartość parametru ‘w’, a mianownik to suma wartości parametrów ‘w’ wszystkich kontrolek znajdujących się w layoucie

pusta linia (‘\n’) dla osiągnięcia lepszego efektu wizualnego. Linia 5 określa rozmiar wyświetlanego tekstu, natomiast kolejna rozmiar całego obiektu TextView jako ułamek wielkości całego interfejsu. Ponieważ kontrolka jest strukturą JSON, musi rozpoczynać ją i kończyć nawias: linie 1 i 7.

Kod drugiej kontrolki TextView, będącej podtytułem aplikacji pokazano na **listingu 2**. Jej budowa jest identyczna jak tej już omówionej. Trzecia kontrolka TextView, opisana **listingiem 3**, ma nowe cechy: w linii 3 zdefiniowano jej nazwę, aby możliwe było modyfikowanie cech podczas działania aplikacji. Ustawiono też styl wyświetlanego tekstu jako pogrubiony (linia 7). **Listing 4** zawiera najkrótszą możliwą definicję kontrolki TextView z określeniem jedynie jej wielkości. Będzie przydatna do wypełnienia dolnej części ekranu interfejsu.

Ostatnie dwie kontrolki opisane są kolejno **listingami 5 i 6**. To kolorowe obiekty typu ‘Button’ (b1 i b2) z białymi napisami odpowiednio ‘ON’ i ‘OFF’.

Jak poukładać kontrolki na ekranie

Projektowanie interfejsu użytkownika można wyobrazić sobie jako umieszczanie mniejszych pudełek w większych, a tych w jeszcze większych i na końcu wszystkiego tego w największym pudle, które mieści cały widok interfejsu. Tymi pudełkami są layouty.

Layout jest strukturą opisującą sposób ułożenia elementów znajdujących się w jej wnętrzu. Mogą to być zarówno kontrolki

Listing 2. Kod opisujący kontrolkę typu TextView będącej podtytułem aplikacji

```
1  {
2  „ty”:”TextView”,
3  „te”:”Kontroluj zdalnie\n z Bluetooth Low Energy”,
4  „tc”:”174,174,174”,
5  „ts”:”20”,
6  „w”:”1”
7  }
```

Listing 3. Kod opisujący kontrolkę typu TextView z możliwością modyfikacji treści

```
1  {
2  „ty”:”TextView”,
3  „n”:”t1”,
4  „te”:”-- OFF --”,
5  „tc”:”255,255,255”,
6  „ts”:”35”,
7  „tl”:”bold”,
8  „w”:”1”
9  }
```

Listing 4. Najkrótsza możliwa definicja kontrolki TextView z określeniem jedynie jej wielkości

```
1  {
2  „ty”:”TextView”,
3  „w”:”3”
4  }
```

Listing 5. Kod opisujący kontrolkę typu Button – „ON”

```
1  {
2  „ty”:”Button”,
3  „n”:”b1”,
4  „te”:”ON”,
5  „tc”:”255,255,255”,
6  „bg”:”3,96,164”,
7  „ts”:”25”
8  }
```

Listing 6. Kod opisujący kontrolkę typu Button – „OFF”

```
1  {
2  „ty”:”Button”,
3  „n”:”b2”,
4  „te”:”OFF”,
5  „tc”:”255,255,255”,
6  „bg”:”0,62,120”,
7  „ts”:”25”
8  }
```

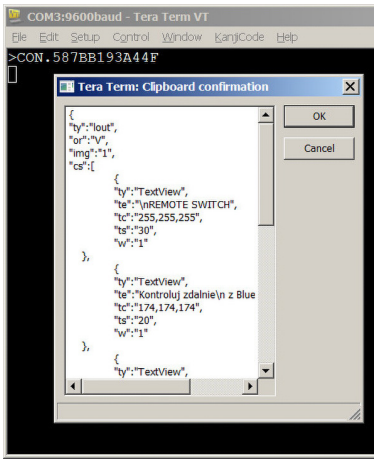
Listing 7. Kod opisujący prosty interfejs użytkownika

```
1  {
2  „ty”: „lout”,
3  „or”:”v”,
4  „bg”:”255,255,255”,
5  „cs”:[{O_1}, {O_2}, {O_3}]
6  }
```

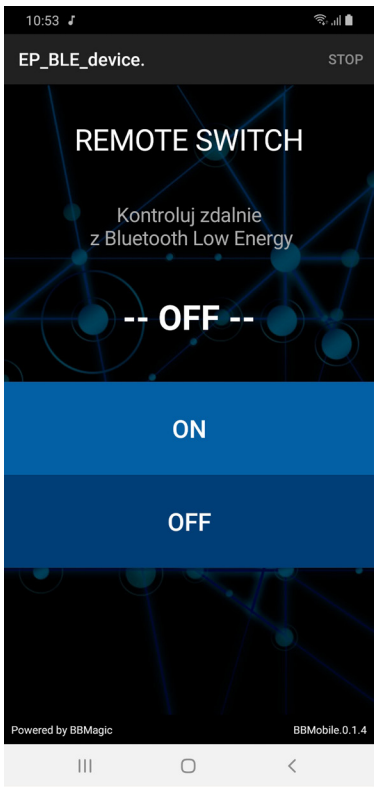
(podstawowe elementy funkcjonalne interfejsu), jak też inne layouty (zawierające np. kontrolki w innym układzie – horyzontalnym lub wertykalnym).

Schemat kodu JSON opisujący prosty interfejs użytkownika pokazano na **listingu 7**. Jest to uniwersalny schemat, na bazie którego budowane są wszystkie aplikacje. Linie 1 i 6 obejmują nawiasami cały opis interfejsu użytkownika. W linii 2 zdefiniowano typ obiektu jako 'layout'. Linia 3 definiuje, w jaki sposób zorganizowane będzie jego wnętrze. Wybrać można jeden z dwóch wariantów: ułożenie wertykalne ("or":"v") lub horyzontalne ("or":"h"). Linia 4 definiuje składowe RGB koloru tła layoutu, będzie on biały. Jako tło można też użyć jednej z siedmiu wbudowanych grafik ponumerowanych od 1 do 7. W takim przypadku linia 4 zmieni swoją postać np. na „img":"1" Kolejna, linia 5 jest tablicą definiującą zawartość layoutu czyli całego interfejsu. W tym przykładzie tablica komponentów ("cs" – components) zawiera symbolicznie przedstawione trzy obiekty: O_1, O_2 i O_3.

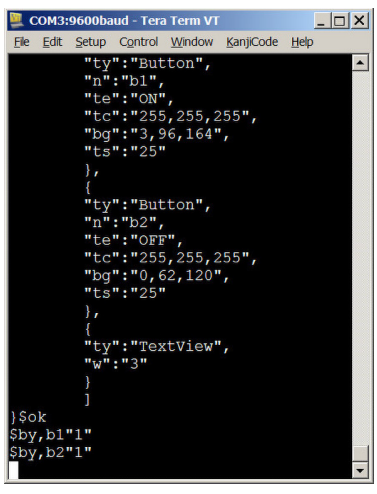
Mając powyższą wiedzę, wracamy teraz do projektowanej aplikacji 'REMOTE SWITCH'. Wszystkie zdefiniowane uprzednio kontrolki ułożymy na ekranie jedna pod drugą. Użyjemy zatem layoutu wertykalnego z **listingu 8** i zobaczymy, jaki otrzymamy efekt końcowy. Naciskamy 'START' w prawym górnym rogu aplikacji BBMobile i z listy urządzeń wybieramy 'EP_BLE_device'. Na ekranie terminalu pojawi się informacja o zestawionym połączeniu. Kopiujemy teraz przygotowany kod z listingu 8 (Ctrl+C), a następnie klikamy prawym przyciskiem myszy w oknie terminalu i potwierdzamy wysłanie danych przyciskiem 'OK' (**rysunek 1**).



Rysunek 1. Wysłanie kodu za pomocą terminalu



Rysunek 2. Widok interfejsu użytkownika



Rysunek 3. Komunikaty wysyłane przez interfejs po dotknięciu kontrolek Button

Listing 8. Kod opisujący layout aplikacji

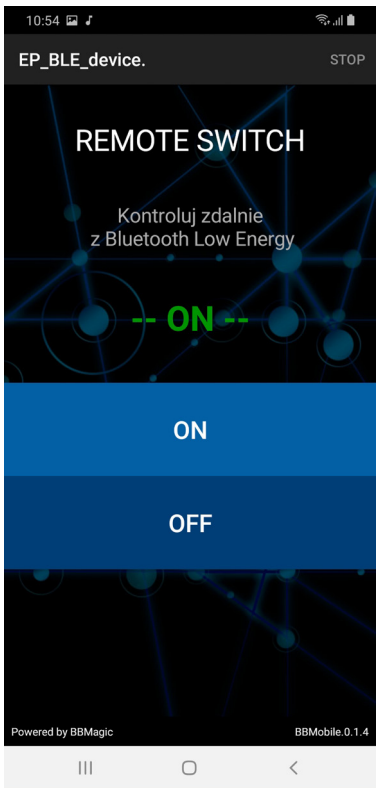
```
{
„ty":"lout",
„or":"V",
„img":"1",
„cs":[
    {
    „ty":"TextView",
    „te":"REMOTE SWITCH",
    „tc":"255,255,255",
    „ts":"30",
    „w":"1"
    },
    {
    „ty":"TextView",
    „te":"Kontroluj zdalnie\n z Bluetooth Low Energy",
    „tc":"174,174,174",
    „ts":"20",
    „w":"1"
    },
    {
    „ty":"TextView",
    „n":"t1",
    „te":"-- OFF --",
    „tc":"255,255,255",
    „ts":"35",
    „tl":"bold",
    „w":"1"
    },
    {
    „ty":"Button",
    „n":"b1",
    „te":"ON",
    „tc":"255,255,255",
    „bg":"3,96,164",
    „ts":"25"
    },
    {
    „ty":"Button",
    „n":"b2",
    „te":"OFF",
    „tc":"255,255,255",
    „bg":"0,62,120",
    „ts":"25"
    },
    {
    „ty":"TextView",
    „w":"3"
    }
    ]
}
```

Kod JSON został właśnie wysłany przez UART do modułu i dalej do aplikacji BBMobile. Jeśli był poprawnie skopiowany i wklejony, to w odpowiedzi otrzymujemy krótkie '$ok', a na ekranie smartfona ukazuje się długo wyczekiwany widok – nasz interfejs (**rysunek 2**).

Kontrolowanie aplikacji

Interfejs jest gotowy. Do pełni szczęścia pozostało jeszcze opanowanie komunikacji z powstałym graficznym interfejsem. Jest ona tak prosta i intuicyjna, że każdy, kto choć przez chwilę trzymał palce na klawiaturze, zrozumie w mig jej meandry. Dotknijmy button z napisem 'ON' (nazwanego 'b1'), a następnie z napisem 'OFF' (nazwanego 'b2'). Na ekranie terminalu pojawią się kolejne komunikaty, jak pokazuje **rysunek 3**.

Konwencja komunikatu zdarzeniowego jest następująca: najpierw znaki '$by,', a następnie nazwa kontrolki, która została uruchomiona i na koniec w cudzysłowie wartość obrazująca jej stan. Ponieważ zdefiniowana przez nas



Rysunek 4. Efekt zmiany parametru 'tc'

1	2	3	4	5	6	7	8	9	10
$set,	t1	:	te	=	"-- ON –"	tc	=	"0,150,0"	'\r\n'

Rysunek 5. Konwencja poleceń modyfikujących parametry kontrolek

kontrolka button sygnalizuje jedynie 'tapnięcie' w ekran, wartość ta jest zawsze równa 1.

Wyślijmy teraz do interfejsu komendy:

```
$set,t1:te="-- ON –"
$set,t1:tc="0,150,0"
```

Możemy wpisać je bezpośrednio z klawiatury i zatwierdzić enterem lub wkleić tak, jak poprzednio zrobiliśmy to z kodem JSON. Pierwsza komenda zmienia wartość parametru 'te' (wyświetlany tekst) w kontrolce, którą nazwaliśmy 't1'. Komenda druga zmienia w tej samej kontrolce wartość parametru 'tc' odpowiedzialnego za kolor tekstu. Efekt ich zadziałania pokazuje **rysunek 4**. Powyższych zmian w interfejsie możemy również dokonać za pomocą jednej komendy:

```
$set,t1:te="-- ON –"tc="0,150,0"
```

Zastosowaną konwencję poleceń modyfikujących parametry kontrolek podsumowano graficznie na **rysunku 5**. Jako pierwszy mamy ciąg znaków '$set,' (pole 1), a następnie nazwę kontrolki, której parametry zamierzamy zmienić (pole 2). Nazwę tę nadaliśmy jej w definicji zawartej w kodzie JSON (listing 8). Dalej dwukropek (pole 3) i nazwę pierwszej modyfikowanej cechy odnalezionej w tabeli 4 (pole 4). Dalej znak przyrównania (pole 5) oraz w cudzysłowie nową wartość modyfikowanej cechy (pole 6). W tym miejscu polecenie mogłoby zostać zakończone znakami "\r\n", ale możliwe jest również kontynuowanie komendy. Mamy zatem nazwę kolejnej modyfikowanej cechy w tej samej kontrolce 't1' (pole 7), znak przyrównania (pole 8) i nową wartość (pole 9). Każda komenda powinna kończyć się znakami powrotu karetki i nowej linii (pole 10).

Możliwe jest dalsze rozbudowywanie polecenia poprzez dodanie po przecinku nazwy kolejnej kontrolki wraz z cechami do modyfikacji. Przykładowe polecenie, które modyfikuje cztery cechy (te, tc, ts, i bg) trzech różnych kontrolek: t1, b1 i b2,wygląda następująco:

```
$set,t1:te="-- ON –"tc="0,150,0",b1:ts="30",b2
:bg="100,100,100"
```

Podsumowanie

Interfejs mamy już zaprojektowany, a komunikację przetestowaną. Pozostaje zrobić ten ostatni krok: zaszyć przesyłanie struktury JSON i komunikatów sterujących w małym mikrokontrolerze. Tym zajmiemy się w kolejnej części publikacji.

Mariusz Żądło
iram@poczta.onet.pl

Płytka rozwojowa ESP32-DEVKITC-V4

Dzięki uprzejmości firmy Soyter Components, mamy dla czytelników EP płytkę rozwojową ESP32-DEVKITC-V4 do zastosowań w IoT z modułem Wi-Fi typu ESP-WROOM-32.

Płytka rozwojowa pozwala na wykorzystanie możliwości układu ESP32 firmy Espressif, który umożliwia łatwą realizację komunikacji poprzez Wi-Fi oraz Bluetooth BLE. Większość linii GPIO została doprowadzona do pinów o rozstawie 2,54 mm, a dodatkowo na płytce znajduje się konwerter USB/UART, który umożliwia programowanie modułu np. w środowisku Arduino IDE.

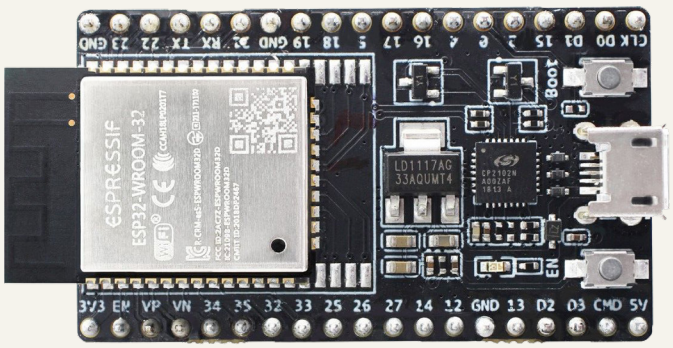
Wybrane parametry modułu:

- Protokoły BT: Bluetooth v4.2 BR/EDR oraz Bluetooth BLE
- Obsługa Bluetooth Audio: CVSD oraz SBC
- Protokoły Wi-Fi: 802.11 b/g/n/d/e/i/k/r (802.11n o przepływności do 150 Mbit/s)
- Tryby pracy Wi-Fi: sta/softAP/SoftAP+sta/P2P
- Zabezpieczenia Wi-Fi: WPA/WPA2/WPA2-Enterprise/WPS
- Szyfrowanie Wi-Fi: AES/RSA/ECC/SHA
- Protokoły sieciowe: IPv4, IPv6, SSL, TCP/UDP/HTTP/FTP/MQTT
- Wbudowana antena PCB (MIFA)
- Wbudowany konwerter USB-UART
- Wyprowadzenia GPIO o rozstawie 2,54 mm
- Zasilanie z USB (gniazdo typu microUSB)
- Wymiary modułu: 54,5×27,9 mm

Klub Aplikantów Próbek

to inicjatywa redakcji „Elektroniki Praktycznej". W kontaktach z firmami redakcja często otrzymuje do przetestowania próbki podzespołów, modułów, a nawet całych urządzeń elektronicznych. Są to zwykle najnowsze typy/modele produktów na rynku. Z chęci podzielenia się z Czytelnikami tymi próbkami zrodziła się inicjatywa pod nazwą Klub Aplikantów Próbek.

Członkiem KAP może stać się każdy, kto zgłosi chęć przetestowania próbki. Wykaz i krótki opis próbek, którymi dysponuje redakcja EP, można znaleźć na stronie ***ep.com.pl/nowosci/kap***. Wystarczy wybrać próbkę i zaplanować jej zastosowanie. Następnie należy wysłać wiadomość na adres: damian.sosnowski@ep.com.pl (Redaktor Prowadzący) z prośbą o przesłanie bezpłatnej próbki, opisując swój pomysł oraz podając dane do wysyłki.



Mile widziane, choć nieobowiązkowe, jest też przysłanie do redakcji EP opisu wykonanej aplikacji próbek. Najciekawsze opisy opublikujemy na naszej stronie ***ep.com.pl*** lub na łamach „Elektroniki Praktycznej".

Z uwagi na ograniczoną liczbę dostępnych próbek i niemałe zainteresowanie nimi, prosimy o opisanie swojego pomysłu na projekt na naszym forum internetowym, w dziale poświęconym Klubowi Aplikantów Próbek ***http://bit.ly/2qeN28e***. Ponadto, by dodatkowo zwiększyć swoje szanse należy polubić fanpage „Elektroniki Praktycznej" na Facebooku (***http://bit.ly/2WygFO9***) oraz udostępnić post, w którym opisujemy rozdawane próbki. W przypadku podobnie interesujących pomysłów na projekty, będziemy uwzględniać to jako dodatkowe kryterium wyboru.