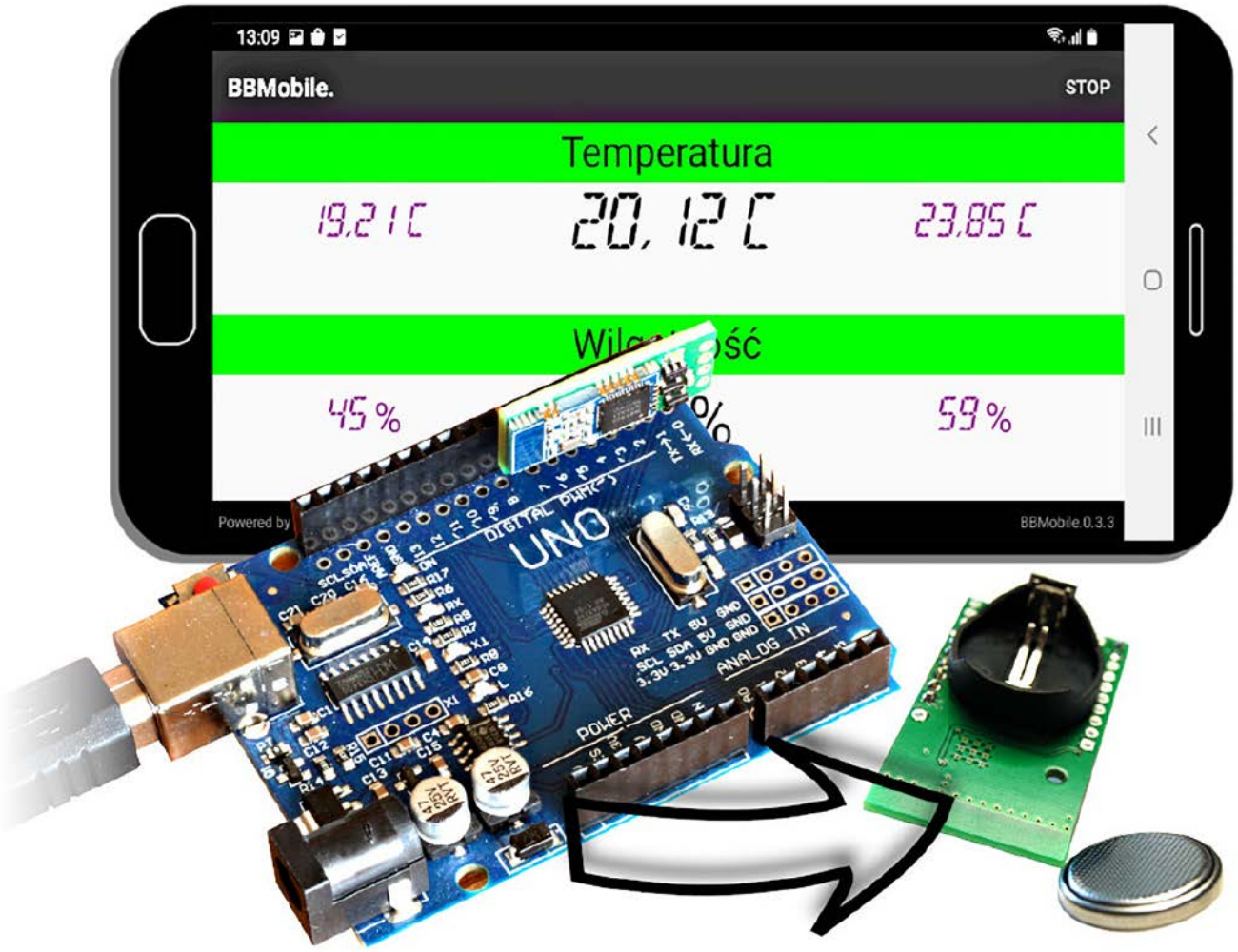


Bez programowania Androida

Projekt z użyciem BLE i aplikacji mobilnej (1)

W artykule zaprezentowano proces powstawania urządzenia elektronicznego, działającego z interfejsem mobilnym na ekranie smartfona. Co ciekawe, tworzenie projektu nie wymagało tworzenia aplikacji dla systemu Android – graficzny interfejs użytkownika jest budowany i obsługiwany przez samo urządzenie.

Dlaczego smartfon?

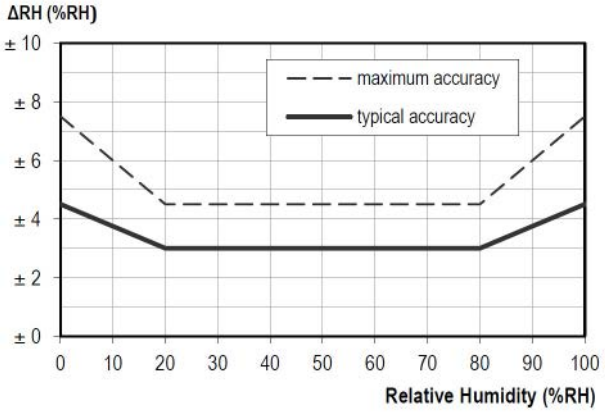
Rozwój techniki mikroprocesorowej, technologii radiowej i internetowej doprowadził nas do sytuacji, w której małe, trzymane w dłoni urządzenie z ekranem może stać się niemal całym światem. Widzimy to na ulicach i przystankach, w autobusach i tramwajach, parkach, placach zabaw, szkołach i biurach. W smartfonach połączonych w globalną sieć jest rozrywka, nauka, kultura, kontakty społeczne, wiedza, współzawodnictwo, biznes, spotkania towarzyskie, dyskusje polityczne itd. A bezkresne morze aplikacji dryfujących na maszynach serwerowych tylko czeka na pobranie, zainstalowanie i wypróbowanie. W opisanym projekcie zastosujemy ten cudowny produkt ewolucji...

Pomysł i założenia

Zbudujemy rejestrator temperatury i wilgotności, ale nie taki, ot... zwyczajny. Nasze urządzenie będzie niezwykłe, ponieważ wyposażymy je w wiele unikalnych cech zmiksowanych w jeden wielofunkcyjny produkt. Rejestrator będzie:

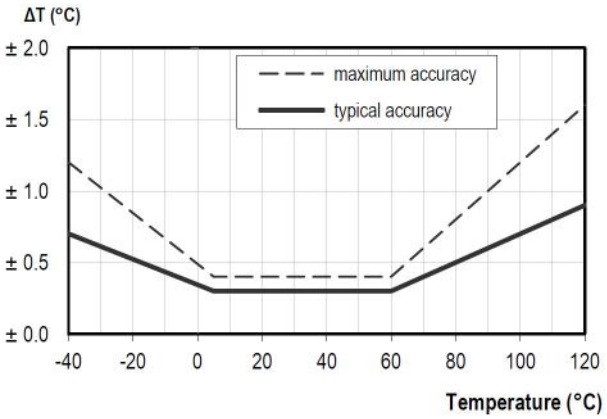
- bezprzewodowy, co oznacza, że może pracować w takim miejscu, które jest poza zasięgiem wzroku. Standardowe termometry i higrometry muszą być widoczne, w przeciwnym razie oczywiście nie odczytamy wskazania, czy to z wyświetlacza numerycznego, czy słupka cieczy w kapilarze;
- zawierał graficzny interfejs na ekranie smartfona, na którym czytelnie wyświetlane zostaną wszystkie informacje;
- potrzebował do pracy bardzo małej ilości energii, tak aby możliwe było zasilenie go nawet z baterii pastylkowej CR2032;

Parameter	Condition	Value	Units
Resolution [1]	12 bit	0.04	%RH
	8 bit	0.7	%RH
Accuracy tolerance [2]	typ	±3.0	%RH
	max	see Figure 2	%RH
Repeatability		±0.1	%RH
Hysteresis		±1	%RH
Nonlinearity		<0.1	%RH
Response time [3]	τ 63%	8	s
Operating Range	extended [4]	0 to 100	%RH
Long Term Drift [5]	Typ.	< 0.25	%RH/yr



Rysunek 1. Fragment dokumentacji układu SHT20 pokazujący jego najważniejsze parametry

Parameter	Condition	Value	Units
Resolution [1]	14 bit	0.01	°C
	12 bit	0.04	°C
Accuracy tolerance [2]	typ	±0.3	°C
	max	see Figure 3	°C
Repeatability		±0.1	°C
Operating Range	extended [4]	-40 t0 125	°C
Response Time [7]	τ 63%	5 to 30	s
Long Term Drift[8]	Typ.	< 0.02	°C/yr



Rysunek 2. Fragment dokumentacji układu SHT20 pokazujący jego parametry w pełnym zakresie temperatury

- wyposażony w interfejs Bluetooth Low Energy służący do zestawiania bezprzewodowego połączenia ze smartfonem lub tabletem;
- mierzył aktualną temperaturę i wilgotność z dużą dokładnością;
- wyświetlał dobową średnią, maksymalną oraz minimalną temperaturę i wilgotność;
- podawał średnią, maksymalną i minimalną temperaturę oraz wilgotność z ostatnich siedmiu dni;
- wyświetlał średnią, maksymalną oraz minimalną temperaturę i wilgotność zarejestrowaną od chwili włączenia zasilania;
- podawał liczbę dni, które upłynęły od startu procesu pomiarowego;
- mierzył i wyświetlał napięcie baterii/akumulatora, aby możliwe było określenie stanu źródła energii.

Najpierw narzędzia

Skoro mamy już pomysł i ogólne założenia, możemy przystąpić do określenia potrzebnych podzespołów i narzędzi. Ponieważ nasze urządzenie opiera się na energooszczędnej technologii **Bluetooth Low Energy (BLE)**, konieczne jest zastosowanie kompatybilnego chipu. Wybierzmy jeden z nordyckiej rodziny **NRF51**. Pomiar temperatury i wilgotności zrealizujemy sensorem **SHT20**. Będzie odpowiedni, ponieważ dysponuje interfejsem I²C oraz dużą dokładnością pomiarów przy zachowaniu reżimu energooszczędności. W trybie uśpienia pobiera typowo 0,15 µA, a podczas pracy 300 µA. Pomiar wilgotności względnej dostępny jest z typową dokładnością ±3% RH w szerokim zakresie temperatur od 20 do 80°C. W przedziale od 0 do 100°C błąd pomiaru nie przekracza 5% RH. Szczegóły zostały pokazane na **rysunku 1**. Warto wspomnieć, że sensor ten pracuje w pełnym zakresie wilgotności względnej, od 0% do 100% RH. Typowa dokładność pomiaru temperatury wynosi ±0,3°C w przedziale od 5 do 60°C. W pełnym przedziale od –40 do 120°C typowy błąd pomiaru jest nie większy niż ±0,75°C. Dokładne dane zostały pokazane na **rysunku 2**.

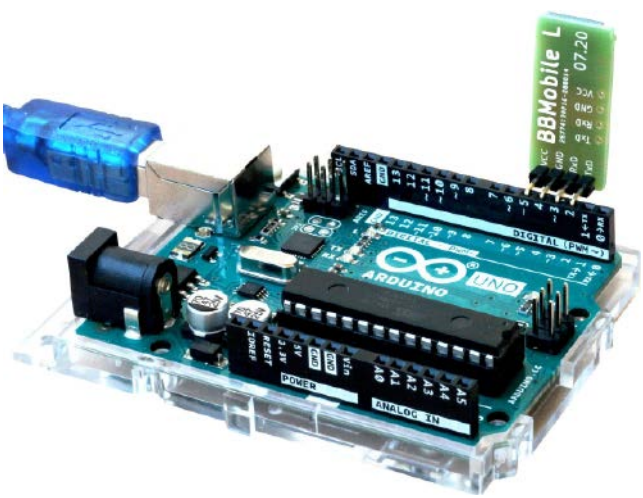
Przystępując do realizacji, rozwiejmy tajemnicę – jak to możliwe, że nie pisząc ani jednej linii kodu aplikacji mobilnej, wyświetlimy dane pomiarowe na ekranie smartfona lub tableta.

Bez pisania aplikacji na Androida

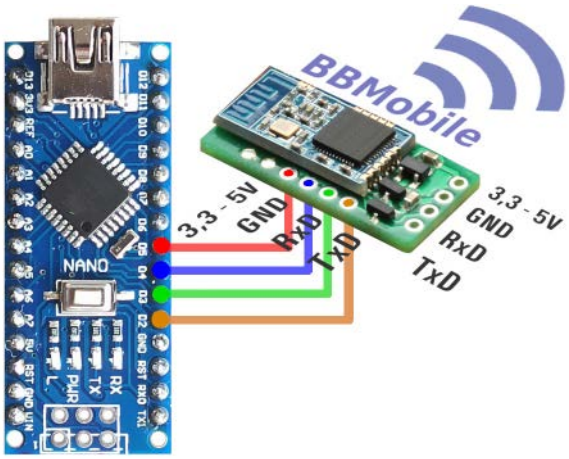
Opracowując elektroniczny produkt współpracujący bezprzewodowo z urządzeniem mobilnym (smartfon, tablet), standardowo mamy do wykonania dwa główne elementy:

1. samo urządzenie, najczęściej wyposażone w radiowy interfejs komunikacyjny: Bluetooth lub Wi-Fi;
2. aplikację instalowaną w systemie operacyjnym urządzenia mobilnego.

Takie sztampowe podejście jest dość czasochłonne i generuje wysokie koszty realizacji projektu, ponieważ poza wykonaniem samego produktu, każdorazowo konieczne jest zaprojektowanie i zaimplementowanie odpowiedniej aplikacji mobilnej. Spróbujmy zoptymalizować ten proces, pomijając drugi jego etap. Zamiast projektować i tworzyć całą aplikację mobilną od zera, użyjemy uniwersalnej – BBMobile dla systemu Android, która pozwoli zbudować i obsłużyć graficzny interfejs użytkownika za pomocą tekstowych komend. Pozostałe do wykonania prace podzielimy na dwa etapy:



Rysunek 3. Sposób dołączenia modułu BBMobile do płytki Arduino UNO

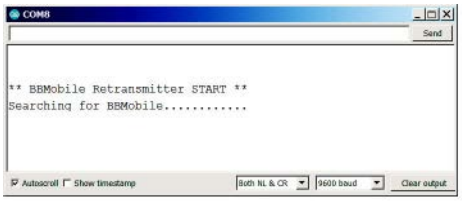


Rysunek 4. Sposób dołączenia modułu BBMobile do płytki Arduino NANO

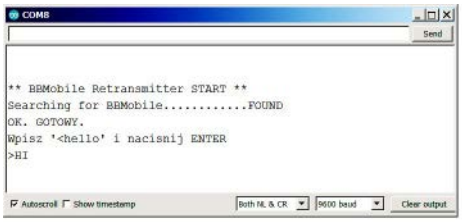
Parametry modułu BBMobile L

- Zasilanie: 3,3V do 5,0V,
- Średni pobór prądu: 1mA,
- Komunikacja przewodowa z mikrokontrolerem – port szeregowy – UART:
  - tolerancja 5V,
  - domyślnie 9600 bps, możliwe 4800 bps,
  - 8N1 – 8 bitów danych, bez bitu parzystości, 1 bit stopu,
- Komunikacja bezprzewodowa z urządzeniem mobilnym – BLE (Bluetooth Low Energy) – Bluetooth od wersji 4.0.
- Tworzenie aplikacji mobilnych poprzez przesłanie UARTem tekstowego kodu JSON,
- Komunikacja z aplikacją poprzez krótkie komunikaty tekstowe,
- Bezpłatna aplikacja BBMobile dla Androida od wersji 4.4.2
- Wymiary: 27 x 12 x 4 mm,

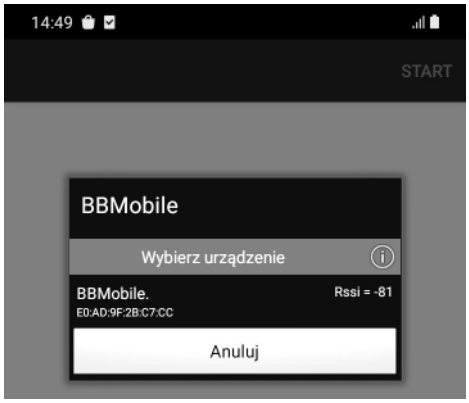
Rysunek 5. Fragment dokumentacji modułu BBMobile



Rysunek 6. Konfiguracja monitora portu szeregowego



Rysunek 7. Potwierdzenie komunikacji z modułem



Rysunek 8. Uruchomienie aplikacji BBMobile

- zaprojektowanie interfejsu graficznego oraz komunikacji z nim za pomocą płytki UNO (**rysunek 3**) bądź NANO (**rysunek 4**), kompatybilnej z Arduino IDE oraz modułu **BBMobile** (jego parametry zostały pokazane na **rysunku 5**);
- przeniesienie zaprojektowanych rozwiązań na ostateczną platformę hardware'ową.

Do projektowania interfejsu mobilnego wykonujemy następujące czynności:

1. Uruchamiamy Arduino IDE i otwieramy szkic *BBMobile_Ardu_Retransmitter.ino*. Szkic można pobrać np. z witryny www.bbmagic.net;
2. Programujemy płytkę UNO lub NANO. Uruchomiony program kopiuje dane pomiędzy dwoma portami szeregowymi. Wszystko, co pojawi się na sprzętowym porcie BBM_SERIAL, wysyłane jest na programowy port MON_SERIAL i odwrotnie. Główna część tego programu została zaprezentowana na **listingu 1**;
3. W Arduino IDE uruchamiamy Serial Monitor (wybierając Tools → Serial Monitor lub naciskając Ctrl+Shift+M). Konieczne jest wybranie ustawień w prawym dolnym rogu: *Both NL&CR* oraz *9600 baud*, tak jak zostało to pokazane na **rysunku 6**. W oknie zobaczymy powitalny komunikat oraz kolejno pojawiające się kropki informujące o oczekiwaniu na komunikację z modułem BBMobile;
4. Teraz podłączamy moduł tak, jak pokazano na rysunku 3 lub rysunku 4. Ponieważ BBMobile zasilany jest z pinów kontrolera (pin 4: GND, pin 5: VCC), bezpiecznej jest najpierw załadować program, a dopiero potem podłączyć moduł. Po jego wykryciu w oknie natychmiast pojawi się komunikat taki, jak na **rysunku 7**;
5. Wpisanie do górnego okna Serial Monitora komendy `<hello` i naciśnięcie ENTER przetestuje komunikację z modułem. W odpowiedzi dostaniemy przyjazne `>HI` (rysunek 7);
6. Na smartfonie (z Bluetooth w wersji minimum 4.0) instalujemy i uruchamiamy aplikację BBMobile. System Android wymaga włączenia usługi lokalizacji (jeśli nie jest włączona), aby możliwe było skanowanie w poszukiwaniu urządzeń Bluetooth Low Energy;
7. Po uruchomieniu aplikacji dotykamy przycisku START w prawym górnym rogu ekranu i wybieramy z listy urządzenie o nazwie BBMobile, tak jak pokazuje to **rysunek 8**;
8. Po chwili połączenie BLE zostanie zestawione, a w głównym oknie Serial Monitora pojawi się nowy komunikat, jak na **rysunku 9**. Komunikat `>CON` informuje o nawiązaniu połączenia i po kropce zawiera sześciobajtowy adres urządzenia, z którym zestawiono komunikację.

Od tej chwili wszystkie komendy wpisane do górnego paska w oknie Serial Monitora zostaną przesłane wprost do aplikacji BBMobile, która będzie je wykonywała. W ten sposób możemy budować i kontrolować

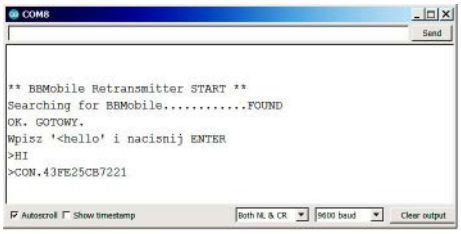
Listing 1. Kod programu testowego *BBMobile_Ardu_Retransmitter.ino*

```
//-----------------------------------------------------
// MAIN CODE HERE, TO RUN REPEATEDLY
//-----------------------------------------------------
VOID LOOP()
{
  I++ ;

  // READ FROM MONITOR PORT AND SEND TO BBMOBILE PORT:
  IF (MON_SERIAL.AVAILABLE()) {
    INT INBYTE = MON_SERIAL.READ();
    BBM_SERIAL.WRITE(INBYTE);
  }

  // READ FROM BBMOBILE PORT AND SEND TO MONITOR PORT:
  IF (BBM_SERIAL.AVAILABLE()) {
    INT INBYTE = BBM_SERIAL.READ();
    MON_SERIAL.WRITE(INBYTE);
  }

  // LED BLINKING
  IF(I ==4000) DIGITALWRITE(LED_BUILTIN, LOW) ;  //-TURN THE LED OFF
  IF(I > 20000)
  {
    I =0 ;
    DIGITALWRITE(LED_BUILTIN, HIGH);  //-TURN THE LED ON
  }
}
```



Rysunek 9. Informacja o zestawieniu połączenia

interfejs na ekranie smartfona za pomocą prostych, tekstowych poleceń. Przystępujemy zatem do prac projektowych.

Pierwszy interfejs mobilny

Jeśli połączenie z aplikacją mamy zestawione, to główne okno Serial Monitora wygląda jak to z rysunku 9. Pora teraz na zbudowanie pierwszego interfejsu mobilnego za pomocą kilku prostych, tekstowych poleceń. Skopiujmy kod JSON (*JavaScript Object Notation*) widoczny na **listingu 2** i wklejmy go do górnego paska Serial Monitora, naciskając prawy przycisk myszy i wybierając Paste lub Wklej. Naciskamy ENTER i jeśli wklejony kod nie zawierał błędów, to w głównym oknie Serial Monitora pojawi się komunikat $ok potwierdzający przyjęcie danych. Ekran smartfona powinien wyglądać jak ten z **rysunku 10**. Przesłany kod JSON jest definicją najprostszego chyba interfejsu zawierającego tylko dwa elementy. Są to nieedytowalne pola tekstowe TextView, które wyświetlają napisy. Nie będziemy tutaj dokładnie analizować składni kodu JSON. Wiele informacji na ten temat można znaleźć pod adresem http://bbmagic.net/co-to-jest-json-i-jak-go-uzyc/.

Skoro wiemy już, że komunikacja z aplikacją BBMobile działa wyśmienicie, to możemy dodać do tego prostego interfejsu kolejne elementy. Nieco bardziej rozbudowany kod zawiera **listing 3**. Znajdują się tu już cztery kontrolki TextView ułożone wertykalnie (jedna pod drugą). Kontrolki zdefiniowane w liniach 3 i 5 (wyświetlające wartości temperatury i wilgotności) otrzymały swoje nazwy, odpowiednio: „t” i „h”. Nadanie kontrolkom unikalnych nazw jest konieczne, aby możliwe było późniejsze zmienianie wyświetlanej przez nie treści. Zwróćmy uwagę, że w czwartej linii kodu polskie znaki w słowie „Wilgotność” zostały zakodowane za pomocą podanych wprost wartości unicode. Takie rozwiązanie spowoduje poprawne ich wyświetlanie niezależnie od tego, jakiego edytora tekstu będziemy używać do przygotowania kodu JSON.

Wklejmy zatem gotowy kod do górnego paska Serial Monitora. Aplikacja mobilna wygląda teraz tak, jak na **rysunku 11**. Wpiszmy kolejne polecenie: `$set,t:te="19,8 C"`, zatwierdzając klawiszem ENTER i jeszcze kolejne `$set,h:te="48 %"` (i oczywiście ENTER na końcu). Łatwo zauważyć, że komendy te zmieniają wyświetlane wartości temperatury i wilgotności. Można je połączyć w jedno polecenie, które będzie wyglądało tak: `$set,t:te="19,8 C",h:te="48 %"`.

Ten interfejs mógłby już sprostać obsłużeniu bezprzewodowego termometru i higrometru. No, może wskazane byłoby jeszcze uatrakcyjnienie go jakąś grafiką. W linii pierwszej **listingu 4** polecenie `"img":"3"` wybiera grafikę o numerze porządkowym 3 jako obrazek

Listing 2. Kod JSON opisujący pierwszy interfejs

```
1  {"ty":"lout","or":"V","cs":[
2  {"ty":"Textview","te":"\nRH T\nRejestrator","tc":"255,0,255","ts":"50","tf":"03"},
3  {"ty":"Textview","te":"\nTo jest projekt bez pisania aplikacji mobilnej","tc":"80,80,80","ts":"30"},
4  {"ty":"Textview","w":"5"}]}
```

Rysunek 10. Wygląd pierwszej aplikacji mobilnej

Rysunek 11. Wygląd aplikacji mobilnej wyświetlającej parametry

Rysunek 12. Wygląd aplikacji mobilnej ze zmienionym tłem i kolorem czcionki

tła. Konieczna okazała się też zmiana koloru czcionki wyświetlanego tekstu, z poprzednio użytego czarnego (składowe R, G, B = 0, 0, 0) na nieco jaśniejszy (składowe R, G, B = 250, 250, 250). Efekt zmian został pokazany na **rysunku 12**.

Listing 3. Kod JSON rozbudowujący pierwszy interfejs

```
1   {"ty":"lout","or":"V","cs":[
2   {"ty":"Textview","te":"\nTemperatura","tc":"0,0,0","ts":"40"},
3   {"ty":"Textview","n":"t","te":"20,1 C","tc":"0,0,0","ts":"50","tf":"02"},
4   {"ty":"Textview","te":"\nWilgotno\u015B\u0107","tc":"0,0,0","ts":"40"},
5   {"ty":"Textview","n":"h","te":"52 %","tc":"0,0,0","ts":"50","tf":"02"},
6   {"ty":"Textview","w":"5"}]}
```

Listing 4. Kod JSON zmieniający wygląd interfejsu

```
1   {"ty":"lout","or":"V","img":"3","cs":[
2   {"ty":"Textview","te":"\nTemperatura","tc":"250,250,250","ts":"40"},
3   {"ty":"Textview","n":"t","te":"20,1 C","tc":"250,250,250","ts":"50","tf":"02"},
4   {"ty":"Textview","te":"\nWilgotno\u015B\u0107","tc":"250,250,250","ts":"40"},
5   {"ty":"Textview","n":"h","te":"52 %","tc":"250,250,250","ts":"50","tf":"02"},
6   {"ty":"Textview","w":"5"}]}
```

Horyzontalne kontrolki dla wartości min. i max.

Zakładamy, że projektowane urządzenie będzie zapamiętywało wartości minimalne i maksymalne temperatury oraz wilgotności, zatem interfejs należy jeszcze nieco rozbudować. Sprawdźmy, czy dobrym pomysłem będzie użycie do tego celu horyzontalnego (jedna obok drugiej) ułożenia kontrolek. Oczywiście nie wszystkich, lecz tylko tych, które wskazują te same wielkości. Efekt wizualny tego pomysłu, tym razem zaprezentowany na obróconym o 90 stopni ekranie smartfona, pokazuje **rysunek 13**. Implementację tego rozwiązania pokazuje kod JSON z **listingu 5**. Opisuje on sześć głównych elementów interfejsu graficznego. Są to:

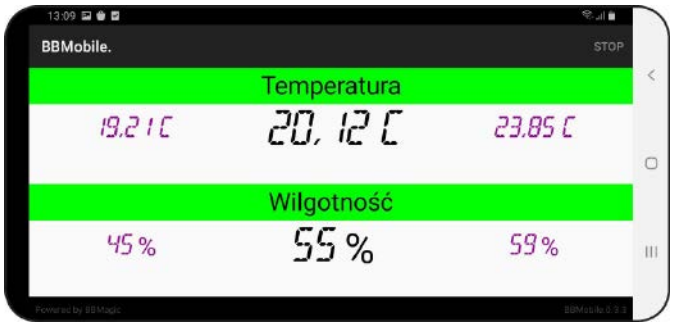
Listing 5. Kod JSON dodający do aplikacji wyświetlanie wartości maksymalnych i minimalnych

```
1   {"ty":"lout","or":"V","cs":[
2   {"ty":"Textview","te":"Temperatura","tc":"0,0,0","bg":"0,255,0","ts":"30"},
3   {"ty":"lout","or":"H","cs":[
4       {"ty":"Textview","n":"tm","te":"19,21 C","tc":"150,0,150","ts":"30","tf":"02"},
5       {"ty":"Textview","n":"t","te":"20,12 C","tc":"0,0,0","ts":"50","tf":"02"},
6       {"ty":"Textview","n":"tx","te":"23,85 C","tc":"150,0,150","ts":"30","tf":"02"}]},
7   {"ty":"Textview","tc":"0,0,0","ts":"30"},
8   {"ty":"Textview","te":"Wilgotno\u015B\u0107","tc":"0,0,0","bg":"0,255,0","ts":"30"},
9   {"ty":"lout","or":"H","cs":[
10      {"ty":"Textview","n":"hm","te":"45 %","tc":"150,0,150","ts":"30","tf":"02"},
11      {"ty":"Textview","n":"h","te":"55 %","tc":"0,0,0","ts":"50","tf":"02"},
12      {"ty":"Textview","n":"hx","te":"59 %","tc":"150,0,150","ts":"30","tf":"02"}]},
13  {"ty":"Textview","w":"5"}]}
```

1. kontrolka wyświetlająca napis „Temperatura” (linia 2);
2. layout układający w swoim wnętrzu horyzontalnie (jedna obok drugiej) trzy kontrolki (linie od 3 do 6). Wyświetlają one wartości temperatury: minimalną, aktualną i maksymalną;
3. puste pole tekstowe podnoszące czytelność interfejsu (linia 7);
4. kontrolka wyświetlająca napis „Wilgotność” (linia 8);
5. drugi layout horyzontalny zawierający w swoim wnętrzu trzy kontrolki (linie od 9 do 12), tym razem wyświetlające wartości wilgotności: minimalną, aktualną i maksymalną;
6. puste pole tekstowe podnoszące czytelność interfejsu (linia 13).

Ta odsłona interfejsu jest już dość blisko celu, który określiliśmy na starcie. Wydaje się, że cyfry są zbyt duże, co uniemożliwi czytelne ich wyświetlenie w pionowej orientacji ekranu. Po prostu nie mieszczą się w jednym, poziomym wierszu i utworzą kolejną linię, co mocno obniży funkcjonalność interfejsu. Popracujemy nad tym w kolejnej części publikacji i zorganizujemy też sterowanie interfejsem.

Mariusz Żądło



Rysunek 13. Wygląd aplikacji mobilnej rozbudowanej o wskazywanie wartości maksymalnych i minimalnych