Autor dziękuje panu Sławomirowi Szweda z firmy Unisystem za dostarczenie wyświetlacza zastosowanego w projekcie wiStation.

DODATKOWE MATERIAŁY NA FTP:

ftp://ep.com.pl
USER: 47858, PASS: 9seghusa

W ofercie AVT*
AVT-5605

Podstawowe informacje:

- Napięcie zasilające: 5...9 V DC.
- Maksymalny/minimalny prąd obciążenia: 240/20 mA.
- Niepewność pomiaru wilgotności pomieszczenia: 2%.
- Zakres pomiarowy wilgotności pomieszczenia: 0...99%.
- Niepewność pomiaru temperatury pomieszczenia: 0,5°C.
- Zakres pomiarowy temperatury pomieszczenia: 0...99°C.

Projekty pokrewne na FTP:
(wymienione artykuły są w całości dostępne na FTP)

AVT-5566	THPStation – rozbudowany termometr z Wi-Fi (EP–1/2017)
AVT-5518	Termometr bezprzewodowy (EP 11/2015)
AVT-5494	Termometr bezprzewodowy z interfejsem USB (EP 4/2015)
AVT-1790	Termometr XXL (EP 2/2014)
AVT-5489	8-kanałowy termometr z alarmem i wyświetlaczem LCD (EP 11/2013)
AVT-5420	Wielopunktowy termometr z rejestracją (EP 10/2013)
AVT-1734	Termometr do wędzarni (EP 4/2013)
AVT-5373	Tlogger – rejestrator temperatury (EP 12/2012)
AVT-1697	Wielogabarytowy termometr LED (EP 8/2012)
AVT-5389	4-kanałowy termometr z wyświetlaczem LED (EP 5/1012)
AVT-5330	Termometr PC (EP 2/2012)
AVT-5230	Rejestrator temperatury z interfejsem USB (EP 4/2010)

*** Uwaga! Elektroniczne zestawy do samodzielnego montażu. Wymagana umiejętność lutowania!**
Podstawową wersją zestawu jest wersja [**B**] nazywana potocznie KITem (z ang. zestaw). Zestaw w wersji [**B**] zawiera elementy elektroniczne (w tym [UK] – jeśli występuje w projekcie), które należy samodzielnie wlutować w dołączoną płytkę drukowaną (PCB). Wykaz elementów znajduje się w dokumentacji, która jest podlinkowana w opisie kitu.
Mając na uwadze różne potrzeby naszych klientów, oferujemy dodatkowe wersje:
- wersja [**C**] zmontowany, uruchomiony i przetestowany zestaw [**B**] (elementy wlutowane w płytkę PCB)
- wersja [**A**] płytka drukowana bez elementów i dokumentacja

Kity w których występuje układ scalony wymagający zaprogramowania, posiadają następujące dodatkowe wersje:
- wersja [**A+**] płytka drukowana [**A**] + zaprogramowany układ [**UK**] i dokumentacja
- wersja [**UK**] zaprogramowany układ

Nie każdy zestaw AVT występuje we wszystkich wersjach! Każda wersja ma załączony ten sam plik pdf! Podczas składania zamówienia upewnij się, którą wersję zamawiasz! ***http://sklep.avt.pl***

wiStation

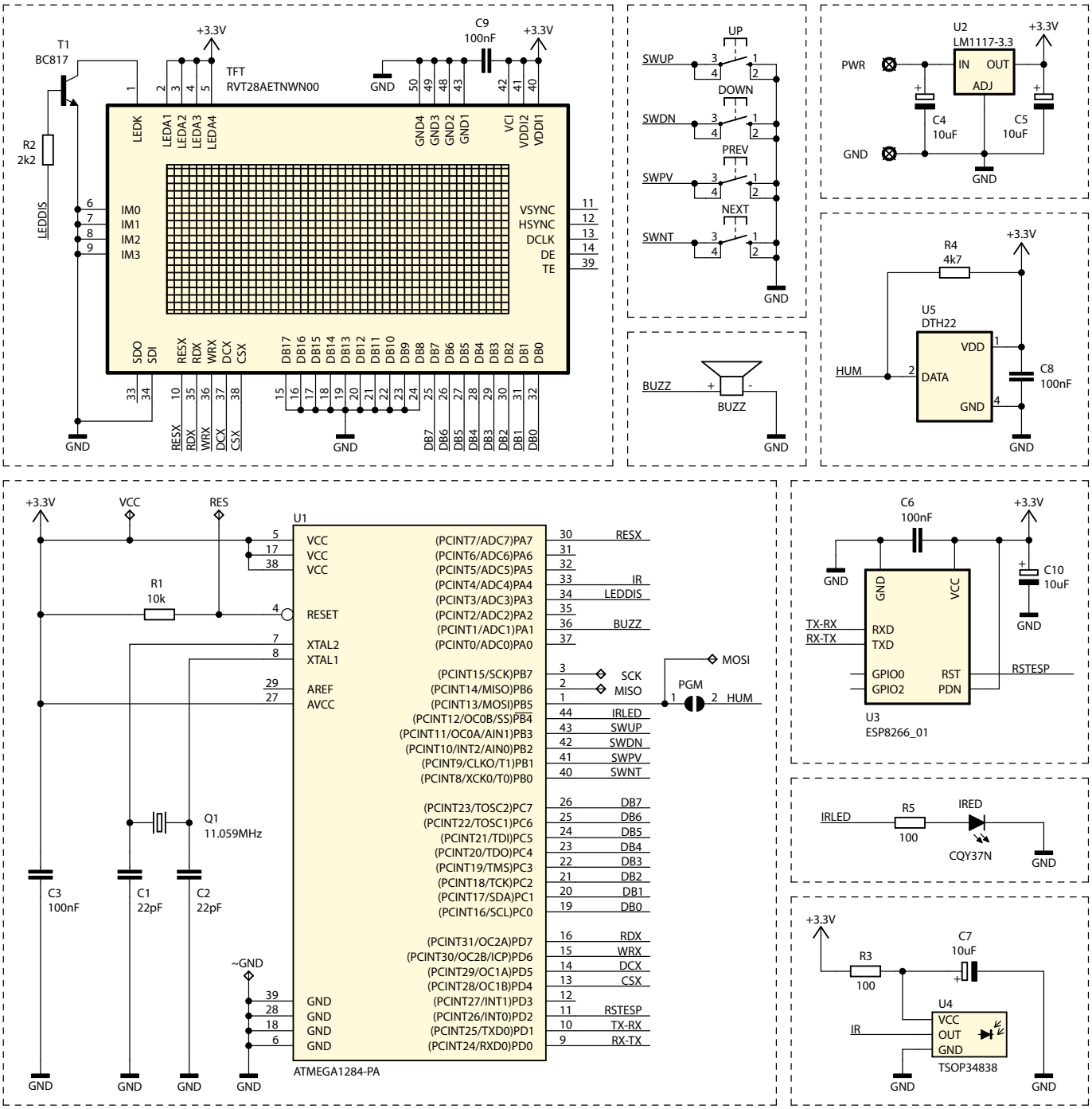
Domowa stacja pogodowa z prognozą pogody

W artykule zaprezentowano opis rozwiązania domowej stacji pogodowej, która oprócz pomiaru parametrów środowiskowych ma funkcjonalność prognozowania pogody. A wszystko dzięki łączności z Internetem zapewnianej przez moduł z popularnym układem ESP8266.

***Rekomendacje:** użyteczny gadżet, który przyda się w każdym domu.*

„Internet Rzeczy" to termin, którego praktycznego znaczenia nie chciałem rozumieć. Tak po prostu, zwyczajnie, z niechęci do dzisiejszej „orwellowskiej rzeczywistości", gdzie wszystko i wszyscy muszą być on-line, aby być elementem wielkiej, globalnej maszyny informacyjnej. Byłem niechętny IoT do czasu, aż natknąłem się na niezwykle już dzisiaj popularny moduł z układem ESP8266 firmy Espressif Systems będący kolejnym przykładem kreatywności chińskiej myśli technicznej. Umożliwia on za przysłowiowe kilka dolarów dołączenie dowolnego systemu mikroprocesorowego do Internetu i to bezprzewodowo! Taka możliwość istniała również wcześniej, jednak wymagało to zaangażowania znacznie większych środków finansowych, sprzętowych oraz niemało pracy. Chińczycy pokazali, że tak nie musi być, że można zaoferować bardzo dobry produkt w przystępnej, żeby nie powiedzieć „śmiesznie niskiej" cenie i spowodować, że komunikacja bezprzewodowa Wi-Fi stanie się szeroko dostępna również dla amatorów.

Moduł, o którym mowa, jest przejściówką UART/Wi-Fi. Domyślnie komunikacja z nim przebiega przez interfejs UART (CMOS 3,3 V) z użyciem komend AT, jednak sam układ ESP8266 stanowiący serce urządzenia (i będący 32-bitowym mikrokontrolerem RISC) jest programowalny, co daje możliwość wyeliminowania dodatkowego mikrokontrolera w wielu nieskomplikowanych systemach mikroprocesorowych. W tym celu producent modułu dostarcza odpowiedni pakiet SDK (w tym API), jednak w sieci znaleźć możemy kilka autorskich projektów IoT bazujących na oryginalnym oprogramowaniu (np. projekt NodeMCU i język LUA), a wyposażonych w rozszerzoną funkcjonalność lub pozwalających na bezpośrednie programowanie urządzenia w języku C, C++ (oficjalne rozszerzenie dla projektu Arduino) czy nawet Basic. Co więcej, moduł jest



Rysunek 1. Schemat ideowy wiStation

sprzedawany w kilku wersjach, różniących się przede wszystkim liczbą wyprowadzeń GPIO, a co za tym idzie, wyglądem i wymiarami obwodu drukowanego. Im większa liczba wyprowadzeń, tym większe są możliwości zastosowania układu jako urządzenia autonomicznego, bo może to oznaczać więcej wbudowanych modułów peryferyjnych.

Najtańszą i teoretycznie najbardziej ograniczoną wersją jest ESP-8266-01, która jest montowana na płytce drukowanej wyposażonej w złącze goldpin mające 2×4 wyprowadzeń. Na to złącze wyprowadzono zasilanie (GND i 3,3 V), sygnały interfejsu UART (RXD i TXD, CMOS 3,3 V), sygnał zerowania RST, sygnał aktywacji chipsetu CH_PD (podłączany do zasilania w trakcie „normalnej" pracy modułu) oraz dwa wyprowadzenia wejścia/wyjścia ogólnego przeznaczenia (GPIO0 i GPIO2). Co ważne, moduł łączy się z sieciami Wi-Fi pracującymi w standardach 802.11 b/g/n w paśmie 2,4 GHz. Ma wbudowany stos TCP/IP i może działać w trybach AP (Access Point), STA (station) oraz mieszanym AP+STA. Moduł wspiera technologie CCMP, TKIP, WEP, CRC, WPA/WPA2 i WPS. Do kontroli kondycji układu w strukturze znajdziemy wbudowany sensor temperatury. Warto również wspomnieć o maksymalnym poborze prądu, którego natężenie zależy od standardu sieci, w której zasięgu pracuje moduł i wynosi: 215 mA dla sieci 802.11b, 197 mA dla „g" i 135 mA dla „n".

Wykaz elementów:
Rezystory: (SMD 0805)
R1: 10 kΩ
R2: 2,2 kΩ
R3, R5: 100 Ω
R4: 4,7 kΩ
Kondensatory: (SMD 0805)
C1, C2: 22 pF
C3, C6, C8, C9: 100 nF
C4, C5, C7, C10: 10 µF/16 V (SMD „A”)
Półprzewodniki:
U1: ATmega1284-PA (TQFP44)
U2: LM1117-3.3 (TO-220)
U4: TSOP34838
U5: DTH22/AM2302
T1: BC817 (SOT23)
IRED: CQY37N (lub inna)
Inne:
TFT: wyświetlacz Riverdi RVT28AETNWN00
U3: moduł ESP8266-01 (2×4 pin)
Q1: rezonator kwarcowy 11,059 MHz (niski)
UP, DOWN, PREV, NEXT: przycisk o wysokości 1 mm
ZIF: gniazdo ZIF ZIF0550DH

Tyle na temat samego modułu, gdyż dziesiątki informacji i praktycznych przykładów wykorzystania bez problemu znajdziemy w Internecie oraz w „Elektronice Praktycznej”. Zatem pora przejść do sedna!

Kilka lat temu w EP 6/2011 opublikowałem projekt stacji pogodowej wyposażonej w mechanizm prognozowania pogody. Korzystała ona z lokalnych, wbudowanych czujników temperatury i ciśnienia atmosferycznego oraz programowego mechanizmu prognozowania pogody opartego na śledzeniu zmian wartości ciśnienia atmosferycznego. Pomyślałem sobie, że skoro dzisiaj tak łatwo możemy włączyć system mikroprocesorowy do sieci, fajnym pomysłem byłoby zbudowanie stacji pogodowej, która szczegółowe dane pogodowe (wraz z prognozą) „ściąga” z sieci Internet, posiłkując się dostępem do odpowiedniego serwera pogodowego.

Medium transmisji i połączenia z siecią już mamy. Za realizację tej funkcjonalności będzie odpowiadał moduł ESP8266-01 sterowany za pomocą komend AT wysyłanych z mikrokontrolera nadrzędnego, który „przy okazji” obsłuży interfejs użytkownika. Pozostaje znalezienie odpowiedniego serwera pogodowego. Wybór padł na serwis ***Wunderground.com***, który skupia stacje pogodowe przyłączone do sieci Internet i dostarcza darmową usługę (oprócz kont płatnych o rozszerzonej funkcjonalności) oraz API pozwalające na integrację dowolnego systemu mikroprocesorowego ze środowiskiem po stronie serwera. Wspomniany serwis wymaga rejestracji, dzięki której dostaniemy dostęp do zgromadzonych tam danych, jak i własny, unikalny klucz, bez którego wspomniana usługa nie może być realizowana. Warto wspomnieć, że ***Wunderground.com*** jest jednym z najbardziej popularnych serwisów tego typu udostępniających najwięcej danych pogodowych. Aby zdobyć wspomniany wcześniej klucz,

Listing 1. Fragment pliku tekstowego, jaki zwraca serwer Wunderground.com na zapytanie o stan bieżącej pogody dla Warszawy

```
{
  „response”: {
  „version”:”0.1”,
  „termsofService”:”http://www.wunderground.com/weather/api/d/terms.html”,
  „features”: {
  „conditions”: 1
  }
    }
  ,     „current_observation”: {
         „image”: {
         „url”:”http://icons.wxug.com/graphics/wu2/logo_130x80.png”,
         „title”:”Weather Underground”,
         „link”:”http://www.wunderground.com”
         },
         „display_location”: {
         „full”:”Warsaw, Poland”,
         „city”:”Warsaw”,
         „state”:”MZ”,
         „state_name”:”Poland”,
         „country”:”PL”,
         „country_iso3166”:”PL”,
         „zip”:”00000”,
         „magic”:”1”,
         „wmo”:”12375”,
         „latitude”:”52.16999817”,
         „longitude”:”20.96999931”,
         „elevation”:”106.1”
         },
         „observation_location”: {
         „full”:”Warsaw, „,
         „city”:”Warsaw”,
         „state”:””,
         „country”:”PL”,
         „country_iso3166”:”PL”,
         „latitude”:”52.16999817”,
         „longitude”:”20.96999931”,
         „elevation”:”351 ft”
         },
         „estimated”: {
         },
         „station_id”:”EPWA”,
         „observation_time”:”Last Updated on January 22, 7:00 AM CET”,
         „observation_time_rfc822”:”Sun, 22 Jan 2017 07:00:00 +0100”,
         „observation_epoch”:”1485064800”,
         „local_time_rfc822”:”Sun, 22 Jan 2017 07:30:28 +0100”,
         „local_epoch”:”1485066628”,
         „local_tz_short”:”CET”,
         „local_tz_long”:”Europe/Warsaw”,
         „local_tz_offset”:”+0100”,
         „weather”:”Overcast”,
         „temperature_string”:”32 F (0 C)”,
         „temp_f”:32,
         „temp_c”:0,
         „relative_humidity”:”100%”,
         „wind_string”:”From the SSW at 4 MPH”,
         „wind_dir”:”SSW”,
         „wind_degrees”:210,
         „wind_mph”:4,
         „wind_gust_mph”:0,
         „wind_kph”:6,
         „wind_gust_kph”:0,
                „pressure_mb”:”1033”,
```

Listing 2. Funkcja inicjalizacyjna interfejsu USART (wraz z niezbędnymi definicjami)

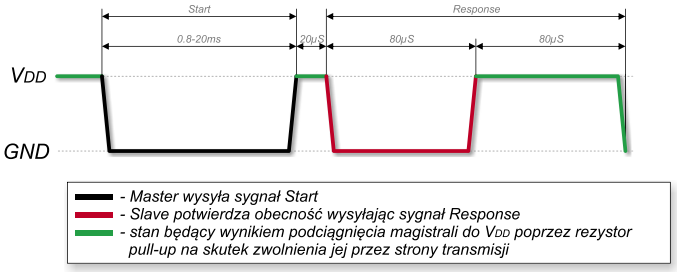
```
#define LOW_SPEED 0
#define HIGH_SPEED 1
#define USART_BAUD_9600 9600
#define UBRR_9600 (F_CPU/16/USART_BAUD_9600-1)
#define USART_BAUD_115200 115200
#define UBRR_115200 (F_CPU/16/USART_BAUD_115200-1)

void USARTinit(uint8_t Speed)
{
    //Ustawienie prędkości 9600 lub 115200 bps
    if(Speed == LOW_SPEED) {UBRR0H = UBRR_9600>>8; UBRR0L = UBRR_9600;}
    else {UBRR0H = UBRR_115200>>8; UBRR0L = UBRR_115200;}
    //Załączenie nadajnika i odbiornika oraz uruchomienie przerwania od RX
    UCSR0B |= (1<<RXEN0)|(1<<TXEN0)|(1<<RXCIE0);
    //Ustawienie formatu ramki: 8bitów danych, 1 bit stopu
    UCSR0C = (3<<UCSZ00);
}
```

Listing 3. Funkcje pozwalające na wysłanie znaku, łańcucha z pamięci RAM i łańcucha z pamięci Flash mikrokontrolera

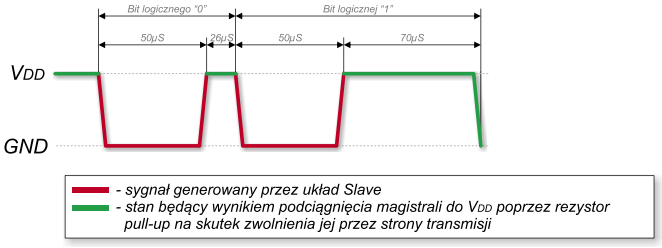
```
void USARTsendByte(uint8_t Byte)
{
    //Czekamy, aż bufor nadawczy będzie wolny
    while (!(UCSR0A & (1<<UDRE0)));
    UDR0 = Byte;
}

void USARTsendString(char *String, uint8_t Length)
{
    uint8_t i=0;
    //Wysyłamy kolejne bajty, aż do końca stringu lub wysłania „Length” bajtów
    while(String[i] && Length--) USARTsendByte(String[i++]);
}

void USARTsendString_P(const char *String)
{
    register char Byte;
    //Wysyłamy łańcuch z pamięci FLASH dopóki nie napotkamy 0
    while ((Byte = pgm_read_byte(String++))) USARTsendByte(Byte);
}
```

Listing 4. Funkcja obsługi przerwania odbiornika interfejsu USART

```
//Deklaracja typu wyliczeniowego statusu odpowiedzi modemu
typedef enum
{
    IDLE, MODEM_OK, ERROR, WIFI_DISCONNECT, WIFI_CONNECTED
} typeStatus;

//Deklaracja typu wyliczeniowego rodzaju pogody
typedef enum
{
    DRIZZLE, RAIN, SNOW, HAIL, FOG, THUNDERSTORM, CLOUDY, SUNNY, PARTLY_CLOUDY, MOSTLY_CLOUDY, UNKNOWN
} typeWeather;

//Deklaracja unii przechowującej parametry bieżącej pogody
typedef union
{
    struct
    {
        char Date[17];  //np: „Mon, 25 Dec 2016” (zawiera \0)
        uint8_t Hour;  //Bieżąca godzina
        uint8_t Minute;  //Bieżąca minuta
        typeWeather Weather;  //np: „Mostly Cloudy”
        char Temperature[6];  //np: „-4.0” (zawiera \0)
        char Humidity[4];  //np: „91%” (zawiera \0)
        char Pressure[5];  //np: „1015” (zawiera \0)
        typeWeather fWeather[3];  //Prognoza pogody na najbliższe 3 dni
        char fTempL[3][4];  //Prognoza najniższej temp. na najbliższe 3 dni, np. -15 (zawiera \0)
        char fTempH[3][4];  //Prognoza najwyższej temp. na najbliższe 3 dni, np. -12 (zawiera \0)
    };
    uint8_t Index[62];
} typeForecast;

ISR(USART0_RX_vect)
{
    static char Line[REPLY_LENGTH];  //Tablica zawierająca ciąg znaków odpowiedzi odebrany interfesjem USART
    static uint8_t Index;  //Index w tablicy Line
    register char lineLenght;  //Długość odczytanej linii
    register char Data = UDR0;  //Pobranie bajta danych z bufora sprzętowego UART
    static uint8_t celciusNr, conditionsNr;  //Zmienne mechanizmu wyszukiwania danych prognozy pogody
    //W łańcuchu wynikowym pomijamy wszystkie znaki mniejsze od SPACJI oraz pomijamy znak CUDZYSŁOWU
    //Analizę odebranego łańcucha rozpoczynamy po napotkaniu znaku końca linii LF (Line Feed)
    if(Data >= ASCII_SP)
        if(Data != ,”’) Line[Index++] = Data;
        else
        {
            if(Data == ASCII_LF)  //usunięto && Index
            {
                Line[Index] = ,\0’;
                //Ustalamy długość odebranej linii odpowiedzi
                lineLenght = strlen(Line);
                replyValue = IDLE;
                if(strcmp_P(Line, PSTR(„OK”)) == 0) replyValue = MODEM_OK;
                else if (strcmp_P(Line, PSTR(„WIFI CONNECTED”)) == 0) replyValue = WIFI_CONNECTED;
                    else if (strcmp_P(Line, PSTR(„WIFI DISCONNECT”)) == 0) replyValue = WIFI_DISCONNECT;
                //Początek odpowiedzi modemu: response: { -> zerujemy strukturę danych pogodowych
                        else if(strncmp_P(Line, PSTR(„response”), 8) == 0) for(uint8_t i=0; i<62; ++i) currentWeather.Index[i] = ,\0’;
                        //Bieżący czas lokalny: local_time_rfc822:Sun, 25 Dec 2016 18:37:35 +0100,
                            else if(strncmp_P(Line, PSTR(„local_time_rfc822”), 17) == 0)
                            {
                                strncpy(currentWeather.Date, &Line[18], 16);  //np: Mon, 25 Dec 2016
                                currentWeather.Hour = ((Line[35]-’0’)*10)+(Line[36]-’0’);
                                currentWeather.Minute = ((Line[38]-’0’)*10)+(Line[39]-’0’);
                            }
                                //Bieżąca pogoda: weather:Mostly Cloudy,
                                else if(strncmp_P(Line, PSTR(„weather”), 7) == 0) currentWeather.Weather = determineWeather(Line);
                                        //Bieżąca temperatura: temp_c:4.0,
                                        else if(strncmp_P(Line, PSTR(„temp_c”), 6) == 0) strncpy(currentWeather.Temperature, &Line[7],
lineLenght-8);
                                        //Bieżąca wilgotność: relative_humidity:91%,
                                        else if(strncmp_P(Line, PSTR(„relative_humidity”), 17) == 0) strncpy(currentWeather.Humidity,
&Line[18], lineLenght-19);
                                            //Bieżące ciśnienie: pressure_mb:1015,
                                            else if(strncmp_P(Line, PSTR(„pressure_mb”), 11) == 0) strncpy(currentWeather.Pressure,
&Line[12], lineLenght-13);
                                                else if(strncmp_P(Line, PSTR(„conditions”), 10) == 0)
                                                {
                                                    if(++conditionsNr>2)
                                                    {
                                                       currentWeather.fWeather[conditionsNr-3] = determineWeather(Line);
                                                       if(conditionsNr == 5) conditionsNr = 0;
                                                    }
                                                }
                                                else if(strncmp_P(Line, PSTR(„celsius”), 7) == 0)
                                                {
                                                    switch(celciusNr)
                                                    {
                                                        case 0:
                                                        case 1: break;
                                                        case 2: strncpy(&currentWeather.fTempH[0][0], &Line[8], lineLenght-8); break;
                                                        case 3: strncpy(&currentWeather.fTempL[0][0], &Line[8], lineLenght-8); break;
                                                        case 4: strncpy(&currentWeather.fTempH[1][0], &Line[8], lineLenght-8); break;
                                                        case 5: strncpy(&currentWeather.fTempL[1][0], &Line[8], lineLenght-8); break;
                                                        case 6: strncpy(&currentWeather.fTempH[2][0], &Line[8], lineLenght-8); break;
                                                        case 7:
                                                        //W tym momencie mamy komplet, więc sygnalizujemy gotowość danych pogodowych
                                                            strncpy(&currentWeather.fTempL[2][0], &Line[8], lineLenght-8);
                                                            espDataReady = 1;
                                                            break;
                                                    }
                                                    celciusNr = (celciusNr+1) & 0x07;
                                                }
                                                //Zerujemy zmienną Index przygotowując się na nadejście nowej linii danych
                                                Index = 0;
            }
        }
}
```



Rysunek 2. Sekwencja sygnałów sterujących na magistrali danych podczas startu transmisji danych



Rysunek 3. Sekwencja sygnałów sterujących na magistrali danych podczas przesyłania bitów danych

Listing 5. Funkcja narzędziowa odpowiedzialna za ustalenie rodzaju pogody

```
typeWeather determineWeather(char *String)
{
    if(strstr_P(String, PSTR(„Drizzle”)) != NULL) return DRIZZLE;
    else if(strstr_P(String, PSTR(„Rain”)) != NULL || strstr_P(String, PSTR(„Flurries”)) != NULL) return RAIN;
    else if(strstr_P(String, PSTR(„Snow”)) != NULL || strstr_P(String, PSTR(„Sleet”)) != NULL) return SNOW;
    else if(strstr_P(String, PSTR(„Hail”)) != NULL || strstr_P(String, PSTR(„Ice”)) != NULL) return HAIL;
    else if(strstr_P(String, PSTR(„Fog”)) != NULL || strstr_P(String, PSTR(„Mist”)) != NULL || strstr_P(String, PSTR(„Haze”)) != NULL) re-
turn FOG;
    else if(strstr_P(String, PSTR(„Thunderstorm”)) != NULL) return THUNDERSTORM;
    else if(strstr_P(String, PSTR(„Sunny”)) != NULL || strstr_P(String, PSTR(„Clear”)) != NULL) return SUNNY;
    else if(strstr_P(String, PSTR(„Partly Cloudy”)) != NULL || strstr_P(String, PSTR(„Scattered Clouds”)) != NULL) return PARTLY_CLOUDY;
    else if(strstr_P(String, PSTR(„Mostly Cloudy”)) != NULL) return MOSTLY_CLOUDY;
    else if(strstr_P(String, PSTR(„Cloudy”)) != NULL || strstr_P(String, PSTR(„Overcast”)) != NULL) return CLOUDY;
    return UNKNOWN;
}
```

wchodzimy na stronę ***https://goo.gl/Jr8gfR*** i zakładamy darmowe konto. Następnie wybieramy z menu górnego **More/Weather API for Developers** i w zakładce **Key Settings** odczytujemy wartość naszego klucza (**Key ID**). Możemy jeszcze sprecyzować nazwę naszego projektu, dla którego zakładamy konto oraz jego stronę www. Wspomniany klucz zapisujemy w celu dalszego wykorzystania.

Kilka zdań na temat API, gdyż pełny opis środowiska zdecydowanie wykraczałby poza zakres tego rodzaju artykułu, a możemy go znaleźć pod adresem ***https://goo.gl/rNr85x***. Załóżmy, że zarejestrowaliśmy darmowe konto w serwisie Wunderground.com, otrzymaliśmy unikalny klucz Key ID. Co dalej? Zadanie jest dość łatwe i szczegółowo opisane we wspomnianej dokumentacji API. W tym momencie musimy wysłać zapytanie do serwera pogodowego o stan bieżącej pogody, korzystając z następującej składni:

`GET /api/YOURKEY/conditions/q/COUNTRY/CITY.json`

gdzie:

- **YOURKEY** – unikalny klucz Key ID, który otrzymaliśmy od serwisu ***Wunderground.com***.
- **COUNTRY** – nazwa państwa w języku angielskim, dla którego oczekujemy prognozy pogody.
- **CITY** – nazwa miasta w języku angielskim, dla którego oczekujemy prognozy pogody.

W celach testowych zapytanie takie możemy również wysłać z poziomu dowolnej przeglądarki WWW, przy czym w takim wypadku zapytanie to musi ulec drobnej korekcie. Oto przykład dla Warszawy: ***https://goo.gl/Wcf6fK***. Podany przykład zapytania dotyczy informacji o bieżącym stanie pogody. W celu wysłania zapytania o stan bieżący i prognozę modyfikujemy je w następujący sposób ***https://goo.gl/p8DmcA***. Co szczególnie istotne, warto wziąć też pod uwagę ograniczenia darmowego konta w serwisie ***Wunderground.com***, które determinują maksymalną liczbę zapytań serwera w jednostce czasu. Korzystając z konta jak wyżej, możemy wysłać maksymalnie 10 zapytań w ciągu minuty i 500 zapytań na dobę, co wydaje się wystarczające w większości zastosowań.

Co takiego zrobi serwer po odebraniu tego rodzaju zapytania? Serwer ten przygotowuje dane dotyczące bieżącego stanu pogody i/lub jej prognozy (w zależności od wysłanego zapytania) i zwraca odpowiedź w postaci pliku tekstowego w formacie JSON. Format, o którym mowa, jest prostym formatem wymiany danych komputerowych zrozumiałych dla ludzi, a bazujących na podzbiorze języka JavaScript. Struktura pliku tekstowego w formacie JSON składa się z obiektów, gdzie każdy z nich stanowi zbiór par typu nazwa pola i jego wartość. Opis obiektu rozpoczyna znak „{„ a kończy znak „}”. Po każdej nazwie pola następuje „:” (dwukropek) a po nim wartość danego pola. Pary nazwa/wartość oddzielone są każdorazowo przecinkiem. Na **listingu 1** pokazano fragment pliku tekstowego, jaki zwraca serwer ***Wundergroud.com*** na zapytanie o stan bieżącej pogody dla Warszawy.

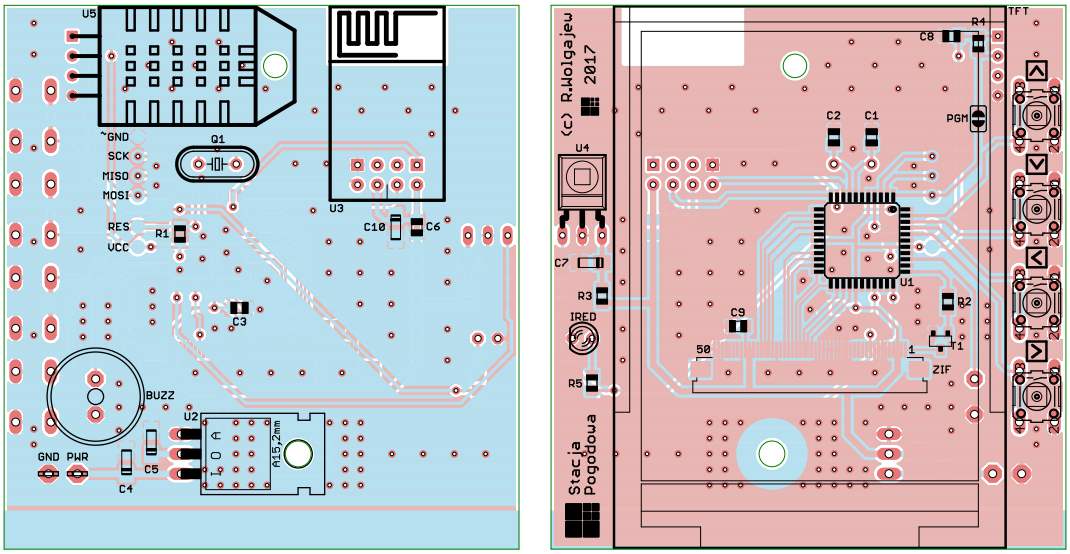
Jak wspomniano, jest to tylko fragment odpowiedzi serwera, gdyż pełna jego treść potrafi składać się – w wypadku pytania o prognozę pogody – z kilku tysięcy znaków, ponieważ zawiera drobiazgowe dane dotyczące pogody, jej prognozy, miejsca i czasu pomiaru, linki do elementów graficznych reprezentujących stan pogody oraz wiele innych informacji. Mimo, taki format odpowiedzi sprawia, że jest ona czytelna dla człowieka, ponieważ łatwo jest analizować zwracane parametry, biorąc pod uwagę fakt, że po nazwie każdego pola występuje łańcuch znaków reprezentujący wartość tego pola, zaś same nazwy specyfikują oczekiwane elementy zapytania. Listę wszystkich zwracanych pól, możliwe ich wartości oraz obsługę błędów jak zawsze znajdziemy w dokumentacji API.

Na tym etapie mamy już wszystkie niezbędne informacje dotyczące naszego medium transmisyjnego i samego serwisu pogodowego, więc pora na przedstawienie schematu ideowego urządzenia wiStation, który pokazano na **rysunku 1**. Jest to system mikroprocesorowy, którego sercem jest mikrokontroler ATmega1284 odpowiedzialny za realizację całej założonej funkcjonalności. Mikrokontroler steruje pracą modułu ESP8266 za pomocą interfejsu USART0 (piny RXD0, TXD0) pracującego z wykorzystaniem przerwania RX Complete Interrupt, realizuje obsługę graficznego interfejsu użytkownika z użyciem kolorowego wyświetlacza TFT o rozdzielczości 320×240 pikseli, klawiatury złożonej z 4 mikroprzycisków oraz bariery podczerwieni zbudowanej z nadajnika IRED w postaci diody LED i odbiornika (układ TSOP34838) pracującego z częstotliwością nośną 38 kHz oraz odpowiada za obsługę scalonego higrometru/termometru DTH-22 (odpowiednik AM2302), dzięki któremu jest możliwy pomiar temperatury i wilgotności wewnętrznej pomieszczenia, w którym znajduje się urządzenie. Wybór tego konkretnego typu mikrokontrolera był podyktowany wyłącznie koniecznością posiadania układu o możliwie największej dostępnej pojemności pamięci Flash, gdyż piktogramy symbolizujące stan pogody i wzorce czcionek ekranowych zajmują ponad 80% wielkości programu obsługi aplikacji (głównie z uwagi na liczbę tych kolorowych piktogramów i to pomimo ich kompresji).

Listing 6. Funkcja odpowiedzialna za odczyt bieżących wartości wilgotności względnej i temperatury zmierzonych przez układ DTH-22

```
#define DATA_NR PB5
#define DATA_DDR DDRB
#define DATA_PIN PINB
#define RESET_DATA DATA_DDR |= (1<<DATA_NR)  //Ściągnięcie magistrali do „0” poprzez ustawienie kierunku portu jako wyjściowy
#define SET_DATA DATA_DDR &= ~(1<<DATA_NR)  //Zwolnienie magistrali i tym samym podciągnięcie rezystorem do plusa zasilania
#define DATA_IS_SET (DATA_PIN & (1<<DATA_NR))
#define DATA_IS_RESET (!(DATA_PIN & (1<<DATA_NR)))

typedef enum
{
    NO_ERRORS = 0,
    BUS_ERROR = 1,
    NO_DEVICE = 2,
    ACK_TOO_LONG = 3,
    DATA_TIMEOUT = 4,
    CHECKSUM_ERROR = 5,
} DHT22_ERROR_t;

DHT22_ERROR_t readDHT22(uint16_t *Humidity, uint16_t *Temperature)
{
    uint8_t uSec, rawData[5];
    //RH - MSB     RH - LSB       T - MSB         T- LSB       Suma kontrolna
    rawData[4] = rawData[3] = rawData[2] = rawData[1] = rawData[0] = 0;
    //Czekamy na ewentualne zwolnienie magistrali, gdyby była zajęta przez jakiekolwiek peryferia
    uSec = 0;
    while(DATA_IS_RESET && ++uSec != 255) _delay_us(1);
    if(uSec == 255) return BUS_ERROR;
    //Wysyłamy sygnał Start
    RESET_DATA; //Master ściąga magistralę do ‚0’
    _delay_ms(10);  //1...10ms
    SET_DATA;  //Master zwalnia magistralę
    //Czekamy na ściągnięcie magistrali do ‚0’ przez Slave’a - sygnał Response (w ciągu 20-40us)
    uSec = 0;
    do
    {
        //Slave nie ściągnął magistrali do ‚0’ w niezbędnym czasie - błąd transmisji (brak układu)
        if (++uSec > 50) return NO_DEVICE;  //Tutaj: 50us
        _delay_us(1);
    } while(DATA_IS_SET);
    //Slave wyzerował magistralę, więc jest obecny, w związku z czym czekamy, aż zwolni magistralę w czasie 80us
    uSec = 0;
    do
    {
        //Slave nie zwolnił magistrali w niezbędnym czasie - błąd transmisji
        if (++uSec > 100) return ACK_TOO_LONG;  //Tutaj: 100us
        _delay_us(1);
    } while(DATA_IS_RESET);
          //Slave zwolnił magistralę w niezbędnym czasie, więc czekamy 80us na rozpoczęcie transmisji, czyli pierwsze zbocze opadające
    uSec = 0;
    do
    {
        //Slave nie rozpoczął transmisji w czasie 80us
        if (++uSec > 100) return ACK_TOO_LONG;  //Tutaj: 100us
        _delay_us(1);
    } while(DATA_IS_SET);
    //Slave wyzerował magistralę - będzie transmitował kolejne bity danych (40 sztuk). Każdy bit danych złożony jest
    //z dwóch „połówek”: logicznego ‚0’ o czasie trwania 50us i logicznej ‚1’, której czas trwania determinuje wartość bitu.
    for(uint8_t Bit=39; Bit!=255; --Bit)
    {
        //Pierwsza „połówka” transmitowanego bitu (stan ‚0’), która zawsze powinna trwać 50us
        uSec = 0;
        do
        {
            //Błąd transmisji
            if (++uSec > 70) return DATA_TIMEOUT;  //Tutaj: 70us
            _delay_us(1);
        } while(DATA_IS_RESET);
        //Druga „połówka” transmitowanego bitu (stan ‚1’), której długość determinuje wartość bitu - max 70us
        uSec = 0;
        do
        {
            //Błąd transmisji
            if (++uSec > 100) return DATA_TIMEOUT;  //Tutaj: 100us
           _delay_us(1);
        } while(DATA_IS_SET);
        // Jeśli długość drugiej «połówki» > 40us to dekodujemy stan wysoki
        if (uSec > 40) rawData[Bit/8] |= (1<<(Bit-((Bit/8)*8)));
    }

    //Sprawdzamy poprawność odebranych danych porównując ich sumę z bajtem sumy kontrolnej
    if(rawData[0] == ((rawData[4]+rawData[3]+rawData[2]+rawData[1]) &0xFF))
    {
        *Humidity = ((rawData[4]<<8)|rawData[3]);
        *Temperature = ((rawData[2]<<8)|rawData[1]);
    }
    else return CHECKSUM_ERROR;
    return NO_ERRORS;
}
```

Nieprzypadkowo wybrano też typ wyświetlacza TFT. Zastosowałem znany mi moduł RVT28AETNWN00 firmy Rivierdi wyposażony w popularny kontroler ILI9341, który idealnie wpisuje się w nasze wymagania, zarówno jeśli chodzi o parametry techniczne, możliwości, rozmiar, jak i niewygórowaną cenę! Wyświetlacz firmy Rivierdi charakteryzuje się następującymi, wybranymi cechami funkcjonalnymi: przekątna 2,83”, rozdzielczość 320×240 pikseli, obszar aktywny 43,2 mm×57,6 mm, możliwość pracy w trybie pionowym (240×320) lub poziomym (320×240), 65 tys./262 tys. kolorów, wbudowany filtr antyodblaskowy, podświetlenie LED, wiele dostępnych interfejsów podłączeniowych (8-, 9-, 16- i 18-bitowa szyna danych), RGB, SPI/I²C (3- i 4-przewodowy), możliwość rozdzielenia szyny danych od szyny rozkazów dla interfejsu równoległego, możliwość wyposażenia w opcjonalny pojemnościowy lub rezystancyjny panel dotykowy, niewielka grubość modułu (ok. 3 mm), taśma połączeniowa ZIF 50 (raster



Rysunek 4. Schemat montażowy wiStation

0,5 mm), mały pobór mocy, napięcie zasilania 2,5…3,3 V.

Wyświetlacz RVT28AETNWN00 idealnie wpasowuje się w wymagania odnośnie do urządzenia, a dodatkowo obsługę wyświetlacza ułatwia zastosowany w nim sterownik ILI9341. Dokładny opis sposobu obsługi wyświetlacza zamieściłem w artykułach „Komputer samochodowy Mee mkII" opublikowanych w EP 7…9/2016. Zainteresowanych czytelników odsyłam do wspomnianych publikacji.

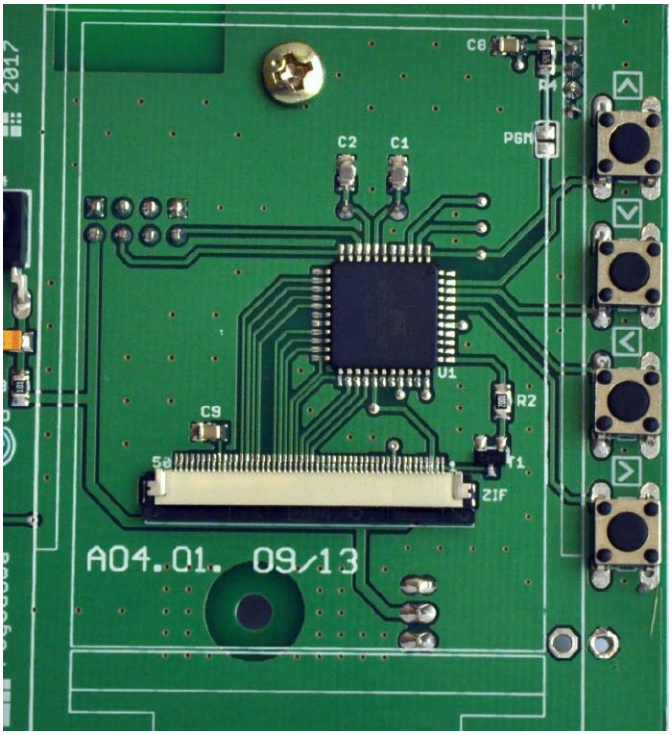
Przejdźmy do sposobu obsługi modułu ESP8266. Jak wspominano, moduł jest sterowany przez mikrokontroler z użyciem interfejsu USART0. Na **listingu 1** pokazano funkcję inicjalizacyjną, której zadaniem jest uruchomienie nadajnika i odbiornika interfejsu USART mikrokontrolera, ustawienie prędkości transmisji (115200 lub 9600 bps) oraz uruchomienie przerwania odbiorczego interfejsu, którego zadaniem będzie odbieranie i analiza danych wysyłanych przez moduł ESP8266. Funkcję, o której mowa, pokazano na **listingu 2**. Kolejne funkcje realizują wysłanie znaku, łańcucha z pamięci RAM i łańcucha z pamięci Flash mikrokontrolera – przedstawiono je na **listingu 3**.

Na **listingu 4** pokazano funkcję obsługi przerwania odbiornika interfejsu USART (USART0_RX_vect), której zadaniem jest obsługa odpowiedzi modemu ESP8266 oraz analiza i dekodowanie danych pogodowych wysyłanych przez serwer ***Wunderground.com***, które funkcja ta umieszcza w specjalnej strukturze danych zdefiniowanej na potrzeby projektu. Jak widać, funkcja korzysta z innej funkcji narzędziowej o nazwie *determineWeather*, której zadaniem jest ustalenie rodzaju pogody na podstawie wyrazów znajdujących się w linii reprezentującej stan pogody. Funkcję *determineWeather* prezentuje **listing 5**.

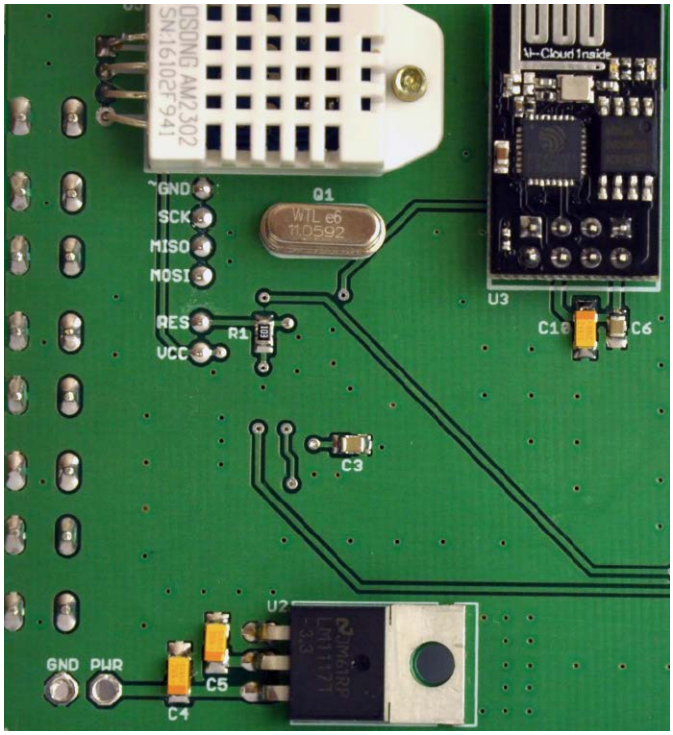
To tyle, jeśli chodzi o obsługę modemu ESP8266 w ramach naszego projektu, jednak urządzenie wiStation wykorzystuje jeszcze dwa ciekawe peryferia w celu realizacji założonej funkcjonalności, o których nie sposób nie wspomnieć. Po pierwsze i o czym wspomniano we wstępie, w ramach interfejsu użytkownika zbudowano barierę podczerwieni pracującą z częstotliwością nośną 38 kHz, która składa się z nadajnika podczerwieni sterowanego z wyjścia OC0B Timera0 pracującego w trybie CTC, którego zadaniem jest generowanie cyklicznego przebiegu prostokątnego o częstotliwości 38 kHz sterującego diodą nadawczą podczerwieni oraz odbiornika podczerwieni TSOP34838, który odbierając taką transmisję, generuje przerwanie zewnętrzne na wejściu PCINT4 mikrokontrolera (przerwanie *Pin Change Interrupt 0*). Zadaniem tego mechanizmu jest automatyczne sterowanie podświetleniem wyświetlacza TFT, jeśli jest on obserwowany przez użytkownika. W takim wypadku emitowana wiązka podczerwieni odbija się od postaci użytkownika i trafia do odbiornika podczerwieni, sterując załączeniem podświetlenia. Jest ono „podtrzymywane" przez 10 sekund po ustąpieniu zdarzenia. Innym ciekawym elementem jest scalony dokładny higrometr/termometr DTH-22 (lub AM2302) sterowany za pomocą 1-przewodowej, szeregowej magistrali danych zbliżonej do 1-Wire. Dzięki niemu mogłem wyposażyć stację pogodową w pomiar wilgotności i temperatury wewnętrznej pomieszczenia, w którym jest używane urządzenie. Jako że jest to rozwiązanie niezmiernie ciekawe a przy okazji niezgodne z dość dobrze znanym standardem 1-Wire, warto opisać zastosowane rozwiązania programowe. Podobnie jak w przypadku magistrali 1-Wire, interfejs komunikacyjny zastosowany w czujniku DTH-22 jest złożony z 1 linii danych oznaczonej SDA, podciągniętej do plusa zasilania poprzez rezystor pull-up. Transmisja danych jest inicjowana **wyłącznie** przez układ Master (mikrokontroler) poprzez wygenerowanie sygnału *Start*, czyli wyzerowanie magistrali przez 1 ms (dopuszczalny zakres 0,8…20 ms), po czym zwolnienie (ustawienie) magistrali i oczekiwanie na sygnał odpowiedzi od układu Slave. W tym momencie układ Slave powinien w czasie 20…40 µs wygenerować sygnał odpowiedzi tzw. *Response* przez wyzerowanie magistrali na czas 80 µs, po czym jej zwolnienie przez 80 µs. W ten sposób układ podrzędny potwierdza swoją obecność na magistrali układowi nadrzędnemu i jednocześnie informuje go, że jest gotowy do transmisji danych. Sekwencję sygnałów sterujących na magistrali danych podczas startu transmisji danych pokazano na **rysunku 2**.

Dalej, w odróżnieniu od standardu 1-Wire, inicjatywę przejmuje układ Slave, ponieważ to on rozpoczyna i kończy transmisję każdego bitu danych, sprowadzając niejako rolę układu Master do roli układu podrzędnego, którego zadaniem jest wyłącznie nasłuchiwanie transmitowanych danych. Każdy bit danych składa się w tym wypadku z dwóch „połówek": pierwszej, w której ramach układ Slave zeruje magistralę przez czas 50 µs i drugiej, w której ramach magistrala jest zwalniana na czas 26 µs przy przesyłaniu (kodowaniu) logicznego zera lub 70 µs w wypadku przesyłania logicznej jedynki. Poszczególne bity przesyłane są w ramach bajtu danych, zawsze począwszy od bitu najbardziej znaczącego. Sekwencję sygnałów sterujących na magistrali danych podczas przesyłania bitów danych pokazano na **rysunku 3**.

Po przesłaniu ostatniego z bitów danych układ Slave ściąga magistralę SDA do logicznego zera na czas 50 µs, po czym zwalnia ją, sygnalizując koniec transmisji całego pakietu danych. Układ DTH-22 każdorazowo transmituje 5 bajtów (40 bitów) danych, z których dwa pierwsze bajty reprezentują 16-bitową liczbę bez znaku, która przechowuje wilgotność względną w jednostce



Fotografia 5. Wygląd płytki drukowanej stacji wiStation od strony elementów (bez wyświetlacza TFT)



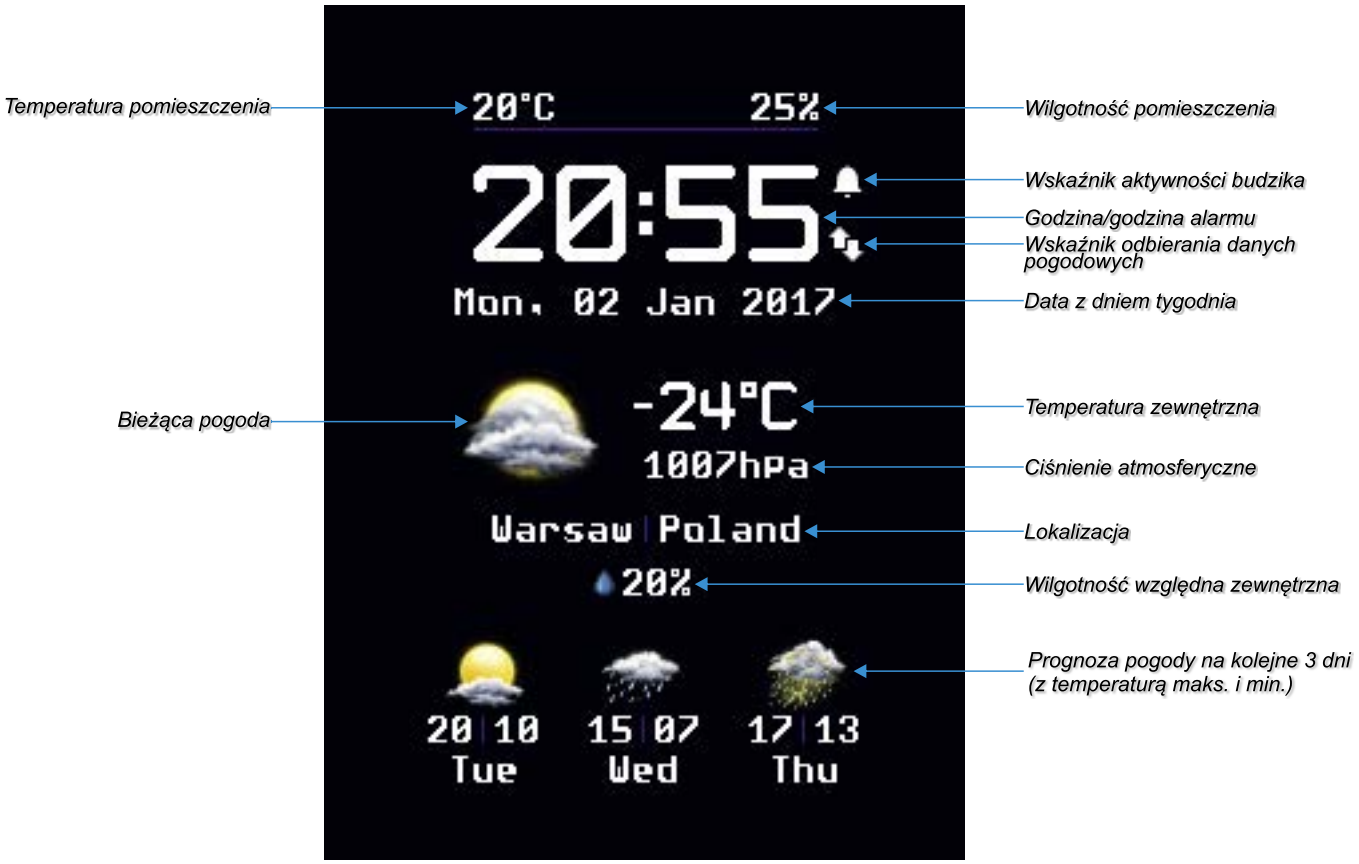
Fotografia 6. Wygląd płytki drukowanej stacji wiStation od strony wyprowadzeń

równej 0,1% (czyli, dla przykładu, liczba 455 oznacza wilgotność względną 45,5%). Dwa kolejne bajty reprezentują 16-bitową liczbę ze znakiem (znak jest kodowany przez najstarszy bit), która przechowuje temperaturę w jednostce równej 0,1°C (czyli, dla przykładu, liczba 324 oznacza temperaturę 32,4°C), zaś ostatni bajt stanowi sumę kontrolną obliczaną jako suma algebraiczna poprzednich czterech bajtów danych i pozwala na sprawdzenie poprawności transmisji. Warto zauważyć, że ujemne temperatury kodowane są w inny sposób, niż to zwyczajowo przyjęto. Nie jest to zapis w tzw. kodzie dopełnienia do dwóch U2, tylko zwyczajny zapis binarny, dla którego ustawiony najstarszy bit (bit 15) oznacza temperaturę ujemną i tak, dla przykładu, zapis 1000000001100101 oznacza temperaturę –10,1°C. Kolejną istotną informacją jeśli maksymalna częstotliwość odczytu danych pomiarowych, która nie może przekraczać 0,5 Hz, co oznacza, że nie należy odpytywać czujnika o dane pomiarowe częściej niż raz na 2 sekundy. Na koniec pozostaje przedstawić prostą implementację funkcji odpowiedzialnej za odczyt bieżących wartości wilgotności względnej i temperatury zmierzonych przez układ DTH-22 – pokazano ją na **listingu 6**.

Montaż

Schemat montażowy układu **wiStation** pokazano na **rysunku 4**. Z uwagi na fakt, iż moduł wyświetlacza TFT podłączony jest do płytki naszego urządzenia z użyciem gniazda ZIF o bardzo gęstym rastrze (50 wyprowadzeń co 0,5 mm) montaż urządzenia rozpoczynamy właśnie od przylutowania tego gniazda. Najprostszym sposobem montażu elementów o tak dużym zagęszczeniu wyprowadzeń, niewymagającym jednocześnie posiadania specjalistycznego sprzętu, jest użycie typowej stacji lutowniczej, cyny z odpowiednią ilością topnika oraz dość cienkiej plecionki rozlutowniczej, która umożliwi usunięcie nadmiaru cyny spomiędzy wyprowadzeń złącza. Należy przy tym uważać, by nie uszkodzić termicznie złącza. Jakość tak wykonanego połączenia sprawdzamy pod lupą, korzystając z miernika ciągłości połączeń. Wspomniana kontrola będzie znacznie łatwiejsza, jeśli zmontowaną płytkę sterownika przemyjemy alkoholem izopropylowym w celu wypłukania nadmiaru kalafonii lutowniczej.

Następnie montujemy elementy na warstwie górnej, czyli mikrokontroler, elementy półprzewodnikowe SMD, elementy bierne a na samym końcu wszystkie elementy przeznaczone do montażu przewlekanego. Przechodzimy na warstwę spodnią, gdzie lutujemy w pierwszej kolejności elementy SMD, a na samym końcu duże podzespoły do montażu przewlekanego, jak czujnik DTH-22 (przykręcony na śrubkę), modem ESP-8266-01, rezonator kwarcowy i stabilizator LM-1117-3.3V. Na samym końcu dołączamy taśmę wyświetlacza TFT do złącza ZIF, dbając o odpowiednie zablokowanie zatrzasków, zaś sam wyświetlacz przyklejamy do płytki sterownika, korzystając z pasków dwustronnej taśmy samoprzylepnej. Aby to było możliwe, należy przykleić wąskie paski (o szerokości około 2 mm) twardego materiału do płytki naszego sterownika w miejscach oznaczonych specjalnym nadrukiem, które będą stanowić dystans między płytką sterownika a modułem TFT niezbędny z powodu montażu na nim elementów SMD. Po przyklejeniu tychże dystansów, wspomniany moduł wyświetlacza mocujemy we wcześniej omówiony sposób. Na **fotografii 5** pokazano wygląd obwodu drukowanego zmontowanego urządzenia **wiStation** od strony elementów (bez wyświetlacza TFT). Warto zauważyć, że w jego dolnej części przewidziano podłużne, odseparowane od masy pole miedzi, do którego możemy przylutować pod kątem 50–60° kawałek laminatu stanowiący niejako stopkę umożliwiającą wygodne postawienie całego urządzenia. Na **fotografii 6** pokazano wygląd obwodu drukowanego zmontowanego urządzenia **wiStation** od strony wyprowadzeń. Warto zauważyć, że po zaprogramowaniu mikrokontrolera należy zewrzeć przy użyciu kropli cyny zworkę oznaczoną PGM, co umożliwi obsługę czujnika DTH-22.



Rysunek 7. Wygląd graficznego interfejsu użytkownika urządzenia wiStation

Obsługa

Konstruując stację pogodową, chciałem, aby formą prezentacji danych i możliwościami dorównywała rozwiązaniom znanym choćby z telefonii komórkowej. Podstawowym zadaniem było w takim razie zaprojektowanie czytelnego i ładnego w swojej formie interfejsu użytkownika z wykorzystaniem efektownych, kolorowych elementów graficznych. Nie powiem, że było to zadanie łatwe, bo w istocie konstrukcja interfejsu graficznego zajęła mi najwięcej czasu, wliczając w to wykonanie wielu drobnych grafik (od nowa lub przetworzenie istniejących i znalezionych w Internecie elementów). Interfejs graficzny użytkownika pokazano na **rysunku 7**, który, mam nadzieję, czytelnie prezentuje wszystkie interesujące informacje.

Do obsługi urządzenia przewidziano 4 mikroprzyciski umownie oznaczone *UP, DOWN, NEXT, PREV*, których funkcjonalność zależy od aktualnie obsługiwanej funkcji. Dodatkowo przewidziano specjalny tryb konfiguracyjny, który pozwala na edycję podstawowych danych niezbędnych z punktu widzenia połączenia z siecią Wi-Fi i współpracy z serwisem ***Wunderground.com***. Wspomniane dane to:

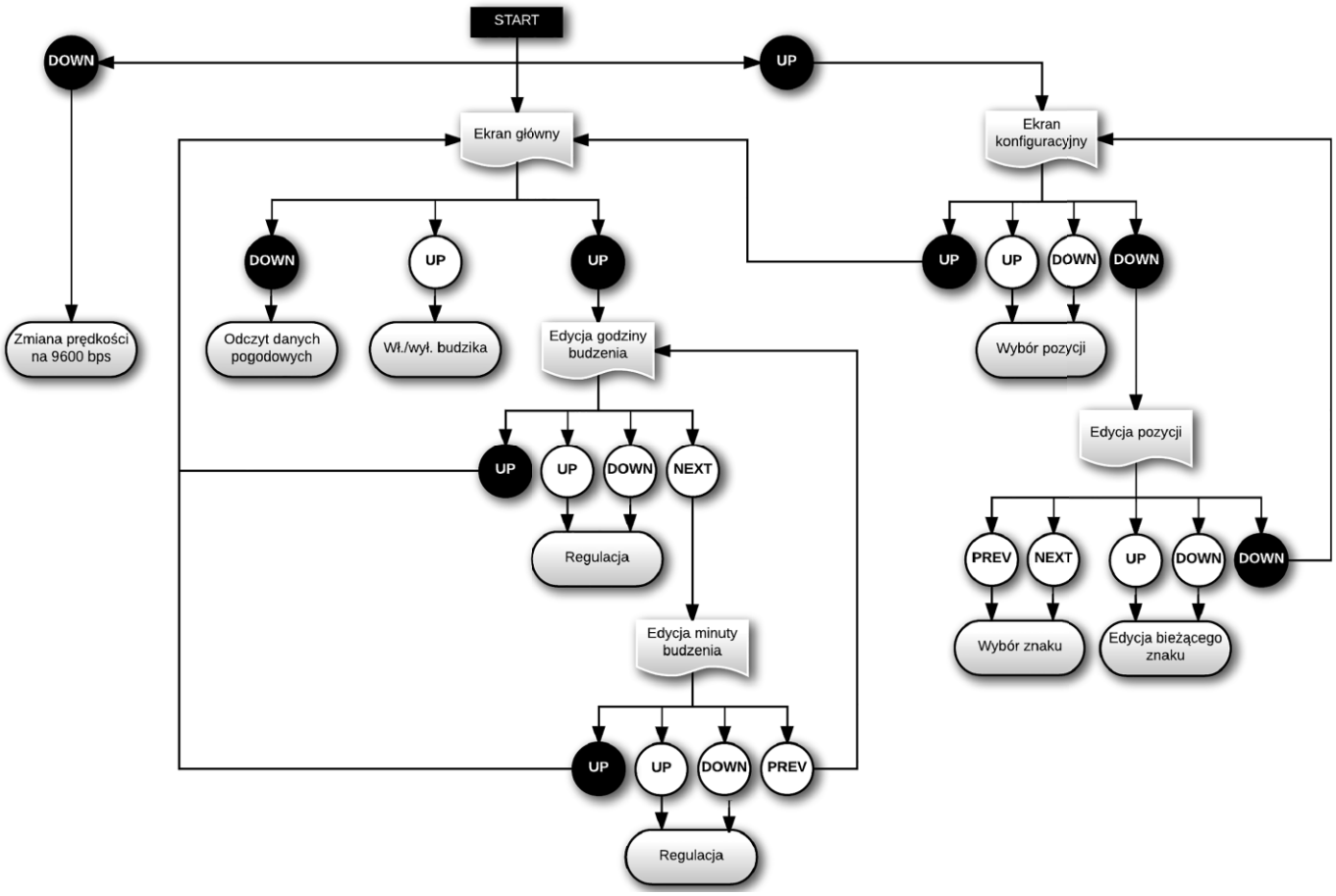
- Nazwa sieci Wi-Fi (SSID).
- Hasło WEP/WPA/WPA2 sieci Wi-Fi.
- Nazwa kraju i miejscowości, której dotyczy prognoza pogody (w języku angielskim).
- Unikalny klucz Key ID pozyskany od serwisu pogodowego.

Wygląd wspomnianego menu konfiguracyjnego pokazano na **rysunku 8.** Wejście w menu konfiguracyjne jest możliwe wyłącznie podczas włączania urządzenia poprzez wciśnięcie przycisku *UP* w trakcie jego uruchamiania lub w sposób automatyczny w wypadku, gdy jeszcze nie skonfigurowano nazwy sieci. Każdorazowo, podczas procesu łączenia się z siecią Wi-Fi (a więc po włączeniu zasilania lub utracie łączności) urządzenie **wiStation** pokazuje ekran połączenia widoczny na **rysunku 9**, gdzie oprócz logo „wifi" jest pokazana animacja pozwalająca na zorientowanie się w zakresie przebiegu procesu połączenia.

Funkcjonalność przycisków przeznaczonych do obsługi stacji pogodowej zależy od aktualnie realizowanej funkcji. Rysunek obrazujący

Rysunek 8. Wygląd menu konfiguracyjnego urządzenia wiStation

Rysunek 9. Wygląd ekranu urządzenia wiStation w czasie łączenia się z siecią Wi-Fi



Rysunek 10. Organizacja menu i sposób obsługi urządzenia wiStation

system Menu oraz sposób obsługi urządzenia (symbole przycisków wypełnione kolorem czarnym oznaczają długie naciśnięcie wybranego przycisku) pokazano na **rysunku 10**.

Podczas pierwszego uruchomienia urządzenia należy przytrzymać wciśnięty przycisk *DOWN*, co spowoduje przestawienie (i jednoczesne zapamiętanie) domyślnej prędkości interfejsu UART modułu ESP8266 z 115 200 bps na 9600 bps. Jest to niezbędne dla poprawnej pracy urządzenia. Uruchomieniu urządzenia towarzyszy wywołanie ekranu połączenia z siecią Wi-Fi, a następnie pokazanie ekranu głównego, na którym, po krótkiej chwili niezbędnej na pobranie danych pogodowych z serwisu ***Wunderground.com***, powinniśmy zobaczyć wszystkie interesujące nas informacje. Manualna inicjacja pobrania danych pogodowych możliwa jest każdorazowo poprzez długie wciśnięcie przycisku *DOWN*, jednak należy mieć tutaj na uwadze ograniczenie liczby możliwych zapytań wysyłanych do serwera, o którym pisałem wcześniej oraz fakt, że ten proces jest powtarzany automatycznie z interwałem 5 minut. Dodatkową funkcjonalnością urządzenia jest funkcja alarmu/budzika, której obsługę umożliwia menu główne aplikacji i która nie wymaga dodatkowego komentarza.

Ustawienie fusebitów:
CKSEL3...0: 1111
SUT1...0: 11
CKDIV8: 1
CKOUT: 1
JTAGEN: 1
EESAVE: 0

Robert Wołgajew, EP