Amatorska stacja pogodowa

Jesteśmy elementem natury, a wiele z podejmowanych przez nas działań ma związek z pogodą. Dlatego myślę, że ludzi od zawsze interesowało przepowiadanie pogody. Mimo rozwoju technologicznego, póki co prognozowanie pogody jest obarczone dużym błędem i nie wiemy, co jaka pogoda na pewno będzie jutro czy za kilka dni. Jako amatorzy nie jesteśmy w stanie przewidywać zjawisk pogodowych nawet w kilku procentach, ale możemy monitorować pogodę na bieżąco i wyciągać swoje mniej lub bardziej trafne wnioski. Projekt urządzenie, które chcę przedstawić, służy właśnie do takich celów.

Rekomendacje: *nietuzinkowa stacja pogodowa, którą – w odróżnieniu od innych urządzeń tego typu – wyposażono w kilka użytecznych gadżetów.*

Urządzenie jest złożone z dwóch bloków:

1. Płytki czujników temperatury, ciśnienia, wilgotności, sensora wyładowań, prędkości i kierunku wiatru oraz ilości opadów umieszczonej na zewnątrz budynku.
2. Interfejsu użytkownika z wyświetlaczem TFT 4,3”, na którym są prezentowane wszystkie mierzone wartości.

Schemat ideowy płytki czujników pokazano na **rysunku 1**. Zastosowane następujące czujniki: czujnik temperatury DS18B20, czujnik ciśnienia BMP280, czujnik wilgotności SI7021 (HUT21), sensor burzy – układ scalony AS3935, czujnik kierunku i prędkości wiatru oraz ilości opadów zakupione w firmie Maplin w Wielkiej Brytanii. Można je kupić również za pomocą sklepu internetowego ***http://sklep.meteoplus.pl***, ponieważ Maplin raczej jest nastawiony na sprzedaż do dystrybutorów i nie zawsze chce przesłać do Polski paczkę z pojedynczymi czujnikami.

Czujnik prędkości wiatru generuje jeden impuls na obrót, co według producenta daje wartość 1,492 mph, a więc w zaokrągleniu 2,4 km/godz. Impulsy są zliczane w procedurze obsługi przerwania. Następnie funkcja wywoływana co 3 sek. oblicza wartość i zeruje zmienną

Dodatkowe materiały do pobrania ze strony www.media.avt.pl

W ofercie AVT* AVT-5654

Podstawowe parametry:
- Odczyt temperatury wewnątrz i na zewnątrz pomieszczenia.
- Odczyt ciśnienia i wilgotności oraz ilości opadów dobowych.
- Pomiar prędkości i kierunku wiatru w czasie rzeczywistym.
- Monitorowanie odległości do wyładowań atmosferycznych.
- Wyświetlanie aktualnej godziny, daty oraz godzin wschodu i zachodu słońca dla danej szerokości geograficznej.
- Informacja głosowa o pełnej godzinie – czas, temperatura i ciśnienie.

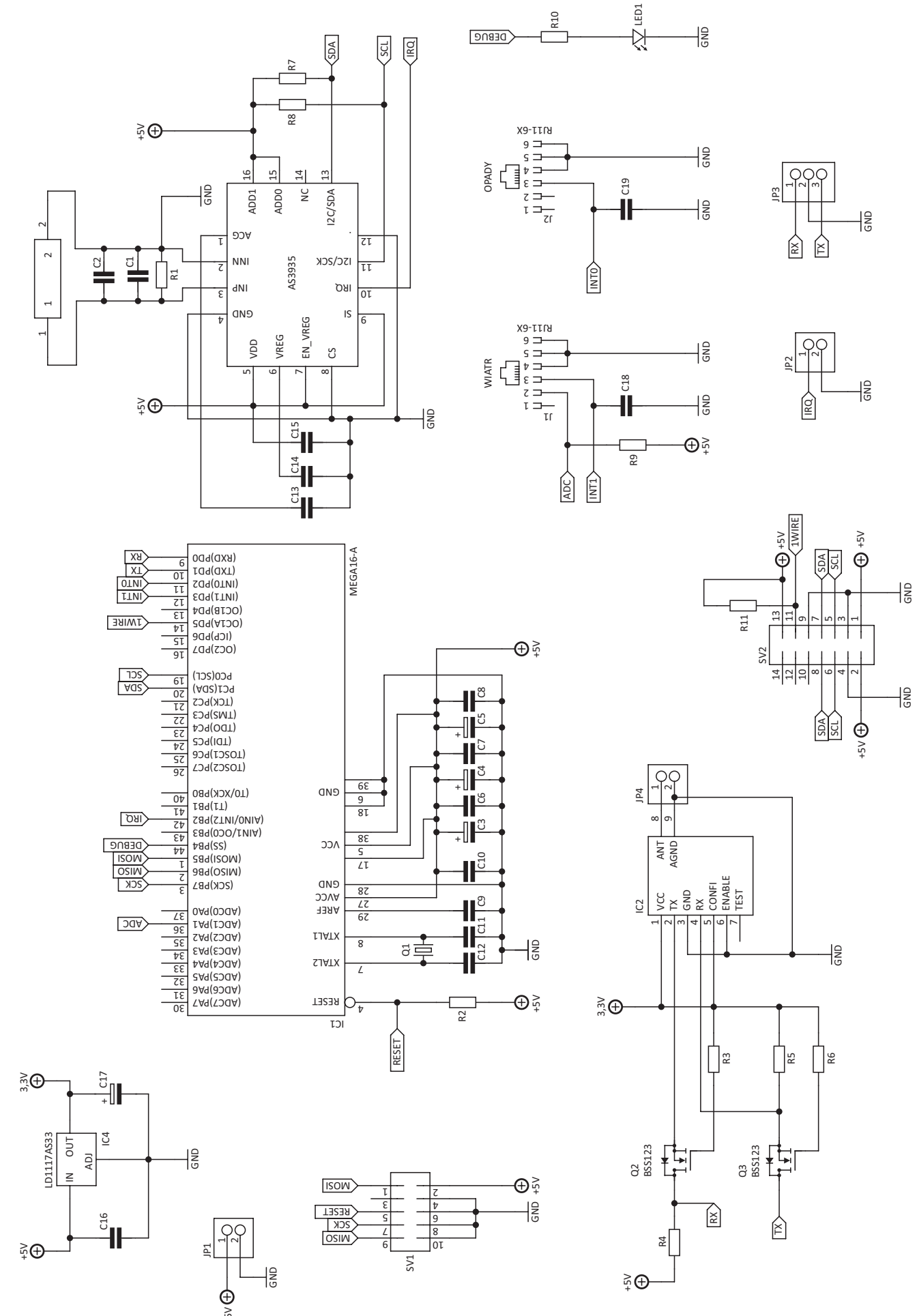
Projekty pokrewne na www.media.avt.pl:

AVT-5639	Bezprzewodowy czujnik warunków atmosferycznych (EP 10/2018)
AVT-5605	wiStation – domowa stacja pogodowa z prognozą pogody (EP 9/2017)
AVT-5566	THPStation – rozbudowany termometr z Wi-Fi (EP-1/2017)
AVT-5489	8-kanałowy termometr z alarmem i wyświetlaczem LCD (EP 11/2013)
AVT-961	Domowa stacja pogodowa (EP 12/2006)
AVT-957	Moduł pomiaru temperatury (EP 11/2006)

*** 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***. W przypadku braku dostępności na ***http://sklep.avt.pl***, osoby zainteresowane zakupem płytek drukowanych (PCB), prosimy o kontakt via email: kity@avt.pl.



Rysunek 1. Schemat ideowy płytki czujników

Listing 1. Zliczanie impulsów w przerwaniu INT1 i obliczenie prędkości wiatru

```
//obsługa przerwania zewnętrznego od INT1
ISR(INT1_vect)
{
    b++;  //licznik impulsów wiatromierza
}

//obliczanie prędkości wiatru
void wind_speed(void)
{
    speed = b*240/3;  //liczba impulsów jest mnożona przez 240, aby działania
                      //odbywały się na liczbach całkowitych; /3 bo funkcja
    speed_dz = speed/100; //wywoływana co 3 sek.  1 imp. = 2,4 km/godz.
    speed_u = speed/10%10;
    b=0;  //zerowanie licznika impulsów
}
```

Listing 2.Wyzmaczanie kierunku wiatru

```
//pomiar za pomocą przetwornika A/C w kanale „kanal”
uint16_t pomiar_ADC(uint8_t kanal)
{
    ADMUX = (ADMUX & 0xf8)| kanal;
    ADCSRA |=(1<<ADSC);  //start pomiaru
    while(ADCSRA & (1<<ADSC));
    adc=ADCW;
    return adc;
}

//sprawdzenie, który kontaktron zadziałał
void wind_dir(void)
{
    for(uint8_t i=0; i<8; i++)
    {
        wzr=pgm_read_word(&adc_keys[i]);        // porównanie z wzorcem ADC
        if(adc>wzr-ADC_RANGE && adc<wzr+ADC_RANGE)
        {
            if (keys[i]<3) keys[i]++;
        }
        else keys[i]=0;
    }
    //przypisanie wartości dla poszczególnych pozycji wiatru
    if(keys[0]==2) dir = 0;  // N
    if(keys[1]==2) dir = 1;  // NE
    if(keys[2]==2) dir = 2;  // E
    if(keys[3]==2) dir = 3;  // SE
    if(keys[4]==2) dir = 4;  // S
    if(keys[5]==2) dir = 5;  // SW
    if(keys[6]==2) dir = 6;  // W
    if(keys[7]==2) dir = 7;  // NW
    tab[i]=dir;  // wypełnianie tablicy próbkami kierunku wiatru
    i++;
    if(i>10) i=0;
    Dominanta();
}
```

Listing 3. Obsługa przerwania INT0

```
//zwiększanie licznika przerwań
ISR(INT0_vect)
{
    a++;
}
//obliczenie ilości opadów
void amount_rain(void)
{
    rain = a*30;  // pozbywamy się ułamków w obliczeniach
    rain_dz = rain/100;  // wartość przed przecinkiem
    rain_u = rain/10%10; // wartość po przecinku
}
```

„b” (**listing 1**). Minimalna prędkość mierzona to 0,8 km/godzinę. Jeżeli chcielibyśmy uzyskać większą rozdzielczość, to należałoby zsumować liczbę impulsów np. z 4 sekund – wtedy minimalna wyznaczona prędkość mogłaby wynosić 0,6 km/godz. Oczywiście, funkcję wywołujemy co 4 sekundy.

Czujnik kierunku wiatru jest wykonany z 8 kontaktronów z dzielnikami rezystorowymi. Możliwy jest pomiar 16 kierunków, ale ja przyjąłem 8 podstawowych. Każdy kontaktron po zwarciu zmienia parametry dzielnika rezystorowego, a co za tym idzie, wartość napięcia na wyjściu. Kierunek określamy mierząc napięcie z czujnika za pomocą przetwornika A/C (**listing 2**). Wynik pomiaru jest następnie przekazywany jest do funkcji void *wind_dir(void)*. W ciągu 3 sekund jest wykonywane 10 pomiarów ,a ostateczny wynik obliczany jest za pomocą dominanty.

Czujnik ilości opadów dokonuje opadu metodą korytkową a każde napełnienie i przechył korytka to 0,279 l/m². Ze względów praktycznych zaokrągliłem tą wartość do 0,3 l/m². Impulsy zliczane są w przerwaniu INT0 a następnie funkcja oblicza ilość opadów (**listing 3**). Zerowanie ilości opadów jest wykonywane automatycznie, o północy.

Schemat montażowy płytki czujników zamieszczono na **rysunku 2**.W urządzeniu prototypowym płytkę zamontowano w obudowie o stopniu szczelności IP65. Mimo szczelności do wnętrza obudowy warto włożyć torebkę z pochłaniaczem wilgoci. Oczywiście, czujnik wilgotności i temperatury jest zamontowany poza obudową. Płytka może być zasilana napięciem 5 V lub 3,3 V. Przy zasilaniu z 3,3 V nie trzeba montować stabilizatora napięcia, więc miedzy wejściem i wyjściem należy zrobić zworkę. Trzeba też zmniejszyć do 2,2 kΩ wartość rezystancji R11, przez którą jest zasilana szyna 1-Wire. Zasilanie płytki z 3,3 V pozwala również na uniknięcie konieczności stosowania

Wykaz elementów:

Płytka czujników

Rezystory: (SMD 0805)
R1, R2,R9, R10: 10 kΩ
R3, R4, R5, R6: 2,2 kΩ
R7, R8, R11: 4,7 kΩ
R11: 2,2 kΩ

Kondensatory: (SMD 0805)
C1: 270 pF lub 1 nF
C2: 680 pF lub brak jeżeli C1 = 1 nF
C3, C4, C5, C17: 47 µF
C6, C7, C8, C9, C10, C15, C16: 100 nF
C11, C12: 22 pF
C13: 10 µF
C14: 1 µF
C18, C19: 10 nF

Półprzewodniki:
IC3: AS3935
IC4: LD1117/3,3
Q2, Q3: BSS123
IC1: ATMega 644p

Inne:
Q1: oscylator smd 16 MHz
IC2: moduł radiowy HM-TRP
L1: MA5532-AE
J1, J2: RJ11-6X
JP1…JP4: gold piny
SV1, SV2: dowolne złącze w rastrze 2,54
LED1: dowolna dioda LED smd wielkość 1206
DS18B20 – temperatura, moduły BMP280 – ciśnienie, SI7021 – wilgotność, wszystkie ele. RC – smd w rozmiarze 0805

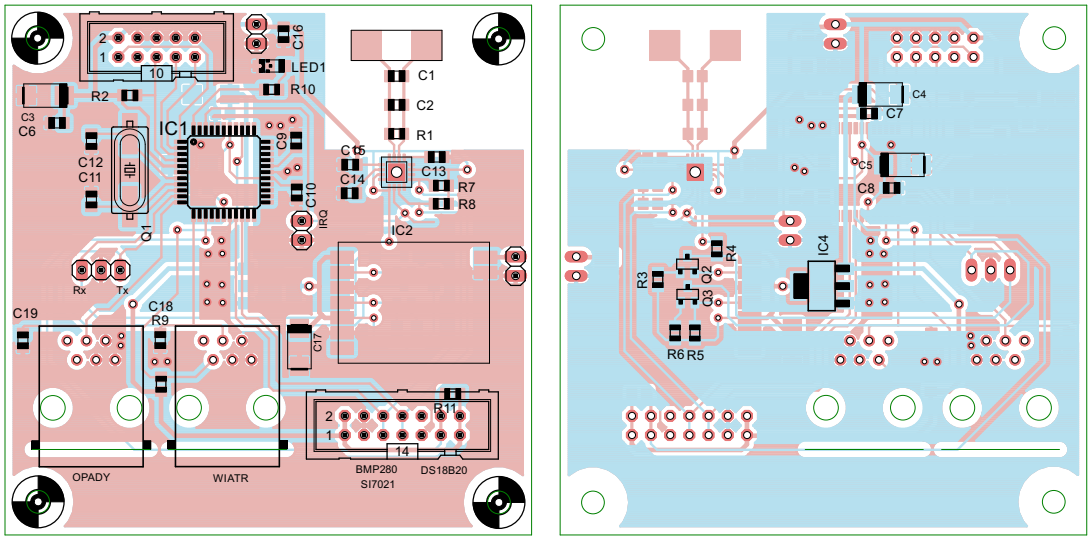
Płytka interfejsu użytkownika

Rezystory: (SMD 0805)
R1, R5, R14: 10 kΩ
R2…R4, R6: 4,7 kΩ
R7…R10, R13: 2,2 kΩ
R11, R12: dobrać dzielnik do fotorezystora
R15, R17, R18: 1 kΩ
R16: 10 Ω
R19, R20: 22 kΩ (fotorezystor PGM5537 16-50 kΩ

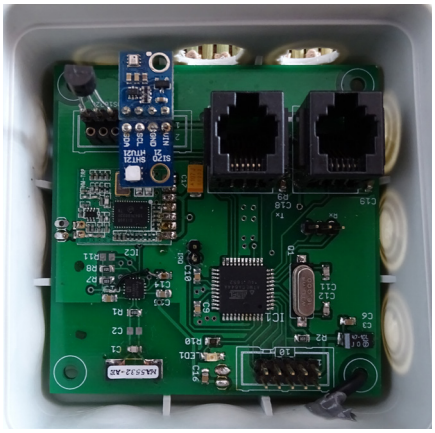
Kondensatory: (SMD 0805)
C1, C2, C24, C25: 22 pF
C3…C5, C9, C10, C13, C16, C19, C23, C26, C27, C29, C30, C35: 100 nF
C6…C8, C20, C21, C28, C31, C34: 47 µF
C11, C17, C18: 1 µF (ceram.)
C12: 100 pF
C14, C15: 100 µF (tantalowy LOW ESR)
C22: 1 nF
C32: 220 µF (elektrolit.)
C33: 10 µF

Półprzewodniki:
Q2, Q3: BSS127 (SOT23)
Q4: BC807 (SOT23)
IC3 :ATmega1284
IC4: ATmega328p
IC5: DS3231M
IC6: LM386M
IC7: TS1117BCP3,3

Inne:
L1, L2: 10 µH (SMD 0805)
Q1, Q5: rezonator 16 MHz
IC1: moduł GSM M590
IC2: moduł radiowy HM-TRP
Diody LED – dowolne SMD 1206
Złącze karty SD – 112A-TAAR R03-MICROSD
Złącze karty SIM – standardowe
Złącza i goldpiny R=2,54
Bateria do DS3231 – CR2032H pozioma



Rysunek 2. Schemat montażowy płytki czujników



Fotografia 3. Płytka czujników zamontowana w obudowie

translatora poziomów napięcia na szynach I²C i UART.

Złącze czujników ciśnienia, wilgotności i temperatury zaprojektowano w taki sposób, że można w przyszłości rozbudować stację o kolejne czujniki dołączone do I²C (np. do pomiaru nasłonecznienia, zapylenie powietrza i inne).

Na **fotografii 3** pokazano zmontowaną płytkę czujników zawierającą układ sensora burzy z obwodem detekcji wyładowań ,moduł radiowy HM-TRP firmy Hoperf Electronic pracujący na częstotliwości 868 MHz, moduły czujników wilgotności i ciśnienia, czujnik temperatury oraz złącza RJ11 do czujników wiatru i opadów.

Moduł radiowy przekazuje cyklicznie odczyty z czujników do głównego procesora. Moduł ten został wybrany nieprzypadkowo ze względu na prostotę implementacji w układzie. Komunikacja z procesorem odbywa się przez interfejs UART – resztę „załatwia" sam moduł. W otwartym terenie połączenie między modułami radiowymi bez anten to ok. 30 m. Warto jednak zastosować antenę wykonaną z kawałka przewodu o długości około 8,5 cm.

Odczyty z czujników wykonywane są cyklicznie w różnych odstępach czasu. Do odmierzania czasu zastosowano wygodne

Listing 4. Pomiar temperatury za pomocą DS18B20

```
void timer0_tick( TSTIMER * tmr )
{
    static uint8_t licznik;
    if(1 == licznik) DS18X20_start_meas( DS18X20_POWER_EXTERN, NULL ); // pomiar
    if(2 == licznik) (  DS18X20_read_meas(gSensorIDs[0], &subzero, &cel, &cel_fract_bits)
); // odczyt
    licznik++;
    if(licznik > 2) licznik = 0;
}
```

Listing 5. Wysłanie obliczonej ilości opadów

```
if(rain_uart!=rain_u) // jeżeli ostatnia wartość jest różna od odczytanej
{
    rain_uart=rain_u;  // przypisujemy wartość odczytaną do zmiennej do wysłania
    sprintf(bufor,(„AT+rain %d, %d „),rain_dz, rain_uart);// napełniamy bufor
    uart_puts(bufor);  // wysyłamy
    uart_puts(„\r\n");
}
```

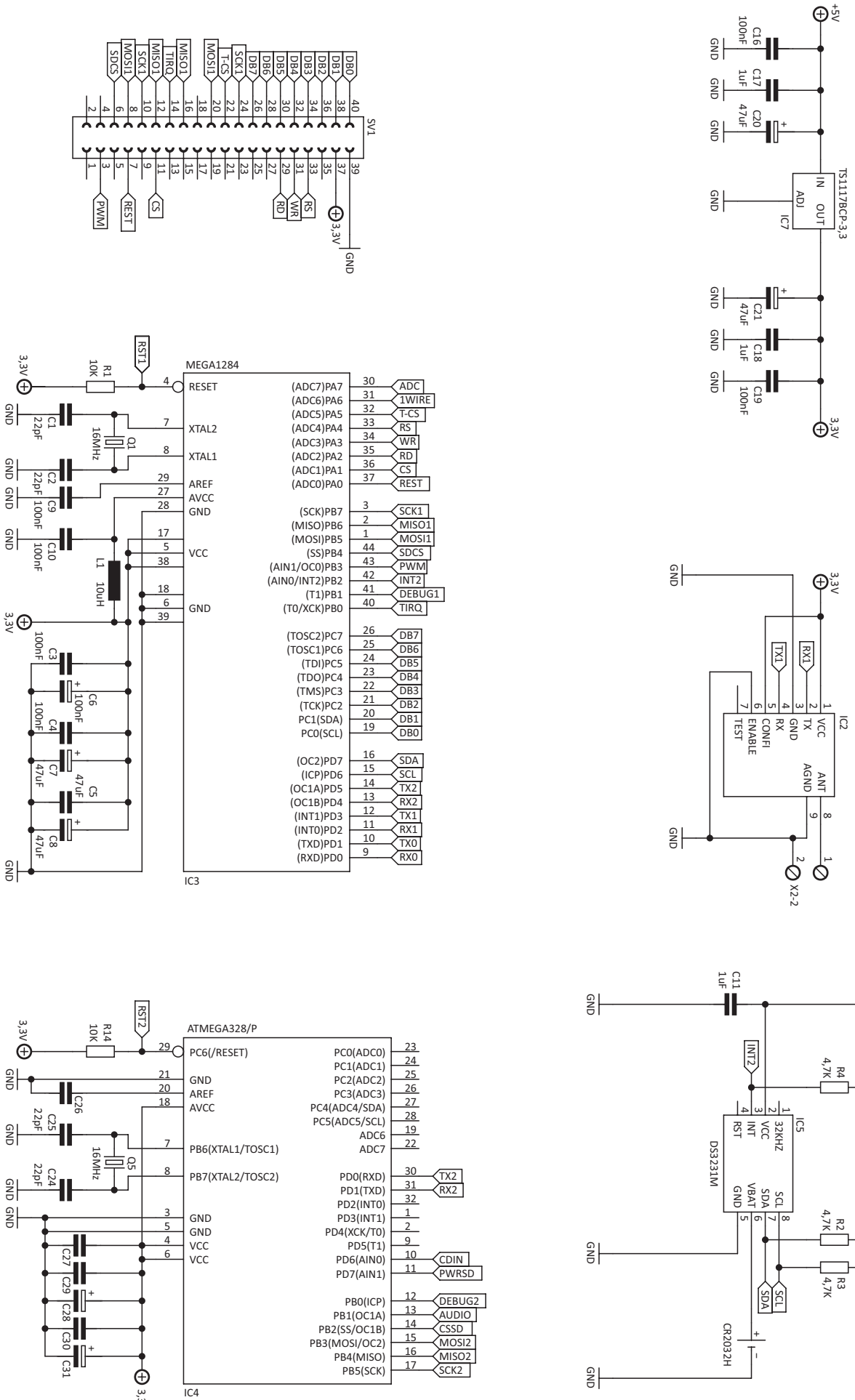
Listing 6. Program główny

```
int main(void)
{
    DDRB|= (1<<PB4);
    TIMER_init();
    INT_init();
    ADC_init();
    i2cSetBitrate( 100 );
    USART_Init(__UBRR);
    BMP280_init();
    AS3935_init();
    czujniki_cnt = search_sensors();
    timer_init( 0, 800,  1, timer0_tick );
    timer_init( 1, 5000, 1, timer1_tick );
    timer_init( 2, 3000, 1, timer2_tick );
    timer_init( 3, 200,  1, timer3_tick );
    timer_init( 4, 2000, 1, timer4_tick );
    timer_init( 5, 3000, 1, timer5_tick );
    register_uart_str_rx_event_callback( Reset );
    sei();
    while(1)
{
        UART_RX_STR_EVENT(uart_buf);
        TIMERS_EVENT();
        if(FLAG)Storm();
    }
}
```

w użyciu timery programowe (**listing 4**). Ciśnienie, wilgotność i ilość opadów odczytywana jest co 5 sekund, natomiast dane z pomiaru siły i kierunku wiatru co 3 sekundy. Pomiar kierunku wiatru odbywa się na zasadzie dominanty (w czasie 3 sekund wykonywane jest 10 pomiarów) i dopiero wtedy wynik przekazywany jest do wysłania do procesora głównego. Siła wiatru to średnia z 3 pomiarów.

Dane odnośnie do temperatury, ciśnienia, wilgotności i opadów przekazywane są do głównego procesora dopiero w momencie zmiany wartości, natomiast dane o wietrze przekazywane są cyklicznie. Przykład wysłania danych zawierających opady pokazano na **listingu 5**.

W momencie wykrycia wyładowania atmosferycznego przez sensor burzy dane o odległości przekazywane są natychmiast. Szczegółowy opis działania sensora burzy AS3935 można znaleźć w październikowym wydaniu EP. Procesor na płytce czujników to ATMega644, ale można

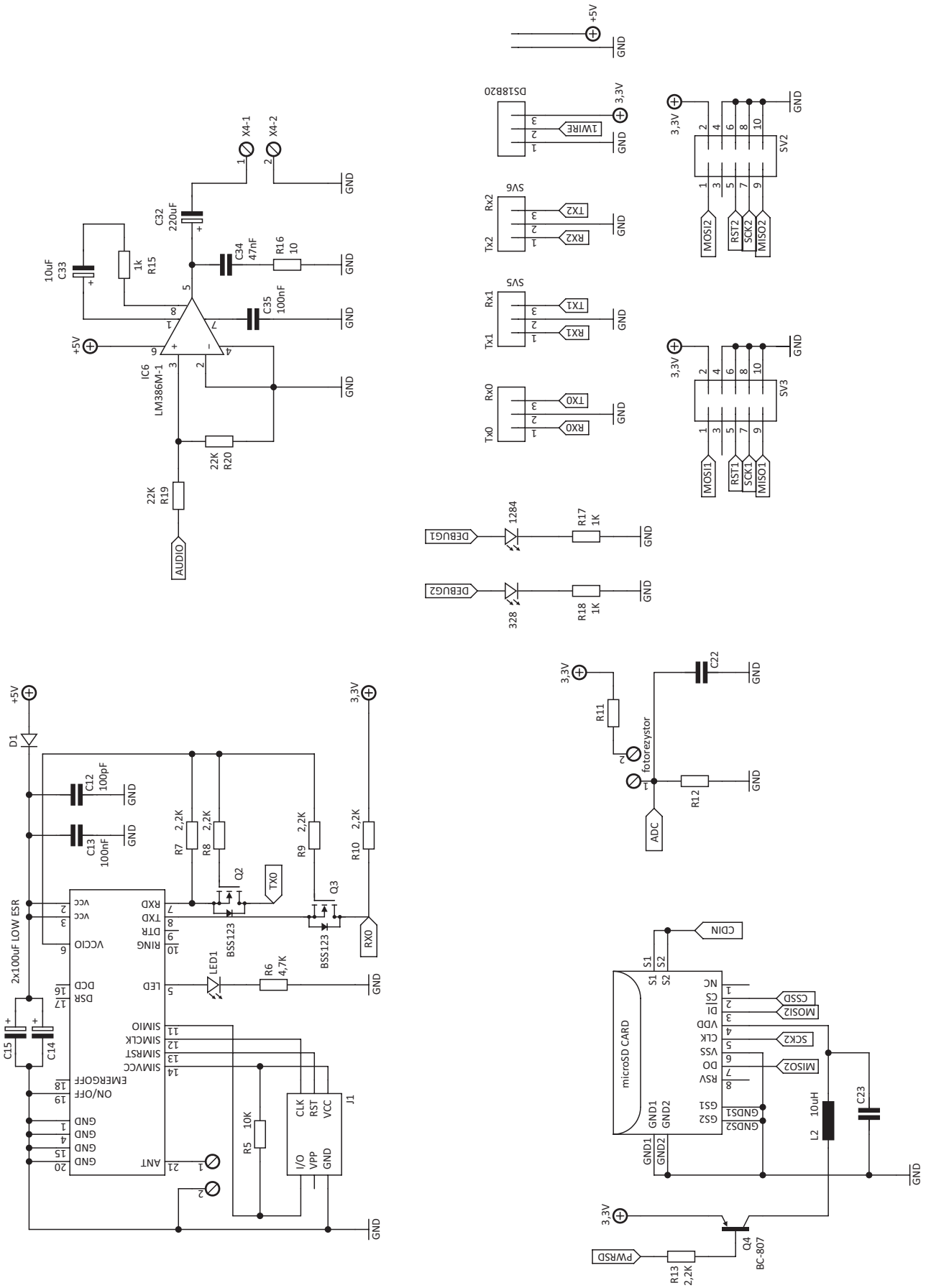


Rysunek 4. Schemat ideowy płytki interfejsu użytkownika

zastosować ATmega16 lub 32 (te same wyprowadzenia). **Uwaga!** Inny procesor to inne rejestry i wektory przerwań więc trzeba to uwzględnić w oprogramowaniu.

Program napisano w języku C dla częstotliwości taktowania 16 MHz. Kod wynikowy zajmuje 8238 bajtów pamięci Flash i 415 bitów pamięci RAM.

Główna funkcja programu i pętla nieskończona to zaledwie kilka linijek kodu – pokazano je na **listingu 6**. Widzimy inicjalizację timerów sprzętowych, przerwań, interfejsów

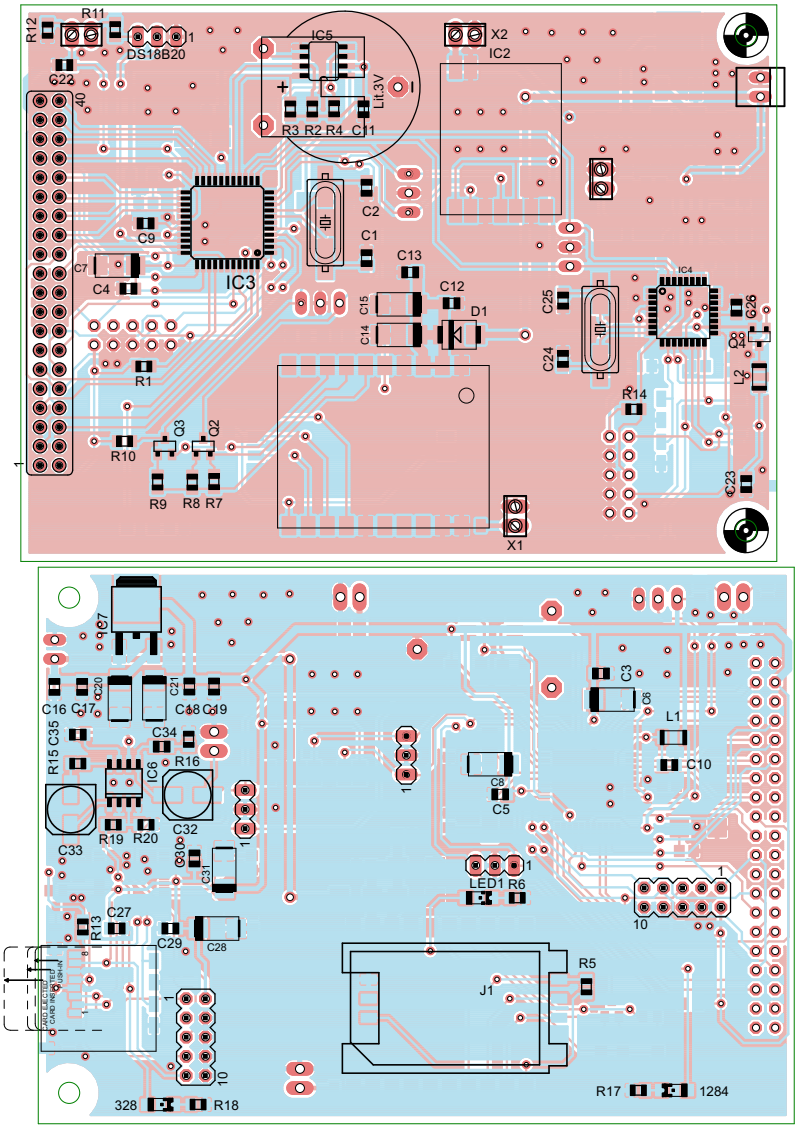


Rysunek 4. cd.

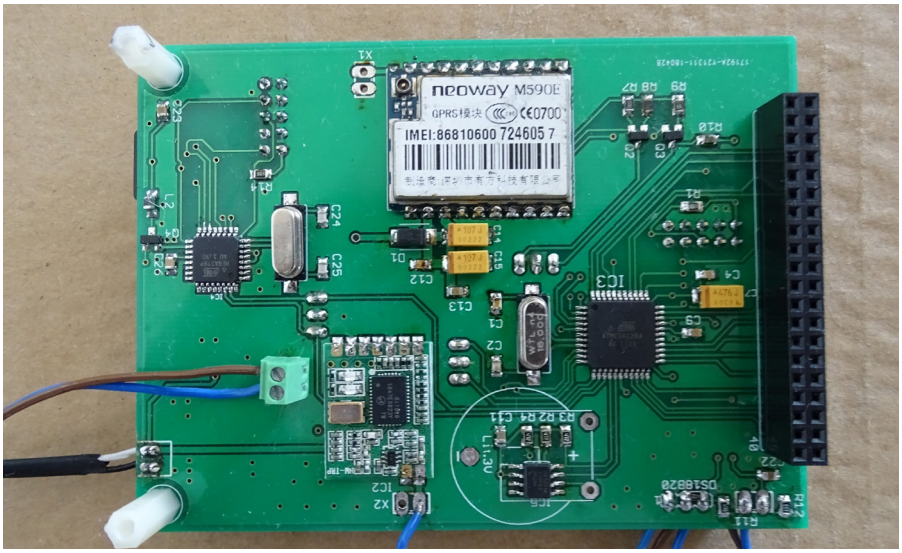
I²C i UART, inicjalizacja timerów programowych oraz rejestrację funkcji callback dla UART. Pętla główna oparta na zdarzeniach wykonuje się z maksymalną prędkością. Reszta kodu w postaci funkcji w oddzielnych plikach programu.

Moduł interfejsu użytkownika (procesora i wyświetlacza)

Na **rysunku 4** pokazano schemat ideowy modułu procesora głównego i wyświetlacza. Jako wyświetlacz wybrałem gotowy moduł ze sterownikiem SSD1963 o wielkości 4,3".



Rysunek 5. Schemat montażowy płytki interfejsu użytkownika



Fotografia 6. Płytka interfejsu użytkownika – widok od góry

Moduł ma gniazdo karty pamięci oraz rezystancyjny panel dotykowy. Do kupienia na znanym portalu aukcyjnym.

Uwaga! Może się zdarzyć, że zakupiony wyświetlacz ma inną orientację kolorów niż RGB. W projekcie użyto wyświetlacza, który pracuje w trybie BRG i tak zostały przygotowane funkcje w inicjalizacji wyświetlacza i definicje kolorów. Przy innej orientacji należy dokonać zmian w plikach *tft.c* i *tft_graf.h*.

Płytkę procesora głównego została tak zaprojektowano, aby można było dołączyć do niej moduł z wyświetlaczem na „kanapkę". Projekt pozwala nam na wybranie wersji wykonania w zależności od potrzeb. Na płytce przewidziano miejsce dla modułu GSM (odpytywanie urządzenia o parametry za pomocą SMS-a) oraz wydzielony blok dodatkowego procesora z gniazdem karty pamięci i wzmacniaczem audio pełniących rolę odtwarzacza komunikatów głosowych (o pełnej godzinie informacja o czasie, temperaturze i ciśnieniu). Linie SPI do karty pamięci i układu sterującego dotykiem z modułu wyświetlacza zostały również wyprowadzone do złącza i odpowiednich nóżek procesora. Takie rozwiązanie pozwala na pełną uniwersalność. Możemy zrezygnować z komunikatów głosowych lub modułu GSM a w końcu wykorzystać płytkę do zupełnie innych rozwiązań w których korzystamy z tego typu wyświetlacza.

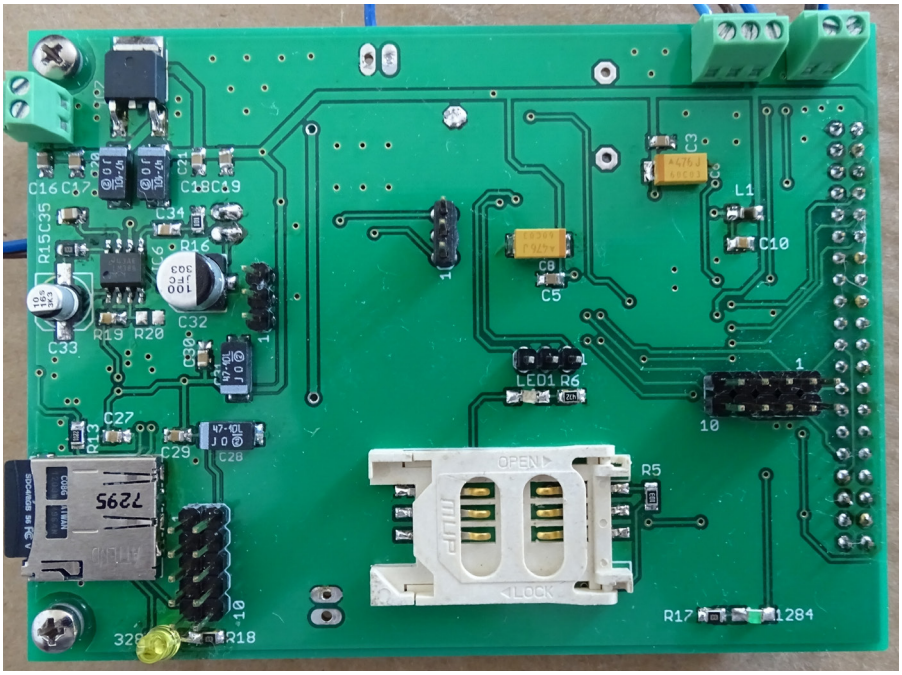
Schemat montażowy modułu interfejsu użytkownika zamieszczono na **rysunku 5**, a jej wygląd po zmontowaniu na **fotografiach 6 i 7**. Na płytce są widoczne moduł radiowy HM-TRP oraz układ RTC typu DS3231 bez wlutowanej baterii podtrzymującej. Wybrano go ze względu na bardzo dobre parametry. Na przykład, ma on wbudowany oscylator skompensowany temperaturowo co powoduje, że dryft oscylatora w zakresie temperatury 0…40°C to jedynie 2 ppm. Zewnętrzny oscylator to ok. 100 ppm, więc dokładność zegara jest rewelacyjna (po szczegóły odsyłam do noty katalogowej układu).

Jako główny procesor pracuje mikrokontroler ATmega1284. Wybrany go ze względu

Listing 7. Funkcja odczytująca dane

```
void DANE_answer(char *bufor)
{
    char *wsk;

    if(!strncasecmp(uart1_buf,"AT+temp",7))
    {
        wsk = strtok(uart1_buf," ");
        wsk = strtok(NULL,",");
        subzero1 = atoi(wsk);
        wsk = strtok(NULL,",");
        cel1 = atoi(wsk);
        wsk= strtok(NULL,",");
        cel_fract_bits1 = atoi(wsk);
        Show_temp_out();
    }
    if(!strncasecmp(uart1_buf,"AT+wind",7))
    {
        wsk = strtok(uart1_buf," ");
        wsk = strtok(NULL,",");
        windspeed_dz = atoi(wsk);
        wsk = strtok(NULL,",");
        windspeed_u = atoi(wsk);
        wsk = strtok(NULL,",");
        wind_dir = atoi(wsk);
        Show_wind();
    }
    if(!strncasecmp(uart1_buf,"AT+rain",7))
    {
        wsk = strtok(uart1_buf," ");
        wsk = strtok(NULL,",");
        rain_dz = atoi(wsk);
        wsk = strtok(NULL,",");
        rain_u = atoi(wsk);
        Show_rain();
    }
    if(!strncasecmp(uart1_buf,"AT+data",7))
    {
        wsk = strtok(uart1_buf," ");
        wsk = strtok(NULL,",");
        press = atoi(wsk);
        wsk = strtok(NULL,",");
        humi = atoi(wsk);
        Show_press();
        Show_humi();
    }
    if(!strncasecmp(uart1_buf,"AT+storm",8))
    {
        wsk = strtok(uart1_buf," ");
        wsk = strtok(NULL,"");
        storm = atoi(wsk);
        Show_storm();
    }
}
```



Fotografia 7. Płytka interfejsu użytkownika – widok od spodu

Listing 8. Funkcja wyświetlająca ciśnienie

```
void Show_press(void)
{
    memset(press_string,0,sizeof(press_string));
    char tab01[6];

    itoa(press,tab01,10);
    strcat(press_string,tab01);
    strcat(press_string," hPa");
    tft_fill_rect(20,230,155,31,black);
    setCurrentFont(&ArialBlack22pt_boldFontInfo);
    if(press<1000) tft_puts(40, 235, press_string, gold, black);
    else tft_puts(30, 235, press_string, gold, black);
}
```

Listing 9. Przerwanie od zegara RTC

```
ISR(INT2_vect)
{
    static uint8_t licznik;

    if(0 == licznik) DS18X20_start_meas( DS18X20_POWER_EXTERN, NULL ); // pomiar
    if(1 == licznik) (  DS18X20_read_meas(gSensorIDs[0], &subzero, &cel, &cel_fract_bits) );
    if(2 == licznik) Show_temp_in();
    licznik++;
    if(licznik>2)licznik=0;
    FLAG = 1;
    pomiar_ADC(PA7);
}
```

Listing 10. Funkcja przekazujące dane do odtwarzacza dźwiękowego

```
void data_to_audio(TDATETIME * dt )
{
    if((dt->hh<23) && (dt->hh>5))
    {
        char buf[20];
        buf[0]=0;
            if ((dt->ss==0) && (dt->mm==0))
        {
            if(subzero)sprintf(buf,("AT+%d,M%d,%d,\r\n"),dt->hh,cel1,press);
            else sprintf(buf,("AT+%d,%d,%d,\r\n"),dt->hh,cel1,press);
            suart_puts(buf);
        }
    }
}
```

na wielkość pamięci Flash i RAM. Znaczną część pamięci Flash zajmują wzorce grafik pokazywanych na wyświetlaczu. Komunikacja ze sterownikiem wyświetlacza odbywa się w trybie 8-bitowym za pomocą PORT-C. Komunikacja I²C z układem RTC odbywa się za pomocą I²C programowego. Interfejsy UART0 i UART1 są używane do komunikacji z modułem radiowym (0) i modułem GSM (1). W programie zaimplementowano dodatkowo programową obsługę UART-a do komunikacji z mikrokontrolerem odtwarzającym komunikaty dźwiękowe.

Sterowanie jasnością podświetlenia wyświetlacza odbywa się za pomocą PWM. Czujnikiem oświetlenia jest fotorezystor z odpowiednio dobranym dzielnikiem rezystancyjnym. Za pomocą przetwornika A/C jest ustalana wymagane wypełnienie przebiegu PWM. Całość, poza wzmacniaczem audio, jest zasilana napięciem 3,3 V, co pozwala na uniknięcie konieczności konwersji poziomu napięcia na liniach sterowania i danych wyświetlacza, układu RTC i modułu radiowego. Tutaj mała uwaga – procesor jest taktowany zewnętrznym oscylatorem 16 MHz. Zgodnie z kartą katalogową, przy taktowaniu tą częstotliwością powinniśmy zasilać procesor napięciem 5 V, ale w zastosowaniach amatorskich możemy sobie pozwolić na „eksperymenty". Prototyp pracuje stabilnie i bezbłędnie od ponad pół roku pomimo zasilania go napięciem 3,3 V przy f_{CLK}=16 MHz.

Główny mikrokontroler poprzez moduł radiowy parsuje dane docierające z czujników w czasie rzeczywistym i wyświetla otrzymane informacje. Źródło funkcji odczytującej dane pokazano na **listingu 7**. W funkcji są cyklicznie sprawdzane kolejne warunki identyfikujące odebrane dane. Otrzymany string jest dzielony na poszczególne tokeny, a otrzymane wartości są przypisywane do odpowiednich zmiennych i wyświetlane za pomocą odpowiednich funkcji. Na przykład, kod funkcji wyświetlającej ciśnienie pokazano na **listingu 8**.

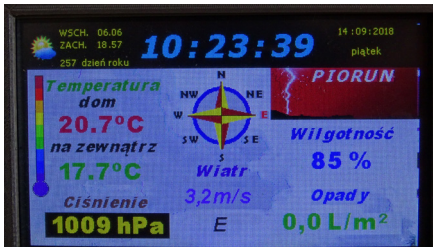
Odczyt temperatury w pomieszczeniu odbywa się w przerwaniu od RTC. Dodatkowo, w przerwaniu dokonujemy pomiaru A/C oraz ustawiamy flagę, za pomocą której wywołujemy funkcję odczytującą rejestry układu RTC (**listing 9**).

Po odczytaniu godziny i daty wartości prezentowane są na wyświetlaczu przy

czym wyświetlanie daty następuje tylko raz na dobę o północy. Wtedy też jest obliczany kolejny dzień roku oraz jest pobierana godzina wschodu i zachodu słońca. Godziny wschodu i zachodu są przechowywane w pamięci Flash w postaci tablic i na podstawie kolejnego dnia roku pobierane do wyświetlania. W godzinach 6:00 do 22:00 o pełnej godzinie do układu głosowego przekazywane są informacje o aktualnej temperaturze i ciśnieniu. Informacja głosowa ma postać „godzina, temp. na zewnątrz, ciśnienie". Tę funkcję obsługuje dodatkowy procesor ze względu na blokujący charakter komunikatów głosowych. Komunikat trwa ok. 12 sekund, a pliki do odtwarzania są pobierane z karty pamięci. Obsługą komunikatów głosowych zajmuje się mikrokontroler ATmega328. Funkcję przekazującą dane do odtwarzacza audio pokazano na **listingu 10**. Procesor odtwarzacza odczytuje te dane i generuje dźwięk, co pokazano na **listingu 11**.

Przy uruchamianiu programu z karty pamięci są ładowane symbol słońca, pioruna oraz tło chmur. Obrazki będące symbolami termometru i róży wiatrów zostały wykonane „na piechotę" ze względu na to, że nie znalazłem zadowalających mnie grafik. Funkcja wczytywania grafiki pokazano na **listingu 12**.

Wygląd obrazu pokazywanego na wyświetlaczu pokazano na **fotografii 8**. Myślę, że nie wymaga on opisywania. Jedyne, na co chcę zwrócić uwagę, to róża wiatrów, na której można zaobserwować zmieniający się kolor liter w zależności od kierunku wiatru. Dodatkowo, zmienia się kolor fontów wyświetlanej temperatury na zewnątrz w zależności od wartości. Poniżej –20°C kolor fioletowy, –20…0°C kolor niebieski, 0…20°C kolor zielony i powyżej 20°C kolor czerwony. Wartość opadów jest zerowana o północy, natomiast odległość od wyładowań atmosferycznych jest zerowana (czyszczona) po upływie 5 minut od ostatniego wykrytego wyładowania.



Fotografia 8. Obraz pokazywany na wyświetlaczu

Obecny w układzie moduł GSM pozwala na odczyt temperatury, ciśnienia i ilości opadów po wysłaniu SMS o treści „ODCZYT". Komunikat SMS jest wysyłany na numer wpisany na stałe do pamięci Flash. Oczywiście, istnieje też możliwość wysłania SMS na numer, z którego odebrano zapytanie. W tym celu, po odebraniu SMS należy sparsować łańcuch znaków i wydzielić z niego token z numerem telefonu, a następnie przypisać ten numer do wiadomości wychodzącej. Szczegółowo zostało to opisane w książce Mirosława Kardasia „Wkuwamy C. Majsterkuj razem ze mną".

Program napisany w języku C zajmuje 41738 bajtów pamięci Flash i 1365 bajtów pamięci RAM. Przy programowaniu wykorzystano biblioteki do obsługi UATR, wyświetlacza TFT , I²C oraz obsługi karty SD i modułu GSM z książek: „Mikrokontrolery AVR język C – podstawy programowania", „Język C – pasja programowania mikrokontrolerów 8-bitowych", „Wkuwamy C. Majsterkuj razem ze mną" autorstwa Mirosława Kardasia. Grafiki i czcionki wykonano za pomocą programu PixelFactory. Pliki dźwiękowe do audio należy przygotować w formacie WAV.

Marek Rębecki
marekrebecki@wp.pl

Listing 11. Obsługa odtwarzacza dźwiękowego

```
void DANE_answer( char *bufor)
{
    char *hour;
    char *temp;
    char *press;
    char Hour[8]={„G"};
    char Temp[9]={„T"};
    char Press[10]={„C"};
    char WAV []={„.wav"};

          if (!strncasecmp(bufor,"AT+",3) )
    {
        hour =strtok(bufor,"+");
        hour =strtok(NULL,",");
        temp =strtok(NULL,",");
        press=strtok(NULL,",");
        strcat(Hour,hour);
        strcat(Hour,WAV);
        strcat(Temp,temp);
        strcat(Temp,WAV);
        strcat(Press,press);
        strcat(Press,WAV);
        pf_mount(&Fs);
        while(1)
        {
            if (pf_opendir(&Dir, „")) break;
            play(Hour);
            play(Temp);
            play(Press);
            break;
        }

    }
}

// funkcja PLAY
static UINT play ( const char *fn )
{
    DWORD sz;
    FRESULT res;

    if ((res = pf_open(fn)) == FR_OK)
    {
        sz = load_header();  // Load file header
        if (sz < 256) return (UINT)sz;
        pf_lseek(0);
        pf_read(&buf[0][0], BUF_SIZE , &rb);  // załaduj pierwszą część bufora
        pf_read(&buf[1][0], BUF_SIZE , &rb);  // załaduj drugą część bufora
        if( !FLAGS.prescaler ) TMR_START;  // start Timera0 (samplowanie)
        DDRD  |= (1<<PD5);  // ustaw pin PWM1 (OC1A) jako wyjście WAŻNE !!!
        while(1)
        {
            if( can_read )
            {  // jeśli flaga ustawiona w obsłudze przerwania
                pf_read(&buf[ nr_buf ^ 0x01 ][0], BUF_SIZE , &rb);  // odczytaj kolejny
bufor
                if( rb < BUF_SIZE ) break;  // jeśli koniec pliku przerwij pętlę while(1)
                can_read = 0;
            }
        }
        DDRD  &= ~(1<<PD5);  // ustaw pin PWM1 (OC1A) jako wejście WAŻNE !!!
        if( !FLAGS.prescaler ) TMR_STOP;  // wyłączenie Timera0 (samplowania)
    }
    return res;
}
```

Listing 12. Funkcja wczytująca i wyświetlająca obrazek

```
void start_graf(void)
{
    ires = mk_petit_init( bufor, sizeof(bufor), 0 );
    while( mk_petit_init( bufor, sizeof(bufor), 0 ));
    pf_mount(&fs);
    tft_bitmap_from_file(0,51,"storm11.rg8",0);
    tft_bitmap_from_file(300,51,"piorun.rg8",0);
    tft_fill_rect(0,0,480,52,black);
    tft_bitmap(5,10,empty,32,32);
    setCurrentFont(&Tahoma8ptFontInfo);
    tft_puts_P(42,3,tab02,yellow,black);
    tft_puts_P(43,18,tab03,yellow,black);
    termometr();
    roza_wiatrow();
    setCurrentFont(&ArialBlack14pt_bolditalicFontInfo);
    tft_puts(25,65,"Temperatura",green1,silver);
    tft_puts(65,85,"dom",black,silver);
    tft_puts(28,140,"na zewnątrz",black,silver);
    tft_puts(45,209,"Ciśnienie",maroon1,silver);
    tft_puts(325,125,"Wilgotność",navy,silver1);
    tft_puts(350,200,"Opady",rainbow2,silver1);
    tft_puts(210,170,"Wiatr",rainbow1,silver);
    setCurrentFont(&Tahoma14pt_bolditalicFontInfo);
    tft_puts(350,55,"PIORUN",white,rainbow7);
}
```