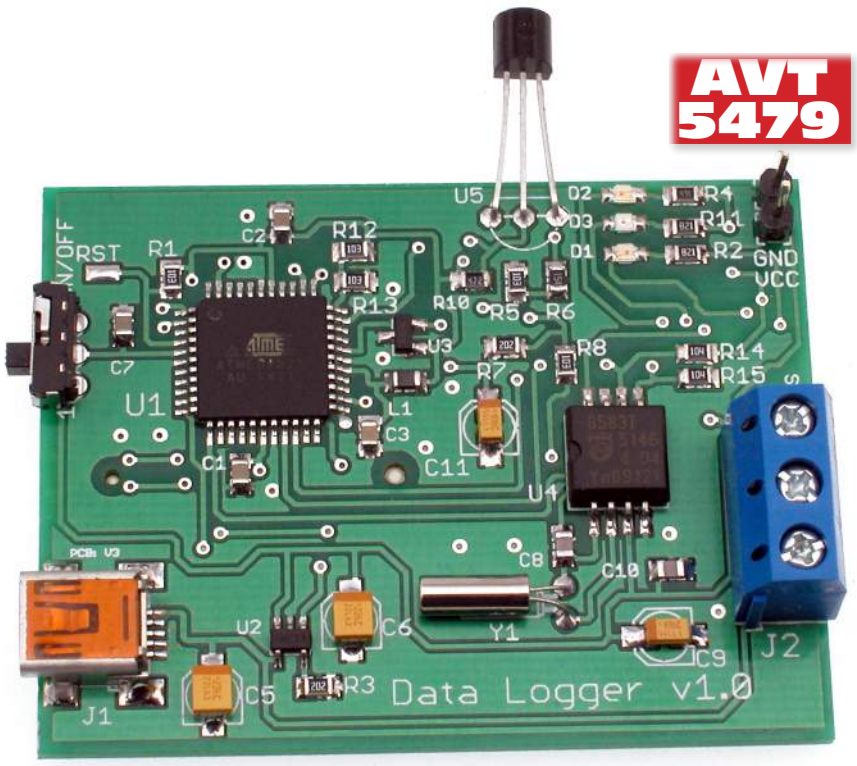
Data Logger – rejestrator temperatury lub napięcia

Data Logger jest uniwersalnym bankiem danych zasilanym z baterii. Pozwala na zapisywanie na karcie SD informacji z czujnika temperatury oraz z dwóch napięciowych wejść analogowych. Ma wbudowany układ ładowania baterii za pomocą USB. Dodatkowo, wyposażono go w zegar czasu rzeczywistego. Pełna konfiguracja urządzenia odbywa się za pomocą zmian w pliku konfiguracyjnym zapisanym na karcie SD. ***Rekomendacje:*** *urządzenie przyda się do monitorowania temperatury lub innych parametrów środowiskowych zamienionych na napięcie (albo samego napięcia w zakresie 0…15 V lub 0…40 V).*



Schemat ideowy rejestratora pokazano na **rysunku 1**. Urządzenie jest zasilane za pomocą baterii litowo-jonowej o pojemności 1800 mAh z telefonu komórkowego. Napięcie zasilania płytki rejestratora mieści się w zakresie 3,1…4,2 V. Dla uproszczenia budowy rejestratora i ze względu na charakter zasilania dostarczanego przez akumulator, nie zastosowano stabilizatora scalonego. Po dołączeniu baterii na stałe jest zasilany tylko układ zegara RTC – U6 (PCF8583T), dzięki czemu nie trzeba po każdym włączeniu urządzenia ustawiać daty i godziny. Pobór prądu przez ten układ jest znikomy i wynosi jedynie 50 µA. Reszta urządzenia jest zasilana za pośrednictwem wyłącznika.

Sercem rejestratora jest mikrokontroler ATmega32A taktowany za pomocą wewnętrznego oscylatora RC o częstotliwości 1 MHz. Do sygnalizowania jego statusu służą dwie diody LED przyłączone do mikrokontrolera. Trzecią diodę dołączono do wyjścia PROG układu scalonej ładowarki akumulatorów Li-Ion – MCP73831. Służy ona do sygnalizowania stanu ładowania baterii (dioda świeci się podczas ładowania baterii, a po jej naładowaniu gaśnie). Ładowarka jest zasilana za pośrednictwem gniazda mini USB. Rolę czujnika temperatury pełni DS18B20 firmy Maxim-Dallas dołączany do mikrokontrolera poprzez interfejs 1-Wire.

Algorytm działania programu pokazano na **rysunku 2**.

Do wejścia przetwornika A/C doprowadzono napięcie zasilające. Gdy jego wartość osiągnie wartość 3,1 V, rejestrator wyłącza się (mikrokontroler jest wprowadzany w tryb *Power Down*). Ten mechanizm ma na celu ochronę akumulatora zasilającego przed zbyt głębokim rozładowaniem. Jeśli dojdzie do wyłączenia zasilania, to należy po prostu naładować akumulator oraz wyłączyć i włączyć zasilanie za pomocą wyłącznika, co spowoduje restart mikrokontrolera.

Na pinie zasilającym część analogową mikrokontrolera (AVCC) zastosowano filtr LC. Dla poprawienia dokładności pomiarów zastosowano zewnętrze źródło napięcia odniesienia – jego rolę pełni układ U5. Jego napięcie wyjściowe wynosi 2,5 V z dokładnością 1%. Pomiar napięcia baterii zasilającej odbywa się poprzez dzielnik napięcia z dwóch rezystorów 10 kΩ/1%, dzięki czemu w skrajnym wypadku (4,2 V) napięcie na wejściu przetwornika nie przekracza 2,5 V, tj. napięcia maksymalnego. Pozostałe dwa kanały, do których można dołączyć napięcia

W ofercie AVT*

AVT-5479 A AVT-5479 B
AVT-5479 UK

Podstawowe informacje:

- Rejestrowanie temperatury, napięcia zasilającego i dwóch napięć wejściowych w pliku CSV na karcie SD.
- Napięcie wejściowe z zakresu 0…15 V lub 0…40 V.
- Napięcie zasilające 3,1…4,2 V (akumulator Li-Ion), maksymalny pobór prądu ok. 20 mA.
- Nieprzerwana praca przez ok. 3 doby przy zasilaniu z akumulatora o pojemności 1200 mAh.
- Mikrokontroler ATmega32, oprogramowanie w języku C.

Dodatkowe materiały na FTP:

ftp://ep.com.pl, user: 63172, pass: 428ofq53

- wzory płytek PCB

Projekty pokrewne na FTP:

(wymienione artykuły są w całości dostępne na FTP)

AVT-5458	Wielokanałowy woltomierz-rejestrator (EP 9/2014)
AVT-5420	Wielopunktowy termometr z rejestracją (EP 10/2013)
AVT-5373	Tlogger – rejestrator temperatury (EP 12/2012)

* Uwaga:
Zestawy AVT mogą występować w następujących wersjach:

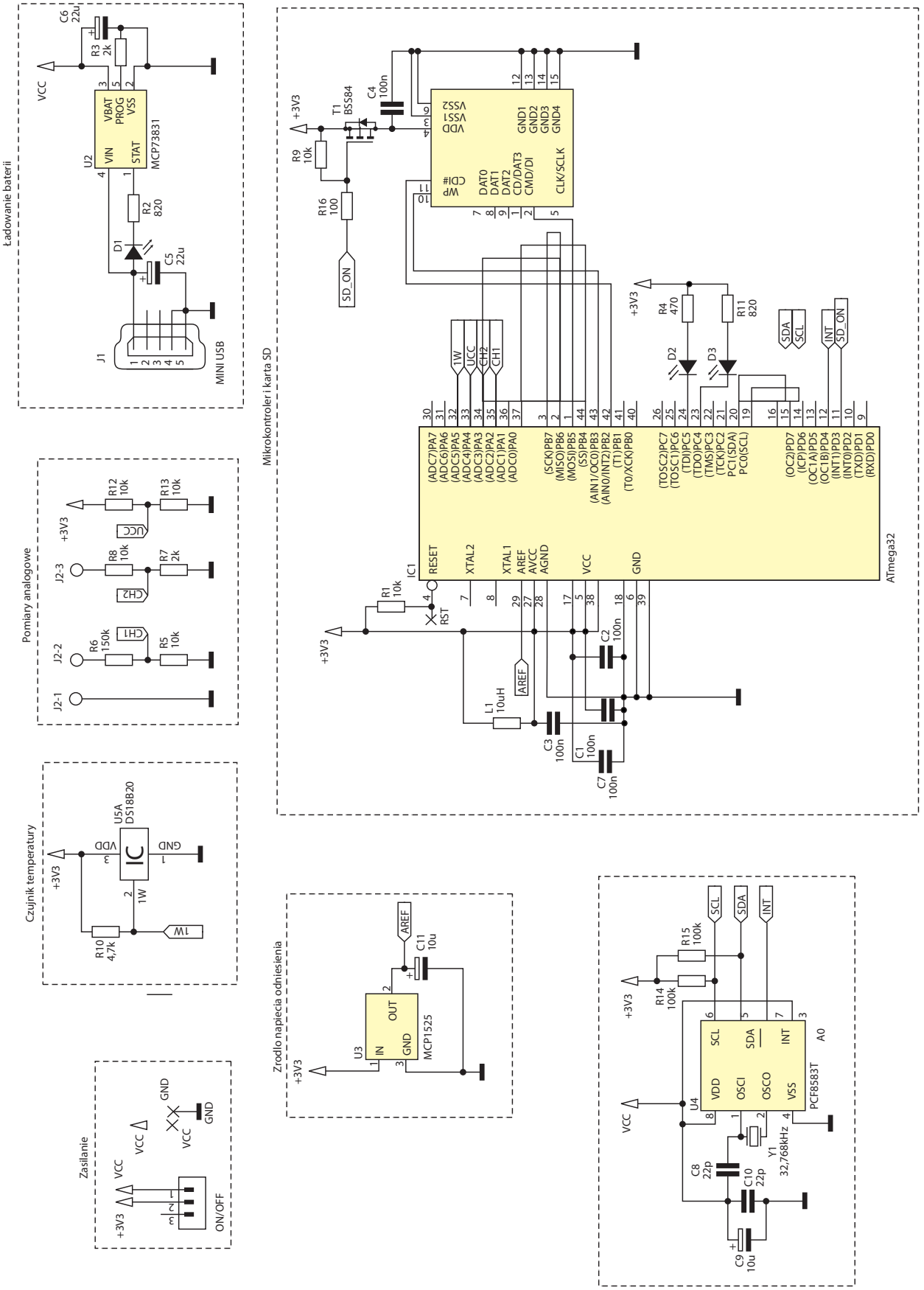
AVT xxxx UK	to zaprogramowany układ. Tylko i wyłącznie. Bez elementów dodatkowych.
AVT xxxx A	płytka drukowana PCB (lub płytki drukowane, jeśli w opisie wyraźnie zaznaczono), bez elementów dodatkowych.
AVT xxxx A+	płytka drukowana i zaprogramowany układ (czyli połączenie wersji A i wersji UK) bez elementów dodatkowych.
AVT xxxx B	płytka drukowana (lub płytki) oraz komplet elementów wymienionych w załączniku pdf
AVT xxxx C	to nic innego jak zmontowany zestaw B, czyli elementy wlutowane w PCB. Należy mieć na uwadze, że o ile nie zaznaczono wyraźnie w opisie, zestaw ten nie ma obudowy ani elementów dodatkowych, które nie zostały wymienione w załączniku pdf
AVT xxxx CD	oprogramowanie (niezęsto spotykana wersja, lecz jeśli występuje, to niezbędne oprogramowanie można ściągnąć, klikając w link umieszczony w opisie kitu)

Nie każdy zestaw AVT występuje we wszystkich wersjach! Każda wersja ma załączony ten sam plik pdf! Podczas składania zamówienia upewnij się, którą wersję zamawiasz! (UK, A, A+, B lub C). http://sklep.avt.pl

z zakresu 0…40 V i 0…15 V również dołączone są przez dzielniki, ale o innym stopniu podziału (150 kΩ/10 kΩ oraz 10 kΩ/2 kΩ). Dla kanału pierwszego współczynnik podziału wynosi 15, natomiast dla drugiego 5. Są to liczby całkowite, dzięki czemu łatwo je zaimplementować przy przeliczaniu wartości z przetwornika na rzeczywiste napięcie (**listing 1**).

W pętli głównej mikrokontroler odczytuje reprezentację napięcia z przetwornika, natomiast jej zamiana na napięcie odbywa się przed zapisem na karcie SD.

Do rejestrowania danych zastosowano kartę SD. Do obsługi programowej wykorzystano bibliotekę FATFs dostępną na stronie internetowej *http://elm-chan.org/*. Wszystkie informacje są zapisywane w pliku DATA.CSV. Dzięki zachowaniu formatu CSV plik można otworzyć np. za



Rysunek 1. Schemat ideowy Data Loggera

pomocą arkusza kalkulacyjnego i w prosty sposób dokonać ich interpretacji.

W związku z tym, że w urządzeniu nie zostały zastosowane żadne elementy interfejsu użytkownika umożliwiające jego skonfigurowanie, odbywa się ono za pomocą pliku zapisywanego na karcie SD. W ten sposób można ustawić: datę, godzinę, częstotliwość wykonywania pomiarów, wykorzystywane kanały pomiarowe. Aby dokonać konfiguracji, należy na karcie SD utworzyć plik *config.txt*. Jego odczyt jest wykonywany tylko raz, przy włączeniu urządzenia. Na **listingu 2** przedstawiono procedurę odczytu pliku konfiguracyjnego. Rozpoczyna się ona od sprawdzenia, czy włożono kartę SD do gniazda. Jeśli tak (warunek CD_ON jest spełniony), to rozpoczyna się inicjalizacja karty. Następnie polecenie *f_open* otwiera plik *config.txt*, jeśli ten istnieje. Gdy wszystkie warunki zostaną spełnione, parametr *f_err_code* zwraca wartość *FR_OK*. Jeżeli ten warunek nie zostanie spełniony, czyli użytkownik nie wgra pliku konfiguracyjnego na kartę, program pobierze dane konfiguracyjne z pamięci EEPROM. Jeśli jednak taki plik istnieje, zaczyna się odczyt danych. Polecenie *f_lseek* ustawia kursor odczytu na samym początku pliku. Polecenie *f_gets* do zmiennej znakowej *rok* odczytuje całą linię, aż do znaków CR+LF, czyli dwa bajty danych (np: 14 oznaczający rok 2014) oraz bajt końca linii. Następnie polecenie *f_lseek* przesuwa kursor o dwa bajty, na kolejną linię, w której podano miesiąc. W ten sam sposób odczytywane są parametry: dzień, godzina, minuta, częstotliwość zapisu (s1) oraz wybrane kanały (s2). Następnie plik jest zamykany.

W związku z faktem, że wykorzystane dane mają być m.in. przesłane do układu RTC, muszą zostać przekonwertowane na liczbę. Do tego celu wykorzystano funkcję *atoi()* zawartą w bibliotece *stdlib.h*. Wszystkie dane są konwerto-

	A	B	C	D	E
1	Date:	29.05.2014			
2	Time	Vcc [V]	U1 [V]	U2 [V]	T1 [°C]

Rysunek 2. Nagłówki dodawane w pliku CSV

Listing 1. Konwersja wartości odczytanej z przetwornika na napięcie

```
while(1)
{
        .
        .
        .
    //pomiar napięcia CH1
    u1_adc = pomiar(2);
    //pomiar napięcia CH2
    u2_adc = pomiar(3);
        .
        .
        .
}
//obliczenie napięcia zasilania
u = u_adc * 244 * 2;
cz_d_u = u / 100000;
cz_u_u = (u/1000) % 100;
//obliczenie napięcia CH1
u1 = u1_adc * 244 * 15;
cz_d_u1 = u1 / 100000;
cz_u_u1 = (u1/1000) % 100;
//obliczenie napięcia CH2
u2 = u2_adc * 244 * 6;
cz_d_u2 = u2 / 100000;
cz_u_u2 = (u2/1000) % 100;
```

Listing 2. Procedura odczytu pliku konfiguracyjnego

```
if (CD_ON)  // sprawdzenie czy włożono kartę
{
    disk_initialize(0);
    _delay_ms(10);
    f_mount(0, &FATFS_Obj);
//inicjalizacja karty oraz otwarcie pliku w katalogu głównym
//o nazwie config.txt
    f_err_code = f_open(&fil_obj, „config.txt”, FA_READ);
    _delay_ms(10);
//sprawdzanie czy plik istnieje
    if(f_err_code == FR_OK)
    {
//ustawienie kursora na początku pliku
        f_lseek(&fil_obj, 0);
//odczyt do zmiennej znakowej całej lini pilku (aż do znaku CR LF)
        f_gets(rok, sizeof rok, &fil_obj);
//przesunięcie kursora o dwa bajty
        f_lseek(&fil_obj, f_tell(&fil_obj) + 2);
        f_gets(miesiac, sizeof miesiac, &fil_obj);
        f_lseek(&fil_obj, f_tell(&fil_obj) + 2);
        f_gets(dzien, sizeof dzien, &fil_obj);
        f_lseek(&fil_obj, f_tell(&fil_obj) + 2);
        f_gets(godzina, sizeof godzina, &fil_obj);
        f_lseek(&fil_obj, f_tell(&fil_obj) + 2);
        f_gets(minuta, sizeof minuta, &fil_obj);
        f_lseek(&fil_obj, f_tell(&fil_obj) + 2);
        f_gets(s1, sizeof s1, &fil_obj);
        f_lseek(&fil_obj, f_tell(&fil_obj) + 2);
        f_gets(s2, sizeof s2, &fil_obj);
        _delay_ms(10);
        f_close(&fil_obj);  //zamknięcie pliku
        set_godziny = atoi(godzina);  //konwersja zmiennych z string na int
        set_minuty = atoi(minuta);
        ram_dane.set_rok = atoi(rok);
        set_miesiac = atoi(miesiac);
        set_dzien = atoi(dzien);
        ram_dane.s1_d = atoi(s1);
        ram_dane.s2_d = atoi(s2);
//warunki zabezpieczające przed sczytaniem błędnych
//danych wprowadzonych przez użytkownika
        if(ram_dane.s1_d >= 10) ram_dane.s1_d = 9;
        if(ram_dane.s2_d >= 7) ram_dane.s2_d = 6;
        if (set_godziny >= 24) set_godziny = 0;
        if (set_minuty >= 60) set_minuty = 0;
        if (ram_dane.set_rok >=100) ram_dane.set_rok = 0;
        if (set_miesiac >= 13) set_miesiac = 1;
        if (set_dzien >=32) set_dzien = 1;
//po ustawieniu danych skopiowanie ich do eepromu
        copy_ram_to_eem();
//ustawienie czasu
        set_time(set_godziny, set_minuty, 0);
//ustawienie daty (rok liczony przez program a nie przez pcf8583)
        set_date(set_dzien, set_miesiac);
    }
//jeśli wykryto brak pliku, program sczytuje dane z eeprom
//(rok, częstotliwość zapisu, typ danych do zapisu)
    else copy_eem_to_ram();
        flaga_start = 1;
    }
//jeśli wykryto brak karty SD, program sczytuje dane z eeprom
//(rok, częstotliwość zapisu, typ danych do zapisu)
    else
    {
        copy_eem_to_ram();
        flaga_start = 1;
    }

//instrukcja ustawia wartość częstotliwości zapisu danych na SD
//w zależności od wprowadzonej liczby
    switch(ram_dane.s1_d)
    {
        case 0:
        {
            f_zapisu = 1;  //dla 0 zapis co 1 sekundę
            break;
        }
    .
    .
    .
         case 9:
         {
             f_zapisu = 7200;  //dla 9 zapis co 2 godziny
             break;
         }
    }
```

Listing 3. Struktura zmiennej z konfiguracją

```
typedef struct {  //deklaracja struktury zapisywanej w eepromie oraz w pamieci
ram
    uint8_t set_rok;
    uint8_t s1_d;
    uint8_t s2_d;
} LDATA;
LDATA ram_dane;
LDATA eem_dane __attribute__((section(„.eeprom”)));  // dane w pamięci EEPROM
```

Listing 4. Procedura wprowadzania mikrokontrolera w stan głębokiego uśpienia *Power Down*

```
void uspienie_low_bat(void)
{
    sd_pwr(0);  //wyłączenie zasilania karty SD
//wprowadź mikrokontroler w tryb Power Down
    flaga = 1;
    set_sleep_mode(SLEEP_MODE_PWR_DOWN);  //tryb Power Down Mode
    sleep_enable();
    sleep_cpu();
}

void sd_pwr( uint8_t OnOff ) {  //wyłączenie karty SD
    if(OnOff) {
        SPCR |= (1<<SPE);
        SD_ON;
        _delay_ms(50);
    } else {
        SPCR &= ~ (1<<SPE);
        DDRB &= ~ (1<<PB4);
        DDRB |= (1<<PB5);
        DDRB |= (1<<PB7);
        PORTB &= ~ (1<<PB4);
        PORTB &= ~ (1<<PB5);
        PORTB &= ~ (1<<PB7);
        SD_OFF;
        _delay_ms(50);
    }
}
```

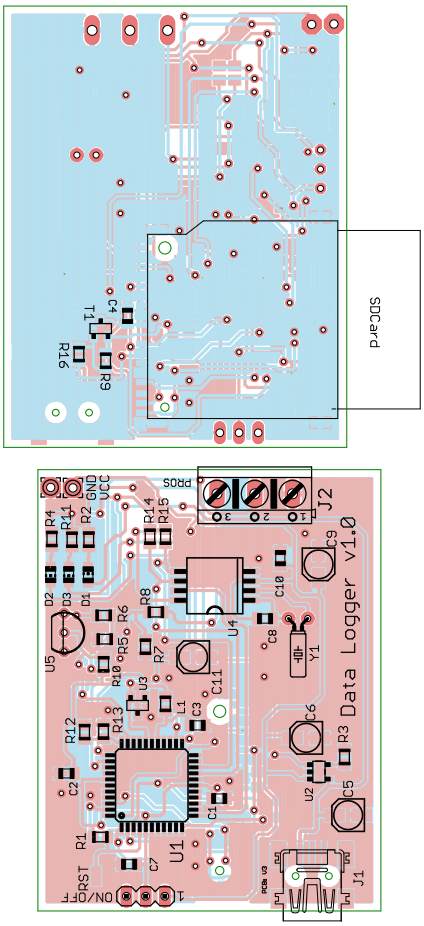
wane na typ *uint8_t*. Informacje, takie jak: rok, częstotliwość zapisu, wybrane kanały są zapisywane w pamięci EEPROM. Aby łatwo zapamiętać te dane w pamięci EEPROM, zdefiniowano strukturę pokazaną na **listingu 3**. Ta sama struktura służy do utworzenia zmiennej w pamięci RAM, dzięki czemu można po prostu skopiować konfigurację z EEPROM do RAM.

Po konwersji zmiennych odbywa się sprawdzenie, czy użytkownik wprowadził poprawne dane. Jeśli nie, program samodzielnie ustawia wartości domyślne. Kolejnym krokiem jest zapisanie danych w pamięci EEPROM oraz przesłanie nastaw do zegara RTC. Następnie w instrukcji *switch* odbywa się ustawienie częstotliwości zapisu danych na kartę SD. W zależności od wprowadzonej wartości zmiennej *ram_dane.s1_d* jest ustawiana wartość zmiennej *f_zapisu*, która to jest porównywana ze zmienną *flaga_licznik*. Ta zmienna jest zwiększana co 1 sekundę przez procedurę obsługi przerwania zewnętrznego INT1. Źródłem przerwania jest wyjście INT układu zegara RTC.

Jak wspomniano, na początku pętli głównej programu jest mierzone napięcie zasilania i jest podejmowana decyzja o wyłączeniu urządzenia lub dalszej pracy. Na **listingu 4** pokazano procedurę wprowadzania mikrokontrolera w stan głębokiego uśpienia *Power Down*. Funkcja *sd_pwr()* wyłącza zasilanie karty SD. Najpierw linie SPI są ustawiane, a następnie jest wyłączane zasilanie poprzez wyzerowanie pinu PD2 sterującego tranzystorem T1 pełniącego rolę włącznika zasilania karty (zależnie od argumentu funkcji można kartę włączyć – argument 1, lub wyłączyć – argument 0). Następnie funkcja *uspienie_low_bat* wprowadza procesor w stan uśpienia.

Kolejnym krokiem w pętli głównej jest porównywanie wartości zmiennych *flaga_licznik* i *f_zapisu*. Jeśli te zmienne są równe, to rozpoczyna się procedura zapisu danych na karcie SD. Po pierwszym uruchomieniu program tworzy lub otwiera plik *DATA.csv*, a następnie dodaje w nim nagłówki, co pokazano na **rysunku 2**. W kolejnych liniach pliku program zapisuje rekordy. To, jakie dane zostaną zapamiętane zależy od wybranego trybu pracy. Użytkownik ma do wyboru 7 trybów zapisu danych. Na **listingu 5** pokazano jeden z nich. Na początku wywołana jest funkcja *card_init()*, która zapisuje aktualną godzinę oraz inicjalizuję kartę. Tu również zostaje włączona na czas zapisu dioda LED2. Następnie, kolejną zapisywane są:

- wartość napięcia zasilającego,
- wartość napięcia na wejściu kanału pierwszego,



Rysunek 3. Schemat montażowy Data Loggera

Wykaz elementów

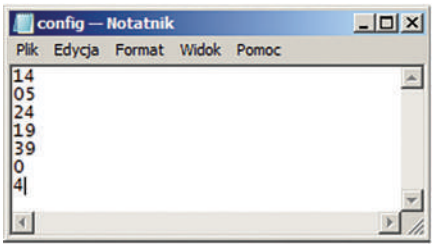
Rezystory: (SMD 0805)
R1, R5, R8, R9, R12, R13: 10 kΩ
R2, R11: 820 Ω
R3, R7: 2 kΩ
R4: 470 Ω
R6: 150 kΩ
R10: 4,7 kΩ
R14, R15: 100 kΩ
R16: 100 Ω

Kondensatory:
C1...C4, C7: 100 nF (SMD 0805)
C5, C6: 22 uF
C8, C10: 22 pF (SMD 0805)
C9, C11: 10uF

Półprzewodniki:
U1: ATmega32A (SMD)
U2: MCP73831
U4: PCF8583T
U5: DS18B20
T1: BSS84
LED1: dioda LED SMD, czerwona
LED2: dioda LED SMD, zielona
LED3: dioda LED SMD, niebieska

Inne:
L1: dławik 10 µH
Y1: kwarc 32,768 kHz
J1: gniazdo MINI-USB, SMD
J2: złącze ARK3/500
Złącze karty SD
Przełącznik HSS1260R, Goldpin 1×2

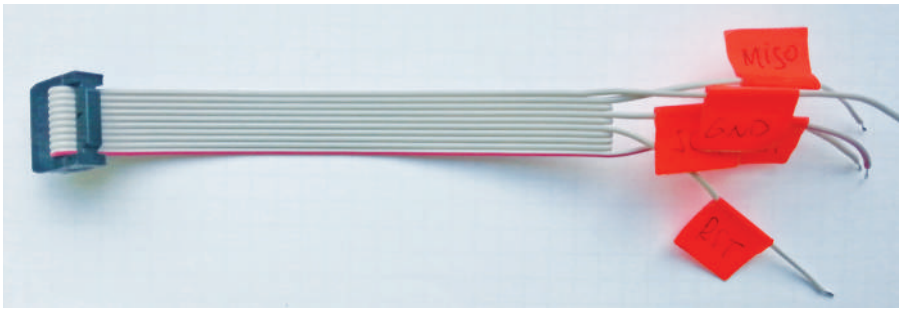
Konfiguracja urządzenia
Na karcie SD należy utworzyć plik tekstowy o nazwie *config.txt* i wprowadzić następujące dane:
Linie 1…3 zawierają datę w kolejności: rok, miesiąc, dzień.
Linie 4…5 zawierają godzinę w kolejności: godzina, minuty.
Linia 6 zawiera częstotliwość zapisu.



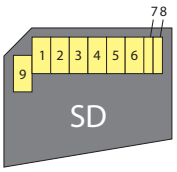
1 s	5 s	10 s	30 s	1 min	5 min	10min	30 min	1 h	2h
0	1	2	3	4	5	6	7	8	9

Np. dla okresu zapisu równego 10 minut, w szóstej linii pliku wpisujemy liczbę „6”.
Linia 7 zawiera wybrane kanały pomiarowe.

0	1	2	3	4	5	6
Wszystkie dane: temperatura, napięcia na CH1 i CH2, napięcie zasilające	Napięcia na CH1 i CH2, temperatura	Napięcie na CH1, temperatura	Napięcie na CH2, temperatura	Napięcia na CH1 i CH2	Temperatura	Temperatura i napięcie zasilające



Fotografia 4. Kabel używany do programowania



Pin	SD	SPI
1	CD/DAT3	X
2	CMD	MOSI
3	VSS1	GND
4	VDD	VDD
5	CLK	SCK
6	VSS2	GND
7	DAT0	MISO
8	DAT1	X
9	DAT2	X

Rysunek 5. Wyprowadzenia interfejsu karty SD

Listing 5. Zapis danych na karcie SD w jednym z trybów pracy

```
LEDW_ON;  //zaświecenie zielonej diody LED
card_init();
    .
    .
f_printf(&fil_obj, „%d”, cz_d_u);
f_printf(&fil_obj, „%s”, „,”);
if (cz_u_u < 10) f_printf(&fil_obj, „%s”, „0”);
if (cz_u_u == 0) f_printf(&fil_obj, „%s”, „00”);
f_printf(&fil_obj, „%d”, cz_u_u);
f_printf(&fil_obj, „%s”, „;”);
f_printf(&fil_obj, „%d”, cz_d_u1);
f_printf(&fil_obj, „%s”, „,”);
if (cz_u_u1 < 10) f_printf(&fil_obj, „%s”, „0”);
if (cz_u_u1 == 0) f_printf(&fil_obj, „%s”, „00”);
f_printf(&fil_obj, „%d”, cz_u_u1);
f_printf(&fil_obj, „%s”, „;”);
f_printf(&fil_obj, „%d”, cz_d_u2);
f_printf(&fil_obj, „%s”, „,”);
if (cz_u_u2 < 10) f_printf(&fil_obj, „%s”, „0”);
if (cz_u_u2 == 0) f_printf(&fil_obj, „%s”, „00”);
f_printf(&fil_obj, „%d”, cz_u_u2);
f_printf(&fil_obj, „%s”, „;”);
if(subzero) f_printf(&fil_obj, „%s”, „-”);
else f_printf(&fil_obj, „%s”, „ „);
f_printf(&fil_obj, „%d”, cel);
f_printf(&fil_obj, „%s”, „,”);
if (cel_fract_bits == 0) f_printf(&fil_obj, „%s”, „0”);
f_printf(&fil_obj, „%d”, cel_fract_bits);
f_printf(&fil_obj, „%s”, „\r\n”);
    .
    .
f_close(&fil_obj);  //zamknięcie pliku
LEDW_OFF;  //zgaszenie zielonej diody LED (świeci w trakcie zapisu)
```

- wartość napięcia na wejściu kanału drugiego
- temperatura otoczenia.

Na końcu jest zapisywany znak nowej linii.

Krótkie, naprzemienne błyskanie diod LED2 i LED3 oznacza brak możliwości zapisu na karcie.

Montaż i uruchomienie

Schemat montażowy rejestratora pokazano na **rysunku 3**. Jego montaż jest typowy i nie wymaga szczegółowego omawiania.

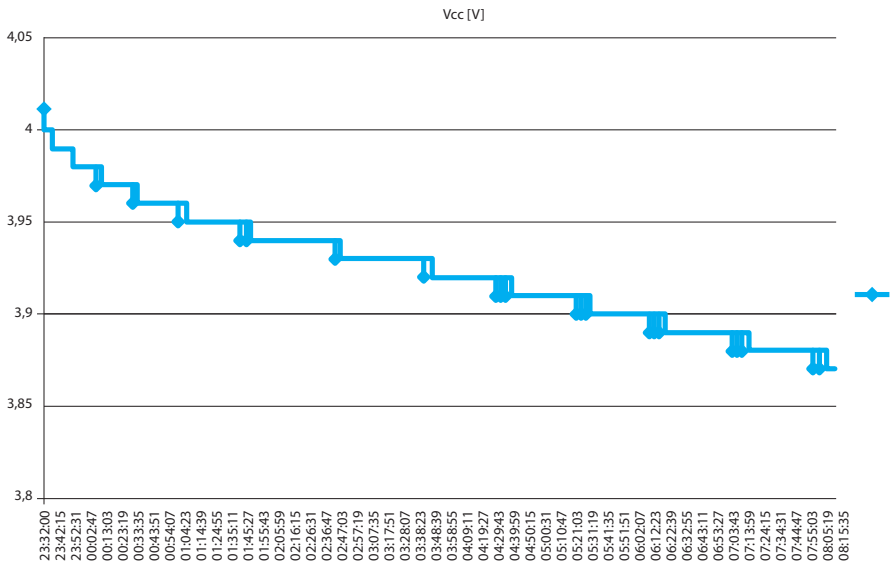
Po zamontowaniu wszystkich komponentów należy sprawdzić czy działa układ ładowania. W tym celu do gniazda USB należy doprowadzić napięcie z interfejsu USB komputera PC lub dołączyć do niego ładowarkę telefonu komórkowego. Jeśli zaświeci się dioda LED1, to bateria ładuje się i układ ładowania działa poprawnie. Następnym krokiem jest zaprogramowanie mikrokontrolera. Jeśli nie dysponujemy podstawką do programowania układów w obudowie TQFP44, należy wykonać kabel do programowania. Jego przykładowy wygląd pokazano na **fotografii 4**.

Karta SD jest połączona z mikrokontrolerem za pomocą sprzętowego interfejsu *SPI*. Dzięki temu na złączu karty są wyprowadzone linie do programowania. Jedyną linią niewykorzystywaną w interfejsie karty SD jest *RESET* i dlatego na płytce drukowanej utworzono pole nazwie *RST*. Na **rysunku 6** pokazano wyprowadzenia karty SD wykorzystywane do programowania mikrokontrolera.

Po zaprogramowaniu mikrokontrolera, po załączeniu zasilania powinna zaświecić się dioda LED2. Jeśli tak się jest, to układ jest gotowy do pracy. Przy pierwszym uruchomieniu należy wgrać na SD plik *config.txt*, ponieważ EEPROM jest czysty. Bez tych ustawień układ nie zadziała. Aby dokładnie ustawić czas układu PCF8583, należy po zapisaniu pliku na karcie włączyć zasilanie 1 sekundę przed ustawioną godziną i minutą. Ta procedura jest konsekwencją początkowego opóźnienia przy starcie programu. Dla przykładu, gdy ustawimy godzinę 18:34, włączamy układ o godzinie 18:33:59. Program zaczyna pracę po ok. 4 sekundach od uruchomienia. Sekwencja startowa wygląda następująco: Po przełączeniu przełącznika ON/OFF na sekundę zapali się dioda LED2, następnie zapali się na 3 sekundy dioda LED3. Gdy zgaśnie układ zaczyna rejestrowanie.

Układ w trybie czuwania pobiera prąd o natężeniu ok. 2,5 mA, natomiast w trakcie zapisu ok. 20 mA. Zapis danych na karcie trwa mniej niż 100 ms. Przy pojemności akumulatora wynoszącej 1200 mAh układ pracuje w sposób ciągły przez ok. 3 doby. Przy włączniku ustawionym w pozycji OFF układ pobiera prąd o natężeniu ok. 50 µA. Wtedy działa tylko układ RTC PCF8583. Po wejściu w tryb *Power Down* rejestrator pobiera ok 0,7 mA, ponieważ wszystkie rezystory podciągające są podłączone do baterii poprzez włącznik. Na **rysunku 6** przedstawiono wykres napięcia zasilania w funkcji czasu w przeciągu 12 godzin, przy włączonym trybie zapisu wszystkich danych co 1 sekundę.

Piotr Rosenbaum
piotr.rosenbaum@gmail.com



Rysunek 6. Wykres napięcia zasilania w funkcji czasu