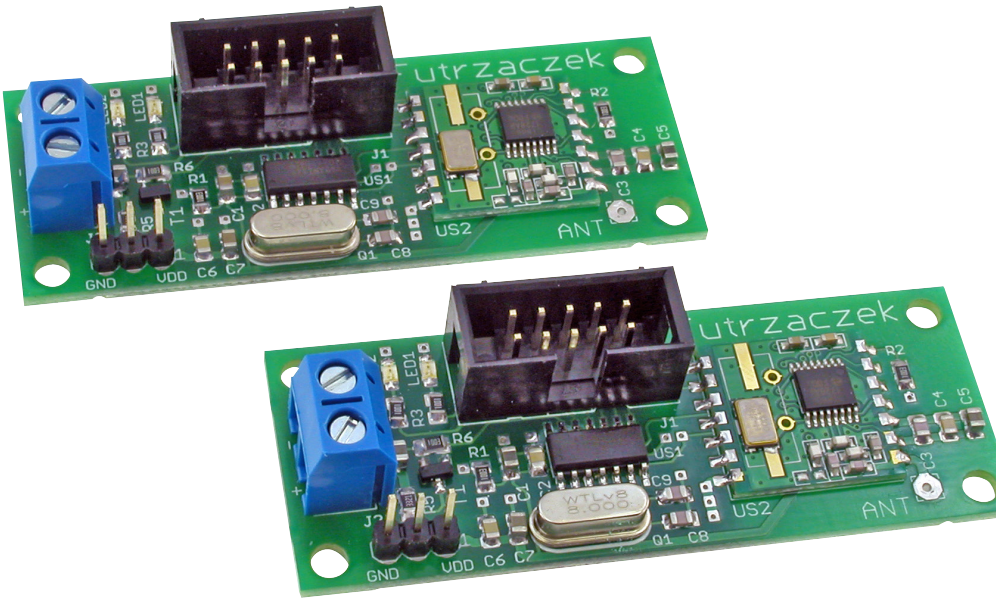
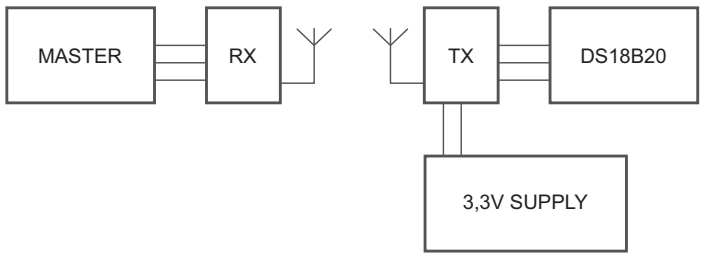
Radiowy przedłużacz DS18B20

Jeden z najpopularniejszych czujników temperatury z interfejsem cyfrowym, jakim niewątpliwie jest DS18B20, musi być połączony z układem nadrzędnym za pomocą dwóch lub trzech przewodów. Prezentowany układ pozwala zrealizować połączenie bezprzewodowo – drogą radiową.





Rysunek 1. Schematyczne przedstawienie użycia opisanych modułów

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

W ofercie AVT* AVT-5720

Podstawowe parametry:
- realizuje połączenie bezprzewodowe magistrali 1-Wire dla czujnika DS18B20,
- reaguje tylko na komendę 0xBE, czyli żądanie o zawartość „scratchpada”,
- współpracuje z systemami zasilanymi napięciem 3,3 lub 5 V,
- komunikacja radiowa w paśmie ISM o częstotliwości 868 MHz,
- wymaga zasilania 3,3 V

Projekty pokrewne na www.media.avt.pl:

AVT-5630	Bezprzewodowy przedłużacz sygnalizatora (EP 7/2018)
AVT-5590	Zdalny włącznik radiowy (EP 6/2017)
AVT-1838	Uniwersalny przedłużacz pilotów RTV (EP 01/2015)
AVT-1840	Włącznik 230 V sterowany dowolnym pilotem na podczerwień (EP 11/2014)
AVT-1815	4-kanałowy przełącznik sterowany dowolnym pilotem IR (EP 8/2014)
AVT-5455	Zdalny włącznik dwukanałowy (EP 6/2014)
AVT-5407	Włącznik sterowany radiowo (EP 8/2013)
AVT-5290	3-kanałowa aparatura do zdalnego sterowania modeli (EP 5/2011)
AVT-1520	Zdalny włącznik radiowy (EP 4/2009)
AVT-2851	Zdalne sterowanie (nie tylko) RC5 (EdW 1/2008)
AVT-1308	Zdalny włącznik 4 urządzeń (EP 8/2006)
AVT-390	8-kanałowy przełącznik sterowany pilotem RC5/SIRC (EP 4/2006)
AVT-559	Radiowy przedłużacz pilotów (EP 2-3/2004)
AVT-517	Radiowy system zdalnego sterowania z kanałem zwrotnym (EP 7-8/2003)
AVT-2482	Czterokanałowe zdalne sterowanie (EdW 4/2001)

* **Uwaga! Elektroniczne zestawy do samodzielnego montażu. Wymagana umiejętność lutowania!**
Podstawową wersją zestawu jest wersja [**B**] nazywana potocznie KIT-em (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 dokumentacji

Kity w których występuje układ scalony wymagający zaprogramowania, mają 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 e-mail: kity@avt.pl.

Wyobraźmy sobie sytuację, w której życzeniem klienta jest panel sterujący, np. wentylacją, umieszczony w jednym kącie pomieszczenia, a czujnik temperatury w zupełnie innym. Na poprowadzenie dodatkowych przewodów nie wyraża zgody. Jeśli w miejscu zamontowania czujnika znajduje się źródło zasilania, np. 12 VDC dla systemu automatycznych rolet, to nasz układ będzie idealnym rozwiązaniem do wybrnięcia z sytuacji. Idea działania jest prosta: jeden moduł (dalej nazywany nadajnikiem, TX) cyklicznie komunikuje się z układem DS18B20 i pozyskane informacje wysyła drogą radiową. Drugi moduł (odbiornik, RX) odbiera te informacje i przechowuje je w pamięci. Podłączony jest w miejsce układu DS18B20 w urządzeniu nadrzędnym. Układ nadrzędny może w każdej chwili, przy użyciu zapytania zgodnego z protokołem 1-wire, zażądać informacji, a wtedy odbiornik przekaże dane, które otrzymał z nadajnika. Schemat blokowy takiego systemu pokazuje **rysunek 1**.

Ponieważ założono, że chodzi o „przedłużenie” jednego czujnika, układ reaguje tylko na jedną komendę: *0xBE*, czyli żądanie o zawartość *scratchpada* (czyli 9 bajtowy blok danych zawierający podstawowe informacje, w tym wynik pomiaru). Pierwsze dwa bajty niosą informację o temperaturze, odczytanie tych wartości wystarczy do obliczenia wyniku. W ten sposób działają najprostsze algorytmy komunikacji z DS18B20, a jeden z nich (napisany w Arduino) można zobaczyć na **listingu 1**. Wszystkie komendy,

Listing 1. Przykładowy kod odpowiadający za komunikację z czujnikiem DS18B20

```
onewire.reset();
onewire.write(0xCC, 0); //skip ROM
onewire.write(0x44, 0); //convert T
delay(750);
onewire.reset();
onewire.write(0xCC, 0); //skip ROM
onewire.write(0xBE, 0); //read scratchpad
data[0] = onewire.read();
data[1] = onewire.read();
temp = (data[0] + (data[1] * 256)) / 16;
```

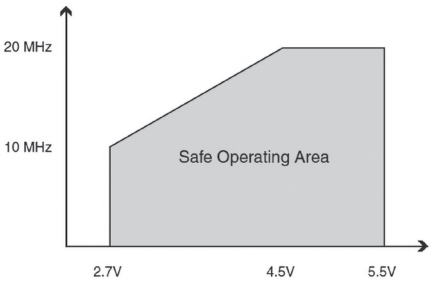
poza zresetowaniem magistrali oraz wspomnianym wcześniej odczytem *scratchpada*, są przez układ pomijane.

W układzie nie zaimplementowano mechanizmów sprawdzających wiarygodność danych otrzymanych drogą radiową. Podczas testów nie zdarzyły się przekłamania, które nie byłyby łatwe do wykrycia – prędzej łączność została całkowicie zerwana, co skutkowało wyzerowaniem bajtów *scratchpada*. Jeżeli jednak dana aplikacja wymaga wysokiej wiarygodności odczytywanych danych, należałoby dodać sprawdzanie sumy kontrolnej, która znajduje się w ostatnim bajcie *scratchpada*.

Budowa

System przedłużający magistralę 1-wire drogą radiową składa się z dwóch modułów – nadawczego i odbiorczego – o identycznej budowie od strony elektrycznej. Różnica tkwi w oprogramowaniu. Schemat ideowy pojedynczego modułu znajduje się na **rysunku 2**. Układem zarządzającym pracą całego modułu, w tym komunikacją radiową, jest mikrokontroler Attiny24A. Jego zasilanie odsprzęgają dwa kondensatory, C1 i C2, umiejscowione blisko odpowiednich wyprowadzeń. Dodatkowy rezystor R1 podciąga wyprowadzenie RESET do napięcia +3,3 V, którym zasilany jest cały moduł.

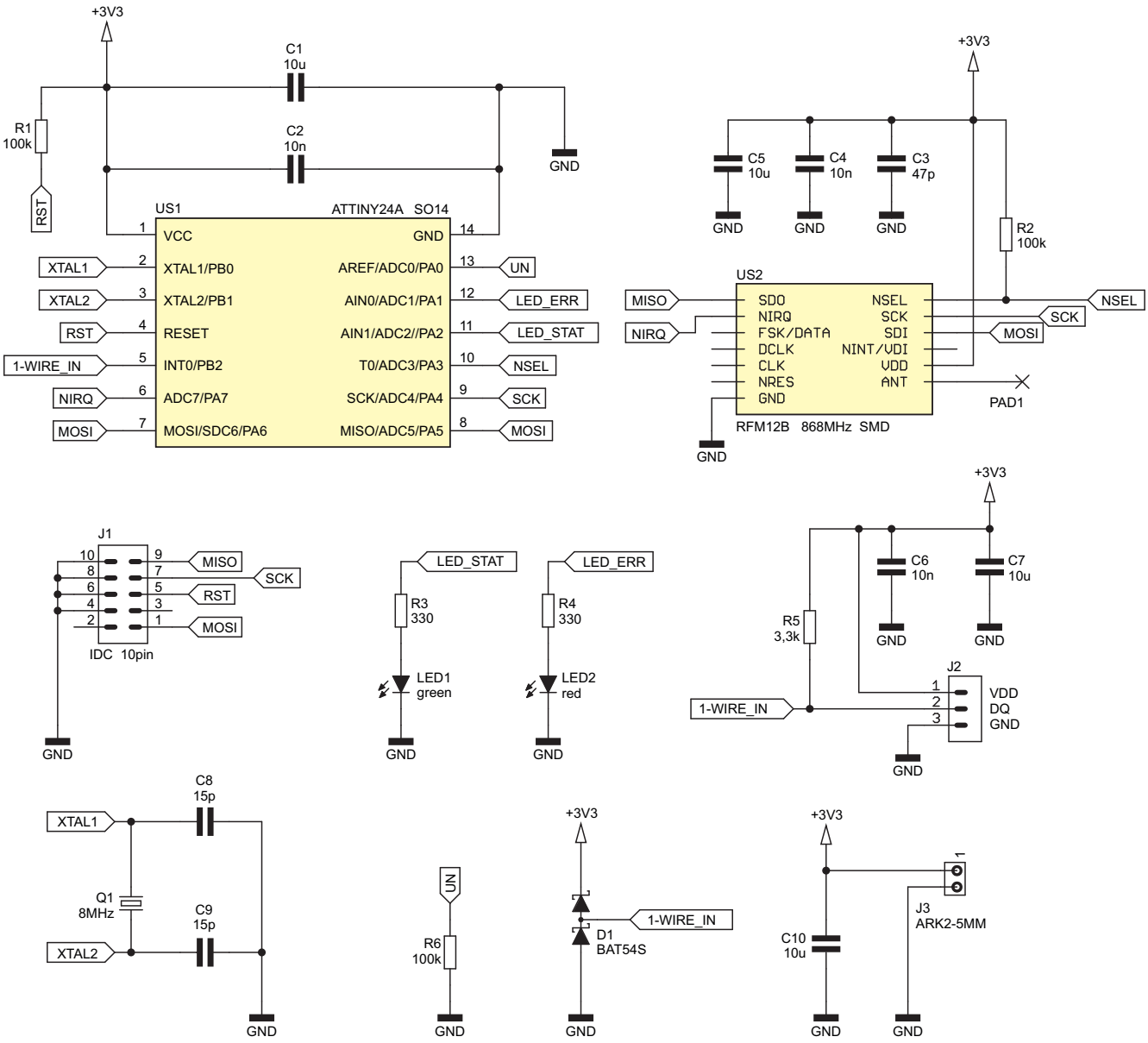
Magistrala 1-wire jest restrykcyjna pod względem zależności czasowych. Dlatego zdecydowano się na stabilizację zegara mikrokontrolera przy użyciu zewnętrznego rezonatora kwarcowego o częstotliwości 8 MHz. Taka, a nie wyższa, wartość częstotliwości została wymuszona relatywnie niskim napięciem zasilania – w tych warunkach,



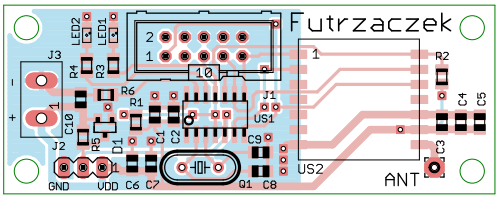
Rysunek 3. Maksymalna częstotliwość zegara w funkcji napięcia zasilającego dla ATtiny24

częstotliwość sygnału zegarowego nie może być większa niż ok. 10 MHz, na co wskazuje wykres z **rysunku 3**.

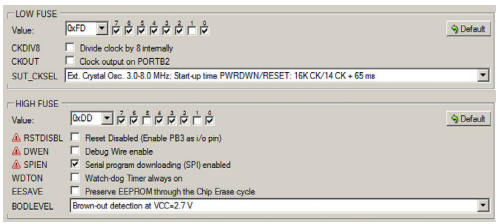
Komunikację bezprzewodową w paśmie ISM na częstotliwości 868 MHz realizuje gotowy układ RFM12B. Komunikuje się z mikrokontrolerem przy użyciu magistrali SPI. Ponieważ te same wyprowadzenia służą do programowania pamięci Flash w Attiny24, eliminacja kolizji została



Rysunek 2. Schemat jednego modułu radiowego „przedłużacza" DS18B20



Rysunek 4. Schemat płytki PCB wraz z rozmieszczeniem elementów



Rysunek 5. Konfiguracja *fusebitów* w programie BitBurner

zrealizowana poprzez dodanie rezystora R2, który podciąga wyprowadzenie NSEL do dodatniej linii zasilania. Oznacza to, że wyprowadzenia dedykowane dla SPI są domyślnie w stanie wysokiej impedancji. Mikrokontroler w odpowiednich momentach ściąga potencjał tej linii do zera, co inicjuje możliwość komunikacji z RFM12B. Wyprowadzenie anteny znajduje się na polu lutowniczym PAD1. Uznano, że wygodniej będzie dolutować zewnętrzną antenę w postaci odcinka przewodu o odpowiedniej długości niż borykać się z umiejscowieniem w obudowie płytki powiększonej o antenę wewnętrzną.

Złącze J1 służy do programowania pamięci Flash mikrokontrolera. Układ wyprowadzeń jest zgodny z typowym standardem ISP KANDA, do którego dedykowana jest większość niewielkich programatorów na złącze USB. Do wskazywania stanu pracy modułu służą dwie diody LED. W zależności od tego, czy jest to nadajnik, czy też odbiornik, ich świecenie bądź miganie ma różne znaczenie, co zostanie szerzej opisane w dalszej części artykułu.

Czujnik typu DS18B20 (w przypadku modułu nadajnika) lub dedykowany układ nadrzędny (w przypadku odbiornika) podłącza się do złącza J2. Jeżeli w systemie znajduje się rezystor podciągający na magistrali 1-wire, o odpowiedniej wartości, R5 należy wymontować.

Podwójna dioda *Schottky* typu BAT54S została użyta do zabezpieczenia delikatnego wejścia mikrokontrolera przed impulsami, które mogą indukować się w przewodzie sygnałowym. Przez większość czasu jest on obciążony relatywnie wysoką impedancją (rzędu kΩ), więc mogą występować zakłócenia.

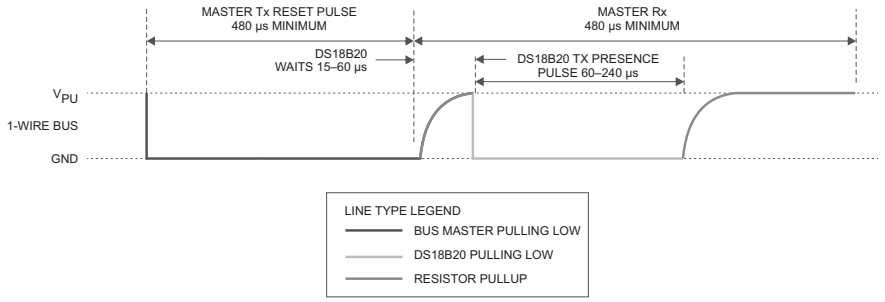
Cały układ pracuje pod napięciem 3,3 V, lecz zadziała poprawnie z układem nadrzędnym, zasilanym napięciem 5 V. Rezystor podciągający powinien być wówczas dołączony do wyższego napięcia. Dzięki temu, w stanie spoczynku, potencjał linii 1-wire wyniesie ok. 3,6 V (3,3 V napięcia zasilającego +0,3 V napięcie przewodzenia górnej diody D1). Taka wartość zostanie poprawnie zinterpretowana jako logiczna „1" zarówno przez moduł odbiornika, jak i układ nadrzędny, o ile próg detekcji tego poziomu logicznego leży dostatecznie nisko. Przykładowo, dla układu ATmega328 zasilanego napięciem 5 V wynosi 3 V, czyli cały system zadziała poprawnie.

Zasilanie dla modułu może zostać dołączone w dwóch miejscach: zarówno do złącza śrubowego typu ARK (J3), jak i szpilkowego (J2). Nie przewidziano stabilizatora napięcia, ponieważ zmniejszyłoby to uniwersalność płytki – w systemie nadrzędnym prawdopodobnie będzie odpowiednie napięcie 3,3 V. W przeciwnym razie do dyspozycji będzie napięcie zdecydowanie wyższe, które trzeba będzie obniżyć dodatkową przetwornicą impulsową (dla zmniejszenia strat mocy).

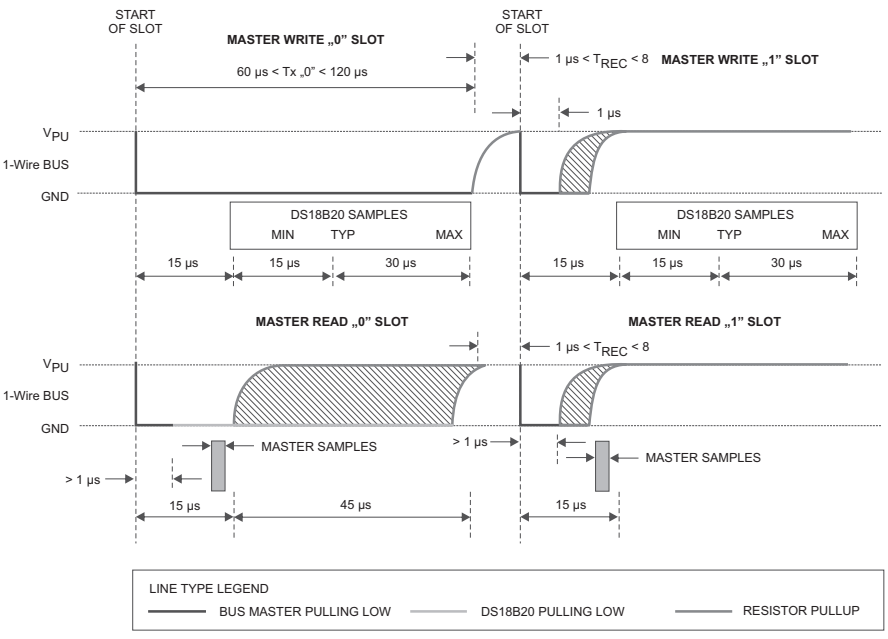
Montaż i uruchomienie

Oba moduły, nadajnik i odbiornik, realizowane są na tej samej płytce drukowanej o wymiarach 25×65 mm. Schemat płytki PCB wraz z rozmieszczeniem elementów pokazuje **rysunek 4**. Obsada elementów dla obu modułów jest identyczna, z ewentualną różnicą w postaci wspomnianego wcześniej rezystora R5. Po przylutowaniu elementów w obudowach do montażu powierzchniowego (SMD), na płytce powinien znaleźć się kwarc XTAL1 oraz wszystkie złącza. Na samym końcu polecam przylutować moduł RFM12B, aby nie uległ uszkodzeniu podczas montażu pozostałych części.

Prawidłowo zmontowane moduły nie wymagają czynności uruchomieniowych. Należy pamiętać o zamontowaniu anten. W najprostszym wykonaniu, funkcję anteny może pełnić odcinek przewodu w izolacji



Rysunek 6. Przebieg czasowy komendy inicjującej czujnik



Rysunek 7. Transmisja do przesyłu komend i danych

Wykaz elementów: (taki sam dla nadajnika i odbiornika)

Rezystory:
R1, R2, R6: 100 kΩ SMD0805
R3, R4: 330 Ω SMD0805
R5: 3,3 kΩ SMD0805 (opis w tekście)

Kondensatory:
C1, C5, C7, C10: 10 µF/10 V SMD0805
C2, C4, C6: 10 nF SMD0805
C3 : 47 pF SMD0805
C8, C9: 15 pF SMD0805

Półprzewodniki:
D1: BAT54S SOT23
LED1: LED zielona SMD0805
LED2: LED czerwona SMD0805
US1: Attiny24A SO14
US2: RFM12B 868 MHz SMD

Inne:
J1: IDC 10pin 2,54 mm proste
J2: goldpin męski 3 pin 2,54 mm
J3: ARK2/500

Listing 2. Obsługa urządzenia nadrzędnego, kluczowe fragmenty kodu

```
//sygnał RESET
uint8_t reset_pulse(void){ //1 - układ obecny, 0 - błąd
          uint8_t presence;
          LOW_1WIRE;
          CLEAR_1WIRE;
          _delay_us(480); //nadaj sygnał RESET, min. 480us
          SET_1WIRE; //zwolnij linię
          _delay_us(60); //odczekaj na odpowiedź, min. 60us
          //weryfikacja odpowiedzi
          if(IS_LOW){presence = 1;} else {presence = 0;}
          _delay_us(440); //odczekaj na zanik odpowiedzi
           //weryfikacja po zaniku
          if(IS_HIGH && presence){presence = 1;}
          else {presence = 0;}
          return presence;
}

//nadanie jednego bitu
void send_bit(uint8_t bit){
          LOW_1WIRE;
          CLEAR_1WIRE;
          _delay_us(1); //nadaj 1 krótkim impulsem
          if(bit == 1){ SET_1WIRE; }
          _delay_us(70); //nadaj 0 dłuższym impulsem
          SET_1WIRE;
}

//odbiór jednego bitu
uint8_t read_bit(void){
          uint8_t bit;
          LOW_1WIRE;
          //wystaw zbocze opadające na krótką chwilę
          CLEAR_1WIRE;
          _delay_us(1);
          SET_1WIRE; //zwolnij linię
          //odczekaj na ew. „przeciągnięcie” impulsu przez czujnik
          _delay_us(10);
          if(IS_HIGH){bit = 1;} else {bit = 0;} //sprawdź stan
          _delay_us(40); //uzupełnienie całego
          return bit;
}

//nadanie jednego bajtu
void send_byte(uint8_t byte){
          uint8_t i=8;
          while(i--){
                    send_bit(byte&1);
                    byte>>=1;
          }
}

//odbiór jednego bajtu
uint8_t read_byte(void){
          uint8_t byte = 0;
          uint8_t i=8;
          while(i--){
                    byte>>=1;
                    byte|=(read_bit()<<7);
          }
          return(byte);
}
```

o długości ok. 17 cm – będzie anteną półfalową dla częstotliwości 868 MHz.

Po podłączeniu zasilania o napięciu 3,3 V (dobrze filtrowane i stabilizowane), mikrokontrolery w obu modułach należy zaprogramować wsadem pamięci Flash. Ponadto, należy zmienić ustawienia *fusebitów* na wartości:

Low Fuse = 0xFD

High Fuse = 0xDD

Szczegółowa konfiguracja jest widoczna na **rysunku 5**, gdzie znajduje się zrzut okna programu BitBurner. Ustawienie progu zadziałania *BOD* na 2,7 V zapobiegnie powstaniu błędów podczas uruchamiania, kiedy napięcie zasilające narasta zbyt wolno. Po tych czynnościach, moduły są gotowe do działania. Średni pobór prądu wynosi 27 mA dla modułu nadawczego oraz 14 mA dla modułu odbiorczego.

Eksploatacja

Po włączeniu zasilania nadajnika, z dołączonym czujnikiem DS18B20, co 750 ms następuje odczyt 9 bajtów *scratchpada*. Dwa pierwsze bajty zawierają informację o temperaturze z najwyższą dostępną rozdzielczością, tj. 12 bitów. Jednocześnie, co 100 ms uruchamiany jest nadajnik, który wysyła zawartość *scratchpada*. Tak częste transmisje służą eliminacji przypadkowego przekłamania, którego wyeliminowanie przy użyciu mechanizmu zwrotnych potwierdzeń byłoby skomplikowane w realizacji, a jednym z założeń projektu była prostota działania.

Diody LED1 i LED2 sygnalizują stan pracy modułu nadajnika. Opis sygnalizacji prezentuje **tabela 1**. Błąd komunikacji z dołączonym układem DS18B20 jest rozpoznawany poprzez brak prawidłowego sygnału obecności (*presence*), wystawianego przez czujnik po zresetowaniu magistrali.

Odbiornik cały czas nasłuchuje danych emitowanych przez nadajnik oraz reaguje na zapytania układu nadrzędnego wysyłane magistralą 1-wire. Jego działanie różni się od modułu nadawczego, dlatego diody LED również mają odmienne znaczenie – **tabela 2**. Dioda LED1 zaświeca się, kiedy przez min. 1,5 s nie zarejestrowano transmisji z nadajnika. Transmisja realizowana jest co 100 ms i każde poprawnie odebrane dane powodują zerowanie licznika czasu. Odbiornik nie weryfikuje poprawności odebranych danych, jest to zadanie układu nadrzędnego. Obserwując mignięcia diody LED2 można zauważyć, że komunikacja odbiornika z układem nadrzędnym przebiega prawidłowo. Opisany układ nie reaguje na takie komendy jak *Search ROM* (*0xF0*) czy *Write Scratchpad* (*0x4E*). Realizuje tylko podstawowy cel, czyli umożliwia odczyt danych z czujnika DS18B20 za pośrednictwem drogi radiowej.

Dla dociekliwych

Magistrala 1-wire stawia ostre wymagania czasowe wobec dołączonych układów. Bezpośrednie przesyłanie stanów logicznych drogą radiową jest niemożliwe, ponieważ w krytycznych przypadkach, decyzja o stanie magistrali musi zostać podjęta w ciągu pojedynczych mikrosekund. Użyte moduły RFM12B są do tego zdecydowanie zbyt wolne (chociażby ze względu na konieczność przełączania pomiędzy nadawaniem a odbiorem), toteż zdecydowano o emulacji urządzenia 1-wire po stronie odbiornika. Zanim przejdziemy

Tabela 1. Opis stanów modułu nadawczego sygnalizowanych diodami LED

Dioda	Sygnalizacja	Opis
LED1	Krótkie mignięcia	DS18B20 działa prawidłowo
	Świecenie ciągłe	Błąd komunikacji z DS18B20
LED2	Bardzo krótkie mignięcia	Praca nadajnika radiowego

Tabela 2. Opis stanów modułu odbiorczego sygnalizowanych diodami LED

Dioda	Sygnalizacja	Opis
LED1	Świecenie ciągłe	Brak sygnału z nadajnika
LED2	Bardzo krótkie mignięcia	Wysłanie scratchpada po magistrali 1-wire

Tabela 3. Opis danych bloku *scratchpad*

Numer bajtu	Znaczenie
0	Temperatura – młodsza część
1	Temperatura – starsza część
2	Górny próg alarmu temperatury (z EEPROM)
3	Dolny próg alarmu temperatury (z EEPROM)
4	Bajt konfiguracyjny
5	Wartość stała = 255 (0xFF)
6	Zarezerwowany
7	Wartość stała = 16 (0x10)
8	CRC (suma kontrolna)

Listing 3. Emulacja urządzenia podrzędnego, kluczowe fragmenty

```
//definicje wymagań czasowych dot. magistrali 1-wire [us]
#define TIME_RESET_PULSE 480
#define TIME_WAIT_AFTER_RESET_PULSE 30
#define TIME_PRESENCE_PULSE 120
#define TIME_ZERO_MIN 60
#define TIME_ZERO_MAX 120
#define TIME_ZERO_SEND 15

//definicje czasów odmierzanych Timerem1
#define TCNT1_SAMPLE 65535-30
#define TCNT1_WAIT_AFTER_RESET_PULSE 65535-30
#define TCNT1_PRESENCE_PULSE 65535-120

//definicje czasów timeout odmierzanych Timerem0
#define TIMEOUT_15MS 255-234

//definicje kolejnych stanów układu
//stan spocznkowy - oczekiwanie na sygnał RESET
#define STATE_IDLE 0
//oczekiwanie na zakończenie impulsu RESET od mastera
#define STATE_RESET_WAIT 1
#define STATE_SEND_PRESENCE 2 //wysłanie impulsu PRESENCE
#define STATE_AFTER_RESET 3 //oczekiwanie na rozkazy po resecie
#define STATE_SEND 4 //wysyłanie bajtów na magistralę
volatile uint8_t state = STATE_IDLE;

volatile uint8_t read_bit_cnt = 0; //licznik odczytanych bitów
volatile uint8_t read_byte = 0; //bajt tworzony z kolejnych bitów
volatile uint8_t read_byte_ready = 0;  //gotowy, odczytany bajt
//licznik do odbioru kolejnych bajtów scratchpada
volatile uint8_t rec_cnt;

volatile uint8_t send_bit_cnt = 0; //numer bitu do wysyłki
volatile uint8_t send_byte_cnt = 0; //numer bajtu do wysyłki
volatile uint8_t send_scratch[10] = {0}; //kompletny scratchpad

// INT0 do obsługi 1-wire
ISR(INT0_vect)
{
          if(IS_LO){ //jeżeli jest stan niski na linii 1-wire
                    //jeżeli trwa wysyłka bajtów scratchpada
                    if(state == STATE_SEND){
                              LED_STAT_1; //załącz diodę STAT
                              //jezeli ma zostać nadane zero
                              if((send_scratch[send_byte_cnt] & (1<<send_bit_cnt)) == 0){
                                        WIRE_LO;
                                        TCNT1 = 0;
                                        //przeciagnij impuls do 15us
                                        while(TCNT1 < TIME_ZERO_SEND);
                                        WIRE_HI;
                              }
                              send_bit_cnt++; //przesun wysylany bit o 1
                              //po wysłaniu całego bitu, przejdą do następnego bajtu
                              if(send_bit_cnt > 7){ send_byte_cnt++; send_bit_cnt = 0;}
                              //po wysłaniu wszystkich bajtów, wróć do stanu IDLE
                              if(send_byte_cnt > 8){ state = STATE_IDLE; LED_STAT_0; }
                    }
                    //w stanie spoczynku wyczyść Timer1 i
                    if(state == STATE_IDLE){TCNT1 = 0; LED_STAT_0;} wyłącz diodę STAT
                    //jeżeli magistrala jest zainicjowana,
                    //odmierzaj czas do zbierania próbek
                    if(state == STATE_AFTER_RESET){ TCNT1 = TCNT1_SAMPLE; }

                    TCNT0 = TIMEOUT_15MS; //wyzeruj licznik timeout
          }

          if(IS_HI) { //jeżeli jest stan wysoki
                    //jeżeli wykryto impuls RESET o prawidlowej długości (od mastera)
                    if(TCNT1 >= TIME_RESET_PULSE && state == STATE_IDLE) {
                              //przełącz stan na oczekiwanie do wygenerowania własnego
                              state = STATE_RESET_WAIT; impulsu (presence)
                              //oczekiwanie na wygenerowanie własnego impulsu
                              TCNT1 = TCNT1_WAIT_AFTER_RESET_PULSE;
                              //wyzeruj licznik odebranych bitów
                              read_bit_cnt = 0;
                              read_byte = 0; //wyzeruj odebrany bajt
                              //wyzeruj licznik timeout
                              TCNT0 = TIMEOUT_15MS;
                    }
          }
}

// Tim1 do odmierzania czasu nadawanych
//i odbieranych impulsów impulsów
ISR(TIM1_OVF_vect){
           //po pelnym resecie: układ jest gotowy do odbioru komendy
          if(state == STATE_AFTER_RESET){
                    TCNT0 = TIMEOUT_15MS; //wyzeruj licznik timeout
                     //master wysyła 1, jeżeli 0 to nie rób nic
                    if(IS_HI){read_byte |= (1 << read_bit_cnt);}
                    //zainkrementuj licznik odebranych bitów
                    read_bit_cnt++;
                    //jeżeli zebrano cały bajt
                    if(read_bit_cnt > 7) {
                              //wyzeruj licznik odebranych bitów
                              read_bit_cnt = 0;
                              //przekopiuj gotowy bajt
                              read_byte_ready = read_byte;
                               //wyzeruj odebrany bajt
                              read_byte = 0;
                              //jeżeli odebrano komendę odczytu scratchpada
                              if(read_byte_ready == 0xBE) {
                                        state = STATE_SEND;
                                        send_bit_cnt = 0;
                                        send_byte_cnt = 0;
                              }
                    }
          }
          //po wyslaniu sygnalu presence
          if(state == STATE_SEND_PRESENCE) {
```

do omówienia kodów źródłowych, zastanówmy się nad zasadą działania tej magistrali. Można tu wyróżnić dwa typy przesyłanych informacji. Pierwszym jest inicjalizacja (*reset*) wszystkich urządzeń podrzędnych, którego przebieg czasowy jest widoczny na **rysunku 6**. Najpierw układ nadrzędny wymusza stan niski na linii, zaś potem czynią to układy podrzędne, po odczekaniu pewnego czasu. Taka „odpowiedź" jest dla układu nadrzędnego znakiem, że ma podłączonych działających „rozmówców". Zatem należy sprawdzić stan linii 1-wire po min. 60 µs (powinien być niski), a potem jeszcze raz, po min. 240 µs – wtedy stan logiczny powinien wrócić do wysokiego. Cała procedura musi trwać min. 960 µs.

Do przesyłu komend i danych stosowany jest inny rodzaj transmisji, szczegóły przedstawia **rysunek 7**. Jest różny w zależności od kierunku transmisji. Urządzenie nadrzędne zawsze rozpoczyna bit. Kiedy nadaje, długość trwania stanu niskiego determinuje znaczenie bitu:

- poniżej 15 µs, jest to logiczna „1"
- od 60 µs do 120 µs, jest to logiczne „0"

Układy podrzędne sprawdzają stan linii po 15...60 µs (typowo 30 µs) od zarejestrowania zbocza opadającego. Krótki impuls zdąży się w tym czasie zakończyć i stan linii wyniesie powróci do „1". Dłuższy impuls zostanie zaś zinterpretowany w tym momencie jako „0".

Wysyłka bitów przez urządzenie podrzędne (poprzedzona odpowiednią komendą) polega na tym, że urządzenie nadrzędne wystawia zbocza opadające. Jeżeli układ podrzędny ma wysłać „1", wówczas nie robi nic i po ok. 10...13 µs od rozpoczęcia impulsu, stan logiczny linii zdąży powrócić do stanu wysokiego, co rejestruje układ nadrzędny. Jeżeli zaś ma zostać wysłane „0", układ podrzędny „przeciąga" uchwycone zbocze opadające do 15 µs, po czym zwalnia linię. Układ nadrzędny, który w tym czasie sprawdza stan linii, odczyta „0". Co istotne, to układ nadrzędny decyduje o tym, ile bitów odczyta. Cały *scratchpad* składa się z 9 bajtów (**tabela 3**), lecz nic nie stoi na przeszkodzie, by odczytać tylko pierwsze dwa – tyle wystarczy do odczytu temperatury. Użyteczny może być również ostatni bajt – zawiera sumę kontrolną (*CRC*), dzięki której można dowiedzieć się, czy otrzymana informacja jest poprawna.

Obsługa magistrali 1-wire, którą trzeba było zaimplementować w nadajniku, jest dobrze znana i opisana w wielu miejscach. Ponieważ pracy nadajnika nie zaburzają jakiekolwiek przerwania zewnętrzne, można było zrealizować ją za pomocą zwykłych opóźnień typu *delay()*. Długości trwania opóźnień dobrano doświadczalnie, za pomocą oscyloskopu, aby zgadzały się z tymi, których wymaga nota katalogowa układu. Kod został napisany w języku C, istotne fragmenty pokazuje **listing 2**.

Listing 3. cd.

```
                    //przejdą w tryb odbioru
                    state = STATE_AFTER_RESET;
                    WIRE_HI; //zwolnij linię
          }
          //po odczekaniu, nadaj sygnal presence
          if(state == STATE_RESET_WAIT) {
                    state = STATE_SEND_PRESENCE;
                    TCNT1 = TCNT1_PRESENCE_PULSE;
                    WIRE_LO;
          }
}

// Tim0 do odmierzania czasu timeout
ISR(TIM0_OVF_vect){
          //jezeli doszlo do przepelnienia tego
          //licznika, wroc do stanu IDLE
          state = STATE_IDLE;
}

int main(void) //pętla główna
{
          //przerwania od przepelnienia
          TIMSK1 |= (1 << TOIE1);
          //Timer1 odlicza co 1us, preskaler 8
          TCCR1B |= (1 << CS11);
          TCNT1 = 0;
          //przerwania od przepelnienia
          TIMSK0 |= (1 << TOIE0);
          //Timer0 odlicza co 64us, preskaler 256
          TCCR0B |= (1<<CS02);
          TCNT0 = 0;
          //przerwania zboczem narastajıcym i opadajıcym od INT0
          MCUCR |= (1<<ISC00);
          GIMSK |= (1 << INT0);

          sei();
}
```

Problem stanowiła emulacja urządzenia podrzędnego. Tutaj konieczne było użycie przerwania sprzętowego do wykrywania zboczy opadających. Ponadto, w przerwaniach nie należy używać funkcji opóźniających, więc odmierzanie czasu trzeba było zrealizować z zastosowaniem liczników, zwłaszcza, że równocześnie może trwać odbiór danych z nadajnika. Rozwiązano to za pomocą prostej maszyny stanów, której najważniejsze fragmenty kodu znajdują się na **listingu 3**. Widoczna jest obsługa przerwań, deklaracja zmiennych i stałych oraz inicjalizacja liczników i przerwania zewnętrznego.

W stanie spoczynku (*IDLE*), układ oczekuje na zbocze opadające. W momencie detekcji zerowany jest *Timer1*. Kiedy wystąpi zbocze narastające, sprawdzany jest stan licznika. Jeżeli był to prawidłowy impuls (tj. o długości min. 480 µs), układ przechodzi do generacji impulsu obecności (*presence*) po odczekaniu ustalonego czasu.

Odbiór danych jest możliwy po wykonaniu pełnej sekwencji inicjalizacji magistrali. Na zboczach opadających jest zadawana wartość licznika *Timer1*, który, w momencie przepełnienia, sprawdza stan linii 1-wire, odczytując w ten sposób wartość odebranego bitu. Jeżeli odebrana sekwencja bitów jest bajtem komendy odczytu zawartości *scratchpad*, układ przechodzi w stan gotowości do wysyłki danych. Każde zbocze opadające generowane przez urządzenie nadrzędne na linii 1-wire, jest analizowane pod kątem ewentualnego przedłużenia do długości odpowiadającej bitowi o wartości „0”.

Licznik *Timer0* odpowiada za automatyczny powrót układu do stanu spoczynku. Podczas komunikacji z układem nadrzędnym, jego wartość jest ustawiana tak, aby uległ przepełnieniu po 15 ms. Jeżeli na linii 1-wire nic się nie wydarzy, to układ przejdzie do stanu początkowego (*idle*), więc jego działanie można porównać do układu *watchdog*. Ten licznik przepełnia się cyklicznie również w stanie oczekiwania, więc przywróci układ do tego stanu w razie niespodziewanego pojawienia się stanu niskiego na linii – spowodowanego np. zakłóceniem.

Przedstawione programy można rozbudować np. o obsługę komendy porównania ROM (*Match ROM*), która stanowi bazę umożliwiającą prawidłowy odczyt temperatury z czujnika oraz emulację jego dwóch najważniejszych funkcji.

Michał Kurzela, EP