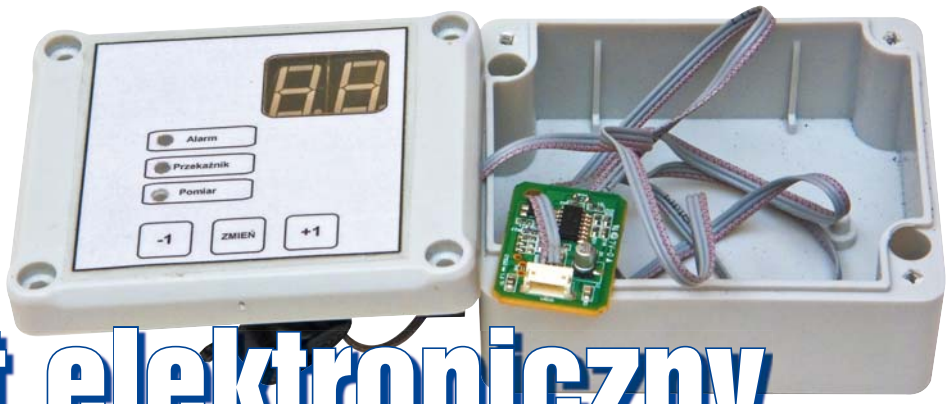
kit 3026 AVT



Higrostat elektroniczny

Urządzenie będące przedmiotem niniejszego artykułu, jak sama jego nazwa wskazuje, zostało przeznaczone do kontrolowania wilgotności. Obecność przekaźnika, do którego można dołączyć układ wykonawczy, np. nawilżacz powietrza, sprawia, że układ może stanowić prosty, autonomiczny regulator wilgotności np. w pokoju, terrarium, szklarni, etc. Zakres regulacji zawiera się w przedziale od 10 do 95%. Pomiar wilgotności został oparty o gotowy, zintegrowany czujnik, dający na swoim wyjściu napięcie proporcjonalne do mierzonej wartości. W prototypie wykorzystany został czujnik RHSM1. Urządzenie wyposażono w czytelne wyświetlacze 7-segmentowe, przeznaczone do prezentowania bieżących pomiarów i ustawianych parametrów. Higrostat ma ponadto wyjście sterowane tranzystorem, przeznaczone do podłączenia buzzera, dzięki czemu możliwe jest generowanie alarmów, gdy wilgotność przekroczy ustalony próg.

W urządzeniu wykorzystany został mikrokontroler z rodziny AVR, który wymaga zaprogramowania. W Elportalu wśród materiałów dodatkowych do tego numeru udostępniony został gotowy kod wynikowy, który wystarczy zapisać w pamięci bez potrzeby wprowadzania żadnych modyfikacji. W przypadku problemów można poprosić o pomoc bardziej doświadczoną osobę lub zakupić zaprogramowany układ w Sklepie AVT. Bardziej zaawansowani Czytelnicy mogą wykorzystać udostępniony również pełny kod źródłowy do wprowadzenia własnych modyfikacji, aby dostosować układ do własnych potrzeb lub dodać nowe funkcje.

Opis układu

Na schemacie pokazanym na **rysunku 1** można wyróżnić cztery bloki funkcjonalne: zasilacz, mikrokontroler z jego najbliższym otoczeniem, blok wyświetlaczy oraz układy peryferyjne. Każdy z bloków ma sygnatury zaczynające się od innej cyfry.

Zasilacz posiada diodę D10, zabezpieczającą układ przed odwrotną polaryzacją. Kondensatory filtrują napięcia, usuwając z nich tętnienia oraz zakłócenia impulsowe. Część podzespołów do poprawnej pracy wymaga napięcia 5V, które jest uzyskiwane ze stabilizatora 7805.

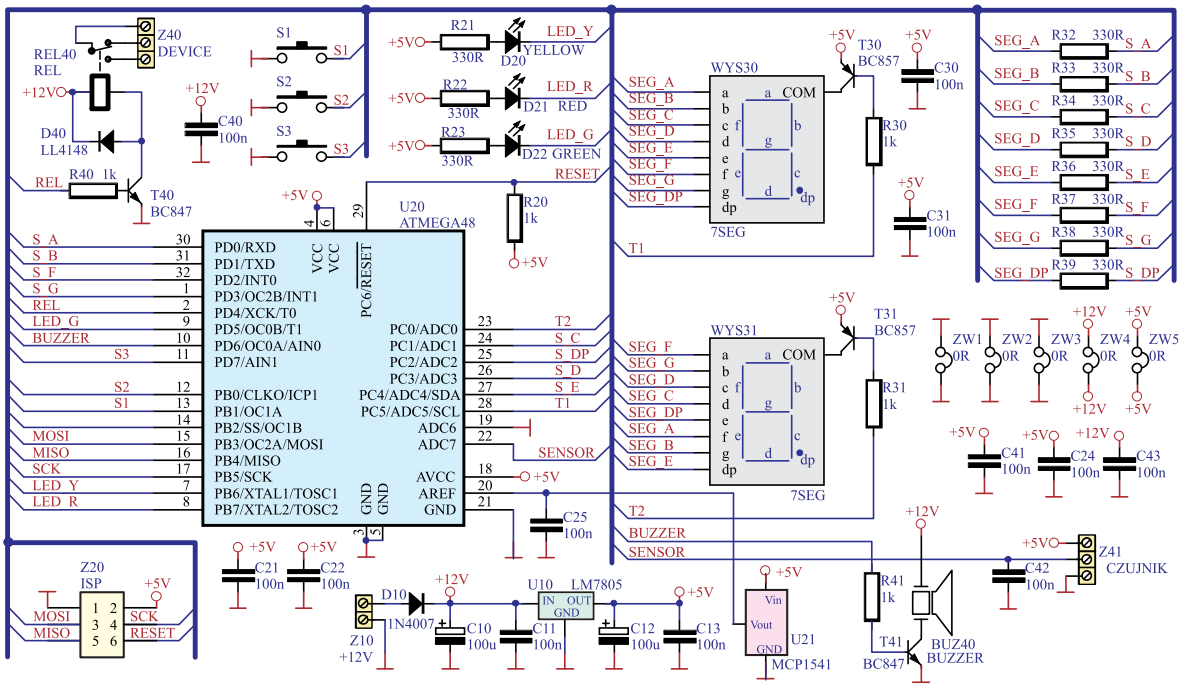
Kolejną grupę elementów stanowi mikrokontroler i jego otoczenie. Wykorzystany został układ ATmega48. Ma on 4kB pamięci programu i 512B pamięci danych, co pozwoliło na komfortowe napisanie programu w języku C++. Większość portów I/O została wykorzystana, więc użycie układu w mniejszej obudowie nie było, niestety, możliwe. Warto zauważyć, że na schemacie nie ma rezonatora kwarcowego. Przeznaczenie urządzenia sprawia, że nie ma potrzeby dokładnego odmierzania czasu, więc w zupełności wystarczył wbudowany generator RC. Takie podejście upraszcza projekt płytki, zmniejsza jej rozmiar i obniża cenę całego urządzenia.

ATmega48 ma wbudowane źródło napięcia odniesienia dla przetwornika analogowo-cyfrowego, ale jest ono mało dokładne i postanowiłem zastosować zewnętrzny układ z oferty Microchipa: MCP1541. Dzięki temu napięcie referencyjne wynosi 4096mV z dokładnością do 1%, i na jeden bit przetwornika przypadają dokładnie 4mV, gdyż przetwornik jest 10-bitowy (1024 stany).

Komunikacja z użytkownikiem odbywa się za pomocą przycisków S1, S2 oraz S3, które pozwalają ustawić pożądaną wartość wilgotności i próg alarmu. Diody LED mają za zadanie sygnalizować stan, w jakim znajduje się procesor: ustawiania wilgotności, ustawiania progu alarmu czy normalnej pracy z wyświetlaniem bieżącej wilgotności.

Na płytce znajduje się złącze Z20, które umożliwi zaprogramowanie mikrokontrolera w systemie (tzn. przylutowanego już do płytki) za pomocą programatora ISP. W razie problemów warto sprawdzić w dokumentacji programatora, jakie są wymagania dotyczące linii RESET, gdyż rezystor podciągający 1k użyty w urządzeniu może w niektórych przy-

Rys. 1

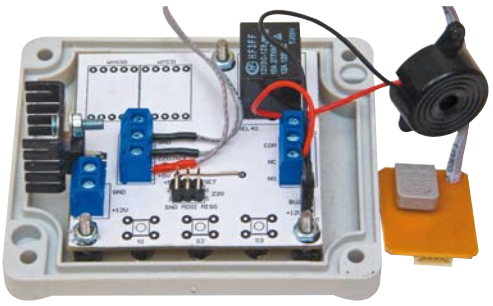


padkach uniemożliwić zapisanie programu do pamięci. Warto więc zaprogramować mikrokontroler przed wlutowaniem R20.

Wyświetlanie wartości zostało zrealizowanie w oparciu o dwa wyświetlacze 7-segmentowe. Urządzenie jest na tyle proste, że nie było potrzeby stosowania wyświetlacza LCD. Do sygnalizacji trybu pracy służą, jak już wspomniano, diody LED, natomiast proste komunikaty określające działanie przekaźnika i buzzera są sygnalizowane literami r, b, i, n, które również da się wyświetlić. Wyświetlacze są sterowane w sposób multipleksowany, co oznacza, że w danej chwili włączony jest tylko jeden i wyświetlany jest na nim zadany znak. Najłatwiej to zrealizować, umieszczając obsługę wyświetlaczy w przerwaniu wyzwalanym okresowo np. po przepełnieniu licznika. Właśnie takie podejście zostało wykorzystane w tym przypadku. Dzięki multipleksowaniu, zamiast 16 linii wystarczyło tylko 10, dzięki czemu liczba linii IO w układzie ATmega48 okazała się wystarczająca. Miłym skutkiem ubocznym jest uproszczenie płytki drukowanej i łatwiejsze prowadzenie ścieżek. Uważni Czytelnicy zauważą, że dodatkowe uproszczenie płytki zrealizowano w jeszcze jeden sposób – oba wyświetlacze są podłączone inaczej. Przykładowo port PD0 włącza segment A w wyświetlaczu WYS30, ale w wyświetlaczu WYS31 ten sam port włącza segment F. Takie podejście komplikuje program, bo potrzebne są dwa osobne zestawy kodów do sterowania każdym z wyświetlaczy, ale dzięki dużej pojemności pamięci programu takie uproszczenie było jak najbardziej sensowne.

Wyświetlacze są włączane przy użyciu tranzystorów T30 oraz T31 pracujących tutaj jako klucze, czyli spełniających jedynie funkcję włącz/wyłącz. Rezystory R32...R39 ograniczają prąd poszczególnych segmentów, aby nie doszło do ich uszkodzenia na skutek przepływu zbyt dużego prądu. Złącze Z40 służy do podłączenia urządzenia wykonawczego, które będzie sterowane za pomocą przekaźnika REL40. Dioda D40 zabezpiecza tranzystor T40 przed uszkodzeniem.

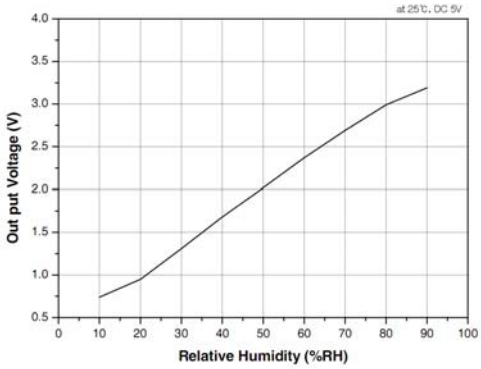
Tranzystor T41 steruje pracą dowolnego buzzera przeznaczonego na napięcie 12V, który można podłączyć do złącza szpilkowego oznaczonego jako BUZ40. Jeżeli Czytelnik nie planuje korzystać z funkcji alarmowania, elementy R41, BUZ40 oraz T41 nie muszą być lutowane. Czujnik jest podłączany do złącza Z41. W prototypie wykorzystano układ RHSM1, który na swoim wyjściu daje liniową zależność napięcia od wilgotności. Charakterystykę tę pokazano na **rysunku 2**.



Oprogramowanie

Pełny kod źródłowy został udostępniony w Elportalu. Został on napisany w języku C++ z wykorzystaniem środowisk WinAVR oraz AVR Studio. Warto krótko omówić najciekawsze fragmenty programu, aby ułatwić nanoszenie ewentualnych zmian, które pozwolą na lepsze przystosowanie urządzenia do własnych potrzeb.

W programie zdefiniowane zostały cztery klasy, odpowiedzialne za obsługę poszczególnych bloków podzespołów. Najistotniejszą klasą niewątpliwie jest klasa *humidity*, odpowiadająca za pomiar wilgotności. Ma ona trzy funkcje składowe: konstruktor, *getVoltage* oraz *getHumidity*. Konstruktor jest odpowiedzialny za wybór trybu pracy układu ADC, na co składa się wyrównanie wyniku konwersji do prawej, wymuszenie pomiaru na kanale 7 (bo tu podłączono wejście czujnika), wybór napięcia odniesienia, wybór częstotliwości taktowania przetwornika oraz jego włączenie. Działanie *getVoltage* sprowadza się do wyzwolenia konwersji osiem razy i obliczenie średniej z zebranych próbek i zwrócenia na końcu napięcia wyrażonego w mV. Obliczanie średniej zostało podyktowane chęcią zwiększenia dokładności pomiaru napięcia i zmniejszenia wpływu ewentualnych zakłóceń. Funkcja *getHumidity* na podstawie zmierzonego napięcia oblicza wilgotność. Sprowadza się to do odszukania w tabeli *humidity_conversionTable* odpowiedniego przedziału napięć, dzięki czemu wiadomo, w jakim przedziale zawiera się bieżąca wilgotność. Dysponując informacją o przedziale wilgotności oraz wartościami napięcia na jego końcach, można dokonać liniowej interpolacji z przybliżeniem do jednego procenta. Zastosowany algorytm jest w sumie bardzo prosty, gdyż sprowadza się on do obliczenia różnicy napięcia między końcem i początkiem znalezionego przedziału. Różnica ta następnie jest dzielona przez 10, co daje informację o przyroście napięcia dla każdego procenta wilgotności w zadanym przedziale. Stąd, jeżeli obliczymy, że różnica napięć między 30% a 40% wynosi 1680 – 1310 = 370mV, to na jeden procent przypadnie 37mV. Wiedząc to, można łatwo stwierdzić, że jeżeli



Rys. 2

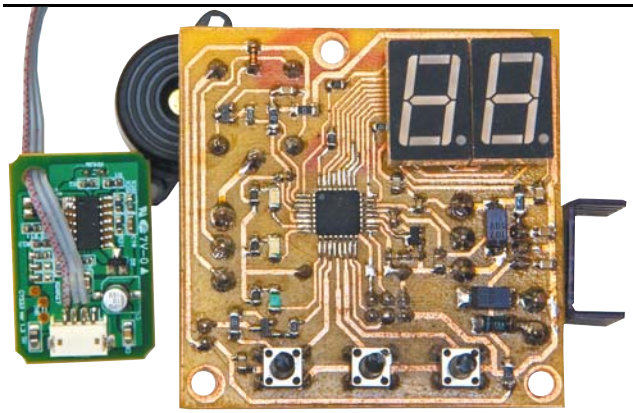
zmierzone napięcie będzie równe 1384mV, to wiadomo, że jest ono większe od dolnej granicy o 1384 – 1310=74mV. Tym samym do dolnej, granicznej wartości 30% należy dodać 74/37=2% i ostatecznie wynikiem będzie wilgotność równa 32%. Fragment kodu odpowiedzialny za wyznaczenie wilgotności pokazano na **listingu 1**.

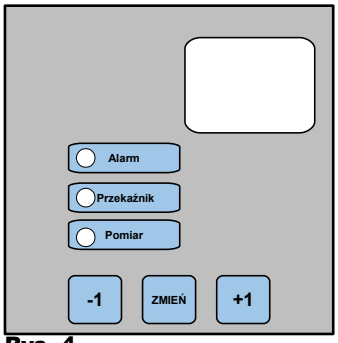
Poszczególne wartości znajdujące się w tabeli *humidity_conversionTable* pochodzą z dokumentacji producenta czujnika. Modyfikując zawartość tej tabeli, można w prosty sposób skalibrować posiadany czujnik lub dostosować urządzenie do pracy z innym przetwornikiem, np. analogowym termometrem.

Drugą klasą wartą uwagi jest klasa *device*. Jej konstruktor przyjmuje jako parametr numer portu urządzenia, którym obiekt tej klasy ma sterować. W układzie są dwa elementy obsługiwane przez tę klasę: przekaźnik oraz buzzer. Oba są dołączone do tego samego portu – PORTD oraz oba są sterowane takim samym poziomem logicznym (niski – układ wyłączony, wysoki – układ włączony). Klasa *device* posiada funkcje *save* oraz *load*, które odpowiednio, zapisują oraz odczytują zmienne do/z pamięci EEPROM i są wykorzystywane do pamiętania wartości zmiennych *value* oraz *inversion*. Dzięki temu wyłączenie zasilania nie spowoduje utraty informacji o ustawieniach wprowadzonych przez użytkownika. Zmiana bieżącej wartości jest możliwa do zrealizowania za pomocą operatorów ++ oraz – – co podnosi czytelność głównego programu. Za pomocą funkcji *getValue* można pobrać bieżącą, ustawioną wartość wilgotności. Zasadniczym elementem klasy *device* jest funkcja *doWork*, przyjmująca jako argument bieżącą wilgotność. Na tej podstawie,

Listing 1

```
unsigned char humidity::getHumidity(){
  unsigned int voltage = getVoltage() ;
  unsigned int index = 0 ;
  unsigned int delta = 0 ;
  //aproksymacja
  for(index=0 ; index<11 ; index++){
    if(voltage < humidity_conversionTable[index]){ break;}
  }
  //zabezpieczenia
  if(index==0){return 10;}
  if(index>10){return 95;}
  //aproksymacja - c.d.
  delta = (humidity_conversionTable[index] - humidity_conversionTable[index-1])/10 ;
  return index*10 + (voltage-humidity_conversionTable[index-1])/delta ;
}
```





Rys. 4

z uwzględnieniem histerezy, podejmowana jest decyzja o włączeniu bądź wyłączeniu przypisanego do obiektu klasy *device* urządzenia (przekaźnika bądź buzzera). Warto podkreślić, że rola histerezy sprowadza się do wyeliminowania oscylacji (szybkiej zmiany stanu np. przekaźnika) na skutek pracy urządzenia na progu zadziałania. W razie potrzeby histereza może zostać zmieniona za pomocą stałej *DEVICE_HYSTERESIS* znajdującej się w pliku *device.h*.

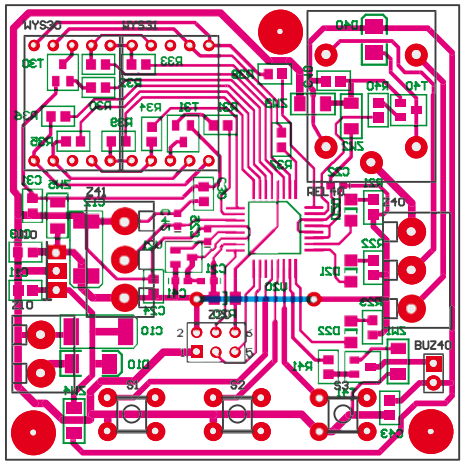
W pętli głównej znajduje się struktura *switch*, sterowana przez zmienną *menu*, która decyduje o stanie, w jakim znajduje się program, czyli jakie zadanie jest w danej chwili realizowane. Zmiana wartości zmiennej *menu* powoduje, że wykonywany jest inny fragment kodu (*case*) i tym samym urządzenie realizuje inną funkcję. Jak nietrudno się domyślić, zmiana wartości zmiennej *menu* następuje po naciśnięciu środkowego przycisku S2.

Warto zauważyć, że pod koniec pętli *while* umieszczona została linia:

if(!(menuTimeout--)){menu = 0 ;}

odpowiadająca za powrót do „głównego menu" po przekroczeniu ustalonego czasu. W ramach tej jednej linii realizowanych jest kilka rzeczy. Po pierwsze, następuje sukcesywne zmniejszanie wartości zmiennej *menuTimeout*. Gdy osiągnie ona wartość zero, do zmiennej *menu* przypisana zostanie wartość zero, co sprawi, że w ramach struktury *switch* wykonywany będzie fragment kodu odpowiedzialny za pomiar wilgotności, wyświetlenie jej wartości na ekranie i sterowanie przekaźnikiem oraz buzzerem – czyli jednym słowem powrót do „głównego ekranu". Interakcje ze strony użytkownika powoduje, że za każdym razem do zmiennej *menuTimeout* przypisywana jest nowa wartość i odliczanie czasu rozpoczyna się od początku.

Rys. 3



Montaż i uruchomienie

Urządzenie zostało zaprojektowane z wykorzystaniem głównie elementów SMD – **rysunek 3**. Z tego względu najwygodniej jest zacząć montaż właśnie od nich. Lutując mikrokontroler, warto mieć pod ręką dobrą plecionkę do odessania nadmiaru cyny z jego wyprowadzeń. Po wlutowaniu układu U20 warto obejrzeć płytkę pod światło i zobaczyć, czy między wyprowadzeniami nie ma zwarć.

Uwaga! W sytuacji, gdy urządzenie ma zostać zamontowane w obudowie Z54, w sposób pokazany na fotografii tytułowej, wyświetlacze oraz przyciski także należy wlutować OD STRONY DRUKU. Zwracam uwagę, że pod wyświetlaczami jest sporo elementów SMD, więc należy je wlutować jako pierwsze, bo potem nie będzie to możliwe. W sytuacji, gdy płytka jest wykonywana w warunkach domowych, warto sprawdzić omomierzem ścieżki, które zostaną zakryte wlutowanymi podzespołami, aby wykluczyć ich uszkodzenie i późniejsze problemy z nanoszeniem poprawek.

W obudowie należy wywiercić otwory na diody LED, przyciski oraz śruby i wyciąć okienko pod wyświetlacze. Najprostszym wyjściem będzie wydrukowanie warstwy opisowej, przyklejenie jej do spodniej części wieczka obudowy i zaznaczenie punktów cięcia/wiercenia. Miejsca przeznaczone na śruby warto nieco rozwiercić, aby łebki śrub mogły się schować i naklejka płasko przylegała do obudowy. Między PCB a wieczko obudowy włożone zostały tulejki dystansowe mające za zadanie trzymać druk w równej odległości od obudowy. Całość skręcono śrubami 3mm. Przyciski wystawały znacząco ponad obudowę, dlatego były szlifowane tak długo, aż nie były „na styk".

Można zastosować „separację optyczną" diod LED, tzn. nałożyć na nie np. koszulki termokurczliwe, aby światło danej diody było widziane tylko w przewidzianym dla niej otworze. Projekt etykiety, pokazany na **rysunku 4**, jest bardzo prosty (wykonany w Open Office), dlatego można pokusić się o jego samodzielne wykonanie w innej postaci. Po wydrukowaniu i wycięciu niezbędnych otworów na diody i wyświetlacze, etykietę warto zalaminować, aby uzyskać lepszy efekt i ochronić przed ścieraniem czy zabrudzeniem.

Interfejs użytkownika nie jest specjalnie skomplikowany. Naciśnięcie środkowego przycisku (S2) spowoduje włączenie kolejne diody LED. Gdy zaświeci się dioda:

- zielona (D22) – układ pokazuje bieżącą wilgotność
- czerwona (D21) – wyświetlacz miga i można ustawić przyciskami S1 oraz S3 pożądaną wilgotność
- żółta (D20) – wyświetlacz miga i przyciskami S1 oraz S3 ustawia się próg alarmu

Każda zmiana powoduje zapisanie nowej wartości do pamięci EEPROM, dzięki czemu po włączeniu urządzenia do sieci nastąpi przywrócenie ostatnich nastaw. Zwracam uwagę, że po pierwszym uruchomieniu należy bezwzględnie ustawić prawidłowe wartości, gdyż po zaprogramowaniu mikrokontrolera znajdą się tam wartości spoza dozwolonego zakresu.

Podczas normalnej pracy (zaświecona zielona dioda D22) przytrzymanie S1 lub S3 przez kilka sekund powoduje zmianę pracy,

odpowiednio, przekaźnika oraz buzzera. Tzn. w zależności od ustawionego trybu, dany element będzie włączany poniżej lub powyżej ustawionego progu. Tym samym higrostat może realizować funkcję osuszania bądź nawilżania. Zmiana trybu jest sygnalizowana wyświetleniem przez parę sekund jednego z czterech komunikatów:

- ri – przekaźnik pracuje w trybie odwróconym
- rn – przekaźnik pracuje w trybie normalnym
- bi – buzzer pracuje w trybie odwróconym
- bn – buzzer pracuje w trybie normalnym

Taka funkcjonalność pozwala również na skonfigurowanie sterownika w taki sposób, aby przy małej wilgotności uruchamiane było nawilżanie, a przy zbyt dużej – alarm.

Włączenie przekaźnika jest sygnalizowane migającą kropką. Jest to informacja dla użytkownika, że układ wykonawczy pracuje. Warto jeszcze dodać, że urządzenie ma stałą, 2% histerezę. Jej wartość można zmienić, jedynie modyfikując program, kompilując go i programując mikrokontroler. Nie jest to oczywiście wymagane, ale w specyficznych sytuacjach, gdy histereza musi być większa bądź mniejsza, można ją (w miarę) prosto zmienić w jednej linijce programu.

Urządzenie zostało przeznaczone do pracy z zewnętrznym zasilaczem, dającym napięcie stałe 12V. Wartość ta nie jest krytyczna i może się różnić w granicach ±3V, jednakże zbyt niska wartość sprawi, że przekaźnik nie będzie się włączał, a zbyt wysoka spowoduje nadmierne nagrzewanie się stabilizatora. Warto wykorzystać radiator do chłodzenia układu 7805.

Mikrokontroler pracuje na domyślnej konfiguracji bitów konfiguracyjnych (fuse bits), dlatego nie ma potrzeby ich zmieniać.

Wykaz elementów

R20	1k 1206	WYS30,WYS31	7seg wsp. anoda
R21-R23,R32-R39	330R 0805	T30,T31	BC857 SOT23
R30,R31,R40,R41	1k 0805	T40,T41	BC847 SOT23
ZW1-ZW5	0R-zwora 1206	U10	LM7805 TO-220
C10,C12	100u tantal_smd	U20	ATMEGA48 TQFP32
C11,C13,C24,C30,C31,C40,C43	100n 0805	U21	MCP1541 SOT23
C21,C22,C25,C41,C42	100n 0603	BUZ40	BUZZER HDR1X2
D10	1N4007 MELF	REL40	REL
D20	YELLOW 1206	S1,S2,S3	uSWITCH
D21	RED 1206	Z10	ARK2
D22	GREEN 1206	Z40,Z41	ARK3
D40	LL4148 mini MELF	Z20	ISP HDR2X3

Komplet podzespołów z płytką jest dostępny w sieci handlowej AVT jako kit szkolny AVT-3026.

Możliwości zmian

Niewątpliwie największe możliwości zmian leżą po stronie oprogramowania. Zmieniając funkcje odpowiedzialne za pomiar, można przystosować urządzenie do pomiaru wielkości innych niż wilgotność. Wymieniając czujnik wilgotności na analogowy czujnik temperatury, można w prosty sposób zrealizować termostat. Budując dwa urządzenia, jedno do pomiaru temperatury a drugie wilgotności, uzyska się bardziej kompleksowe rozwiązanie dla potrzeb np. domowego terrarium.

Innym przykładem zastosowania urządzenia może być monitorowanie jakości powietrza za pomocą czujnika dwutlenku węgla (CO_2). Po stwierdzeniu zbyt wysokiego poziomu CO_2 nastąpi uruchomienie wentylacji.

Jakub Borzdyński
jakub.borzdynski@elportal.pl