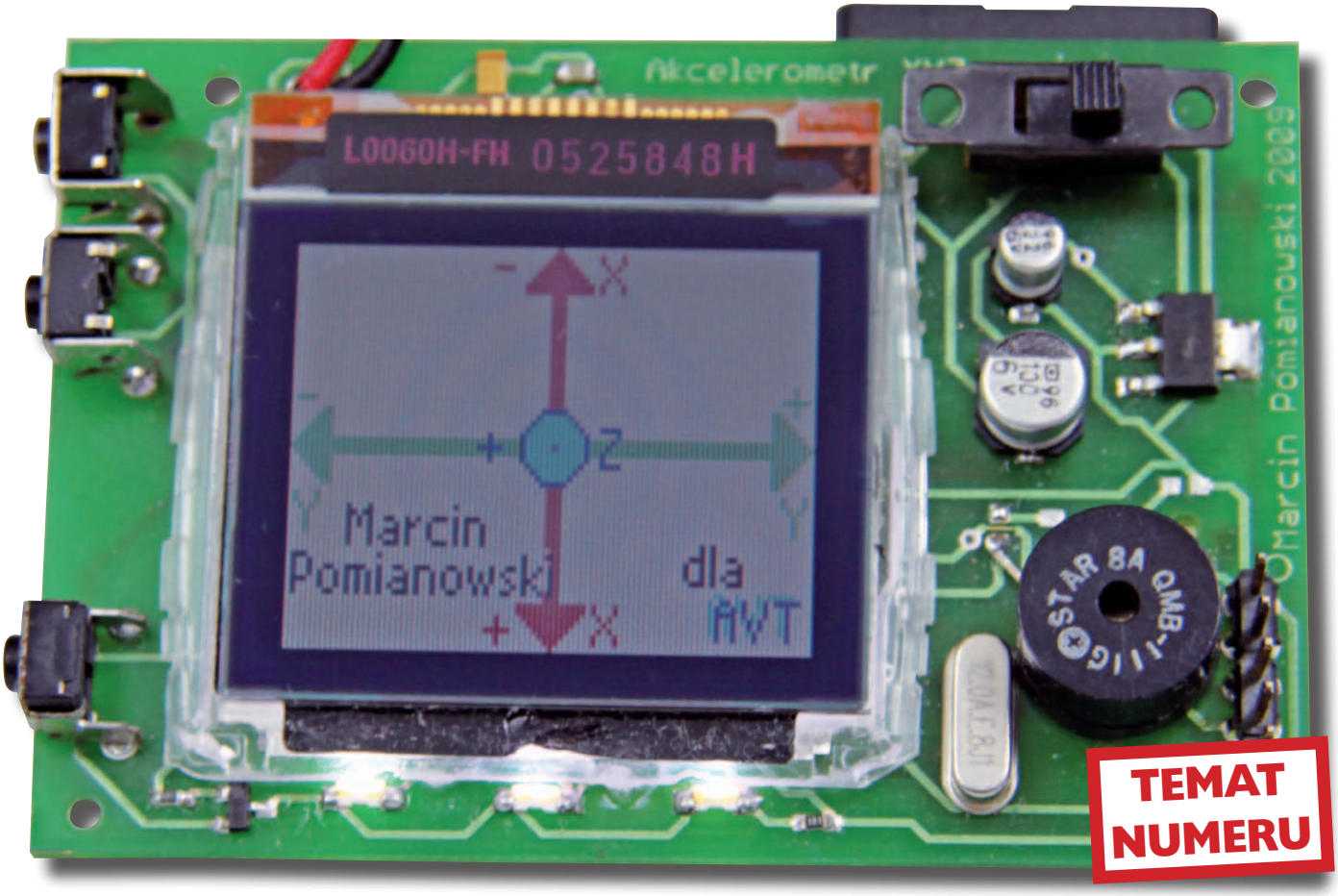


TEMAT NUMERU

Kieszonkowy akcelerometr
Miernik przyśpieszenia XYZ

AVT 5223

Przedstawiamy projekt kieszonkowego, zasilanego z baterii akcelerometru, wykonanego w oparciu o czujnik MEMS, mierzącego przyśpieszenie w kierunkach X, Y i Z. Na wbudowanym wyświetlaczu prezentowany jest wynik pomiaru przyśpieszenia we wszystkich osiach wraz ze zmierzonymi wartościami liczbowymi. Urządzenie może stanowić podstawę do samodzielnie konstruowanych układów np. umożliwiających automatom orientację w przestrzeni, lokalizację złóż i ukrytych obiektów o dużej masie, przeprowadzania doświadczeń fizycznych, budowy ciekawego interfejsu użytkownika itd.

Rekomendacje: *urządzenie ma bardzo duże walory edukacyjne, a ograniczeniem jego zastosowania jest tylko wyobraźnia.*

W dobie miniaturyzacji większość znanych nam od lat przedmiotów staje się coraz mniejsza. Niektóre z nich „schodzą" z półek, by znaleźć nowe miejsce w naszej kieszeni. Na przykład radioodbiornik (lampowy) sam w sobie był kiedyś sporą szafką. Później zmniejszył znacznie swoje rozmiary dzięki cudownym tranzystorom, by ostatecznie stać się małym pudełeczkiem. Ale cud miniaturyzacji to nie tylko możliwość korzystania z osobistego radioodbiornika w komunikacji miejskiej, to także zwiększona dostępność pewnych rozwiązań.

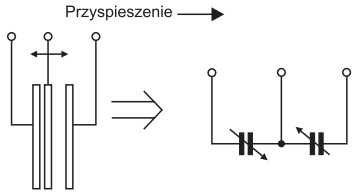
Teraz każdy może kupić za niezbyt wygórowaną kwotę mały układ scalony, który pełni rolę stosowanych do niedawna dużych, drogich i precyzyjnych akcelerometrów, opartych na tensometrach. Rozwiązania te były delikatne, a zbyt duże przeciążenie mogło spowodować uszkodzenie cennych

AVT-5223 w ofercie AVT:
AVT-5223A – płytka drukowana

Podstawowe informacje:
- płytka o wymiarach 51×75 mm
- napięcie zasilania 3,6...10 V
- zakresy pomiarowe ±2,5; ±3,3; ±6,7; ±10 g
- zmienna częstotliwość odświeżania wykresów (4 prędkości)
- zapis wyników na kartach SD
- zapis próbek z częstotliwościami (około) 675, 430, 176, 93, 57 Hz
- cztery tryby wyświetlania wartości przyspieszeń

Dodatkowe materiały na CD i FTP:
ftp://ep.com.pl, user: 12686, pass: 2b7r7b68
- wzory płytek PCB
- listingi
- karty katalogowe i noty aplikacyjne elementów oznaczonych na Wykazie Elementów kolorem czerwonym

Projekty pokrewne na CD i FTP:
(wymienione artykuły są w całości dostępne na CD)
Elektroniczny miernik przyśpieszenia (EP 8/1998)
Miernik przyśpieszenia (EP 8/2005)



Rys. 1. Uproszczony model akcelerometru dla jednej osi czułości

tensometrów. Z pomocą przyszła technologia MEMS, która pozwoliła na zbudowanie między innymi miniaturowych akcelerometrów, np. w postaci układów scalonych o wymiarach 6×6×1,45 mm. Oczywiście taka miniaturyzacja ma swoją cenę. Dokładność wskazań akcelerometru wykonanego w technologii MEMS jest znacząco niższa od pierwowzoru, jednak zupełnie wystarczająca do zastosowania jako czujnik dający podstawowe informacje o przyśpieszeniu.

Siła, przyśpieszenie, przeciążenie

Jaką informację można uzyskać z akcelerometru? Aby odpowiedzieć na to pytanie, należy przywołać budowę i zasadę działania takiego czujnika. Na **rys. 1** pokazano uproszczony model akcelerometru (dla jednej osi czułości), zaczerpnięty z dokumentacji producenta. W uproszczeniu układ detekcyjny jest zbudowany z kondensatora, w którym umieszczono dodatkową, ruchomą elektrodę. Z nią powiązana jest masa bezwładna, która przemieszcza się pod wpływem działania sił zewnętrznych. Każde przemieszczenie ma swoje odzwierciedlenie w zmianie pojemności kondensatorów składowych.

Zakładając, że ruch masy bezwładnej jest możliwy w osi zgodnej z osią pionową (prostopadłą do powierzchni ziemi), na tę masę będzie działała siła grawitacji ziemskiej. Siła ta jest odpowiedzialna za występowanie przyspieszenia ziemskiego, zatem tak umieszczony czujnik wykrywa przyspieszenie ziemskie, wywołane siłą grawitacji.

Przyspieszenie wykrywane przez akcelerometr jest mierzone w jednostkach „g", oznaczających krotność grawitacji ziemskiej. Zatem 1 g oznacza przyspieszenie ziemskie, czyli w przybliżeniu 9,81 m/s². Jednostka taka jest bardzo wygodna ze względu na intuicyjne rozumienie jej wartości. Jak wiemy z lekcji fizyki, to przyciąganie ziemskie sprawia, że wszystkie ciała spadają. Przyspieszenie tego spadku jest równe przyspieszeniu ziemskiemu (w próżni). Jednak rola grawitacji nie kończy się na ściąganiu skoczków z powrotem na ziemię. Kiedy stoimy na ziemi grawitacja, działa dalej, a efekt jej działania możemy zaobserwować, np. stając na wadze.

Jaki zatem będzie efekt, jeśli znajdziemy się na pokładzie pionowo startującej rakiety i przy starcie zarejestrujemy przyspieszenie o wartości 2 g? Zdecydowanie poczujemy, że nasze ciało jest wciskane w podłogę rakiety. Zjawisko takie nazywa się przeciążeniem. Przeciążenie to stan, w którym znajduje się ciało w wyniku działania sił innych niż siła grawitacji. Przy takim przeciążeniu na nasze ciało działa podwojona siła grawitacji, a więc „ważymy" wtedy dwukrotnie więcej. To właśnie ta dodatkowa „waga" naszego ciała powoduje uczucie „wciskania" w podłogę.

Przy analizie przyspieszeń należy mieć na uwadze fakt, iż przyspieszenie ziemskie zależy od szerokości geograficznej oraz wysokości nad poziomem morza, a dodatkowo na jego wartość mogą mieć wpływ rozmaite czynniki. Wraz ze wzrostem wysokości przyspieszenie maleje, co jest spowodowane zmniejszaniem się siły grawitacji. Zmniejszanie szerokości geograficznej również powoduje spadek przyspieszenia, co wynika z działania pozornej siły odśrodkowej, która powstaje na skutek ruchu obrotowego Ziemi.

Duże znaczenie ma też to, co znajduje się pod i nad nami. Wszelkiego typu złoża i inne czynniki znacznie zmieniające gęstość skorupy ziemskiej mają wpływ na wartość przyspieszenia ziemskiego. Ważnym czynnikiem jest również ruch ciał niebieskich, a w szczególności Księżyca i Słońca. Wpływ ich grawitacji ma bardzo duże odzwierciedlenie na Ziemi. Na przykład grawitacja Księżyca powoduje zjawiska przypływów i odpływów.

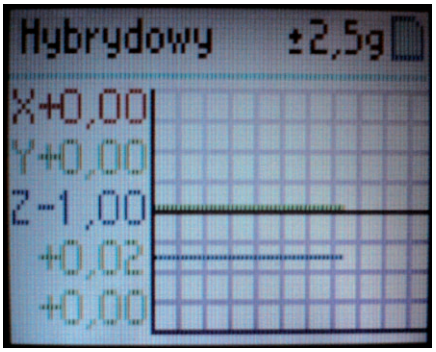
Mając na uwadze wymienione czynniki, wyposażyłem urządzenie w możliwość kalibracji wskazań. Kalibracja może odbywać się w jednym z dwóch trybów. Tryb pierwszy zakłada automatyczne ustalenie osi, która jest osią pokrywającą się w największym stopniu z kierunkiem działania siły grawitacji, i ustalenie poprawki sprowadzającej zarejestrowaną wartość do wskazania 1 g. Pozostałe dwie osie są sprowadzane do wartości zero. Drugi tryb kalibracji przewiduje, że niepotrzebna nam jest informacja o przyciąganiu ziemskim, interesują nas tylko zmiany przyspieszenia na każdej z osi. W takim przypadku wszystkie osie są zerowane. Kalibracja może być również wyłączona.

Możliwości urządzenia

Zbudowane urządzenie ma następujące funkcje prezentacji danych:

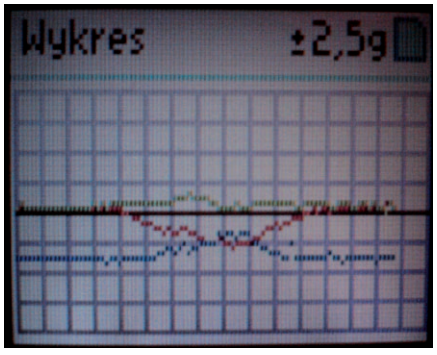
- ekran hybrydowy (**fot. 2**) zawierający małe pole wykresu obrazujące przebieg zmian przyspieszeń na wszystkich trzech osiach, liczbowe wartości przyspieszeń na każdej osi (z dokładnością do setnych) oraz globalne przyspieszenie maksymalne (na plus) i minimalne (na minus);
- ekran wyświetlający chwilowe wartości przyśpieszeń w postaci liczbowej (**fot. 3**) we wszystkich trzech osiach (środkowa kolumna) wraz z wartościami maksymalnymi (prawa kolumna) i minimalnymi (lewa kolumna) na każdej z osi;
- ekran wykresu (**fot. 4**) z możliwością dowolnego wyboru aktywnych osi;
- ekran z kołową, graficzną reprezentacją kierunku działania i wartości przeciążenia (**fot. 5**), podobny do ekranów znanych z telewizyjnych relacji wyścigów Formuły 1.

Wartości maksymalne można zresetować przez ponowne wejście w dany typ wyświetlania. Ponieważ menu zapamiętuje ostatnią pozycję, oznacza to dwukrotne wciśnięcie klawisza wyboru.

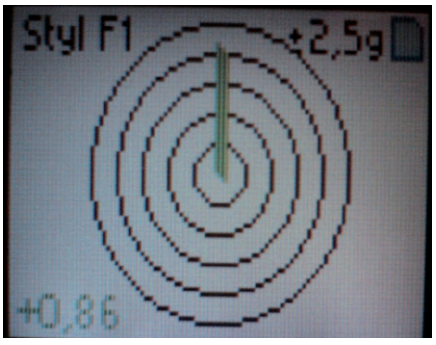


Fot. 2. Ekran hybrydowy z polem wykresu

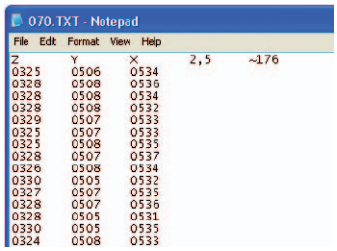
Fot. 3. Ekran z chwilowymi wartościami przyśpieszeń



Fot. 4. Ekran wykresu z możliwością wyboru osi



Fot. 5. Ekran z wykresem kołowym



Rys. 6. Zrzut ekranu z zawartością pliku

Dodatkowo dane mogą zostać zapisane na karcie SD i później zobrazowane na komputerze za pomocą arkusza kalkulacyjnego.

Konfiguracji podlegają:
- częstotliwość zapisu próbek na karcie SD;
- częstotliwość odświeżania wykresu na ekranie LCD;
- zakres pomiarowy akcelerometru;
- czułość przy wyświetlaniu danych na ekranie kołowym;
- sposób kalibracji wskazań;
- kontrast wyświetlacza;
- jasność wyświetlacza;
- włączenie/wyłączenie dźwięków.

Urządzenie ma również możliwość zapisu wprowadzonych ustawień w wewnętrznej pamięci EEPROM mikroprocesora. Zapisane ustawienia są automatycznie ładowane przy uruchomieniu urządzenia.

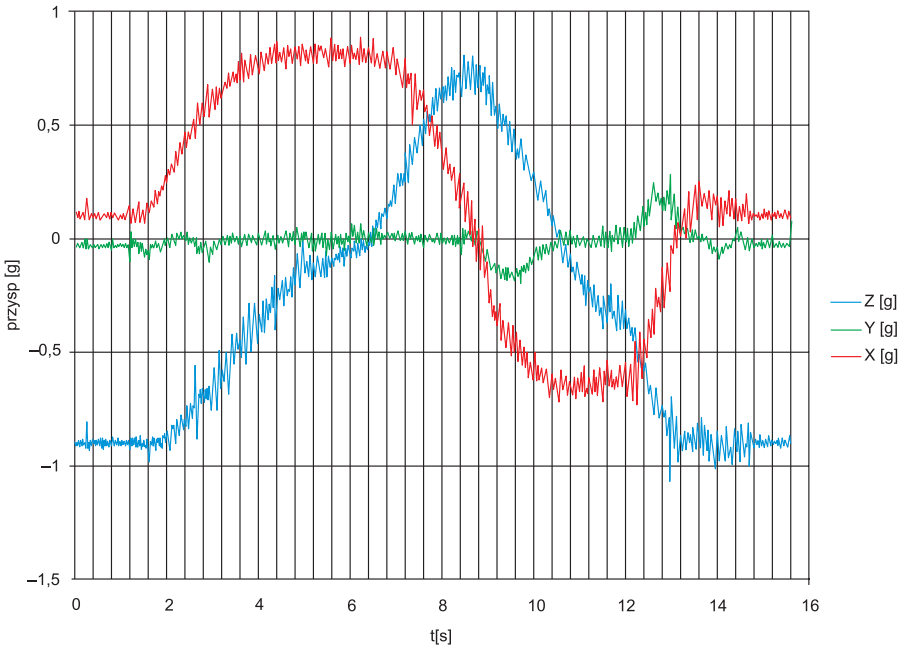
Zapisane na karcie SD wyniki mają postać plików tekstowych, w których w trzech kolumnach odpowiadających za osie X, Y i Z są zapisane wartości kolejnych próbek z przetwornika analogowo-cyfrowego. Dodatkowo zapisany jest zakres pomiarowy, na którym był przeprowadzony zapis oraz orientacyjna częstotliwość próbkowania. Przykładowa zawartość pliku jest widoczna na zrzucie ekranu (**rys. 6**). Zapisany zestaw danych pozwala na łatwe obliczenie wartości przyspieszeń oraz czasu ich wystąpienia. Plik tekstowy jest sformatowany tak, aby można go z łatwością zaimportować do arkusza kalkulacyjnego, następnie przeliczyć wyniki na wartości wyrażone w *g* oraz narysować wykres. Wartości przyspieszeń (z uwzględnieniem znaku) można wyliczyć ze wzoru:

$$a = zakres \cdot 2 \cdot \frac{wynik\_z\_ADC - 511}{1023}[g]$$

Aby wyznaczyć czas, najlepiej dodać w arkuszu dodatkową kolumnę z numerem próbki i dzielić go przez częstotliwość próbkowania:

$$t = \frac{numer\_próbki}{f_{próbk}}[s]$$

Najlepiej zrobić sobie szablon z wpisanymi formułami przeliczającymi próbki na wartości przyspieszeń i odpowiednio wyskalowanym wykresem. Przykładowy wykres otrzymany przy użyciu szablonu widać na zrzucie ekranu na **rys. 7**, a jego odpowiednik na ekranie LCD urządzenia na **fot. 8**. Został on zarejestrowany przy wykonaniu pełnego



Rys. 7. Wykres otrzymany przy użyciu szablonu MS Excel

obrotu wokół osi Y (ułożenie osi widać na zdjęciu tytułowym). W dziewiątej sekundzie widać wyraźną zmianę przyspieszenia na osi Y. Powstało ono przez delikatne odchylenie urządzenia, przy przekładaniu go z ręki do ręki. Łatwo zaobserwować pewną niezgodność poziomów przebiegu na wyświetlaczu i komputerze. Wynikają one z faktu, iż próbki są zapisywane na karcie SD bez uwzględniania kalibracji. Funkcję taką można oczywiście bardzo łatwo dodać, jednak uznałem, że „surowe" dane z przetwornika pozwalają na uzyskanie rzeczywistej informacji.

Użytkownik ma do dyspozycji 3 przyciski: S1 służy do wywołania menu lub do zatwierdzenia wyboru opcji; S2 służy do poruszania się po menu (przewijanie w górę), a podczas wyświetlania wyników służy jako pauza; S3 służy do poruszania się po menu (przewijanie w dół), a podczas wyświetlania wyników uruchamia lub stopuje zapis na karcie SD. Menu główne pokazano na **fot. 9**.

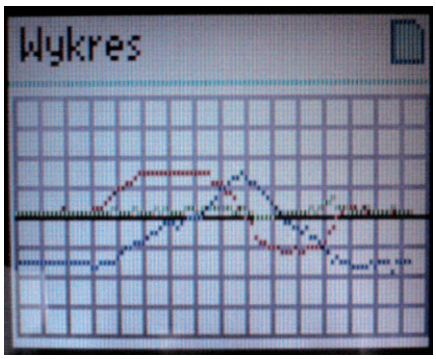
Opis urządzenia

Schemat ideowy miernika przyśpieszenia umieszczono na **rys. 10**. Pracą wszystkich komponentów zastosowanych w urządzeniu zarządza mikrokontroler firmy Atmel – ATmega32A (IC1). Jest to wersja zasilana obniżonym napięciem, którą mimo to można taktować dość dużą częstotliwością zegarową. Kondensatory C15...C17 filtrują napięcie zasilania części cyfrowej i są umieszczone na płytce blisko nóżek zasilających mikrokontroler. Napięcie zasilające część analogową jest filtrowane przez dławik L1 oraz kondensator C13. Ponieważ wykorzystywany w programie przetwornik A/C pracuje w pełnym zakresie napięcia zasilania, jego wejście napięcia odniesienia należy połączyć z masą przez kondensator. Zalecenie takie znajduje się w dokumentacji producenta; w opisywanym urządzeniu realizuje je kondensator C14.

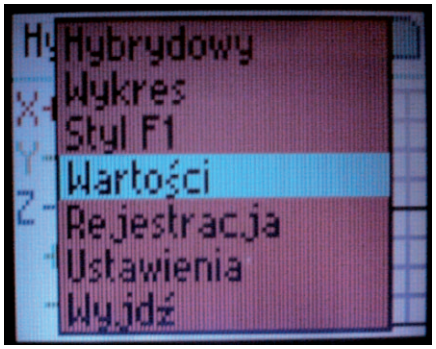
Stabilne rozpoczęcie pracy po włączeniu zasilania zapewnia układ generowania impulsu zerowania, zbudowany z kondensatora C1 i rezystora R1. Mikroprocesor jest taktowany sygnałem o częstotliwości 12 MHz, generowanym przy użyciu zewnętrznego rezonatora kwarcowego (Q1) z kondensatorami C5 i C5.

Układ MMA7261QT

Zastosowany akcelerometr (IC3) jest trzyosiowym czujnikiem przyspieszenia produkowanym przez firmę Freescale. Ma trzy wejścia sterujące: *g_sel1*, *g_sel2* – słu-

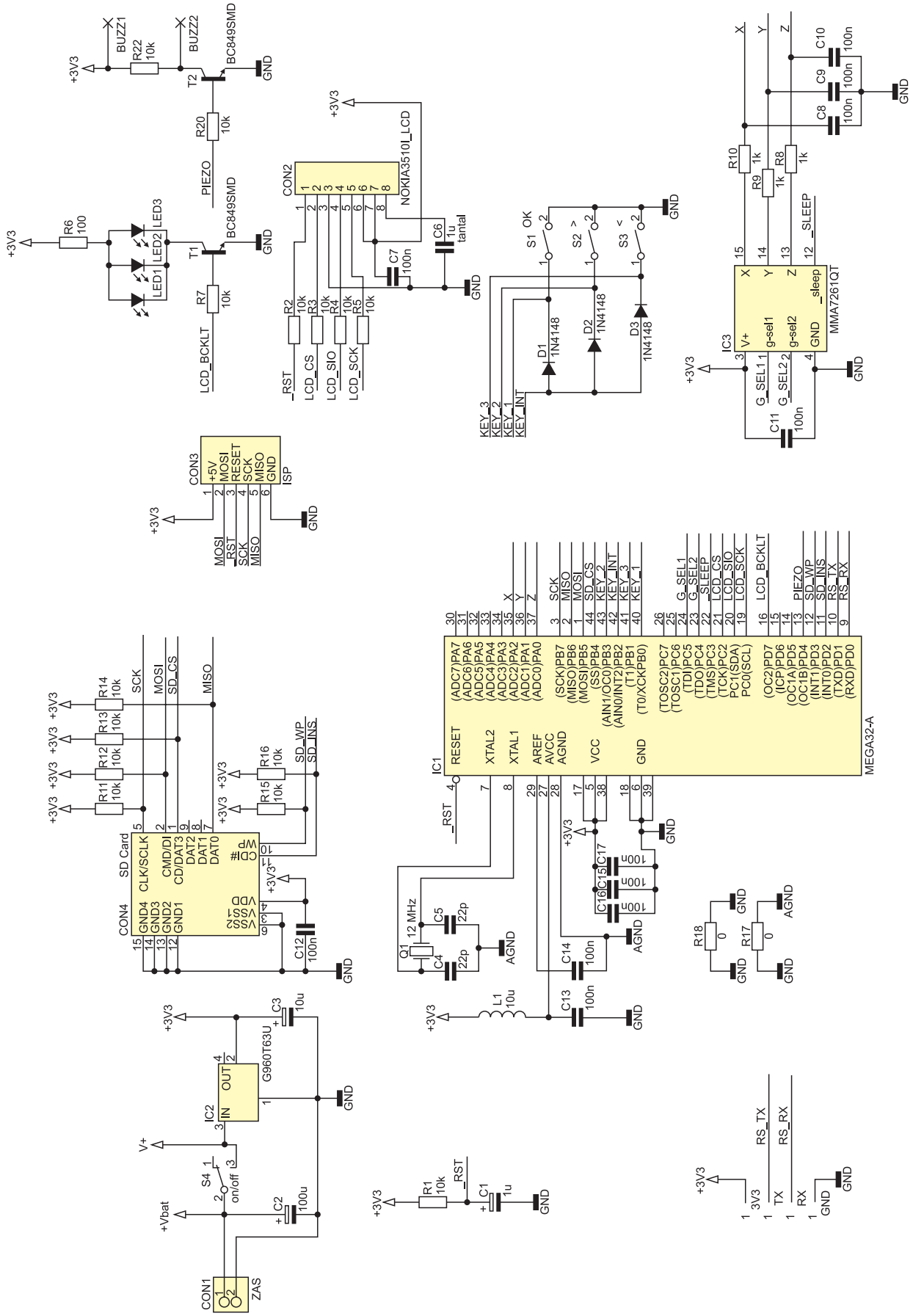


Rys. 8. Wykres na ekranie urządzenia



Fot. 9. Menu główne programu

żące do wyboru zakresu pomiarowego oraz *sleep* – służące do wprowadzenia układu w stan uśpienia. Wszystkie trzy sygnały są podłączone do portów mikroprocesora. Sygnały z wyjść analogowych akcelerometru są filtrowane przez filtry dolnoprzepustowe zbudowane z kondensatorów C8...C10 oraz rezystorów R8...R10. Filtr taki jest zalecany przez producenta i zaczerpnięty z dokumentacji układu. Dodatkowo napięcie zasilające akcelerometr jest filtrowane przez kondensator C11.



Rys. 10. Schemat ideowy akcelerometru

Wykorzystany układ wykrywa znak przyspieszenia, czyli przyśpieszenie w każdej osi może być (umownie) dodatnie lub ujemne. Wraz ze zmianami przyspieszenia wykrywanymi przez czujnik na jego wyjściach pojawia się sygnał analogowy (dla każdej osi oddzielne wyjście), proporcjonalny do wartości przyspieszenia. Sygnał ten przyjmuje wartości z zakresu od 0 V do napięcia zasilania czujnika, przy czym w środku tego zakresu znajduje się wartość napięcia odpowiadająca wskazaniu „0 g".

Wykorzystany czujnik zawiera cztery zakresy czułości: ±2,5 g, ±3,3 g, ±6,7 g, ±10 g. Dokładność wskazań czujnika zależy od wybranego zakresu i jest największa dla najmniejszego zakresu pomiarowego. Zakres pomiarowy może być zmieniany w dowolnej chwili pracy czujnika.

Karty SD

Do rejestracji przebiegów czasowych przyspieszeń zastosowałem bardzo popularne karty SD (ang. Secure Digital), które są umieszczane w dedykowanym złączu (CON4). Karta SD może pracować w jednym z dwóch trybów: SDBus lub SPI. Tryb SPI oferuje wolniejszą transmisję, natomiast zdaje się być stworzony do umożliwienia współpracy karty z popularnymi mikroprocesorami. Ograniczenie prędkości transmisji pomiędzy kartą a mikroprocesorem nie ma w przypadku opisywanego urządzenia żadnego znaczenia, ponieważ prędkości oferowane w trybie SPI są więcej niż wystarczające. Ze złączem kart pamięci powiązane są rezystory podciągające linie magistrali SPI, które na ogół mogą zostać pominięte przy montażu. Zostały one umieszczone na płytce na wszelki wypadek i należy je montować, wyłącznie jeśli wystąpią problemy ze współpracą urządzenia z jakąś kartą pamięci. Osobiście przetestowałem pracę urządzenia przy współpracy z dwiema kartami. Pierwsza to karta MicroSD (umieszczona w przejściówce zmieniającej ją w kartę normalnych rozmiarów) firmy SanDisk o pojemności 128 MB. Druga karta była standardowej wielkości, miała pojemność 2 GB i została wyprodukowana przez firmę Kingston. Zastosowanie kart różnych producentów charakteryzujących się znacząco różnymi pojemnościami pozwala mi twierdzić, że zapewniona jest kompatybilność z większością dostępnych na rynku kart. Jednak ze względu na występowanie pewnych różnic w pojawiających się nowych odmianach kart, może zdarzyć się, że jakaś karta nie będzie chciała współpracować z urządzeniem.

Złącze kart udostępnia dodatkowo dwa sygnały związane z fizycznym stanem karty – sygnał włożenia karty oraz sygnał blokady zapisu. Wyprowadzenia tych sygnałów muszą być podciągnięte do plusa zasilania, co jest zrealizowane przez rezystory R15 i R16. Sygnał z wyjścia sygnalizującego obecność karty w złączu jest wykorzystywany do generacji przerwania (wyzwalanego każdą zmianą poziomu logicznego – reaguje na oba zbocza), które odpowiada za wykrycie włożenia i wyjęcia karty oraz przeprowadzenie jej poprawnej inicjalizacji. Stan karty jest dodatkowo sygnalizowany kolorową ikonką na wyświetlaczu LCD. Ikonka w kolorze szarym oznacza brak karty SD lub jej błąd. Ikonka niebieska pojawia się po poprawnym zainicjalizowaniu karty, natomiast czerwona oznacza trwający zapis. Kondensator C12 filtruje napięcie zasilające kartę. Dodatkowo poprawnej inicjalizacji karty towarzyszy krótki sygnał dźwiękowy. Przy wystąpieniu błędu inicjalizacji bądź wyjęciu karty urządzenie odtwarza dłuższy sygnał o niższej częstotliwości.

Wyświetlacz LCD

Pierwszą częścią interfejsu komunikacji z użytkownikiem jest kolorowy wyświetlacz graficzny od telefonu komórkowego Nokia 3510i. Wyświetlacz ma rozdzielczość 96×65 pikseli i możliwość wyświetlania 4096 kolorów. Wyprowadzenia wyświetlacza należy dolutować do specjalnie zaprojektowanych pól lutowniczych (CON2). W przeciwieństwie do wielu innych przypadków oznaczania telefonów, w tym modelu bardzo ważne jest występowanie litery „i" na końcu. Telefony bez tej litery, oznaczane identycznym numerem, miały wyświetlacze monochromatyczne.

Wyświetlacz komunikuje się z mikroprocesorem przez synchroniczny interfejs szeregowy. Składa się on z linii zegarowej, linii danych oraz dwóch linii sterujących stanem wyświetlacza. Są to: linia *Reset*, sprzężona z układem zerującym mikroprocesor przy włączeniu zasilania oraz linia wyboru wyświetlacza. Do poprawnej pracy wbudowanej w wewnętrzny sterownik wyświetlacza przetwornicy niezbędny jest zewnętrzny kondensator tantalowy C6. Kondensator C7 filtruje napięcie zasilania wyświetlacza.

Drugą częścią interfejsu użytkownika jest prosta klawiatura. Przyciski S1...S3 są podłączone bezpośrednio do portów mikrokontrolera. Aby usprawnić pracę programu sterującego i wyeliminować okresowe sprawdzanie stanu trzech linii wejściowych, wyprowadziłem dodatkowy sygnał przerwania od klawiatury. Aby nie dodawać kolejnego układu scalonego, na diodach D1...D3 zbudowałem bramkę iloczynu logicznego. Kiedy którykolwiek z przycisków zostanie wciśnięty (np. S1), na powiązanym z nim wejściem mikroprocesora zostanie wymuszony logiczny stan niski. Dioda powiązana z tym przyciskiem (D1) zacznie przewodzić, wymuszając stan niski na wejściu przerwania klawiatury. Pozostałe diody (D2 i D3) pozostaną spolaryzowane zaporowo, dzięki czemu wciśnięcie jednego przycisku, nie powoduje zmiany stanu na wszystkich wejściach obsługujących klawiaturę. Rolę rezystorów podciągających w tak zbudowanej bramce pełnią wewnętrzne rezystory podciągające mikrokontrolera. Zastosowanie prostej bramki DTL sprzyja miniaturyzacji urządzenia oraz ułatwia i upraszcza prowadzenie połączeń na płytce drukowanej (nie trzeba prowadzić zasiania do układu scalonego, uniezależnia prowadzenie ścieżek od rozkładu jego wyprowadzeń).

Układ IC2 to stabilizator napięcia 3,3 V. Kondensatory elektrolityczne C2 i C3 filtrują napięcie wejściowe i wyjściowe. Przełącznik S4 służy do włączania zasilania urządzenia, zaś złącze CON1 do doprowadzenia zasilania. Dla zastosowanego stabilizatora maksymalne napięcie wejściowe nie powinno przekraczać 10 V, a najlepsze warunki pracy stabilizatora uzyskuje się przy zasilaniu równym 5 V.

Rezystor R18 o wartości 0 Ω jest zworą łączącą masę. Rezystor R17 stanowi jedyne połączenie oddzielonych mas, cyfrowej (GND) i analogowej (AGND). Zabieg taki został zastosowany z myślą o minimalizacji zakłóceń na wejściu przetwornika analogowego.

Klucz tranzystorowy zbudowany z tranzystora T1 oraz rezystorów R6 i R7 steruje białymi diodami LED (LED1...LED3) podświetlającymi wyświetlacz LCD. Klucz tranzystorowy zbudowany z tranzystora T2 oraz rezystorów R20 i R22 steruje membraną piezoelektryczną.

Złącze CON3 służy do programowania mikroprocesora w trybie ISP.

Wykaz elementów

Rezystory: (SMD 0603):
R1...R5, R20, R22: 10 kΩ
R6: 100 Ω
R7: 10 kΩ
R8...R10: 1 kΩ
R11...R16: 10 kΩ
R17, R18: 0 Ω (zwora)
Kondensatory:
C1: 1 µF (A/3216-18R)
C2: 100 µF (153CLV-0505)
C3: 10 µF (153CLV-0405)
C4, C5: 22 pF (0603)
C6: 1 µF tantalowy (0805)
C7...C17: 100 nF (0603)
Półprzewodniki:
D1...D3: 1N4148 (MINIMELF)
IC1: ATmega32A (TQFP44)
IC2: G960T63U (SOT223)
IC3: MMA7261QT (QFN16)
LED1...LED3: dioda LED SMD (1206)
T1, T2: BC849SMD (SOT23)
Inne:
CON1: 1×2 (zasilanie)
CON3: goldpin 2×16
L1: dławik 10 µH
Q1: kwarc 12 MHz (HC49/S)
S1...S4: przyciski (np. produkcji OMRON)

Na CD: karty katalogowe i noty aplikacyjne elementów oznaczonych na wykazie elementów kolorem czerwonym

List. 1. Funkcja odczytująca kolejne wartości próbek

```
ISR(ADC_vect)
{
    cli();                                      //wyłaczenie systemu pzrerwań
    static uint8_t conv_count;
    uint16_t wynik = ADCW;
    ADCSRA &= ~_BV(ADEN);                       //wyłączenie ADC
    if(ADMUX == 0x40)                           //gdy kanał 0
    {
      ADC_Z = wynik;                            //zapisz wynik do zmiennej
      conv_count++;                             //zwiększ licznik próbek
    }
    else if(ADMUX == 0x41) ADC_Y = wynik;       //kanał 1
    else if(ADMUX == 0x42) ADC_X = wynik;       //kanał 2

    if(rec && (conv_count == delay_SD))         //gdy rejestracja i ustalona
    {                                           //częstotliwość zapisu na SD
      writeFile((wynik/1000)+48);               //zapisz cyfrę oznaczającą tysiące
      writeFile((wynik/100)-(wynik/1000)*10+48); //zapisz cyfrę oznaczającą setki
      writeFile((wynik/10)-(wynik/100)*10+48);  //zapisz cyfrę oznaczającą dziesiątki
      writeFile(wynik-(wynik/10)*10+48);        //zapisz cyfrę oznaczającą jedności
      if(ADMUX==0x42)                           //jeśli zapisano wynik dla ostatniej
      {                                         //osi
         writeFile(0x0D); //CR                    utwórz przejście do nowej linii
         writeFile(0x0A); //LF
         conv_count = 0;                        //zeruj licznik próbek
      }
      else writeFile(,\t'); //TAB                // gdy oś 1 lub 2 zapisz tabulator
    }

    ADMUX++;                                    //następny kanał
    if (ADMUX == 0x43) ADMUX = 0x40;            //zapetlenie zmian kanału
    ADCSRA |= _BV(ADEN)|_BV(ADSC);              //włacz ADC

    sei();                                      //włacz system przerwań
}
```

Program

Program napisano w języku C, przy użyciu kompilatora AVR GCC. Zużywa on 80% pamięci programu mikroprocesora oraz 51% pamięci danych. W pisaniu pewnych fragmentów programu bardzo pomocne okazały się zasoby Internetu. Początkowo do testów wyświetlacza wykorzystałem bibliotekę napisaną przez Roberta Obermayera z Niemiec, dostępną pod adresem: *http://hobbyelektronik.org/Elo/AVR/3510i/*. Jej uruchomienie wymagało przetłumaczenia niemieckich komentarzy i poświęcenia sporo czasu na zrozumienie działania funkcji.

List. 2. Obsługa plików w pamięci EEPROM

```
rec = 1;                                    //ustawienie flagi nagrywania
uint8_t f_count = eeprom_read_byte((uint8_t*)0x0001);
                                            //odczyt numeru pliku z wewn. EEPROM
fileName[0] = (f_count/100)+48;             //wpisanie do nazwy nowego pliku
kolejnego numeru
fileName[1] = (f_count/10)-(f_count/100)*10+48;
fileName[2] = f_count-(f_count/10)*10+48;
createFile(fileName);                       //utworzenie pliku
flushBuffer();                              //opróżnienie bufora zapisu
writeFile(,Z');                             //tworzenie nagłówka tabeli - nazwy
osi
writeFile(,\t'); //TAB
writeFile(,Y');
writeFile(,\t');
writeFile(,X');
writeFile(,\t');
writeFile((zakres/10)+48);                  //ustawiony zakres
writeFile(,,');
writeFile(zakres-(zakres/10)*10+48);
writeFile(,\t');
writeFile(,~');                             //szacowana częstotliwość zapisu
if(delay_SD == 1)
{
    writeFile(,6');
    writeFile(,7');
    writeFile(,5');
}
else if(delay_SD == 2)
{
    writeFile(,3');
    writeFile(,3');
    writeFile(,0');
}
else if(delay_SD == 6)
{
    writeFile(,1');
    writeFile(,7');
    writeFile(,6');
}
else if(delay_SD == 12)
{
    writeFile(,9');
    writeFile(,3');
}
else if(delay_SD == 20)
{
    writeFile(,5');
    writeFile(,7');
}
writeFile(0x0D); //CR                       //nowa linia
writeFile(0x0A); //LF
```

Ostatecznie, wysiłek opłacił się i zastosowałem ją w projekcie, bez tracenia czasu na tworzenie i testowanie własnej. Niemniej jednak konieczne było wprowadzenie pewnych zmian. Poza oczywistą zmianą mikroprocesora docelowego, jego częstotliwości taktowania oraz konfiguracji portów, znalazłem jeden, drobny błąd, który uniemożliwiał przeniesienie kodu na inny mikroprocesor. Dodałem też polskie znaki diakrytyczne do tablicy znaków, a usunąłem te, które uznałem za nieprzydatne – celem oszczędzenia pamięci programu, w której jest zapisywana tablica.

Kolejnym ułatwieniem osiągniętym dzięki możliwościom Internetu było uzyskanie biblioteki do obsługi kart SD ze wsparciem systemu plików FAT32. Sprawdziłem bardzo dużo bibliotek, łącznie z najpopularniejszymi. Jednak napotkałem ogromne trudności w ich implementacji i uruchomieniu. Ostatecznie najlepsza okazała się dość prosta, lecz w pełni sprawna biblioteka napisana przez CC Dharmaniego. Biblioteka ta wraz z krótkim opisem i dyskusją na jej temat jest dostępna pod adresem: *http://www.dharmanitech.com/2009/01/sd-card-interfacing-with-atmega8-fat32.html*. W tej bibliotece znalazłem błąd, który powodował złą obsługę niektórych kart SD. Swoje wnioski przekazałem autorowi, który umieścił zaproponowaną przeze mnie poprawkę w kolejnej aktualizacji biblioteki. Aby wykorzystać tę bibliotekę w projekcie, musiałem dokonać w niej dużych zmian. W oryginale operacja zapisu pliku odbywała się przez wywołanie procedury tworzenia pliku, która pobierała dane z portu szeregowego, zapisywała wszystko do pliku i zamykała plik. W moim przypadku należało postąpić bardziej tradycyjnie – przy rozpoczęciu zapisu stworzyć nowy plik i rozpocząć jego zapis. Plik musi pozostać otwarty, by umożliwić zbieranie kolejnych próbek w normalnym cyklu pracy urządzenia. Po każdorazowym zapełnieniu bufora zapisu kolejny fragment

List. 3. Funkcja zapisu do pliku

```
void writeFile (unsigned char data)
{
   buffer[i++] = data;                                  //zapisz nowy znak w buforze i zwiększ licznik znaków
   size++;                                              //zwiększ licznik rozmiaru pliku
     if(i == 512)                                       //jeśli bufor pełny
     {
       i=0;                                             //zeruj licznik bufora
       SD_writeSingleBlock (startBlock);                //zapisz bufor
      flushBuffer();                                    //wyczyść bufor
       j++;                                             //zwiększ licznik sektorów przypadajacych na klaster
       if(j != sectorPerCluster) startBlock++;          //jeśli są wolne sektory w klastrze, wybierz kolejny sektor
     }

   if(j == sectorPerCluster)                            //jeśli zapełnione wszystkie sektory w klastrze
   {
      j = 0;                                            //zeruj licznik sektorów w klastrze
      prevCluster = cluster;                            //zapamiętaj obecny klaster

      cluster = searchNextFreeCluster(prevCluster);     //wyszukaj następny wolny klaster, zaczynajac od obecnego

      if(cluster == 0)                                  //jeśli wszystkie klastry pełne - koniec procedury
      {
         //no free cluster!
        return;
      }

      getSetNextCluster(prevCluster, SET, cluster);     //powiąż nowy klaster z poprzednim

      getSetNextCluster (cluster, SET, EOF);            //oznacz klaster znacznikiem końca pliku
      startBlock = getFirstSector (cluster);            //ustaw pierwszy sektor w nowym klastrze jako aktywny
   }
}
```

pliku zostaje zapisany, aż do momentu, gdy użytkownik zakończy rejestrację – wtedy plik zostaje zamknięty. Tak więc konieczne było rozbicie oryginalnej procedury zapisu pliku na trzy procedury składowe: tworzenie (otwieranie) pliku, zapis do pliku, zamknięcie pliku. Ta z pozoru prosta operacja okazała się zajęciem na wiele wieczorów. Ostatecznie jednak udało mi się uzyskać zamierzony efekt, a powstały kod działa stabilnie.

Zbieranie wyników z akcelerometru polega na cyklicznym odczytywaniu wyniku konwersji sygnału z przetwornika analogowo-cyfrowego. Ponieważ mikroprocesor jest wyposażony tylko w jeden taki przetwornik, odczyt wartości dla różnych osi musi być multipleksowany. Oznacza to, że w rzeczywistości próbki nie pochodzą z tej samej chwili. Każda z próbek jest przesunięta względem poprzedniej o czas konwersji, który wynosi około 1,9 ms, co przy ogólnej dokładności akcelerometrów MEMS, występującym w urządzeniu szumie i edukacyjno-doświadczalnym charakterze urządzenia jest błędem do zaakceptowania. Odczyt kolejnych wartości próbek odbywa się cyklicznie w podprogramie obsługi przerwania zgłaszanego po zakończeniu konwersji próbki (**list. 1**). Wartości próbek dla poszczególnych osi są przechowywane w zmiennych globalnych ADC_*. Zmiana osi odbywa się przez inkrementację rejestru sterującego wewnętrznym multiplekserem analogowym mikroprocesora. Zmienna statyczna *conv_count* jest inkrementowana po zbadaniu wszystkich trzech osi jeśli jest ona równa globalnej zmiennej *delay_SD* oznaczającej częstotliwość zapisu próbek na karcie SD, to wartości próbek dla wszystkich osi w tej inkrementacji zostaje przesłana do zapisu. W tym miejscu następuje też formatowanie wyników w kolumny i wiersze, aby umożliwić ich łatwy import do arkusza kalkulacyjnego.

Proces zapisu rozpoczyna się od stworzenia nowego pliku nazwanego kolejnym numerem zapisu, przechowywanym w pamięci EEPROM mikroprocesora. Następnie tworzony jest nagłówek tabeli z nazwami osi oraz wartościami zakresu i częstotliwości próbkowania. Odpowiedzialny za to fragment programu przedstawia **list. 2**.

Zmodyfikowaną przeze mnie procedurę zapisu do pliku pokazano na **list. 3**. Do czasu zapełnienia bufora realizuje ona wyłącznie dodawanie do niego kolejnych znaków. Po zapełnieniu bufora (o rozmiarze 512 bajtów) następuje zapis całej jego zawartości na kartę SD. Dzieje się tak ze względu na to, że najmniejsza możliwa ilość danych do zapisania na karcie jest równa rozmiarowi jej bloku, czyli 512 bajtów. W systemie FAT32 dysk podzielony jest na klastry, przy czym każdy klaster składa się z pewnej liczby sektorów, która jest zależna od pojemności karty. Sektor ma rozmiar 512 bajtów. Sektory są zapisywane zawsze po kolei w obrębie jednego klastra. Natomiast klastry mogą być zapisywane bez zachowania ciągłości. Umożliwia to łatwe zapisywanie nowych plików w miejsce skasowanych danych, jednak oznacza konieczność ścisłej kontroli powiązania ze sobą kolejnych klastrów. W przypadku złego powiązania klastrów otrzymany plik będzie widoczny jako uszkodzony. Kod odpowiedzialny za kontrolę miejsca zapisu oraz powiązanie kolejnych klastrów zajmowanych przez plik znajduje się w kolejnych liniach listingu.

Kalibrację z wykrywaniem osi pionowej jest pokazano na **list. 3**. Po pierwsze,

List. 4. Funkcja kalibracji z wykrywaniem osi pionowej

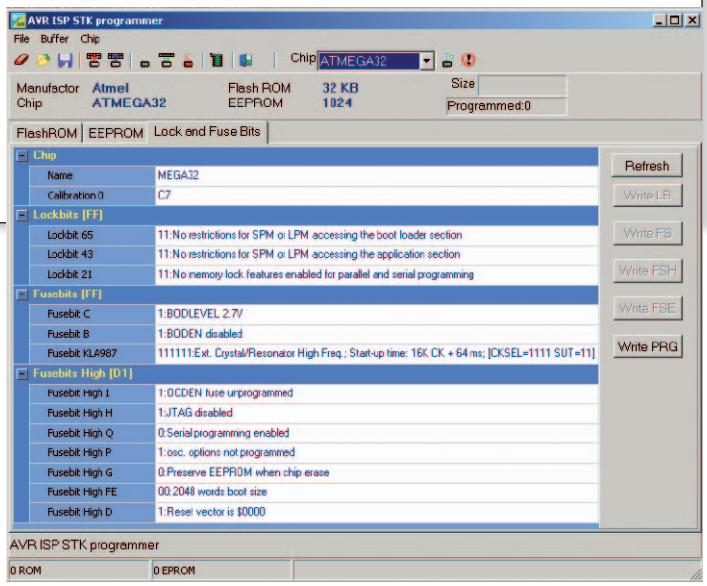
```
if(ADC_X > 511) X -= 511;           //wyznaczenie delt dla wszystkich osi
else X = 511 - ADC_X;
if(ADC_Y > 511) Y -= 511;
else Y = 511 - ADC_Y;
if(ADC_Z > 511) Z -= 511;
else Z = 511 - ADC_Z;
                                    //wyznaczenie osi pionowej
if(X > Y)
{
   if(X > Z) os_pion = 1;
   else os_pion = 3;
}
else
{
   if(Y > Z) os_pion = 2;
   else os_pion = 3;
}

switch(os_pion)                     //wyznaczanie poprawek
{
   case 1: if(ADC_X > 511) popr_X = ((5110/zakres) + 511) - ADC_X;
           else popr_X = (511 - (5110/zakres)) - ADC_X;
           popr_Y = 511 - ADC_Y;
           popr_Z = 511 - ADC_Z;
           break;
   case 2: popr_X = 511 - ADC_X;
           if(ADC_Y > 511) popr_Y = ((5110/zakres) + 511) - ADC_Y;
           else popr_Y = (511 - (5110/zakres)) - ADC_Y;
           popr_Z = 511 - ADC_Z;
           break;
   case 3: popr_X = 511 - ADC_X;
           popr_Y = 511 - ADC_Y;
           if(ADC_Z > 511) popr_Z = ((5110/zakres) + 511) - ADC_Z;
           else popr_Z = (511 - (5110/zakres)) - ADC_Z;
           break;
}
```

List. 5. Funkcja odświeżania ekranu wyników

```
void G_LCDRefresh1(uint8_t os , uint16_t wyn)
{
   uint8_t dg = 1023/(GRAPH_H-1);                                  //wyznaczenie działki

   if(dt == LCD_W - GRAPH_X)                                       //zapętlenie wykresu
   {
      dt = 0;                                                      //zerowanie zmiennej odcinka czasu
      LCD_SetColor(WHITE);                                         //czyszczenie wykresu
      LCD_Rect(GRAPH_X, GRAPH_Y, GRAPH_W, GRAPH_H, 1);
      DrawGrid();                                                  //procedura rysująca siatkę
   }

   LCD_SetColor(WHITE);                                            //czyszczenie pola wyniku
   LCD_Rect(7, (10*os+6), 25, 8, 1);
   if(wyn > max[os-1]) LCD_Rect(1, 46, 31, 8, 1);                  //oraz pól max i min, jeśli potrzeba
   else if(wyn < min[os-1]) LCD_Rect(1, 56, 31, 8, 1);


   if(os == 1) LCD_SetColor(RED);                                  //wybór koloru osi
   else if(os == 2) LCD_SetColor(GREEN);
   else LCD_SetColor(BLUE);

   LCD_Pixel((GRAPH_X +1 + dt), (GRAPH_Y + GRAPH_H - 2 - (wyn/dg))); //rysowanie na wykresie

   PrintResult(7, (10*os+6), wyn);                                 //konwersja i wyświetlanie obecnego wyniku

   if(wyn > max[os-1])                                             //wyświetlenie max i min, w razie potrzeby
   {
      max[os-1] = wyn;
      PrintResult(7, 46, wyn);
   }
   else if(wyn < min[os-1])
   {
      min[os-1] = wyn;
      PrintResult(7, 56, wyn);
   }

   if(os == 3) dt ++;
}
```

wyznaczane są bezwzględne odchylenia od wartości 0 g dla każdej osi. Następnie wykrywana jest oś pionowa, która powinna mieć największą wartość bezwzględnego odchylenia. Po wykryciu i oznaczeniu osi pionowej następuje wyliczenie poprawek dla poszczególnych osi. Zmienne globalne *popr_** są typu całkowitego ze znakiem, co upraszcza ich uwzględnianie – wyznaczoną poprawkę wystarczy dodać do otrzymanego wyniku konwersji. W przypadku kalibracji ze sprowadzeniem wszystkich wartości do zera poprawki są wyliczane bezpośrednio, z pominięciem wykrywania osi pionowej.

Procedura odświeżająca podstawowy ekran (widok hybrydowy) jest pokazana na **list. 4**. Jako pierwsza wyliczona jest zmiana wyniku przetwarzania, odpowiadająca zmianie położenia punktu na osi pionowej wykresu. W kolejnym kroku sprawdzane jest, czy jest jeszcze miejsce na wykresie, jeśli nie, to zostaje on wyczyszczony. Przed wyświetleniem nowych wartości liczbowych również należy najpierw wyczyścić pole wyników, co jest realizowane w dalszych liniach. Po przygotowaniu obrazu następuje wybór koloru, uzależniony od odświeżanej osi (X – czerwony, Y – zielony, Z – niebieski). Następnie kolorowany jest punkt wykresu z wyliczonych na podstawie zmiennej zawierającej wynik konwersji sygnału z przetwornika a/c dla danej osi, zmiennej przechowującej zakres (rzeczywista wartość zakresu pomnożona 10 razy, czyli 25 dla zakresu 2,5), zmiennej działki pionowej *dg* oraz zmiennej pozycji na osi czasu *dt*. Stałe *GRAPH_** zawierają wysokość i szerokość wykresu w pikselach. Zmienna pozycji na osi czasu zostaje inkrementowana każdorazowo po aktualizacji wszystkich trzech osi. Wartości liczbowe są wyświetlane przy użyciu oddzielnej procedury *PrintResult* (**list. 5**). Procedura przyjmuje jako argumenty dwie liczby całkowite, oznaczające pozycję na wyświetlaczu, która jest początkiem wyświetlanego wyniku. Trzeci argument to wynik konwersji. Na początku wynik konwersji jest zamieniany na wartość przyspieszenia ze znakiem. Następnie wypisywany jest odpowiedni znak i wartość przyspieszenia. Stała 48, dodawana do każdej cyfry wyniku, jest przesunięciem na początek zestawu cyfr w tablicy znaków.



Rys. 11. Ustawienie fusebitów mikrokontrolera ATmega32

Na koniec opisu sposobu działania programu chciałbym zwrócić szczególną uwagę Czytelników na ustawienie fusebitów. Po pierwsze, należy zmienić ustawienia bitów odpowiadających za ustalenie źródła i zakresu częstotliwości taktowania mikroprocesora. W przypadku tego projektu jest to ustawianie na zewnętrzny rezonator kwarcowy o wysokich częstotliwościach. Dodatkowo, dobrym pomysłem jest wyłączenie automatycznego kasowania pamięci EEPROM podczas programowania mikroprocesora, aby przedłużyć jej żywotność. Ważne jest też wyłączenie interfejsu JTAG, który jest domyślnie włączony, przez co portem C mikrokontrolera nie można sterować z poziomu własnej aplikacji. Na zrzucie ekranu (**rys. 11**) pokazano okno programatora zintegrowanego w środowisku BASCOM AVR, gdzie widać poprawnie ustawione FuseBity. Bardzo lubię używać tego programatora do ustawiania FuseBitów, ponieważ poszczególne ustawienia są tam bardzo dobrze opisane i trudno jest popełnić krytyczny błąd, powodujący np. uniemożliwienie programowania mikrokontrolera przez interfejs szeregowy lub ustawienie złego źródła sygnału taktującego.

Montaż i uruchomienie

Schemat montażowy miernika pokazano na **rys. 12**. Prawie całe urządzenie zostało zbudowane w technologii SMT. Jedyne elementy przewlekane to: dławik, rezonator kwarcowy, złącze ISP oraz przyciski.

Czytelnikom, którzy nie mają doświadczenia z montażem powierzchniowym, odradzam samodzielne lutowanie. Dodatkową trudnością, nawet dla bardziej doświadczonych, jest przylutowanie obudowy akcelerometru. Został on umieszczony w obudowie QFN (ang. Quad Flat Non – leaded), która ma postać kostki bez

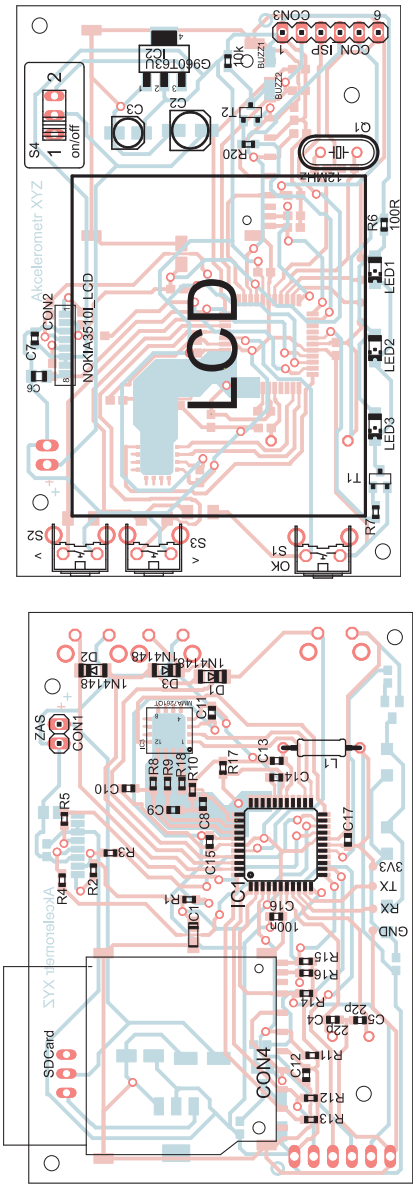
nóżek. Wyprowadzenia stanowią płaskie, metaliczne pola rozmieszczone na obrzeżach od spodu układu scalonego. Dla pocieszenia powiem, że osobiście nie dysponuję lutownicą typu HotAir. Akcelerometr z powodzeniem przylutowałem przy użyciu prostej kolbowej stacji lutowniczej. Metoda, jaką opracowałem do tego celu, jest bardzo prosta, choć zupełnie nieprofesjonalna. Zaczyna się ona już na etapie projektowania płytki obwodu drukowanego. Do footprintu układu scalonego dodaję fragmenty ścieżki o długości 1...1,5 mm (również przy polach niepodłączonych), rozchodzące się promieniście na zewnątrz obrysu układu scalonego. Jeśli płytka jest pokrywana lakierem lub inną powłoką ochronną, należy pozostawić te dodatkowe kawałki ścieżek odsłonięte. W procesie lutowania pokrywamy pola lutownicze wraz z dodatkowymi fragmentami ścieżek cienką warstwą cyny (dodatkowo, nawet jeśli płytka była pocynowana). Dodatkowa ilość cyny powinna tworzyć cienką, możliwie płaską powierzchnię. Na tak przygotowane pola kładziemy układ scalony i delikatnie go ustawiamy, tak aby pola lutownicze układu i płytki pokryły się na wszystkich brzegach. Następnie stosując lekki docisk układu, rozgrzewamy pola lutownicze pod układem, przykładając grot lutownicy do kolejnych kawałków dodanych ścieżek, wystających poza układ scalony.

List. 6. Wyświetlanie wartości liczbowych

```
void PrintResult(uint8_t x, uint8_t y, uint32_t wyn)
{
   int16_t converted = ((((((float)zakres/10)*2)/1024)*wyn)-((float)
zakres/10))*100;

   if(converted < 0)                                    //jesli ujemne
   {
      LCD_Char(,-', x, y);                              //minus
      converted *= -1;
   }
   else LCD_Char(,+', x, y);                            //jesli dodatnie

   LCD_Char((converted/100)+48, (x+5), y);
   LCD_Char(,,', (x+11), y);
   LCD_Char((converted/10)-(converted/100)*10+48, (x+14), y);
   LCD_Char(converted-(converted/10)*10+48, (x+20), y);
}
```

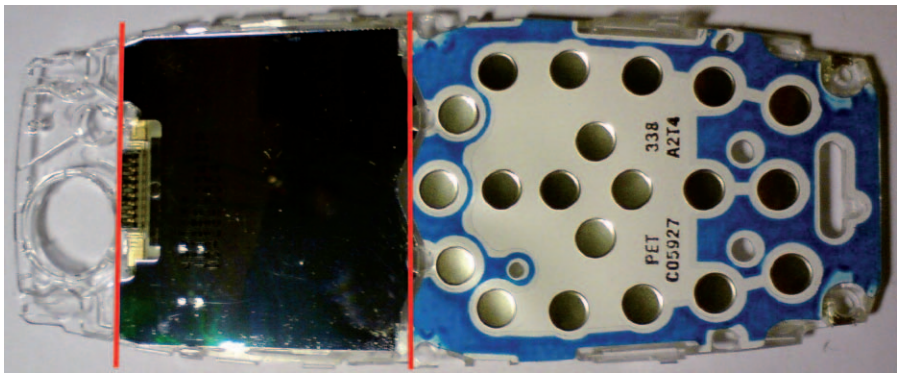
Drugim problemem, z którym musimy się zmierzyć, jest przygotowanie wyświetlacza. Zakupiony wyświetlacz jest w rzeczywistości większym modułem i zawiera również głośnik oraz strukturę nośną wraz z naklejonymi przyciskami klawiatury bąbelkowej, (wygląd takiego modułu przedstawia **fot. 13**). Można oczywiście wymontować sam wyświetlacz – jest on przyklejony do struktury nośnej. Jednak element nośny z tworzywa ma dużą zaletę – jest w nim zintegrowana struktura optymalizująca warunki podświet-lania wyświetlacza. Dzięki niej wprowadzone krawędziowo światło podświetla wyświetlacz z dużą skutecznością, zapobiegając efektowi słabnięcia podświetlenia wraz z odległością od źródła światła. Dlatego zdecydowałem się na wykorzystanie tego ułatwienia – postanowiłem wyciąć część struktury nośnej zawierającej wyświetlacz. Linie cięcia zaznaczone na spodniej części modułu widać na **fot. 14**. Pierwsza linia przebiega tuż nad czarną wkładką zawierającą styki. Druga jest poprowadzona przez otwory, w które wchodzą diody LED przylutowane do płyty głównej telefonu. Przy pracy należy zachować szczególną ostrożność, wyświetlacz jest dość wytrzymały, jednak nawet małe pęknięcie może być fatalne w skutkach.

Właściwy proces montażu rozpoczynamy tradycyjnie od elementów najmniejszych. W tym przypadku będą to wszystkie tranzystory i diody oraz większość kondensatorów i rezystorów (wyjątkiem są duże kondensatory elektrolityczne C2 i C3 oraz elementy w pobliżu złącza wyświetlacza). Czytelnicy, którzy będą próbowali przylutować akcelerometr przy użyciu lutownicy kolbowej, powinni pominąć montaż elementów znajdujących się blisko ak-



Rys. 12. Schemat montażowy akcelerometru

Fot. 13. Wygląd modułu wyświetlacza



Fot. 14. Sposób odcięcia wyświetlacza

celerometru, aby nie utrudniać sobie dostępu w kolejnych krokach. Dobrym pomysłem będzie wlutowanie w następnej kolejności stabilizatora. Nawet bez kondensatorów C2 i C3 możemy wtedy sprawdzić poprawność jego działania i uniknąć ewentualnego uszkodzenia układów scalonych.

Po sprawdzeniu zasilania możemy przylutować akcelerometr (wraz z elementami znajdującymi się najbliżej, jeśli jeszcze tego nie zrobiliśmy) oraz mikroprocesor. W dalszej kolejności montujemy: rezonator kwarcowy, dławik, przyciski włącznik zasilania, złącze ISP, buzzer. Montując dławik, należy tak przyciąć jego wyprowadzenia, aby nie wystawały z otworów w płytce po stronie wyświetlacza. Na końcu delikatnie lutujemy wyświetlacz. Ponieważ połączenie polegające wyłącznie na docisku nóżek wyświetlacza do pól na płytce było trudne do realizacji i w próbach dawało mało stabilne działanie wyświetlacza, zdecydowałem się na przylutowanie złącza do płytki. Dla pewności postanowiłem przylutować drugą stronę wkładki ze stykami, do pól na zintegrowanym sterowniku wyświetlacza. Wydaje mi się, że wygodniej jest najpierw przeprowadzić tę operację, a następnie włożyć wyświetlacz z powrotem w strukturę nośną, ustawić go na płytce i ostrożnie przylutować zewnętrzne styki do płytki, uważając na możliwość tworzenia się mostków pomiędzy sąsiednimi polami. Po przylutowaniu wyświetlacza uzupełniamy brakujące rezystory i kondensatory w pobliżu złącza wyświetlacza. Zmontowane urządzenie wymaga zaprogramowania, do tego służy wyprowadzone złącze ISP. Aby wyeliminować problemy związane z załadowaniem złych ustawień, dobrze jest zaprogramować również wewnętrzną pamięć EEPROM mikroprocesora. Zrzut zawartości pamięci EEPROM z zapisanymi poprawnie ustawieniami jest dołączony do plików programu.

Urządzenie należy zasilić napięciem z przedziału 3,6...10 V. Osobiście z powodzeniem stosowałem do zasilania baterię litowo-jonową o napięciu znamionowym 3,6 V, pochodzącą ze starego telefonu komórkowego. Bateria taka mieści się całkowicie w obrysie płytki i jest idealna do używania urządzenia w terenie.

Dla dociekliwych

Przy wkładaniu karty SD mogą pojawić się znaczące spadki napięcia zasilania i karta SD może zostać źle zainicjalizowana, co prowadzi do braku lub błędnej współpracy. W takim przypadku można próbować skompensować wpływ dołączania karty na stabilność pracy urządzenia przez zwiększenie wartości pojemności kondensatora C12.

W pamięci programu mikroprocesora pozostało wolne ponad 6 kB, więc możliwości wprowadzenia zmian są duże. Na złącza wyprowadzono linie interfejsu szeregowego, wraz zasilaniem. Po podłączeniu konwertera poziomów (np. MAX232) lub translatora USART/USB (FT232) można podłączyć urządzenie bezpośrednio do komputera, co po dopisaniu dodatkowych funkcji może służyć np. do wysyłania zarejestrowanych próbek na bieżąco do komputera i wizualizację w czasie rzeczywistym. Podłączone do komputera urządzenie można próbować wykorzystać jako kontroler, jak w konsoli Nintendo Wii.

Maksymalną częstotliwość próbkowania sygnałów z akcelerometru można zwiększyć przez zmniejszenie dzielnika częstotliwości taktującego przetwornik analogowo-cyfrowy. Tym samym zmniejszy się przesunięcie pomiędzy próbkami z różnych osi w tym samym czasie. Należy jednak pamiętać, że zwiększenie częstotliwości próbkowania oznacza szybki przyrost rozmiaru rejestrowanego pliku. Ma to duże znaczenie przy obróbce danych, ponieważ starsze arkusze kalkulacyjne mają ograniczenie ilości danych przy rysowaniu wykresów. Nowsze arkusze kalkulacyjne potrafią wyrysować wykres z większej ilości danych, jednak praca na takim wykresie może być uciążliwa, ze względu na znaczące czasy przeliczania wykresu składającego się z dużych ilości danych.

Marcin Pomianowski
m.pomianowski@mchtr.pw.edu.pl