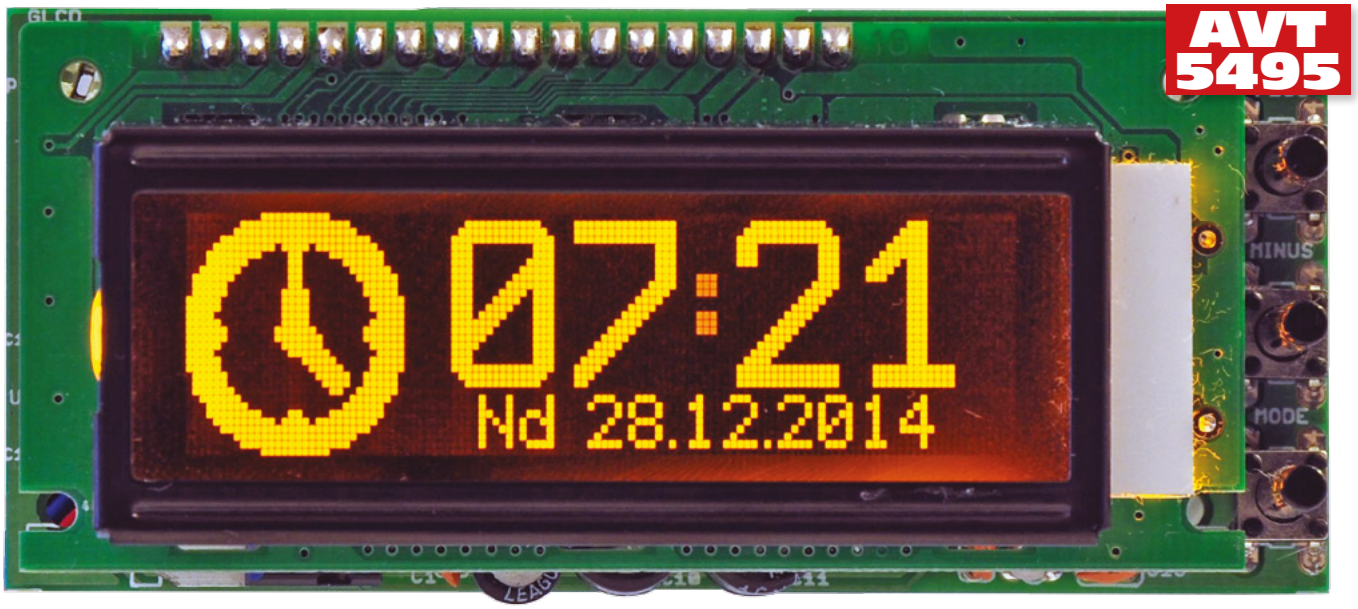


Mee Uniwersalny komputer samochodowy Mee

Zbierając wieloletnie doświadczenia w przedmiotowej materii, postanowiłem zaprojektować nowe urządzenie, które to musiało spełniać następujące kryteria: pełna funkcjonalność typowego komputera pokładowego, czytelny, intuicyjny i atrakcyjny interfejs użytkownika, możliwość personalizacji urządzenia poprzez dobór wyświetlacza o żądanym kolorze podświetlenia jak i kolorze wyświetlanej treści.

Rekomendacje: *komputer jest przeznaczony dla osób, które chcą podwyższyć funkcjonalność swojego samochodu.*

Mamy XXI wiek, lądownik Philae napędzany własnymi silnikami ląduje na komecie 67P/Czuriumow-Gierasimienko, zaś amerykańska agencja kosmiczna NASA ogłasza, że program załogowych lotów na Marsa rozpocznie się na dobre w okolicach 2025 roku! A na jakim etapie rozwoju jest rynek motoryzacyjny? Powiem Wam na jakim – jest w rękach ekonomistów i speców od marketingu! Bo jak inaczej wytłumaczyć fakt, iż przy dzisiejszym stanie wiedzy, stopniu zaawansowania technologicznego obecnie produkowanych pojazdów (ale nawet tych sprzed 20 lat), które to już od dawna są wyposażane w różne komputery i sterowniki nadal elementem opcjonalnym (czyt. wymagającym dopłaty) jest dla wielu producentów z branży motoryzacyjnej tak zwany komputer pokładowy. Sytuacja jest tym bardziej kuriozalna, iż wszystkie dane, które standardowo pokazuje urządzenie tego typu, dostępne są bez problemu z poziomu komunikacji ze sterownikiem silnika ECU, w jaki jest wyposażony każdy nowszy pojazd, więc ich ewentualne wyświetlenie to sprawa kilku linijek dodatkowego kodu. Pikanterii sprawie dodaje fakt, że tak forsowana przez Unię Europejską polityka ekologiczna, która to w ostatnim czasie wymusiła na producentach pojazdów wyposażanie ich w czujniki niskiego ciśnienia powietrza w oponach, tłumacząc tego typu wymóg potrzebą

W ofercie AVT*

AVT-5495 A	AVT-5495 B	AVT-5495 C	AVT-5495 UK

Podstawowe informacje:

- Napięcie zasilające: 8…15 V DC,
- Maksymalny prąd obciążenia źródła +12V: 14 mA.
- Prąd podtrzymania zegara RTC (ze źródła BATT): 1 mA.
- Maksymalny prąd podświetlenia (ze źródła ILL+): 15 mA.
- Rozdzielczość pomiarowa temperatury: 0,5°C.
- Zakres pomiarowy temperatury zewnętrznej i wewnętrznej: -55°C…+99°C
- Zakres pomiarowy prędkości pojazdu: 0…255 km/godz.
- Zakres pomiarowy prędkości obrotowej silnika: 0…9999 obr./min.
- Zakres pomiarowy chwilowego zużycia paliwa: 0…99,9 l/100 km.
- Zakres pomiarowy średniego zużycia paliwa: 0…25,5 l/100 km.
- Zakres pomiarowy paliwa dostępnego w baku: 0…99,9 l.
- Zakres pomiarowy przejechanego dystansu: 0…9999 km.
- Zakres pomiarowy dostępnego dystansu: 0…999 km.
- Zakres pomiarowy liczby zapłonów: 0…9999.
- Zakresy regulacji parametrów konfiguracyjnych:
  - Stała wtryskiwacza: 1…999 ml/min.
  - Stała przetwornika drogi: 1…99 imp./obr.
  - Obwód opony: 1…255 cm.
  - Liczba cylindrów: 2…8.

Dodatkowe materiały na FTP:

ftp://ep.com.pl, user: 64311, pass: 877yqakt

- wzory płytek PCB

Projekty pokrewne na FTP:

(wymienione artykuły są w całości dostępne na FTP)

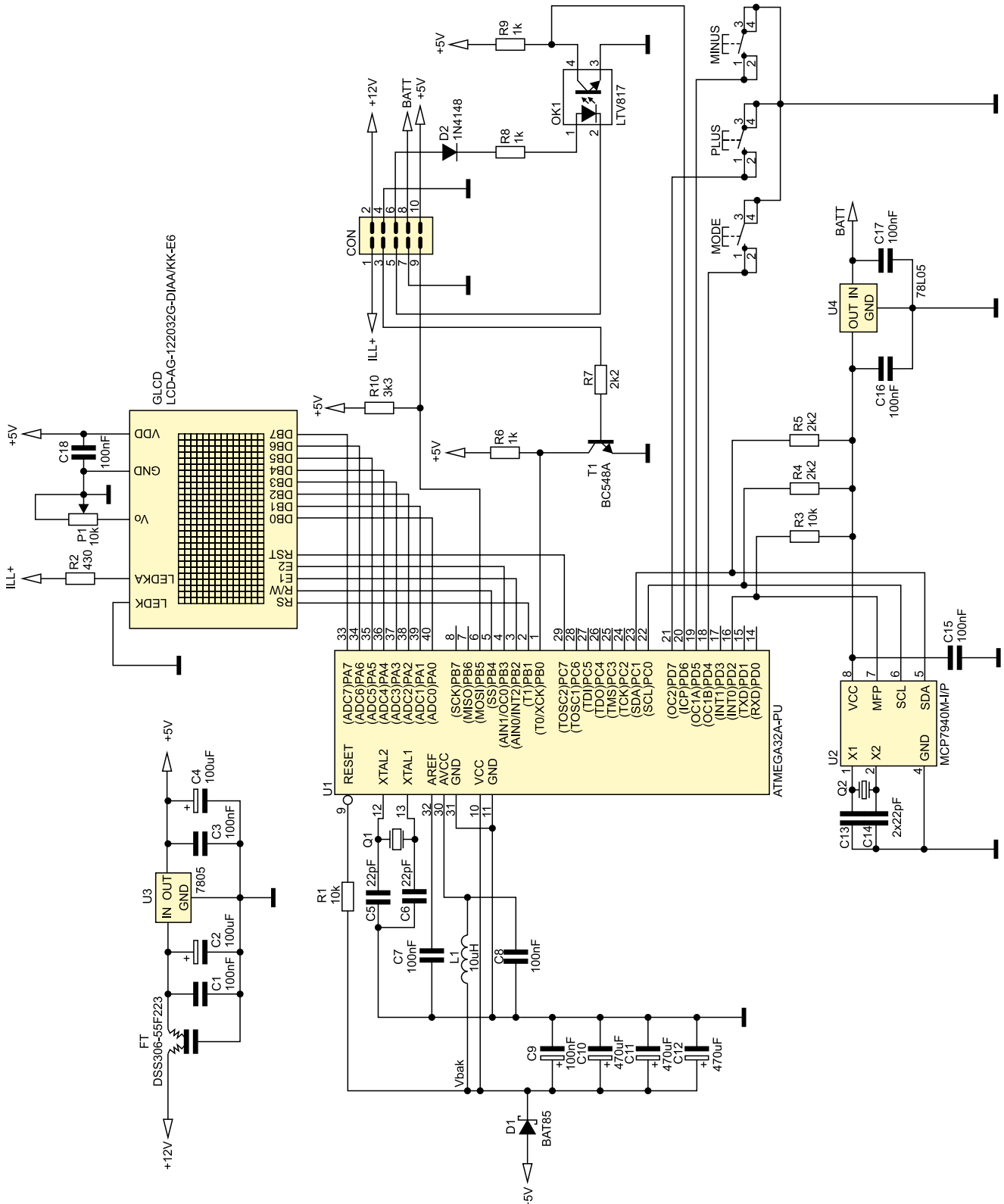
AVT-3095	Komputer samochodowy (EdW 4-5/2014)
AVT-5405	TripCo – komputer samochodowy (EP 7/2013)
AVT-5395	TIDex – komputer dla samochodów z silnikiem Diesla (EP 5/2013)
AVT-5397	Komputer pokładowy z funkcją tempomatu (EP 5/2013)
AVT-1664	Transceiver CAN (EP 2/2012)
AVT-5280	Urządzenie diagnostyczne do sieci CAN (EP 3/2011)
AVT-5271	VAGlogger – Przyrząd diagnostyczny dla samochodów z grupy VW – Audi (EP 1/2011)
AVT-5260	Obrotomierz cyfrowy (EP 10/2010)
AVT-2799	Mikroprocesorowy obrotomierz stroboskopowy (EdW 9/2006)
AVT-434	Komputer samochodowy (EP 9-10/2005)
AVT-2711	Obrotomierz (EdW 2/2004)

* Uwaga:
Zestawy AVT mogą występować w następujących wersjach:
AVT xxxx UK to zaprogramowany układ. Tylko i wyłącznie. Bez elementów dodatkowych.
AVT xxxx A płytka drukowana PCB (lub płytki drukowane, jeśli w opisie wyraźnie zaznaczono), bez elementów dodatkowych.
AVT xxxx A+ płytka drukowana i zaprogramowany układ (czyli połączenie wersji A i wersji UK) bez elementów dodatkowych.
AVT xxxx B płytka drukowana (lub płytki) oraz komplet elementów wymieniony w załączniku pdf
AVT xxxx C to nic innego jak zmontowany zestaw B, czyli elementy wlutowane w PCB. Należy mieć na uwadze, że o ile nie zaznaczono wyraźnie w opisie, zestaw ten nie ma obudowy ani elementów dodatkowych, które nie zostały wymienione w załączniku pdf
AVT xxxx CD oprogramowanie (nieczęsto spotykana wersja, lecz jeśli występuje, to niezbędne oprogramowanie można ściągnąć, klikając w link umieszczony w opisie kitu)
Nie każdy zestaw AVT występuje we wszystkich wersjach! Każda wersja ma załączony ten sam plik pdf! Podczas składania zamówienia upewnij się, którą wersję zamawiasz! (UK, A, A+, B lub C). *http://sklep.avt.pl*

zmniejszania emisji szkodliwych gazów do atmosfery (bo odpowiednio napompowana opona to mniejsze opory tarcia, co powoduje mniejsze spalanie paliwa) jakoś nie mogła „poradzić sobie" ze zmuszeniem tychże producentów do standardowego montażu komputerów pokładowych, które pokazując statystyki spalania jawnie wpisywałyby się we wszechobecną modę na rozwiązania ekologiczne.

No dobrze, polskim zwyczajem, trochę ponarzekałem, lecz czas brać się do „roboty". Wszak na naszych ulicach jeszcze jeździ sporo starszych pojazdów, których użytkownicy nawet nie mogli pomarzyć o takim luksusie. Ale czy to tak naprawdę luksus? Nie sądzę! Wszak jest to urządzenie raczej nieskomplikowane, do którego działania potrzebne są w zasadzie tylko 2 sygnały: sygnał prędkości pojazdu dostępny zwyczajowo w złączu ISO każdego radioodbiornika oraz sygnał wtryskiwacza, który to możemy „pobrać" z dowolnego wtryskiwacza paliwa umieszczonego pod maską pojazdu (bez znaczenia czy pojazd jest zasilany benzyną czy olejem napędowym).

Pewnie zastanawiacie się czy nie przesadzam trochę w kwestii prostoty tego typu rozwiązań? Otóż nie, a opinię swoją poprzeć mogę kilkuletnim doświadczeniem



Rysunek 1. Schemat ideowy komputera pokładowego Mee

w konstrukcji tego typu urządzeń, którego owocem były choćby takie projekty jak publikowane na łamach EP projekty TIDex (EP 05/2013 i KIT AVT5395) i TripCo (EP 07/2013 i KIT AVT5405). Po cóż w takim razie kolejny? Odpowiedź jest prosta! Prezentowane wcześniej urządzenia, mimo, że cechowały się zbliżoną do bieżącego projektu funkcjonalnością, nie były rozwiązaniami uniwersalnymi w każdym tego słowa znaczeniu. TIDex współpracować mógł tylko i wyłącznie z pojazdami marki Opel, jako że korzystał z wbudowanego w pojazd, oryginalnego wyświetlacza pokładowego, zaś TripCo, mimo iż wyposażony był w szalenie atrakcyjny interfejs użytkownika zbudowany przy użyciu nowoczesnego, graficznego wyświetlacza OLED, właśnie z tego powodu (a dokładnie ceny takiego elementu i sposobu jego podłączenia) nie był w zasięgu przeciętnego Kowalskiego! Zbierając wieloletnie doświadczenia w przedmiotowej materii, postanowiłem zaprojektować nowe urządzenie, które to musiało spełniać pewne, wstępne i nienaruszalne założenia funkcjonalno/instalacyjne. W związku z tym, nowy projekt musiał spełniać następujące kryteria:

Tabela 1. Sposób konfigurowania i funkcjonalność modułów peryferyjnych mikrokontrolera ATmega32A-PU dla mechanizmu pomiarowego komputerach Mee

Nazwa	Sposób konfiguracji	Funkcjonalność
Timer0	Licznik impulsów zewnętrznych podawanych na wejście T0 – zlicza przy zboczu opadającym.	Liczy impulsy przetwornika drogi WEG zapewniając pomiar przejechanego dystansu
Timer1	Licznik taktowany wewnętrznym sygnałem zegarowym o częstotliwości 65536 Hz (Preskaler=64, 65536 impulsów na 1 ms). Wejście Input Capture dołączone do układu formującego przebieg z wtryskiwacza paliwa.	Mierzy sumaryczny czas wtrysku w czasie 1 sekundy zapewniając pomiar zużytego paliwa.
Przerwanie INT0	Wyzwalane zboczem opadającym na wejściu INT0, do którego podłączono wyjście częstotliwości wzorcowej zegara RTC	Służy do odmierzania czasu pomiaru równego 1 s.

Listing 1. Funkcje odpowiedzialne za realizację mechanizmu pomiarowego urządzenia Mee

```
void initMeasurment(void)
{
//Konfigurujemy Timer0 odpowiedzialny za zliczanie impulsów drogi WEG
    TCCR0 = (1<<CS02)|(1<<CS01);  //Impulsy podawane na wejście T0 - aktywne zbocze opadające
    TIMSK |= (1<<TOIE0);
//Konfigurujemy Timer1 odpowiedzialny za obliczanie sumarycznego czasu wtrysku
    TCCR1B |= (1<<ICNC1);  //Włączenie układu: Input Capture Noise Canceler
    TCCR1B &= ~(1<<ICES1);  //Zbocze opadające na ICP wyzwala przechwytywanie wartości licznika
    TCCR1B |= (1<<CS11)|(1<<CS10);  //Licznik rusza, Preskaler = 64, 65.536 impulsów na 1ms
    TIMSK |= (1<<TICIE1);  //Input Capture Interrupt Enable
//Podciągamy porty wejsciowe INT0, T0 i ICP1 do VCC
    PORTB |= (1<<PB0);  //T0
    PORTD |= (1<<PD6)|(1<<PD2);  //ICP1 i INT0
//Konfigurujemy przerwanie zewnętrzne INT0, które wyzwalane jest sygnałem 1Hz z układu MCP7940
    MCUCR = (1<<ISC01);  //Zbocze opadające na wejściu INT0 wywołuje przerwanie INT0
    GICR |= (1<<INT0);  //External Interrupt Request 0 Enable
}


//######################################################################
ISR(INT0_vect)  //Przerwanie zachodzi co 1s, gdyż wyzwalane jest sygnałem 1Hz z układu RTC
{
    STOP_TIMER0;  //Zatrzymujemy Timer0 zliczający impulsy WEG
    TIMSK &= ~(1<<TICIE1);  //Dezaktywujemy tymczasem przerwanie ICP
//Obliczamy sumaryczną liczbę impulsów WEG zliczoną w ciągu 1s
    WEGpulses = ((uint16_t)T0overflows << 8) + TCNT0;
//Sumaryczna długość czasów wtrysku w ciągu 1s
    injectionTime = Injections;
//Liczba obrotów wału silnika w czasie 1s (równa liczbie zboczy impulsów wtrysku dla silnika czterosuwowego)
    rotationsPerSecond = injectionPulses;
//Retartujemy zmienne by rozpocząć proces obliczeniowy od nowa
    TCNT0 = 0;
    T0overflows = 0;
    Injections = 0;
    injectionPulses = 0;
    START_TIMER0;
    TCCR1B &= ~(1<<ICES1);  //Zbocze opadające na ICP wyzwala przechwytywanie wartości licznika
    TIFR |= (1<<ICF1);  //Skasowanie ewentualnej flagi przerwania ICP
    TIMSK |= (1<<TICIE1);  //Input Capture Interrupt Enable
//Zgłoszenie gotowości danych obliczeniowych dla funkcji Main
    dataReady = 1;
}

//######################################################################

ISR(TIMER0_OVF_vect)  //Przerwanie liczy liczbę przepełnień Timera0 zliczającego impulsy z wejcia T0 (WEG)
{
    T0overflows++;
}

//######################################################################

ISR(TIMER1_CAPT_vect)  //Przerwanie od przechwycenia Timera1 - uruchamiane początkowo zboczem opadającym na ICP1
{
    static uint16_t lastCapture;
    uint16_t pulseWidth;
//Ustawione było zbocze opadające na ICP1 więc odczytujemy zatrzasniętą wartosc licznika Timer1
    if(!(TCCR1B & (1<<ICES1)))  lastCapture = ICR1;
//Wystąpił impuls rosnący, więc mamy już długosc impulsu wtrysku - sumujemy czas wtrysków w ciągu 1s
    else
    {
//Obliczamy długość bieżącego impulsu na wejściu ICP1
        pulseWidth = ICR1 - lastCapture;
//Pod uwagę bierzemy tylko i wyłącznie impulsy, których długość mieści się w zakresie dopuszczalnym
        if(pulseWidth>MIN_INJ_TIME && pulseWidth<MAX_INJ_TIME) Injections += pulseWidth;
    }
    TCCR1B ^= (1<<ICES1);  // Zmiana zbocza wyzwalającego przerwanie ICP na zbocze przeciwne, niż aktualne
//Zliczamy impulsy wtrysku (w zasadzie zbocza impulsów, ale podzielone przez 2 dadzą liczbę impulsów)
    injectionPulses++;
}
```

- Pełna funkcjonalność typowego komputera pokładowego.
- Czytelny, intuicyjny i atrakcyjny (najlepiej graficzny) interfejs użytkownika.
- Możliwość personalizacji urządzenia poprzez dobór wyświetlacza o żądanym kolorze podświetlenia jak i kolorze wyświetlanej treści.
- Możliwość regulowania jasności podświetlenia wyświetlacza graficznego zgodnie z ustawieniem jasności podświetlenia zegarów pojazdu.
- Łatwość montażu uzyskana poprzez zastosowanie łatwo dostępnych, niedrogich i prostych w montażu elementów elektronicznych.
- Niewielkie wymiary zewnętrzne.
- Łatwość instalowania uzyskana poprzez zastosowanie pojedynczego złącza połączeniowego o niewielkiej liczbie wyprowadzeń (a więc i liczbie niezbędnych sygnałów sterujących).
- Niska cena całego urządzenia **na poziomie 100–120 zł**!

Ponadto, postawiłem dość wysokie wymagania dotyczące funkcjonalności komputera pokładowego, którego to zdecydowałem się wyposażyć w następującą funkcjonalność:

- Pomiar i pokazywanie temperatury wewnątrz i na zewnątrz pojazdu oraz ostrzeżenia o śliskiej nawierzchni (przy temperaturze zewnętrznej niższej niż 3°C).
- Funkcja automatycznego powiadamiania o konieczności włączenia świateł mijania (po osiągnięciu przez pojazd prędkości powyżej 5 km/godz.).
- Pokazywanie chwilowej prędkości pojazdu (w km/godz.).
- Pokazywanie średniej prędkości pojazdu na przejechanym odcinku drogi (w km/godz.).
- Pokazywanie maksymalnej prędkości pojazdu na przejechanym odcinku drogi (w km/godz.).
- Pokazywanie prędkości obrotowej silnika (w obr./min).
- Pokazywanie chwilowego zużycia paliwa (w l/godz. dla prędkości ≤ 5 km/godz. oraz l/100 km dla pozostałych prędkości).
- Pokazywanie średniego zużycia paliwa (w l/100 km).
- Pokazywanie paliwa pozostającego w baku pojazdu(w litrach).
- Pokazywanie przewidywanego zasięgu pojazdu na paliwie pozostającym w baku pojazdu (w km).
- Pokazywanie przejechanego dystansu od ostatniego zerowania stanu licznika (w km).
- Pokazywanie liczby uruchomień zapłonu.
- Pokazywanie aktualnego czasu i daty (z zastosowaniem mechanizmu podtrzymania zasilania).
- Cykliczne (co 1 minutę) pokazywanie imienin dla bieżącego dnia roku (wbudowany kalendarz imienin).

Czy możliwe jest zbudowanie urządzenia spełniające powyższe kryteria a dodatkowo wyróżniającego się tak niskim kosztem implementacji? Jak najbardziej, a by się o tym przekonać wystarczy spojrzeć na schemat ideowy komputera samochodowego Mee pokazany na **rysunku 1**.

Komputer zbudowano z użyciem mikrokontrolera firmy Atmel typu ATmega32A-PU, zegara czasu rzeczywistego typu MCP7940M-I/P (Microchip) wyposażonego w mechanizm awaryjnego przełącznika zasilania oraz graficznego wyświetlacza LCD o rozdzielczości 122×32 piksele wyposażonego w sterownik ekranu NJU6450A (odpowiednik SED1520) pełniącego rolę graficznego interfejsu użytkownika GUI. Mikrokontroler, jak to zwykle bywa, stanowi „serce" niniejszego sterownika i realizuje całą, założoną funkcjonalność urządzenia posiłkując się w tym celu szeregiem wewnętrznych modułów peryferyjnych. Mikrokontroler jest taktowany zewnętrznym rezonatorem kwarcowym 4,194 MHz, dzięki czemu jest zapewniona duża dokładność pomiarów czasu, niezbędna z punktu widzenia zastosowanych mechanizmów sprzętowo-programowych realizujących funkcje pomiarowe oraz mały pobór mocy ze źródła zasilania. W celu realizacji założonej funkcjonalności, w programie obsługi niniejszego sterownika, wykorzystano dwa sprzętowe układy czasowo-licznikowe znajdujące się „na pokładzie" mikrokontrolera oraz jedno przerwanie zewnętrzne. Sposób konfiguracji wspomnianych modułów peryferyjnych oraz realizowaną przez nie funkcjonalność pokazano w **tabeli 1**.

Co bardziej dociekliwi Czytelnicy mogą zapoznać się z listingiem wszystkich funkcji, które odpowiedzialne są za realizację wspomnianego mechanizmu pomiarowego, a które pokazano na **listingu 1**.

Dodatkowo, w programie obsługi urządzenia, wykorzystano także ostatni, dostępny układ czasowo-licznikowy mikrokontrolera, Timer2 został skonfigurowany do pracy w trybie CTC. Jego zadaniem jest generowanie cyklicznych przerwań systemowych (co 10 ms) służących mechanizmowi nieblokującej obsługi klawiatury użytkownika (przyciski PLUS, MODE i MINUS) stanowiącej element interfejsu GUI. Dzięki zastosowaniu tego typu rozwiązania, program obsługi aplikacji urządzenia nie wykorzystuje żadnych z wbudowanych w kompilator GCC opóźnień, co zapewnia jego bezproblemową pracę, jak i możliwość prostej detekcji czasu naciśnięcia przycisku (krótki/długi/przytrzymanie itd.), dzięki czemu udało się w sposób znaczący zoptymalizować sposób obsługi całego urządzenia zwiększając tym samym ergonomię pracy. Tak jak pokazano w tab. 1, urządzenie dokonuje w czasie każdej jednej sekundy pomiaru sumarycznej liczby impulsów doprowadzanych na wejście T0 mikrokontrolera z przetwornika drogi pojazdu jak i pomiaru sumarycznego czasu wtrysków, których to sygnał doprowadzony jest na wejście przechwytujące licznika Timer1 (wejście ICP1). W celu realizacji drugiej z funkcjonalności zaprojektowano kompletny i bezpieczny układ wejściowy (przy użyciu popularnego optoizolatora LTV817) formujący sygnał wtryskiwacza dla potrzeb wejściowych obwodów mikrokontrolera. Na **listingu 2** przedstawiono wzory zaczerpnięte bezpośrednio z programu obsługi aplikacji a służące obliczeniu wszystkich parametrów rzeczywistych komputera pokładowego.

Czy nie daje Wam do myślenia wzór na obliczenie dostępnego dystansu *availableDistance*? Przecież na pierwszy rzut oka, te wszystkie dzielenia dałoby się uprościć? Właśnie! Na pierwszy rzut oka, jednak zawsze należy mieć na uwadze fakt, na jakiego rodzaju zmiennych operujemy i w jakim zakresie. W tym wypadku zmienne oraz wynik obliczeń są 32-bitowe. Gdyby nie wprowadzono wstępnego dzielenie zmiennej *Accu.remainingFuel* przez 1000, a zmiennej *Accu.Distance* przez 10, to szybko przekroczylibyśmy 32-bitowy wynik mnożenia z licznika ułamka w przedstawionym wzorze, co skutkowałoby niepoprawnym wynikiem obliczeń. To częsty błąd początkujących programistów, więc zwracam szczególną uwagę na zakresy zmiennych oraz wyników poszczególnych obliczeń.

Idąc dalej, uważny Czytelnik z pewnością zauważy pewną „subtelność" dotyczącą sposobu zasilania mikrokontrolera polegającą na implementacji prostego układu chwilowego podtrzymania zasilania zbudowanego przy użyciu diody Schottky D1 oraz bloku kondensatorów elektrolitycznych C10…C12. W jakim celu wprowadzono tego typu rozwiązanie sprzętowe? Urządzenie oblicza i wyświetla średnie wartości zużycia paliwa i prędkości, objętość paliwa pozostającego w baku pojazdu oraz pokonany dystans. Jak łatwo się domyślić i co potwierdzają wzory podane powyżej, w celu wyznaczenia wspomnianych wartości obliczeniowych jest niezbędna znajomość całkowitego zużycia paliwa oraz dystansu od momentu wyzerowania stosownych liczników, a co za tym idzie, staje się niezbędny mechanizm akumulowania mierzonych wartości nawet po zaniku napięcia zasilającego (po wyłączeniu zapłonu). Mikrokontroler ATmega32A-PU dysponuje nieulotną pamięcią EEPROM, jednak ma ona ograniczoną (zwykle do ok. 100 tys.) liczbę gwarantowanych cyklów zapisu. W takim razie jak i kiedy dokonywać zapisu niezbędnych wartości obliczeniowych by nie spowodować szybkiego uszkodzenia tej

Listing 2. Wzory zaczerpnięte bezpośrednio z programu obsługi aplikacji, służące do obliczenia wszystkich parametrów rzeczywistych komputera pokładowego

```
spentFuelPer1s = ((1UL*Config.Cylinders*injectionTime*Config.CcPerMin)/3932UL);
Accu.spentFuel += spentFuelPer1s;
if(Accu.remainingFuel >= spentFuelPer1s) Accu.remainingFuel -= spentFuelPer1s;
Accu.Distance += ((1UL*WEGpulses*Config.Wheel) / (100UL*Config.PulsPerRot));
Speed = ((36UL*WEGpulses*Config.Wheel) / (1000UL*Config.PulsPerRot));

if(Speed<=5) Consum = ((3UL*Config.Cylinders*injectionTime*Config.CcPerMin) / 327680UL);
else
Consum = ((3UL*Config.Cylinders*injectionTime*Config.CcPerMin*Config.PulsPerRot) / (118UL*WEGpulses*Config.Wheel));

SpeedAvg = ((36UL*Accu.Distance)/(10UL*Accu.Measurements));
if(Accu.Distance>999)
{
ConsumAvg = (Accu.spentFuel/Accu.Distance);  //1/100km *10
        availableDistance = (((Accu.remainingFuel/1000)*(Accu.Distance/10)) / Accu.spentFuel)*10;  //km
}
else {ConsumAvg = 0; availableDistance = 0;}
Gdzie:
InjectionTime - sumaryczny czas wtrysku zliczony w czasie 1s [ms*65.536].
WEGpulses - liczba impulsów z przetwornika drogi zliczona w czasie 1 sekundy.
spentFuelPer1s - paliwo spalone w czasie ostatniej sekundy [ul].
Accu.Measurements - akumulator liczby interwałów pomiarowych [s].
Accu.spentFuel - akumulator ilości spalonego paliwa [ul].
Accu.remainingFuel - akumulator ilości paliwa pozostającego w baku [ul].
Accu.Distance - akumulator przejechanego dystansu [m].
Consum - chwilowe zużycie paliwa [l*10/godz.], dla prędkości≤5 km/godz. lub [l*10/100 km], dla prędkości>5 km/godz.
ConsumAvg - średnie zużycie paliwa [l*10/100 km].
Speed - prędkość chwilowa [km/godz.].
SpeedAvg - prędkośc średnia [km/godz.].
availableDistance - orientacyjny, dostępny dystans na paliwie pozostającym w baku [km].
Config.CcPerMin - stała wtryskiwacza [ml/min].
Config.PulsPerRot - stała przetwornika drogi [imp/obr].
Config.Cylinders - liczba wtryskiwaczy paliwa.
Config.Wheel - obwód opony [cm].
```

pamięci? Odpowiedź wydaje się dość prosta, choć samo rozwiązanie – moim zdaniem – całkiem interesujące. Niezbędne wartości obliczeniowe „zebrane" w specjalną strukturę danych zaopatrzoną w sumę kontrolną CRC8 (pozwalającą na sprawdzenie integralności danych) zapisywane są każdorazowo przy wyłączaniu zapłonu (oraz cyklicznie, co 1 sekundę, w pamięci RAM układu RTC). Do wykrycia momentu wyłączania zapłonu zastosowano wbudowany w mikrokontroler przetwornik A/C pracujący w trybie Free Running i monitorujący napięcie *Vbak*, czyli napięcie zasilające wyłącznie mikrokontroler (po zaniku zasilania dioda D1 zapewnia separację zasilania mikrokontrolera od reszty urządzenia a kondensatory C10…C12 zapewniają odpowiedni czas podtrzymania zasilania).

Na pierwszy „rzut oka" nie wydaje się, by komputer pokładowy w jakikolwiek sposób używał przetwornika A/C, ponieważ żaden z kanałów wejściowych nie jest przez niego używany. To prawda. Patrząc na schemat układu i nie mając do dyspozycji listingu programu można by wysnuć taki wniosek. Jest jednak inaczej. Przetwornik A/C mierzy wewnętrzne napięcie odniesienia V_{BG}=1.22 V (wartość dla mikrokontrolera ATmega32A-PU), dzięki temu, iż wewnętrzny, analogowy multiplekser przetwornika może zostać właśnie w ten sposób ustawiony. Napięciem odniesienia jest z kolei napięcie zasilające mikrokontroler dostarczane na wyprowadzenie AVCC (czyli nasze Vbak). Spadek tego napięcia, podczas wyłączania zapłonu, powoduje wzrost wartości wyniku przetwarzania zgodnie z wyrażeniem (korzystamy z 8-bitowej rozdzielczości przetwornika): $V_{ADC} = (V_{BG} * 256)/ Vbak$.

Rysunek 2. Wykres zależności napięcia podtrzymania w funkcji czasu w momencie wyłączenia zapłonu

Procedura obsługi przerwania przetwornika A/C każdorazowo sprawdza czy nie został przekroczony zdefiniowany wcześniej próg obliczeniowy, a jeśli ma to miejsce, to inicjuje proces zapisywania danych krytycznych do wbudowanej pamięci EEPROM, po czym czeka, aż napięcie zasilania spadnie do poziomu zerowania mikrokontrolera, które jest wykonywane przez blok BOD (typowo przy wartości 2,7 V). Wspomniany próg zadziałania ustawiono na 4,3 V, co oznacza, że czas opadania napięcia zasilającego od wartości 4,3 V do 2,7 V jest czasem, w którym mikrokontroler musi przeprowadzić zapis wszystkich danych krytycznych – w naszym wypadku 24 bajtów danych. Jak pokazały testy praktyczne, zastosowanie wspomnianego wcześniej rozwiązania sprzętowego (dioda D1 i kondensatory C10…C12) i mechanizmów programowych zapewnia 100% skuteczność zapisu danych z bardzo dużym marginesem czasowym. Wykres zależności napięcia podtrzymania w funkcji czasu w momencie wyłączenia zapłonu pokazano na **rysunku 2**. Czas dostępny na zapis danych to około 350 ms, co dla wymaganej, maksymalnej wartości rzędu 190 ms (obliczonej i potwierdzonej rzeczywistym pomiarem) daje spory margines bezpieczeństwa. Co oczywiste, potencjalne uszkodzenie w bloku sprzętowym odpowiedzialnym za podtrzymanie napięcia zasilającego (np. uszkodzenie kondensatorów elektrolitycznych C10…C12) będzie skutkowało błędnymi zapisami pamięci EEPROM, gdyż dołączana (do struktury danych) suma kontrolna nie będzie zgodna z oczekiwaną. W takim wypadku oprogramowanie w pierwszej kolejności odczyta kopię struktury danych akumulatorów obliczeniowych z pamięci RAM zegara RTC, do której cyklicznie ją zapisuje i której zawartość podtrzymywana jest nawet w czasie wyłączenia zapłonu (dzięki napięciu BATT), a następnie ponownie sprawdzi integralność odczytanych danych. Jeśli nawet kopia tychże danych okaże się błędna (np. od pojazdu na chwile odłączono akumulator lub sam komputer pokładowy został odłączony od zasilania BATT), urządzenie poinformuje o tym fakcie użytkownika wyświetlając komunikat

Listing 3. Listing konfiguracji oraz procedury obsługi przerwania przetwornika A/C

```
#define VBG 122  //Voltage BandGap, 122=1.22V itd
#define CRITICAL_VOLTAGE 430  //430 = 4.30V itd
#define TRESHOLD (VBG*256/CRITICAL_VOLTAGE)  //Obliczony próg zadziałania zapisu do EEPROMa (dla rozdz. 8bitów)
//Uruchomienie monitora napięcia zasilającego - pozwala na zapis akumulatorów obliczeniowych do EEPROMu przy zaniku
//zasilania. Vref=AVCC, Vin=VBG (1.22V), justowanie wyniku pomiaru do lewej (ADHC zawiera 8-bitowy wynik pomiaru)
ADMUX = (1<<REFS0)|(1<<MUX4)|(1<<MUX3)|(1<<MUX2)|(1<<MUX1)|(1<<ADLAR);
//Uruchomienie przetwornika ADC w trybie Free Running (preskaler=128)
ADCSRA = (1<<ADEN)|(1<<ADSC)|(1<<ADATE)|(1<<ADPS2)|(1<<ADPS1)|(1<<ADPS0);
//Zezwolenie na przerwanie ADC nastąpi później - w funkcji Main

ISR(ADC_vect)
{
    register uint8_t Voltage = ADCH;
    if(Voltage>TRESHOLD)
    {
//Zapis krytycznych danych umieszczonych w strukturze do pamięci EEPROM
        eeprom_write_block(&Accu, &AccuEE, sizeof(Accu));
        _delay_ms(2000);  //Czekamy na zresetowanie mikrokontrolera przez układ BOD
    }
}
```

„Save Error" tuż po włączeniu zasilania, w czasie sprawdzania integralności tychże danych oraz wyzeruje je wszystkie, ponieważ nie będzie wtedy pewności, że ich wartości są prawidłowe. Procedury konfigurowania przetwornika A/C oraz obsługi jego przerwania pokazano na **listingu 3**.

Na koniec kilka słów uwagi na temat zastosowanego układu zegara czasu rzeczywistego – układu firmy Microchip MCP7940. Pojawia się pytanie, dlaczego zastosowałem taki „nietypowy" układ zegara RTC, skoro zwykle w to miejsce wybierane są peryferia znacznie bardziej popularne. Odpowiedź jest prosta: cena, która oscyluje na poziomie poniżej 1$ za sztukę przy zakupie detalicznym! Pamiętamy przecież, iż jednym z założeń projektu była możliwie najniższa cena końcowa, stąd proces doboru każdego z elementów składowych został dobrze przemyślany, by spełnić wspomniane wymagania.

Układ scalony zegara MCP7940 oprócz standardowego zegara i kalendarza (z automatyczną korekcją dla lat przestępnych) następującą funkcjonalność:

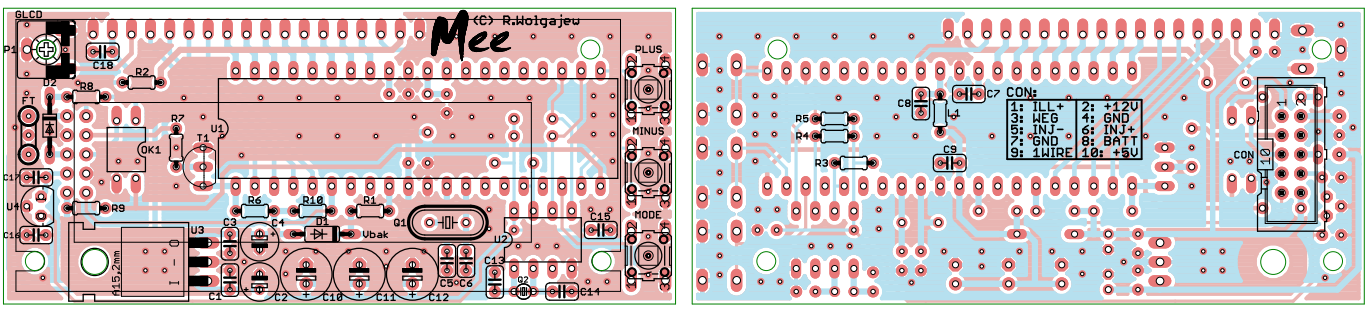
Możliwość ustawienia dwóch, niezależnych alarmów o bardzo rozbudowanej funkcjonalności, jeśli chodzi o dostępne kryteria wywołujące alarm, w tym możliwość zmiany stanu wyjścia MFP, na skutek wywołania alarmu.

- Możliwość programowej kompensacji niedokładności rezonatora kwarcowego (w zakresie -127…+127 ppm.
- Możliwość generowania częstotliwości wzorcowej na wyprowadzeniu MFP z zakresu: 1 Hz, 4096 Hz, 8192 Hz, 32768 Hz.
- Możliwość programowego, niezależnego (od alarmów i generatora częstotliwości wzorcowej) sterowania wyjściem MFP.
- 64 bajty dodatkowej pamięci RAM dostępnej dla aplikacji użytkownika.

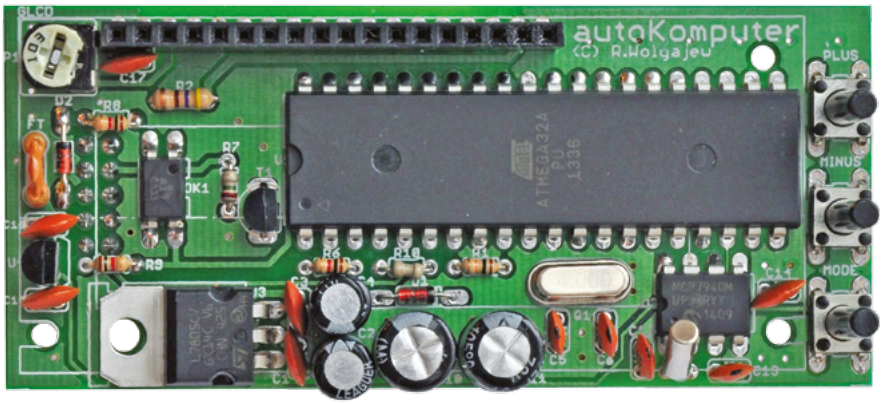
Listing 4. List.4. Listing pliku nagłówkowego zegara RTC typu MCP7940

```
//Definicje typów strukturalnych do obsługi zegara RTC
typedef struct
{
    uint8_t Hour;  //0...23
    uint8_t Minute;  //0...59
    uint8_t Second;  //0...59
}timeType;

typedef struct
{
    uint8_t weekDay;  //1...7
    uint8_t Day;  //1...31
    uint8_t Month;  //1...12
    uint8_t Year;  //0...99
}dateType;

//Definicje adresów układu MCP7940 w trybie zapisu/odczytu
#define MCP7940_WRITE_ADDR 0xDE
#define MCP7940_READ_ADDR 0xDF

//Definicje najważniejszych rejestrów sterujących i ich właściwości
#define TIME_START_REG 0x00
#define START_OSCILLATOR (1<<7)
#define STOP_OSCILLATOR (0<<7)
#define DATE_START_REG 0x03
#define CONTROL_REG 0x07
#define MFP_AS_OUTPUT_1 (1<<7)
#define MFP_AS_OUTPUT_0 (0<<7)
#define MFP_AS_SQUARE_OUTPUT (1<<6)
#define MFP_AS_NORMAL_OUTPUT (0<<6)
#define NO_ALARMS_ACTIVE (0<<4)
#define ALARM0_ACTIVE (1<<4)
#define ALARM1_ACTIVE (2<<4)
#define BOTH_ALARMS_ACTIVE (3<<4)
#define EXTERNAL_OSCILLATOR (1<<3)
#define INTERNAL_OSCILLATOR (0<<3)
#define SQUARE_1HZ 0x00
#define SQUARE_4096HZ 0x01
#define SQUARE_8192HZ 0x02
#define SQUARE_32768HZ 0x03
#define RAM_START_REG 0x20
```

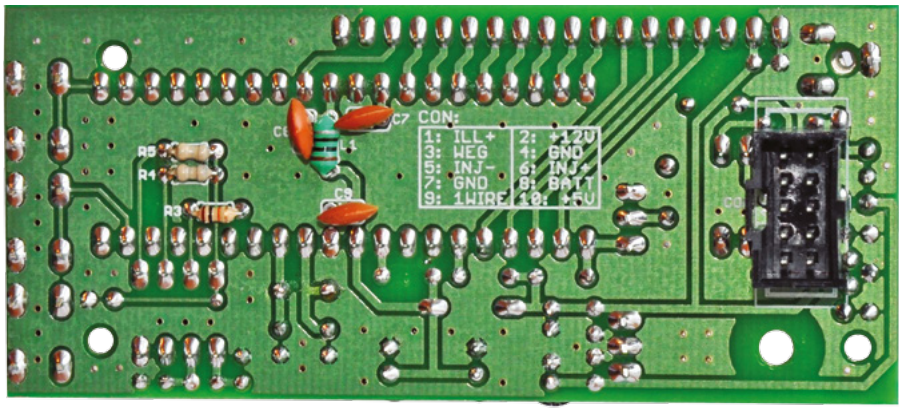
Nie wdając się niepotrzebnie w szczegóły karty katalogowej zaprezentuję kluczowe funkcje narzędziowe niezbędne do nawiązania współpracy z naszym układem zegara RTC. Na **listing 1** pokazano plik nagłówkowy, który tym razem jest dość krótki, gdyż nie zależało mi na ujęciu w jego treści całej funkcjonalności układu (tzn. możliwości alarmowania i kompensacji niedokładności rezonatora kwarcowego). Dla poprawienia czytelności kodu wprowadzono dwa nowe typy strukturalne odpowiedzialne za przechowywanie czasu i daty zegara RTC. Na **listingu 4** pokazano kluczowe funkcje przeznaczone do obsługi zegara MCP7940: funkcję odpowiedzialną



Rysunek 3. Schemat montażowy komputera pokładowego Mee



Fotografia 4. Wygląd prototypowego obwodu drukowanego od strony elementów (bez wyświetlacza LCD)



Fotografia 5. Wygląd prototypowego obwodu drukowanego od strony wyprowadzeń (złącza CON)

za skonfigurowanie układu, funkcje przeznaczone do zapisu i do odczytu godziny oraz daty, funkcje przeznaczone do odczytu i zapisu wbudowanej w układ MCP7940 pamięci RAM przeznaczonej na dane użytkownika. Dodatkowo, na wspomnianym listingu, pokazano dość ciekawą a niezmiernie prostą funkcję (rzadko implementowaną), która umożliwia określenie czy rok podany jako argument wywołania (w zakresie 0…99) jest przestępnym czy też nie. Funkcja ta jest wykorzystywana do wyszukiwania imienin w kalendarzu imion, aby nie pominąć 29 lutego. Wracając do tematu funkcji obsługujących zegar RTC warto zauważyć, iż w tym konkretnym rozwiązaniu, rozdzielono mechanizmy dotyczące daty i godziny, gdyż nie zawsze jest celowe operowanie tymi dwoma ustawieniami, w związku z czym nie ma powodu, aby zwiększać ruch na magistrali I²C.

Już zupełnie na koniec dodam, iż zdecydowaną większość pamięci programu aplikacji zajęły wzorce obrazków wyświetlanych w ramach graficznego interfejsu użytkownika, tablica używanych czcionek oraz wbudowany kalendarz imienin, którego stałe znakowe zajmują ponad 10 kB pamięci Flash! Bez tej ostatniej funkcjonalności, program obsługi aplikacji swobodnie zmieściłby się w pamięci programu mikrokontrolera ATmega16A-PU, lecz z uwagi na niewielką różnicę w cenie obu mikrokontrolerów (ok. 4 zł), ostatecznie nie zrezygnowałem z tej niespotykanej opcji.

Montaż

Schemat montażowy komputera pokładowego Mee pokazano na **rysunku 3**. Zbudowano go na dwustronnej płytce drukowanej, z elementów przeznaczonych wyłącznie do montażu przewlekanego, przy czym należy zauważyć, iż elementy te montowane są po obu stronach laminatu. Co bardzo ważne, w przypadku środowisk o sporej liczbie zaburzeń, jakim bez wątpienia jest instalacja samochodowa, zadbano o odpowiednie prowadzenie masy i sygnałów krytycznych.

Montaż rozpoczynamy od wlutowania rezystorów, następnie montujemy kondensatory, elementy mechaniczne takie jak przyciski i złącze goldpin przeznaczone do podłączenia wyświetlacza graficznego, a na samym końcu elementy półprzewodnikowe, które można umieścić w podstawkach. Jako ostatnie należy wlutować gniazdo połączeniowe CON – montujemy je od spodu (*BOTTOM*). Opcjonalnie, aby zmniejszyć odległość wyświetlacza LCD od obwodu drukowanego, kondensatory elektrolityczne C2, C4, C10…C12 mogą być zamontowane od spodu (*BOTTOM*). Na **rysunkach 4** i **5** pokazano wygląd prototypowego obwodu drukowanego zmontowanego komputera pokładowego Mee, bez wyświetlacza LCD.

Podłączenie

Podłączenie sterownika powinno być wykonane przez doświadczonego elektryka bądź elektronika samochodowego przy odłączonym akumulatorze. Urządzenie należy zamontować w suchym miejscu, z dala od urządzeń elektrycznych lub elektronicznych, której działanie mogłoby zakłócić funkcjonowanie sterownika (typu sterownik silnika ECU, moduł kontroli nadwozia BCM czy alarm). Najlepiej, aby komputer był zamontowany w odpowiedniej, najlepiej ekranowanej obudowie chroniącej go przed zwarciem, zawilgoceniem, uszkodzeniem mechanicznym czy zakłóceniami EMI. W celu wykorzystania wszystkich, dostępnych funkcji urządzenia, sterownik należy dołączyć **tylko do dwóch** (co nie jest bez

Wykaz elementów

Rezystory: (0,125 W)
R1, R3: 10 kΩ
R2: 430 Ω/ 0,25 W
R4, R5, R7: 2,2 kΩ
R6, R8, R9: 1 kΩ
R10: 3,3 kΩ
P1: 10 kΩ (pot. montażowy, poziomy)
Kondensatory:
C1, C3, C7…C9, C15…C18: 100 nF (ceram., R=2,54 mm)
C2, C4: 100 µF/16 V (elektrolit., 5/2 mm)
C5, C6, C13, C14: 22 pF (R=2,54 mm)
C10…C12: 470 µF/10 V (6,3/2,54 mm)
Półprzewodniki:
U1: ATmega32A-PU (DIL40)
U2: MCP7940M-I/P (DIL8)
U3: 7805 (TO-220)
U4: 78L05 (TO92)
T1: BC548A (TO92)
D1: BAT85 (DO35)
D2: 1N4148 (DO35)
OK1: LTV817 (DIL4)
Inne:
GLCD: wyświetlacz graficzny LCD-AG-122032G-DIA A/KK-E6 (dystr. Artronic) lub podobny 122×32px, sterownik NJU6450A/SED1520, PCB 80 mm×36 mm
L1: 10 µH (dławik osiowy, miniaturowy R=5 mm)
FT: filtr EMI Murata typu DSS306-55F223 lub podobny
Q1: 4,194 MHz (niski)
Q2: 32768 Hz
PLUS, MINUS, MODE: microswitch z ośką min. 16 mm
CON: złącze męskie, rastrowe 2×5 pin z wycięciem (raster 2,54 mm)

Listing 5. Kluczowe funkcje przeznaczone do obsługi zegara MCP7940

```
void RTCinit(uint8_t Settings)
{
    i2cInit();  //Inicjalizacja magistrali TWI (262kHz)
    i2cStart();
    i2cWriteByte(MCP7940_WRITE_ADDR);  //Adres MCP7940 do zapisu
    i2cWriteByte(CONTROL_REG);  //Adres startowy rejestru ustawień zegara RTC
    i2cWriteByte(Settings);  //Ustawienia zegara RTC
    i2cStop();
}

void RTCwriteTime(timeType *Time)
{
    i2cStart();
    i2cWriteByte(MCP7940_WRITE_ADDR);  //Adres MCP7940 do zapisu
    i2cWriteByte(TIME_START_REG);  //Adres startowy rejestru czasu (w tym przypadku sekund)
    i2cWriteByte(((Time->Second/10)<<4)|(Time->Second%10)|START_OSCILLATOR);  //Sekundy w zapisie BCD + start
oscylatora
    i2cWriteByte(((Time->Minute/10)<<4)|(Time->Minute%10));  //Minuty w zapisie BCD
    i2cWriteByte(((Time->Hour/10)<<4)|(Time->Hour%10));  //Godziny w zapisie BCD (standardowo zapis 24-godzinny)
    i2cStop();
}

void RTCreadTime(timeType *Time)
{
    register uint8_t readByte;
    i2cStart();
    i2cWriteByte(MCP7940_WRITE_ADDR);  //Adres MCP7940 do zapisu
    i2cWriteByte(TIME_START_REG);  //Adres startowy rejestru czasu (w tym przypadku sekund)
    i2cStart();  //I2C Restart
    i2cWriteByte(MCP7940_READ_ADDR);  //Adres MCP7940 do odczytu
    readByte = i2cReadByte(ACK) & 0x7F;  //Sekundy w zapisie BCD - maskujemy bit pracującego oscylatora (bit7)
    Time->Second = ((readByte>>4)*10) + (readByte&0x0F);
    readByte = i2cReadByte(ACK);  //Minuty w zapisie BCD
    Time->Minute = ((readByte>>4)*10) + (readByte&0x0F);
    readByte = i2cReadByte(NACK);  //Godziny w zapisie BCD (standardowo zapis 24-godzinny)
    Time->Hour = (((readByte&0x30)>>4)*10) + (readByte&0x0F);
    i2cStop();
}

void RTCwriteDate(dateType *Date)
{
    i2cStart();
    i2cWriteByte(MCP7940_WRITE_ADDR);  //Adres MCP7940 do zapisu
    i2cWriteByte(DATE_START_REG);  //Adres startowy rejestru daty (w tym przypadku dni tygodnia)
    i2cWriteByte(Date->weekDay);  //Dzień tygodnia
    i2cWriteByte(((Date->Day/10)<<4)|(Date->Day%10));  //Dzień w zapisie BCD
    i2cWriteByte(((Date->Month/10)<<4)|(Date->Month%10));  //Miesiąc w zapisie BCD
    i2cWriteByte(((Date->Year/10)<<4)|(Date->Year%10));  //Rok w zapisie BCD
    i2cStop();
}

void RTCreadDate(dateType *Date)
{
    register uint8_t readByte;
    i2cStart();
    i2cWriteByte(MCP7940_WRITE_ADDR);  //Adres MCP7940 do zapisu
    i2cWriteByte(DATE_START_REG);  //Adres startowy rejestru daty (w tym przypadku dni tygodnia)
    i2cStart();  //I2C Restart
    i2cWriteByte(MCP7940_READ_ADDR);  //Adres MCP7940 do odczytu
    Date->weekDay = i2cReadByte(ACK) & 0x07;  //Dzień tygodnia
    readByte = i2cReadByte(ACK);  //Dzień w zapisie BCD
    Date->Day = ((readByte>>4)*10) + (readByte&0x0F);
    readByte = i2cReadByte(ACK);  //Miesiąc w zapisie BCD, pomijamy bit4, który mówi nam czy rok jest przestępny
    Date->Month = (((readByte&0x10)>>4)*10) + (readByte&0x0F);
    readByte = i2cReadByte(NACK);  //Rok w zapisie BCD
    Date->Year = ((readByte>>4)*10) + (readByte&0x0F);
    i2cStop();
}

void RTCwriteRAM(uint8_t *dataToWrite, uint8_t Bytes)
{
    i2cStart();
    i2cWriteByte(MCP7940_WRITE_ADDR);  //Adres MCP7940 do zapisu
    i2cWriteByte(RAM_START_REG);  //Adres startowy wbudowanej pamięci RAM zegara RTC
    while(Bytes--) i2cWriteByte(*dataToWrite++);
    i2cStop();
}

void RTCreadRAM(uint8_t *dataToRead, uint8_t Bytes)
{
    i2cStart();
    i2cWriteByte(MCP7940_WRITE_ADDR);  //Adres MCP7940 do zapisu
    i2cWriteByte(RAM_START_REG);  //Adres startowy wbudowanej pamięci RAM zegara RTC
    i2cStart();  //I2C Restart
    i2cWriteByte(MCP7940_READ_ADDR);  //Adres MCP7940 do odczytu
    while(Bytes--) *dataToRead++ = i2cReadByte(Bytes? ACK:NACK);
    i2cStop();
}

uint8_t checkYearLeap(uint8_t Year)  //Year: 00...99
{
    if(((Year+2000)%4 == 0 && (Year+2000)%100 != 0) || (Year+2000)%400 == 0) return 1;  //Rok przestępny
    return 0;  //Rok nieprzestępny
}
```

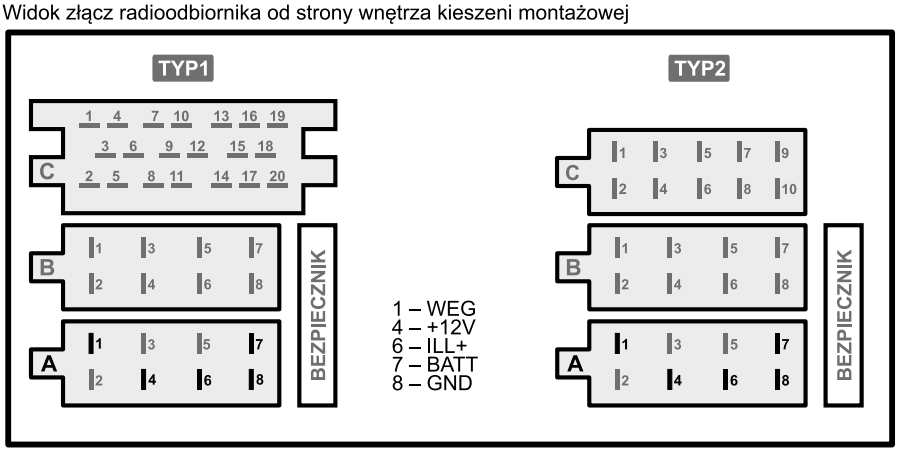
znaczenia dla początkujących elektroników) podzespołów samochodu:

- Złącza radioodbiornika: to podstawowe podłączenie umożliwiające zasilenie układu po włączeniu stacyjki (**+12 V**), podtrzymanie pracy zegara RTC (**BATT**), doprowadzenie sygnału prędkości pojazdu (**WEG**) oraz napięcia do podświetlenia wyświetlacza graficznego (**ILL+**).
- Wtryskiwacza paliwa: to połączenie umożliwia realizację funkcji komputera pokładowego (**INJ+, INJ-**).

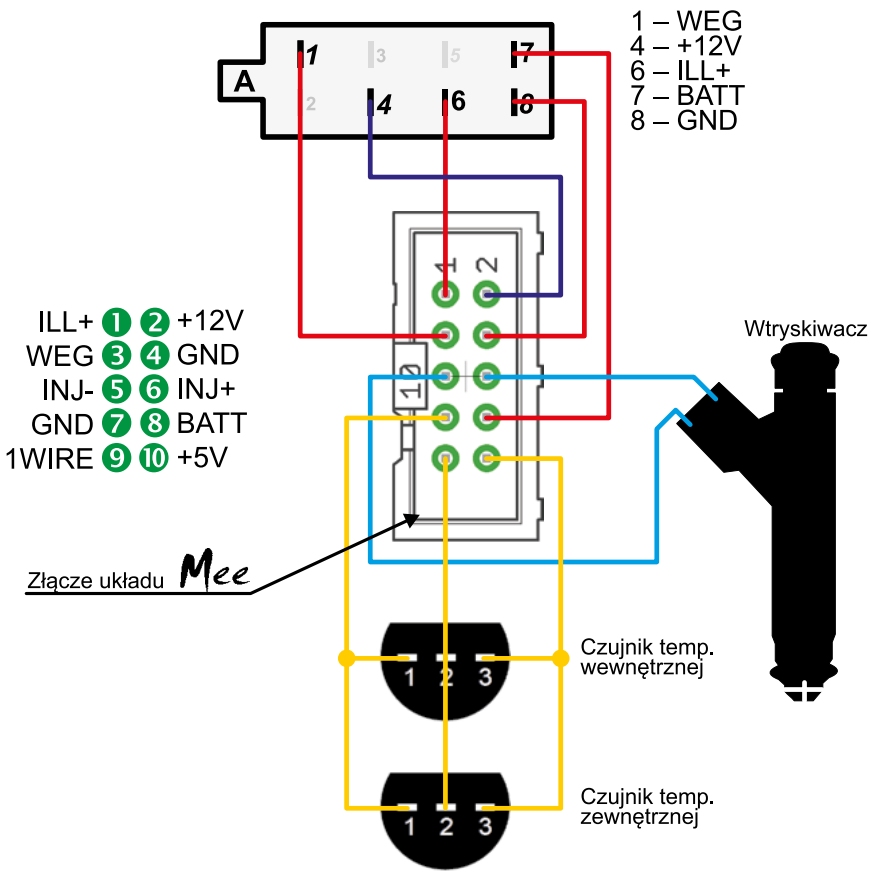
Ponadto, urządzenie Mee wyposażono w możliwość dołączenia dwóch scalonych termometrów cyfrowych typu DS18S20 (lub DS1820), dzięki którym jest możliwy pomiar temperatury wewnątrz i na zewnątrz pojazdu. Termometry te należy dołączyć do złącza CON (jego pinów oznaczonych **GND**, **+5 V** i **1WIRE**) najlepiej za pomocą 3-żyłowego, ekranowanego odcinka przewodu. Należy eksperymentalnie dobrać miejsce zamocowania obu czujników, aby odwzorować średnią temperaturę panującą wewnątrz pojazdu oraz temperaturę zewnętrzną. Dla czujnika temperatury wewnętrznej unikać należy jego montażu w pobliżu nawiewów, drzwi, okien itp. Najlepszym miejscem w tym wypadku, wydaje się być tylna część tunelu środkowego, bądź kieszeń-schowek pod radioodbiornikiem. Z kolei, dla czujnika temperatury zewnętrznej (który to należy zabezpieczyć przed działaniem warunków zewnętrznych, takich jak wilgoć i woda) najlepszym miejscem montażu wydaje się przestrzeń za przednim zderzakiem pojazdu. Wygląd typowych złącz ISO radioodbiornika wraz z zaznaczeniem interesujących nas wyprowadzeń pokazano na **rysunku 6**.

Co ciekawe, w zależności od marki producenta pojazdu, piny o numerach **4** i **7** (oznaczone **+12 V** i **BATT**) mogą być zamienione miejscami w stosunku do pokazanych na **rysunku 6** opisów, w związku z czym ich rzeczywiste funkcje należy ustalić doświadczalnie. A zatem, napięcie na pinie **+12 V** powinno występować wyłącznie po włączeniu zapłonu (przekręceniu kluczyka), zaś na pinie **BATT** permanentnie (tzn. dopóki akumulator podłączony jest do instalacji pojazdu). Napięcie na pinie **ILL+** to zwyczajowo przebieg prostokątny o amplitudzie ok. 12 V i wypełnieniu zależnym od ustawienia pokrętła regulującego jasność podświetlenia zegarów pojazdu (jak i innych akcesoriów, które posiadają elementy podświetlone). Co oczywiste, w naszym wypadku, wykorzystywane jest do regulacji jasności podświetlenia wyświetlacza urządzenia Mee. Jeśli nie zależy nam na takiej funkcjonalności, pin **ILL+** należy połączyć na stałe z pinem **+12 V** złącza **CON.**

Widok złącz radioodbiornika od strony wnętrza kieszeni montażowej



Rysunek 6. Rodzaje złącz radioodbiornika z opisem wyprowadzeń



Rysunek 7. Sposób dołączenia poszczególnych wyprowadzeń złącza CON do instalacji pojazdu

Sposób dołączenia urządzenia do wtryskiwacza paliwa

Połączenie w tym zakresie należy wykonać nader starannie zachowując przy tym dużą ostrożność, by nie doprowadzić do zwarcia przewodów zasilających wtryskiwacz, co mogłoby skutkować uszkodzeniem wyjściowych obwodów sterujących elektronicznego układu sterującego pracą silnika ECU. Każdy wtryskiwacz ma dwa wyprowadzenia. Pierwsze, to stały sygnał +12 V, który zostaje podany po przekręceniu kluczyka stacyjki i który to należy dołączyć do wejścia **INJ+** naszego urządzenia. Drugie, to sygnał sterujący z modułu ECU (komutowana masa), który to z kolei podłączyć należy do wejścia

INJ-. Wspomniane przewody należy starannie zabezpieczyć przed możliwością ewentualnego przetarcia izolacji i powstania zwarcia – dotyczy to zwłaszcza wszelkiego rodzaju otworów przelotowych, przez które zostaną one przeprowadzone.

Aby maksymalnie zabezpieczyć wejściowy układ pomiarowy czasu wtrysku od zaburzeń (np. od będącej zwykle w pobliżu listwy zapłonowej) najlepiej zastosować dwużyłowy przewód ekranowany o odpowiednim przekroju zaś ekran tego przewodu podłączyć po obu stronach do masy pojazdu. Możliwe jest także podłączenie układu formującego impulsy wtryskiwaczy paliwa bezpośrednio do odpowiedniego wyjścia sterownika silnika ECU, na którym to występuje przebieg sterujący pracą wtryskiwaczy. Wtedy należy odpowiednio zmniejszyć wartość rezystora R8 ustalającego prąd diody LED transoptora LTV817.

Aby ułatwić zainstalowanie komputera w pojeździe, po stronie wyprowadzeń obwodu drukowanego, a więc również złącza CON, znalazł się opis poszczególnych jego wyprowadzeń. Sposób dołączenia poszczególnych wyprowadzeń złącza CON do instalacji pojazdu pokazano na **rysunku 7**.

Obsługa

Komputer pokładowy Mee jest urządzeniem, które zwykle będzie obsługiwane podczas jazdy samochodem, więc ergonomia i prostota obsługi oraz czytelność interfejsu użytkownika była podstawowym kryterium przy konstruowaniu stosownych procedur sterujących. Zgodnie z tymi podstawowymi założeniami, na płytce sterownika przewidziano jedynie 3 elementy sterujące (microswitche) umownie oznaczone **PLUS**, **MINUS** i **MODE**). Jak łatwo się domyślić, przyciski **PLUS** i **MINUS** służą do zmiany aktualnie wyświetlanego ekranu Menu lub też do regulacji edytowalnych parametrów pracy urządzenia (jeśli jesteśmy w trybie edycji jakiegoś parametru). Dodatkowo, w większości trybów pracy długotrwałe przytrzymanie przycisków **PLUS** i **MINUS** zwiększa szybkość regulacji wybranego parametru urządzenia. Oprócz wspomnianej powyżej funkcjonalności, przyciski **PLUS** i **MODE** mają dodatkowe funkcje: krótkie przyciśnięcie przycisku **MODE** powoduje zapisanie bieżącego trybu pracy urządzenia w nieulotnej pamięci układu by tryb ten stał się aktywnym trybem pracy po ponownym włączeniu zapłonu, zaś długie przyciśnięcie przycisku **PLUS** powoduje wykasowanie liczników dystansu i średnich wartości obliczeniowych. Nie są to jednak wszystkie funkcje wspomnianych switchy, gdyż ich funkcjonalność zależna jest od aktualnego trybu pracy komputera.

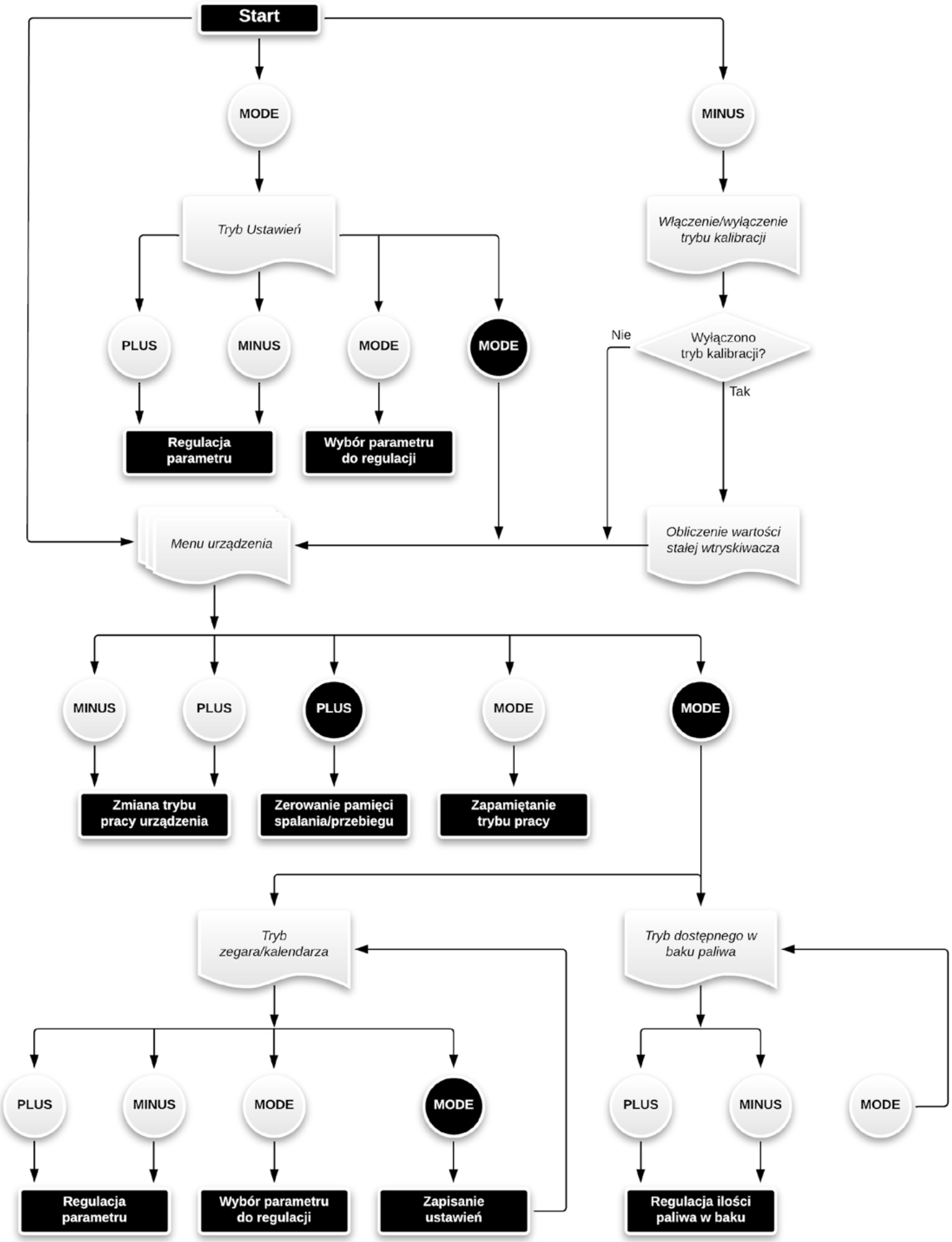
Na **rysunku 8** pokazano wszystkie dostępne tryby pracy sterownika, natomiast na **rysunku 9** pokazano diagram obrazujący strukturę menu jak również sposób obsługi urządzenia (**symbole przycisków wypełnione kolorem czarnym oznaczają długie naciśnięcie wybranego przycisku**). Warto zauważyć, że funkcja pokazywania ilości dostępnego paliwa w baku nie korzysta z sygnału informującego o rzeczywistym poziomie paliwa, gdyż jej implementacja w takim przypadku, skalowanie jak i samo podłączenie sterownika do instalacji pojazdu byłoby dość kłopotliwe. Mechanizm jej działania jest bardzo prosty i zakłada każdorazowe uzupełnianie bieżącego odczytu o ilość zatankowanego paliwa, co jest możliwe poprzez wejście w tryb edycji paliwa dostępnego w baku (długie wciśnięcie przycisku **MODE** w trakcie wyświetlania stosownego menu). Reasumując, po każdym tankowaniu pojazdu należy wejść w edycję dostępnego paliwa i zwiększyć jego ilość (regulacja następuje o pełne litry paliwa). Mechanizm pomiarowy komputera pokładowego będzie następnie odejmował od wartości dostępnego paliwa zużywane paliwo, co umożliwia realizację wspomnianej funkcjonalności.

Rysunek 8. Dostępne tryby pracy komputera pokładowego Mee wraz z opisem wyświetlanych wartości

Nastawy i tryb kalibrowania stałej wtryskiwacza

Komputer pokładowy wyposażono w specjalny tryb pracy służący do konfigurowania. Dzięki niemu możemy określić pewne niezbędne parametry regulacyjne nieodzowne z punktu widzenia funkcjonalności. Ten tryb pracy uruchamiamy poprzez naciśnięcie i przytrzymanie przycisku **MODE** podczas włączania urządzenia (czyli zapłonu pojazdu). W tym trybie pracy urządzenia możemy określić wielkość następujących stałych niezbędnych w procesie obliczania zużycia paliwa, prędkości jazdy oraz drogi:

- **Stała wtryskiwacza** (w [ml/min]): jest to wielkość charakterystyczna dla



Rysunek 9. Diagram obrazujący system Menu i sposób obsługi komputera pokładowego Mee

Ustawienia fuse-bitów:
CKSEL3...0: 1111
CKOPT: 0
SUT1...0: 01
BODEN: 0
BODLEVEL: 1
JTAGEN: 1
EESAVE: 0

Symbol wyświetlacza	Kolor tekstu	Kolor tła
AG-122032G-BIW W/B-E6 PBF		
AG-122032G-DIA A/KK-E6 PBF		
AG-122032G-DIW W/KK-E6 PBF		
AG-122032G-FHA K/A-E6 PBF		
AG-122032G-FHW K/W-E6 PBF		
AG-122032G-YIY Y/G-E6		

Rysunek 10. Wyświetlacze, które można zastosować w aplikacji komputera pokładowego Mee

każdego wtryskiwacza elektronicznego wtrysku paliwa informująca nas o ilości paliwa, jakie może on wprowadzić do komory spalania w jednostce czasu (przy założeniu 100% czasu otwarcia zaworu i stałym, charakterystycznym dla każdego wtryskiwacza ciśnieniu zasilającym).

- **Stała przetwornika drogi** (impulsatora, w [imp/obr]): jest to wielkość charakterystyczna dla każdego impulsatora informująca nas o liczbie impulsów przypadających na 1 obrót koła (osi); dla przykładu w większości nowych modeli marki Opel jest to wartość 29 imp./obr.
- **Obwód opony** (w [cm]).
- **Liczba cylindrów** (a więc i liczba wtryskiwaczy zamontowanych w silniku pojazdu).

Wspomniane stałe można, co prawda znaleźć w Internecie posiłkując się wszelkiego rodzaju forami o tematyce motoryzacyjnej lub elektronicznej, lecz wydaje się, że lepszym sposobem na ich pozyskanie jest wyznaczenie ich na drodze empirycznej. Dla przykładu, obwód opony, a w zasadzie drogę, jaką pokona pojazd w czasie jednego, pełnego obrotu koła, możemy wyznaczyć dość prosto. W tym celu zaznaczamy (np. kredą) najniżej położone miejsce styku opony z powierzchnią drogi, następnie standardowo obciążony pojazd przetaczamy, aby koło wykonało jeden pełny obrót, po czym mierzymy pokonany odcinek drogi. Wszystkie wprowadzone wartości zostaną zachowane w nieulotnej pamięci EEPROM urządzenia, po czym sterownik przejdzie do normalnego trybu pracy. Niestety, jak pokazała praktyka, pewnych trudności może czasami nastręczać znalezienie parametrów stosowanych w naszym pojeździe wtryskiwaczy, jako że elementy te są często wykonywane na zamówienie producenta pojazdu i na próżno szukać ich oznaczeń na stronach producentów stosownych podzespołów. Dlatego przewidziano pewien mechanizm, za pomocą którego komputer jest w stanie samodzielnie wyznaczyć poszukiwaną stałą na podstawie informacji o zużytym paliwie i pomiarze sumarycznego czasu wtrysków. Do tego celu przewidziano specjalny tryb kalibracyjny, który może zostać uruchomiony poprzez naciśnięcie i przytrzymanie przycisku **MINUS** podczas włączania urządzenia, co zostanie zasygnalizowane wyświetleniem okna informacyjnego z komunikatem „**Calibr. ON**”. Ponowne wykonanie wspomnianych czynności (podczas ponownego włączania urządzenia z uruchomionym wcześniej trybem kalibracyjnym) powoduje obliczenie żądanej stałej wtryskiwacza a następnie opuszczenie procesu kalibracji sygnalizowane wyświetleniem okna informacyjnego z napisem „**Calibr. OFF**”. Co oczywiste, do czasu zakończenia procesu kalibracji nie są dostępne następujące wartości obliczeniowe: chwilowe i średnie zużycie paliwa oraz ilość paliwa dostępnego w baku pojazdu (jest to sygnalizowane wyświetleniem napisu „**—**” w odpowiednich polach wspomnianych wartości). Aby przeprowadzenie procesu kalibracji miało w ogóle sens należy zastosować następujący algorytm postępowania:

- Zużyć całe, dostępne paliwo, aż do zaświecenia się lampki sygnalizującej tzw. rezerwę.
- Zatankować **20 litrów** paliwa.
- Uruchomić procedurę kalibracji.
- Zużyć całe, dostępne paliwo (zatankowane wcześniej **20 l**), aż do ponownego zaświecenia się lampki sygnalizującej tzw. rezerwę paliwa.
- Zakończyć procedurę kalibracji.

Po wykonaniu tych czynności komputer obliczy i zapisze, w nieulotnej pamięci EEPROM mikrokontrolera, wartość stałej wtryskiwacza, po czym przejdzie do normalnego trybu pracy. Gdyby obliczona przez sterownik wartość stałej wtryskiwacza powodowała zaniżanie lub zawyżanie rzeczywistego spalania paliwa, w każdej chwili możemy dokonać odpowiedniej jej korekty poprzez wywołanie nastaw i zwiększenie (w przypadku zaniżania spalania) lub zmniejszenie (w przypadku zawyżania spalania) wspomnianej wartości. Należy zaznaczyć, iż tak jak w wypadku oryginalnych rozwiązań typu „komputer pokładowy”, obliczane wartości zużycia paliwa są obarczone pewnym acz niewielkim błędem wynikającym choćby z założenia stałego ciśnienia zasilającego wtryskiwacz czy też z zaokrągleń obliczeniowych, jednak testy praktyczne pokazały, iż maksymalny błąd pomiarowy jest na poziomie 0,5 litra na całą pojemność baku pojazdu, czyli ok. 1% (w tym wypadku było to 45 litrów).

Dla porządku przedstawię typy wyświetlaczy, które to można zastosować w komputerze pokładowym, a których rodzaje – biorąc pod uwagę kolor treści i tła – pokazano na **rysunku 10**. Należy jednak mieć na uwadze, że niektóre z nich (zwłaszcza te, w wykonaniu „negatywowym”) cechuje dość niska częstotliwość odświeżania w przypadku pracy w niskich temperaturach (nawet poniżej 10°C), co powoduje lekkie smużenie przy zmianie wyświetlanej treści. Niestety, jak to w życiu bywa, coś za coś! Dzięki temu uzyskujemy bardzo atrakcyjną cenę urządzenia, ponieważ cena samego wyświetlacza stanowi prawie połowę ceny całego urządzenia!

Robert Wołgajew, EP