TripCo

Komputer samochodowy

AVT 5405

Pomimo coraz bogatszego „uzbrojenia” aut w elektronikę, tematyka związana z różnymi układami poprawiającymi komfort użytkowania samochodu cieszy się ogromnym zainteresowaniem. W artykule opisano uniwersalny komputer samochodowy, który steruje pracą klimatyzacji, pokazuje temperaturę wewnątrz pojazdu, automatycznie włącza światła, wyświetla prędkość, zużycie paliwa itd.

Rekomendacje: *taki komputer przyda się w niejednym, starszym samochodzie, a lektura artykułu może być cenną wskazówką, w jaki sposób otrzymać niezbędne informacje z instalacji samochodowej.*

Tematyka motoryzacyjna jest tematem wielu ciekawych opracowań publikowanych na łamach naszego miesięcznika, ponieważ ta gałąź techniki tworzy ogromną płaszczyznę dla innowacyjnych rozwiązań, dla których granicą jest jedynie pomysłowość – nie wolno też zapominać o homologacji samochodu. Dość ciekawym przykładem tej grupy produktów są układy automatyki wspomagające codzienne użytkowanie pojazdu lub poprawiające komfort, jak np. autonomiczne moduły komputerów pokładowych realizujące typowe funkcje urządzeń fabrycznych. To ciekawe zagadnienie nie umknęło także mojej uwadze, czego przykładem może być kilka projektów ogólnie pojętej automatyki znajdującej zastosowanie w motoryzacji. Ostatnim z tego typu rozwiązań mojego autorstwa był prezentowany na łamach Elektroniki Praktycznej nr 5/2103 sterownik TIDex będący w pełni funkcjonalnym urządzeniem typu komputer pokładowy, a przeznaczonym do pojazdów marki Opel wyposażonych w wyświetlacz systemu audio tzw. TID (*Triple Info Display*). Urządzenie z założenia mogło współpracować wyłącznie z montowanymi we wspomnianych pojazdach wyświetlaczami systemu audio, co ogranicza zastosowanie prezentowanego sterownika tylko w zakresie jednej marki pojazdów. Idąc naprzeciw oczekiwaniom Czytelników oraz – nie ukrywam – zaspokajając własną ciekawość, postanowiłem zbudować urządzenie o zbliżonej funkcjonalności, lecz wyposażone w autonomiczny, efektowny interfejs użytkownika. Jak poprzednio, sterownik TripCo realizuje następujące funkcje:

- Steruje pracą układu klimatyzacji manualnej utrzymując zadaną temperaturę wewnątrz pojazdu poprzez automatyczny dobór prędkości pracy wentylatora dmuchawy i/lub włączanie/wyłączanie sprężarki układu klimatyzacji.
- Pokazuje temperaturę wewnątrz pojazdu.
- Zapewnia funkcję automatycznego włącznika świateł mijania po osiągnięciu przez pojazd prędkości > 5 km/godz.
- Pokazuje chwilową prędkość pojazdu (w km/godz.).
- Pokazuje średnią prędkość pojazdu na zadanym odcinku drogi (w km/godz.).
- Pokazuje maksymalną prędkość pojazdu na zadanym odcinku drogi (w km/godz.).
- Pokazuje chwilowe zużycie paliwa (w l/godz. dla prędkości ≤ 5 km/godz. oraz w l/100 km dla pozostałych prędkości).
- Pokazuje średnie zużycie paliwa (w l/100 km).
- Pokazuje całkowite zużycie paliwa (w l).
- Pokazuje koszt zużytego paliwa.
- Pokazuje przejechany dystans od ostatniego kasowania (w km).
- Pokazuje liczbę uruchomień zapłonu.

Algorytm działania urządzenia w zakresie dostępnej funkcjonalności w zasadzie nie uległ zmianom, nie będzie tutaj omawiany, natomiast dociekliwych Czytelników zachęcam do lektury artykułu na temat układu TIDex. Tym razem uwagę swoją skupiłem

W ofercie AVT*
AVT-5405 A

Podstawowe informacje:
- Napięcie zasilania: 12…15 V DC.
- Średni prąd obciążenia (przekaźniki wyłączone/załączone): 20 mA/100 mA.
- Zakres regulacji temperatury: 10…30°C.
- Skok regulacji: 1°C.
- Histereza regulacji: +1,5°C dla trybu schładzania, -1,5°C dla trybu ogrzewania.
- Dokładność pomiaru temperatury: 0,5°C.
- Zakres pomiarowy temperatury: 0…99,5°C.
- Zakres pomiarowy prędkości: 0…255 km/godz.
- Zakres pomiarowy chwilowego zużycia paliwa: 0…99,9 l/100 km.
- Zakres pomiarowy średniego zużycia paliwa: 0…25,5 l/100 km.
- Zakres pomiarowy zużytego paliwa: 0…999,9 l.
- Zakres pomiarowy przejechanego dystansu: 0…9999 km.
- Zakresy regulacji parametrów konfiguracyjnych: stała wtryskiwacza: 1…999 ml/min, stała przetwornika drogi: 1…99 imp./obr., obwód opony: 1…255 cm.

Dodatkowe informacje:
Wideo obrazujące obsługę urządzenia: *http://www.youtube.com/watch?v=V9lTNW922lw&feature=youtu.be*

Dodatkowe materiały na CD/FTP:
ftp://ep.com.pl, user: 52617, pass: 30lct328
- wzory płytek PCB
- karty katalogowe i noty aplikacyjne elementów oznaczonych w Wykazie elementów kolorem czerwonym

Projekty pokrewne na CD/FTP:
(wymienione artykuły są w całości dostępne na CD)

AVT-5395	Komputer samochodowy TIDex (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)
---	Obrotomierz cyfrowo-analogowy (EdW 6/2010)
AVT-2799	Mikroprocesorowy obrotomierz stroboskopowy (EdW 9/2006)
AVT-434	Komputer samochodowy (EP 9-10/2005)

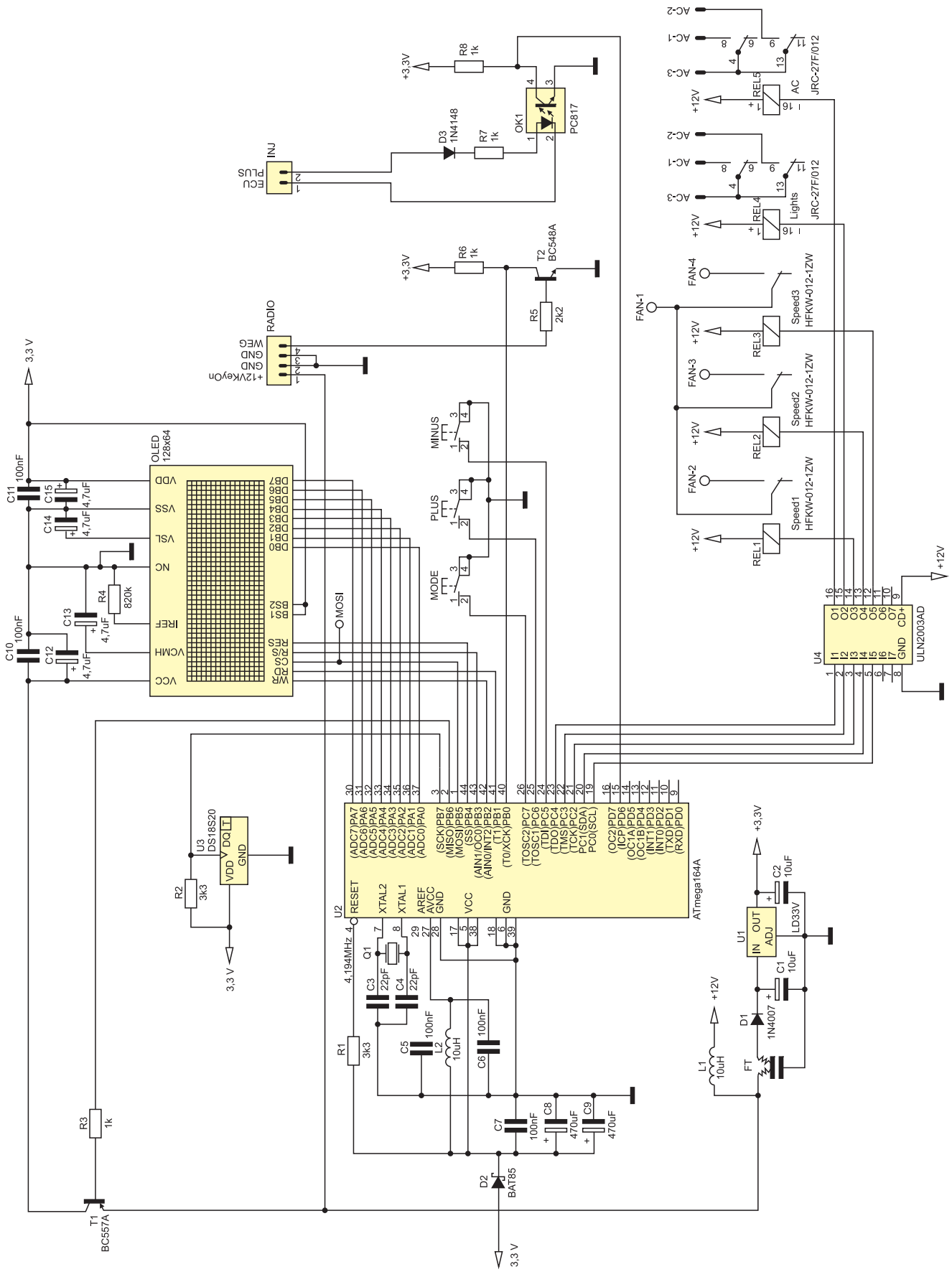
* Uwaga:
Zestawy AVT mogą występować w następujących wersjach:

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**

na dobraniu optymalnego z punktu widzenia ergonomii obsługi, sposobu interakcji urządzenia z użytkownikiem, na co składał się rodzaj zastosowanego wyświetlacza jak i sposób obsługi sterownika. Proces ten okazał się niezbyt zajmującym wyzwaniem, gdyż moje wcześniejsze doświadczenia z panelami OLED marki Winstar szybko wyłoniły przysłowiowego zwycięzcę. Jak łatwo się domyślić i tym razem postawiłem na graficzny wyświetlacz OLED pracujący w 16 poziomach szarości. Wybór ten podyktowany był następującymi cechami użytkowymi:

- optymalna wielkość przekątnej ekranu 2.7”,



Rysunek 1. Schemat ideowy układu TripCo

Tabela 1. Konfiguracja i funkcjonalność timerów mikrokontrolera ATmega164A

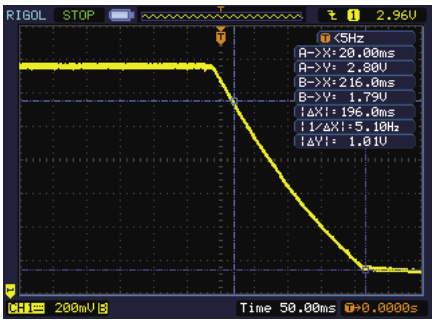
Nazwa	Konfiguracja	Funkcjonalność
Timer0	Licznik impulsów zewnętrznych podawanych na wejście T0 – zlicza przy zboczu opadającym.	Liczy impulsy przetwornika drogi WEG
Timer1	Taktowany wewnętrznym sygnałem zegarowym, preskaler=64 (65.536 impulsów na 1 ms). Wejście Input Capture podłączone do układu formującego przebieg z wtryskiwacza paliwa.	Mierzy sumaryczny czas wtrysku w czasie 1 sekundy
Timer2	Taktowany wewnętrznym sygnałem zegarowym, preskaler=1024. Tryb Normalny. Przerwanie co 62.5 ms.	Odmierza czas pomiaru równy 1 s.

- wystarczająca rozdzielczość 128×64px,
- możliwość wyświetlania 16 poziomów szarości (a zasadzie ustalania poziomów intensywności koloru wyświetlacza), co poprawia estetykę pokazywanych piktogramów,
- doskonały i nieosiągalny dla paneli LCD kontrast (ponad 2000:1),
- duża jasność (ok. 80 cd/m²),
- doskonała czerń (matryca OLED sama jest źródłem światła),
- bardzo szeroki kąt widzenia w każdej płaszczyźnie wynoszący ponad 160 stopni,
- dostępność kilku wersji kolorystycznych panela (zielony, żółty, czerwony, niebieski, biały),
- znacznie poprawiony czas życia matrycy (ponad 100 tys. godzin),
- szeroki zakres temperatury pracy (-40… +85 ºC),
- łatwość implementacji (zastosowany sterownik SSD1325 posiada akcelerację funkcji graficznych),
- niewygórowana cena.

Jak widać, powyższe cechy użytkowe idealnie pozycjonują wspomniany wyświetlacz do zastosowań w branży motoryzacyjnej i nie tylko. Jest to wszak technologia optymalna wszędzie tam, gdzie jest wymagany obraz o doskonałej czytelności przy zachowaniu obniżonego poboru energii.

Przejdźmy, zatem do schematu sterownika, który pokazano na **rysunku 1**. Jak widać, jest nieskomplikowany układ mikroprocesorowy zbudowany z wykorzystaniem popularnego mikrokontrolera firmy Atmel typu ATmega164A, scalonego termometru DS18S20, drivera mocy ULN2003 sterującego przekaźnikami wykonawczymi dużej mocy oraz kilku innych elementów dyskretnych. Ponadto, na płytce układu zabudowano kompletny zasilacz układu sterownika wyposażony w scalony stabilizator napięcia 3,3 V, szeregowo-równoległy filtr EMI firmy Murata oraz komplet wygodnych złącz pozwalających na bezproblemowe połączenie z instalacją elektryczną pojazdu. Należy zwrócić uwagę na fakt, że przekaźniki sterujące załączaniem układu klimatyzacji manualnej oraz wyłącznika świateł mają dwa komplety styków, a wynika to z faktu, że wyłączniki odpowiadających im obwodów, które to styki „bocznikują", wyposażone są także w więcej niż jeden styk wykonawczy i aby poprawnie załączyć wybrany układ, należy dobrać przekaźnik o takim samym układzie styków.

Serce sterownika stanowi mikrokontroler. Realizuje on całą, założoną funkcjonalność urządzenia posiłkując się w tym celu szeregiem wewnętrznych modułów peryferyjnych. W odróżnieniu jednak od prezentowanego wcześniej sterownika TIDex, w tym wypadku zastosowano mikrokontroler o 2-krotnie większej pamięci Flash, co zostało podyktowane koniecznością przechowywania w niej wzorców zastosowanych czcionek oraz wyświetlanych piktogramów w formacie bitmap. Mikrokontroler jest taktowany za pomocą zewnętrznego rezonatora kwarcowego o częstotliwości 4,194 MHz, co zapewnia dużą dokładność pomiarów czasu, wykonywanych w programie obsługi urządzenia. Niska częstotliwość zegarowa (w porównaniu do poprzedniego rozwiązania) wynika z konieczności zapewnienia poprawnej pracy mikrokontrolera przy bardzo napięciu zasilania rzędu 1,8 V, co zostanie wyjaśnione w dalszej części artykułu. W programie obsługi sterownika wykorzystano wszystkie wbudowane timery, a ich konfigurację oraz realizowaną funkcjonalność opisano w **tabeli 1**.



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

W celu pomiaru czasu trwania wtrysków, zbudowano kompletny i bezpieczny układ wejściowy (przy użyciu popularnego optoizolatora PC817) formujący sygnał wtryskiwacza dla potrzeb wejściowych obwodów mikrokontrolera. Sygnał ten jest podawany na wejście przechwytujące układu czasowo – licznikowego Timer1, dzięki czemu w dość prosty sposób zrealizowano pomiar czasu trwania wtrysku (akumulowany jest sumaryczny czas trwania wtrysku w okresie 1 sekundy). W ramce umieszczono wzory służące do obliczenia ważniejszych danych dla ustawień Timerów, jak w tabeli 1.

W sterowniku TripCo zastosowano specjalny układ podtrzymania napięcia zasilania mikrokontrolera, gwarantujący poprawność funkcjonowania akumulatorów obliczeniowych pomimo wyłączania zasilania urządzenia. Dane „zebrane" w specjalną strukturę zaopatrzoną w sumę kontrolną CRC8 zapisywane są każdorazowo przy wyłączaniu zapłonu. W celu detekcji momentu wyłączania zapłonu zastosowano wbudowany w mikrokontroler przetwornik A/C pracujący w trybie *Free Running* i mierzący napię-

Wzory służące do obliczania ważniejszych danych:

```
Accu.Petrol += ((1UL*INJECTORS*InjectionTime*Config.CcPerMin)/ 3932UL);  //Akumulator zużytego paliwa [ul]
Accu.Distance += ((1UL*WEGpulses*Config.Wheel) / (100UL*Config.PulsPerRot));  //Akumulator dystansu [m]
Consum = ((3UL*INJECTORS*InjectionTime*Config.CcPerMin) / 327680UL);  //Chwilowe zużycie paliwa [l/h *10], gdy prędkość ≤ 5 km/h
Consum = ((100UL*Consum) / Speed);  //Jeśli prędkość>5 km/h to zużycie chwilowe podajemy w l/100km *10
ConsumAvg = (Accu.Petrol/Accu.Distance);  //Średnie zużycie paliwa [l/100km *10]
Speed = ((36UL*WEGpulses*Config.Wheel) / (1000UL*Config.PulsPerRot)); //Chwilowa prędkość [km/h]
SpeedAvg = ((36UL*Accu.Distance)/(10UL*Accu.Measurements));  //Średnia prędkość [km/h]
```

gdzie:
- INJECTORS – liczba wtryskiwaczy
- InjectionTime – sumaryczny czas wtrysku [ms/40]
- Config.CcPerMin – stała wtryskiwacza [ml/min]
- Config.PulsPerRot – stała przetwornika drogi [imp/obr]
- Accu.Petrol – akumulator zużycia paliwa [μl]
- Accu.Distance – akumulator przejechanego dystansu [m]
- WEGpulses – liczba impulsów z przetwornika drogi zliczona w czasie 1 sekundy
- Config.Wheel – obwód opony [cm]
- Accu.Measurements – liczba pomiarów (co 1 sekundę)

cie *Vbak,* czyli napięcie zasilające wyłącznie mikrokontroler. Po zaniku zasilania dioda D2 umożliwia separację zasilania mikrokontrolera od reszty urządzenia, a kondensatory C8…C9 zapewniają odpowiedni czas podtrzymania zasilania. Co ciekawe, na pierwszy „rzut oka" nie wydaje się by nasz sterownik w jakikolwiek sposób używał przetwornika A/C, ponieważ żaden z kanałów wejściowych nie jest przez niego w tym celu używany. Jest jednak inaczej. Przetwornik A/C mierzy specjalne, wewnętrzne napięcie odniesienia V_{BG}=1,1 V dzięki wbudowanemu multiplekserowi analogowemu. Napięciem odniesienia jest napięcie zasilające mikrokontroler, czyli napięcie dostarczane na wyprowadzenie AVCC (czyli nasze *Vbak*). Jego spadek podczas wyłączania zapłonu, powoduje wzrost wartości wyniku przetwarzania według wzoru jak niżej (korzystamy z 8-bitowej rozdzielczości przetwornika) $V_{ADC}=(V_{BG}*256)/Vbak$.

Procedura obsługi przerwania przetwornika ADC sprawdza każdorazowo 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 resetowania mikrokontrolera, które to dokonywane jest przez uruchomiony wcześniej układ BOD (typowo przy wartości 1.8 V). Wspomniany próg zadziałania ustawiono na wartość 2.8 V, co oznacza, iż czas opadania napięcia zasilającego od wartości 2.8 V do wartości 1.8 V (resetowania mikrokontrolera) jest czasem, w którym musi on przeprowadzić zapis wszystkich danych krytycznych – w naszym przypadku 20 bajtów danych. Jak pokazały testy praktyczne, zastosowanie wspomnianego wcześniej rozwiązania sprzętowego (dioda D2 i kondensatory C8…C9) 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**.

Jak widać, czas dostępny na zapis danych wynosi około 196ms co dla wymaganej maksymalnej wartości rzędu 66 ms (20 bajtów po 3.3 ms na zapis jednego bajta) daje spory margines bezpieczeństwa. Co oczywiste, potencjalne uszkodzenie w bloku sprzętowym odpowiedzialnym za podtrzymanie napięcia zasilającego skutkować będzie błędnymi zapisami pamięci EEPROM, gdyż dołączana (do struktury danych) suma kontrolna nie będzie zgodna z obliczoną. W takim przypadku układ TripCo poinformuje użytkownika wyświetlając okno informacyjne z komunikatem *„save error"* tuż po włączeniu zasilania (dokładnie w czasie sprawdzania integralności danych zachowanych w pamięci EEPROM. Listing konfiguracji przetwornika ADC oraz procedury obsługi jego przerwania pokazano na **listingu 1**.

Listing 1. Konfiguracja oraz procedury obsługi przerwania przetwornika A/C

```
#define VBG 11  //Voltage BandGap, 11=1.1V itd
#define CRITICAL_VOLTAGE 28  //28=2.8V itd
//Obliczony próg dla zapisu do EEPROMa (dla rozdz. 8bitów)
#define TRESHOLD (VBG*256/CRITICAL_VOLTAGE)

/* Vref=AVCC, Vin=VBG (wewn.źródło 1.1V), justowanie wyniku pomiaru do lewej
(ADHC zawiera 8-bitowy wynik pomiaru)
Uruchomienie przetwornika ADC w trybie Free Running i zezwolenie na przerwanie
po każdej konwersji (preskaler=128) */
ADMUX = (1<<REFS0)|(1<<MUX4)|(1<<MUX3)|(1<<MUX2)|(1<<MUX1)|(1<<ADLAR);
ADCSRA = (1<<ADEN)|(1<<ADSC)|(1<<ADATE)|(1<<ADPS2)|(1<<ADPS1)|(1<<ADPS0);

ISR(ADC_vect)
{
  register uint8_t Voltage = ADCH;
  if(Voltage>TRESHOLD)
  {
    OLEDoff();  //Wyłączenie software'owe panelu OLED
    //Zapis krytycznych danych do pamięci EEPROM - 66ms
    eeprom_write_block(&Accu, &AccuEE, sizeof(Accu));
    _delay_ms(40);
    //Wyłączenie zasilania panela OLED (tranzystor sterujący T1)
    PORTB |= (1<<PB6);
    //Czekamy na restart mikrokontrolera przez układ BOD
    _delay_ms(2000);
  }
}
```

Kończąc artykuł poświęcony rozwiązaniom konstrukcyjnym oraz programowym zastosowanym w układzie TripCo chciałbym na chwilę zatrzymać się nad zagadnieniem obsługi panela OLED w zakresie wyświetlania grafik oraz własnych czcionek. Co prawda, budowę i sposób obsługi wyświetlaczy tego typu (w języku Bascom) opisałem w Elektronice Praktycznej 01/2011 i 02/2011, jednak od tego czasu firma Winstar zmodyfikowała produkowane wyświetlacze OLED, co wymusiło zmiany w procedurze inicjalizacji. Ponadto, przedstawione niżej rozwiązania zoptymalizowano pod kątem języka C i nowych narzędzi do generowania obiektów graficznych, przez co zdecydowanie zwiększono możliwości w zakresie tworzenia graficznych interfejsów użytkownika. Nie wchodząc w takim razie w szczegóły budowy i funkcjonowania wyświetlacza, gdyż jak wspomniałem obszerną wiedzę w tym zakresie można znaleźć w moich poprzednich artykułach, przypomnę kilka podstawowych informacji na temat sposobu obsługi i inicjalizacji takiego rodzaju mediów.

Sterowanie pracą wyświetlaczy tego typu polega na wysyłaniu komend sterujących, za pomocą których ustawiamy wstępne parametry konfiguracyjne oraz na zapisie lub odczycie danych do/z pamięci wideo. Zapis powoduje natychmiastowe wyświetlenie pikseli o danych atrybutach.

Interpretacja rodzaju danych, które mają być odebrane przez chipset SSD1325, jest zdeterminowana stanem wyprowadzenia RS. Jest to typowe rozwiązanie w sterowaniu wyświetlaczami wszelkiego typu. Poziom niski na tym wyprowadzeniu decyduje o tym, iż przesyłana wartość zostanie potraktowana jako komenda sterująca lub też argument przesłanej wcześniej komendy sterującej, natomiast poziom wysoki spowoduje, iż dana zostanie zinterpretowana jako dana pamięci

Listing 2. Listing funkcji zapisu komendy sterującej (lub argumentu tej komendy)

```
static void WriteCmnd(uint8_t Command)
{
  RESET_RS;
  RESET_WR;
  RESET_CS;
  DATA_PORT = Command;
  SET_CS;
  SET_WR;
  SET_RS;
}
```

Listing 3. Listing funkcji zapisu danej pamięci obrazu GDDRAM

```
static void WriteGDDram(uint8_t Data)
{
  RESET_WR;
  RESET_CS;
  DATA_PORT = Data;
  SET_CS;
  SET_WR;
}
```

wideo GDDRAM. W związku z powyższą organizacją procedur sterujących można wyróżnić dwa rodzaje sekwencji operacji zapisu do układu SSD1325 (operacje odczytu przebiegają analogicznie, lecz nie będą tematem naszych rozważań):

- sekwencja zapisu komendy sterującej (zapisu danych do rejestrów konfiguracyjnych),
- sekwencja zapisu danych do pamięci obrazu GDDRAM.

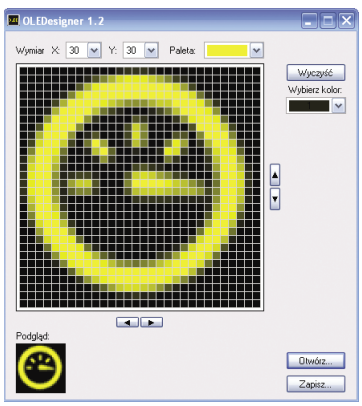
Wspomnianym sekwencjom sterującym odpowiadają funkcje, które pokazano na **listingu 1** oraz **listingu 2**. Posiłkując się tymi dwoma „elementarnymi" funkcjami umożliwiającymi komunikację ze sterownikiem panelu, możemy przejść do funkcji służącej do inicjalizacji sterownika SSD1325 po włączeniu zasilania wyświetlacza, którą to zamieszczono na **listingu 4**, zaś odpowiednie definicje na **listingu 5**. W tym momencie, potrafimy już zainicjować interesujący nas panel OLED w związku, z czym przejdziemy do opisu funkcji odpowiedzialnych za wyświetlenie przygotowanej wcześniej grafiki jak i czcionek ekranowych, których w urządzeniu TripCo zastosowano aż 3 rodzaje. Ste-

rownik SSD1325 ma dość nietypową organizację pamięci ekranu, w której każdy bajt tej pamięci przechowuje dane dla 2 kolejnych pikseli obrazu (stąd jest możliwe wyświetlenie 16 poziomów jasności każdego piksela), do wygenerowania odpowiednich tablic wykorzystano program OLEDesigner v. 1.2, której autorem jest Marcin Popławski. Wygląd okna najnowszej wersji programu OLED Designer przedstawiono na **rysunku 3**. Za pomocą tego prostego, ale jakże użytecznego narzędzia, jesteśmy w stanie zaprojektować dowolny element przyszłego interfejsu użytkownika. Co więcej, bieżąca wersja programu obsługuje następujące formaty danych:

- OPF (od Oled Picture File) czyli plik binarny o odpowiedniej strukturze przeznaczony do zastosowań w kompilatorze Bascom,
- BMP o dowolnej palecie barw, gdyż program automatycznie przekształci taki obrazek do trybu monochromatycznego (oraz „przytnie" do formatu 128×64, jeśli rozmiar obrazka będzie nieodpowiedni),
- C z tablicą wzorca obrazka zapisaną przy użyciu kompresji mRLE (zmodyfikowany algorytm RLE),
- C z tablicą wzorca obrazka zapisaną przy użyciu typowej kompresji RLE.

Otrzymany w ten sposób obraz możemy edytować i zapisywać lub odczytywać korzystając z dowolnego z dostępnych formatów danych. Edytor jest intuicyjny i łatwy w obsłudze. Po wyborze z rozwijanej listy odpowiedniego koloru i wciśnięciu lewego przycisku myszy, nanosimy wybrany kolor na obraz. W celu łatwiejszej identyfikacji kolory w palecie są ponumerowane. Pod prawym przyciskiem myszy kryje się gumka, za pomocą której usuwamy niepotrzebny kolor. Tworzone przez program Pliki OPF jak i C mają następującą strukturę:

- pierwszy bajt określa szerokość obrazka,
- drugi bajt określa wysokość obrazka,
- pozostałe bajty to dane kolejnych pikseli obrazu poddane prostemu algorytmowi kompresji, przy czym zgodnie z organizacją pamięci sterownika SSD1325, każdy wspomniany bajt przechowuje informację o dwóch, kolejnych punktach obrazu.



Rysunek 3. Wygląd programu OLEDesigner (v.1.2)

Jak pokazały testy praktyczne, najlepszy stopień kompresji obrazków przechowywanych w ten sposób daje zmodyfikowany algorytm RLE (nazwany tutaj mRLE), którego sposób działania opisano w **tabeli 2**.

Pomimo swojej prostoty zapewnia on całkiem spory stopień kompresji, co przekłada się na zmniejszenie zajętości pamię-

Listing 4. Listing funkcji inicjalizacji sterownika SSD1325 i włączenia panelu OLED

```
void InitOLED(void)
{
  //Port danych OLEDa jako wyjscia ze stanem „0"
  DATA_DDR = 0xFF;
  //Porty sterujące OLEDa jako wyjscia ze stanem „0"
  CONTROL_DDR |= (1<<WR_PIN) | (1<<RD_PIN) | (1<<CS_PIN) | (1<<RS_PIN) |
(1<<RST_PIN);
  //Port sterujący zasilaniem OLEDa jako wyjscie ze stanem „1" czyli zasilanie
WYŁĄCZONE
  VCC_PORT |= (1<<VCC_PIN);
  VCC_DDR |= (1<<VCC_PIN);
  RESET_RST;  //Zerowanie sprzętowe sterownika SSD1325
  _delay_ms(1);
  SET_RST;
  //Konfiguracja wszystkich parametrów sprzętowych sterownika SSD1325
  WriteCmnd(SET_COLUMN_ADDR);  //0...63 (Pojedynczy adres przechowuje dane 2
pixeli obrazu)
  WriteCmnd(0x00);
  WriteCmnd(0x3f);
  WriteCmnd(SET_ROW_ADDR);  //0...63
  WriteCmnd(0x00);
  WriteCmnd(0x3f);
  WriteCmnd(SET_CONTRAST_CTRL);
  WriteCmnd(0x7f);  //Contrast Current: Maximum=127, ewentualnie 0x40
  WriteCmnd(SET_CURRENT_RANGE | FULL_CURRENT);
  WriteCmnd(SET_DISP_OFFSET);
  WriteCmnd(0x4c);  //Vertical Scroll=76
  WriteCmnd(SET_DISP_START_LINE);
  WriteCmnd(0x00);
  WriteCmnd(SET_REMAP);
  WriteCmnd(0x52);  //Enable Nibble Re-map, Enable COM Re-map, Enable COM
Split Odd Even
  WriteCmnd(SET_MASTER_CONFIG);
  WriteCmnd(0x02);  //External Vcc power supply
  WriteCmnd(SET_DISP_MODE_NORMAL);
  WriteCmnd(SET_MULTIPLEX_RATIO);
  WriteCmnd(0x3f);  //Mux=64
  WriteCmnd(SET_PHASE_LENGTH);
  WriteCmnd(0x55);  //Phase 1 & Phase 2 period = 5 DCLKs
  WriteCmnd(SET_ROW_PERIOD);
  WriteCmnd(0x51);  //R=81 DCLKs
  WriteCmnd(SET_DISP_CLOCK);
  WriteCmnd(0x91);  //Fosc=800kHz, Div=2, Fdlck=400kHz
  WriteCmnd(SET_VCOMH_VOLTAGE);
  WriteCmnd(0x1C);  //0.8*VREF
  WriteCmnd(SET_PRECHARGE_VOLTAGE);
  WriteCmnd(0x10);  //0.67*VREF
  WriteCmnd(SET_GRAY_SCALE_TABLE);
  WriteCmnd(0x00);
  WriteCmnd(0x11);
  WriteCmnd(0x21);
  WriteCmnd(0x32);
  WriteCmnd(0x43);
  WriteCmnd(0x54);
  WriteCmnd(0x65);
  WriteCmnd(0x76);
  ClsOLED();  //Wyczyszczenie pamięci ekranu
  OLEDon(); //Załączenie panela OLED
}
```

Listing 5. Definicje rozkazów sterujących sterownika SSD1325

```
#define SET_COLUMN_ADDR 0x15
#define SET_ROW_ADDR 0x75
#define SET_CONTRAST_CTRL 0x81
#define SET_CURRENT_RANGE 0x84
#define FULL_CURRENT 0x02
#define HALF_CURRENT 0x01
#define QUARTER_CURRENT 0x00
#define SET_REMAP 0xA0
#define SET_DISP_START_LINE 0xA1
#define SET_DISP_OFFSET 0xA2
#define SET_DISP_MODE_NORMAL 0xA4
#define SET_DISP_MODE_INVERSE 0xA7
#define SET_MULTIPLEX_RATIO 0xA8
#define SET_MASTER_CONFIG 0xAD
#define SET_PRECHARGE_COMP_ENABLE 0xB0
#define SET_PHASE_LENGTH 0xB1
#define SET_ROW_PERIOD 0xB2
#define SET_DISP_CLOCK 0xB3
#define SET_PRECHARGE_COMP_LEVEL 0xB4
#define SET_GRAY_SCALE_TABLE 0xB8
#define SET_PRECHARGE_VOLTAGE 0xBC
#define SET_VCOMH_VOLTAGE 0xBE
#define SET_SEGMENT_LOW_VOLTAGE 0xBF
#define GRAPHIC_ACCEL_OPTION 0x23
#define ENABLE_FILLING 0x01
#define DRAW_RECTANGLE 0x24
#define SET_DISPLAY_ON 0xAF
#define SET_DISPLAY_OFF 0xAE
```

ci Flash mikrokontrolera, przechowującej wzorce obrazków i czcionek. Na **listingu 6** przedstawiono funkcję odpowiedzialną za odczytanie, dekompresję i wyświetlenie tak zapisanego obrazka.

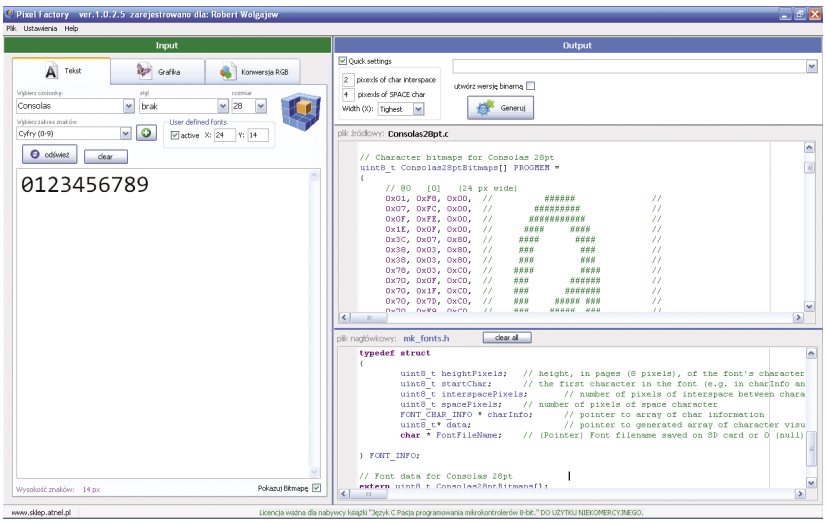
Na koniec tej tematyki chciałbym pokrótce omówić zagadnienie generowania i wyświetlania czcionek. Do utworzenia stosownych tablic i struktur zawierających używane w programie obsługi czcionki zastosowano doskonały program firmy Atnel (*www.atnel.pl*) o nazwie PixelFactory, którego autorem jest Mirosław Kardaś. Wygląd głównego okna programu pokazano na **rysunku 4**. Oprócz innych funkcji, ten program umożliwia utworzenie wzorców czcionek na podstawie wybranych przez użytkownika fontów dostępnych w systemie Windows, wybór wielkości i zakresu dostępnych znaków. Wynikiem działania programu są dwa pliki: plik C zawierający tablice wzorców wygenerowanych znaków i 2 struktury opisujące ich parametry fizyczne oraz plik H zawierający deklaracje używanych tablic, struktur i typów danych. Nie będę w tym miejscu rozpisywał się na temat obsługi programu PixelFactory, gdyż jest ona intuicyjna, a stosowne artykuły znaleźć można w Internecie, lecz skupię się na implementacji odpowiednich procedur dla wyświetlaczy OLED zaznaczając przt tym, iż w naszym programie obsługi zmodyfikujemy nieco sposób dostępu do wzorców znaków, ponieważ nie jest nam potrzebna tak duża elastyczność użytkowa, zaimplementowana przez autora programu.

Tabela 2. Sposób działania algorytmu kompresującego dane obrazków

Liczba powtórzeń	Dane przed kompresją	Dane po kompresji
1	0xAA	0xAA
2	0xAA 0xAA	0xAA 0xAA 0x00
3	0xAA 0xAA 0xAA	0xAA 0xAA 0x01
4	0xAA 0xAA 0xAA 0xAA	0xAA 0xAA 0x02
5	0xAA 0xAA 0xAA 0xAA 0xAA	0xAA 0xAA 0x03
. . .	. . .	. . .
257	257 x 0xAA	0xAA 0xAA 0xFF

Listing 6. Funkcja odpowiedzialna za odczytanie, dekompresję i wyświetlenie obrazka

```
//Bajty Picture[0] i Picture[1] opisują szerokość i wysokość obrazka,
pozostale bajty stanowią treść obrazu: 1 bajt na 2 pixele
void DrawPicture(uint8_t X1, uint8_t Y1, const uint8_t *Picture)
{
  register uint8_t byteA, byteB;
  uint16_t bytesToSend, repeats, i=2;
  //Obliczamy liczbę bajtów jakie należy wysłać do sterownika modułu OLED
  bytesToSend = (pgm_read_byte(&Picture[0])>>1)*pgm_read_byte(&Picture[1]);
  //Ustawiamy aktywne okno pamięci obrazu sterownika SSD1325 by uprościć zapis
danych
  SetActiveWindow(X1, Y1, X1+pgm_read_byte(&Picture[0])-1, Y1+pgm_read_
byte(&Picture[1])-1);
  do
  {
    if(bytesToSend == 1)  //Dla ostatniego bajta danych
    {
      WriteGDDram(pgm_read_byte(&Picture[i]));
      bytesToSend--;
    }
    else
    {
      byteA = pgm_read_byte(&Picture[i]);  //Odczytujemy bajt "i"
      byteB = pgm_read_byte(&Picture[i+1]);  //Odczytujemy bajt "i+1"
      if( byteA != byteB)  //Sprawdzamy czy kolejne bajty są różne
      {
        WriteGDDram(byteA);
        bytesToSend--;
        i++;
      }
      else
/* Jeśli kolejne bajty sa takie same to ustalamy liczbę powtórzeń danego bajta
(+2 powtórzenia obligatoryjne) wynikające ze sposobu zapisu kompresji typu
mRLE (zmodyfikowane RLE) */
      {
        repeats = pgm_read_byte(&Picture[i+2])+2;
        while(repeats--) {WriteGDDram(byteA); bytesToSend--;} //Wysyłamy
liczbę powtórzeń bajta
        i +=3;  //Przesuwamy indeks o 3 miejsca
      }
    }
  } while(bytesToSend);
  SetActiveWindow(0, 0, 127, 63);
}
```



Rysunek 4. Wygląd głównego okna programu PixelFactory

Wykaz elementów

Rezystory: (1/8 W)
R1, R2: 3,3 kΩ
R3, R6…R8: 1 kΩ
R4: 820 kΩ
R5: 2,2 kΩ

Kondensatory:
C1, C2: 10 µF/16 V (elektrolityczny)
C3, C4: 22 pF (ceramiczny)
C5…C7, C10, C11: 100 nF (ceramiczny)
C8, C9: 470 µF/10 V (elektrolityczny)
C12…C15: 4,7 µF/10 V (elektrolityczny)

Półprzewodniki:
U1: NCP1117DT33G
U2: ATmega164A (TQFP44)
U3: DS18S20
U4: ULN2003A (SO16)
D1: 1N4007
D2: BAT85
D3: 1N4148
T1: BC557A
T2: BC548A
OK1: PC817

Inne:
OLED: wyświetlacz OLED Winstar WEX012864LLPP3N00000 (kolor dobrany przez użytkownika)
L1: dławik 100 µH
L2: dławik 10 µH
Q1: rezonator kwarcowy 4.194 MHz
FT: filtr EMI Murata typu DSS306-55F223
REL1…REL3: przekaźnik HFKW-012-1ZW
REL4, REL5: przekaźnik JRC-27F/012
RADIO: gniazdo męskie kątowe 90° 4pin (NSL25-4W)
AC, LIGHTS: gniazdo męskie kątowe 90° 3pin (NSL25-3W)
INJ: gniazdo męskie kątowe 90° 2pin (NSL25-2W)
FUN: złącze śrubowe typu AK500/4
MODE, PLUS, MINUS: microswitch z długą oską
ZIF: złącze typu ZIF do montażu powierzchniowego (raster 0.5mm, 30-pin, górny kontakt)

Ustawienia ważniejszych fuse-bitów:
CKSEL3...0: 1101
SUT1...0: 11
BODLEVEL2...0: 110
CKOUT: 1
JTAGEN: 1
CKDIV8: 1
OCDEN: 1
EESAVE: 0

Listing 7. Struktura opisująca zestaw znaków
```
typedef struct  //Deklaracja struktury przechowującej parametry bieżącej
czcionki
{
  uint8_t Width;  //Rzeczywista szerokość znaku (px)
  uint8_t Height;  //Rzeczywista wysokość znaku (px)
  uint8_t Interspace;  //Odstęp pomiędzy znakami (px)
  uint8_t BytesPerChar;  //Liczba bajtów danych tablicy wzorców przypadających
na definicję 1 znaku
  uint8_t FirstCharCode;  //Kod ASCII pierwszego znaku w tablicy wzorców
  const uint8_t *Bitmap;  //Wskaźnik to tablicy zawierającej wzorce
poszczególnych znaków
} fontDescription;
```

Listing 8. Funkcja SetFont
```
void SetFont(const fontDescription *Font)
{
  CurrentFont.Width = pgm_read_byte(&Font->Width);
  CurrentFont.Height = pgm_read_byte(&Font->Height);
  CurrentFont.Interspace = pgm_read_byte(&Font->Interspace);
  CurrentFont.BytesPerChar = pgm_read_byte(&Font->BytesPerChar);
  CurrentFont.FirstCharCode = pgm_read_byte(&Font->FirstCharCode);
  CurrentFont.Bitmap = (uint8_t*)pgm_read_word(&Font->Bitmap);
}
```

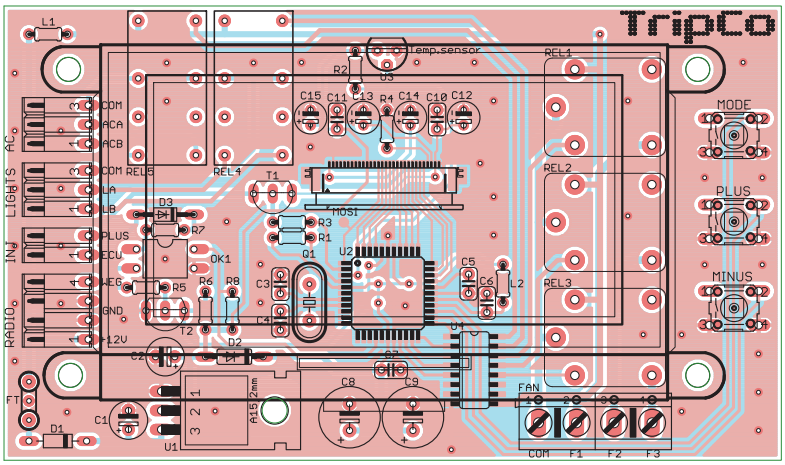
Najważniejszą strukturą z punktu widzenia programu obsługi urządzenia jest ta, która niesie informację na temat cech fizycznych zestawu znaków jak i samego znaku. Budowę struktury pokazano na **listingu 7**. Tablica wzorców poszczególnych znaków, do której wskaźnik umieszczony zostanie we wspomnianej strukturze, jest typową tablicą *uint8_t Tablica[] PROGMEM* zawierającą wygenerowane przez program PixelFactory wzorce znaków. Nie korzystamy zarazem z trzeciej struktury, którą tworzy oprogramowanie, a która to zawiera informację o kodach znaków (i stosownym offsecie) umieszczonych w tablicy wzorców, gdyż w programie obsługi urządzenia korzysta się z ciągłego przedziału dostępnych znaków standardu ASCII w odpowiednio ograniczonym zakresie (np. same cyfry lub małe litery). Dla wygody przewidziano dodatkową funkcję SetFont, której zadaniem jest odczytanie z pamięci Flash parametrów bieżącej czcionki do zmiennej umieszczonej w pamięci RAM, co upraszcza nieco konstrukcję właściwej funkcji odpowiedzialnej za wyświetlanie tekstu. Konstrukcję wspomnianej funkcji jak i docelowej funkcji odpowiedzialnej za wyświetlanie napisów (z uwzględnieniem specyfiki sterownika SSD1325) zamieszczono na **listingu 8** i **listingu 9**.

Montaż

Schemat montażowy komputerka TripComp pokazano na **rysunku 5**. Zmontowano go na płytce dwustronnej przeważnie używając elementów przeznaczonych do montażu przewlekanego. Co bardzo ważne w przypadku środowisk o sporej liczbie zaburzeń, którym bez wątpienia jest instalacja samochodowa, zadbano o odpowiednie prowadzenie masy oraz ścieżek sygnałów krytycznych.

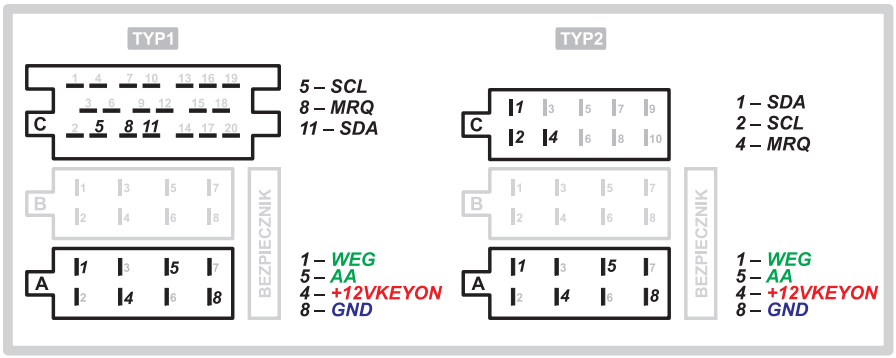
Montaż rozpoczynamy od wlutowania elementów SMD, czyli w naszym przypadku mikrokontrolera oraz złącza ZIF. Montaż tego typu peryferiów możemy wykonać na co najmniej dwa sposoby w zależności od sprzętu lutowniczego jakim dysponujemy. Sposób pierwszy to użycie specjalnej

Listing 9. Funkcja odpowiedzialna za wyświetlanie napisów na wyświetlaczu OLED
```
/* Funkcja narzędziowa, która testuje kolejne pary bitów bajta, argumentu wywołania by utworzyć bajt wynikowy
zawierający dane 2 pixeli w 16 o dcieniach. Liczbę testowanych par bitów zawiera argument bitsNr, który musi byc
parzysty, np. dla bitsNr=6 testowane są pary: 7-6, 5-4, 3-2 */

void sendBitsPairs(uint8_t Byte, uint8_t bitsNr, uint8_t Color, uint8_t Background)
{
  register uint8_t displayData, bitIndex, boundary = (bitsNr == 8) ? 255: 7 - bitsNr;
  for( bitIndex=7; bitIndex != boundary ; bitIndex-=2 )
  {
    if(Byte & (1<<bitIndex)) displayData = Color <<4; else displayData = Background <<4;
    if(Byte & (1<<(bitIndex-1))) displayData |= Color; else displayData |= Background;
    WriteGDDram(displayData );
  }
}

//Właściwa funkcja do obsługi czcionek
void DrawText(uint8_t X1, uint8_t Y1, char *Text, uint8_t Color, uint8_t Background)
{
  register char Character;
  register uint8_t widthIndex, heightIndex, widthByteNr, readByte;
  uint16_t offset;

  while ((Character = *Text++))  //Sprawdzamy kody ASCII kolejnych znaków napisu wejściowego, dla SPACJI tylko
wymazujemy
  {
    if(Character == ' ') DrawRectangle(X1, Y1, X1+CurrentFont.Width-1, Y1+CurrentFont.Height-1, ((Background<<4) |
Background));
    else
    {
      SetActiveWindow(X1, Y1, X1+CurrentFont.Width-1, Y1+CurrentFont.Height-1);
//Obliczamy offset położenia wzroca w tablicy wzorców znaków
      offset = (Character-CurrentFont.FirstCharCode)*CurrentFont.BytesPerChar;
      for(heightIndex = 0; heightIndex < CurrentFont.Height; heightIndex++)
      {
        for(widthIndex = 0, widthByteNr = 0; widthIndex < CurrentFont.Width; widthIndex += 8, widthByteNr++)
        {
//Odczytujemy kolejny bajt definicji znaku umieszczony w tablicy Bitmap pod odpowiednim adresem
          readByte = pgm_read_byte(&CurrentFont.Bitmap[offset++]);
/* W zależności od tego czy do czynienia mamy z bajtem, który jest całkowicie wykorzystany, jeśli chodzi o definicję
wzorca, czy też tylko część jego bitów opisuje wzorzec (dla czcionek o szerokości niebędącej wielokrotnością liczby 8)
testujemy odpowiednie grupy bitów wysyłając dane do panelu OLED */
          if( ((widthByteNr+1)*8) <= CurrentFont.Width ) sendBitsPairs( readByte, 8, Color, Background);
          else sendBitsPairs( readByte, CurrentFont.Width - (widthByteNr *8), Color, Background);
        }
      }
    }
//Przesuwamy współrzędną X1 o szerokość znaku plus zdefiniowany odstęp
    X1+=(CurrentFont.Width+CurrentFont.Interspace);
  }
  SetActiveWindow(0, 0, 127, 63);
}
```



Rysunek 5. Schemat montażowy sterownika TripCo

Widok złącz radioodbiornika od strony wnętrza kieszeni montażowej.

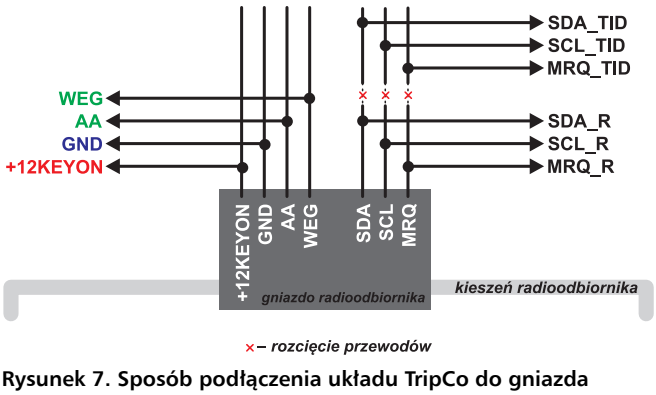


Rysunek 6. Rodzaje złącz radioodbiornika wraz z opisem wyprowadzeń

stacji lutowniczej typu Hot Air oraz odpowiednich, przeznaczonych do tego celu, topników. Sposób drugi (stosowany przeze mnie) to montaż przy użyciu typowej stacji lutowniczej, dobrej cyny z odpowiednią ilością topnika oraz plecionki rozlutowniczej, która umożliwia usunięcie nadmiaru cyny spomiędzy wyprowadzeń układów. Należy przy tym uważać by nie uszkodzić termicznie elementów. Następnie lutujemy rezystory, kondensatory, przekaźniki, złącza, przyciski, a na końcu pozostałe półprzewodniki. Należy zwrócić szczególną uwagę na konieczność pocynowania grubą warstwą cyny ścieżek przewodzących duże prądy, tj. ścieżek styków wykonawczych przekaźników REL1…REL3. Z uwagi na zagęszczenie wyprowadzeń złącza ZIF i mikrokontrolera przed pierwszym podłączeniem układu należy jeszcze raz sprawdzić jakość wykonanych połączeń, aby nie dopuścić do ewentualnych zwarć. Wspomniana kontrola będzie znacznie łatwiejsza, jeśli zmontowana płytkę sterownika przemyjemy alkoholem izopropylowym w celu wypłukania nadmiaru kalafonii lutowniczej. Tuż przed przykręceniem wyświetlacza do obwodu drukowanego naszego układu (należy użyć odpowiedniej długości dystansów), należy go podłączyć korzystając ze złącza ZIF umieszczonego po stronie elementów.

Dołączenie sterownika

Dołączenie sterownika do instalacji samochodowej powinno zostać wykonane przez doświadczonego elektryka bądź elektronika samochodowego, najlepiej przy odłączonym akumulatorze. Urządzenie należy zamontować w suchym miejscu, z dala od wszelkiego rodzaju elektroniki, której działanie mogłoby zakłócać funkcjonowanie sterownika (typu sterownik silnika ECU, moduł kontroli nadwozia BCM czy alarm), zaopatrując w odpowiednią ekranowaną obudowę chroniącą przez zwarciem, zawilgoceniem, uszkodzeniem mechanicznym i zakłóceniami EMI.



Rysunek 7. Sposób podłączenia układu TripCo do gniazda radioodbiornika

W celu wykorzystania wszystkich dostępnych funkcji urządzenia, sterownik należy podłączyć do następujących „modułów" samochodu:

- Złącze radioodbiornika: to podstawowe podłączenie umożliwiające zasilenie układu, po włączeniu stacyjki, oraz doprowadzenie sygnału prędkości pojazdu (WEG).
- Panel sterowania nawiewem i klimatyzacją: to podłączenie umożliwia automatyczne sterowanie układem klimatyzacji manualnej oraz wentylatorem nawiewu (bardzo ważny jest w tym przypadku przekrój przewodów połączeniowych z uwagi na duże prądy – min. 1,5 mm^2).
- Panel sterowania wyłącznikiem świateł: to połączenie umożliwia automatyczne włączanie świateł mijania.
- Wtryskiwacz paliwa: to podłączenie umożliwia realizację funkcji komputera pokładowego.

Ponadto, układ TripCo wyposażono w zintegrowany, dokładny termometr scalony, który powinien być przyłączony do płytki urządzenia za pomocą 3-żyłowego odcinka przewodu. Należy eksperymentalnie dobrać miejsce zamocowania czujnika, aby odwzorować średnią temperaturę panującą wewnątrz pojazdu i uniknąć niepotrzebnych zadziałań automatyki. Unikać należy montażu czujnika w pobliżu nawiewów, drzwi, okien itp. Najlepszym miejscem wydaje się być tylna część tunelu środkowego bądź kieszeń-schowek pod radioodbiornikiem. Zgodnie z tym, co napisano wcześniej, poniższy sterownik realizuje swoje funkcje przy użyciu przekaźników dużej mocy, których styki wykonawcze podłączone są do styków odpowiadających im wyłączników zamontowanych oryginalnie w pojeździe. Dodatkowo, należy zwrócić uwagę na fakt, iż przekaźniki sterujące załączaniem układu klimatyzacji manualnej oraz wyłącznika świateł mają dwa komplety styków, co wynika wyłącznie z faktu, iż wyłączniki odpowiadających im układów (których to styki „bocznikują") wyposażone są także w więcej niż jeden styk wykonawczy i aby „poprawnie" załączyć wybrany układ należy dobrać przekaźnik o takim samym układzie styków. Wygląd typowych złącz radioodbiornika wraz z zaznaczeniem interesujących nas wyprowadzeń pokazano na **rysunku 6**. Na **rysunkach 7…9** pokazano sposób dołączenia sterownika do poszczególnych modułów pojazdu (na przykładzie samochodu Opel Meriva, rok modelowy 2004).

Sposób dołączenia sterownika do wtryskiwacza paliwa

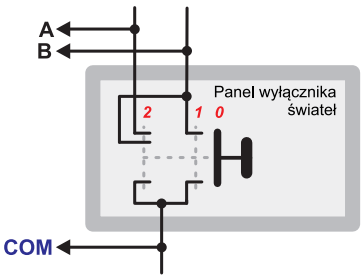
To połączenie należy wykonać nader starannie zachowując dużą ostrożność, by nie doprowadzić do zwarcia przewodów zasilających wtryskiwacz, co mogłoby skutkować uszkodzeniem wyjściowych obwodów sterujących elektronicznego układu sterującego pracą silnika ECU. Każdy wtryskiwacz ma 2 wyprowadzenia: pierwsze z nich to sygnał +12 V, który zostaje podany po przekręceniu kluczyka stacyjki i który należy dołączyć do wejścia „plus" sterownika. Drugie to sygnał sterujący z modułu ECU (komutowana masa), który należy dołączyć do wejścia ECU sterownika. Wspomniane przewody trzeba starannie zabezpieczyć przed możliwością ewentualnego przetarcia izolacji i powstania zwarcia – dotyczy to zwłaszcza otworów przelotowych, przez które zostaną one przeprowadzone. Aby maksymalnie zabezpieczyć wejściowy układ pomiarowy czasu wtrysku przed zaburzeniami (np. występującymi zwykle w pobliżu listwy zapłonowej), najlepiej zastosować dwużyłowy przewód ekranowany o odpowiednim przekroju, a ekran tego przewodu dołączyć na obu końcach 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. W takim przypadku należy odpowiednio zmniejszyć wartość rezystora R7 ustalającego prąd diody LED transoptora PC817. Na **rysunku 10** przedstawiono rysunek montażowy układu TripCo wraz z opisem wyprowadzeń poszczególnych złącz.

Obsługa

Jako, że urządzenie TripCo jest medium, które może być obsługiwane podczas jazdy samochodem, ergonomia i prostota obsługi tegoż sterownika jak i 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 regulacji aktualnie wyświetlanych parametrów pracy, zaś przycisk MODE służy do zmiany aktualnie wyświetlanego ekranu Menu. Dodatkowo, dla większości trybów pracy, długotrwałe przytrzymanie przycisków PLUS i MINUS zwiększa szybkość regulacji wybranego parametru urządzenia. Oprócz wspomnianej powyżej funkcjonalności, przyciski MINUS i MODE posiadają dodatkowe funkcje: długie 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 MINUS powoduje wykasowanie liczników dystansu i średnich wartości obliczeniowych. Rysunek wszystkich dostępnych trybów pracy sterownika pokazano na **rysunku 11**, zaś na **rysunku 12** pokazano diagram obrazujący system Menu jak również sposób obsługi urządzenia (symbole przycisków wypełnione kolorem czarnym oznaczają długie naciśnięcie wybranego przycisku).

Setup i tryb kalibracji stałej wtrysku

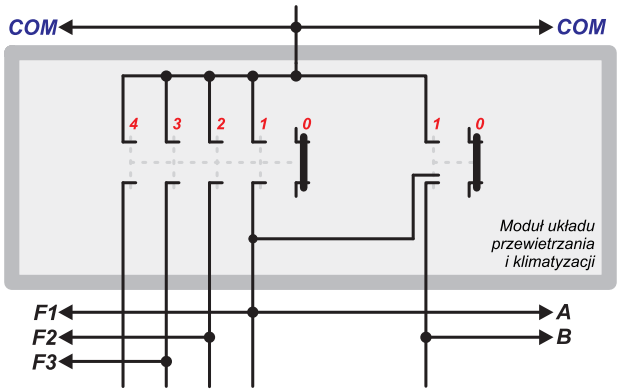
Urządzenie TripCo wyposażono w specjalny, konfiguracyjny tryb pracy, dzięki któremu możemy określić pewne, niezbędne parametry regulacyjne nieodzowne z punktu widzenia funkcjonalności komputera pokładowego. Ten tryb



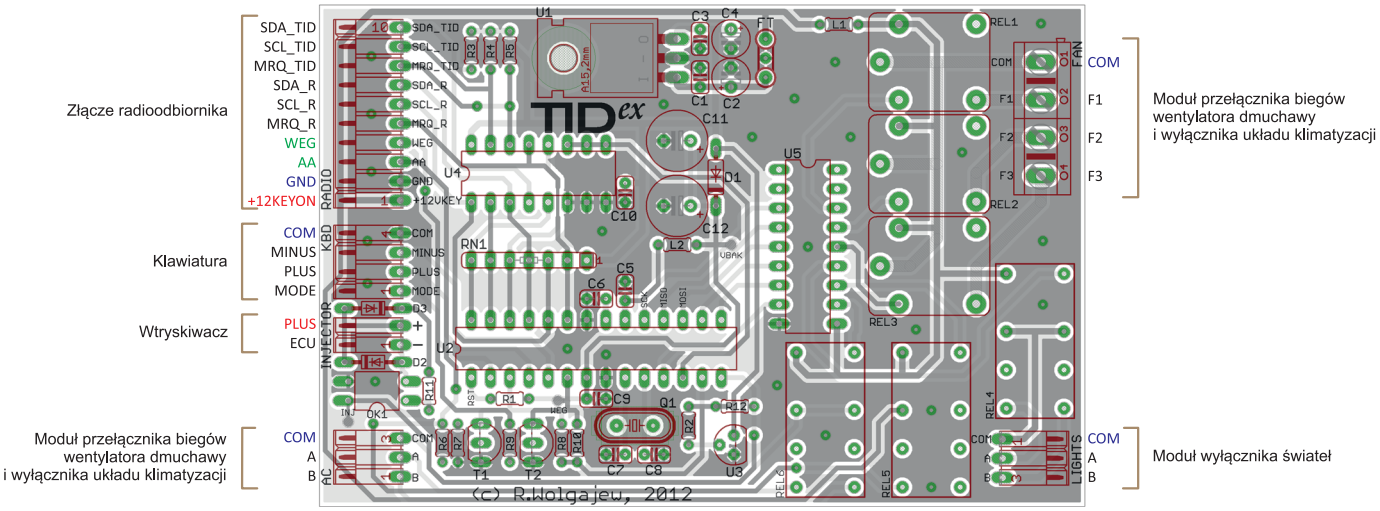
Rysunek 8. Sposób podłączenia układu TripCo do modułu wyłącznika świateł

pracy uruchamiamy poprzez naciśnięcie i przytrzymanie przycisku MODE podczas włączania urządzenia (zapłonu), co zostanie zasygnalizowane przez sterownik wyświetleniem okna informacyjnego z napisem *„setup, wait"*. W tym trybie pracy urządzenia możemy określić wielkość następujących stałych niezbędnych w procesie obliczania zużycia paliwa, prędkości jazdy, drogi oraz kosztów:

- *Stała wtryskiwacza* (w [ml/min]): jest to wielkość charakterystyczna dla każdego wtryskiwacza elektronicznego wtrysku paliwa informująca nas o ilości paliwa, jakie może on wprowadzić do komory spalania w jednostce czasu



Rysunek 9. Sposób podłączenia układu TripCo do modułu przełącznika biegów wentylatora dmuchawy i wyłącznika układu klimatyzacji A/C



Rysunek 10. Rysunek montażowy układu TripCo wraz z opisem wyprowadzeń poszczególnych złącz

Rysunek 11. Dostępne tryby pracy sterownika TripCo wraz z opisem wyświetlanych wartości

(przy założeniu 100% czasu otwarcia zaworu i stałym, charakterystycznym dla każdego wtryskiwacza ciśnieniu zasilającym).

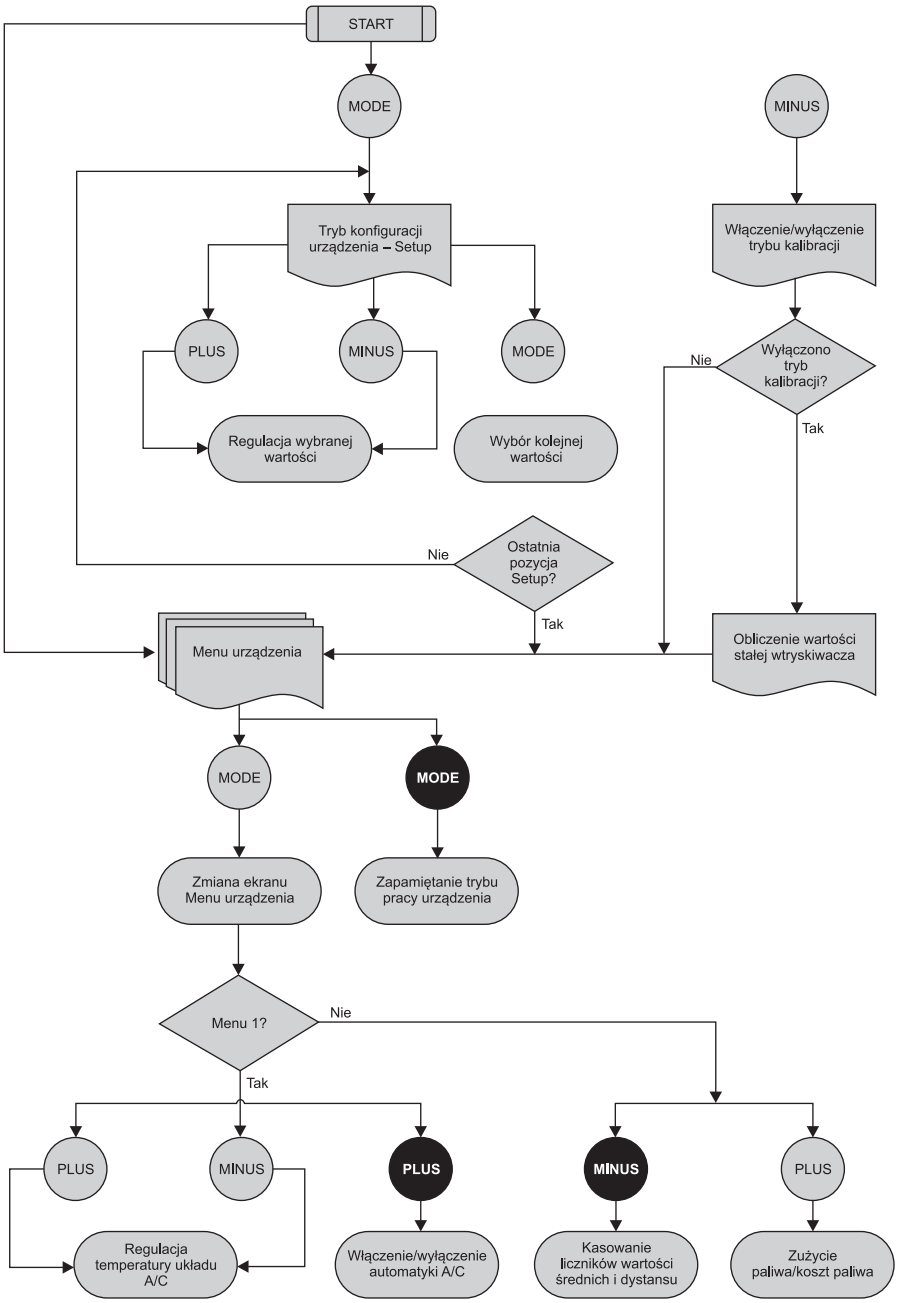
- *Stała przetwornika drogi* (impulsatora, w [imp/obr]): jest to wielkość charakterystyczna dla każdego impulsatora informująca nas o liczbie impulsów przypadających na 1 obrót koła (osi); dla przykładu w większości nowych modeli marki Opel jest to wartość 29 imp/obr.
- *Obwód opony* (w [cm]).
- *Koszt 1l paliwa* (w [zł]).

Wspomniane stałe można znaleźć w Internecie posiłkując się wszelkiego rodzaju forami o tematyce elektronicznej. Obwód opony, a w zasadzie drogę, jaką pokona pojazd w czasie jednego, pełnego obrotu koła należy wyznaczyć empirycznie. W tym celu zaznaczamy (np. kredą) najniżej położone miejsce styku opony z powierzchnią drogi, następnie standardowo obciążony pojazd przetaczamy tak by koło wykonało jeden, pełen obrót, po czym mierzymy pokonany odcinek drogi. Ostatnią pozycją Menu trybu konfiguracyjnego jest możliwość wprowadzenia własnego napisu użytkownika, który to możemy później wywołać jako jeden z trybów pracy urządzenia. 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, ponieważ te elementy są często wykonywane na zamówienie producenta pojazdu i na próżno szukać ich oznaczeń na stronach producentów

stosownych podzespołów. Na szczęście przewidziano pewien mechanizm, za pomocą którego sterownik TripCo 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 zasygnalizowane zostanie przez sterownik wyświetleniem okna informacyjnego z napisem *„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, średnie i całkowite zużycie paliwa (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 paliwa.
- Zatankować 20 litrów paliwa.
- Uruchomić procedurę kalibracji.
- Zużyć całe, dostępne paliwo (zatankowane wcześniej 20 litrów), aż do ponownego zaświecenia się lampki sygnalizującej tzw. rezerwę paliwa.
- Zakończyć procedurę kalibracji.

Po wykonaniu tych czynności układ TripCo obliczy, wyświetli i zapisze w nieulotnej pamięci 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 wejście w system Setup i zwiększenie (w przypadku zaniżania spalania) lub zmniejszenie (w przypadku zawyżania spalania) wspomnianej wartości. Należy zaznaczyć, iż tak jak w przypadku 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. Dodatkową pozycją Menu jest możliwość ustawienia kontrastu wyświetlacza OLED (w 128 krokach).



Rysunek 12. Diagram obrazujący system Menu i sposób obsługi sterownika TripCo

Robert Wołgajew, EP