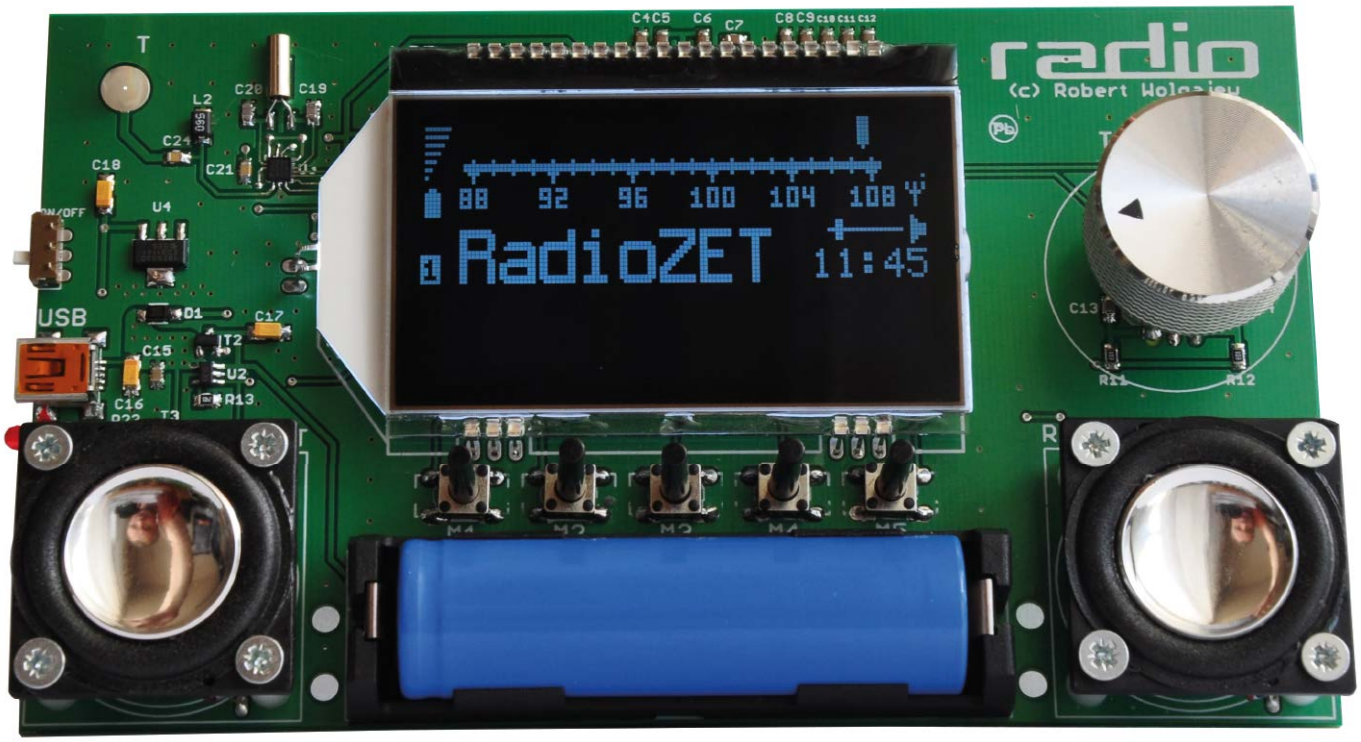


radio

Radioodbiornik stereofoniczny z RDS-em

Wiedziony sentymentem do starych rozwiązań i zaopatrzony w dzisiejszą wiedzę z dziedziny elektroniki, postanowiłem zbudować prosty radioodbiornik wyposażony w nowoczesne funkcje. Rezultat opublikowano na łamach Elektroniki Praktycznej 6-7/2013. Radioodbiornik pocketRadio był jednak urządzeniem przenośnym, do którego funkcjonowania niezbędny był odpowiedni zestaw słuchawkowy oraz pakiet baterii zasilających, co odsuwało go nieco od idei pierwowzoru. Postanowiłem zbudować radio, które nawiązywałoby do starszych konstrukcji, ale w znacznie nowocześniejszym wydaniu i do tego w bardzo atrakcyjnej cenie. Tak jak wtedy, tak i teraz, do budowy naszego urządzenia postanowiłem wykorzystać doskonały układ scalonego odbiornika FM pod postacią układu Si4703 produkowanego przez firmę Silicon Labs, specjalistę w dziedzinie tego typu rozwiązań.

Rekomendacje: *nowoczesny odbiornik radiowy, który przyda się w domu i w trakcie wypadów wakacyjnych.*

Część z Was, drodzy Czytelnicy, pamięta zapewne dobre czasy rozkwitu polskiej elektroniki użytkowej, które przypadły na lata siedemdziesiąte, osiemdziesiąte i dziewięćdziesiąte ubiegłego wieku. Mnie osobiście urządzenia z tego okresu kojarzą się będą z polską myślą techniczną, niezawodnością, dbałością o szczegóły wykonania oraz niejednokrotnie – innowacyjnością. Wystarczy wspomnieć zestawy „wieżowe" takich firm, jak Diora, Zakłady Kasprzaka, nie mówiąc już o samych początkach (w moim postrzeganiu tego okresu), które to zawsze będę utożsamiał z radioodbiornikiem przenośnym o wdzięcznej nazwie Jowita 2 w obudowie częściowo wykonanej z drewna. Na zawsze w mojej pamięci pozostanie odbiornik tego typu, który funkcjonował w domu mojej bliskiej rodziny do początków XXI wieku! Powiedzcie, proszę, które z urządzeń produkowanych dzisiaj przetrwa 30lat?! Czemu zaprzepaszczono ten cały dorobek nazywany dzisiaj dumnie „know how"? Moim zdaniem dlatego, że Państwo jako całość nie miało nigdy i nie ma do dzisiaj odpowiedniej polityki rozwoju i ochrony rynku wewnętrznego…ale to temat na inny artykuł.

Z rozrzewnieniem wspominam swoje pierwsze konstrukcje budowane w tym czasie,

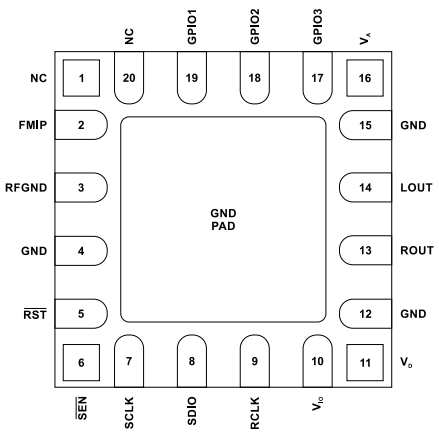
które to mimo swojej prostoty dawały wiele satysfakcji. Jak chyba każdy swoją przygodę z elektroniką rozpoczynałem od zbudowania właśnie takiego prostego radyjka złożonego zaledwie z kilku elementów, jak dla przykładu legendarne radyjko AM w pudełku od zapałek, zbudowane na bazie „zasłużonego" układu UL1111, znane wszystkim czytelnikom „Młodego Technika". Mimo prostoty układy tego rodzaju nie były wcale łatwe do uruchomienia, a to za sprawą wielu elementów indukcyjnych, których konfiguracja znacząco wpływała na efekt końcowy. Cóż z tego, że takie urządzenia pozostawiały wiele do życzenia, jeśli chodzi o jakość emitowanego przezeń dźwięku, jeśli dawały wiele nieskrępowanej radości i nadziei rozwoju w przyszłości.

Wiedziony sentymentem do starych rozwiązań i zaopatrzony w dzisiejszą wiedzę z dziedziny elektroniki, postanowiłem jakiś czas temu zbudować prosty radioodbiornik wyposażony w nowoczesne funkcje, którego ziszczeniem był projekt o nazwie pocketRadio opublikowany na łamach Elektroniki Praktycznej 6…7/2013. Radioodbiornik pocketRadio był jednak urządzeniem przenośnym, do którego funkcjonowania niezbędny był odpowiedni zestaw słuchawkowy oraz pakiet baterii zasilających, co odsuwało go nieco od idei pierwowzoru. Tym razem postanowiłem zbudować radio, które nawiązywałoby do dawnych konstrukcji, ale w znacznie nowocześniejszym wydaniu i do tego w bardzo atrakcyjnej cenie.

Tak jak wtedy, tak i teraz, do budowy urządzenia postanowiłem wykorzystać doskonały układ scalony odbiornika FM typu Si4703 firmy Silicon Labs, specjalistę w dziedzinie tego typu rozwiązań (i nie tylko). Układ ten jest kompletnym odbiornikiem radiowym przeznaczonym do odbioru emisji w paśmie FM charakteryzującym się następującymi, wybranymi cechami użytkowymi:

- Odbiór stacji radiowych w pełnym zakresie częstotliwości pasma FM (76…108 MHz).
- Cyfrowa synteza częstotliwości z wbudowanym oscylatorem VCO.
- Wbudowane układy AFC (Automatic Frequency Control) i AGC (Automatic Gain Control).
- Obsługa konfigurowalnej funkcji Seek (przeszukiwanie pasma).
- Pomiar mocy sygnału antenowego.
- Wbudowana funkcja regulacji głośności sygnału wyjściowego.
- Wbudowany układ oscylatora dla rezonatora kwarcowego 32768 Hz.
- Obsługa interfesju I²C oraz SPI.
- Minimalna liczba niezbędnych elementów zewnętrznych w typowej aplikacji.
- Brak konieczności jakiegokolwiek strojenia obwodów radiowych, gdyż w układzie wykorzystano cyfrową obróbkę sygnałów za pomocą zaawansowanych technik DSP.
- Szeroki zakres napięć zasilania (2,7…5,5 V).
- Niewielki pobór mocy (w tym tryb o ekstremalnie małym zapotrzebowaniu na energię).
- Wbudowany regulator napięcia typu LDO.
- Obsługa systemu RDS/RDBS.
- Małe wymiary obudowy (3 mm×3 mm).

Cechy użytkowe układu idealnie predestynują go do zastosowań w sprzęcie tego typu zwłaszcza, że aplikacja układu ogranicza się do zaledwie kilku elementów zewnętrznych. Wynika to z faktu, o czym wspomniano pokrótce wcześniej, że w budowie układu Si4703 wykorzystano zaawansowany, cyfrowy tor przetwarzania sygnału



Rysunek 1. Wygląd obudowy układu Si4703

Tabela 1. Opis funkcji wyprowadzeń układu Si4703

Numer pinu	Nazwa	Opis
1, 20	NC	Niepodłączone
2	FMIP	Wejście sygnału antenowego
3	RFGND	Masa części radiowej układu (należy połączyć z polem masy PCB)
4, 12, 15, PAD	GND	Masa (należy połączyć z polem masy PCB)
5	RST	Reset układu (aktywny stan niski)
6	SEN	Wejście aktywacji i wyboru rodzaju magistrali sterującej (aktywny stan niski)
7	SCLK	Wejście zegarowe magistrali sterującej
8	SDIO	Wejście/wyjście danych magistrali sterującej
9	RCLK	Wejście zewnętrznego sygnału zegarowego syntezera częstotliwości
10	VIO	Napięcie zasilania układów wejścia/wyjścia odbiornika
11	VD	Napięcie zasilania części cyfrowej układu
13	ROUT	Wyjście audio – kanał prawy
14	LOUT	Wyjście audio – kanał lewy
16	VA	Napięcie zasilania części analogowej układu
17	GPIO3	Uniwersalny, programowalny port IO (może pełnić rolę wskaźnika sygnału Stereo)
18	GPIO2	Uniwersalny, programowalny port IO (może pełnić rolę przerwania od gotowości danych RDS lub zakończenia strojenia/przeszukiwania pasma)
19	GPIO1	Uniwersalny, programowalny port IO

Podstawowe informacje:

- Napięcie zasilania: 5 V (np. USB)
- Maksymalny prąd obciążenia (wyświetlacz załączony/przyciemniony/wyłączony): 60 mA/50 mA/27 mA
- Zakres częstotliwości radioodbiornika FM: 87.5…108 MHz
- Typ obsługiwanych wiadomości RDS: PS (Program Service), RT (Radio Text), CT (Clock & Time)
- Maksymalna moc wyjściowa audio: 2×320 mW przy obciążeniu 4 Ω
- Impedancja obciążenia: 4…16 Ω

Dodatkowe materiały na FTP:

ftp://ep.com.pl, user: 66465, pass: td79fgh6

- wzory płytek PCB

Projekty pokrewne na FTP:

(wymienione artykuły są w całości dostępne na FTP)

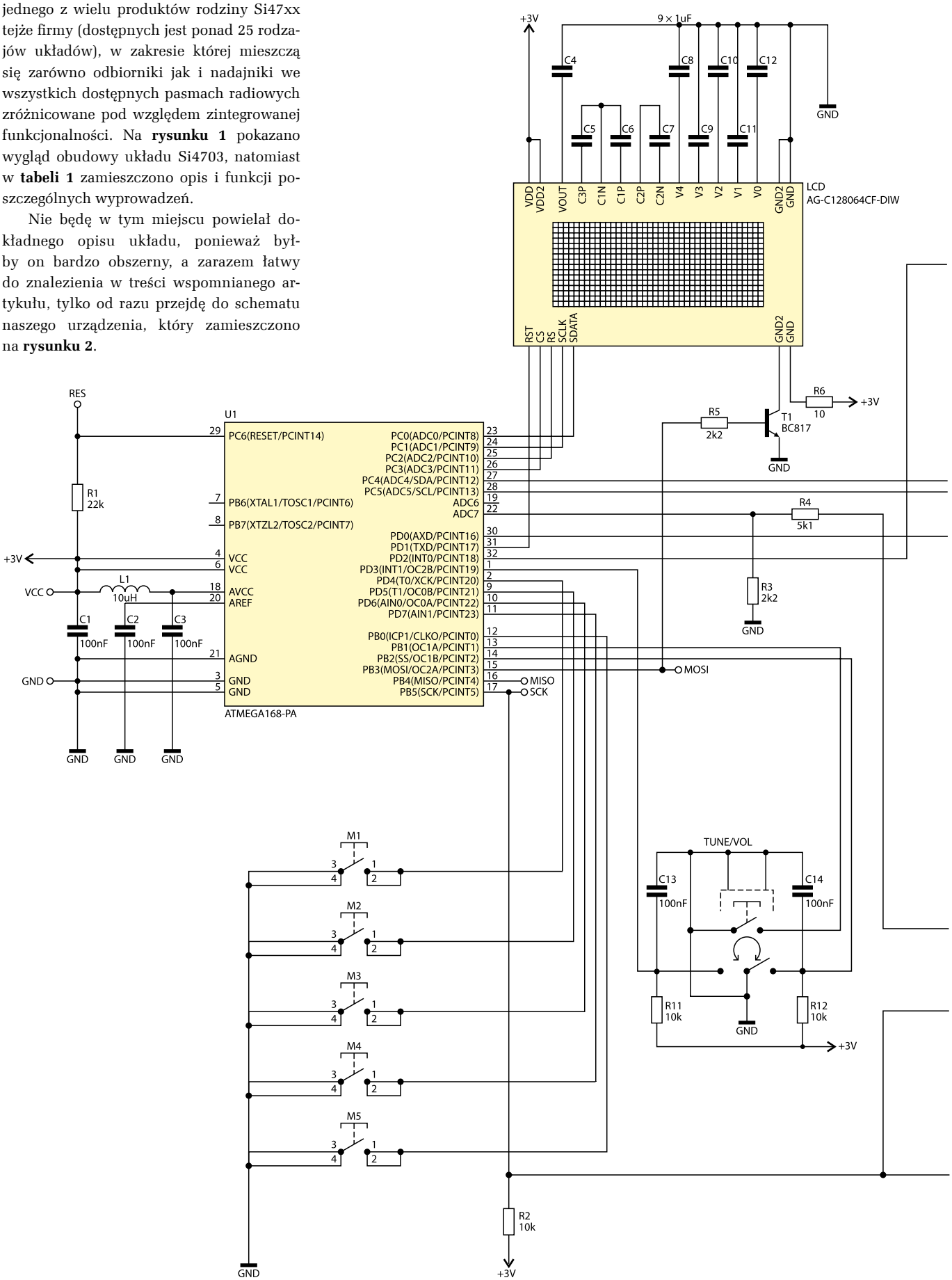
AVT-5401	PocketRadio – radioodbiornik kieszonkowy z RDS (EP 6-7/2013)
AVT-5317	Lampowo-tranzystorowy odbiornik UKF (EP 11/2011)
AVT-5242	Radioodbiornik internetowy (EP 7/2010)
AVT-5016	Amplituner FM z RDS (EP 6-7/2001)
AVT-2469	Odbiornik UKF FM (EdW 1/2001)
AVT-2330	Miniaturowy odbiornik FM stereo (EdW 2/1999)

* 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 wymienionych 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

za pomocą przetworników A/C oraz C/A oraz procesora DSP. Zresztą wystarczy wspomnieć, iż element Si4703 jest przykładem jednego z wielu produktów rodziny Si47xx tejże firmy (dostępnych jest ponad 25 rodzajów układów), w zakresie której mieszczą się zarówno odbiorniki jak i nadajniki we wszystkich dostępnych pasmach radiowych zróżnicowane pod względem zintegrowanej funkcjonalności. Na **rysunku 1** pokazano wygląd obudowy układu Si4703, natomiast w **tabeli 1** zamieszczono opis i funkcji poszczególnych wyprowadzeń.

Nie będę w tym miejscu powielał dokładnego opisu układu, ponieważ byłby on bardzo obszerny, a zarazem łatwy do znalezienia w treści wspomnianego artykułu, tylko od razu przejdę do schematu naszego urządzenia, który zamieszczono na **rysunku 2**.

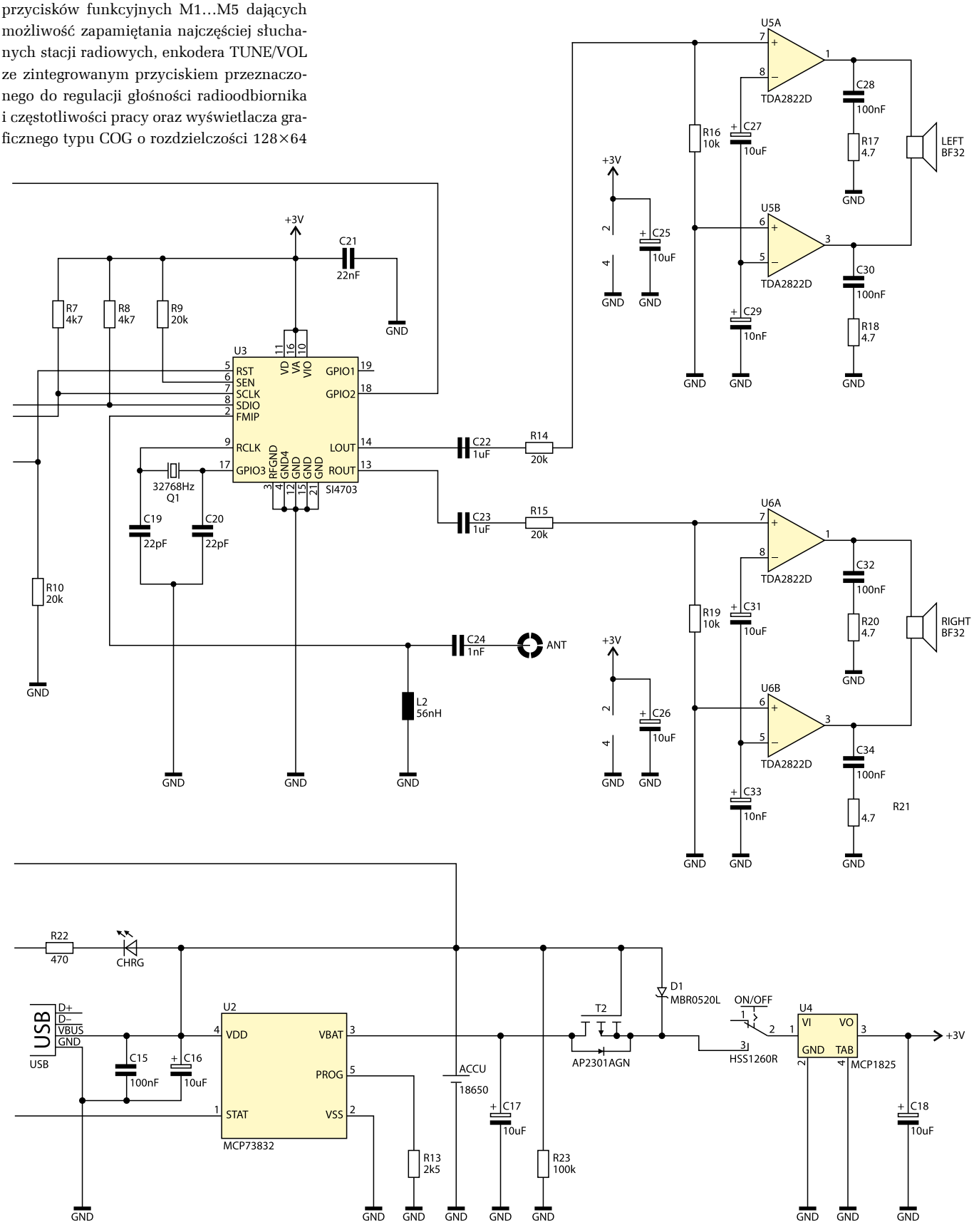
Jest to nieskomplikowany system mikroprocesorowy zbudowany z wykorzystaniem popularnego mikrokontrolera ATmega168PA taktowanego wewnętrznym, wysokostabilnym oscylatorem o częstotliwości 8 MHz, którego zadaniem jest obsługa układu Si4703



Rysunek 2. Schemat ideowy urządzenia Radio

za pomocą interfejsu TWI, obsługa i dekodowanie wiadomości systemu RDS z wykorzystaniem przerwania zewnętrznego INT0 oraz odpowiednio skonfigurowanego wyprowadzenia GPIO2 układu Si4703. Mikrokontroler odpowiada także za obsługę interfejsu użytkownika złożonego z pięciu przycisków funkcyjnych M1…M5 dających możliwość zapamiętania najczęściej słuchanych stacji radiowych, enkodera TUNE/VOL ze zintegrowanym przyciskiem przeznaczonego do regulacji głośności radioodbiornika i częstotliwości pracy oraz wyświetlacza graficznego typu COG o rozdzielczości 128×64 piksele z kontrolerem ST7565R. Wybór tego rodzaju wyświetlacza podyktowany był jego niską ceną oraz rozdzielczością wystarczającą na potrzeby tejże aplikacji. Nie bez znaczenia jest też fakt, iż sterownik ST7565R komunikuje się z mikrokontrolerem za pomocą interfejsu SPI, co znacznie ogranicza liczbę niezbędnych połączeń i pozwala na zastosowanie układu o niewielkiej liczbie wyprowadzeń.

Wróćmy jednak na chwilę do naszego „głównego bohatera”, którym jest radio FM pod postacią układu Si4703, a które to wymaga pewnego, osobliwego procesu

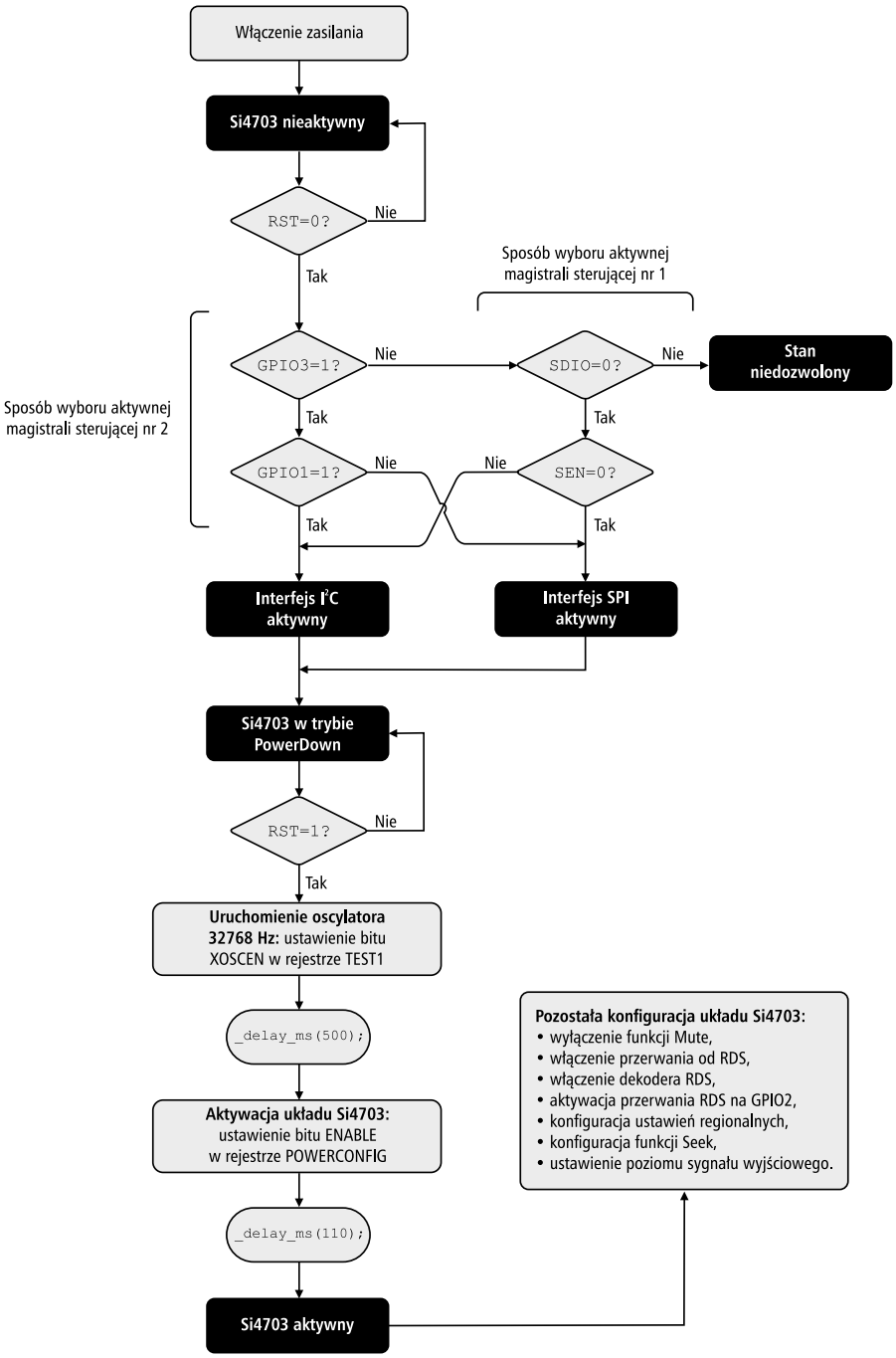


inicjalizacji, jako że wyposażono je w dwa interfejsy komunikacyjne. Jak to często bywa, obsługa i inicjalizacja naszego radyjka jest możliwa dzięki wyposażeniu go w magistralę sterującą (I²C lub SPI), za pomocą której dokonujemy niezbędnej konfiguracji. W tym celu posługujemy się szeregiem 16-bitowych rejestrów konfiguracyjnych kontrolujących pracę radioodbiornika. Aby jednak rozpocząć właściwą transmisję, konieczne jest poprawne zainicjowanie układu, które ma na celu wybór aktywnej magistrali sterującej (pomiędzy I²C i SPI) oraz uruchomienie wewnętrznego oscylatora, niezbędnego z punktu widzenia części radiowej układu. Przewidziano dwa sposoby wyboru rodzaju aktywnej magistrali sterującej, różniące się liczbą niezbędnych wyprowadzeń układu Si4703 zaangażowanych w ten proces. Pierwszy zakłada wykorzystanie wyprowadzeń GPIO3, SEN oraz SDIO układu, drugi – GPIO3 i GPIO1. Należy jednak zaznaczyć, iż rekomendowanym sposobem wyboru aktywnej magistrali sterującej przy wykorzystaniu wewnętrznego oscylatora dla rezonatora 32768 Hz jest sposób pierwszy, gdyż wewnętrzne moduły peryferyjne układu Si4703 zapewniają niezbędne ściągnięcie wyprowadzenia GPIO3 pracującego w układzie oscylatora 32768 Hz do masy w czasie, gdy sygnał RST jest wyzerowany. Po wykonaniu procedury wyboru aktywnej magistrali sterującej (tu I²C), niezbędne jest uruchomienie oscylatora 32768 Hz (ewentualnie dostarczenie takiego sygnału z zewnątrz do wyprowadzenia RCLK) oraz aktywacja układu Si4703 (za pomocą bitów Enable/Disable rejestru POWERCONFIG). Graf kompletnej procedury inicjalizacji układu Si4703 z opcjonalnym wyborem rodzaju magistrali sterującej pokazano na **rysunku 3**.

Po wykonaniu inicjalizacji układu w skład, której wchodzi wybór aktywnej magistrali sterującej, możemy przystąpić do konfiguracji parametrów sprzętowych naszego radyjka. Jak napisałem, nie będę w tym miejscu powielał informacji na wspomniany temat, gdyż zainteresowani Czytelnicy z łatwością znajdą je w treści artykułu nt. urządzenia „pocketRadio', zatem idźmy.

Wykaz elementów

Rezystory: (SMD 0805):
R1: 22 kΩ
R2, R11, R12, R16, R19: 10 kΩ
R3, R5, R22: 2,2 kΩ
R4: 5,1 kΩ
R6: 10 Ω
R7, R8: 4,7 kΩ
R9, R10, R14, R15: 20 kΩ
R13: 2,5 kΩ
R17, R18, R20, R21: 4,7 Ω
R22: 470 Ω
R23: 100 kΩ
Kondensatory: (SMD 0805):
C1…C3, C13…C15, C28, C30, C32, C34: 100 nF
C4…C12, C22, C23: 1 µF
C16…C18, C25…C27, C31: 10 µF/10 V (SMD „A")
C19, C20: 22 pF
C21: 22 nF
C24: 1 nF
C29, C33: 10 nF
Półprzewodniki:
U1: ATmega168PA (TQFP32)
U2: MCP73832 (SOT-23/5)
U3: Si4703 (QFN20)
U4: MCP1825S-3002ED (SOT223)
U5, U6: TDA2822D (SO8)
T1: BC817 (SOT23)
T2: AP2301AGN (SOT23)
D1: MBR0520L (SOD123)
CHRG: czerwona dioda LED 3 mm
Inne:
LCD: wyświetlacz graficzny AG-C128064CF--DIW (COG, 128×64 px, sterownik ST7565R)
L1: dławik 10 µH (SMD 0805)
L2: dławik 56 nH (SMD 1206)
Q1: rezonator kwarcowy, zegarkowy 32768 Hz
M1…M5: przycisk TACT-69N-F (wysokość 30 mm)
LEFT, RIGHT: głośnik miniaturowy VISATON VS-BF32-8
TUNE/VOL: enkoder ze zintegrowanym przyciskiem (długość oski 30 mm)
USB: gniazdo mini USB-B SMT typu DS1104--BN0SR
ON/OFF: przełącznik hebelkowy do druku HSS1260R
ACCU: koszyczek akumulatora MR18650 typu KEYSTONE KEYS1043 plus akumulator LI-Ion MR18650 3,7 V/2200 mAh typu ACCU-ICR18650-2.2
KNOB: gałka aluminiowa ze wskaźnikiem typu MC-131-6.4 (Ø25×15 mm)



Rysunek 3. Graf kompletnej procedury inicjalizacji układu Si4703

Mikrokontroler, będący niejako „sercem" całego urządzenia, odpowiada jeszcze za następujące, wcześniej niewymienione funkcjonalności:

- Steruje intensywnością podświetlenia wyświetlacza graficznego wykorzystując w tym celu układ czasowo-licznikowy Timer2 pracujący w trybie PWM (jego kanał OC2A) oraz tranzystor T1, dzięki czemu pozwala na zmniejszenie intensywności podświetlenia po pewnym czasie bezczynności po stronie użytkownika a następnie dalsze wygaszenie go (po kolejnym czasie bezczynności), co przyczynia się do znacznej oszczędności energii.
- Mierzy napięcie akumulatora ACCU zasilającego urządzenie, dzięki wykorzystaniu wbudowanego przetwornika A/C, wewnętrznego źródła napięcia odniesienia 1,1 V oraz rezystancyjnego dzielnika napięcia R3/R4, co pozwala na wizualizację stanu akumulatora w ramach graficznego interfejsu użytkownika.
- Cyklicznie bada stan wyprowadzenia STAT scalonego układu ładowania MCP73832, dzięki czemu otrzymuje informację o statusie ładowania, którą to także wyświetla ramach graficznego interfejsu użytkownika.

Na tą chwilę to tyle, jeśli chodzi o blok cyfrowy urządzenia. W ramach części analogowej zbudowano natomiast końcówkę mocy, w której zastosowano niedrogie układy typu TDA2822, tym razem w wersji SMD (stąd literka „D" w oznaczeniu układu). Układ TDA2822 jest zintegrowanym, stereofonicznym wzmacniaczem małej mocy charakteryzującym się dość dobrymi parametrami elektrycznymi i możliwością pracy w szerokim zakresie napięcia zasilania 1,8…15 V, co idealnie predestynuje go do zastosowań w sprzęcie przenośnym. W naszym urządzeniu każdy z tych scalaków pracuje w układzie mostkowym, co pozwoliło na efektywne zwiększenie maksymalnej mocy wyjściowej, przy napięcia zasilania 3 V i impedancji obciążenia 4 Ω jest na poziomie 320 mW na kanał, a więc wydaje się wystarczająca w konstrukcji tego typu. Wyjścia układów TDA2822D dołączono do głośników miniaturowych VS-BF32-8 produkcji firmy Visaton, które charakteryzują się dobrymi parametrami elektrycznymi i bardzo efektownym wyglądem. Moc maksymalna tych przetworników to 2 W, impedancja 8 Ω, zaś pasmo przenoszenia rozciąga się od 150 Hz do 20 kHz.

Wyjście antenowe układu Si4703 wyprowadzono poprzez filtr L2/C24 do zacisku ANT radioodbiornika, do którego należy przymocować kawałek przewodu lub też prostą antenę teleskopową, zaś ich długość dobrać eksperymentalnie, w czym z pewnością pomoże wskaźnik mocy sygnału antenowego dostępny w ramach graficznego interfejsu użytkownika.

Na koniec kilka słów o bloku zasilającym, w którego budowie wykorzystano zaawansowany, specjalizowany układ ładowania akumulatorów Li-Ion i Li-Po pod postacią układu MCP73832. Układ ten integruje w sobie kompletny system ładowania, który charakteryzuje się następującą funkcjonalnością:

- Szeroki zakres napięcia zasilania 3,75…6 V.
- Duża dokładność regulacji ±0,75%.
- Programowany prąd ładowania szybkiego w zakresie 15…500 mA (tylko jeden rezystor, w naszym wypadku R13).
- Możliwość wyboru wartości prądu ładowania wstępnego (w odniesieniu do zdefiniowanego powyżej prądu ładowania szybkiego): 10%, 20%, 40% lub opcja nieaktywna.
- Możliwość wyboru poziomu naładowania akumulatora (a dokładnie „reszty" do 100% pojemności), po którym układ przechodzi do trybu ładowania konserwacyjnego: 5%, 7,5%, 10% lub 20%.
- Wbudowany mechanizm wykrywania dołączonego akumulatora.
- Trójstanowe wyjście statusu procesu ładowania STAT.
- Automatyczne przejście do trybu power-down o małym poborze mocy.
- Dostępność układu w bardzo małej, 5-wyprowadzeniowej obudowie SOT-23.

Układ MCP73832 idealnie nadaje się do zastosowania w aplikacjach ładowarek akumulatorów litowych, ponieważ automatycznie nadzoruje proces ładowania wybierając odpowiedni tryb oraz mechanizm kontroli, zaś jedynym „zmartwieniem" użytkownika jest wybór prądu ładowania szybkiego, którego dokonujemy za pomocą rezystora włączonego pomiędzy wyprowadzenie PROG a masę. Prąd ustala się zgodnie z wzorem

$$I_{REG} = 1000V/R_{PROG},$$

gdzie:

- I_{REG} prąd wyrażony w mA,
- R_{PROG} rezystancja w kΩ.

W wypadku naszego urządzenia rezystor R_{PROG} ma rezystancję 2,5 kΩ, co ustawia prąd ładowania szybkiego na wartość 400 mA. Nie bez powodu ustawiono tego typu prąd ładowania. Wszak należy pamiętać, iż dla wygody, do zasilania radioodbiornika przewidziano dołączenie go do portu USB komputera (lub popularnego zasilacza sieciowego z portem USB przeznaczonego do ładowania telefonów komórkowych), a ten pozwala na maksymalny pobór prądu rzędu 500 mA w trybie high-power (zaś typowo 100 mA). Za przełączenie w tryb high-power portu USB odpowiada aplikacja urządzenia podłączanego, co w naszym wypadku nie ma miejsca, więc w większości wypadków nasz układ ładowania dołączony do komputera nie będzie mógł skorzystać z pełnej, zaprogramowanej zdolności prądowej, co wydłuży czas ładowania akumulatora. W przypadku zasilaczy sieciowych z portem USB przeznaczonych do ładowania telefonów komórkowych sytuacja taka nie będzie miała miejsca i czas ładowania ulegnie stosownemu skróceniu. Tak jak wspomniano wcześniej, układ MCP73832 udostępnia specjalne wyprowadzenie STAT, które informuje użytkownika o statusie procesu ładowania według specyfikacji pokazanej w **tabeli 2**.

Kilka dodatkowych słów uwagi wymaga opcjonalny układ współdzielenia obciążenia zbudowany przy użyciu tranzystora T2 typu MOSFET z kanałem P, diody Schottky'iego D1 oraz rezystora R23. Dlaczego niezbędna była implementacja tego rodzaju rozwiązania? Otóż, układ MCP73832 nie posiada w swojej strukturze odpowiednich bloków funkcjonalnych odpowiedzialnych za współdzielenie obciążenia, to znaczy odpowiedzialnych za uwzględnienie w procesie ładowania faktu, iż w czasie, gdy układ nadzoruje proces ładowania akumulatora tenże akumulator zasila urządzenie, które pobiera z niego prąd. Taka sytuacja, po pierwsze powoduje w najlepszym wypadku wydłużenie samego

Tabela 2. Znaczenie stanu wyprowadzenia STAT układu MCP73832 podczas procesu ładowania

Stan procesu ładowania	Stan wyprowadzenia STAT
Tryb power-down układu MCP73832	HIGH-Z
Brak akumulatora	HIGH-Z
Ładowanie wstępne	L
Ładowanie szybkie (tryb constant-current)	L
Ładowanie konserwacyjne (tryb constant-voltage)	L
Proces ładowania zakończony	HIGH-Z

procesu ładowania, zaś w skrajnych wypadkach może go zaburzyć czy też spowodować, iż proces ładowania nigdy nie dobiegnie końca (gdyż odbiornik pobierając nieustannie prąd z ogniwa, a więc de facto z układu nadzorującego, nie pozwoli tym samym na skuteczną detekcję końca procesu ładowania). Aby temu zapobiec, zastosowano „przełącznik" w postaci tranzystora MOSFET, którego bramkę podłączono bezpośrednio do napięcia USB zasilającego ładowarkę. Przy obecności napięcia USB (czyli de facto poziomu wysokiego na bramce tranzystora) tranzystor T2 przechodzi w stan wyłączenia odłączając tym samym ładowany akumulator od obciążenia. W tym samym czasie obciążenie, jakim jest w naszym wypadku radyjko, zostaje zasilone bezpośrednio z napięcia portu USB, a dokładnie rzecz ujmując, poprzez diodę D1. W przypadku odłączenia naszego urządzenia od napięcia zasilającego USB, bramka tranzystora T2 zostaje ściągnięta do masy (poprzez rezystor R24) powodując przewodzenie tegoż tranzystora a więc tym samym zasilenie naszego urządzenia z akumulatora ACCU. W tym przypadku dioda D1 pełni nieco inną funkcję, a mianowicie zabezpiecza przed przepływem prądu wstecznego tj. z akumulatora w kierunku źródła napięcia zasilającego (USB). W ten prosty sposób zbudowano nieskomplikowany i w pełni funkcjonalny układ współdzielenia obciążenia, który czasami występuje w innych typach scalonych kontrolerów ładowania produkcji firmy Microchip. Dalej, wyjście z układu ładowania, poprzez prosty przełącznik mechaniczny, wprowadzono na wejście stabilizatora LDO typu MCP1825, który zapewnia stały poziom napięcia zasilającego system mikroprocesorowy (3 V) niezależnie do stanu układu ładowania. Rozwiązanie takie ma dodatkową zaletę w postaci możliwości ładowania wbudowanego akumulatora nawet wtedy, gdy radioodbiornik pozostaje wyłączony (przy użyciu przełącznika ON/OFF). Właśnie dla takiego przypadku dodano opcjonalne elementy w postaci rezystora R22 i diody LED CHRG, które stanowią prosty interfejs sygnalizacyjny w momencie, gdy urządzenie pozostaje wyłączone, a więc trudno byłoby się zorientować, na jakim etapie ładowania znajduje się zaimplementowany układ ładowania. Świecenie się diody LED świadczy o trwającym procesie ładowania.

Na koniec, można by powiedzieć „na deser", przedstawię kilka praktycznych informacji na temat zastosowanego wyświetlacza LCD. Muszę o tym napisać, ponieważ, mimo iż ten element dosyć popularny, to na próżno szukać dobrze opisanych rozwiązań programowych. Co więcej, w swojej praktyce elektronika-programisty stosowałem już przeróżne wyświetlacze graficzne, począwszy od monochromatycznych o niewielkiej rozdzielczości, a skończywszy na zaawansowanych wyświetlaczach TFT ze zintegrowanym panelem dotykowy, jednak pierwszy raz stosuję monochromatyczny wyświetlacz graficzny wykonany w technologii COG, czyli ze sterownikiem zintegrowanym na szkle. Tym razem zdecydowałem się na tego typu rozwiązanie z uwagi na bardzo atrakcyjną cenę zastosowanego peryferium, które to jest ponad 5 razy tańsze, niż doskonały wyświetlacz OLED zastosowany w urządzeniu „pocketRadio". Wspomniany wyświetlacz wyposażony został w dość prosty sterownik ekranu o oznaczeniu ST7565R, który do komunikacji z procesorem sterującym wykorzystuje interfejs SPI, przez co do nawiązania komunikacji niezbędne jest wykorzystanie wyłącznie pięciu portów mikrokontrolera. W tym celu przewidziano następujące sygnały sterujące:

- SDATA (lub SI) będące szeregowym wejściem danych magistrali SPI.
- SCLK będące wejściem sygnału zegarowego magistrali SPI.
- RS (lub A0) wskazujące na rodzaj danych, jakie przesyłane są do sterownika wyświetlacza LCD (0: przesyłane dane to rozkazy sterujące, 1: przesyłane dane to dane pamięci obrazu).
- CS będące wejściem aktywacji sterownika ekranu (tzw. Chip Select).
- RST (lub RESET) będące wejściem zerowania sterownika ekranu.

Sterownik ST7565R wyposażony został w pamięć ekranu o pojemności 132×65 bitów, która jest zorganizowana w taki sposób, iż cały ekran podzielono na pionowe elementy o wysokości 8 bitów (czyli jednego bajta). W ten sposób pamięć ekranu składa się z 132 kolumn i 9 wierszy (tutaj nazywanych stronami – „Pages"), przy czym ostatni wiersz ma wysokość wyłącznie jednego bitu, co – jakby nie patrzeć – jest dość dziwnym rozwiązaniem. W przypadku naszego wyświetlacza nie jest wykorzystywana cała, dostępna pamięć sterownika, więc mamy do dyspozycji 128 kolumn i 8 wierszy. Każdy, „pionowy" bajt reprezentuje 8 pikseli ekranu ułożonych w pionie, przy czym od góry występuje bit najmniej znaczący D0 aż ku dołowi, gdzie zlokalizowano bit najbardziej znaczący D7. Zapis do pamięci ekranu standardowo przebiega od lewej do prawej oraz powoduje automatyczną inkrementację adresu kolejnej kolumny, a w przypadku osiągnięcia końca wiersza, inkrementowany zostaje adres wiersza. Jest to ustawienie konfigurowalne, które może zostać zmienione poprzez wysłanie do sterownika odpowiedniego rozkazu sterującego z towarzyszącymi mu danymi. Wyświetlacz, w czym łatwo się zorientować po krótkiej analizie dostępnych wejść sterujących, nie umożliwia przeprowadzenia operacji odczytu pamięci obrazu czy ustawień rejestrów sterujących, gdyż nie dysponuje stosownym wyprowadzeniem, które przełączałoby układ ST7565R w tryb odczytu, w związku z czym jedyną operacją jaka jest w tym przypadku dostępna jest zapis rejestrów konfiguracyjnych lub zapis do pamięci obrazu. W naszej aplikacji nie stanowi to żadnego problemu, gdyż jak się później okaże korzystamy z implementacji wyłącznie dwóch funkcji rysujących obraz i czcionkę na ekranie. Trzeba jednak podkreślić, iż czasami stanowić to może spore ograniczenie, dla przykładu w przypadku chęci implementacji funkcji rysującej linie czy inne, proste elementy graficzne. W takim

Listing 1. Funkcje odpowiedzialne za realizację programowej obsługi magistrali SPI

```
//Software SPI port definitions
#define SPI_PORT PORTC
#define SPI_DDR DDRC
#define MOSI_PIN PC0
#define SCK_PIN PC1
#define CS_PIN PC3

#define RESET_SCK SPI_PORT &= ~(1<<SCK_PIN)
#define SET_SCK SPI_PORT |= (1<<SCK_PIN)

#define RESET_MOSI SPI_PORT &= ~(1<<MOSI_PIN)
#define SET_MOSI SPI_PORT |= (1<<MOSI_PIN)
#define SCK_TICK SET_SCK; RESET_SCK

#define RESET_CS SPI_PORT &= ~(1<<CS_PIN)
#define SET_CS SPI_PORT |= (1<<CS_PIN)


inline void SPIinit(void)
{
    //MOSI, SCK & CS as outputs with "0"
    SPI_DDR |= (1<<MOSI_PIN)|(1<<SCK_PIN)|(1<<CS_PIN);
}


inline void SPIsendByte(uint8_t Byte)
{
    //MSB first
    if(Byte&0x80) SET_MOSI; else RESET_MOSI; SCK_TICK;
    if(Byte&0x40) SET_MOSI; else RESET_MOSI; SCK_TICK;
    if(Byte&0x20) SET_MOSI; else RESET_MOSI; SCK_TICK;
    if(Byte&0x10) SET_MOSI; else RESET_MOSI; SCK_TICK;
    if(Byte&0x08) SET_MOSI; else RESET_MOSI; SCK_TICK;
    if(Byte&0x04) SET_MOSI; else RESET_MOSI; SCK_TICK;
    if(Byte&0x02) SET_MOSI; else RESET_MOSI; SCK_TICK;
    if(Byte&0x01) SET_MOSI; else RESET_MOSI; SCK_TICK;
}
```

Listing 2. Funkcje odpowiedzialne za realizację podstawowej komunikacji ze sterownikiem ST7565R

```
void ST7565writeData(uint8_t Data)
{
    SET_RS;
    SPIsendByte(Data);
}

void ST7565writeCommand(uint8_t
Command)
{
    RESET_RS;
    SPIsendByte(Command);
}
```

wypadku, biorąc pod uwagę fakt, iż nie mamy możliwości odczytania zawartości ekranu przed jego aktualizacją, rysowana linia lub inny prosty element graficzny „zamazywałby" informację, która znajduje się „pod nim", jeśli dotyczy tego samego „pionowego" bajta danych. Aby uporać się z tym problemem należałoby buforować całą pamięć sterownika w pamięci RAM mikrokontrolera i na tej ostatniej pamięci przeprowadzać wszelkie operacje graficzne, po czym należałby przepisywać całą jej zawartość (lub przynajmniej zmienione obszary) do pamięci sterownika ekranu. Jest to rozwiązanie wystarczające, ale bardzo wymagające, gdyż angażuje 1024 bajty pamięci RAM mikrokontrolera. Niemniej jednak, my nie korzystamy z tego rodzaju implementacji, gdyż nie ma takiej potrzeby.

Przejdźmy zatem do konkretów implementacyjnych. Zaczniemy od implementacji funkcji odpowiedzialnych za realizację programowej obsługi magistrali SPI (inicjalizacji portów i funkcji wysyłającej bajt danych), pokazanych na **listingu 1**. Kolejne dwie funkcje to podstawowe funkcje przeznaczone do komunikacji ze sterownikiem ST7565R, które pozwalają na zapis do pamięci ekranu sterownika jak i wysłanie rozkazu sterującego – zamieszczono je na **listingu 2**. Kolejna funkcja to funkcja, której wywołanie gwarantuje poprawną inicjalizację sterownika ST7565R, w ramach to której wysyłanych jest ciąg rozkazów sterujących i towarzyszących im danych tylko po to, bay ustawić szereg właściwości sprzętowych tego układu, bez których niemożliwa byłaby poprawna praca wyświetlacza. Aby jednak zrozumieć ciało tejże funkcji, niezbędna jest znajomość pliku nagłówkowego, który pokazano na **listingu 3**.

Listing 3. Listing pliku nagłówkowego sterownika ST7565R

```
#define COG_RS_PORT PORTC  //COG display port definitions
#define COG_RS_DDR DDRC
#define COG_RST_PORT PORTD
#define COG_RST_DDR DDRD
#define COG_RS_NR PC2  //RS pin (1=Data, 0=Command)
#define COG_RST_NR PD1  //RESET pin
#define RESET_RS COG_RS_PORT &= ~(1<<COG_RS_NR)
#define SET_RS COG_RS_PORT |= (1<<COG_RS_NR)
#define RESET_RST COG_RST_PORT &= ~(1<<COG_RST_NR)
#define SET_RST COG_RST_PORT |= (1<<COG_RST_NR)

//Supported commands (CMD stands for "command") and their possible settings
(without "CMD")
#define CMD_DISPLAY_OFF 0xAE
#define CMD_DISPLAY_ON 0xAF
#define CMD_SET_DISP_START_LINE 0x40
#define CMD_SET_PAGE 0xB0
#define CMD_SET_COLUMN_MSB 0x10
#define CMD_SET_COLUMN_LSB 0x00
#define CMD_SET_ADC_NORMAL 0xA0
#define CMD_SET_ADC_REVERSE 0xA1
#define CMD_SET_DISP_NORMAL 0xA6
#define CMD_SET_DISP_REVERSE 0xA7
#define CMD_SET_ALL_PIXELS_NORMAL 0xA4
#define CMD_SET_ALL_PIXELS_ON  0xA5
#define CMD_SET_BIAS_1_9 0xA2
#define CMD_SET_BIAS_1_7 0xA3
#define CMD_READ_MODIFY_WRITE_START 0xE0
#define CMD_READ_MODIFY_WRITE_END 0xEE
#define CMD_INTERNAL_RESET 0xE2
#define CMD_SET_COM_NORMAL 0xC0
#define CMD_SET_COM_REVERSE 0xC8
#define CMD_SET_POWER_CONTROL 0x28
    #define BOOSTER_OFF (0<<2)
    #define BOOSTER_ON (1<<2)
    #define VOLTAGE_REGULATOR_OFF (0<<1)
    #define VOLTAGE_REGULATOR_ON (1<<1)
    #define VOLTAGE_FOLLOWER_OFF (0<<0)
    #define VOLTAGE_FOLLOWER_ON (1<<0)

#define CMD_SET_V0_RESISTOR_RATIO 0x20  //0...7
#define CMD_SET_ELECTR_VOLUME_MODE 0x81  //0...3F
#define CMD_SET_STATIC_INDICATOR_OFF 0xAC
#define CMD_SET_STATIC_INDICATOR_ON 0xAD
#define CMD_SET_BOOSTER_RATIO 0xF8
    #define BOOSTER_234 0x00
    #define BOOSTER_5 0x01
    #define BOOSTER_6 0x03

#define CMD_NOP 0xE3
#define CMD_TEST 0xF0
```

Nie będę wdawał się w tym miejscu w szczegóły implementacyjne, gdyż łatwo je znaleźć w dokumentacji sterownika, w związku z czym przejdę od razu do obiecanej funkcji inicjalizacyjnej, którą pokazano na **listingu 4**.

Kolejne funkcje to trzy niewielkie funkcje narzędziowe, które umożliwiają: ustawienie adresu w pamięci ekranu, od którego zacznie

Listing 4. Funkcja odpowiedzialna za inicjalizację sterownika ST7565R

```
void ST7565init(void)
{
    SPIinit();  //MOSI, SCK & CS as outputs with '0'
    COG_RS_DDR |= (1<<COG_RS_NR);  //RESET, RS as outputs with '0'
    COG_RST_DDR |= (1<<COG_RST_NR);
    RESET_RST; _delay_ms(5); SET_RST; _delay_ms(5);  //Hardware RESET
    //Hardware configuration
    ST7565writeCommand(CMD_SET_ADC_REVERSE);  //ADC Select = Reverse
    ST7565writeCommand(CMD_SET_COM_NORMAL);  //COM scan direction = Normal
    ST7565writeCommand(CMD_SET_DISP_NORMAL);  //LCD mode = Normal
    ST7565writeCommand(CMD_SET_ALL_PIXELS_NORMAL);  //Pixels mode = Normal
    //Turn on voltage converter (VC=1, VR=0, VF=0)
    ST7565writeCommand(CMD_SET_POWER_CONTROL|BOOSTER_ON);
    _delay_ms(50);
    //Turn on voltage regulator (VC=1, VR=1, VF=0)
    ST7565writeCommand(CMD_SET_POWER_CONTROL|BOOSTER_ON|VOLTAGE_REGULATOR_ON);
    _delay_ms(50);
    //Turn on voltage follower (VC=1, VR=1, VF=1)
    ST7565writeCommand(CMD_SET_POWER_CONTROL|BOOSTER_ON|VOLTAGE_REGULATOR_
ON|VOLTAGE_FOLLOWER_ON);
    _delay_ms(10);
    ST7565writeCommand(CMD_SET_BIAS_1_9);  //LCD drive voltage bias = 1/9
    ST7565writeCommand(CMD_SET_V0_RESISTOR_RATIO|6);  //Voltage Regulator
Internal Resistor Ratio Set (0x00~0x07)

    ST7565writeCommand(CMD_SET_ELECTR_VOLUME_MODE);  //Electronic volume mode
set
    ST7565writeCommand(0x14);  //0x01~0x3F
    ST7565writeCommand(CMD_SET_BOOSTER_RATIO);  //Boster ratio select
    ST7565writeCommand(BOOSTER_234);  //Booster ratio = 2x,3x,4x
    ST7565writeCommand(CMD_SET_DISP_START_LINE|0);  //Display start address =
0
    ST7565writeCommand(CMD_SET_PAGE|0);  //Page = 0
    ST7565writeCommand(CMD_SET_COLUMN_MSB|0);  //Column MSB = 0
    ST7565writeCommand(CMD_SET_COLUMN_LSB|0);  //Column LSB = 0
    ST7565writeCommand(CMD_DISPLAY_ON);  //Display ON
}
```

się najbliższy zapis do tejże pamięci (numeru kolumny i wiersza), ustawienie kontrastu wyświetlacza LCD oraz wyczyszczenie pamięci ekranu – pokazano je **listingu 5**.

Pora na funkcję, która umożliwia wyświetlenie obrazka na ekranie wyświetlacza. Zdziwicie się, jaka łatwa jest jej implementacja. Jednym z jej argumentów jest wskaźnik do tablicy w pamięci *Flash* (*const uint8_t *Bitmap*), gdzie znajduje się „treść" obrazka. Pierwsze dwa bajty tejże tablicy to szerokość i wysokość obrazka wyrażona w pikselach (wysokość musi być wielokrotnością cyfry 8, co wynika z organizacji pamięci obrazu), zaś pozostałe bajty reprezentują treść obrazu widzianą w ten sam sposób, w jaki zorganizowana jest pamięć sterownika ST7565R. Tę funkcję pokazano na **listingu 6**.

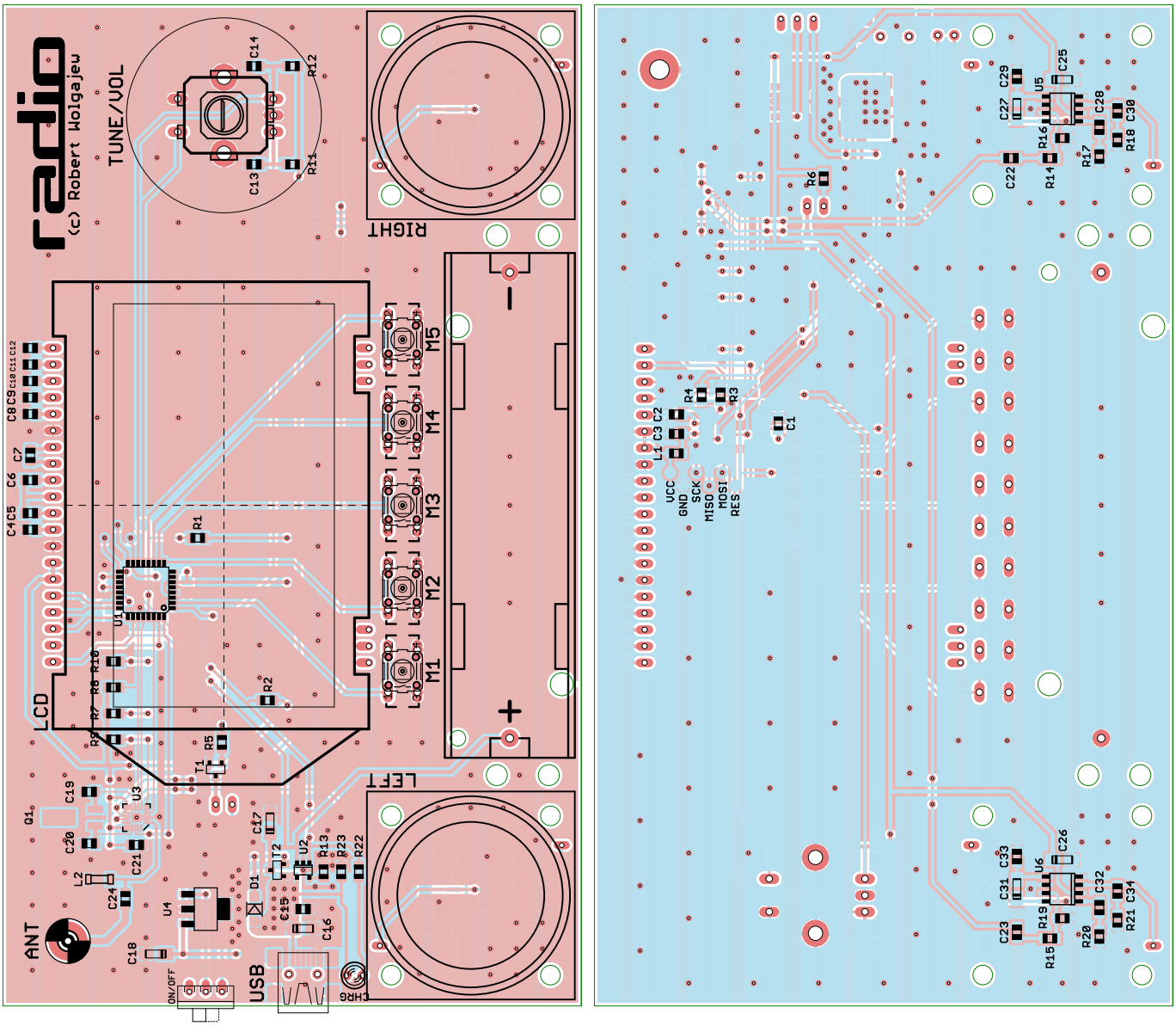
Na koniec funkcja odpowiedzialna za wyświetlanie znaków na ekranie wyświetlacza ze sterownikiem ST7565R. Zanim jednak przejdę do jej implementacji przedstawię strukturę danych, która przechowuje wszystkie, niezbędne z punktu widzenia obsługi czcionek, zmienne i której wprowadzenie znacznie ułatwia pracę z takimi czcionkami. Na jej bazie powołana zostanie globalna zmienna *static fontDescription CurrentFont*, która będzie przechowywała parametry aktualnie wybranej czcionki. Wspomnianą strukturę pokazano na **listingu 7**.

Myślę, że nazwy i opis poszczególnych pól tejże struktury dobitnie wyraża pełnione przez nie funkcje, przez co zbyteczne jest ich dodatkowe tłumaczenie. Dodam tylko, że stosowne, gotowe struktury danych zawierające wzorce czcionek o różnym wyglądzie wygenerować możemy używając kilku, dostępnych w Internecie darmowych jak i płatnych aplikacji, których pozwolę sobie nie reklamować w tym miejscu. Na **listingu 8** przedstawiono z kolei funkcję narzędziową, za pomocą której ustawimy wszystkie pola zmiennej globalnej odpowiedzialnej za przechowywanie parametrów aktualnej czcionki ekranowej, a której argumentem jest wskaźnik do struktury tego typu przechowywanej w pamięci Flash mikrokontrolera. Na **listingu 9** zamieszczono funkcję odpowiedzialną za wyświetlenie pojedynczego znaku za pomocą aktualnie używanej czcionki, natomiast na **listingu 10** funkcję odpowiedzialną za wyświetlenie ciągu znaków z pamięci RAM mikrokontrolera.

To tyle, jeśli chodzi o obsługę naszego, wydawałoby się niepozornego, acz bardzo ciekawego wyświetlacza LCD, w związku z czym idźmy dalej.

Montaż

Schemat montażowy radioodbiornika pokazano na **rysunku 4**. Zaprojektowano bardzo zwartą konstrukcję obwodu drukowanego ze zdecydowaną przewagą niewielkich elementów SMD po to, aby całe urządzenie wymiarami przypominało małe radyjko, co czyni jest mobilnym a zarazem atrakcyjnym wizualnie. Z uwagi na zastosowanie niewielkich elementów SMD o dość kłopotliwych obudowach, montaż tego typu układu najlepiej jest przeprowadzić z użyciem stacji lutowniczej typu Hot Air, dobrych topników



Rysunek 4. Schemat montażowy urządzenia Radio

Listing 5. Ciała funkcji narzędziowych sterownika ST7565R
```
void glcdGotoXY(uint8_t Column, uint8_t Page)
{
    glcdColumn = Column; glcdPage = Page;  //Variables glcdColumn/glcdPage are global variables of the module
    ST7565writeCommand(CMD_SET_PAGE|(Page & 0x07));  //Page = 0~7
    ST7565writeCommand(CMD_SET_COLUMN_MSB|(Column>>4));  //Column A7~A4
    ST7565writeCommand(CMD_SET_COLUMN_LSB|(Column & 0x0F));  //Column A3~A0
}

void glcdSetContrast(uint8_t Contrast)
{
    ST7565writeCommand(CMD_SET_ELECTR_VOLUME_MODE);
    ST7565writeCommand(Contrast & 0x3F);
}

void glcdCls(void)
{
    for(uint8_t Page = 0; Page < 8; Page++)
    {
        glcdGotoXY(0, Page);
        for(uint8_t Column = 0; Column<SCREEN_WIDTH; Column++) ST7565writeData(0x00);
    }
}
```

Listing 6. Funkcja odpowiedzialna za wyświetlenie obrazka na ekranie wyświetlacza LCD
```
void glcdDrawBitmap(uint8_t X, uint8_t Y, const uint8_t *Bitmap)
{
    register uint8_t Width, Height;
    Width = pgm_read_byte(Bitmap++);  //The first byte is picture width (in pixels)
    Height = pgm_read_byte(Bitmap++)>>3;  //The second byte is picture height (in pixels). Must be 8, 16, 24 etc.
    for(uint8_t Page = 0; Page < Height; Page++)
    {
        glcdGotoXY(X, Y+Page);
        for(uint8_t Column = 0; Column < Width; Column++) ST7565writeData(pgm_read_byte(Bitmap++));
    }
}
```

Listing 7. Struktura danych (i deklaracja stosownej zmiennej) przechowująca parametry aktualnie używanej czcionki
```
typedef struct
{
    uint8_t Width;  //Font width (pixels)
    uint8_t Height;  //Font height (bytes!)
    uint8_t Interspace;  //Interspace width (pixels)
    uint8_t BytesPerChar;  //Bytes per character definition
    uint8_t FirstCharCode;  //The ASCII code of the first character in the font data array
    const uint8_t *Bitmap;  //Pointer to the array describing subsequent characters
} fontDescription;
static fontDescription CurrentFont
```

Listing 8. Ciało funkcji narzędziowej odpowiedzialnej za ustawienie parametrów aktualnej czcionki ekranowej
```
void glcdSetFont(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);
}
```

lutowniczych oraz dysponując sporym doświadczeniem w tej kwestii. Dotyczy to zwłaszcza układu scalonego radioodbiornika Si4703, którego niewielka obudowa o wymiarach 3 mm×3 mm ma 20 wyprowadzeń umieszczonych pod spodem i na obrysie obudowy. Jak zwykle, montaż zaczynamy od przylutowania wszystkich układów scalonych po obu stronach laminatu oraz gniazda USB. Następnie lutujemy pozostałe elementy półprzewodnikowe, rezystory, kondensatory, pozostałe elementy bierne a na końcu przyciski M1…M5, koszyczek do umieszczenia akumulatora MR18650 oraz enkoder TUNE/VOL. Miniaturowe głośniczki LEFT/RIGHT mocujemy do płytki urządzenia za pomocą tulei dystansowych (4 sztuki dla każdego głośnika) zaś same ich połączenie wykonujemy kawałkiem drutu miedzianego.

Z uwagi na zagęszczenie wyprowadzeń układów scalonych, przed pierwszym włączeniem zasilania należy jeszcze raz sprawdzić jakość wykonanych połączeń, by nie dopuścić do ewentualnych zwarć. Wspomniana kontrola będzie znacznie łatwiejsza, jeśli zmontowaną płytkę przemyjemy alkoholem izopropylowym w celu wypłukania nadmiaru kalafonii lutowniczej.

Na samym końcu, do tak przygotowanej płytki, montujemy wyświetlacz LCD (zaopatrując go wcześniej w moduł podświetlenia przyklejony od spodu) zwyczajnie lutując jego wyprowadzenia w przeznaczone do tego celu pola lutownicze, gdyż połączenia te zapewniają mu jednocześnie wystarczający montaż mechaniczny. Poprawnie zmontowany układ powinien działać od razu po podłączeniu zasilania.

Obsługa

Jako, że radio jest z założenia przenośne, które to może być obsługiwane w nieoptymalnych warunkach rzeczywistych, ergonomia i prostota obsługi układu jak i czytelność interfejsu użytkownika była podstawowym kryterium przy konstruowaniu stosownych procedur sterujących. Zgodnie z tymi podstawowymi założeniami, na płytce sterownika przewidziano wyłącznie 6 przycisków sterujących dających bezpośredni dostęp do realizowanej przez nie funkcjonalności. Jak łatwo się domyślić, przyciski umownie oznaczone jako M1…M5 służą do wywołania uprzednio zapamiętanej stacji radiowej, do zapamiętania bieżącej częstotliwości w wybranym banku danych nieulotnej pamięci mikrokontrolera (przy długim wciśnięciu – ok. 500 ms) lub do uruchomienia procedury przeszukiwania pasma radiowego

Listing 9. Ciało funkcji odpowiedzialnej za wyświetlenie pojedynczego znaku aktualnie używaną czcionką

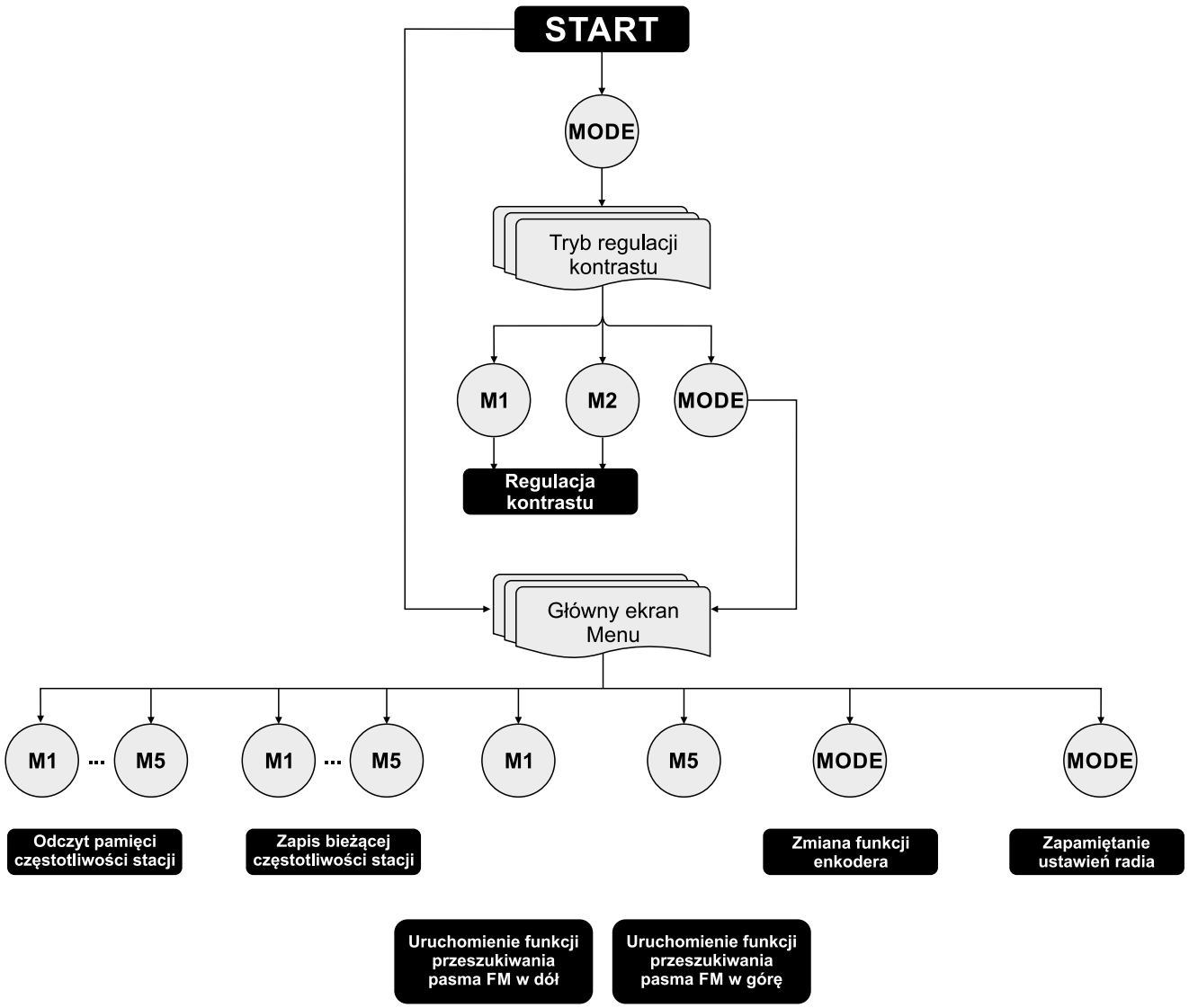
```
void glcdDrawChar(char Character, uint8_t Inversion)
{
    register uint8_t readByte, positionX = glcdColumn, positionY = glcdPage;
    const uint8_t *dataPointer;
    //Now we calculate pointer to the character's data
    dataPointer = &CurrentFont.Bitmap[(CurrentFont.BytesPerChar*(Character-CurrentFont.FirstCharCode))];
    for(uint8_t Page = 0; Page < CurrentFont.Height; Page++)
    {
        glcdGotoXY(positionX, positionY++);
        for(uint8_t Column = 0; Column < CurrentFont.Width; Column++)
        {
            if(Character == , ,) readByte = 0x00; else readByte = pgm_read_byte(dataPointer++);
            ST7565writeData(readByte);
        }
    }
}
```

Ustawienie fusebitów:
CKSEL3...0: 1111
SUT1...0: 11
CKDIV8: 1
CKOUT: 1
DWEN: 1
EESAVE: 0

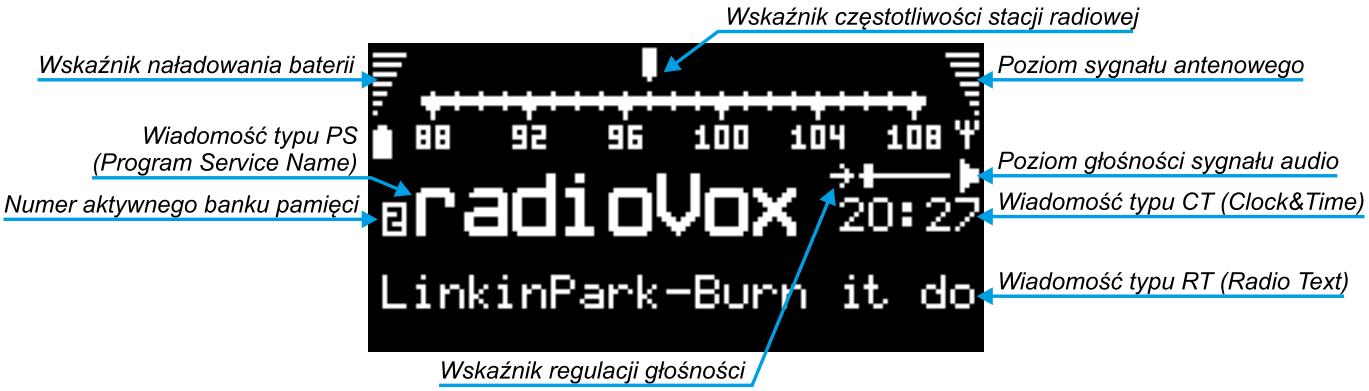
Listing 10. Ciało funkcji odpowiedzialnej za wyświetlenie ciągu znaków z pamięci RAM mikrokontrolera

```
void glcdDrawString(uint8_t Column, uint8_t Page, char *String)
{
    glcdGotoXY(Column, Page);
    while(*String)
    {
        glcdDrawChar(*String++, Inversion);
        //Now we have to calculate the new column address where the
        //next character will be displayed
        glcdColumn += CurrentFont.Width + CurrentFont.Interspace;
    }
}
```

FM w poszukiwaniu aktywnej stacji FM (wyłącznie M1 lub M5, bardzo długie wciśnięcie, powyżej 1000 ms), natomiast przycisk zintegrowany w ośce enkodera, umownie oznaczony jako MODE, służy do zmiany funkcjonalności realizowanej przez tenże enkoder. Każdorazowe jego przyciśnięcie powoduje zmianę rodzaju regulacji, jakie powodować będzie pokręcanie osią enkodera, co sygnalizowane jest w ramach graficznego interfejsu użytkownika poprzez pokazanie ikonki „strzałki" obok suwaka reprezentującego poziom głośności, w przypadku, gdy aktywną funkcją pokrętła enkodera jest, co oczywiste, regulacja głośności. W innym wypadku pokręcanie ośki enkodera zmieniać będzie



Rysunek 5. Diagram obrazujący system Menu i sposób obsługi układu Radio



Rysunek 6. Wygląd interfejsu użytkownika układu Radio

częstotliwość stacji radiowej. Kompletny diagram obrazujący funkcjonalność systemu Menu jak również sposób obsługi urządzenia pokazano na rysunku 5 (symbole przycisków wypełnione kolorem szarym oznaczają długie naciśnięcie wybranego przycisku (poniżej 500ms) a przyciski wypełnione kolorem czarnym oznaczają bardzo długie wciśnięcie danego przycisku – ponad 1000 ms), zaś na rysunku 6 pokazano wygląd interfejsu użytkownika. Dodatkowo, urządzenie Radio wyposażono w mechanizm redukcji poboru mocy, którego działanie polega na automatycznym przyciemnianiu podświetlenia wyświetlacza LCD po czasie około 10 sekund bezczynności (braku działań po stronie użytkownika) i całkowitym wyłączeniu tegoż podświetlenia po kolejnych 30 sekundach. Aktywacja podświetlenia następuję z chwilą przyciśnięcia dowolnego z elementów sterujących. Dodatkową, ostatnią funkcjonalnością, w jaką wyposażono nasze urządzenie, jest możliwość regulacji kontrastu wbudowanego wyświetlacza LCD. Wywołanie tej funkcji możliwe jest wyłącznie podczas włączania urządzenia poprzez naciśnięcie i przytrzymanie przycisku zintegrowanego w ośce enkodera w czasie startu programu obsługi. Po wykonaniu tej czynności na ekranie urządzenia pokazana będzie bieżąca wartość kontrastu, którą to zmieniamy przy pomocy przycisków M1 (in minus) i M2 (in plus) obserwując jednocześnie efekty działania stosownych zmian na ekranie urządzenia. Wyjście z tej opcji, któremu towarzyszy zapamiętanie bieżącego kontrastu wyświetlacza LCD w nieulotnej pamięci EEPROM mikrokontrolera, możliwe jest poprzez ponowne przyciśnięcie przycisku zintegrowanego w ośce enkodera, co spowoduje przejście do Menu głównego interfejsu użytkownika. Na koniec, już tylko dla porządku, dodam, iż informacje tekstowe przesyłane dzięki systemowi RDS w postaci grupy typu Radio Text prezentowane są w formie automatycznie przewijanego tekstu w dolnej części ekranu.

Robert Wołgajew, EP