BIBLIOTEKA
POLSKIEGO KRÓTKOFALOWCA

24

Krzysztof Dąbrowski
OE1KDA

Raspberry Pi
w krótkofalarstwie

Wiedeń 2014

Raspberry Pi w krótkofalarstwie

Krzysztof Dąbrowski OE1KDA

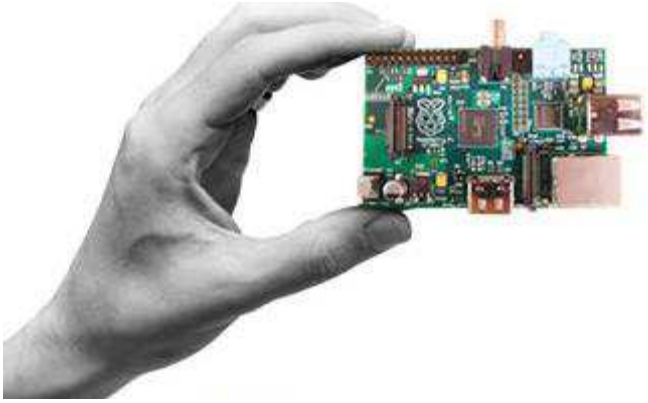
Wydanie 1

Wiedeń, sierpień 2014

Spis treści

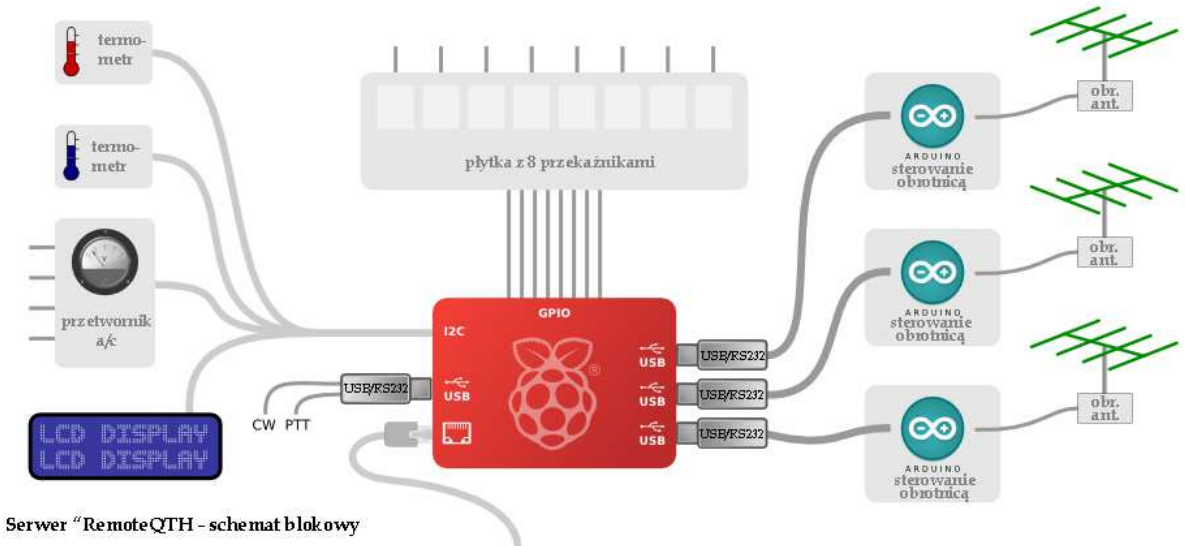
Wstęp

W wielu zastosowaniach amatorskich miniaturowe komputery w rodzaju „Raspberry Pi”, „Cubieboard”, „Beagleboard” czy „Banana Pi” stanowią praktyczną alternatywę w stosunku do rozbudowanych stacjonarnych komputerów PC albo nawet do zwykłych przenośnych. Ich cennymi zaletami są przystępna cena – dla najtańszych nawet poniżej 200 złotych, pobór mocy około 3 W (w porównaniu



z co najmniej kilkudziesięcioma watami dla PC), małe wymiary, oraz pochodny od Linuksa system operacyjny pozwalający na łatwe – łatwiejsze aniżeli w przypadku „Arduino” – podłączenie urządzeń peryferyjnych i to tylko tych, które akurat są potrzebne, a nie wszelkich możliwych jak w PC. Mniej doświadczeni użytkownicy powinni przed rozpoczęciem pracy na „Raspberry Pi” zdobyć z literatury, np. z poz. [6], pewne minimum wiedzy zarówno o samym mikrokomputerze jak i o Linuksie. Zawarte w rozdziale pierwszym krótkie zestawienie podstawowych informacji o mikrokomputerze, systemie operacyjnym, jego obsłudze i niektórych krokach konfiguracyjnych ma charakter encyklopedyczny i w żadnym wypadku nie jest pomyślane jako namiastka systematycznego kursu dla użytkowników „Raspberry Pi” i Linuksa w jakiejkolwiek postaci. Ma ono jedynie ułatwić zrozumienie zamieszczonych w dalszych rozdziałach sposobów instalacji i uruchamiania programów.
„Raspberry”, przewidziany początkowo dla szkół zyskał sobie do tego stopnia sympatię krótkofalowców, że w wielu krajach używana jest jego nazwa we własnych językach, przynajmniej wymiennie z oficjalną „Raspberry Pi” – po polsku i czesku „Malina”, po niemiecku „Himbeere” itd.
Podstawowe informacje o systemie D-STAR i sposobach korzystania z jego możliwości są zawarte w tomie 1 niniejszej serii, a informacje uzupełniające o transmisji danych, programach służących do niej w tym programie D-RATS – w tomach 2 i 15. Systemom APRS/DPRS poświęcony jest tom 8, a Echolinkowi – tom 19. Pełny spis dotychczas opublikowanych pozycji znajduje się na końcu skryptu.

Wyposażenie i moc obliczeniowa „Raspberry” przewyższają znacznie możliwości „Aduino” ale nie znaczy to, że „Arduino” grozi odejście w zapomnienie. Wprost przeciwnie – dziedziny zastosowań obu rodzajów komputerów na tyle się różnią, że każdy znajduje rację bytu w swojej klasie. Oprócz tego coraz częściej spotykane są rozwiązania kompleksowe, w których „Raspberry” steruje pracą jednego lub wielu mikrokomputerów „Arduino” wykonujących ściśle ograniczone, przewidziane dla nich zadania (przykład w witrynach [31], [32]).



Serwer “RemoteQTH - schemat blokowy

Opisany tam serwer „Remote QTH” pozwala na zdalne sterowanie własną radiostacją i antenami przez internet, odczyt danych telementrycznych i obrazów z kamer internetowych. Konstruktor oferuje odpowiednie płytki drukowane i oprogramowanie. Prywatne serwery na „Raspberry” kontaktują się z centralnym serwerem „RemoteQTH” podając mu swój aktualny adres IP co uniezależnia je od znanych usług internetowych jak *dyndns* czy *no-ip*.

Poza tym „Arduino” nie posiada własnego systemu operacyjnego (w pamięci procesora znajduje się jedynie prosty program ładujący) i udostępnia wszystkie zasoby programom użytkowym. Pozwala to na wykorzystanie go w takich zastosowaniach czasu rzeczywistego jak generacja różnego rodzaju sygnałów. W odróżnieniu od niego praca „Maliny” odbywa się pod kontrolą wielozadaniowego systemu operacyjnego (będącego pochodną Linuksa) nie spełniającego wymogów pracy w czasie rzeczywistym. Nie wyklucza to wprawdzie wszystkich zastosowań w czasie rzeczywistym, ale niektóre z nich mogą nie działać w sposób zgodny z oczekiwaniami lub też nie zawsze.

W skrypcie przedstawiono najbardziej interesujące, zdaniem autora, krótkofalarskie zastosowania „Maliny” ale nie wyczerpuje to oczywiście wszystkich możliwości. W literaturze i w Internecie często pojawiają się nowe rozwiązania i pomysły, a dzięki dostępności modułów rozszerzeń gama zastosowań jest praktycznie nieograniczona. Także wiele konstrukcji o charakterze ogólnym, j.np. różnego rodzaju serwery – FTP, HTTP, wydruku itd. – może znaleźć zastosowanie w krótkofalarstwie.
Teoretycznie możliwe jest wprawdzie wykorzystanie „Maliny” w sposób klasyczny w połączeniu z monitorem, klawiaturą i myszą np. do pracy emisjami cyfrowymi (z programem fldigi itp.) albo bezpośrednio do odbioru radiowego przy użyciu odbiorników programowalnych (ang. SDR) ale często prowadzi to do przeciążeń mikrokomputera i zawieszania się programów. Praktyczniej więc i wygodniej skorzystać do tego celu ze zwykłych komputerów PC. W wybranych przez autora zastosowaniach „Malina” pracuje w pewnym ukryciu pośrednicząc w wymianie danych i sterując niezbędnymi do tego celu urządzeniami.

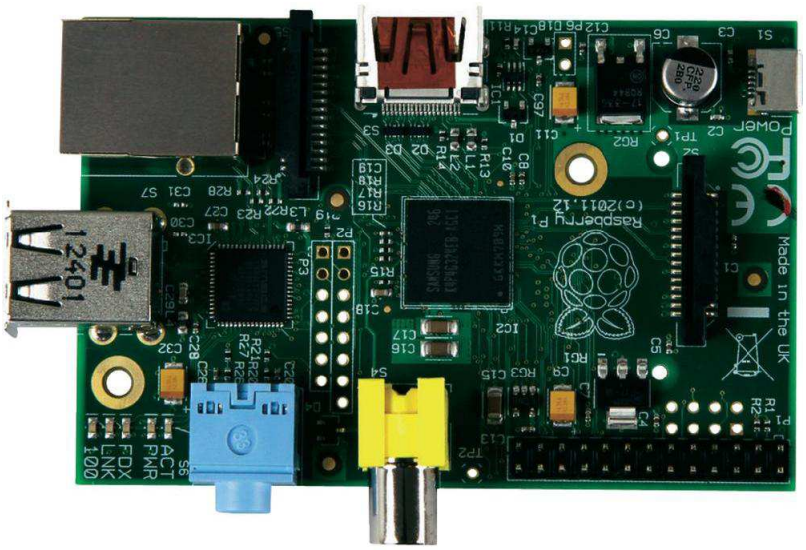
Czytelnicy mający pewne doświadczenie w programowaniu mogą uzupełnić przedstawione rozwiązania o własne funkcje np. pomiarowe, telemetryczne realizowane za pomocą pracujących równolegle skryptów *shella* albo prostych programów w Pythonie czy innym znanym języku. Przykładem takiego uzupełnienia jest program służący do wyłączania systemu po naciśnięciu przycisku.

Uwaga: w przytoczonych w skrypcie poleceniach dla systemu i skryptach na końcach zdań zawierających rozkazy dla mikrokomputera przeważnie pominięto kropkę, aby nie budzić wątpliwości czy należy ona do polecenia czy też nie. Autor prosi aby czytelnicy nie traktowali tego jako błąd. Z tego też powodu następne zdania często zaczynają się po wolnej linii. Dla lepszego odróżnienia od reszty tekstu polecenia i treść skryptów podane są kursywą.
Kody źródłowe wielu z instalowanych programów można znaleźć w internecie pod adresami, spod których „Raspberry” pobiera wersje instalacyjne np. *github.com*.

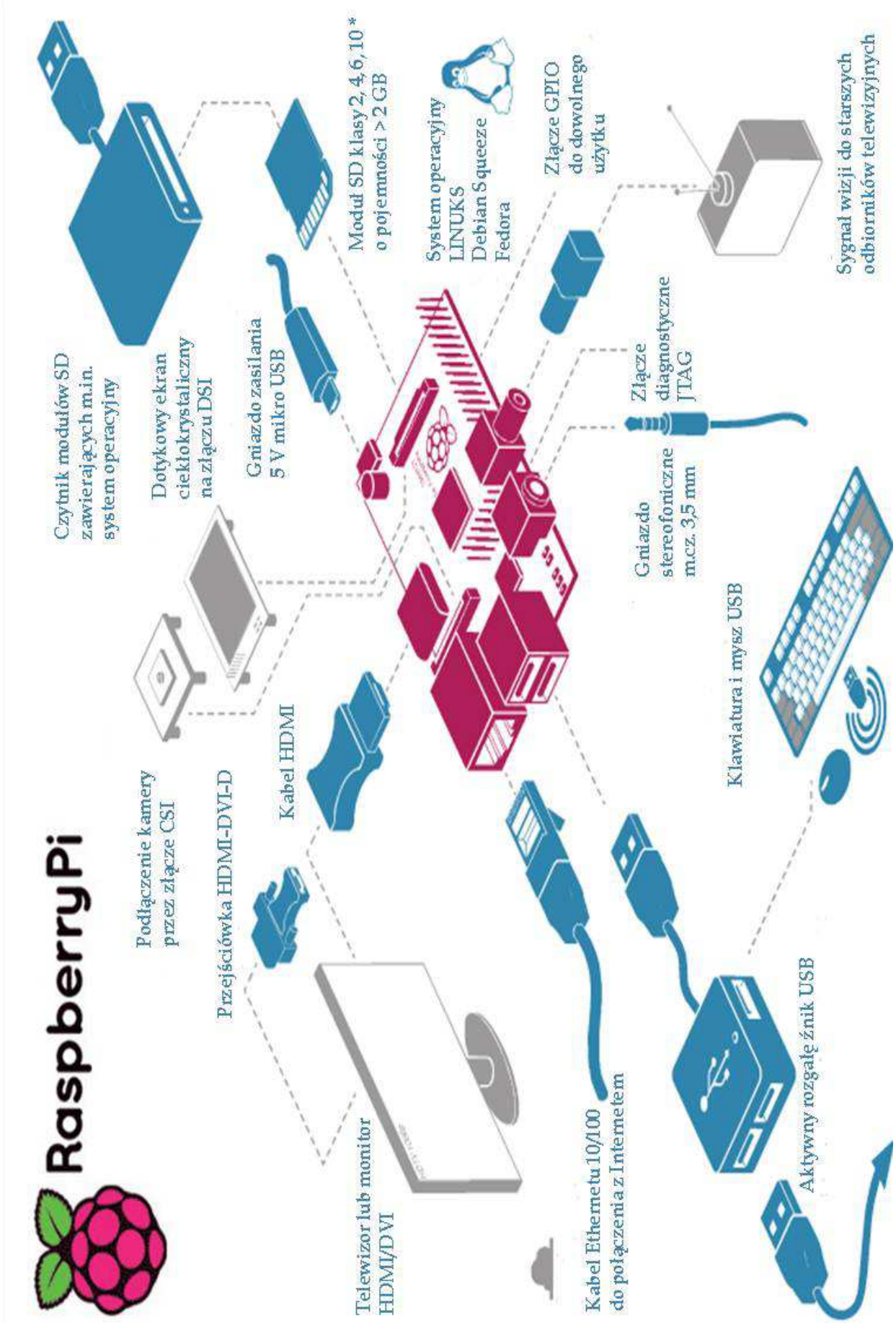
Krzysztof Dąbrowski OE1KDA
Wiedeń
Sierpień 2014

Malinowy mikrokomputer

Mikrokomputer (rys. 1.1) o wymiarach 86 x 54 x 17 mm, a więc nieco większych od paczki papierosów posiada w modelu B dwa złącza USB 2.0, złącze Ethernetu 10/100 MBit/s, wyjście HDMI, analogowe wizyjne i dźwięku, złącze logiczne GPIO o 26 kontaktach – do wykorzystania przykładowo jako złącze równoległe, szeregowe (UART), magistrale I2C, SPI, i mikrozłącze USB służące do zasilania. 32-bitowy procesor ARM11 pracujący z częstotliwością zegarową 700 MHz i 512 MB pamięci roboczej RAM pozwalają na korzystanie ze specjalnie przystosowanych wersji Linuksa – najczęściej obecnie jest to odmiana „Debiana 7" pn. „Raspbian Wheezy". System operacyjny i programy użytkowe (w tym oczywiście krótkofalarskie [1]) instalowane są w modułach pamięci SD – o pojemności 8 – 32 GB, dla których przewidziana jest osobna kieszeń (rys. 1.2). Zestawienie najważniejszych parametrów wszystkich obecnie dostępnych modeli zawiera tabela 1.1.



W większości przypadków „Malina" pracuje w zastosowaniach autonomicznych, w których klawiatura, mysz i monitor potrzebne są jedynie w fazie instalacji, uruchamiania i konfigurowania programów a następnie tylko do celów diagnostycznych bądź wymiany oprogramowania na nowsze czy bardziej odpowiadające zmienionym potrzebom. Stosowane są standardowe klawiatury i myszy USB. Możliwe jest oczywiście też zainstalowanie typowych programów „biurowych" j.np. edytora „AbiWord" itd. albo nawet gier ale jest to raczej rzadziej spotykane.
W razie potrzeby do złącza USB można podłączyć też popularne i niedrogie podsystemy dźwiękowe, modemy WiFi, krótkofalarskie modemy TNC, odbiorniki DVB-T, odbiorniki programowalne (SDR), zewnętrzne pamięci masowe – także twarde dyski – lub inne znane powszechnie urządzenia peryferyjne – przy większej liczbie przez dodatkowe rozgałęźniki (ang. *hub*) pasywne lub aktywne w zależności od sumarycznego poboru przez nie prądu.
Połączenie z Internetem odbywa się albo bezprzewodowo przez modem WiFi (dla obecnych wersji systemu operacyjnego musi on być wyposażony w obwód RTL8188CUS) albo kablowo przez złącze Ethernetu.



Rys. 1.2. Urządzenia peryferyjne i możliwości ich podłączenia

Tabela 1.1. Porównanie modeli „Raspberry Pi A”, „Raspberry Pi B” i „Raspberry Pi B+”

Parametr	Model A	Model B	Model B+
Procesor	ARM 1176 JYF-F, 700 MHz		
Pamięć RAM	256 MB	512 MB	
Złącza USB	2 x USB 2.0		4 x USB 2.0
Złacze Ethernet	---	10/100, gniazdo RJ45	
Pozostałe złącza	HDMI, wizyjne RCA, fonii 3,5 mm, mikroUSB (zasilanie), GP10, wyświetlacz DSI	HDMI, wizyjne RCA, fonii 3,5 mm, mikroUSB (zasilanie), GP10, wyświetlacz DSI	HDMI, wizyjne RCA, fonii 3,5 mm, mikroUSB (zasilanie), GP10, wyświetlacz DSI
Pamięć programu	złącze dla SDHC, MMC lub SDIO, 4 – 32 GB		złącze dla mikroSD, 4 – 32 GB
Złącze GPIO	26 kontaktów, w tym UART, I2C, SPI		40 kontaktów, w tym UART, I2C, SPI
System wizyjny	Broadcom Video Core 4		
System operacyjny	najczęściej stosowany „Wheezy/Raspbian”, alternatywnie dostępne oddzielnie lub w zestawie „NOOBS”: Archlinux, Open ELEC, Piodora, RASP BMC, RISC OS		
Zasilanie	5 V, 500 mA	5 V, 700 mA	5 V, 700 mA
Wymiary	86 x 54 x 17 mm		85 x 56 x 17 mm

Tabela 1.2. Konkurenci „Maliny”

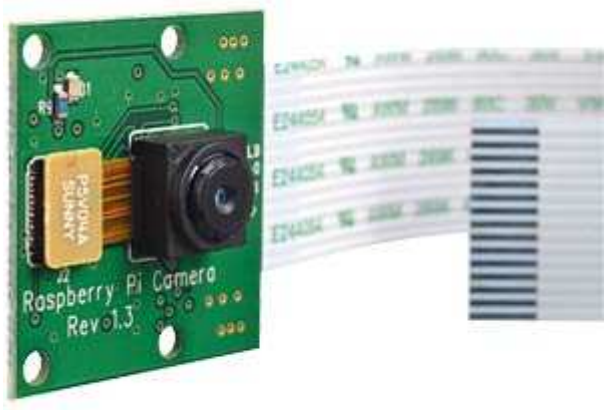
Parametr	Banana Pi	Cubieboard A20	Beagleboard
Procesor	Cortex A-7 Dual Core, 2 x 1 GHz	Cortex A-7 Dual Core, 2 x 1 GHz	ARM Cortex A-8, TI Sitara AM3358
Pamięć RAM	1 GB	1 GB	512 MB
Złącza USB	2 x USB 2.0, 1 x OTG	2 x USB 2.0, 1 x OTG	1 x USB 2.0
Złacze Ethernet	10/100/1000, gn. RJ45	10/100, gniazdo RJ45	10/100, gniazdo RJ45
Pozostałe złącza	HDMI, wizyjne RCA, fonii 3,5 mm, wejście mikrofonowe, GP10, mikroUSB (5 V, 2 A), wyświetlacz DSI, kamera CSI, SATA, łącze na podczerwień	HDMI, SATA, LINE-IN, LINE-OUT, łącze na podczerwień, gniaz-do koncentryczne do zasilania 5 V/500 mA	mikroHDMI, gniazdo koncentryczne do zasilania 5 V
Pamięć programu	złącze dla SD, MMC, maks. 64 GB	4 GB na płytce, złącze dla mikroSD	4 GB na płytce, złącze dla mikroSD
Złącze GPIO	26 kontaktów, rozkład identyczny jak w „Raspberry”	96 kont., zastosowanie dla I2C, SPI, CSI/TS, RGB/LVDS, FM-IN, ADC, CVBS, VGA, SPDIF-OUT, R-TP	2 listwy po 46 kontak-tów, zastosowanie dla I2C, SPI, UART, GPIO (65 kontaktów), LCD itd.
System wizyjny	Mali400MP2	Mali400MP2	SGX530 3D
System operacyjny	Raspbian, Android 4.4, Ubuntu, Debian	Android, Ubuntu, Fedora, Wheezy/ Raspbian	Android
Wymiary	92 x 60 mm	99 x 60 x 20 mm	86 x 53 mm
Uwagi	Praktycznie kompaty-bilny z „Raspberry Pi”		Do użytku w układach sterujących i automatyki

Tabela 1.3. Sygnalizatory – diody świecące – na płytce drukowanej. Kolejność – od środka plytki ku krawędzi.

Skrócone oznaczenie	Dioda	Kolor	Znaczenie
ACT	D5		Dostęp do pamięci SD
PWR	D6		Zasilanie
FDX	D7		Sieć LAN podłączona, pełny dupleks
LNK	D8		Wymiana danych z siecią LAN
100	D9		Szybkość 100 MB/s

Rys. 1.3. System operacyjny i modem WiFi do „Maliny”

Wśród użytkowników największą popularność zyskał sobie system operacyjny „Raspbian” występujący też pod nazwą „Wheezy”. Dzięki skutecznemu wykorzystaniu zawartego w CPU koprocesora matematycznego jest on szybszy od konkurentów i przyspiesza w ten sposób wykonywanie zainstalowanych programów. Jest on dostępny w handlu w gotowej postaci w pamięciach SDHC lub mikro SD (wymaga to użycia przejściówki z mikro SD na SDHC) ale można go również pobrać z internetu i samodzielnie zapisać do pamięci. Jest on również zawarty w niektórych dystrybucjach programów użytkowych, w tym również programów krótkofalarskich. Przykładem może być opisany dalej program dla miniprzemiennika D-Starowego „ircDDB Gateway”. Alternatywą dla „Raspbiana” może być zestaw „NOOBS”, na który składają się oprócz „Raspbiana” także „Archlinux”, „Open ELEC”, „Piodora” (odmiana „Fedory”), „RASP BMC” i „RISC OS”. Tylko część z nich jednak posiada graficzną powierzchnię obsługi. W trakcie pierwszego uruchomienia użytkownik może wybrać jeden z nich do stałego użytku. Jest on instalowany w pamięci tak, że następnie startuje już automatycznie. Możliwe jest jednak korzystanie i z pozostałych systemów. Należy tylko w trakcie startu systemu trzymać wciśnięty klawisz dużych liter (ang. *shift*) aby otrzymać okno wyboru systemu. Instalacja systemów „Windows” albo OS X nie jest możliwa.



Rys. 1.4. Miniaturowa kamera podłączana do złącza CSI

Złącze GPIO

Oprócz wymienionych powyżej standardowych złączy (USB, HDMI) „Malina” jest wyposażona w 26-kontaktowe programowalne złącze logiczne. Część kontaktów jest wykorzystywana do realizacji standardowych złączy (magistrali) I2C (nóżki 3, 5), SPI (19, 21, 23), UART-u (8, 10) ale pozostałe mogą być dowolnie użyte w programach jako wejścia lub wyjścia logiczne, w tym także wyjścia impulsów o modulowanej szerokości (PWM). W zastosowaniach krótkofalarskich mogą one służyć przykładowo do kluczowania nadajników (*PTT*), odczytu stanu blokady szumów, przełączania urządzeń pomocniczych a UART przykładowo do podłączenia modemu TNC dla APRS i packet-radio.

Oznaczenia nóżek i występujacych na nich sygnałów przedstawiono na ilustracji 1.5a, a sposób numeracji kontaktów na płytce drukowanej – na 1.5b. Część z nich jest połączona z napięciami zasilania 3,3 V, 5 V lub z masą.
Dla ułatwienia orientacji nóżki (na ilustracji ale nie w naturze) zaznaczone są różnymi kolorami w zależności od ich funkcji:

- Kolor czerwony – odpowiada napięciu 5 V,
- Pomarańczowy – napięciu 3,3 V,
- Czarny – masie,
- Zielony – nóżkom programowalnym do ogólnego użytku,
- Fioletowy – złączu SPI,
- Turkusowy – złączu I2C.

Nie zaleca się lutowania przewodów do kontaktów złącza GPIO – lepiej korzystać z pasujących wtyczek.

Uwaga:
Poziomy napięć na złączu szeregowym wynoszą 3,3 V. Podanie na nie sygnałów o napięciu 5 V lub zwarcia do masy grożą nieodwracalnymi uszkodzeniami.
Dla otrzymania standardowych poziomów napięć RS232 najlepiej wykorzystać obwód scalony MAX3232.
Prosty tranzystorowy układ dopasowania poziomów do normy TTL podano w rozdziale 2.

Rys. 1.5a. Wyprowadzenia na złączu GPIO

Rys. 1.5b. Numeracja kontaktów na złączu GPIO i jego położenie. Nóżki o numerach 1 i 2 znajdują się najbliżej krawędzi płytki

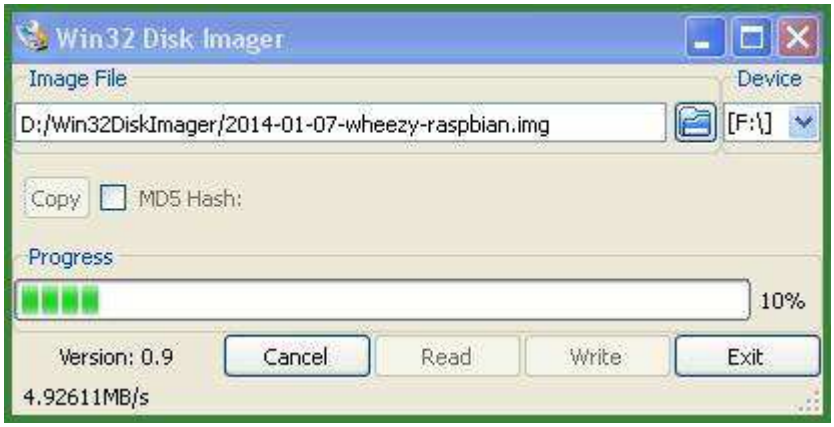
Podstawowe wiadomości o Linuksie

Celem tego podrozdziału jest przypomnienie czytelnikowi podstawowych wiadomości i elementarnych poleceń systemu Linuks i pochodnych – a zwłaszcza „Raspbiana" – w ich najczęściej stosowanych wariantach oraz zwrócenie uwagi na najważniejsze punkty konfiguracji. Nie jest on pomyślany jako systematyczny kurs korzystania z systemu – w tym celu czytelnik powinien zapoznać się z obszerniejszą i pod tym kątem opracowaną literaturą. Jedną z przykładowych pozycji przedstawiono na rys. 1.6.

W przykładach w dalszej części rozdziału autor przyjmuje dla ułatwienia domyślnego użytkownika „pi" i domyślne hasło „raspberry". Oczywiście w dobrze pojętym własnym interesie należy zmienić obie te dane, aby uniemożliwić osobom obcym dostęp do systemu – zwłaszcza jeżeli jest on połączony z Internetem.

„Malina" sprzedawana jest prawie zawsze bez systemu operacyjnego. Najwygodniej jest dokupić gotowy zainstalowany w pamięci system i po włożeniu modułu SDHC do kieszeni przystąpić od razu do jego konfiguracji (rys. 1.8). Można też pobrać z internetu na PC obraz pamięci i po zapisaniu go na dysku przepisać na moduł pamięciowy. Nie można go jednak zwyczajnie skopiować za pomocą „Eksploratora" Windows. Do tego celu konieczny jest program kopiujący tego rodzaju obrazy pamięci. Jednym z takich programów dla systemu „Windows" jest dający się szybko i bezproblemowo zainstalować „Win32 Disk Imager" (rys. 1.7). Po jego uruchomieniu należy w lewym górnym polu wybrać zapisany na twardym dysku obraz systemu (na ilustracji plik *.img* dla „Raspbiana Wheezy"), w prawym górnym literę odpowiadajacą podłączonemu do komputera czytnikowi modułów pamięciowych i nacisnąć ekranowy przycisk „**Write**". O przebiegu zapisu informuje znajdujący się w dolnej części okna wskaźnik paskowy. Zapis pliku w pamięci zewnętrznej powoduje jej sformatowanie i utratę wszystkich zawartych w niej danych dlatego należy zwrócić szczególną uwagę na wybranie właściwej litery.

Ten sam program może służyć także do odczytu zawartości modułu pamięci i do jego zapisu w postaci pliku *.img*. Wymaga to, jak się łatwo domyślić, naciśnięcia przycisku „**Read**" – oczywiście po uprzednim wyborze litery odpowiadajacej czytnikowi i podaniu nazwy pliku (np. *kopia_8_8_2014.img*). Przycisk „**Cancel**" służy do przerwania zapisu lub odczytu, a „**Exit**" do wyłączenia programu.

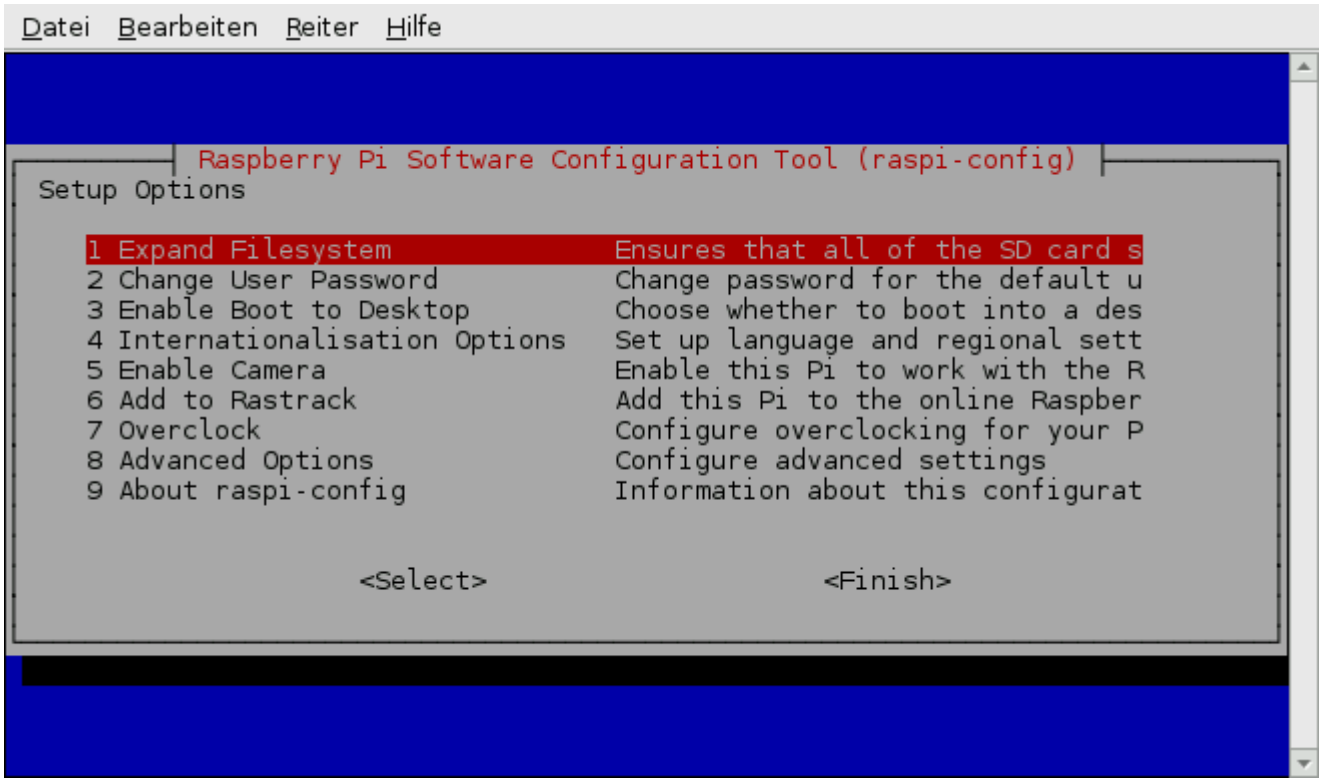


Rys. 1.7. Okno programu „Win32 Disk Imager" w trakcie kopiowania

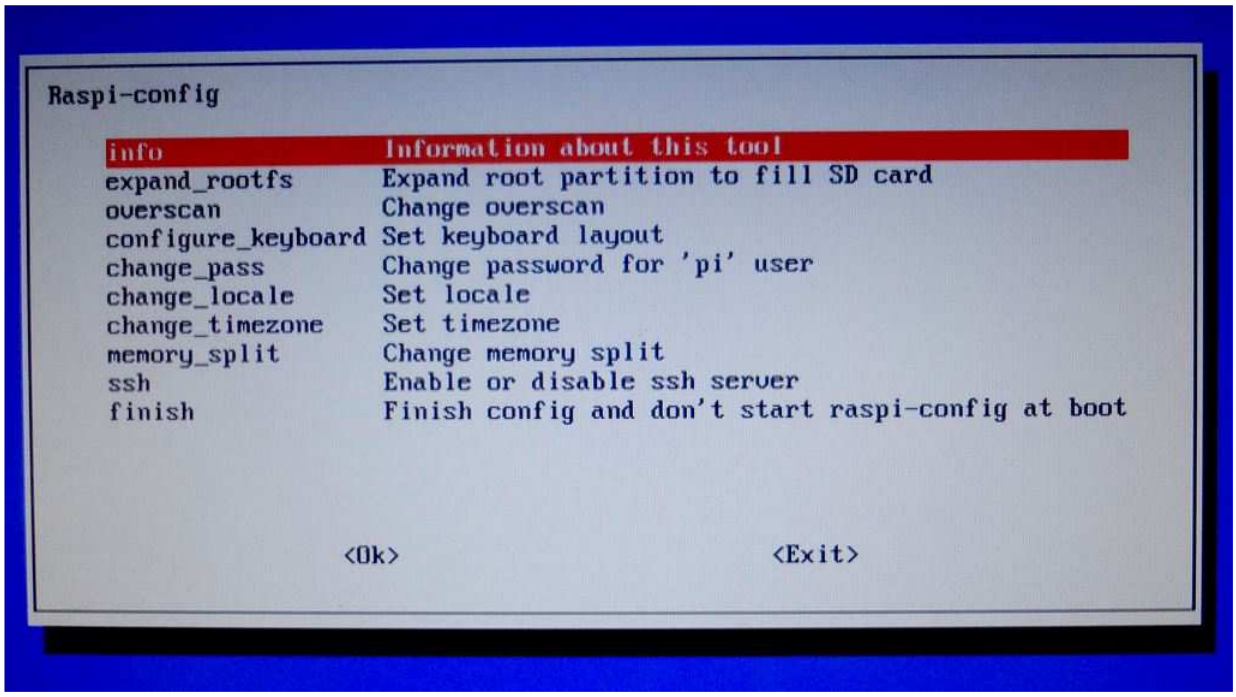
„Raspbian" zajmuje około 2 GB pamięci, ale ponieważ konieczne jest jeszcze miejsce na programy użytkowe i aktualizacje systemu moduł pamięci powiniem mieć pojemność co najmniej 4 GB, a w przypadku instalowania większej ilości lub obszerniejszych programów korzystnie jest stosować

moduły o pojemnościach od 8 GB wzwyż. Różnica cen modułów 4 i 8 GB jest już obecnie praktycznie nieistotna.
Po pierwszym uruchomieniu systemu na „Raspberry” konieczne jest przeprowadzenie jego podstawowej konfiguracji. Wymaga to uprzedniego podłączenia klawiatury i monitora.
Oczywiście program konfiguracyjny można wywoływać również później w miarę potrzeb. Do tego celu służy polecenie:
sudo raspi-config

Szczegółowe omówienie wszystkich, widocznych na rys. 1.8a, punktów konfiguracji wykraczałoby poza ramy nimiejszej publikacji, dlatego też autor pragnie zwrócić tylko uwagę na niektóre punkty.
W punkcie 2 należy ze względów bezpieczeństwa zmienić hasło dostępu, w punkcie 4 – dopasować układ klawiatury i w punkcie 7 – włączyć serwer SSH zapewniający następnie dostęp telnetowy przez lokalną sieć bez konieczności każdorazowego podłączania monitora i klawiatury. Punkt 3 pozwala na automatyczne wywoływanie graficznej powierzchni obsługi w momencie startu. Z poziomu wiersza poleceń można ją wywołać w dowolnym momencie za pomocą polecenia *startx*. Nie zaleca się natomiast podwyższania częstotliwości zegarowej, chociaż jest to przewidziane w konfiguracji (punkt 7). Przyspieszenie procesora może nie tylko odbić się niekorzystnie na stabilności pracy systemu ale także powoduje jego przegrzewanie się i wymaga dodatkowego chłodzenia. W handlu dostępne są wprawdzie zestawy radiatorów naklejanych na obwody scalone mikrokomputera ale lepiej i tak nie dać się skusić pozornym korzyściom wynikającym z przyspieszenia pracy.
W przypadkach rzeczywiście wymagających większej szybkości pracy i mocy przerobowej lepiej rozważyć zastosowanie innego modelu mikrokomputera.



Rys. 1.8a. Okno konfiguracyjne „Raspbiana”. Możliwe są różnice w wyglądzie i zestawie punktów menu konfiguracyjnego w zależności od aktualnie używanej wersji systemu.



Rys. 1.8.b.

Rys. 1.9. Graficzna powierzchnia obsługi „Raspbiana”

Widoczny na ilustracji symbol „LXTerminal” służy do otwarcia okna wiersza poleceń – odpowiadającemu oknu „cmd” (dawniejszemu oknu DOS) dla „Windows”, „WiFi Config” służy do konfiguracji bezprzewodowego dostępu do Internetu, „Midori” wywołuje przeglądarkę internetową, a „Shutdown” – służy do wyłączenia „Maliny”.
Graficzna powierzchnia obsługi z pewnością ułatwi życie użytkownikom przyzwyczajonym do korzystania z okien „Windows” ale w większości zastosowań autonomicznych jej wywołanie nie jest konie-

czne. Niektóre programy, j.np. przedstawiony dalej „DVAPTool” istnieją na razie tylko w wersji okienkowej i wymagają jej uruchomienia.
Do aktualizacji systemu po pewnym czasie korzystania z niego służy polecenie
apt-get update
podawane w wierszu poleceń a do instalowania nowych programów, bibliotek czy sterowników stosowane jest polecenie
sudo apt-get install <nazwy programów>

System plików DOS-u przejęty także przez „Windows” jest wzorowany na systemie „Unixa” dlatego też poruszanie się w nim nie powinno zasadniczo sprawiać kłopotów, jeżli tylko pamięta się, że do rozgraniczania nazw służy nie ukośnik „\” a zwykła ukośna kreska dzielenia „/”.
Standardowo dla każdego zapisanego w systemie użytkownika zakładany jest katalog początkowy znajdujący się w katalogu */home*, a więc dla użytkownika „pi” będzie to katalog */home/pi*.
Nazwa aktualnego katalogu roboczego jest wyświetlana w odpowiedzi na polecenie
pwd
a do zmiany katalogu służy polecenie
cd

W szczególności polecenie *cd* **..** (z dwoma kropkami) powoduje przejście do katalogu nadrzędnego, a pojedyńcza kropka oznacza aktualny katalog. Polecenie *cd* **.** wprowadzone ręcznie nie ma wprawdzie większego sensu ale kropka może występować w bardziej złożonych, względnych ścieżkach dostępu w różnego rodzaju skryptach przeznaczonych dla wielu użytkowników. Najwyższy w hierarchii katalog „root” reprezentuje w zapisie ścieżek ukośna kreska na początku.
Do wywołania spisu treści (zawartości) katalogu służy polecenie
ls – dające wynik w skróconej postaci.
Dla wywołania pełnego spisu wraz ze wszystkimi informacjami służy
ls –al

Standardowo po znaku minus podawane są dodatkowe parametry decydujące dokładniej o sposobie wykonania polecenia lub programu. Szczegółowe informacje o poleceniach i programach systemowych wywołuje się przez rozkaz *man* (skrót od słowa *manual*), a więc w celu zapoznania się z możliwościami i argumentami polecenia *ls* należy posłużyć się poleceniem
man ls.

Dla wyświetlenia większej ilości danych, nie mieszczących się na raz na ekranie korzysta się z rozkazu *more* poprzedzonego pionową kreską, a więc dla powyższych przykładów byłoby to
ls -al|more
albo
man ls|more

Wyświetlana jest wówczas tylko ilość danych mieszcząca się na ekranie, a do wywołania następnej porcji konieczne jest naciśnięcie klawisza „Return”, albo klawisza odstępu dla wywołania tylko jednej linii tekstu.
Do zakładania katalogów służy polecenie *mkdir <nazwa>*, do ich kasowania – *rmdir <nazwa>*, do kopiowania plików *cp <nazwa1> <nazwa2>*, do odczytania zawartości plików tekstowych –
cat <nazwa>
a do przesuwania plików do innego katalogu polecenie
mv <nazwa> <ścieżka docelowa>

Kasowania plików dokonuje się za pomocą polecenia *rm <nazwa>* albo też *mv <nazwa>* – w przeciwieństwie do poprzedniego wariantu bez podania celu.
Obecna wersja „Raspbiana” zawiera także dwa edytory tekstowe „vi” i „nano”. Użytkownicy przyzwyczajeni do komfortu edytorów Windowsowskich mogą na początku czuć się nieco zagubieni korzystając z nich.

Oprócz programów – wywoływanych za pomocą ich nazw – system wykonuje również pliki wsadowe czyli skrypty *shella*– w „Raspbianie" stosowany jest *Bash shell (bash)*. Są to pliki tekstowe zawierające przeważnie polecenia systemowe, wywołania programów albo dalszych skryptów za pomocą ich nazwy i ewentualnie koniecznych parametrów oraz proste struktury sterujące przebiegiem j.np. *if...then...else...fi, case...esac, until...done*. Znak krzyżyka „#" na początku linii oznacza komentarz. Linie takie mogą zawierać dowolny tekst i są pomijane w trakcie wykonywania skryptu. Skrypty wsadowe wzorowane na Unixowych (z rozszerzeniem *.bat*) były dawniej dość często stosowane pod systemem DOS i pierwszymi wersjami „Windows" ale ostatnio straciły na popularności.
Oprócz zwykłych użytkowników, mających ograniczone uprawnienia w Linuksie i wszystkich jego odmianach występuje użytkownik o uprawnieniach administratora, noszący nazwę „root" lub „superuser". Od tej właśnie ostatniej nazwy pochodzi występujący w niektórych cytowanych w skrypcie przykładach poleceń początkowy człon *sudo* – będący skrótem od „superuser do". Powoduje to wykonanie dalszego ciągu polecenia np. wywoływanego programu z najwyższymi uprawnieniami. Dodania nowego użytkownika dokonuje się za pomocą polecenia
sudo adduser <nazwa użytkownika>

do zmiany parametrów dla danego użytkownika
sudo usermod <nazwa użytkownika>

a do skasowania
userdel <nazwa użytkownika>

Uprawnienia dostępu do plików i katalogów są we wszystkich systemach pochodzących od Unixa ustalane trójstopniowo:
– dla właściciela,
– dla grupy, do której jest przypisany,
– i dla wszystkich pozostałych użytkowników.
Dla każdego z nich przyznawane są prawa odczytu (oznaczane literą „r" w spisie), zapisu czyli modyfikacji (oznaczane literą „w") i wykonywania (oznaczane literą „x") – co jest istotne tylko dla skryptów i programów.
W spisie zawartości katalogów przy każdym zawartym w nim pliku lub katalogu podawana jest literowa kombinacja przyznanych uprawnień np.
rwxrwxrwx – co oznaczałoby przyznanie wszystkim pełnych praw korzystania z pliku. W miejscu nie przyznanych uprawnień wyświetlany jest myślnik, np.
rwxr-xr-x – co oznacza, że tylko właściciel ma prawo modyfikować dany plik. Prawo do wykonywania pliku wymaga przyznania uprawnień do jego odczytu, a więc występowanie samej litery „x" bez litery „r" jest błędem natomiast sytuacja odwrotna może być sensowna np. dla plików tekstowych.
Do przyznania uprawnień służy polecenie *chmod <xyz>* gdzie x, y i z są liczbami w zakresie 0–7 (a wiec trzybitowymi; ósemkowymi), powstałymi przez zsumowanie wartości dwójkowych wymienionych uprawnień: „r" odpowiada wartość 4, „w" – wartość 2, a „x" – wartość 1. Kombinacji uprawnień *rwxr-xr-x* (prawo odczytu i wykonywania dla wszystkich a modyfikacji tylko przez właściciela) odpowiada więc wartość 755, a do ich nadania służy więc polecenie *chmod 755*. Polecenie *chmod 644* pozwala wszystkim na odczyt pliku niewykonywalnego pozostawiając możliwość jego modyfikacji tylko właścicielowi.
Do zakończenia pracy systemu służy polecenie
sudo shutdown –h now
a do wyzerowania i powtórnego uruchomienia
sudo shutdown –r now

Najczęściej stosowanymi w środowisku Linuksa językami programowania są tradycyjnie już C i częściowo asembler a w ostatnich czasach także Python, C++, Java, BASIC i Perl.

Tabela 1.4. Najważniejsze katalogi Linuksa i systemów pochodnych

Ścieżka	Znaczenie
/	katalog początkowy systemu
/etc	zawiera pliki konfiguracyjne systemu i poszczególnych programów, np. /etc/network/interfaces – plik konfiguracyjny złączy LAN i WLAN; /etc/rc.local – plik zawierający automatyczne wywołania wykonywany po uruchomieniu systemu, /etc/aprx.conf – plik konfiguracyjny programu aprx itd. /etc/network/interfaces – konfiguracja dostępu do Internetu /etc/hostname – zawiera nazwę własnego systemu /etc/modprobe.d/alsa-base.conf – zawiera konfigurację systemu dźwiękowego
/bin	zawiera pliki wykonywalne – skrypty lub programy
/tmp	zawiera pliki tymczasowe
/lib	mieści biblioteki systemu operacyjnego, np. /lib/libc – jedna z podstawowych bibliotek systemu
/dev	pliki reprezentujące urządzenia peryferyjne, np. /dev/ttyUSB0 – pierwsze wirtualne złącze RS232 na USB, /dev/ttyS0 – pierwsze rzeczywiste złącze szeregowe, /dev/wlan0 – pierwszy modem WiFi, /dev/eth0 – pierwsze złącze Ethernetu, /dev/soundmodem0 – pierwszy podsystem dźwiękowy, itd. Następne urządzenia otrzymują kolejne numery.
/home	katalog zbiorczy zawierający katalogi użytkowników, np. /home/pi – katalog użytkownika „pi” na „Raspberry” /home/pi/bin – katalog plików wykonywalnych użytkownika „pi’ /home/pi/bin/DVAP/DVAPTool – program DVAPTool w katalogu plików wykonywalnych użytkownika „pi“
/root	katalog użytkownika „root”
/usr	katalog dla danych używanych przez zainstalowane programy
/usr/bin	zawiera pliki wykonywalne – skrypty lub programy
/usr/lib	biblioteki programów
/usr/local/bin	zawiera pliki wykonywalne – skrypty lub programy
/srv	katalog dla danych używanych przez uruchomione usługi systemu
/sys	wirtualny katalog dla danych systemowych

Konfiguracja dostępu do Internetu

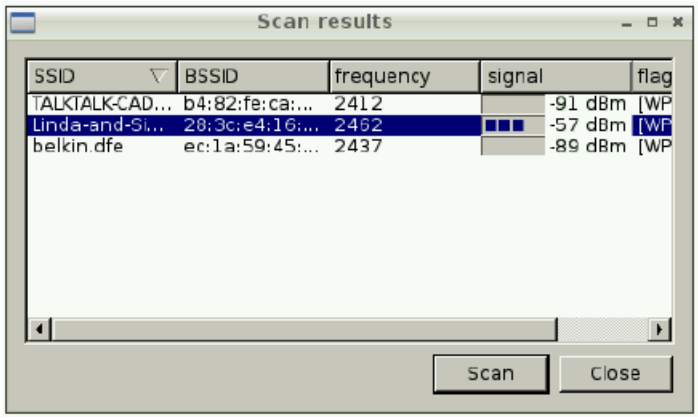
Aktualizacja systemu operacyjnego, pobranie programów użytkowych a także praca krótkofalarskich bramek APRS i przemienników echolinkowych albo D-Starowych wymaga połączenia mikrokomputera z internetem. Modele B i B+ pozwalają na połączenie kablowe poprzez złącze ethernetowe, a wszystkie trzy – na połączenie radiowe WiFi przy użyciu miniaturowego modemu USB (rys. 1.3).
Konfiguracji dostępu bezprzewodowego można dokonać w dwojaki sposób: za pomocą okienkowego programu „WiFi Config" (rys. 1.10) lub z poziomu wiersza poleceń modyfikując tekstowy plik */etc/network/interfaces*.

Rys. 1.10 Wywołanie „WiFi Config"

Konfiguracja okienkowa

Po uruchomieniu programu na ekranie wyświetlane jest widoczne po lewej stronie okno (rys. 1.11).
W pierwszym polu od góry widoczna jest systemowa nazwa modemu WiFi – tutaj „wlan0", który należy oczywiście uprzednio włączyć do gniazda USB mikrokomputera. .
Po naciśnięciu przycisku „**Scan**" („Szukaj") system rozpoczyna poszukiwanie dostępnych sieci bezprzewodowych. Spis tych sieci jest wyświetlany w oknie z rys. 1.12.
Przykłady okien pochodzą z witryny *http://learn.adafruit.com* dzięki czemu autor nie musi zdradzać szczegółów własnej konfiguracji.

Rys. 1.11. Okno główne programu konfigurującego



Po wybraniu ze spisu pożądanej sieci (własnej!) i dwukrotnym nasiśnięciu myszą na jej nazwę jest ona wpisywana do okna głównego. W oknie tym, w zakładce WPS należy wpisać hasło dostępu do sieci i nacisnąć przycisk „**Add**" („Dodaj").

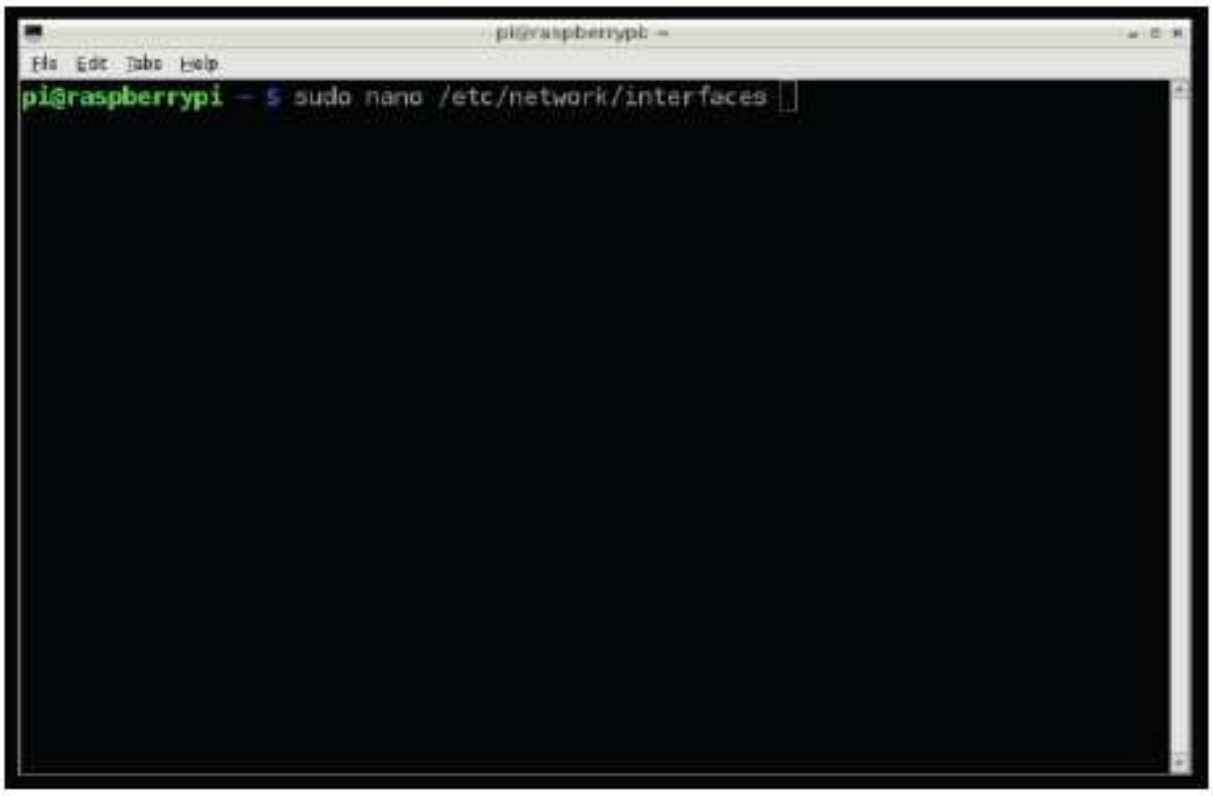
Rys. 1.12. Okno poszukiwania sieci

Przyciski „**Connect**” („Połącz”) i „**Disconnect**” („Rozłącz”) służą do wypróbowania połączenia z siecią (rys. 1.13).
Warto zapisać sobie podany w oknie adres IP, ponieważ jego znajomość może się przydać w dostępie do „Maliny” przez sieć domową.

Rys. 1.13. Okno główne po wybraniu sieci

Konfiguracja z poziomu wiersza poleceń

Po uruchomieniu mikrokomputera bez podłączenia modemu WiFi należy otworzyć okno terminalowe (rys. 1.14) „LXTerminal” (lub nie wywoływać wogóle graficznej powierzchni obsługi) i w wierszu poleceń podać następujący rozkaz (wywołujący edytor *nano*):
sudo nano /etc/network/interfaces



Rys. 1.14. Okno wiersza poleceń

Plik ten zawiera m.in. następujące, lub podobne, wpisy albo przynajmniej niektóre z nich:

auto lo
iface lo inet loopback
iface eth0 inet dhcp

allow hotplug wlan0
auto wlan0

iface wlan0 inet dhcp
wpa-ssid „nazwa sieci”
wpa-psk „hasło dostępu”

W liniach *wpa-ssid* i *wpa-psk* należy podać – w cudzysłowach – odpowiednio dane własnej sieci.

Rys. 1.15. Plik */etc/network/interfaces* w oknie edytora „nano”

Po zamknięciu edytora za pomocą kombinacji CTRL-X (na pytanie czy zapisać dane należy oczywiście odpowiedzieć „Y”) można wyłączyć „Raspberry”, włożyć modem WiFi do gniazda USB i uruchomić mikrokomputer.
W celu odczytania używanego przez „Malinę” adresu IP należy w wierszu poleceń podać rozkaz:
sudo ipconfig

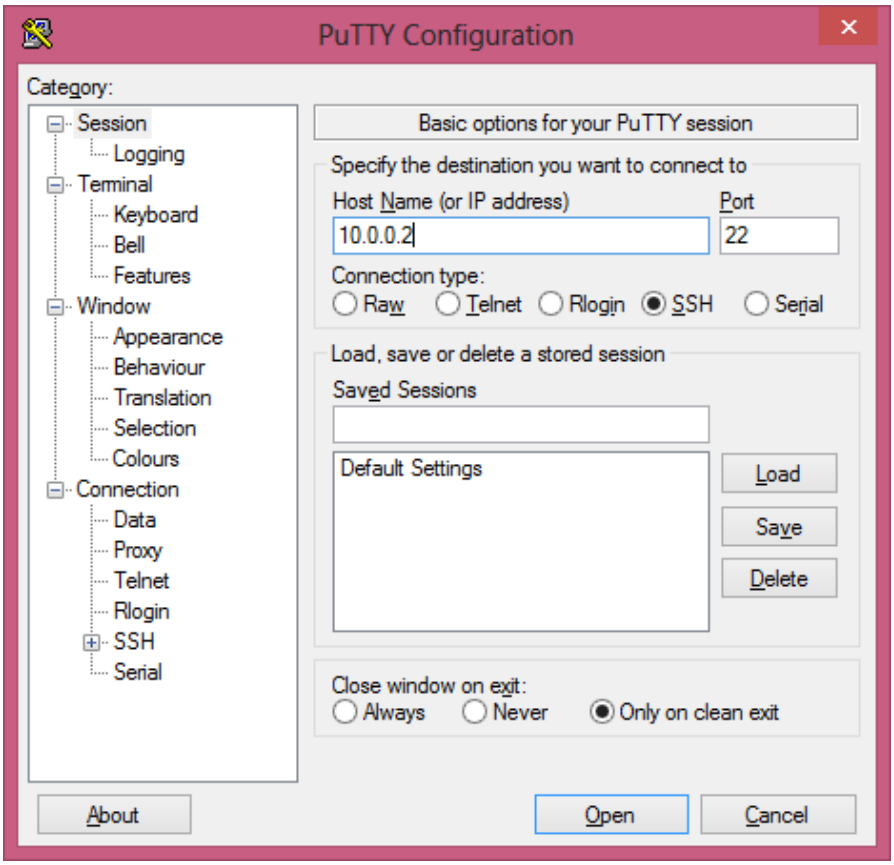
W podanej powyżej konfiguracji mikrokomputer korzysta z adresów IP przyznanych przez serwer DHCP zawarty w punkcie dostępowym do internetu (ang. *router*). W niektórych sytuacjach korzystniej jest używać stałych adresów np. w dostępie kablowym. Należy wówczas zamiast wpisu
iface eth0 inet dhcp
wprowadzić polecenie *static*, a następnie podać adres, maskę sieci i adres bramki internetowej:
iface eth0 inet static
address 192.168.1.2

netmask 255.255.255.0
gateway 192.168.1.1

Oczywiście podane tutaj adresy należy traktować jako przykładowe i w konfiguracji wprowadzić adresy odpowiadające rzeczywiście pasującym do własnej sieci. Maska sieci może w każdym przypadku być zgodna z podaną w przykładzie, natomiast adres mikrokomputera musi być jednym z nieużywanych dotąd we własnej sieci i przykładowo leżeć w zakresie 192.168.1.1 – 192.168.1.254, albo 10.0.0.1 – 10.0.0.254 itp. zależnie od rzeczywiście używanych adresów. Przed wpisaniem należy oczywiście sprawdzić również adres bramki internetowej.
Nazwę, pod którą „Raspberry" występuje w sieci można zmienić w pliku */etc/hostname*.

Przy okazji omawiania dostępu do Internetu nie sposób pominąc jeszcze jednej ważnej związanej z tym sprawy. Niektóre z przedstawionych dalej programów korzystają z niestandardowych kanałów logicznych TCP/IP (ang. *socket* lub *port*), które bardzo często są zablokowane w modemie dostępowym do internetu (ang. *router*). Przykładowo Echolink korzysta z kanałów TCP 5200 i UDP – 5198 i 5199 o ile nie omija ich za pomocą serwera *Proxy*, programy do łączności z siecią D-STAR – z kilku kanałów o numerach powyżej 20000 (dokładne numery zależą od konkretnego programu) itd. Ich prawidłowa praca może wymagać udostępnienia potrzebnych kanałów w modemie internetowym lub w innych zabezpieczeniach przeciwłamaniowych (ang. *firewall*) na łączu internetowym.
Ze względu na dużą liczbę modeli urządzeń odsyłamy czytelników do ich instrukcji.

Dostęp w sieci lokalnej



Rys. 1.16. Okno konfiguracji i poleceń telnetowego programu terminalowego PuTTY. Przykładowym adresem „Maliny" jest 10.0.0.2 a standardowym kanałem logicznym – 22.

W początkowej fazie uruchamiania „Maliny" konieczne jest podłączenie jej do monitora oraz połączenie z klawiaturą i ewentualnie myszą (w przypadku korzystania z graficznej powierzchni obsługi). Po należytym skonfigurowaniu systemu, dostępu do internetu i programów użytkowych korzystanie z mo-

nitora, klawiatury i myszy nie jest już przeważnie konieczne. Wiele czynności serwisowych można wykonać korzystając z dostępu przez sieć lokalną. Wymaga to uruchomienia w konfiguracji „Raspberry" serwera SSH i oczywiście znajomości adresu IP mikrokomputera.
Na PC korzysta się z dowolnego programu klienta Telnetu, np. z popularnego „PuTTY" (rys. 1.6). W polu „**Host name (or IP address)**" należy podać adres „Raspberry" w sieci lokalnej lub nazwę, pod którą w niej występuje i wybrać połączenie SSH a następnie połączyć się z nim naciskając przycisk „**Open**". Kanał logiczny (w polu „**Port**") pozostaje bez zmiany. Po uzyskaniu połączenia na ekranie komputera wyświetlane jest okno wiersza poleceń, np. podobne do pokazanego na ilustracji 1.14.
Po uzyskaniu połączenia na monitorze PC otwierane jest okno wiersza poleceń „Raspberry"(rys. 1.14), w którym w zwykły sposób można wywoływać polecenia systemu, skrypty, programy lub modyfikować pliki za pomocą edytorów „vi" czy „nano" (rys. 1.15) czyli praktycznie w pełni obsługiwać mikrokomputer bez podłączania do niego klawiatury i monitora.
Używany przez „Raspberry" adres IP (o ile nie jest on stały, a przyznawany przez DHCP) można odczytać z powierzchni obsługi punktu dostępowego (ang. *router*) do Internetu.

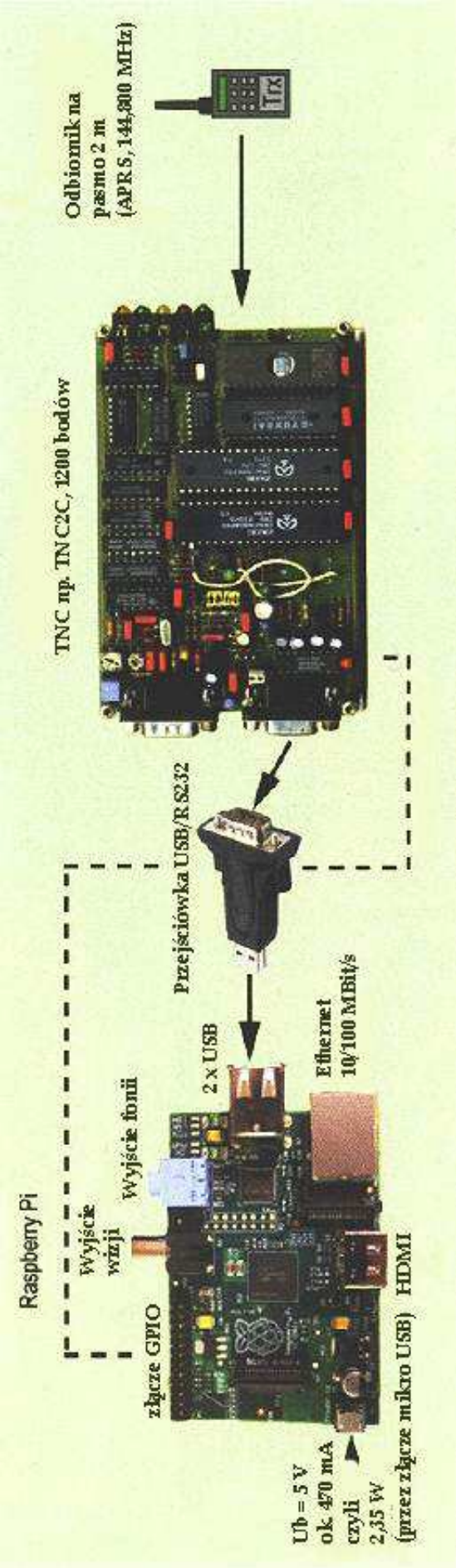
Przy użyciu VNC (Virtual Network Computing) możliwe jest zdalne korzystanie z graficznej powierzchni obsługi „Maliny". Wymaga to zainstalowania serwera VNC na „Malinie" za pomocą polecenia
sudo apt-get install tightvncserver
i klienta na PC. Do popularnych klientów VNC dla „Windows" i innych systemów należy „RealVNC" [30]. System Mac OS X jest wyposażony standardowo w klienta VNC i nie wymaga do tego celu instalowania dodatkowego programu. Dostęp przez VNC powinien być także zabezpieczony za pomocą hasła.
Do wywołania serwera na „Malinie" służy polecenie
vncserver :1

Najwygodniej jest oczywiście jeżeli serwer będzie wywoływany automatycznie. W tym celu należy przejść do katalogu */home/pi* za pomocą polecenia
cd /home/pi
i następnie do ukrytego katalogu *.config*
cd .config
a w nim założyć katalog *autostart*
mkdir autostart

Po przejściu do tego ostatniego katalogu przez
cd autostart
należy np. przy użyciu edytora nano założyć w nim plik *tightvnc.desktop*
nano tightvnc.desktop
o następującej zawartości
[Desktop Entry]
Type=Application
Name=TightVNC
Exec=vncserver :1
StartupNotify=false

Do uzyskania połączenia klienta z serwerem konieczne jest podanie po adresie IP numeru kanału logicznego :1, np. *10.0.0.2:1*.

Bramka radiowo-internetowa APRS

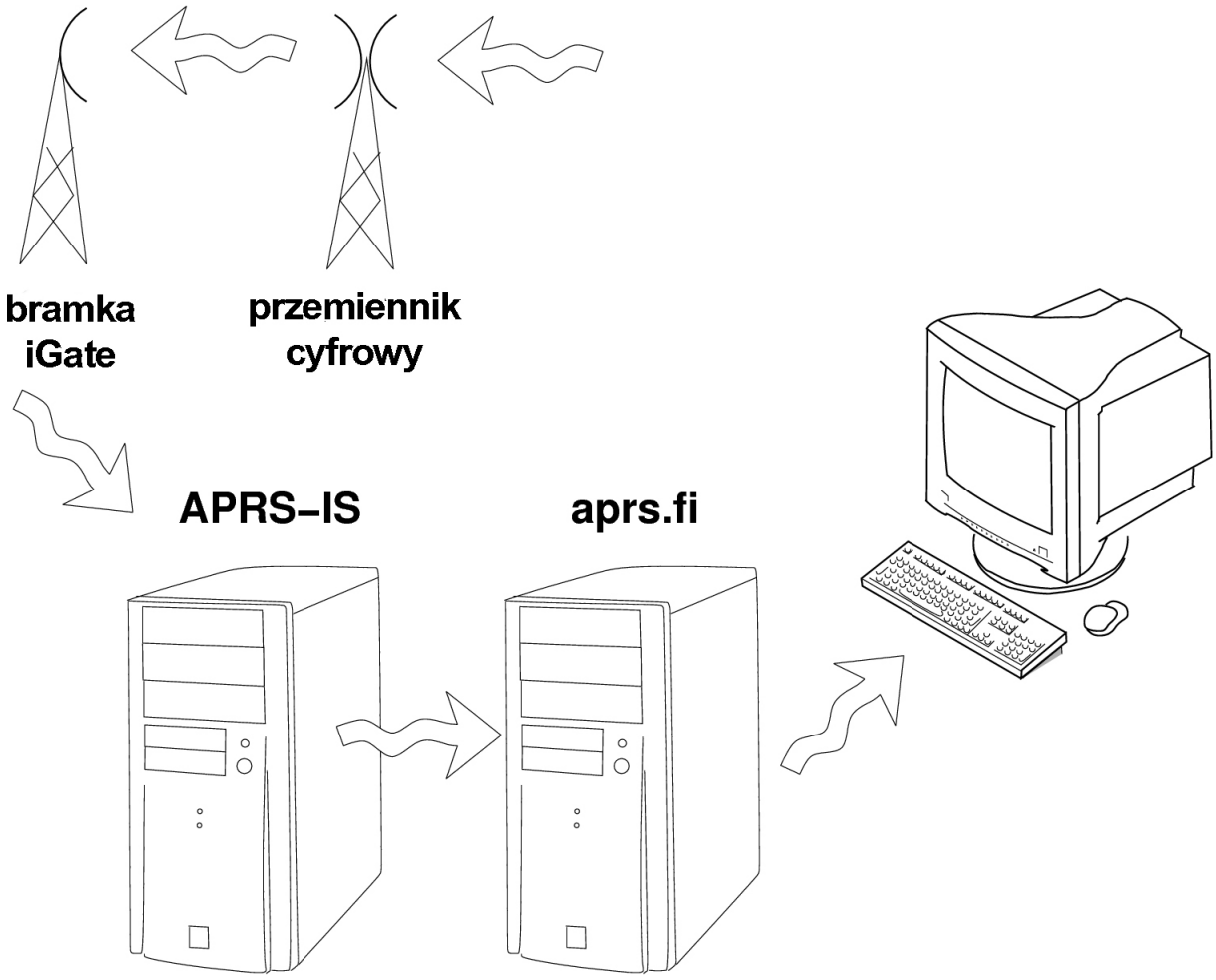


Jednym z częściej publikowanych zastosowań „Raspberry" jest wykorzystanie go jako bramki – z oprogramowaniem „aprsx" albo „aprsd" – lub jako modemu nadawczego APRS. Ogólnie rzecz biorąc możliwe są dwa rodzaje rozwiązań: z użyciem modemu TNC2 pracującego w trybie KISS (rys. 2.1) albo – z użyciem systemu dźwiękowego USB (modemu dźwiekowego). W obu przypadkach konieczna jest dodatkowa radiostacja FM, najczęściej pracująca na standardowej częstotliwości 144,800 albo na lokalnej 432,500 MHz albo odbiornik dla bramek czysto odbiorczych przekazujących dane do internetowych serwerów APRS-IS i dalej do serwisu *arps.fi*.
Zarejestrowanie się na serwerze dostępowym APRS-IS np. *rotate.aprs.net:14580*, *rotate.aprs2.net:14580* albo *euro.aprs2.net:14580* wymaga podania własnego znaku wywoławczego i hasła obliczanego na jego podstawie np. na stronach [15], [16].

Bramka na TNC2

Układ bramki radiowo-internetowej APRS z TNC2 wymaga podłączenia go do złącza USB „Maliny". Ponieważ prawie wszystkie konstrukcje TNC2 są wyposażone jedynie w złącza COM konieczne jest użycie standardowej przejściówki USB/RS232. Połączenie TNC2 z radiostacją jest zrealizowane w zwykły sposób. W przypadku bramki czysto odbiorczej konieczne jest jedynie połączenie wyjścia słuchawkowego radiostacji z wejściem m.cz. modemu TNC.
Program APRX był w początkowych wersjach przeznaczony dla bramek czysto odbiorczych ale w nowszych służy też jako program nadawczy.
Pracuje on w środowisku Unixa i pochodnych stawiając bardzo skromne wymagania systemowe. Jedyną biblioteką niezbędną do jego działania jest „libc". Jest on więc dobrze predystynowany do użytku na komputerach klasy „Maliny".
Może on przekazywać pakiety APRS zarówno z kanału radiowego do APRS-IS jak i odwtrotnie a także obsługiwać radiowy przekaźnik cyfrowy. Teoretycznie może on współpracować z większą liczbą modemów TNC ale jest to oczywiście zależne od mocy przetwarzania i zasobów – zwłaszcza pamięci RAM – komputera.
W przypadku podłączenia kilku odbiorników APRX może przekazywać odebrane przez nie dane do APRS-IS pod znakiem ze wspólnym rozszerzeniem lub z oddzielnymi dla każdego z nich.

Pakiety o adresach docelowych RFONLY, NOGATE, TCPIP, TCPXX nie są transmitowane do Internetu, podobnie jak pakiety o adresach nadawcy: WIDE*, RELAY*. TRACE*, TCPIP, TCPXX, NOCALL, N0CALL.



Rys. 2.2. Praca bramki odbiorczej w sieci APRS-IS

Instalacja aprx wymaga następujących dwóch poleceń (numer wersji ulega oczywiście zmianie w miarę upływu czasu i podany poniżej należy traktować jako przykład):
wget http://ham.zmailer.org/oh2mqk/aprx/aprx_2.04.455-1_armhf.deb
sudo dpkg -i aprx_2.04.455-1_armhf.deb
a po jej zakończeniu konieczne jest dopasowanie konfiguracji:
sudo nano /etc/aprx.conf

W konfiguracji bramki czysto odbiorczej („iGate") należy w pliku */etc/aprx.conf* podać jedynie własny znak i parametry komunikacji z TNC przez złącze szeregowe (systemową nazwę złącza i szybkość transmisji zgodną z ustawioną w TNC):

```
mycall  OE1KDA-1
 <aprsis>
   server   rotate.aprs.net  14580
 </aprsis>
 <interface>
   serial-device /dev/ttyUSB0  19200  8n1    KISS
   tx-ok   false        # parametr domyślny, tylko odbiór. Dla stacji nadawczej „true"
   timeout 900        # 900 sekund oczekiwania na odbiór. Można przyjąć wartość domyślną
 </interface>
```

W internetowej bramce DPRS / APRS w linii *serial-device* po nazwie i szybkości zamiast typu KISS podawany jest typ DPRS. Do złącza szeregowego podłączona musi być wówczas radiostacja D-STAR. Tej samej konfiguracji można użyć w cyfrowym przemienniku DPRS / APRS.

Dla przemiennika cyfrowego z TNC w trybie KISS konfiguracja wygląda jak następuje:

```
mycall  OE1KDA-1
myloc lat ssmm.mmN   lon ssmm.mmE # współrzędne w formacie APRS
 <interface>
   serial-device /dev/ttyUSB0  19200  8n1    KISS
   tx-ok   true
 </interface>
 <digipeater>
   transmitter  $mycall
   <source>
     source     $mycall
   </source>
 </digipeater>
```

Dla połączenia funkcji bramki z funkcją przemiennika należy do powyższej konfiguracji przemiennika dodać sekcję *<aprsis> ... </aprsis>* z pierwszego przykładu.
W sekcji tej zamiast podanego w przykładzie adresu *rotate.aprs.net* można skorzystać z *rotate.aprs2.net* lub innego. Szczegóły w Internecie w witrynie APRS-IS ([25], [26]).
Jeżeli TNC2 nie jest stale przełączony na tryb KISS można w sekcji *<interface>...</intyerface>* dodać polecenie
initstring „\x0dKISS ON\x0dRESET\x0d”
dla TNC z oprogramowaniem TAPR lub
initstring „\0xd033@K\0xd”
dla dla TNC z niemieckim oprogramowaniem TF
albo podobny podany w instrukcji TNC.
Dla skrócenia czasu wykrywania przerw w łączności z serwerem można w sekcji *<aprsis> ... </aprsis>* dodać polecenie
heartbeat-timeout 1m

W bardziej zaawansowanej konfiguracji internetowej bramki nadawczo-odbiorczej „TX-iGate” dodano filtry dopuszczające lub eliminujące pakiety pochodzące od podanych adresów. Przykładowe znaki (adresy*) prefiks_plus, prefiks_minus* należy oczywiście zastąpić przez rzeczywiste w oddzielnych liniach dla każdego adresu. Konfiguracja bez podania filtrów oznacza retransmitowanie wszystkich pakietów bez ich sprawdzania. W bramce czysto odbiorczej definicje filtrów w konfiguracji nie są zasadniczo potrzebne i mogą być przydatne tylko do prób albo w sytuacjach szczególnych.

```
<aprsis>
   server  rotate.aprs.net   14580
   #filter      b/prefiks_plus        # przekazywanie zawsze pakietów od tej stacji
 </aprsis>
 <digipeater>
   transmitter  $mycall               # kanał radiowy
   <source>
     source    $mycall                # kanał radiowy
   </source>
   <source>
     source       APRSIS
     relay-mode   3rd-party
     viscous-delay 5
     #filter      b/prefiks_plus      # przekazywanie zawsze pakietów pochodzących od tej stacji
     #filter      -b/prefiks_minus    # ignorowanie pakietów
     via-path    WIDE1-1
```

```
   </source>
 </digipeater>
```

Polecenie *viscous delay* definiuje czas, w którym te same pakiety nie są retransmitowane dzięki czemu unika się zbędnych powtórzeń blokujących kanał radiowy. Czas ignorowania pakietów leży najczęściej w przedziale od 1 do 9 sekund.
Do uruchomienia i zatrzymania programu służą odpowiednio polecenia:
/etc/init.d/aprx start i
/etc/init.d/aprx stop

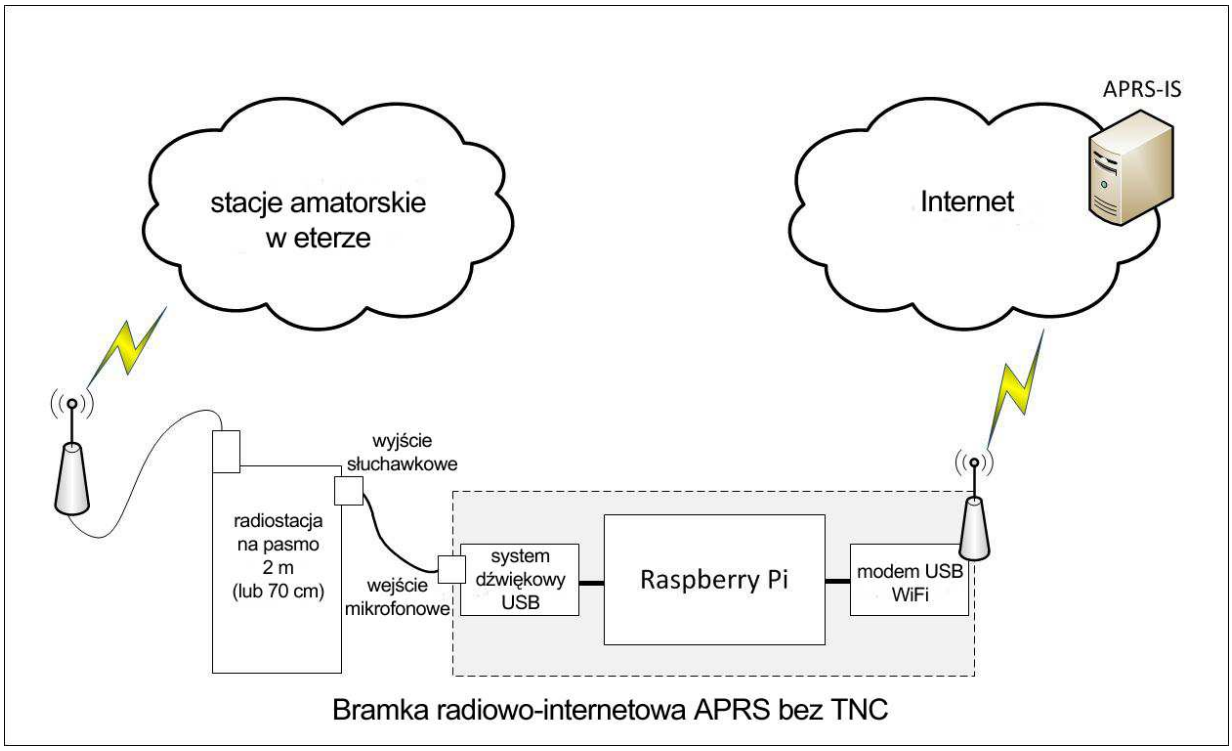
Dodatkowy argument wywołania *–d* powoduje wydawanie do kanału STDOUT informacji diagnostycznych a *–f /etc/aprx.conf* odczyt danych z podanego pliku konfiguracyjnego. W przykładzie podano domyślną nazwę i ścieżkę dostępu do pliku – co oczywiście jest zbędne, ale można w ten sposób korzystać z alternatywnych konfiguracji dostosowanych do różnych celów.
Informacje diagnostyczne są oczywiście przydatne tylko w fazie uruchamiania bramki a nie w czasie jej zwykłej pracy. Argument *–v* powoduje kierowanie kopii odebranych pakietów do kanału STDOUT i jest przydatny również do celów diagnostycznych.
Dalsze przykłady konfiguracji podane są w dostępnej w Internecie [25] instrukcji *aprx-manual.pdf*.

Bramka z dodatkowym podsystemem dźwiękowym

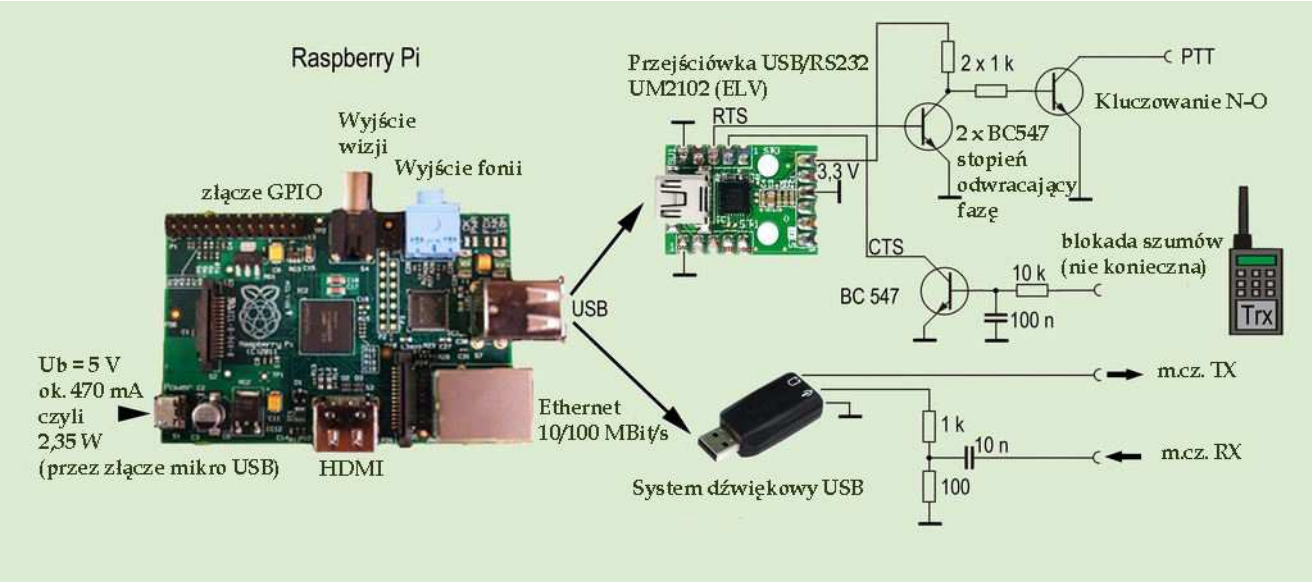
W tym rozwiązaniu zamiast modemu TNC podłączonego do złącza USB zastosowano standardowy zewnętrzny system dźwiękowy USB i programowe dekodowanie pakietów AX.25 w programie *aprx* lub w *multimon.*

Rys. 2.3. Konstrukcja bramki z radiostacją ręczną

W bramce odbiorczej wyjście słuchawkowe lub głośnikowe radiostacji jest połączone z wejściem mikrofonowym tego dodatkowego systemu dźwiękowego. Jego użycie jest konieczne ponieważ „Malina” nie posiada własnego wejścia mikrofonowego a jedynie wyjście sygnału m.cz.
Zastąpienie jej przez „Banana Pi” pozwoliłoby uniknąć tej niedogodności ponieważ „Banan” jest już wyposażony w wejście mikrofonowe (patrz tabela 1.2).



Rys. 2.4. Schemat blokowy bramki



Rys. 2.5. Połączenie „Raspberry” z radiostacją

Schemat na ilustracji 2.5 ilustruje dokładniej sposób połączenia „Maliny” z radiostacją i to nie tylko z odbiornikiem ale i z nadajnikiem. Pozwala to na uruchomienie nadawczo-odbiorczej bramki APRS ale schemat połączeń jest przydatny także do innych rozwiązań j.np. radiowo-internetowe bramki „Echolinku”. Zamiast pokazanej na ilustracji przejściówki USB/RS232 firmy ELV można oczywiście użyć standardowej przejściówki dowolnego producenta. Niektóre z programów korzystają także ze złącza GPIO, do którego można wtedy wtedy podłączyć tranzystor kluczujacy nadajnik.
Ponieważ sygnał stanu blokady szumów (jej otwarcia) można w prosty sposób pobrać tylko z niektórych modeli radiostacji a poza tym nie wszystkie programy odczytują jej stan wejścia to jest często niepotrzebne. W przypadku bramki czysto odbiorczej układ znacznie się upraszcza.

Instalacja i konfiguracja *aprx* przebiega identycznie jak w pierwszym przypadku ale przedtem należy skonfigurować system tak, aby domyślnie korzystał z zewnętrznego podsystemu dźwiękowego zamiast ze standardowego, nie posiadającego niestety wejścia m.cz. Jest on zasadniczo zbędny i najlepiej wyłączyć go w celu zmniejszenia obciążenia systemu.
Polecenie
nano /etc/modprobe.d/alsa-base.conf
otwiera w edytorze plik konfiguracyjny, w którym konieczna jest zmiana indeksów obu systemów dźwiękowych:
options snd-usb-audio index=0 (zamiast -2 wpisać 0)
następnie można wyłączyć standardowy system pracujący na zasadzie modulacji szerokości impulsów:
sudo apt-get remove pulseaudio
zainstalować sterownik modemu dźwiękowego:
sudo apt-get install soundmodem
i obsługę protokołu AX.25:
sudo apt-get install ax25-tools ax25-apps

Konfiguracja obsługi AX.25 i modemu dźwiękowego wymaga założenia i dostosowania do rzeczywistej sytuacji plików:
— /etc/ax25/axports

```
# /etc/ax25/axports
#
#name  callsign      speed  paclen  window  description
APRS   OE1KDA-10    1200   255     2       APRS-iGate
```

— /etc/ax25/soundmodem.conf

Konfiguracji modemu dźwiękowego można dokonać za pomocą programu okienkowego *soundmodemconfig*:

```
<?xml version="1.0"?>
<modem>
  <configuration name="APRS">
    <chaccess txdelay="150" slottime="100" ppersist="40" fulldup="0" txtail="10"/>
    <audio type="alsa" device="plughw:0,0" halfdup="1" capturechannelmode="Mono"/>
    <ptt file="none" gpio="0" hamlib_model="" hamlib_params=""/>
    <channel name="Channel 0">
      <mod mode="afsk" bps="1200" f0="1200" f1="2200" diffenc="1"/>
      <demod mode="afsk" bps="1200" f0="1200" f1="2200" diffdec="1"/>
      <pkt mode="MKISS" ifname="sm0" hwaddr="OE1KDA-10" ip="10.6.0.1"
netmask="255.255.255.0" broadcast="10.6.0.255"/>
    </channel>
  </configuration>
</modem>
```

Po skonfigurowaniu modemu pozostaje jeszcze wyregulowanie siły głosu, tak aby uniknąć przesterowania. Wygodnym narzędziem do tego celu jest program *alsamixer* (rys. 2.6).

Rys. 2.6. Mikser ALSA na „Raspberry”

Pomocna w tym może być także funkcja oscyloskopu programu *soundmodemconfig*. Praktycznym ustawieniem jest wysterowanie ok. 70% i wyłączenie przedwzmacniacza podsystemu dźwiękowego. Ale jest to zależne od konkretnego wyposażenia.
Dobrane ustawienia miksera należy zapisać w pamięci do stałego użytku za pomocą polecenia:
sudo alsactl store

Sposób instalacji i konfiguracji programu aprx jest identyczny z opisanym w poprzednim punkcie. Automatyczne wywołanie sterownika modemu dźwiękowego w trakcie uruchamiania systemu zapewnia wpis w pliku */etc/rc.local*:
/usr/sbin/soundmodem /etc/ax25/soundmodem.conf -R -M >/dev/null 2>/dev/null &
sleep 1

A w pliku */etc/default/aprx* należy skorygować wpis
 # STARTAPRX: start aprx on boot. Should be set to "yes" once you have
 # configured aprx.
STARTAPRX="no" (zmienić na "yes")

Bramka z odbiornikiem DVB-T

W niektórych rozwiązaniach bramek odbiorczych („iGate”) zamiast odbiorników albo radiostacji amatorskich stosowane są miniaturowe komputerowe odbiorniki telewizyjne DVB-T zawierające procesor RTL2832. Są to rozwiązania stosunkowo niedrogie i nie blokujące radiostacji, z której wykorzystywany byłby tylko odbiornik.
Odbiornik jest standardowo podłączany do złącza USB komputerów PC i po zainstalowaniu oprogramowania służy do odbioru telewizji. Niektóre typy odbiorników, właśnie wyposażone w procesor RTL2832 można, korzystając ze sterowników opracowanych przez krótkofalowców wykorzystać jako odbiorniki programowalne (ang. *SDR*) podłączone do komputerów PC lub innych – w tym również do „Raspberry”. Odbiornik taki może pracować w odbiorczej bramce APRS albo zostać po prostu udostępniony w sieci lokalnej i dzięki temu umieszczony w miejscu zapewniającym najlepsze warunki odbioru.
W zależności od wbudowanego scalonego obwodu odbiorczego układ pokrywa zakres od ok. 25 MHz do ok. 1,9 GHz (dla popularnego R820T, dla E4000 zakres zaczyna się od 64 MHz i ma przerwę w paśmie 23 cm) lub zbliżony. W każdym jednak przypadku zakres odbioru obejmuje kilka pasm amatorskich, a wśród nich istotne dla APRS pasma 2 m i 70 cm. Na zdjęciu 2.7 pokazano odbiornik DELOCK 61959 wyposażony w wymagany procesor i przystosowany nie tylko do odbioru DVB-T ale również i DVB-C. W chwilach wolnych od odbioru APRS czy innych zastosowań krótkofalarskich można go więc wykorzystywać i do odbioru telewizji.

Współpraca mikrokomputera z odbiornikiem wymaga zainstalowania jego sterownika:
cd ~/src
sudo apt-get install git build-essential cmake libusb-1.0-0-dev
git clone git://git.osmocom.org/rtl-sdr.git
cd rtl-sdr
mkdir build
cd build
cmake .. -DINSTALL_UDEV_RULES=ON
make
sudo make install
sudo ldconfig

Następnie należy podłączyć odbiornik, sprawdzić poprawność jego działania i ewentualnie ustawić wzmocnienie po wywołaniu programu:
rtl_test

Zainstalowany w ten sposób odbiornik DVB-T można użyć nie tylko do odbioru komunikatów APRS ale np. jako zdalnie dostępny odbiornik panoramiczny w sieci lokalnej.

Do dekodowania pakietów APRS (AX.25) służy „multimonNG”:
cd ~/src
sudo apt-get install qt4-qmake libpulse-dev libx11-dev patch pulseaudio
git clone https://github.com/EliasOenal/multimonNG.git
cd multimonNG
mkdir build
cd build
qmake ../multimon-ng.pro
make
sudo make install

Kolejnym krokiem jest instalacja oprogramowania bramki „iGate“:
cd ~/src
sudo apt-get install python-pkg-resources
git clone https://github.com/asdil12/pymultimonaprs.git
cd pymultimonaprs
sudo ln -s /usr/bin/python2.7 /usr/bin/python2
sudo python2 setup.py install

Dostęp do sieci APRS-IS wymaga użycia hasła, generowanego na podstawie znaku wywoławczego stacji. Dla jego otrzymania można skorzystać z podanych już adresów internetowych albo z załączonego z oprogramowaniem bramki skryptu napisanego w języku „Python”:
cd ~/src/pymultimonaprs
./keygen.py ZNAK
Key for ZNAK: .....

Hasło to wraz z własnym znakiem (bramki odbiorcze nie wymagają przecież żadnej dodatkowej licencji nawet jeśli pracują stale i bez nadzoru operatora, bo nie posiadają nadajników), współrzędnymi geograficznymi, nazwą serwera APRS-IS itp. należy następnie wpisać do pliku konfiguracyjnego bramki:
sudo nano /etc/pymultimonaprs.json

Pozostałe zawarte w nim parametry mogą na początek zachować wartości domyślne.

Dla sprawdzenia poprawności pracy bramki służy następujące wywołanie:
rtl_fm -f 144800000 -s 22050 -p 18 -g 42.0 - | multimon-ng -a AFSK1200 -A

Do zakończenia pracy programu służy kombinacja CTRL-C.
Automatyczne wywołanie programu po uruchomieniu „Maliny” zapewnia następujący skrypt */etc/init.d/pymultimonaprs*:

```
#!/bin/sh
### BEGIN INIT INFO
# Provides: pymultimonaprs
# Required-Start: $all
# Required-Stop: $syslog
# Default-Start: 2 3 4 5
# Default-Stop: 0 1 6
# Short-Description: start/stop of pymultimonaprs
### END INIT INFO
case "$1" in
 start)
   sudo pymultimonaprs --syslog &
   ;;
 stop)
   sudo killall pymultimonaprs
   ;;
 *)
   echo "Usage: /etc/init.d/pymultimonaprs {start|stop}"
   exit 1
   ;;
esac
exit 0
```

Skrypt ten musi oczywiście otrzymać uprawnienia do wykonywania go:
sudo chmod +x /etc/init.d/pymultimonaprs

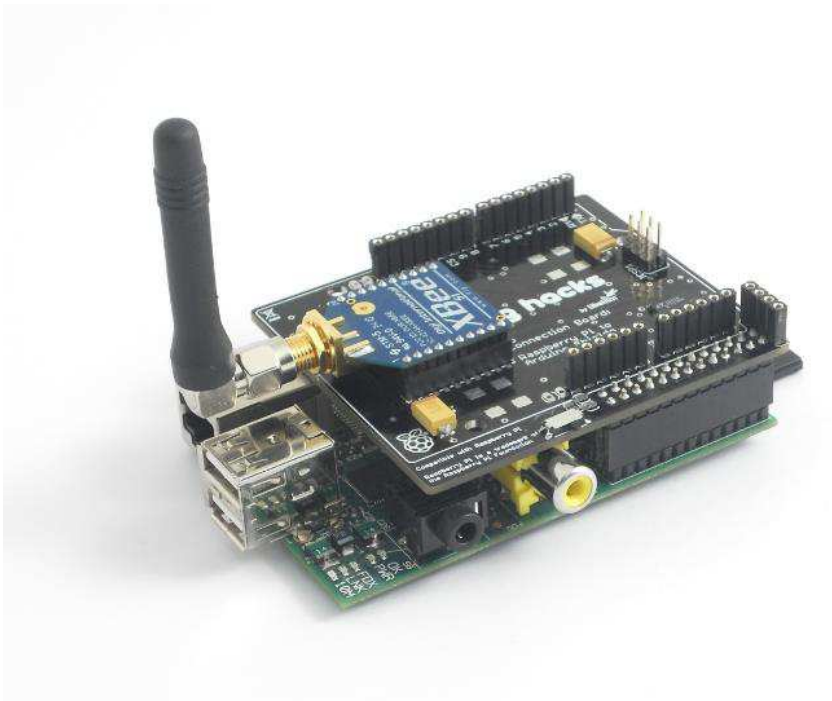
Automatyczne wywołanie skryptu wymaga jeszcze dwóch modyfikacji skryptów wykonywanych w trakcie uruchomiania systemu:
sudo update-rc.d pymultimonaprs defaults
sudo /etc/init.d/pymultimonaprs start

TNC-2 dla „Maliny”

Rys. 2.8. Miniaturowy modem TNC-2 dla „Maliny“

TNC-Pi [33] jest miniaturowym modemem TNC-2 wzorowanym na modelu TNC-X (*http://tnc-x.com*). Wymiary płytki odpowiadają wymiarom „Maliny“ i jest ona połączona z nią za pośrednictwem złącza GPIO a konkretnie rzecz biorąc występującego na nim złącza szeregowego. Konstrukcja ta jest przewidziana do wykorzystania w przemiennikach APRS i bramkach internetowych APRS opartych na „Raspberry“. Do połączenia z radiostacją służy widoczne z boku 9-kontaktowe gniazdo sub-D.
W konstrukcji, opartej na TNC-X, zastosowano mikroprocesor 16F1847, modem MX-614 a w torze m.cz. wzmacniacz operacyjny MCP6023. Parametry konfiguracyjne są zapisywane w szeregowej pamięci EEPRO typu 23K640. Widoczny u góry wtyk GPIO pozwala na piętrowe łączenie kilku TNC – dla bardziej rozbudowanych bramek.
TNC-Pi może w trybie KISS współpracować z programem aprsx, co pozwala na wykorzystanie go w jednym z powyżej opisanych rozwiązań bramki radiowo-internetowej APRS.

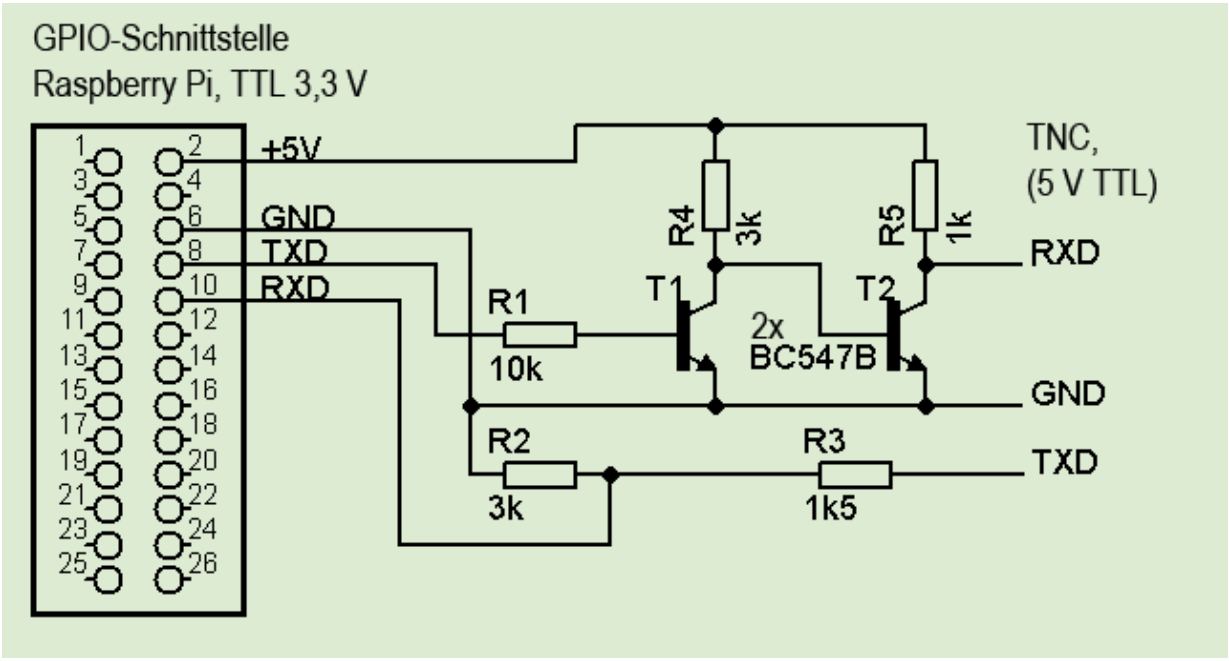
Moduł Xbee dla „Maliny“



W rozwiązaniach różnego rodzaju bramek internetowych, lokalnych łączy dla danych (np. telemetrycznych), lokalnych sieci obsługujących kamery internetowe albo różnego rodzaju czujniki przydany może być moduł radiowy Xbee. Do połączenia z mikrokomputerem korzysta on ze złącza szeregowego (UART-u) w gnieździe GPIO.

Rys. 2.9. Radiowy moduł Xbee dla „Maliny“

Układ dopasowania poziomów 3,3 V do TTL na złączu szeregowym

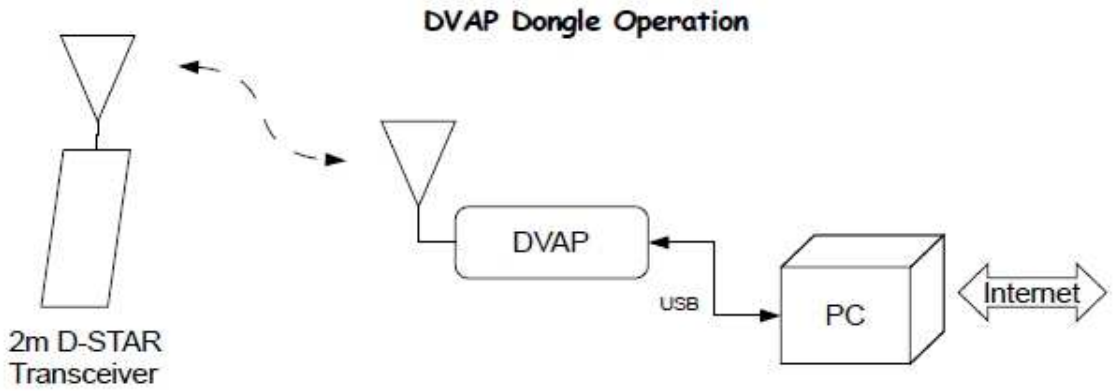


Rys. 2.10. Układ dopasowania poziomów napięć na złączu szeregowym

Jak już wspomniano w rozdziale 1 złącze GPIO „Maliny“ jest dostosowane do poziomów napięć 3,3 V. Podanie na jego wejścia napięć wyższych np. 5 V a także zwarcie do masy może spowodować nieodwracalne uszkodzenia mikrokomputera. Dla uzyskania standardowych poziomów napięć +/-12 V najwygodniej posłużyć się obwodem MAX3232. Niektóre typy TNC-2 są jednak wyposażone dodatkowo w złącza TTL i w tym przypadku pomocny jest układ przedstawiony na schemacie 2.10.

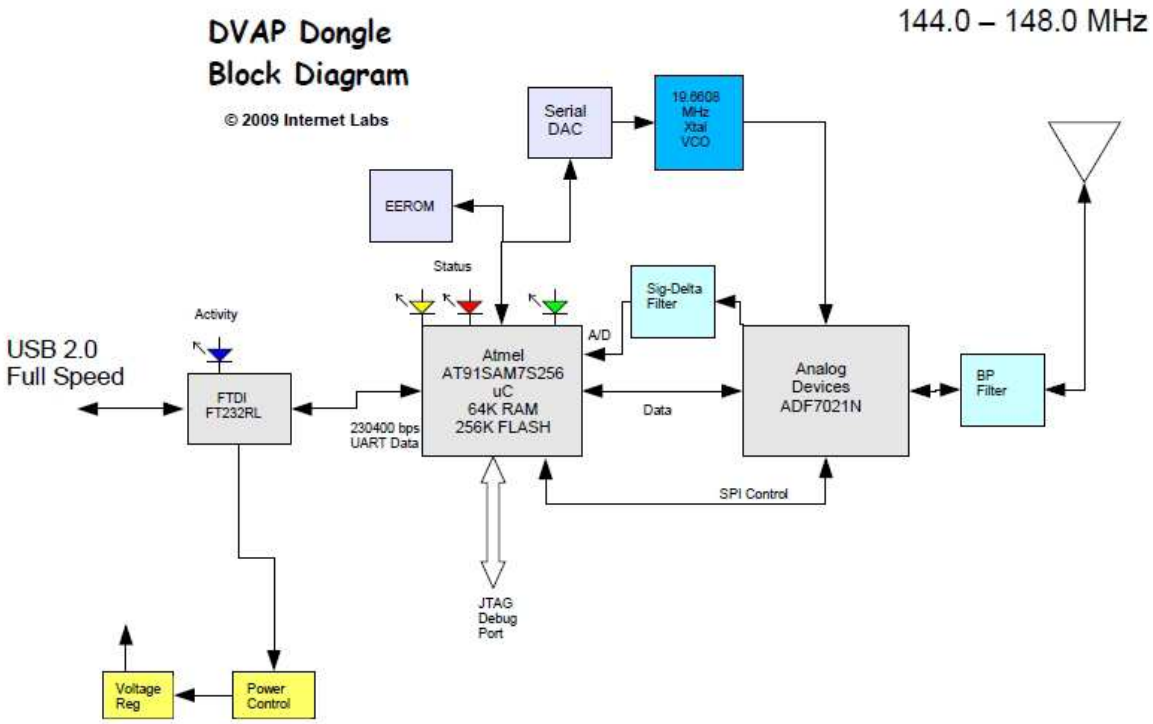
Mikroprzemiennik D-STAR

Miłośników cyfrowego dźwięku zainteresuje z pewnością konstrukcja mikroprzemiennika lub inaczej mówiąc lokalnego radiowego punktu dostępowego do sieci D-STAR.
Przemiennik składa się z miniaturowej radiostacji DVAP o mocy 10 mW pracującej w paśmie 2 m lub 70 cm (do wyboru są dwa modele: DV-ACCESS-2 i DV-ACCESS-70) i komputera sterującego pracą stacji i zapewniającego jej dostęp do przemienników lub reflektorów sieci D-STAR przez Internet (rys. 3.1 i 3.2).
Oprócz oprogramowania dla „Maliny" istnie także wersja „DVAPTool" dla komputerów PC z systemem „Windows" i dla „Mac OS".
Na zdjęciu 3.1 mikroprzemiennik jest zasilany z 5 V akumulatora litowo-jonowego. U góry nad wtyczką kabla miniradiostacji (leżącej po prawej stronie, w czerwonej obudowie) widoczny jest moduł WiFi z migającym na niebiesko sygnalizatorem połączenia z siecią.
Sposób korzystania z mikroprzemiennika jest identyczny ja dla każdej zwykłej stacji przemiennikowej – do łączności służy radiostacja D-STAR z odpowiednio zaprogramowanymi polami adresowymi dostrojona do simpleksowej częstotliwości pracy przemiennika.
Zasięg przemiennika można zwiększyć podłączając go do anteny wewnętrznej o większym zysku lub do anteny zewnętrznej (radiostacja DVAP jest wyposażona w standardowe złącze SMA). W wielu sytuacjach ograniczene zasięgu do najbliższych okolic jej zainstalowania jest jednak pożądane.



Rys. 3.2. Schemat blokowy i sposób korzystania z przemiennika

Transceiwery „DV-ACCESS”

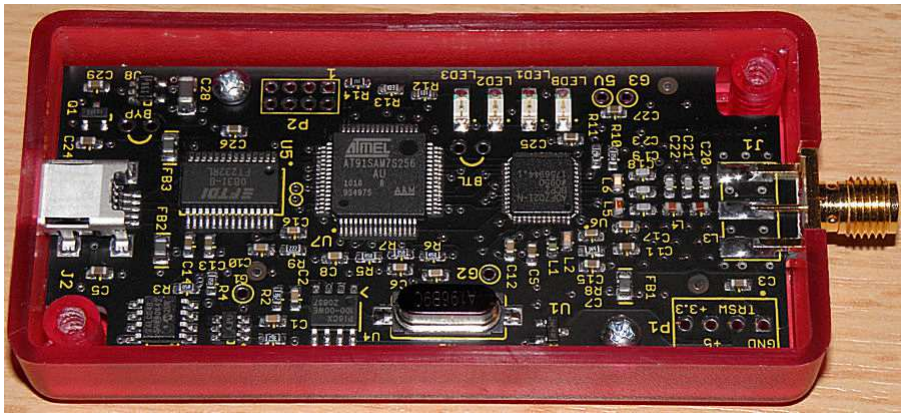


Rys. 3.3. Schemat blokowy mikroradiostacji DVAP

Miniaturowe radiostacje *DVAP Dongle* występujące również pod nazwą *DV-ACCESSS* zawierają modem GMSK 4800 bit/s i pracują simpleksowo w jednym z pasm 2 m albo 70 cm (w wersji europejskiej odpowiednio 144–146 lub 430–440 MHz). Jako transceiwera użyto w nich obwodu ADF7021N zawierającego nadajnik o mocy maksymalnej 10 mW. Możliwe jest też jej zmniejszenie w programie sterującym. Radiostacje połączone są z komputerem za pomocą złącza USB.
Oprócz modulacji GMSK DVAP może pracować analogową emisją FM i posiada dekoder DTMF. Wewnątrz półprzezroczystej obudowy znajdują się diody świecące informujące o jej pracy. Dioda czerwona sygnalizuje nadawanie danych radiowo, zielona ich odbiór od radiostacji użytkownika, a migająca niebieska – transmisję danych z komputera do „DVAP Dongle”. Stan gotowości do użycia po włączeniu i uruchomieniu programu sygnalizuje miganie diody niebieskiej. Po włączeniu, ale przed uzyskaniem połączenia z programem sterującym dioda zielona miga powoli.
Dioda żółta sygnalizuje wystąpienie błędnych pakietów danych, co może się zdarzać od czasu do czasu i nie powinno budzić niepokoju jeśli nie występuje zbyt często. Miganie diogy czerwonej sygnalizuje błąd w urządzeniu. Szybkie miganie diod czerwonej, zielonej i żółtej oznacza wystąpienie w układzie PLL Szybkie miganie diod czerwonej, zielonej i żółtej oznacza wystąpienie w układzie PLL błędu uniemożliwiającego nadawanie.
Schemat blokowy *DVAP Dongle* przedstawia rys. 3.3.
Radiostacja jest wyposażona w standardowe gniazdo antenowe SMA 50 Ω dzięki czemu można ją bez problemów połączyć z anteną dowolnego typu. Po drugiej stronie obudowy znajduje się miniaturowe gniazdo USB mini B, a więc również do połączenia z komputerem używa się łatwo dostępnego kabla USB, służącego również do jej zasilania.
Nie zawiera ona wokodera AMBE a więc do pracy w sieci D-STAR konieczne jest użycie radiostacji D-Starowej a nie analogowej, jak to jest możliwe w niektórych innych rozwiązaniach. Radiostacja D-Starowa pozwala jednak na pełniejsze wykorzystanie możliwości sieci aniżeli w rozwiązaniach opartych o sprzęt analogowy.
Podłączenie radiostacji do anteny zewnętrznej może zwiększyć zasięg nawet do ponad kilometra – zależnie od jej zysku i warunków zewnętrznych. Nie zalecane jest jednak stosowanie dodatkowych wzmacniaczy mocy ponieważ czystość sygnału nadawanego nie jest wystarczająca do tego celu. Dla załączonej anteny producent podaje zasięgi dochodzące do 100 m zależnie od otoczenia.

Generator sterujący nie jest stabilizowany temperaturowo co może spowodować wystąpienie dryfu częstotliwości. Jeżeli jest on zbyt duży należy skorygować częstotliwość pracy w oknie programu (a nie w radiostacji D-Starowej). Zasadniczo stabilność częstotliwości w temperaturze pokojowej jest wystarczająca do bezpieczeństwowej pracy ale wahania temperatury w plenerze lub samochodzie mogą spowodować odchyłki dochodzące do kilku kHz.

Rys. 3.4. Widok zewnętrzny radiostacji *DVAP Dongle* (zdjęcie z witryny producenta)



Rys. 3.5. Konstrukcja wewnętrzna. Po lewej stronie widoczny obwód FTDI FT232RL do komunikacji przez złącze USB, w środkowej części procesor sterujący AT91SAM7S256 (ARM 7) firmy ATMEL, a po prawej obwód nadawczo-odbiorczy ADF7021N. U góry widoczne są cztery kolorowe diody świecące. Zdjęcie z witryny producenta.

Tabela 3.1. Najważniejsze parametry radiostacji *DVAP Dongle*

Odbiornik	
Zakres pracy w modelu dla regionu 1	144,0–146,0 MHz lub 430,0–440,0 MHz
Stabilność częstotliwości	+/- 50 x 10^{-6} w zakresie temperatur -20–+70 °C
Czułość	~ -110 dBm
Selektywność	~ 13,5 kHz, filtr Butterwotha 5 rzędu
Demodulator	GMSK – korelator 2FSK, dyskryminator liniowy dla FM
Nadajnik	
Zakres pracy w modelu dla regionu 1	144,0–146,0 MHz lub 430,0–440,0 MHz
Stabilność częstotliwości	+/- 50 x 10^{-6} w zakresie temperatur -20–+70 °C
Moc wyjściowa	regulowana w zakresie -12 dBm do +10 dBm (10 mW)
Modulacja	4800 bit/s GMSK – 2FSK, dewiacja +/- 1200 Hz +/- 5 kHz dewiacja analogowa FM
Gniazdo antenowe	SMA 50 Ω
Gniazdo USB	mini B
Złącze USB	2.0
Wirtualne złącze szeregowe	230400 bit/s, 8N1

Przemiennik z „DVAPTool”

Oprogramowanie „DVAPTool” jest dostępne w kilku wersjach: dla komputerów PC (systemu „Windows”), MAC (OS X) i „Raspberry”. „DVAPTool” dla „Maliny”, autorstwa AA4RC, jest obecnie dostępny w wersji 1.04 a do jego instalacji na R-Pi służą następujące polecenia:

sudo apt-get install qt4-dev-tools
curl -O http://opendstar.org/tools/DVAPTool-1.04-rpi.tgz
sudo tar xzPf DVAPTool-1.04-rpi.tgz

Do ręcznego wywołania programu stosowane jest następujące polecenie (z katalogu użytkownika):
./DVAPTool

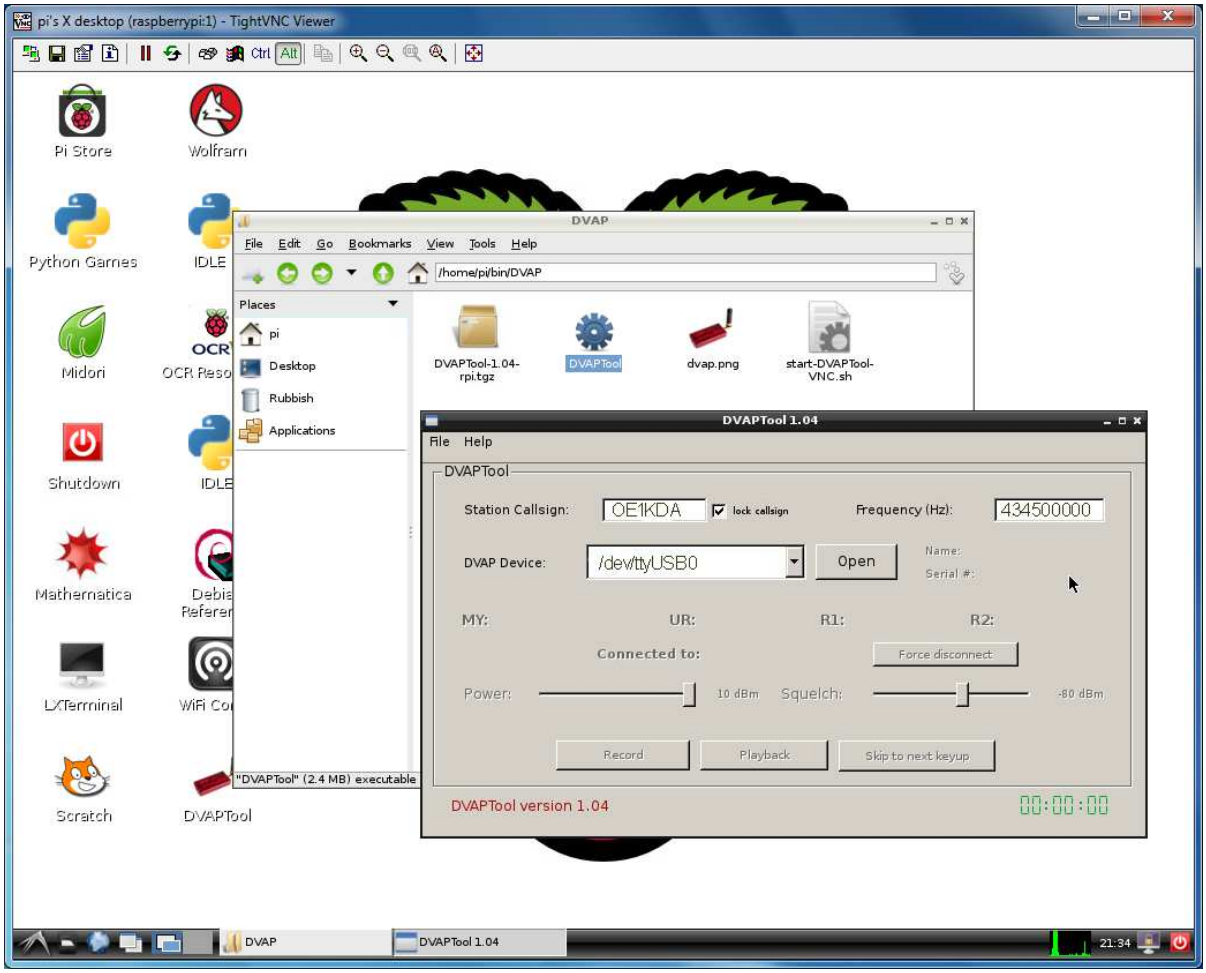
Wygodniej jest jednak zautomatyzować wywołanie dodając do pliku */etc/rc.local* polecenie uruchomienia graficznej powierzchni obsługi:

su pi –c startx

i zakładając w katalogu */home/pi* plik o nazwie *.xinitrc* o następującej zawartości

#!/bin/sh
#.xinitrc
 # Start DVAPTool
exec /home/pi/DVAPTool –open

Po nadaniu plikowi uprawnień do wykonywania go: *chmod 755 .xinitrc*
oprogramowanie punktu dostępowego startuje automatycznie po uruchomieniu „Raspberry” i wywołaniu graficznej powierzchni obsługi.



Rys. 3.6. Okno programu.

Po pierwszym wywołaniu konieczne jest wpisanie danych konfiguracyjnych: w polu „**Station call**” znaku wywoławczego (własnego, zarejestrowanego w sieci D-STAR, bez dodatkowych rozszerzeń jak w innych wersjach) i w polu „**Frequency (Hz)**” częstotliwości pracy z dokładnością do Hz (rys. 3.2). W polu „**DVAP Device**” widoczna jest systemowa nazwa przyznana miniradiostacji („/dev/ttyUSB0” itp.) a poniżej znajdują się suwaki służące do regulacji jego mocy wyjściowej i progu otwarcia blokady szumów. Przycisk „**Open**” służy do nawiązania połączenia z transceiwerem DVAP – podpis zmienia się wówczas na ”**Close**” i służy on wtedy do przerwania połączenia.
Zaznaczenie pola „**Lock callsign**” powoduje, że mikroprzemiennik jest dostępny tylko dla jego operatora a po usunięciu – dla wszystkich. Zasięg przemiennika można powiększyć, w korzystnych warunkach nawet do kilku km, podłączając go do anteny zewnętrznej. W trakcie odbioru sygnału w oknie wyświetlana jest zawartość pól adresowych MYCALL, RPT1, RPT2 i URCALL odbieranej stacji oraz paskowy wskaźnik siły odbioru. W polach przemienników podawane jest słowo „DIRECT”.
Przyciski „**Record**” („Nagranie”) i „**Playback**” („Odtwarzanie”) służą do nagrywania i odtwarzania prowadzonych łączności. Przycisk „**Skip to next keyup**” pozwala na przeskoczenie do następnej nagranej relacji.
Zapowiedzi przemiennika znajdują sięw plikach o rozszerzeniu *.dvrec*:
alreadylinced.dvrec – informująca o uprzednim nawiązaniu połączenia z przemiennikiem lub reflektorem (np. po próbie nawiązania połączenia jeśli mikroprzemiennik jest już połączony),
alreadyunlinked.dvrec – przy próbie rozłączenia jeśli nie jest on z niczym połączony,
dvap-id.dvrec – odpowiedź na polecenie „I”,
gatewayunknown.dvrec – gdy nawiązanie połączenia niemożliwe,
remotesystem.linked – informacja o nawiązaniu połączenia,
remotesystemunlinked.dvrec – informacja o przerwaniu połączenia.
W celu nagrania własnych zapowiedzi należy rozłączyć się z siecią D-STAR i nacisnąć przycisk „**Record**” („Nagranie”), nagrać tekst, a na zakończenie nacisnąć przycisk „**Stop recording**” („Koniec nagrania”). Do jego odsłuchania służy przycisk „**Playback**” („Odtwarzanie”). Zapowiedź jest zapisywana zawsze w pliku *dvaptool.dvrec*, który po uzyskaniu poprawnego wyniku (zgodnego z życzeniem operatora) należy zapisać pod pasującą – jedną z podanych powyżej – nazwą, po czym można przystąpić do nagrywania następnej zapowiedzi.

Praca przemiennika i ustawienia radiostacji

Do pracy przez mikroprzemiennik (lub jak kto woli – punkt dostępowy) konieczna jest dowolna radiostacja D-Starowa – praktycznie, jeśli będzie to radiostacja przenośna. Jej moc wyjściowa powinna być zredukowana do minimum aby uniknąć przesterowania przemiennika.
Użytkownik mikroprzemiennika ma do dyspozycji ogólnie znane polecenia połączenia z wybranym przemiennikiem sieci lub reflektorem np.: SR9UVMBL, REF032CL, rozłączenia za pomocą „U” na ósmej pozycji; do pracy przez połączoną stację sieci lub w zasięgu lokalnym służy jak zwykle adres CQCQCQ w polu URCALL.
Oprócz tego do wywołania głosowej informacji o przemienniku służy polecenie „DVAP I” a do wywołania jego funkcji echa – „DVAP E”. Jak zwykle litery „I” i „E” muszą się znajdować na ósmej pozycji.
Pola przemienników RPT1 i RPT2 nie są używane – należy je pozostawić puste ale jakakolwiek inna zawartość też nie nie powoduje błędów w pracy przemiennika. Wołanie korespondenta po znaku nie jest możliwe.

Tabela 3.2. Przykład programowania radiostacji do pracy przez przemiennik DVAP. W tabeli podano tylko istotne pola pod ich używanymi w programie konfiguracyjnym nazwami. Simpleksowa częstotliwość pracy jest dowolna ale musi być zgodna z podaną w „DVAPTool".

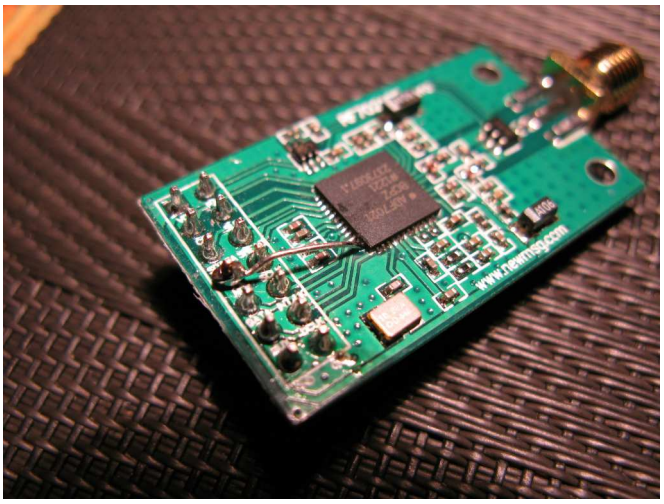
CH	Freq	DUP	Offset freq	Mode	Name	YOUR / URCALL	RPT1	RPT2
1	434,50000		0,00000	DV	DVAP CQ	CQCQCQ		
2	434,50000		0,00000	DV	DVAP I	DVAP I		
3	434,50000		0,00000	DV	DVAP E	DVAP E		
4	434,50000		0,00000	DV	Rozlacz	U		
5	434,50000		0,00000	DV	SR9UVMBL	SR9UVMBL		
6	434,50000		0,00000	DV	SR9UVMCL	SR9UVMCL		
6	434,50000		0,00000	DV	SR9UVZBL	SR9UVZBL		
7	434,50000		0,00000	DV	REF032CL	REF032CL		

Otrzymywany przy próbie połączenia meldunek „**gateway unknown**" („bramka nieznana" = „cel nieznany") może być spowodowany tym, że punkt dostępowy nie zdążył się zameldować w bazie danych użytkowników sieci (US Trust) ponieważ nawiązywanie połączenia przez Raspberry z Internetem trwało zbyt długo. W pliku *.xinitrc* można przed wywołaniem „DVAPTool" dodać wówczas pewne opóźnienie np. *sleep 90* albo coś koło tego.
Zamiast stałego opóźnienia można też wprowadzić opóźnienie stopniowane uzależnione od czasu rzeczywiście potrzebnego na połączenie się z Internetem, jak w poniższym przykładzie:

```
# Avoids 'gateway unknown' errors.
until [ "`ping -c 1 auth.dstargateway.org > /dev/null 2>&1; echo $?`" -eq 0 ]
      do

            sleep 5s

done #until [ "`ping -c 1 auth.dstargateway.org       > /dev/null 2>&1; echo $?`" -eq 0 ]

if [ ! -z $VNCDESKTOP ]
      then

            exec lxterminal -e /home/pi/bin/DVAP/DVAPTool -open

fi #if [ ! -z $VNCDESKTOP ]
```

Wadą programu w obecnej wersji jest fakt, że łączy się on tylko z przemiennikami będącymi w kontakcie z US Trust natomiast nie pozwala na korzystanie z przemienników tylko i wyłącznie połączonych z ircDDB (operator otrzymuje odpowiedź „**gateway unknown**"), co podobno po dodatkowych wysiłkach konfiguracyjnych było jeszcze możliwe w wersji 1.02.
Najważniejszymi zaletami tego rozwiązania są małe wymiary i ciężar, pobór mocy poniżej 3 W oraz łatwość uruchomienia i nieskomplikowany sposób korzystania z przemiennika. Za wady można uznać brak dostatecznej informacji o oprogramowaniu i konstrukcji transceiwera DVAP i jego cenę rzędu 250 funtów brytyjskich. Stosunkowo niewielki zasięg (zwłaszcza z dostarczoną anteną) ogranicza wprawdzie jego zastosowanie do bliskich okolic mieszkania lub innego miejsca pobytu, ale w zależności od konkternej sytuacji można widzieć w tym wadę albo zaletę. W każdym razie użycie przemiennika w rejonach nie pokrytych zasięgiem stałych przemienników sieci i możliwość prowadzenia w ten sposób QSO jest dla wielu osób atrakcyjna.
Na korzystanie i z bramek połączonych tylko „ircDDB" pozwala alternatywne oprogramowanie „ircDDB Gateway" dostępne w internecie pod adresem [17] i szczegółowo przedstawione w następnym rozdziale.

Opisana konstrukcja jest jedną z kilku podobnych opracowanych przez krótkofalowców. Podobną ale opartą o moduł transceiwera RF7021SE (modem GMSK na obwodzie ADF7021) i wyraźnie tańszą jest DVSP2 [18]. Chiński moduł, dostępny na *aliexpress.com*, wymaga dwóch niewielkich przeróbek: doprowadzenia do styku brakującej nogi ADF7021 i usunięcia generatora kwarcowego 16 MHz. Modyfikacje należy przeprowadzić bardzo starannie aby nie uszkodzić płytki drukowanej. Konstrukcję i oprogramowanie opracowali SQ9MDD i SQ9ATC. Czytelników zainteresowanych aktualnym stanem prac i możliwością zakupu gotowych płytek odsyłam do witryny konstruktorów.



Rozwiązania mikroprzemienników można także oprzeć na konstrukcjach DVRPTR albo UP4DAR.

Rys. 3.7. Moduł nadawczo-odbiorczy DVSP2. Zdjęcie z witryny [18].

Przemiennik z „ircDDB Gateway”

Oprogramowanie przemiennika „ircDDB Gateway” jest dostępne w internecie pod adresem [20]. Dla przemiennika opartego na radiostacji „DVAP Dongle” należy pobrać program z pozycji drugiej w tabeli, natomiast dla innych rozwiązań technicznych – odpowiednio z innych pozycji. Pobrany z witryny skomprymowany plik w formacie *.tar* zawiera obraz pamięci przeznaczony do wpisania na moduł SD za pomocą specjalnego programu – nie można go kopiować w zwykły sposób. Dla Windows może być to przykładowo opisany już „Win32 Disk Imager” (rys. 3.9) – dostępny również w tej samej witrynie. Plik zawiera zarówno oprogramowanie przemiennika jak i podaną w tabeli wersję systemu operacyjnego „Wheezy”. Po rozpakowaniu całość zajmuje około 4 GB, a więc moduł pamięci powinien mieć co najmniej 8 GB pojemności, aby można było zainstalować na nim programy potrzebne do innych celów albo w przyszłości aktualizacje systemu i oprogramowania przemiennika.

Downloads

Home | Aknowledgemer | Contact Us | Downloads | How To's | Links

Images and Installation

There is more than one way of getting your Raspberry PI up and running your D-Star radio kit using the Jonathan Naylor packages, here are a few alternatives.

Before you start straight out of the box, there are certain basics we must have in place.

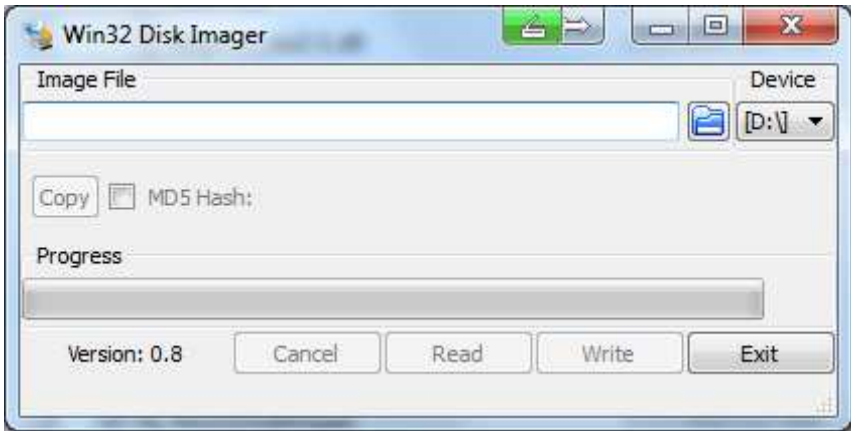
1. A Raspberry PI (only tried on type 'B') and a power source, dedicated PSU or decent quality micro-USB cable.
2. An SD card (4Gb or above).
3. A PC with an internet connection.
4. An SD card reader/writer

If the dates for the **ircDDBGateway** or **Repeater** packages listed here for the complete image is older than the updates shown on the homepage, then download the image that suits your installation and simply run 'KLXupdate' as documented on the desktop.

As the developer (Jonathan Naylor G4KLX) of the software bundled on the images below appears to moving toward a '*one size fits all*' solution to make life easier for us, I will only be maintaining the 'D_StarRepeater' option from the list unless requested otherwise. However, all the images below have the 'klxupdater' installed and therefor could be brought up-to-date in minutes if they are your chosen option.

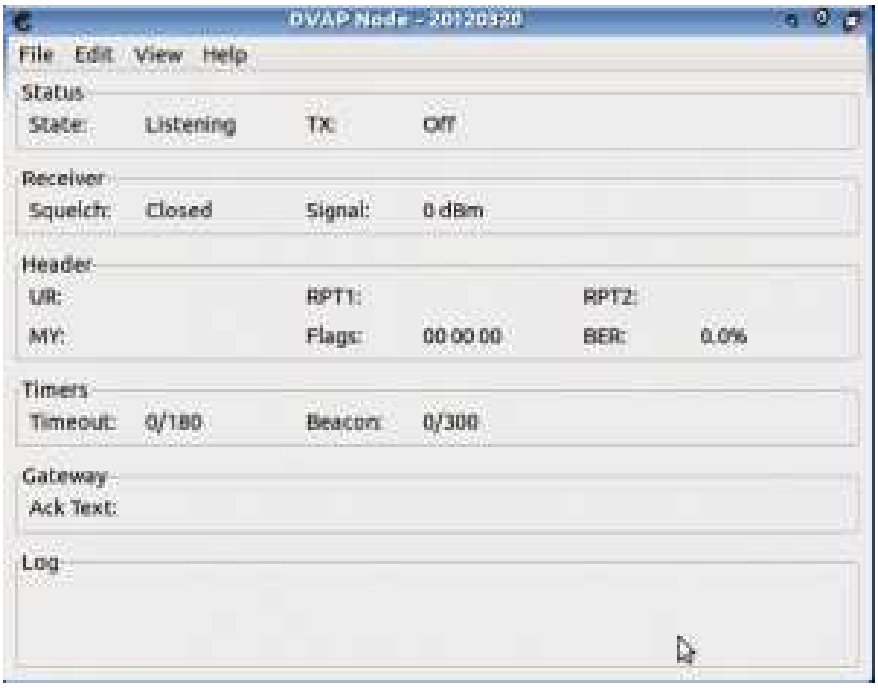
Download File	Description	Base Image	ircDDBGateway	Ds/Repeater
D-StarRepeater+ircDDB+VNC	An 'Autostart' Raspberry PI image for D-StarRepeater **(B+ Compatible)**	Dedbian Wheezy HF 25-09-2013	02-06-2014	21-05-2013
DVAP+ircDDB+VNC	An 'Autostart' Raspberry PI image for DVAP	Dedbian Wheezy HF 25-09-2013	01-01-2014	31-12-2013
GMSK+ircDDB+VNC	An 'Autostart' Raspberry PI image for GMSK modem	Dedbian Wheezy HF 25-09-2013	01-01-2014	31-12-2013

Rys. 3.8. Pobranie programu „DVAP+ircDDB+VNC” z Internetu



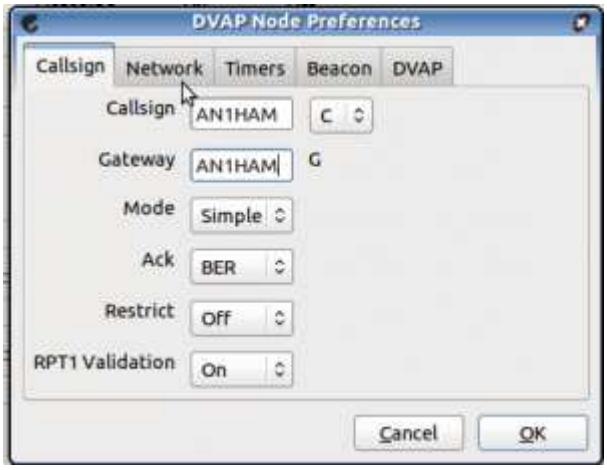
Rys. 3.9. Okno główne programu „Win 32 Disk Imager“ służącego do kopiowania obrazów pamięci

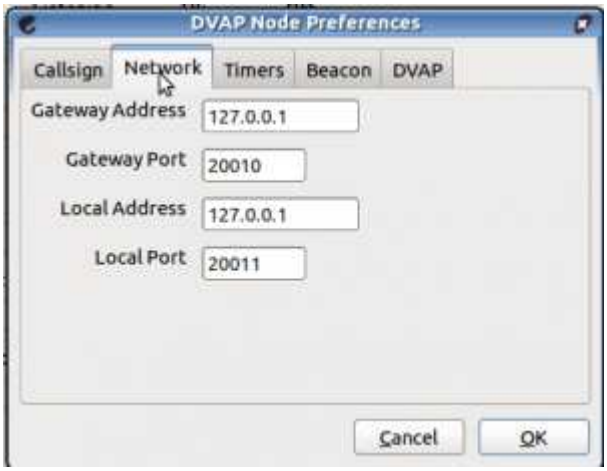
Ustawienia „DVAPNode“



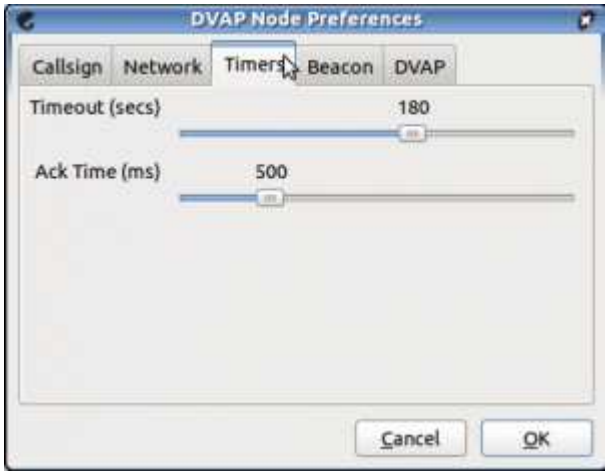
Rys. 3.10. Wywołanie konfiguracji „DVAPNode“ z okna głównego programu. Program wywołuje się z poziomu wiersza poleceń za pomocą polecenia *sudo dvapnode –gui* albo za pomocą symbolu z poziomu graficznej powierzchni obsługi. Do otwarcia okna konfiguracyjnego służy menu „Edit/Preferences“.

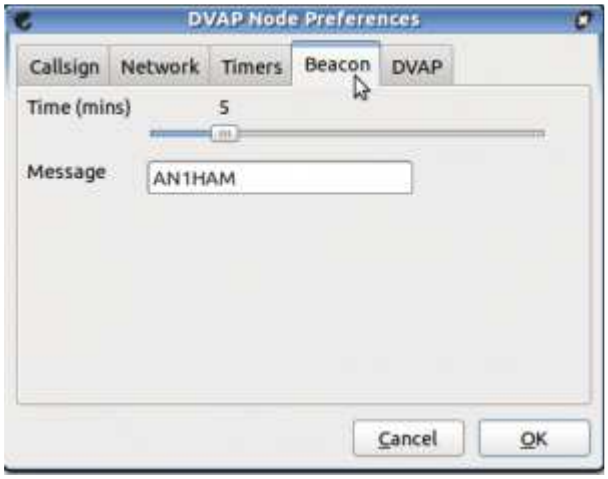
W podanej na rys. 3.11 konfiguracji należy na początek zmienić tylko podane dalej parametry a resztę pozostawić bez zmian. Do najważniejszych z nich należy oczywiście znak wywoławczy wraz z literą oznaczającą pasmo – zgodnie z ogólnie znanymi zasadami czyli C dla pasma 2 m lub B dla pasma 70 cm – i częstotliwość pracy zależna od używanego modelu miniradiostacji. Bramka zawsze otrzymuje literę G. W zależności od potrzeb można obniżyć moc wyjściową nadajnika poniżej 10 mW. Próg blokady szumów i korektę częstotliwości można podać po próbnym okresie pracy.
Po zakończeniu konfiguracji „DVAP Node“ należy wyłączyć program, ponownie uruchomić „Malinę“ i przystąpić do konfiguracji „ircDDB Gateway“. „IrcDDB Gateway“ wywołuje się z poziomu wiersza poleceń za pomocą polecenia *sudo ircddbgateway –gui*. okno konfiguracyjne otwiera się również za pomocą menu „**Edit/Preferences**“.



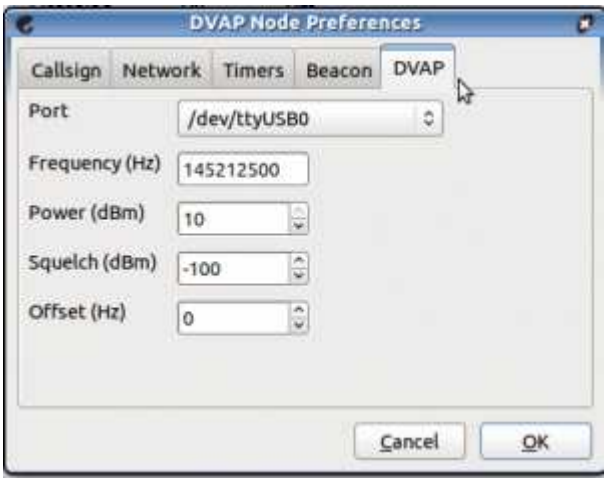


Rys. 3.11ab. Pierwsze dwie zakładki konfiguracji „DVAP Node“. W polach „**Callsign**“ („Znak“) i „**Gateway**“ („Bramka“) należy podać własny znak wywoławczy i do pierwszego z nich wybrać literę odpowiadającą pasmu pracy – zgodnie z ogólnie przyjętymi zasadami. Tryb pracy pozostaje simplek-sowy. Pole „**Restrict**“ decyduje o ograniczeniu dostępu dla innych użytkowników.



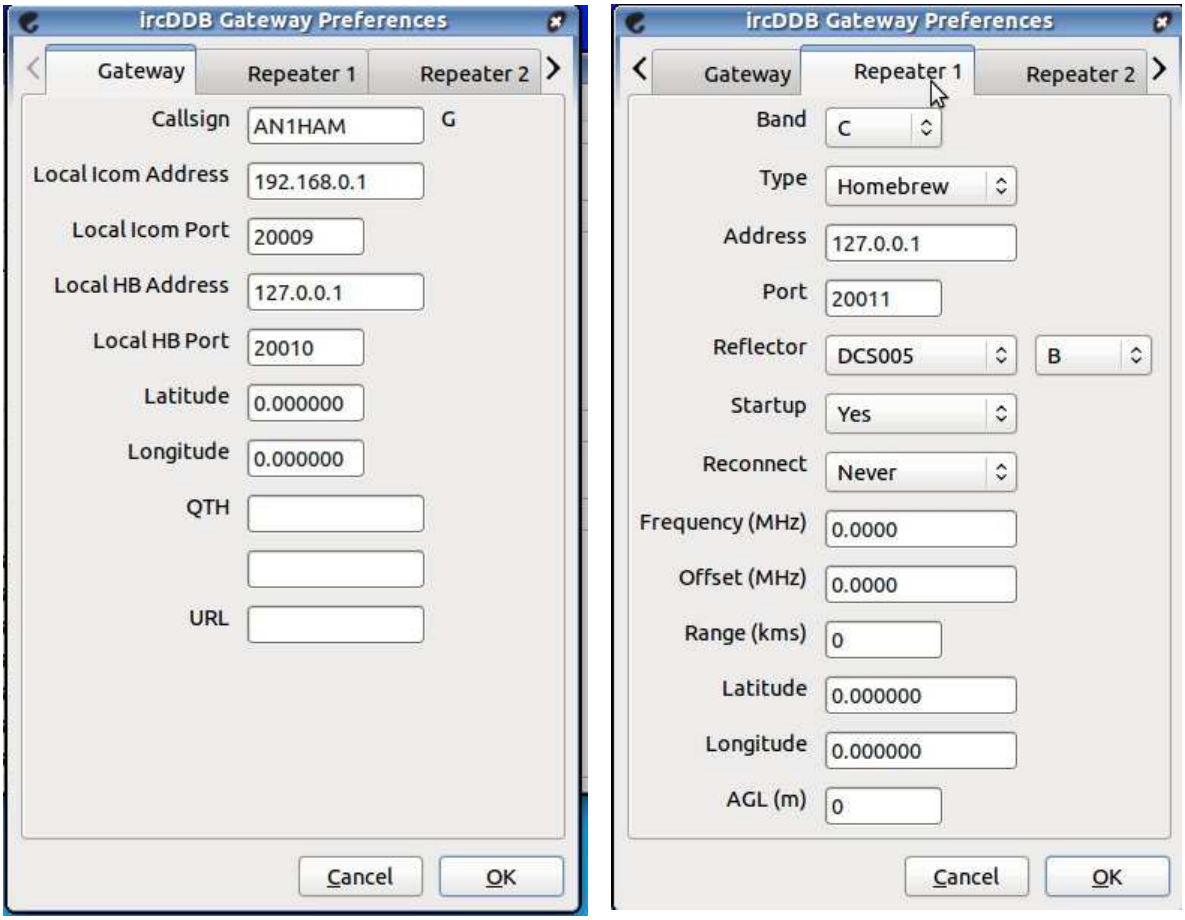


Rys. 3.11cd. W zakładce radiolatarni („**Beacon**“) należy w polu komunikatu („**Message**“) podać własny znak i ew. dalszy tekst. Pozostałe pola i ustawienia mogą zachować wartości domyślne.

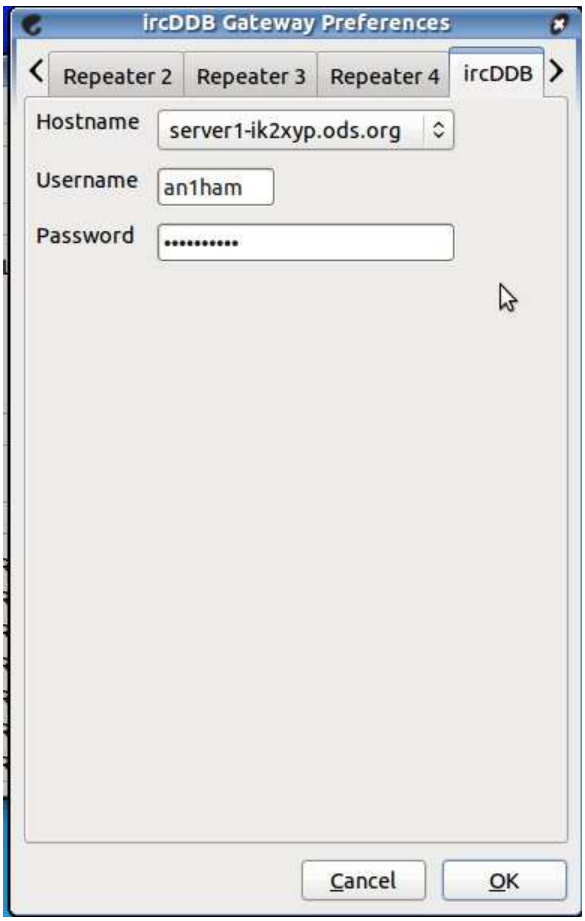


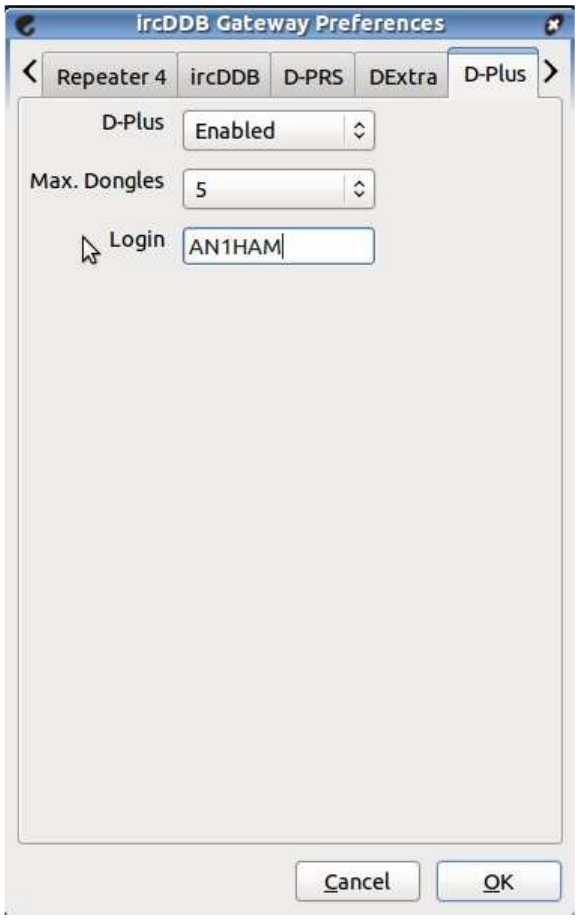
Rys. 3.11e. W polu częstotliwości roboczej („**Frequency [Hz]**”) należy wprowadzić częstotliwość pracy z dokładnością do Hz. U góry – w polu „**Port**” – widoczna systemowa nazwa radiostacji.

Ustawienia „ircDDB Gateway”



Rys. 3.12ab. Pierwsze dwie zakładki konfiguracji „ircDDB Gateway”. W polu „**Callsign**” („Znak”) zakładki bramki („**Gateway**”) należy podać własny znak wywoławczy, adres w lokalnej (domowej) sieci LAN i współrzędne geograficzne. W zakładce „**Repeater 1**” należy ustawić odpowiednią literą oznaczającą pasmo i ewentualnie wybrać domyślny używany reflektor. Współrzędne geograficzne muszą być oczywiście identyczne z podanymi w poprzedniej konfiguracji. Pozostałe parametry (również i na pozostałych zakładkach) mogą zachować wartości domyślne.

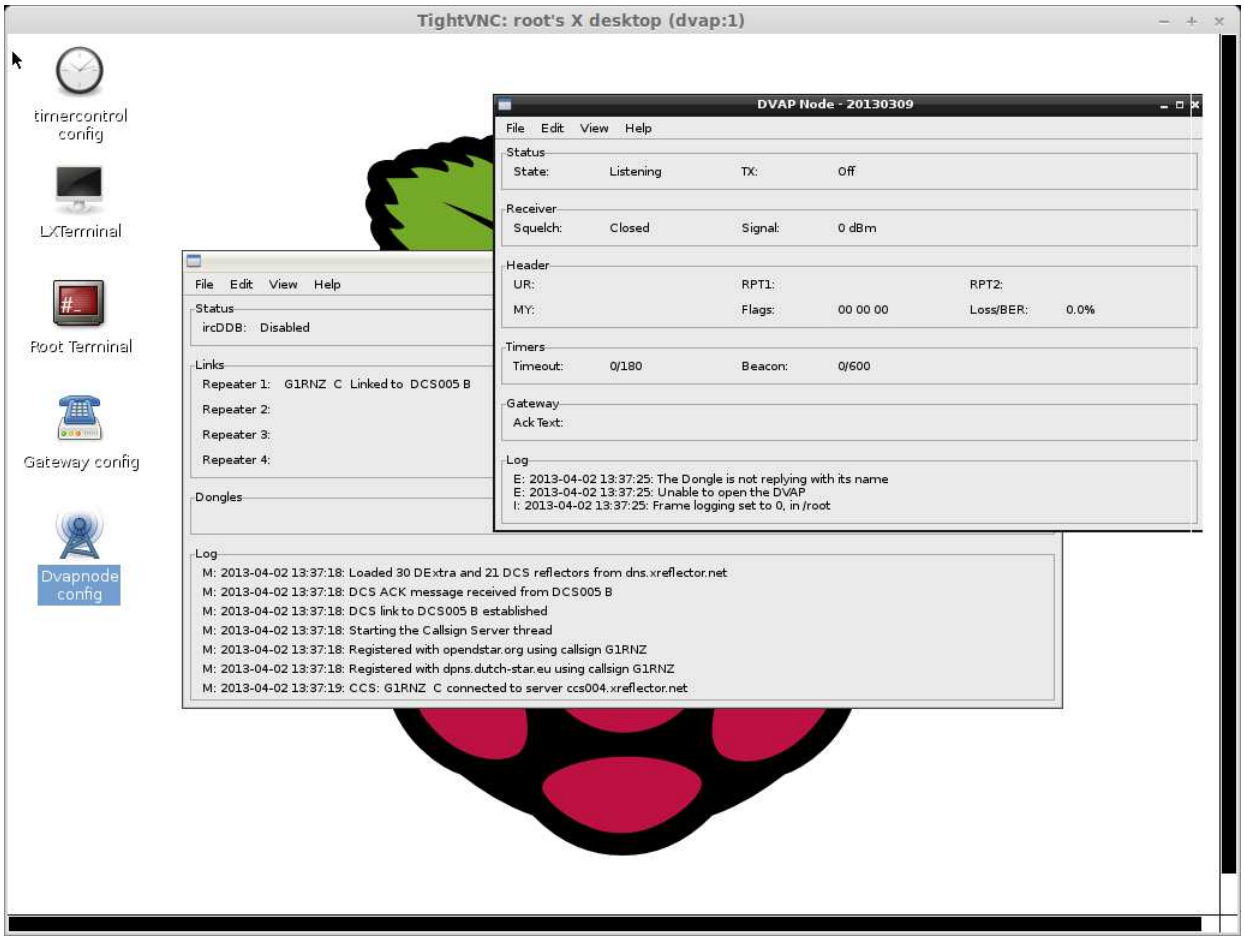




Rys. 3.12cd. W zakładce „ircDDB” wybierany jest testowy serwer dla „ircDDB Gateway” – najlepiej najbliżej położony – a poniżej podaje się nazwę użytkownika (np. własny znak) i dowolne hasło dostępu (nie jest ono wogóle używane). W zakładce „D-Plus” można włączyć „D-Plus” dla połączeń z przemiennikami lub reflektorami nie dozwolonymi domyślnie. W polu nazwy użytkownika („**Login**”) wystarczy podać własny znak wywoławczy.

Po zakończeniu kofiguracji należy zamknąć „ircDDB Gateway” po czym ponownie wystartować „DVAP Node” i jeśli licznik transmisji radiolatarni zmienia stan (co oznacza prawidłową pracę) można uruchomić „ircDDB Gateway” do zwykłego użytku.

Praca przemiennika



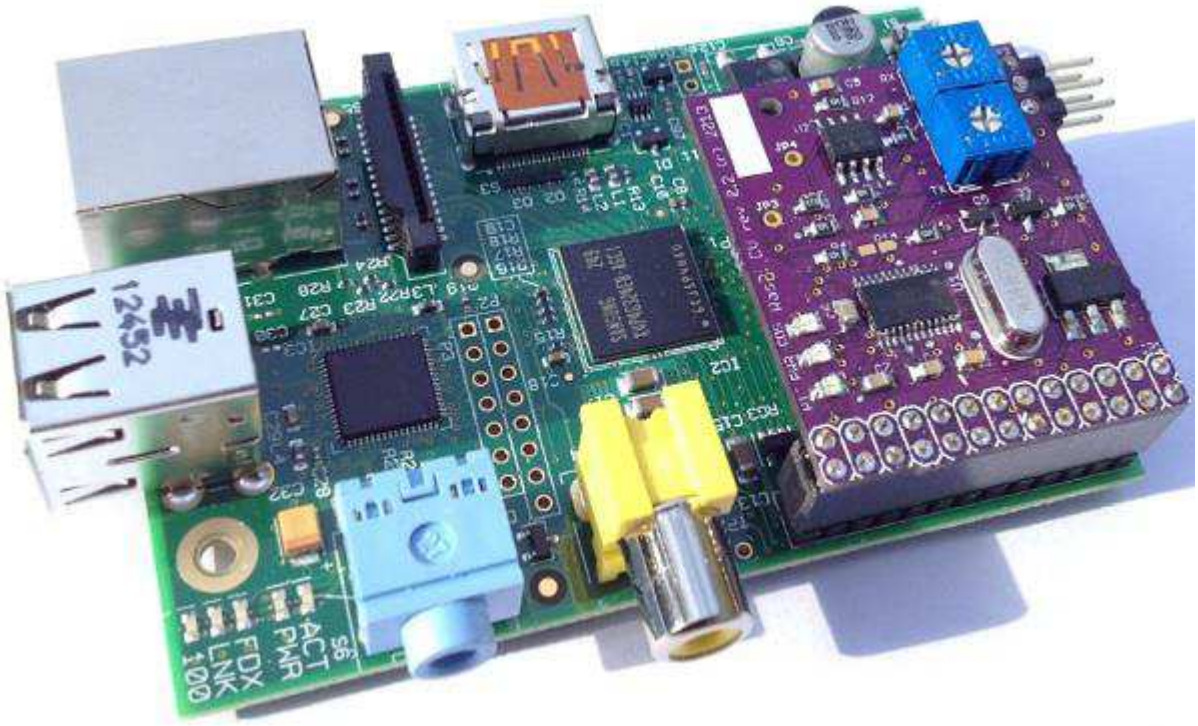
Rys. 3.13. Okna przemiennika i bramki w trakcie pracy. Widok z ekranu PC za pośrednictwem serwera „TightVNC” na „Raspberry Pi”.

Ustawienia radiostacji

Ustawienia radiostacji są zasadniczo identyczne jak dla przemiennika z „DVAP Tool” (patrz tabela 3.2) ale w polach RPT1 i RPT2 należy podać – ustalony w powyższej konfiguracji – znak przemiennika i bramki tak jak dla zwykłych przemienników sieci.
Dodatkowo do poleceń „I” (informacja) i „E” (echo) użytkownik ma do dyspozycji polecenia „HALT” i „REBOOT” służące odpowiednio do zatrzymania lub powtórnego uruchomienia przemiennika. Wywołują one podane w konfiguracji przemiennika polecenia *sudo shutdown –h now* i *sudo shutdown –r now*. Analogicznie jak wszystkie są one wpisywane do pola adresu docelowego „UR”.

Modem GMSK

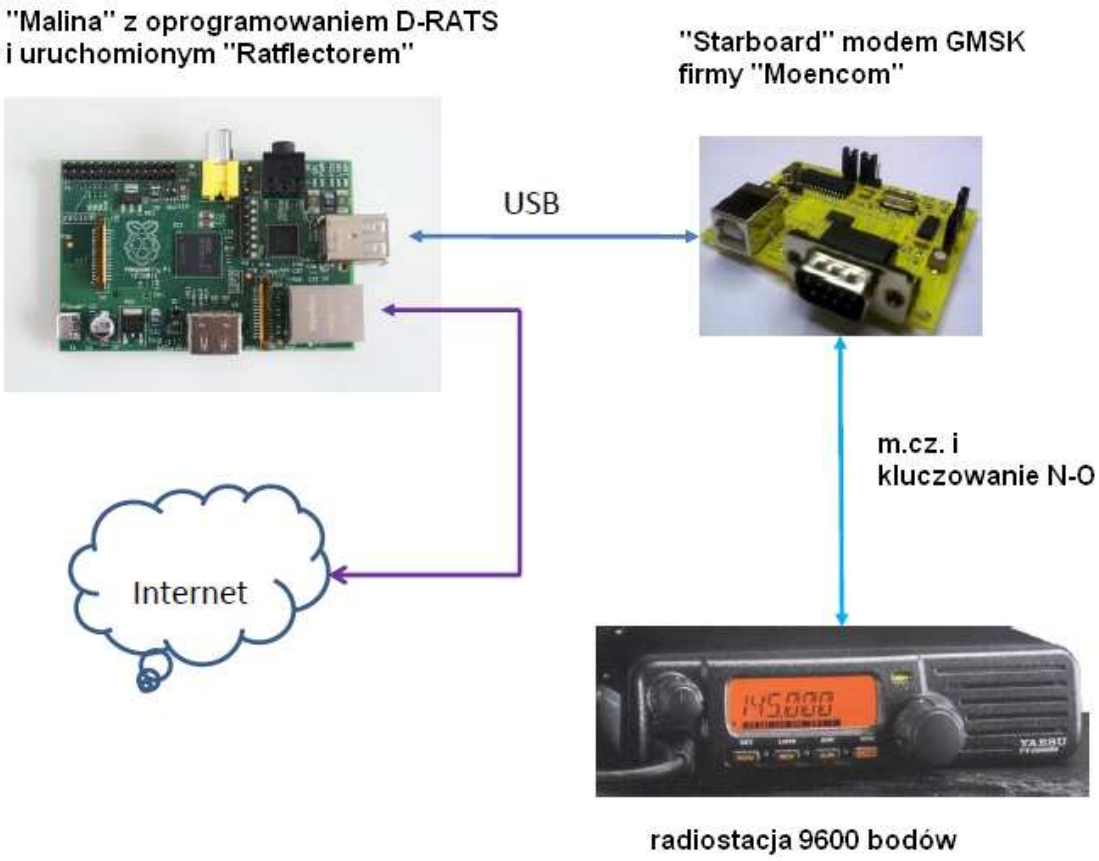
Zamiast radiostacji *DVAP Dongle* w bramkach D-Starowych można użyć także modemu GMSK w połączeniu ze zwykłymi analogowymi radiostacjami FM (w trybie wąskopasmowym FM-N). Radiostacja musi być dostosowana do transmisji danych z przepływnością 9600 bit/s. Wejście i wyjście modemu są połączone z gniazdem danych radiostacji analogicznie jak dla transmisji packet-radio z szybkością 9600 bit/s.
Przykładowe rozwiązanie modemu CMX589 w postaci płytki wtykanej na złącze GPIO „Raspberry” (*http://ki6zum.com/dstar/dv_overview.htm*) przedstawia zdjęcie 3.14. Modemy takie produkuje także firma Moencom (*www.moencomm.com*).
Zamiast programu „DVAP+ircDDB+VNC” należy w tym przypadku z Internetu pobrać program „GMSK+ircDDB+VNC” (rys. 3.8 u dołu).



Rys. 3.14. Modem GMSK dla „Maliny” na złączu GPIO

Własny „Ratflector”

Po zainstalowaniu na „Malinie” oprogramowania D-RATS (w witrynie *www.d-rats.com* dostępne są wersje dla kilku dystrybucji Linuksa, w tym dla Debiana) można przy wykorzystaniu modemu GMSK i radiostacji analogowej FM lub radiostacji D-Starowej z przejściówką USB/RS-232 uruchomić własny „Ratflector”. Więcej informacji na temat programu D-RATS i konfiguracji „Ratflectora” znajduje się w drugim tomie „Biblioteki polskiego krótkofalowca”. Przykładowe rozwiązanie przedstawia rys. 3.15.



Rys. 3.15. Konstrukcja „Ratflectora” z modemem GMSK i radiostacją analogową

Instalację D-RATS najwygodniej jest przeprowadzić bezpośrednio z „Maliny” w następujących krokach:
– instalacja d-rats
sudo apt-get install d-rats
w jej trakcie automatycznie instalowane są pakiety python-libxml2, python-libxslt1 i python-serial
– ostatni niezbędny pakiet python-glade2 musi być zainstalowany ręcznie
sudo apt-get install python-glade2
– po zakończeniu instalacji (może ona w sumie zająć dłuższy czas i automatycznie instalować dodatkowe pakiety wymagane przez pakiet python-glade2 należy dokonać konfiguracji D-RATS.

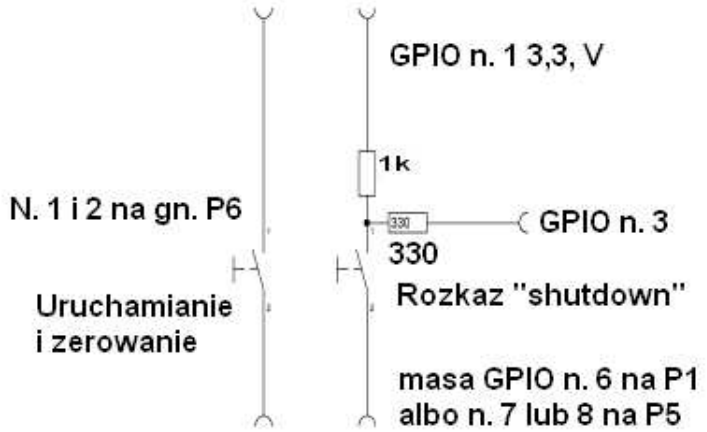
Korzystanie ze złącza szeregowego – przez przejściówkę USB/RS232 – (dla współpracy z radiostacją D-STAR zamiast modemu GMSK) może wymagać dodania użytkownika (w przykładzie użytkownika „pi”) do grupy „dialout”:

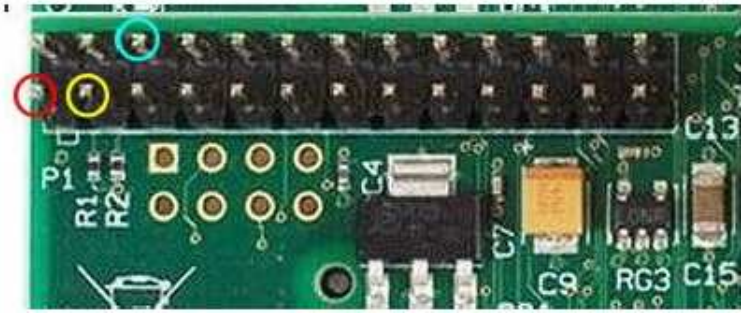
sudo usermod -a -G dialout pi

Uwaga: niektóre modele radiostacji j. np. IC-2820E wymagają poprawnych poziomów napięć na złączu szeregowym tzn. +/-12 V a nie 0/12 V. Niestety nie wszystkie modele przejściówek są zgodne z normą RS-232. W razie nieprawidłowej pracy (braku nadawania) warto sprawdzić sygnał wyjściowy TXD np. za pomocą oscyloskopu i skonsultować się z instrukcją sprzętu.

Uruchamianie i wyłączanie systemu za pomocą przycisku

W większości opisanych i wielu innych zastosowaniach technicznych „Malina" pracuje autonomicznie bez podłączonego monitora i klawiatury dlatego też bezpośrednie podanie polecenia wyłączenia systemu („**shutdown**") nie jest możliwe. Pozostaje podanie go zdalnie przez komputer za pomocą połączenia SSH lub „TightVNC" ale można to też rozwiązać korzystając z prostego skryptu w „Pythonie" i przełącznika lub klawisza podłączonego zo złącza GPIO. W poniższym rozwiązaniu naciśnięcie klawisza prawego powoduje wyłączenie systemu (podanie rozkazu „**shutdown –h now**") a lewego – podłączonego do gniazda P6 – wyzerowanie i powtórne uruchomienie systemu. Można oczywiście z niego zrezygnować pozostawiając jedynie przycisk wyłączania aby uniknąć wyłączania napięcia zasilania w trakcie pracy systemu co mogłoby doprowadzić do uszkodzenia zawartości pamięci SD. Przycisk prawy jest podłączony do gniazda P1 (masę można także połączyć z gniazdem P5).





Rys. 3.16. Podłączenie klawiszy służących do wyłączania i zerowania systemu. Poniżej widok płytki ze złączem GPIO (P1). Kolorem czerwonym zaznaczono nóżkę 1 – napięcie 3,3 V, żółtym nóżkę 3, a jasnoniebieskim nóżkę 6, masę.

Stan przycisków jest odpytywany przez pracujący rownolegle do głównego program w języku „Python". Uruchomienie go wymaga zainstalowania pakietów python-dev i python-rpi.gpio (o ile nie zostały one już wcześniej zainstalowane do innych celów):
sudo apt-get install python-dev
sudo apt-get install python-rpi.gpio

Następnie należy założyć np. za pomocą edytora *nano* plik programu
sudo nano /usr/local/bin/shutdownbutton.py

o następującej treści

```
import RPi.GPIO as GPIO
import os
import time
GPIO.setmode(GPIO.BOARD)
GPIOsetup(3, GPIO.IN)
while True:
   if not (GPIO.input(3)):
      os.system("sudo shutdown –h now")
time.sleep(.5)
```

Plikowi należy przyznać następujące prawa własności i dostępu:

sudo chown root:root /usr/local/bin/shutdownbutton.py
sudo chmod 0755 /usr/local/bin/shutdownbutton.py

i dopisać jego wywołanie do *rc.local* (przed ostatnią linią – exit()):
sudo nano /etc/rc.local

…
sudo python /usr/local/bin/shutdownbutton &
exit0

Bramki Echolinkowe

Przedstawione poniżej rozwiązania bramek Echolinkowych mogą być przydatnie nie tylko jako stałe dostępne publicznie bramki sieci ale także jako prywatne punkty dostępowe w rejonach leżących poza zasięgiem stałych bramek sieci, jako bramki eksperymentalne, bramki uruchomione na czas imprez krótkofalarskich albo akcji czy ćwiczeń ratunkowych. Mogą one też przyczynić się do wzrostu zainteresowania mniej używanymi pasmami j.np. 29, 50 czy 70 MHz i w konsekwencji do zwiększenia aktywności w nich. Mogą stanowić ośrodki spotkań dla grup o szczególnych zainteresowaniach – umożliwiając np. prowadzenie długich dyskusji technicznych, łączności z zagranicą w obcych językach albo łączności innymi emisjami zamiast fonii bez nużenia tym szerokiego grona pozostałych słuchaczy. Do połączenia z siecią Echolinku można skorzystać ze zwykłych dojść do Internetu albo wykorzystać sieć Hamnetu, tam gdzie już istnieje albo nawet uruchomić lokalne łącze Hamnetu jedynie do tego celu.

„SvxLink”

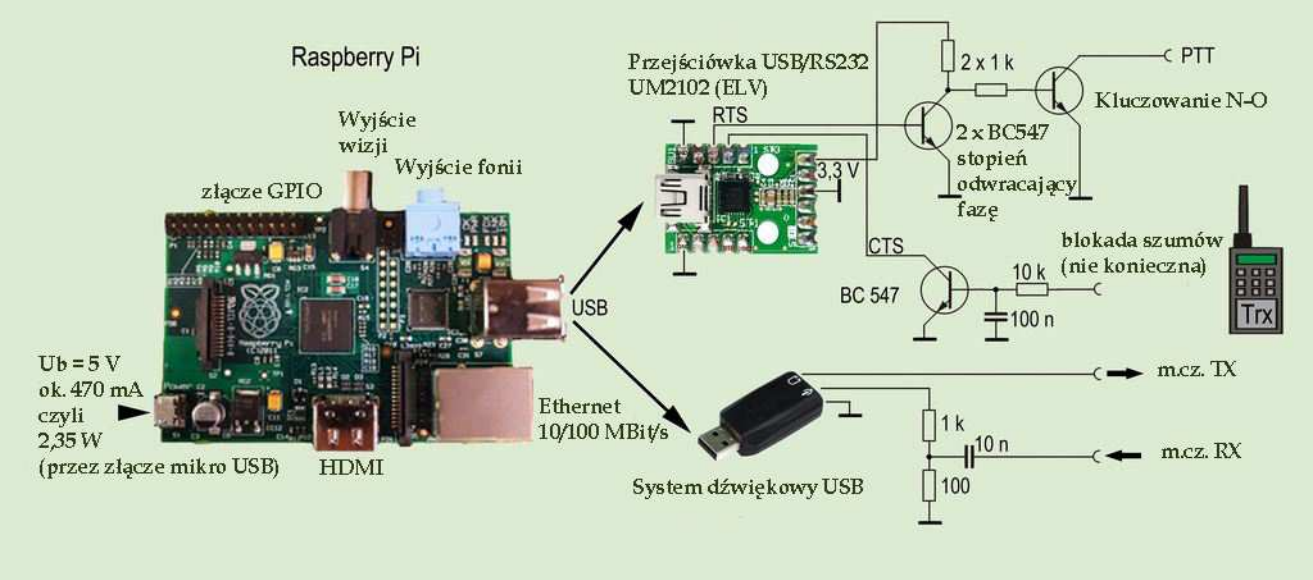
Niski pobór energii jest ważnym argumentem przemawiającym za użyciem „Raspberry Pi” w bramkach radiowo-internetowych Echolinku pracujących na okrągło albo dostosowanych do zasilania awaryjnego z akumulatorów. Mogą to być zarówno typowe bramki o większym zasięgu jak i lokalne punkty dostępowe podobne do opisanych uprzednio dla sieci D-STAR. Jest to zależne jedynie od radiowego wyposażenia bramki, potrzeb i oczywiście od uzyskanych zezwoleń.
Ogólnie rzecz biorąc uzyskania oddzielnego zezwolenia wymagają bramki i przemienniki dostępne publicznie i pracujące na okrągło albo prawie, bez bezpośredniego nadzoru operatora i pod własnym znakiem, natomiast podobne stacje o małym zasięgu uruchamiane tylko na czas potrzebny operatorowi do pracy w eterze (albo eksperymentalne) i znajdujące się pod jego bezpośrednim nadzorem mogą pracować pod znakiem operatora i w ramach jego licencji indywidualnej. Ich częstotliwość pracy nie musi być wówczas koordynowana ale oczywiście nie mogą one zakłócać pracy innych użytkowników pasma. W poszczególnych sieciach muszą być ewentualnie spełnione dodatkowe warunki o charakterze techniczno-organizacyjnym j.np. rejestracja znaku. Jest to warunkiem koniecznym do pracy w sieci D-STAR (bez rejestracji możliwa jest jedynia praca lokalna przez przemienniki lub bezpośrednia bez ich użycia ale niemożliwe jest uruchamianie własnych punktów dostępowych) ale również uruchomienie przemiennika echolinkowego wymaga rejestracji dodatkowego znaku. Dla przemienników publicznych o dostępie dupleksowym znaki te mają rozszerzenie „-R” natomiast dla pomocniczych stacji simpleksowych w sieci lub dla indywidualnych punktów dostępowych – mikroprzemienników – jest to znak operatora z rozszerzeniem „-L”.
Najbardziej rozpowszechnionym obecnie oprogramowaniem dla malinowej bramki Echolinku jest „SvxLink”. Może on być także instalowany na „Cubieboardzie”. Bezpłatny program autorstwa Tobiasa Blomberga SM0SVX ma strukturę modularną i oprócz obsługi bramki echolinkowej pozwala na uruchomienie głosowego przemiennika-papugi – powtarzającego odebrane relacje na tej samej częstotliwości, skrzynki głosowej, na współpracę z siecią APRS, z meteorologiczną lotniskową siecią METAR („**MET**eorological **A**erodrome **R**eport”), rozpowszechnianie komunikatów propagacyjnych np. otrzymywanych z sieci *vhfdx.net* itd. a planowana jest dalsza rozbudowa programu. Zawiera on programowe kodery i dekodery DTMF i CTCSS oraz kodeki (wokodery) alternatywne w stosunku do standardowego echolinkowego kodeka GSM i zapewniające lepszą jakość głosu w połączeniach z innymi stacjami „SvxLinku”. W odróżnieniu od klasycznych bramek echolinkowych polecenia dla bramek „SvxLinku” są zawsze zakończone znakiem #.
Uruchomienie oprogramowania „SvxLink” na „Raspberry Pi” wymaga jego skompilowania, ale przedtem niezbędne jest zainstalowanie dodatkowych bibliotek Libsigc++:
sudo apt-get install subversion libsigc++-2.0-dev g++ make libsigc++-1.2-dev libgsm1-dev libpopt-dev tcl8.5-dev libgcrypt-dev libspeex-dev libasound2-dev alsa-utils
a następnie po przejściu do katalogu */home* pobrania kodu programu:
sudo svn co svn://svn.code.sf.net/p/svxlink/svn/trunk svxlink-trunk

Dla zmniejszenia obciążenia CPU w trakcie pracy korzystna okazuje się zmiana częstotliwości próbkowania z 16 na 8 kHz i korekta parametrów kompilacji w pliku *makefile.cfg*:

CXXFLAGS += -DINTERNAL_SAMPLE_RATE=8000
RELEASE_CFLAGS=-g -O2 -march=armv6 -mfloat-abi=hard -mfpu=vfp

Następnie można już skompilować i zainstalować SvxLink za pomocą poleceń:
sudo make release
sudo make install

Schemat połączeń elementów pramki z radiostacją przedstawiony jest na rys. 4.1. Do przełączania nadawanie-odbiór można zastosować przejściówkę USB/RS232 jak to pokazano na rysunku albo wykorzystać wyprowadzenia na złączu GPIO.



Rys. 4.1. Połączenie „Raspberry” z radiostacją w bramce echolinkowej

Ze względu na to, że standardowy system dźwiękowy „Maliny” nie posiada niezbędnego dla Echolinku wejścia mikrofonowego (posiada je natomiast „Banana Pi”) konieczne jest użycie zewnętrznego systemu dźwiękowego USB. Aby system mógł z niego korzystać konieczne jest zadeklarowanie go w pliku */etc/modules* przez dodanie wpisu „snd-usb-audio” i skasowanie lub wykomentowanie ewentualnych linii odnoszących się do systemu standardowego o treści „snd-bcm2835” lub podobnych.
Konieczna jest też zmiana w pliku */etc/modprobe.d/alsa-base.conf*: w linii „options snd-usb-audio index=–2” na końcu należy wpisać „0”.
Trzeciej niezbędnej zmiany należy dokonać w pliku */etc/asound.conf*, zmieniając jego zawartość na:

```
pcm.!default {
  type plug
  slave {
    channels 1
      pcm “hw:0,0"
  }
}
pcm.low {
  type plug
  slave {
    pcm “hw:0,0"
  }
}
```

W plikach konfiguracyjnych *Echolink.conf* i *Svxlink.conf* (znajdujących się w katalogach */etc/svxlink* i */etc/svxlink/svxlink.d* należy wprowadzić dane dotyczące stacji i jej operatora.

W pliku *Echolink.conf* są to przede wszystkim linie CALLSIGN (znak wywoławczy), PASSWORD (hasło dostępu), SYSOPNAME (imię operatora), LOCATON (QTH i QRG) oraz ewentualnie wymuszenia korzystania wyłącznie z kodeka GSM (USE_GSM_ONLY=1).
W pliku *Svxlink.conf* należy przede wszystkim podać częstotliwość próbkowania podsystemu dźwiękowego (44100), w linii MODULES zadeklarować moduł echolinkowy (ModuleEchoLink) lub inne w zależności od stopnia rozbudowy stacji, podać znak wywoławczy (w linii CALLSIGN) oraz skorygować zawartość poniższych linii:
AUDIO_DEV=**alsa:default**
…
SERIAL_PORT=**/dev/ttyUSB0**
…
AUDIO_DEV=**alsa:hw:0**
…
PTT_PORT=**/dev/ttyUSB0**

Szczegółowa instrukcja konfiguracji i uruchomienia „SvxLinku" znajduje się w Internecie m.in. pod adresami [21], [23] oraz w poz [24].
Automatyczny start programu po włączerniu „Raspberry" uzyskuje się poprzez dodanie do pliku */etc/rc.local* wpisu:
 # Uruchomienie Svxlinku
sleep 10
svxlink –daemon
sleep 5
echo „SvxLink uruchomiony …"

Polecenia dla „SvxLinku"

SvxLink składa się z modułów wywoływanych za pomocą liczb z dodatkiem krzyżyka – w kodzie DTMF. Oczywiście, zależnie od wyposażenia stacji, nie wszystkie moduły muszą i mogą być aktywne – zwłaszcza na mikrokomputerach typu „Raspberry". Stosunkowo najczęściej uruchamiany i używany jest moduł Echolinku.

Tabela 4.1 Moduły „SvxLinku" i ich wywołanie

Wywołanie modułu	Moduł
0#	Pomoc: polecenie 1# – pomoc dla modułu papugi, 2# – pomoc dla modułu Echolinku itd.
1#	Papuga – przemiennik zapamiętujący i powtarzający relacje na tej samej częstotliwości, 0# – dodatkowa pomoc dla papugi
2#	Echolink (patrz tabela 4.2)
3#	informacje METAR
4#	skrzynka głosowa: 1# – odczyt wiadomości (po odsłuchaniu wiadomości 1 – skasowanie, 2 – nadanie odpowiedzi i skasowanie, 3 – powtórne odtworzenie), 2# – nadanie wiadomości,
5#	informacje propagacyjne, są nadawane głosem po ich otrzymaniu
6#	wywołanie selektywne (koder)
#	wyjście z używanego modułu

Tabela 4.2. Najważniejsze polecenia echolinkowe dla „SvxLinku”

Polecenie	Znaczenie
0#	wywołanie pomocy
1#	informacja o aktywnych połączeniach
2#	podanie numeru bramki w sieci Echolinku
31#	nawiązanie połączenia z przypadkowo wybraną bramką –R lub -L
32#	nawiązanie połączenia z przypadkowo wybranym serwerem konferencyjnym
4#	ponowne nawiązanie ostatniego połączenia
51#	włączenie trybu nasłuchu
52#	wyłączenie trybu nasłuchu
6* kod#	nawiązanie połączenia przez podanie znaku stacji zakończonego krzyżykiem lub kropką jeśli podana tylko jego część

„Thelinkbox”

W bramkach echolinkowych można też zamiast *SvxLinku* użyć opracowanego przez ON1ARF programu *Thelinkbox*. Nie daje on wprawdzie tylu możliwości co poprzedni, ale jest w zupełności wystarczający do uruchomienia bramki dla szerszego grona użytkowników lub tylko prywatnej czy eksperymentalnej. Program współpracuje z zewnętrznym podsystemem dźwiękowym USB, a do kluczowania nadajnika można wykorzystać albo tranzystor wykonawczy sterowany przez sygnał z nóżki 18 GPIO albo przejściówkę USB/RS232. Parametry konfiguracyjne programu podawane są w pliku */home/tlb/tlb.conf*.
Linuks i systemy od niego pochodne posiadają przeważnie kilka różnych sterowników API dla systemu dźwiękowego. Do najważniejszych z nich należą OSS (*open sound system*), ALSA (*Advanced Linux Sound Architecture*) i portaudio (uniwersalny API dla różnych systemów). ThelinkBox korzysta ze sterownika OSS.
Ponieważ „Raspbian” jest standardowo wyposażony w sterownik ALSA konieczna jest dodatkowa instalacja OSS. Jest to właściwe dodatkowy program pośredniczący pomiędzy sterownikiem ALSA i programami użytkowymi. Zainstalowanie ALSA-OSS wymaga następujących poleceń:
sudo apt-get install alsa-tools alsa-oss
sudo modprobe snd-pcm-oss
sudo modprobe snd-mixer-oss

Po zakończeniu instalacji należy za pomocą polecenia
ls –l /dev/dsp
sprawdzić, czy zainstalowane są dwa urządzenia
/dev/dsp i
/dev/dsp1

Plik */dev/dsp* jest związany ze standardowym systemem dźwiękowym „Maliny” bcm2835, a */dev/dsp1* – z zewnętrznym systemem USB. Po sprawdzeniu należy jeszcze na końcu pliku */etc/modules* dodać następujące wpisy:
snd-mixer-oss
snd-pcm-oss

Pik ten zawiera oprócz tego linię:
snd-bcm2835

Po ponownym uruchomieniu systemu należy jeszcze raz sprawdzić za pomocą *ls –l /dev/dsp* czy w dalszym ciągu zainstalowane są urządzenia *dsp* i *dsp1*.

W celu zainstalowania i skompilowania programu bramki echolinkowej należy posłużyć się poleceniami:

wget http://download.thelinkbox.net/thelinkbox-latest.tgz
tar zxvf thelinkbox-latest.tgz
*cd /home/pi/thelinkbox**
./configure
make && sudo make install

Po założeniu w katalogu */home/pi* podkatalogu *tlb* należy skopiować do niego przykładowe pliki konfiguracyjne:
cp ../thelinkbox-0.53/tlb.conf.sample tlb.conf
cp ../thelinkbox-0.53/port.conf.sample port0.conf
cp ../thelinkbox-0.53/tlb.cmds.sample tlb.cmds
a następnie w pliku *tlb.conf* wpisać własne dane od znaku wywoławczego poczynając:
ConferenceCall = OE1KDA-L
WorkingDid = /home/pi/tlb ; zamiast /usr/home/tlb
include /home/pi/tlb/port0.conf ; zamiast /usr/home/tlb/port0.conf
ConfEnable =0 ; niedozwolone konferencje między użytkownikami
ConferencePass = <własne hasło>
ConferenceQTH = <QTH>
EmailAdr = <własny adres elektroniczny>
RunAsUser = pi
ConfTimeout = 3000

Dla wykorzystania wyjścia na złączu GPIO do kluczowania nadajnika należy zmienić w pliku *port0.conf* wpisy na następujące:
TxKeyMethod = 7 ; dla kluczowania przez przejściówkę USB wartość 3 (RTS)
; jako DevName podać /dev/ttyUSB0
GpioSysClassId = 17 ; dla przewodu 17
a poza tym
RxCosMetrod = 0 ; włączenie VOX-u
AudioDevice = /dev/dsp1
PCMRate = 48000 ; usunąć znak komentarza na początku linii
CWID = <własny znak>
VoxThreshold = 500
VoxHoldTime = 2250

Wywołaniem „Thelinkboxu” jest następujace polecenie:
/usr/local/libexec/tlb –f /home/pi/tlb/tlb.conf
Jednym ze sposobów automatycznego wywołania programu jest wpisanie powyższej linii do pliku */etc/rc.local*.

Poziomy sygnałów dźwiękowych można skorygować w systemowym mikserze – programie *alsamixer*. Najwygodniej jest połączyć się z serwerem ECHOTEST, nagrać się i po odsłuchaniu własnego nagrania skorygować odpowiednio siłę głosu. Po ustawieniu pożądanej siły głosu należy wyjść z *alsamixera* za pomocą klawisza ESC i zapisać ustawienia posługując się poleceniem
sudo alsactl store
po czym rozłączyć się z serwerem echa.
Plik *tlb.cmds* zawiera dostępne rozkazy DTMF.

Odbiornik w sieci lokalnej

Paluszkowy odbiornik DVB-T w połączeniu z „Maliną” może służyć nie tylko do odbioru APRS ale i jako odbiornik SDR dostępny w sieci lokalnej ([28], [29]) albo nawet przez Internet. Odbiorniki programowalne są najczęściej podłączanie do komputerów bezpośrednio albo przy użyciu stosunkowo krótkich kabli USB. Połączenie ich z siecią domową pozwala nie tylko na dostęp z dowolnego komputera ale przede wszystkim na umieszczenie odbiornika dalej od domowych źródeł zakłóceń, a szczególnie od zakłóceń powodowanych przez komputery.

Rys. 5.1. Praca odbiornika DVB-T w sieci

Sposób uruchomienia odbiornika na „Raspberry” jest już wprawdzie opisany w rozdziale poświęconym bramkom APRS ale aby uniknąć niejasności warto przypomnieć najważniejsze kroki. W odróżnieniu od poprzedniego przypadku dodatkowo do instalacji programu obsługującego odbiornik na „Raspberry” (ale bez oprogramowania bramki „iGate”) konieczne jest też uruchomienie programu odbiorczego na PC i skonfigurowanie go tak, aby zamiast z odbiornika podłączonego do własnego złącza USB odbierał dane przez domową sieć LAN. Jednym z przydatnych do tego celu programów jest SDR#.
W zależności od modelu odbiorniki takie pokrywają zakresy ok. 25–1900 MHz lub 64–1700 MHz ale z dodatkowym konwerterem mogą pracować także i na falach krótkich.
Instalacja oprogramowania na „Malinie” wymaga następujących kroków:

```
cd ~/src
sudo apt-get install git build-essential cmake libusb-1.0-0-dev
git clone git://git.osmocom.org/rtl-sdr.git
cd rtl-sdr
mkdir build
cd build
cmake .. -DINSTALL_UDEV_RULES=ON
make
sudo make install
sudo ldconfig
```

Założony w trakcie instalacji plik */home/pi/rtl-adr.rules* należy skopiować pod lekko zmienioną nazwą:
/etc/udev/rules.d
do katalogu */etc/udev*.
Następnie należy podłączyć odbiornik, sprawdzić poprawność jego działania i ewentualnie ustawić wzmocnienie po wywołaniu programu:

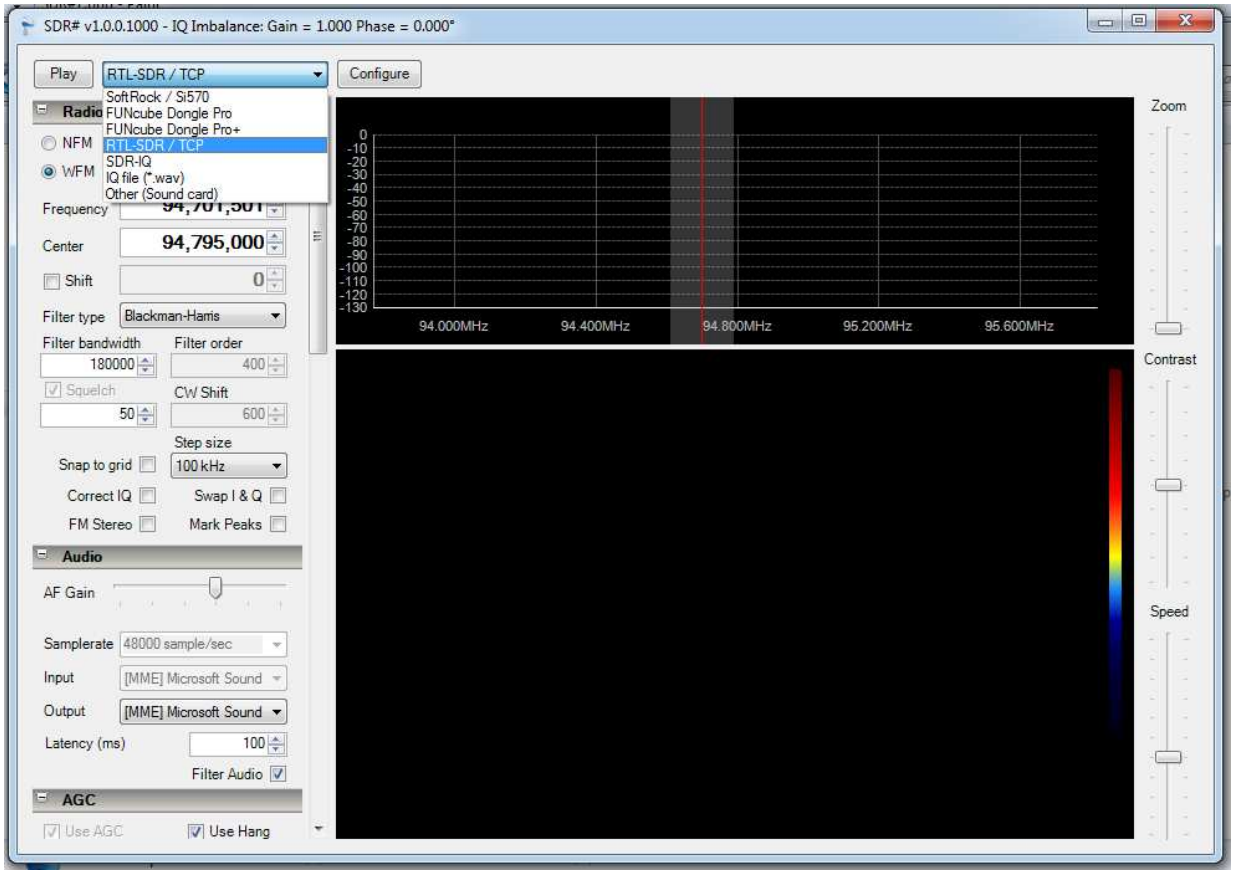
```
rtl_test
```

W przypadku wystąpienia trudności w uruchomieniu odbiornika można usunąć z pamięci „Maliny” moduł *dvb_usb_rtl28xxu* za pomocą polecenia

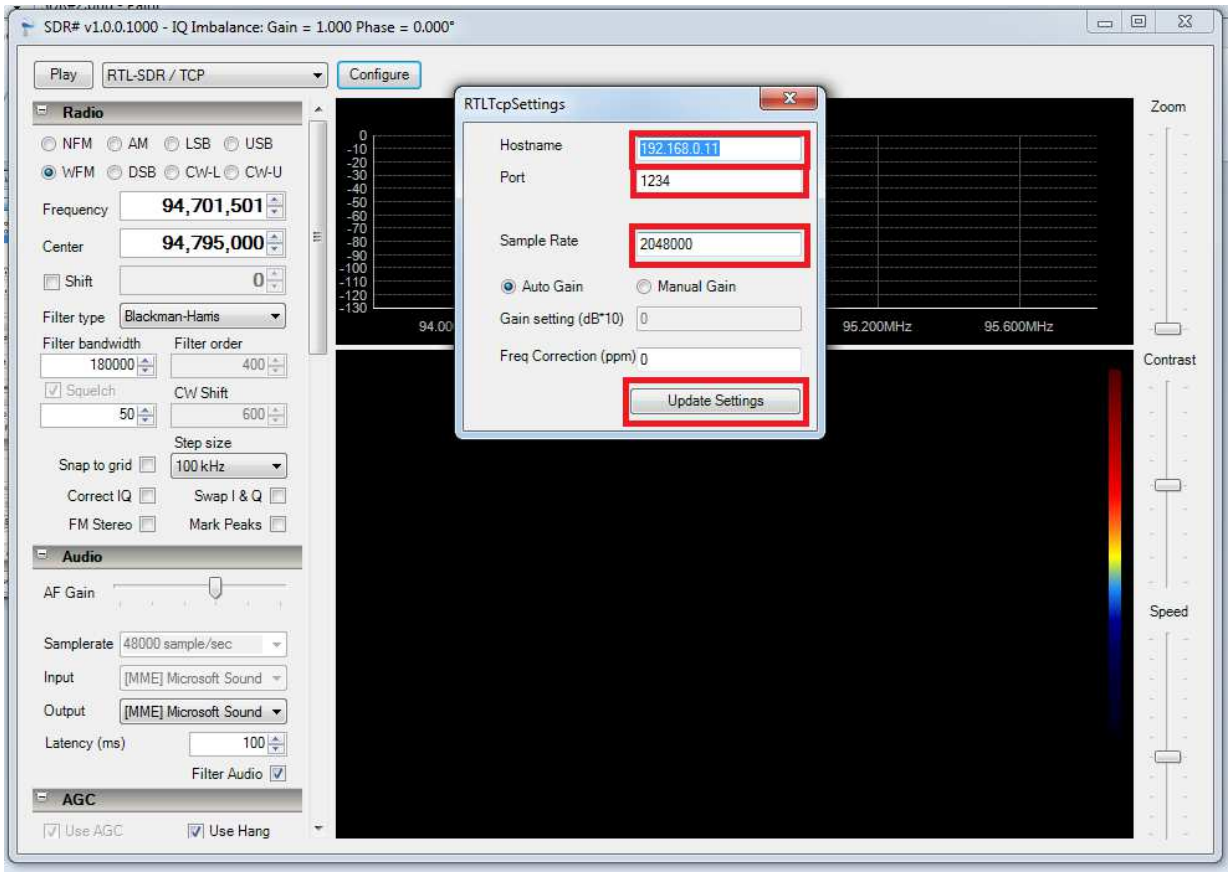
sudo mmod dvb_usb_rtl28xxu

Do wywołania spisu znajdujących się w pamięci modułów służy rozkaz *lsmod*.
Gdy *rtl_test* nie zgłasza już występowania żadnych błędów przychodzi pora na uruchomienie serwera radiowego za pomocą polecenia
rtl_tcp –a <adres IP>

W poleceniu podawany jest adres „Maliny” w sieci.
Na komputerze PC użytkownika uruchamiany jest program odbiorczy SDR#. W jego konfiguracji w spisie odbiorników należy wybrać odbiornik „RTL-SDR / TCP“ (rys. 5.2).



Rys. 5.2. Wybór odbiornika w konfiguracji SDR#



Rys. 5.3. Dalsza konfiguracja odbiornika

Rys. 5.4. Po dodaniu przedwzmacniacza możliwy jest odbiór satelitów amatorskich

Ilustracja 5.4 przedstawia przykład użycia odbiornika DVB-T, w połączeniu z „Maliną”, do odbioru satelitów amatorskich. W konstrukcji użyto niskoszumnego przedwzmacniacza LNA70-1 produkcji

DG0VE (*www.dg0ve.de*) ale oczywiście może to być przedwzmacniacz dowolnego typu i konstrukcji. Jest on niezbędny ze względu na stosunkowo niską czułość odbiorników DVB-T tego typu. Do śledzenia przelotów satelitów znakomicie nadaje się „Orbitron" autorstwa Sebastiana Stoffa (*www.stoff.pl*). Zamiast odbiornika DVB-T można także zastosować „Fun Cube Dongle Pro+". Dzięki jego większej czułości dodatkowy przedwzmacniacz może okazać się zbędny. W jednym i w drugim przypadku konieczna jest antena zewnętrzna.

Odbiornik „Fun Cube Dongle Pro+"

Rys. 5.5. Odbiornik „FCDPro+" podłączony do „Maliny"

Odbiornik *FCDPro+* korzysta ze złączy USB i HID i zasadniczo nie wymaga instalacji żadnych dodatkowych sterowników (to samo dotyczy również i innych systemów operacyjnych j.np. „Windows"). Konieczne jest jednak zainstalowanie programu sterującego odbiornikiem – np. opisanego dalej *fcdcontrol*. Może on być obsługiwany bezpośrednio lub w lokalnej sieci za pomocą SSH.
Instalacja programu sterującego wymaga następujących kroków (niektóre z nich j.np. instalacja biblioteki *libusb* mogą nie być konieczne o ile została ona już wcześniej zainstalowana w związku z innymi potrzebami):
pi@raspberrypi ~ $ *sudo apt-get install git-core libusb-1.0-0-dev*

następnie dla „FCDPro"
pi@raspberrypi ~ $ *wget https://raw.github.com/csete/qthid/master/funcube-dongle.rules*
pi@raspberrypi ~ $ *sudo cp funcube-dongle.rules /etc/udev/rules.d/*

lub dla „FCDPro+"
pi@raspberrypi ~ $ *wget https://raw.github.com/csete/qthid/fcdpp/funcube-dongle-proplus.rules*
pi@raspberrypi ~ $ *sudo cp funcube-dongle-proplus.rules /etc/udev/rules.d/*

kolejnym krokiem jest kompilacja właściwego programu
pi@raspberrypi ~ $ *git clone git://gitorious.org/~csete/fcdcontrol/fcdcontrol-proplus.git fcdcontrol-proplus*

```
pi@raspberrypi ~ $  cd fcdcontrol-proplus
pi@raspberrypi ~/fcdcontrol-proplus $ make
```

dla „FCDPro+" zamiast *make* należy w ostatniej linii podać *make fcdpp*.

Po zakończeniu kompilacji w katalogu pojawia się plik programu o nazwie *fcdctl*.
Po jego wywołaniu wyświetlana jest następująca informacja:

```
pi@raspberrypi ~/fcdcontrol-proplus $ ./fcdctl
  FCDcontrol V 0.4.1-fcdpp
  USAGE: ./fcdctl options [arguments]
   -l  --list              List all FCDs in the system
   -s  --status            Gets FCD current status
   -f  --frequency <frequency>   Sets FCD frequency in MHz
   -g  --gain <gain>          Enable/disable LNA gain (0 or 1)
   -i  --index <index>        Which dongle to show/set (default: 0, i.e. first)
   -h  --help                 Shows this help
```

Poniżej podano przykłady wywołania programu z różnymi parametrami:

```
pi@raspberrypi ~/fcdcontrol-proplus $ ./fcdctl -l
 nr  USB path      firmware  frequency           LNA gain  audio device
 0   0001:000a:02  20.01      97.300000 MHz  disabled   (not found)

pi@raspberrypi ~/fcdcontrol-proplus $ ./fcdctl -s
 FCD present in application mode.
 FCD firmware version: 20.01.
 FCD frequency: 97.300000 MHz.
 FCD LNA gain: disabled.

pi@raspberrypi ~/fcdcontrol-proplus $ ./fcdctl -f 144.567
 Freq set to 144.567000 MHz.

pi@raspberrypi ~/fcdcontrol-proplus $ ./fcdctl -g 1
 LNA gain enabled.
```

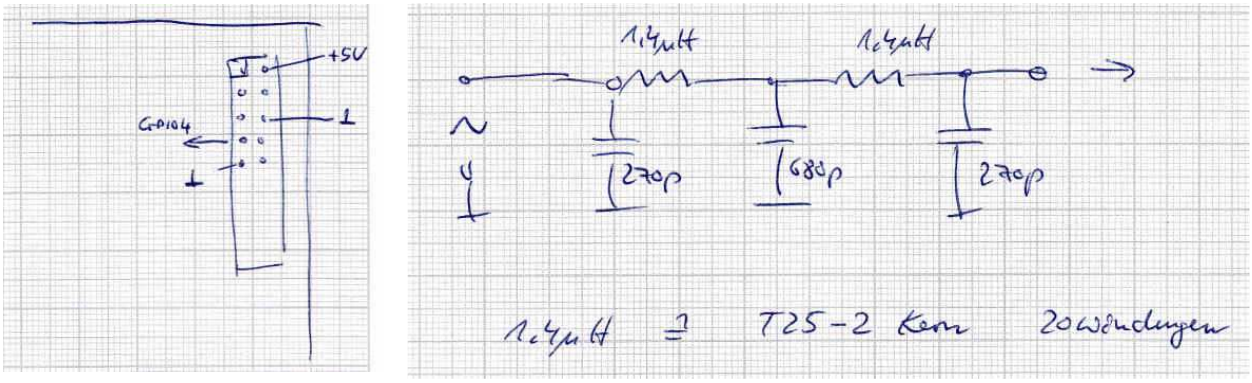
Radiolatarnia WSPR

Kolejnym wartym omówienia rozwiązaniem jest konstrukcja radiolatarni WSPR małej mocy na amatorskie zakresy fal długich, średnich i krótkich. Opracowany przez PE1NNZ program „WsprryPi" [12] jest wzorowany na dostępnym pod adresem [10] programie „pifm", dzięki któremu „Raspberry" może pracować w szerokim zakresie częstotliwości od 1 do 250 MHz jako nadajnik FM o mocy 10 mW. Do nóżki 7 gniazda GPIO – sygnału GPIO4 (i masy – kontakt 9) należy dołączyć filtr dolnoprzepustowy zapewniający stłumienie harmonicznych (rys. 6.1) a do wyjścia filtru podłączana jest antena nadawcza. W niektórych konstrukcjach stosowany jest dodatkowy wzmacniacz podwyższający moc wyjściową do kilkudziesięciu lub 100 mW (rys. 6.2 i 6.3) ale inni eksperymentatorzy obniżali za pomocą dodatkowych tłumików moc wyjściową nawet do 100 µW osiągając przy tym na KF zasięgi europejskie [13]. Napięcie zasilania powinno być dobrze stabilizowane i pozbawione przydźwięku sieci, który mógłby zakłócać sygnał nadawany. Więcej przykładów rozwiązań wzmacniaczy mocy i filtrów dolnoprzepustowych dla radiolararmi małej mocy zawiera tom 17 niniejszej serii.
Przed rozpoczęciem pracy w eterze należy nastawić czas systemowy z dokładnością do 1 sekundy lub synchronizować go z jednym z internetowych serwerów NTP. Niezbędne jest także przeprowadzenie kalibracji częstotliwości aby bezbłędnie utrafić w stosunkowo wąskie podzakresy WSPR. Konieczną do tego celu poprawkę częstotliwości można uwzględnić w wywołaniu programu lub korygując odpowiednio wartość częstotliwości zegarowej w kodzie programu (stałej F_XTAL; patrz dodatek A).
Stosunkowo najprostszym sposobem kalibracji częstotliwości jest dostrojenie sygnału nadawanego dokładnie do częstotliwości odbieranej stacji radiofonicznej (czyli na zero dudnień) a następnie porównanie ustawienia częstotliwości nadawania „Maliny" z rzeczywistą równą częstotliwości stacji. Otrzymaną poprawkę uwzględnia się następnie dostrajając nadajnik do pasm WSPR.
W trakcie uruchamiana i pracy nadajnika należy zwrócić szczególną uwagę aby nie przekroczyć dopuszczalnego prądu obciążenia i zakresu napięć na wyjściu GPIO (0 – 3,3 V). Filtr dolnoprzepustowy należy włączyć przez kondensator separujący, unikać zwarć w układzie i narażenia „R-Pi" na wpływ elektryczności statycznej. Antenę najlepiej podłączyć do nadajnika przez transformator separujący albo dodatkowy stopień separujący – tranzystorowy albo nawet na bramkach logicznych CMOS na niższych pasmach KF. Wyjście można także dodatkowo zabezpieczyć przed przepięciami za pomocą diod.
Program jest dostępny bezpłatnie na zasadach licencji GNU.
Przed zainstalowaniem „WsprryPi" i innych omawianych dalej programów warto zadbać o aktualizację systemu operacyjnego, tak aby korzystać z jego możliwie najnowszej wersji.
Procedura instalacji lub aktualizacji „WsprryPi" jest prosta i wymaga podania następujących poleceń:

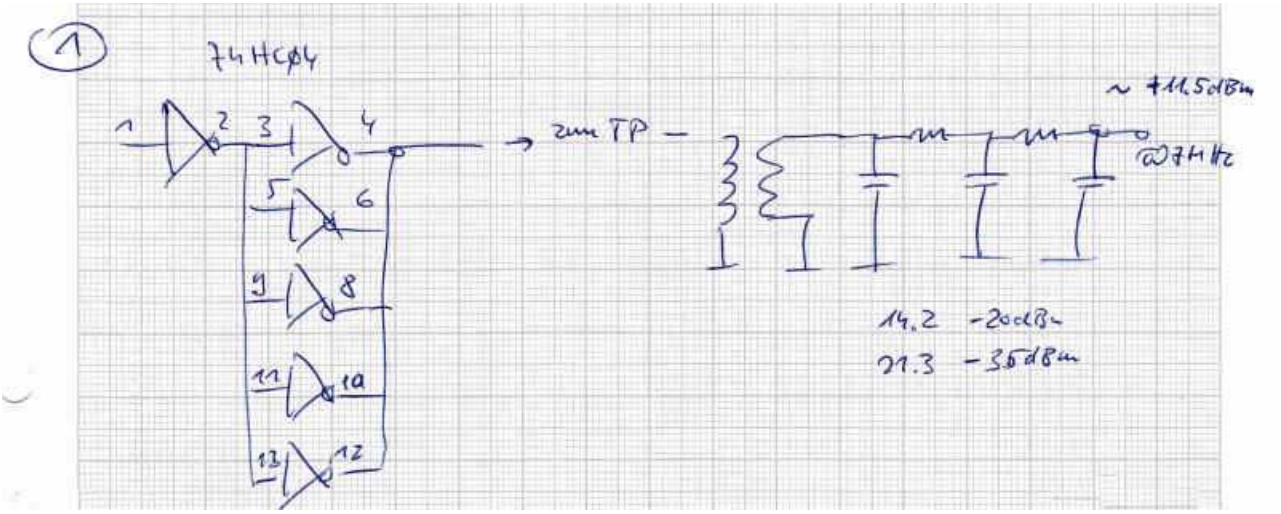
```
sudo apt-get install git
rm -rf WsprryPi
git clone https://github.com/threeme3/WsprryPi.git
cd WsprryPi
```

Znak wywoławczy, lokator stacji i moc podawane są wywołaniu:
sudo ./wspr <[prefiks]/znak[/sufiks]> <lokator> <moc w dBm> [<częstotliwość w Hz> ...]
np.: *sudo ./wspr OE1KDA JN88 10 7040074 0 0 10140174 0 0*
gdzie zera oznaczają odcinki czau bez nadawania.
Wybór emisji WSPR lub WSPR-15 następuje automatycznie w zależności od zakresu pracy (patrz tabela 6.1): WSPR-15 stosowana jest jedynie w wybranych wycinkach pasm 2200, 630 i 160 m (patrz: skrypt nr 23, „Technika słabych sygnałów" tom 3).
Po wprowadzeniu poprawek w kodzie źródłowym programu (np. stałej F_XTAL) należy go skompilować (po uprzednim zainstalowaniu kompilatora gcc):

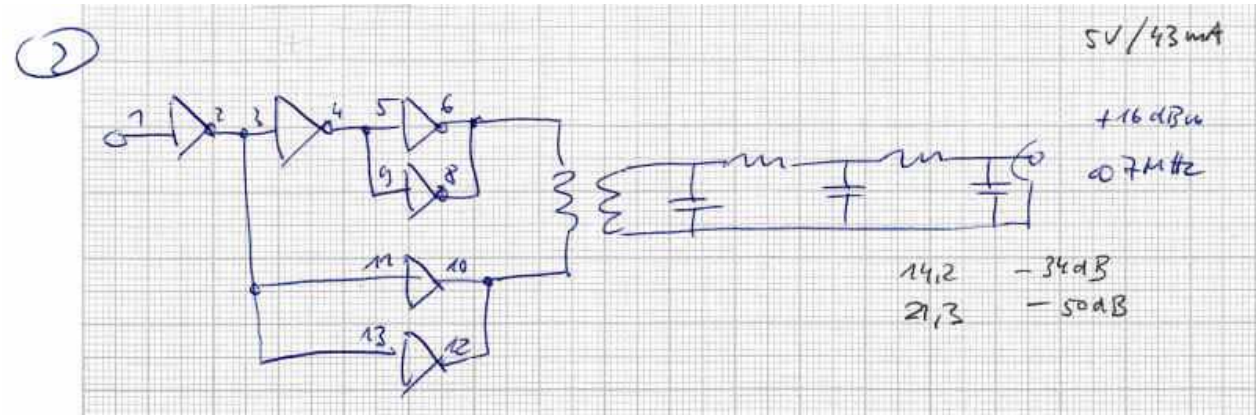
```
sudo apt-get install gcc
gcc -lm -std=c99 wspr.c -owspr
```



Rys. 6.1. Podłączenie filtru dolnoprzepustowego do „Raspberry”. Filtr po prawej stronie jest obliczony dla pasma 7 MHz i zawiera dwie indukcyjności po 1,4 µH. Są one nawinięte na rdzeniach pierścieniowych T25-2 (czerwonych) i mają po 20 zwojów. Filtr jest nieskomplikowany i stosunkowo łatwo można go przeliczyć na inne pasma amatorskie. Sygnał wyjściowy jest pobierany z nóżki GPIO4.



Rys. 6.2. Dodatkowy wzmacniacz mocy na równolegle połączonych bramkach logicznych 74HC04 (można zastosować inne typy np. popularne 74HC240). Przy zasilaniu napięciem 5 V wzmacniacz daje ok 11 dBm mocy wyjściowej. Filtr dolnoprzepustowy włączony jest przez transformator o przekładni 1:1. Uzyskano tłumienie harmonicznych ok. 32 dB dla pasma 14 MHz i ok. 47 dB dla pasma 21 MHz w stosunku do nośnej.

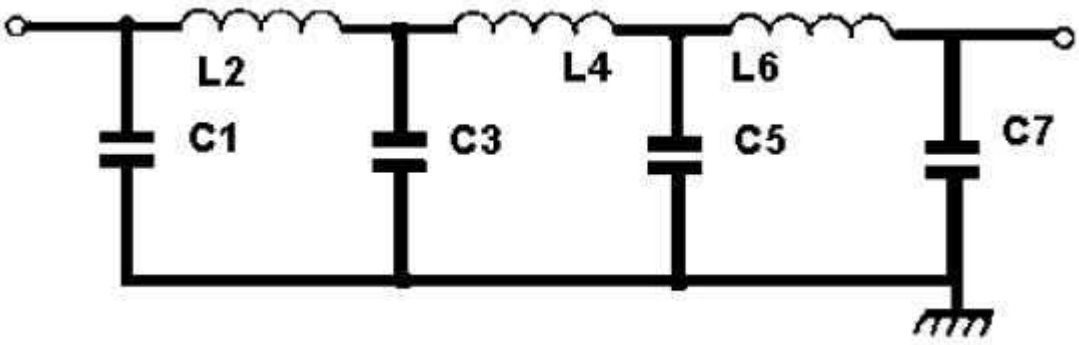


Rys. 6.3. Przeciwsobny wzmacniacz mocy zapewnia lepsze stłumienie parzystych harmonicznych. Moc wyjściowa wzrasta do ok. 16 dBm (ok. 40 mW), tłumienie harmonicznych dla pasma 14 MHz – do 50 dB i dla pasma 21 MHz do 66 dB w stosunku do nośnej. Transformator o przekładni 1:1 jest nawinięty

na rdzeniu ferrytowym. Dla poprawy tłumiernia wyższych harmonicznych można włączyć dodatkowo filtr dolnoprzepustowy o częstotliwości granicznej 30 MHz.

Tabela 6.1. Ustalone międzynarodowo podzakresy WSPR i WSPR-15. Standardowo używana jest emisja WSPR o cyklu 2 minutowym o ile nie podano inaczej.

Pasmo	Zakres częstotliwości	Emisja
Fale długie	137,400 – 137,600 kHz	
	137,600 – 137,625 KHz	WSPR-15
Fale średnie	475,600 – 475,800 kHz	
	475,800 – 475,825 kHz	WSPR-15
160 m	1830,000 – 1830,200 kHz	
	1830,200 – 1830,225 kHz	WSPR-15
80 m	3594,000 – 3594,200 kHz	
60 m	5288,600 – 5288,800 kHz	
40 m	7040,000 – 7040,200 kHz	
30 m	10140,100 – 10140,300 kHz	
20 m	14097,000 – 14097,200 kHz	
17 m	18106,000 – 18106,200 kHz	
15 m	21096,000 – 21096,200 kHz	
12 m	24926,000 – 24926,200 kHz	
10 m	28126,000 – 28126,200 kHz	
6 m	50294,400 – 50294,600 kHz	
4 m	70092,400 – 70092,600 kHz	
2 m	144,490400 – 144,490600 MHz	



Rys. 6.4. Filtr dolnoprzepustowy 7 rzędu

Tabela 6.2. Elementy filtru dolnoprzepustowego 7 rzędu o opornościach wejściowej i wyjściowej 50 Ω dla różnych pasm amatorskich

Pasmo [MHz[	F gr. [MHz]	F -3 dB [MHz]	F -30 dB [MHz]	C1,7 [pF]	C3,5 [pF]	L2,6 [µH]	L4 [µH]
0,137				2,2 n//10 n	4,7 n//10 n		
0,500				2,2 n//2,2 n	10 n		
1,8	2,16	2,76	4,0	820	2200	4,442	5,608
3,5	4,125	5,11	7,3	470	1200	2,434	3,012
5,2				680	1200		
7,0	7,36	9,04	12,9	270	680	1,380	1,698
10,1	10,37	11,62	15,8	270	560	1,090	1,257
14,0	14,40	16,41	22,5	180	390	0,773	0,904
18,068	18,93	22,89	32,3	110	270	0,548	0,668
21,0	21,55	27,62	39,9	82	220	0,444	0,561
24,98	25,24	28,94	39,8	100	220	0,438	0,515
28–30	31,66	40,52	58,5	56	150	0,303	0,382

Pasmo [MHz]	L2, L6 [zw.]	L4 [zw.]	Rdzeń	Przewód nr
0,137	105	105	T50-2 (czerwony)	
0,500	64	70	T50-2	
1,8	30	34	T50-2	30
3,5	25	27	T37-2	28
5,2	23	24	T37-2	
7,0	19	21	T37-6 (żółty)	26
10,1	19	20	T37-6	26
14,0	16	17	T37-6	24
18,068	13	15	T37-6	24
21,0	12	14	T37-6	24
24,98	12	13	T37-6	22
28–30	10	11	T37-6	22

Oznaczenie przewodu wg normy amerykańskiej. Średnica przewodu nie jest krytyczna, uzwojenie musi się tylko zmieścić na rdzeniu. Typy rdzeni wybrane dla mocy nie przekraczających 10 W. Dla fal długich i średnich pojemności podane są w nF i są uzyskiwane przez równoległe łączenie kondensatorów.

Dodatek A
Kod źródłowy „WsprryPi”

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#include <ctype.h>
#include <dirent.h>
#include <math.h>
#include <fcntl.h>
#include <assert.h>
#include <sys/mman.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <signal.h>
#include <malloc.h>
#include <time.h>

#define F_XTAL      (19229581.05021504427657747984 4352)        // calibrated 19.2MHz XTAL frequency
#define F_PLLD_CLK   (26.0 * F_XTAL)                    // 500MHz PLLD reference clock

#define N_ITER  1400  // number of PWM operations per symbol; larger values gives less spurs at the cost of frequency resolution; e.g. use 22500 for HF usage up to 30MHz, 12000 up to 50MHz, 1600 for VHF usage up to 144 Mhz, F_PWM_CLK needs to be adjusted when changing N_ITER
//#define F_PWM_CLK    (31500000.0)   // 31.5MHz PWM clock   use with N_ITER=22500
#define F_PWM_CLK    (33970588.235294117647058823529413)   // 31.5MHz calibrated PWM clock use with N_ITER=1400

#define WSPR_SYMTIME (8192.0/12000.0)  // symbol time

#define POLYNOM_1 0xf2d05351    // polynoms for
#define POLYNOM_2 0xe4613c47    // parity generator

/* RF code: */

#define BCM2708_PERI_BASE       0x20000000
#define GPIO_BASE               (BCM2708_PERI_BASE + 0x200000) /* GPIO controller */
#define PAGE_SIZE (4*1024)
#define BLOCK_SIZE (4*1024)

int  mem_fd;
char *gpio_mem, *gpio_map;
char *spi0_mem, *spi0_map;


// I/O access
volatile unsigned *gpio = NULL;
volatile unsigned *allof7e = NULL;

// GPIO setup macros. Always use INP_GPIO(x) before using OUT_GPIO(x) or SET_GPIO_ALT(x,y)
#define INP_GPIO(g) *(gpio+((g)/10)) &= ~(7<<(((g)%10)*3))
#define OUT_GPIO(g) *(gpio+((g)/10)) |=  (1<<(((g)%10)*3))
#define SET_GPIO_ALT(g,a) *(gpio+(((g)/10))) |= (((a)<=3?(a)+4:(a)==4?3:2)<<(((g)%10)*3))
```

```
#define GPIO_SET *(gpio+7)  // sets   bits which are 1 ignores bits which are 0
#define GPIO_CLR *(gpio+10) // clears bits which are 1 ignores bits which are 0
#define GPIO_GET *(gpio+13) // sets   bits which are 1 ignores bits which are 0

#define ACCESS(base) *(volatile int*)((int)allof7e+base-0x7e000000)
#define SETBIT(base, bit) ACCESS(base) |= 1<<bit
#define CLRBIT(base, bit) ACCESS(base) &= ~(1<<bit)
#define CM_GP0CTL (0x7e101070)
#define GPFSEL0 (0x7E200000)
#define PADS_GPIO_0_27  (0x7e10002c)
#define CM_GP0DIV (0x7e101074)
#define CLKBASE (0x7E101000)
#define DMABASE (0x7E007000)
#define PWMBASE  (0x7e20C000) /* PWM controller */

struct GPCTL {
   char SRC        : 4;
   char ENAB       : 1;
   char KILL       : 1;
   char            : 1;
   char BUSY       : 1;
   char FLIP       : 1;
   char MASH       : 2;
   unsigned int    : 13;
   char PASSWD     : 8;
};

void getRealMemPage(void** vAddr, void** pAddr) {
   void* a = (void*)valloc(4096);

   ((int*)a)[0] = 1;  // use page to force allocation.

   mlock(a, 4096);  // lock into ram.

   *vAddr = a;  // yay - we know the virtual address

   unsigned long long frameinfo;

   int fp = open("/proc/self/pagemap", 'r');
   lseek(fp, ((int)a)/4096*8, SEEK_SET);
   read(fp, &frameinfo, sizeof(frameinfo));

   *pAddr = (void*)((int)(frameinfo*4096));
}

void freeRealMemPage(void* vAddr) {

   munlock(vAddr, 4096);  // unlock ram.

   free(vAddr);
}

struct CB {
   volatile unsigned int TI;
```

```
    volatile unsigned int SOURCE_AD;
    volatile unsigned int DEST_AD;
    volatile unsigned int TXFR_LEN;
    volatile unsigned int STRIDE;
    volatile unsigned int NEXTCONBK;
    volatile unsigned int RES1;
    volatile unsigned int RES2;

};

struct DMAregs {
    volatile unsigned int CS;
    volatile unsigned int CONBLK_AD;
    volatile unsigned int TI;
    volatile unsigned int SOURCE_AD;
    volatile unsigned int DEST_AD;
    volatile unsigned int TXFR_LEN;
    volatile unsigned int STRIDE;
    volatile unsigned int NEXTCONBK;
    volatile unsigned int DEBUG;
};

struct PageInfo {
    void* p;   // physical address
    void* v;   // virtual address
};

struct PageInfo constPage;
struct PageInfo instrPage;
struct PageInfo instrs[1024];

double fracs[1024];

void txon()
{
    if(allof7e == NULL){
      allof7e = (unsigned *)mmap(
                NULL,
                0x01000000,  //len
                PROT_READ|PROT_WRITE,
                MAP_SHARED,
                mem_fd,
                0x20000000  //base
            );
      if ((int)allof7e==-1) exit(-1);
    }

    SETBIT(GPFSEL0 , 14);
    CLRBIT(GPFSEL0 , 13);
    CLRBIT(GPFSEL0 , 12);

    // Set GPIO drive strength, more info: http://www.scribd.com/doc/101830961/GPIO-Pads-Control2
    //ACCESS(PADS_GPIO_0_27) = 0x5a000018 + 0;  //2mA -3.4dBm
    //ACCESS(PADS_GPIO_0_27) = 0x5a000018 + 1;  //4mA +2.1dBm
    //ACCESS(PADS_GPIO_0_27) = 0x5a000018 + 2;  //6mA +4.9dBm
```

```
   //ACCESS(PADS_GPIO_0_27) = 0x5a000018 + 3;  //8mA +6.6dBm(default)
   //ACCESS(PADS_GPIO_0_27) = 0x5a000018 + 4;  //10mA +8.2dBm
   //ACCESS(PADS_GPIO_0_27) = 0x5a000018 + 5;  //12mA +9.2dBm
   //ACCESS(PADS_GPIO_0_27) = 0x5a000018 + 6;  //14mA +10.0dBm
   ACCESS(PADS_GPIO_0_27) = 0x5a000018 + 7;  //16mA +10.6dBm

   struct GPCTL setupword = {6/*SRC*/, 1, 0, 0, 0, 1,0x5a};
   ACCESS(CM_GP0CTL) = *((int*)&setupword);
}

void txoff()
{
   struct GPCTL setupword = {6/*SRC*/, 0, 0, 0, 0, 1,0x5a};
   ACCESS(CM_GP0CTL) = *((int*)&setupword);
}

void setfreq(long freq)
{
   ACCESS(CM_GP0DIV) = (0x5a << 24) + freq;
}

void txSym(int sym, double tsym)
{
   int bufPtr=0;
   int clocksPerIter = (int)((F_PWM_CLK/((double)N_ITER)) * tsym);
   //printf("tsym=%f iter=%u clocksPerIter=%u tsymerr=%f\n", tsym, N_ITER, clocksPerIter, tsym -
((float)clocksPerIter*(float)N_ITER)/F_PWM_CLK );
   int i = sym*3 + 511;
   double dval = -1.0 * fracs[i] - 0.5; // ratio between -0.5 and 0.5 of frequency position that is in
between two fractional clock divider bins (frequency goes up for dval from -0.5 to 0.5)
   int k = (int)(round(dval));  // integer component
   double frac = (dval - (double)k)/2 + 0.5;
   unsigned int fracval = (frac*clocksPerIter);
   //printf("i=%d *i=%u %u fracval=%u dval=%f sym=%d\n", i, ((int*)(constPage.v))[i-1],
((int*)(constPage.v))[i+1], fracval, dval, sym);
   int j;
   for(j=0; j!=N_ITER; j++){
      bufPtr++;
      while( ACCESS(DMABASE + 0x04 /* CurBlock*/) ==  (int)(instrs[bufPtr].p)) usleep(100);
      ((struct CB*)(instrs[bufPtr].v))->SOURCE_AD = (int)constPage.p + (i-1)*4;

      bufPtr++;
      while( ACCESS(DMABASE + 0x04 /* CurBlock*/) ==  (int)(instrs[bufPtr].p)) usleep(100);
      ((struct CB*)(instrs[bufPtr].v))->TXFR_LEN = clocksPerIter-fracval;

      bufPtr++;
      while( ACCESS(DMABASE + 0x04 /* CurBlock*/) ==  (int)(instrs[bufPtr].p)) usleep(100);
      ((struct CB*)(instrs[bufPtr].v))->SOURCE_AD = (int)constPage.p + (i+1)*4;

      bufPtr=(bufPtr+1) % (1024);
      while( ACCESS(DMABASE + 0x04 /* CurBlock*/) ==  (int)(instrs[bufPtr].p)) usleep(100);
      ((struct CB*)(instrs[bufPtr].v))->TXFR_LEN = fracval;
   }
}
```

```
void unSetupDMA(){
    printf("exiting\n");
    struct DMAregs* DMA0 = (struct DMAregs*)&(ACCESS(DMABASE));
    DMA0->CS =1<<31;  // reset dma controller
    txoff();
}

void handSig() {
  exit(0);
}
void setupDMATab( float centerFreq, double symOffset, double tsym, int nsym ){
   // make data page contents - it's essientially 1024 different commands for the
   // DMA controller to send to the clock module at the correct time.
  int i;
  for(i=1; i<1023; i+=3){
     double freq = centerFreq + ((double)(-511 + i))*symOffset/3.0;
     double divisor = F_PLLD_CLK/freq;
     unsigned long integer_part = (unsigned long) divisor;
     unsigned long fractional_part = (divisor - integer_part) * (1 << 12);
     unsigned long tuning_word = (0x5a << 24) + integer_part * (1 << 12) + fractional_part;
     if(fractional_part == 0 || fractional_part == 1023){
       if((-511 + i) >= 0 && (-511 + i) <= (nsym * 3))
         printf("warning: symbol %u unusable because fractional divider is out of range, try near
frequency.\n", i/3);
     }
     ((int*)(constPage.v))[i-1] = tuning_word - 1;
     ((int*)(constPage.v))[i] = tuning_word;
     ((int*)(constPage.v))[i+1] = tuning_word + 1;
     double actual_freq = F_PLLD_CLK/((double)integer_part +
(double)fractional_part/(double)(1<<12));
     double freq_corr = freq - actual_freq;
     double delta = F_PLLD_CLK/((double)integer_part + (double)fractional_part/(double)(1<<12)) -
F_PLLD_CLK/((double)integer_part + ((double)fractional_part+1.0)/(double)(1<<12));
     int clocksPerIter = (int)((F_PWM_CLK/((double)N_ITER)) * tsym);
     double resolution = 2.0 * delta / ((double)clocksPerIter);
     if(resolution > symOffset ){
       printf("warning: PWM/PLL fractional divider has not enough resolution: %fHz while %fHz is
required, try lower frequency or decrease N_ITER in code to achieve more resolution.\n", resolution,
symOffset);
       exit(0);
     }
     fracs[i] = freq_corr/delta;
     //printf("i=%u f=%f fa=%f corr=%f delta=%f percfrac=%f int=%u frac=%u tuning_word=%u
resolution=%fmHz\n", i, freq, actual_freq, freq_corr, delta, fracs[i], integer_part, fractional_part,
tuning_word, resolution *1000);
   }
}

void setupDMA(){
   atexit(unSetupDMA);
   signal (SIGINT, handSig);
   signal (SIGTERM, handSig);
   signal (SIGHUP, handSig);
   signal (SIGQUIT, handSig);
```

```
  // allocate a few pages of ram
  getRealMemPage(&constPage.v, &constPage.p);

  int instrCnt = 0;

  while (instrCnt<1024) {
   getRealMemPage(&instrPage.v, &instrPage.p);

   // make copy instructions
   struct CB* instr0= (struct CB*)instrPage.v;
   int i;
   for (i=0; i<4096/sizeof(struct CB); i++) {
    instrs[instrCnt].v = (void*)((int)instrPage.v + sizeof(struct CB)*i);
    instrs[instrCnt].p = (void*)((int)instrPage.p + sizeof(struct CB)*i);
    instr0->SOURCE_AD = (unsigned int)constPage.p+2048;
    instr0->DEST_AD = PWMBASE+0x18 /* FIF1 */;
    instr0->TXFR_LEN = 4;
    instr0->STRIDE = 0;
    //instr0->NEXTCONBK = (int)instrPage.p + sizeof(struct CB)*(i+1);
    instr0->TI = (1/* DREQ  */<<6) | (5 /* PWM */<<16) |  (1<<26/* no wide*/) ;
    instr0->RES1 = 0;
    instr0->RES2 = 0;

    if (i%2) {
     instr0->DEST_AD = CM_GP0DIV;
     instr0->STRIDE = 4;
     instr0->TI = (1<<26/* no wide*/) ;
    }

    if (instrCnt!=0) ((struct CB*)(instrs[instrCnt-1].v))->NEXTCONBK = (int)instrs[instrCnt].p;
    instr0++;
    instrCnt++;
   }
  }
  ((struct CB*)(instrs[1023].v))->NEXTCONBK = (int)instrs[0].p;

  // set up a clock for the PWM
  ACCESS(CLKBASE + 40*4 /*PWMCLK_CNTL*/) = 0x5A000026;  // Source=PLLD and disable
  usleep(1000);
//   ACCESS(CLKBASE + 41*4 /*PWMCLK_DIV*/)  = 0x5A002800;
  ACCESS(CLKBASE + 41*4 /*PWMCLK_DIV*/)  = 0x5A002000;  // set PWM div to 2, for
250MHz
  ACCESS(CLKBASE + 40*4 /*PWMCLK_CNTL*/) = 0x5A000016;  // Source=PLLD and enable
  usleep(1000);

  // set up pwm
  ACCESS(PWMBASE + 0x0 /* CTRL*/) = 0;
  usleep(1000);
  ACCESS(PWMBASE + 0x4 /* status*/) = -1;  // clear errors
  usleep(1000);
  ACCESS(PWMBASE + 0x0 /* CTRL*/) = -1; //(1<<13 /* Use fifo */) | (1<<10 /* repeat */) | (1<<9
/* serializer */) | (1<<8 /* enable ch */) ;
  usleep(1000);
  ACCESS(PWMBASE + 0x8 /* DMAC*/) = (1<<31 /* DMA enable */) | 0x0707;
```

```
  //activate dma
  struct DMAregs* DMA0 = (struct DMAregs*)&(ACCESS(DMABASE));
  DMA0->CS =1<<31;  // reset
  DMA0->CONBLK_AD=0;
  DMA0->TI=0;
  DMA0->CONBLK_AD = (unsigned int)(instrPage.p);
  DMA0->CS =(1<<0)|(255 <<16);  // enable bit = 0, clear end flag = 1, prio=19-16
}


//
// Set up a memory regions to access GPIO
//
void setup_io()
{
   /* open /dev/mem */
   if ((mem_fd = open("/dev/mem", O_RDWR|O_SYNC) ) < 0) {
      printf("can't open /dev/mem \n");
      exit (-1);
   }

   /* mmap GPIO */

   // Allocate MAP block
   if ((gpio_mem = malloc(BLOCK_SIZE + (PAGE_SIZE-1))) == NULL) {
      printf("allocation error \n");
      exit (-1);
   }

   // Make sure pointer is on 4K boundary
   if ((unsigned long)gpio_mem % PAGE_SIZE)
      gpio_mem += PAGE_SIZE - ((unsigned long)gpio_mem % PAGE_SIZE);

   // Now map it
   gpio_map = (unsigned char *)mmap(
               gpio_mem,
               BLOCK_SIZE,
               PROT_READ|PROT_WRITE,
               MAP_SHARED|MAP_FIXED,
               mem_fd,
               GPIO_BASE
            );

   if ((long)gpio_map < 0) {
      printf("mmap error %d\n", (int)gpio_map);
      exit (-1);
   }

   // Always use volatile pointer!
   gpio = (volatile unsigned *)gpio_map;
}

void setup_gpios()
{
  int g;
```

```
  // Switch GPIO 7..11 to output mode

   /************************************************************************\
    * You are about to change the GPIO settings of your computer.          *
    * Mess this up and it will stop working!                               *
    * It might be a good idea to 'sync' before running this program        *
    * so at least you still have your code changes written to the SD-card! *
   \************************************************************************/

   // Set GPIO pins 7-11 to output
   for (g=7; g<=11; g++) {
      INP_GPIO(g); // must use INP_GPIO before we can use OUT_GPIO
      //OUT_GPIO(g);
   }
}

void strupr(char *str)
{   while(*str)
   {
      *str = toupper(*str);
      str++;
   }
}

void wspr(char* call, char* l, char* dbm, unsigned char* symbols)
{
  // pack prefix in nadd, call in n1, grid, dbm in n2
  char* c, buf[16];
  strncpy(buf, call, 16);
  c=buf;
  strupr(c);
  unsigned long ng,nadd=0;

  if(strchr(c, '/')){ //prefix-suffix
    nadd=2;
    int i=strchr(c, '/')-c; //stroke position
    int n=strlen(c)-i-1; //suffix len, prefix-call len
    c[i]='\0';
    if(n==1) ng=60000-32768+(c[i+1]>='0'&&c[i+1]<='9'?c[i+1]-'0':c[i+1]==' '?38:c[i+1]-'A'+10); //
suffix /A to /Z, /0 to /9
    if(n==2) ng=60000+26+10*(c[i+1]-'0')+(c[i+2]-'0'); // suffix /10 to /99
    if(n>2){ // prefix EA8/, right align
      ng=(i<3?36:c[i-3]>='0'&&c[i-3]<='9'?c[i-3]-'0':c[i-3]-'A'+10);
      ng=37*ng+(i<2?36:c[i-2]>='0'&&c[i-2]<='9'?c[i-2]-'0':c[i-2]-'A'+10);
      ng=37*ng+(i<1?36:c[i-1]>='0'&&c[i-1]<='9'?c[i-1]-'0':c[i-1]-'A'+10);
      if(ng<32768) nadd=1; else ng=ng-32768;
      c=c+i+1;
    }
  }

  int i=(isdigit(c[2])?2:isdigit(c[1])?1:0); //last prefix digit of de-suffixed/de-prefixed callsign
  int n=strlen(c)-i-1; //2nd part of call len
  unsigned long n1;
  n1=(i<2?36:c[i-2]>='0'&&c[i-2]<='9'?c[i-2]-'0':c[i-2]-'A'+10);
  n1=36*n1+(i<1?36:c[i-1]>='0'&&c[i-1]<='9'?c[i-1]-'0':c[i-1]-'A'+10);
```

```
  n1=10*n1+c[i]-'0';
  n1=27*n1+(n<1?26:c[i+1]-'A');
  n1=27*n1+(n<2?26:c[i+2]-'A');
  n1=27*n1+(n<3?26:c[i+3]-'A');

  //if(rand() % 2) nadd=0;
  if(!nadd){
    strupr(l); //grid square Maidenhead locator (uppercase)
    ng=180*(179-10*(l[0]-'A')-(l[2]-'0'))+10*(l[1]-'A')+(l[3]-'0');
  }
  int p = atoi(dbm);    //EIRP in dBm={0,3,7,10,13,17,20,23,27,30,33,37,40,43,47,50,53,57,60}
  int corr[]={0,-1,1,0,-1,2,1,0,-1,1};
  p=p>60?60:p<0?0:p+corr[p%10];
  unsigned long n2=(ng<<7)|(p+64+nadd);

  // pack n1,n2,zero-tail into 50 bits
  char packed[11] = {n1>>20, n1>>12, n1>>4, ((n1&0x0f)<<4)|((n2>>18)&0x0f),
n2>>10, n2>>2, (n2&0x03)<<6, 0, 0, 0, 0};

  // convolutional encoding K=32, r=1/2, Layland-Lushbaugh polynomials
  int k = 0;
  int j,s;
  int nstate = 0;
  unsigned char symbol[176];
  for(j=0;j!=sizeof(packed);j++){
    for(i=7;i>=0;i--){
      unsigned long poly[2] = { 0xf2d05351L, 0xe4613c47L };
      nstate = (nstate<<1) | ((packed[j]>>i)&1);
      for(s=0;s!=2;s++){   //convolve
        unsigned long n = nstate & poly[s];
        int even = 0;  // even := parity(n)
        while(n){
          even = 1 - even;
          n = n & (n - 1);
        }
        symbol[k] = even;
        k++;
      }
    }
  }

  // interleave symbols
  const unsigned char npr3[162] = {
    1,1,0,0,0,0,0,0,1,0,0,0,1,1,1,0,0,0,1,0,0,1,0,1,1,1,1,0,0,0,0,0,
    0,0,1,0,0,1,0,1,0,0,0,0,0,0,1,0,1,1,0,0,1,1,0,1,0,0,0,1,1,0,1,0,
    0,0,0,1,1,0,1,0,1,0,1,0,1,0,0,1,0,0,1,0,1,1,0,0,0,1,1,0,1,0,1,0,
    0,0,1,0,0,0,0,0,1,0,0,1,0,0,1,1,1,0,1,1,0,0,1,1,0,1,0,0,0,1,1,1,
    0,0,0,0,0,1,0,1,0,0,1,1,0,0,0,0,0,0,0,0,1,1,0,1,0,1,1,0,0,0,1,1,0,
    0,0 };
  for(i=0;i!=162;i++){
    // j0 := bit reversed_values_smaller_than_161[i]
    unsigned char j0;
    p=-1;
    for(k=0;p!=i;k++){
      for(j=0;j!=8;j++)   // j0:=bit_reverse(k)
```

```
        j0 = ((k>>j)&1)|(j0<<1);
       if(j0<162)
        p++;
    }
    symbols[j0]=npr3[j0]|symbol[i]<<1; //interleave and add sync vector
  }
}

void wait_every(int minute)
{
 time_t t;
 struct tm* ptm;
 for(;;){
  time(&t);
  ptm = gmtime(&t);
  if((ptm->tm_min % minute) == 0 && ptm->tm_sec == 0) break;
  usleep(1000);
 }
 usleep(1000000); // wait another second
}

int main(int argc, char *argv[])
{
 unsigned char symbols[162];
 int i;
 double centre_freq;
 int wspr15;
 double wspr_symtime;
 int nbands = argc - 4;
 int band = 0;

 if(argc < 5){
  printf("Usage: wspr <[prefix/]callsign[/A-Z,/0-9,/00-99]> <locator> <power in dBm> [<frequency in
Hz or 0 for interval> ...]\n");
  printf("\te.g.: sudo ./wspr K1JT/P JO21 10 7040074 0 0 10140174 0 0\n");
  printf("\tchoose freq in range +/-100 Hz around one of center frequencies: 137500, 475700, 1838100,
3594100, 5288700, 7040100, 10140200, 14097100, 18106100, 21096100, 24926100, 28126100,
50294500, 70092500, 144490500 Hz (WSPR-2), or in range +/-12 Hz around 137612, 475812,
1838212 Hz (WSPR-15).\n");
  return 1;
 }
 // argv[1]=callsign, argv[2]=locator, argv[3]=power(dBm)
 wspr(argv[1], argv[2], argv[3], symbols);
 printf("Symbols: ");
 for (i = 0; i < sizeof(symbols)/sizeof(*symbols); i++)
  printf("%d,", symbols[i]);
 printf("\n");
 setup_io();
 setup_gpios();
 txon();
 setupDMA();
 printf("Ready for transmit...\n");

 for(;;)
 {
```

```
    txoff();
    centre_freq = atof(argv[band + 4]);
    wspr15 = (centre_freq > 137600 && centre_freq < 137625) || \
             (centre_freq > 475800 && centre_freq < 475825) || \
             (centre_freq > 1838200 && centre_freq < 1838225);
    wspr_symtime = (wspr15) ? 8.0 * WSPR_SYMTIME : WSPR_SYMTIME;
    band++;
    if(band >= nbands)
      band = 0;
    if(centre_freq) setupDMATab(centre_freq, 1.0/wspr_symtime, wspr_symtime, 4);
    wait_every((wspr15) ? 15 : 2);
    time_t t;
    time(&t);
    char buf[256];
    strcpy(buf,ctime(&t));
    buf[strlen(buf)-1]='\0';
    printf("%s - %s@%f\n", buf, (wspr15)?"wspr-15":"wspr-2", centre_freq);
    if(centre_freq){
      txon();
      for (i = 0; i < 162; i++) {
        txSym(symbols[i], wspr_symtime);
        //txSym(atoi(argv[5]), wspr_symtime);
      }
    }
  }
  return 0;
}
```

Dodatek B
Generator sygnałowy

Program pozwala na użycie „Raspberry” jako generatora sygnałowego do celów laboratoryjnych lub podobnych. Przytoczony poniżej kod źródłowy daje się łatwo modyfikwać np. w celu uzyskania wobulatora itp. Generator pracuje w szerokim zakresie częstotliwości obejmującym fale krótkie i UKF.
W połączeniu z odbiornikami szerokozakresowymi typu FDC albo DVB-T (RTL) można dzięki niemu łatwo uruchomić prosty analizator widma.
Dla uzyskania sygnałów o większej czystości widmowej można zamiast niego użyć syntezera cyfrowego opartego na AD9850, Si570 itp.
Przy założeniu, że podany poniżej kod jest zapisany w pliku o nazwie *freq_pi.c* do jego skompilowania służy polecenie:
gcc -Wall -O4 -o freq_pi freq_pi.c -std=gnu99 -lm
do zainstalowania:
cp freq_pi /usr/local/bin/
a do wywołania
./freq_pi albo tylko *freq_pi* (po zainstalowaniu w katalogu /usr/local/bin).

Kod źródłowy

```
/*
       This file is part of freq_pi Copyright (c) Jan Panteltje 2013-always
 email: panteltje@yahoo.com

This code contains parts of code from Pifm.c

Start GPL license:
    This program is free software; you can redistribute it and/or modify
    it under the terms of the GNU General Public License as published by
    the Free Software Foundation; either version 2 of the License, or
    (at your option) any later version.

    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU General Public License for more details.

    You should have received a copy of the GNU General Public License
    along with this program; if not, write to the Free Software
    Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.

If you take this code and port it to that Redmond crap you STILL must
release SOURCE code,
this program has secret beyond bit level encrypted bits that make it
traceable and you will be bitten by the FSF if you violate these terms.
*/

/* set TABs to 4 for correct formatting of this file, or if you do not know
what that is replace all TABs with 4 spaces, else you cannot READ this */

/*
Program name:
fre_pi

Function:
programmable frequency generator on GPIO_4 pin 7
```

```
To compile:
gcc -Wall -O4 -o test test.c -std=gnu99 -lm


To install:
cp freq_pi /usr/local/bin/

To run:
freq_pi
*/

#define PROGRAM_VERSION "0.1"


#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <dirent.h>
#include <math.h>
#include <fcntl.h>
#include <assert.h>
#include <malloc.h>
#include <sys/mman.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <signal.h>
#include <unistd.h>
#include <getopt.h>
#include <stdint.h>
#include <time.h>
#include <getopt.h>
#include <ctype.h>
#include <errno.h>
#include <fcntl.h>


#define PAGE_SIZE (4*1024)
#define BLOCK_SIZE (4*1024)

int  mem_fd;
char *gpio_mem, *gpio_map;
char *spi0_mem, *spi0_map;


// I/O access
volatile unsigned *gpio;
volatile unsigned *allof7e;

// GPIO setup macros. Always use INP_GPIO(x) before using OUT_GPIO(x) or
SET_GPIO_ALT(x,y)
#define INP_GPIO(g) *(gpio+((g)/10)) &= ~(7<<(((g)%10)*3))
#define OUT_GPIO(g) *(gpio+((g)/10)) |=  (1<<(((g)%10)*3))
#define SET_GPIO_ALT(g,a) *(gpio+(((g)/10))) |=
(((a)<=3?(a)+4:(a)==4?3:2)<<(((g)%10)*3))

#define GPIO_SET *(gpio+7) // sets bits which are 1 ignores bits which are 0
#define GPIO_CLR *(gpio+10)// clears bits which are 1 ignores bits which are
0
#define GPIO_GET *(gpio+13)// sets bits which are 1 ignores bits which are 0

#define ACCESS(base) *(volatile int*)((int)allof7e+base-0x7e000000)
```

```
#define SETBIT(base, bit) ACCESS(base) |= 1<<bit
#define CLRBIT(base, bit) ACCESS(base) &= ~(1<<bit)

#define CM_GP0CTL (0x7e101070)
#define GPFSEL0 (0x7E200000)
#define CM_GP0DIV (0x7e101074)
#define CLKBASE (0x7E101000)
#define DMABASE (0x7E007000)
#define PWMBASE (0x7e20C000) /* PWM controller */


struct GPCTL
{
char SRC         : 4;
char ENAB        : 1;
char KILL        : 1;
char             : 1;
char BUSY        : 1;
char FLIP        : 1;
char MASH        : 2;
unsigned int     : 13;
char PASSWD      : 8;
};


void getRealMemPage(void** vAddr, void** pAddr)
{
void* a = valloc(4096);

((int*)a)[0] = 1;  // use page to force allocation.

mlock(a, 4096);  // lock into ram.

*vAddr = a;  // yay - we know the virtual address

unsigned long long frameinfo;

int fp = open("/proc/self/pagemap", 'r');
lseek(fp, ((int)a)/4096*8, SEEK_SET);
read(fp, &frameinfo, sizeof(frameinfo));

*pAddr = (void*)((int)(frameinfo*4096));
}


void freeRealMemPage(void* vAddr)
{
munlock(vAddr, 4096);  // unlock ram.

free(vAddr);
}


void start_rf_output(int source)
{
/* open /dev/mem */
if ((mem_fd = open("/dev/mem", O_RDWR|O_SYNC)) < 0)
    {
    printf("can't open /dev/mem \n");
    exit (-1);
    }
```

```
allof7e = (unsigned *)mmap(NULL, 0x01000000,  /*len */ PROT_READ|PROT_WRITE,
MAP_SHARED, mem_fd, 0x20000000  /* base */);

if ((int)allof7e==-1) exit(-1);

SETBIT(GPFSEL0 , 14);
CLRBIT(GPFSEL0 , 13);
CLRBIT(GPFSEL0 , 12);

/*
Clock source
0 = GND
1 = oscillator
2 = testdebug0
3 = testdebug1
4 = PLLA per
5 = PLLC per
6 = PLLD per
7 = HDMI auxiliary
8-15 = GND
*/


struct GPCTL setupword = {source/*SRC*/, 1, 0, 0, 0, 1,0x5a};

ACCESS(CM_GP0CTL) = *((int*)&setupword);
}


void modulate(int m)
{
ACCESS(CM_GP0DIV) = (0x5a << 24) + 0x4d72 + m;
}


struct CB
{
 volatile unsigned int TI;
 volatile unsigned int SOURCE_AD;
 volatile unsigned int DEST_AD;
 volatile unsigned int TXFR_LEN;
 volatile unsigned int STRIDE;
 volatile unsigned int NEXTCONBK;
 volatile unsigned int RES1;
 volatile unsigned int RES2;
};


struct DMAregs
{
 volatile unsigned int CS;
 volatile unsigned int CONBLK_AD;
 volatile unsigned int TI;
 volatile unsigned int SOURCE_AD;
 volatile unsigned int DEST_AD;
 volatile unsigned int TXFR_LEN;
 volatile unsigned int STRIDE;
 volatile unsigned int NEXTCONBK;
 volatile unsigned int DEBUG;
};
```

```
struct PageInfo
{
 void* p;  // physical address
 void* v;   // virtual address
};


struct PageInfo constPage;
struct PageInfo instrPage;
struct PageInfo instrs[1024];


void print_usage()
{
fprintf(stderr,\
"\nPanteltje freq_pi-%s\n\
Usage:\nfreq_pic -f frequency [-h]\n\
-f int        frequency to ouput on GPIO_4 pin 7.\n\
-h            help (this help).\n\
\n",\
PROGRAM_VERSION);
fprintf(stderr,\
"Example for 1 MHz:\n\
freq_pi -f 100000000\n\
\n"\
);

} /* end function print_usage */


int main(int argc, char **argv)
{
int a;
uint32_t frequency = 0; // -Wall
uint32_t ua;

/* defaults */


/* end defaults */


/* proces any command line arguments */
while(1)
    {
    a = getopt(argc, argv, "f:h");
    if(a == -1) break;

    switch(a)
        {
        case 'f':    // frequency
            a = atoi(optarg);


            frequency = a;
            break;
        case 'h': // help
            print_usage();
            exit(1);
```

```
            break;
            break;
        case -1:
            break;
        case '?':
            if (isprint(optopt))
                 {
                 fprintf(stderr, "send_iq: unknown option `-%c'.\n",
optopt);
                 }
            else
                {
                fprintf(stderr, "send_iq: unknown option character
`\\x%x'.\n", optopt);
                }
            print_usage();

            exit(1);
            break;
        default:
            print_usage();

            exit(1);
            break;
        }/* end switch a */
    }/* end while getopt() */


/* set frequeny */

/*
From ducument BCM2835-ARM-Peripherals.pdf
 page 105
   6.3 General Purpose GPIO Clocks


The General Purpose clocks can be output to GPIO pins.
They run from the peripherals clock sources and use clock generators with
noise-shaping MASH dividers.
These allow the GPIO clocks to be used to drive audio devices.
The fractional divider operates by periodically dropping source clock
pulses, therefore the output frequency will periodically switch between:
source_frequency / DIVI, and   source_frequency / (DIVI + 1)

Jitter is therefore reduced by increasing the source clock frequency.
In applications where jitter is a concern, the fastest available clock
source should be used.
The General Purpose clocks have MASH noise-shaping dividers which push this
fractional divider jitter out of the audio band.
MASH noise-shaping is incorporated to push the fractional divider jitter out
of the audio band if required.
The MASH can be programmed for 1, 2 or 3-stage filtering. MASH filter, the
frequency is spread around the requested frequency and the user must ensure
that the module is not exposed to frequencies higher than 25MHz.
Also, the MASH filter imposes a low limit on the range of DIVI.

MASH     min DIVI min output freq   average output freq      max output freq
0 (int divide) 1  source / (DIVI) source / (DIVI)      source / (DIVI)
1              2  source / (DIVI) source / (DIVI + DIVF / 1024) source /
(DIVI + 1)
```

2 3 source / (DIVI - 1) source / (DIVI + DIVF / 1024) source / (DIVI + 2)
3 5 source / (DIVI - 3) source / (DIVI + DIVF / 1024) source / (DIVI + 4)
Table 6-32 Effect of MASH Filter on Frequency

The following example illustrates the spreading of output clock frequency resulting from the
use of the MASH filter. Note that the spread is greater for lower divisors.

PLL freq (MHz)	target freq (MHz)	MASH	divisor	DIVI	DIVF	min freq (MHz)	ave freq (MHz)	max freq (MHz)	error
650	18.32	0	35.480	35	492	18.57	18.57	18.57	ok
650	18.32	1	35.480	35	492	18.06	18.32	18.57	ok
650	18.32	2	35.480	35	492	17.57	18.32	19.12	ok
650	18.32	3	35.480	35	492	16.67	18.32	20.31	ok
400	18.32	0	21.834	21	854	19.05	19.05	19.05	ok
400	18.32	1	21.834	21	854	18.18	18.32	19.05	ok
400	18.32	2	21.834	21	854	17.39	18.32	20.00	ok
400	18.32	3	21.834	21	854	16.00	18.32	22.22	ok
200	18.32	0	10.917	10	939	20.00	20.00	20.00	ok
200	18.32	1	10.917	10	939	18.18	18.32	20.00	ok
200	18.32	2	10.917	10	939	16.67	18.32	22.22	ok
200	18.32	3	10.917	10	939	14.29	18.32	28.57	error

Table 6-33 Example of Frequency Spread when using MASH Filtering

Operating Frequency
The maximum operating frequency of the General Purpose clocks is ~125 MHz at 1.2V but this will be reduced if the GPIO pins are heavily loaded or have a capacitive load.

Register Definitions
Clock Manager General Purpose Clocks Control (CM_GP0CTL, GP1CTL & GP2CTL)

Address 0x 7e10 1070 CM_GP0CTL
0x 7e10 1078 CM_GP1CTL
0x 7e10 1080 CM_GP2CTL

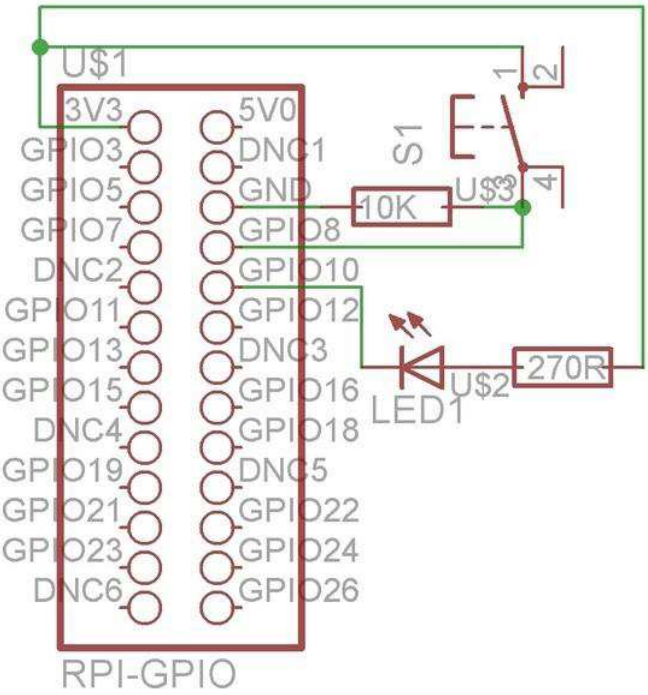
Bit Number	Field Name	Description	Read/ Write	Reset
31-24	PASSWD	Clock Manager password "5a"	W	0
23-11	-	Unused	R	0
10-9	MASH	MASH control 0 = integer division 1 = 1-stage MASH (equivalent to non-MASH dividers) 2 = 2-stage MASH 3 = 3-stage MASH To avoid lock-ups and glitches do not change this control while BUSY=1 and do not change this control at the same time as asserting ENAB.	R/W	0
8	FLIP	Invert the clock generator output This is intended for use in test/debug only. Switching this control will generate an edge on the clock generator output. To avoid output glitches do not switch this control while BUSY=1.	R/W	0
7	BUSY	Clock generator is running Indicates the clock generator is running. To avoid glitches and lock-ups, clock sources and setups must not be changed while this flag is set.	R	0

```
6         -      Unused                                                R     0
5       KILL     Kill the clock generator                             R/W    0
                 0 = no action
                 1 = stop and reset the clock generator
                 This is intended for test/debug only. Using this control
                 may cause a glitch on the clock generator output.
4       ENAB     Enable the clock generator                           R/W    0
                 This requests the clock to start or stop without
                 glitches. The output clock will not stop immediately
                 because the cycle must be allowed to complete to
                 avoid glitches. The BUSY flag will go low when the
                 final cycle is completed.
3-0       SRC    Clock source                                         R/W    0
                 0 = GND
                 1 = oscillator
                 2 = testdebug0
                 3 = testdebug1
                 4 = PLLA per
                 5 = PLLC per
                 6 = PLLD per
                 7 = HDMI auxiliary
                 8-15 = GND
                 To avoid lock-ups and glitches do not change this
                 control while BUSY=1 and do not change this control
                 at the same time as asserting ENAB.
06 February 2012 Broadcom Europe Ltd. 406 Science Park Milton Road Cambridge
CB4 0WW              Page 107
                     2012 Broadcom Corporation. All rights reserved


 Clock Manager General Purpose Clock Divisors (CM_GP0DIV, CM_GP1DIV &
                                          CM_GP2DIV)
Address       0x 7e10 1074 CM_GP0DIV
              0x 7e10 107c CM_GP1DIV
              0x 7e10 1084 CM_GP2DIV

Bit      Field                                                  Read/
                                Description                          Reset
Number    Name                                                  Write
31-24   PASSWD    Clock Manager password "5a"                      W     0
23-12     DIVI    Integer part of divisor                         R/W    0
                  This value has a minimum limit determined by the
                  MASH setting. See text for details. To avoid lock-ups
                  and glitches do not change this control while BUSY=1.
11-0      DIVF    Fractional part of divisor                      R/W    0
                  To avoid lock-ups and glitches do not change this
                  control while BUSY=1.
                  Table 6-35 General Purpose Clock Divisors
*/



//struct GPCTL setupword = {6 /* clock source */, 1 /* enable */,  0 /* not
kill */, 0, 0, 1 /* 1 stage MASH (equivalent to no MASH */, 0x5a /* password
*/};

//ACCESS(CM_GP0CTL) = *((int*)&setupword);


//ACCESS(CM_GP0DIV) = (0x5a << 24) + 0x4d72 + m
```

```
uint16_t divi; // integer part divider [23:12]      12 bits wide, max 4095
uint16_t divf; // fractional part divider [11:0]    12 bits wide, max 4095


/*
clock sources are 650 MHz, 400 MHz, and 200 MHz

So the lowest frequency we can make is 200,000,000 / 4095.4095 =
48,835.165323516 Hz

But I get 61,043 Hz

4095.4095 * 61043 = 249,996,082.108499999 Hz....

But for MASH 1,
 MASH     min DIVI min output freq   average output freq     max output freq
 0 (int divide) 1  source / (DIVI) source / (DIVI)     source / ( DIVI )
* 1             2  source / (DIVI) source / (DIVI + DIVF / 1024) source /
(DIVI + 1)
 2              3  source / (DIVI - 1) source / (DIVI + DIVF / 1024) source
/ (DIVI + 2)
 3              5  source / (DIVI - 3) source / (DIVI + DIVF / 1024) source
/ (DIVI + 4)

200,000,000 / (4095 = 48840.048840048
200,000,000 / (4095 + (4095/1024)) = 48792.400011912


So
61043  * 4099

61043 * (4095 + .3999023437) = 249995496.238766479 Hz, looks like we have a
250 MHz clock.

Lowest frequency then is 61,043 Hz,
highest frequency then is 250,000,000 / 1 = 250,000,000 Hz
*/

//#define PLL0_FREQUENCY 250000000.0
#define PLL0_FREQUENCY 500000000.0

//if(frequency

int clock_source;
/*
Clock source
0 = GND
1 = oscillator
2 = testdebug0
3 = testdebug1
4 = PLLA per
5 = PLLC per
6 = PLLD per
7 = HDMI auxiliary
8-15 = GND
*/

/* init hardware */
```

```
clock_source = 6; /* this GPIO_4 pin 7 allows only the 200 MHz clock????? as
source, the other clock GPIO lines are not on a pin in revision 2 board, so
we have to work with 200 MHz, and that seems to be 250 MHz */

start_rf_output(clock_source);

/* calculate divider */
double da;

da = PLL0_FREQUENCY / (double) frequency;

divi = (int) da;
divf = 4096.0 *(da - (double)divi);

//fprintf(stderr, "frequency=%d da=%f divi=%d divf=%d\n", frequency, da,
divi, divf);

if( (divi > 4095.0) || (divi < 1.0))
    {
    fprintf(stderr, "freq_pi: requested frequency out of range,
aborting.\n");

    exit(1);
    }


if(divf > 4095.0)
    {
    fprintf(stderr, "freq_pi: requested frequency out of range,
aborting.\n");

    exit(1);
    }

ua = (0x5a << 24) + (divi << 12) + divf;

ACCESS(CM_GP0DIV) = ua;


//fprintf(stderr, "WAS here\n");

exit(0);
} /* end function  main */
```

Dodatek C
Programowanie GPIO



Wykorzystanie wejść i wyjść logicznych GPIO we własnych progamach ilustruje przykład programu w języku Python. Naciśnięcie przycisku S1 powoduje zaświecenie się diody, jego puszczenie – zgaśnięcie.

```
# example1.py
# Import potrzebnego modułu.
import RPi.GPIO as GPIO
# Wybór sposobu numeracji nóżek.
GPIO.setmode(GPIO.BOARD)
# GPIO nóżka 10 – wyjście.
GPIO.setup(10, GPIO.OUT)
# GPIO nóżka 8 – wejście.
GPIO.setup(8, GPIO.IN)
# Inicjalizacja GPIO10 – poziom wysoki (true) tak, że dioda jest zgaszona
GPIO.output(10, True)
while 1:
   if GPIO.input(8):
      GPIO.output( 10, False)
   else:
      # Zgaszenie diody świecącej gdy przycisk nie naciśnięty.
      GPIO.output( 10, True)
```

Literatura i adresy internetowe

[1] https://packages.debian.org/stable/hamradio – bogaty zbiór programów krótkofalarskich dla „Wheezy”
[2] www.raspberrypi.org – informacje ogólne, sprzęt, programy i systemy operacyjne
[3] www.raspbian.org – system operacyjny i jego dokumentacja
[4] www.sklep.avt.pl – literatura, moduły rozszerzeń
[5] www.conrad.pl – mikrokomputer, obudowy, modem WiFi, Raspbian na SD, moduły rozszerzeń i akcesoria
[6] „Raspberry Pi. Przewodnik użytkownika”, Gareth Halfacree, Eben Upton, tłum. Mikołaj Szczepaniak, wyd. Helion, ISBN 978-83-246-7313-1, 9788324673131, kod handlowy sklepu AVT: KS-130902
[7] „Moduły rozszerzeń dla RaspberryPi (1). Płytka stykowa, moduł I/O, moduł wejść analogowych”, „Elektronika praktyczna” 6/2013, str. 20
[8] „Moduły rozszerzeń dla Raspberry Pi (2). Płytka do komunikacji szeregowej RaspberryPi_COM”, „Elektronika praktyczna” 7/2013, str. 42
[9] „Moduły rozszerzeń dla Raspberry Pi” i „RasbPI GetDuino Board. Połączenie Arduino oraz Raspberry Pi”, „Elektronika Praktyczna” 1/2014.
[10] http://omattos.com/pifm.tar.gz – program nadajnika radiowego na Raspberry Pi.
[11] http://www.icrobotics.co.uk/wiki/index.php/Turning_the_Raspberry_Pi_Into_an_FM_Transmitter – krótka instrukcja do programu pifm
[12] https://github.com/threeme3/WsprryPi – program „WsprryPi”, jego kod źródłowy w języku C i instrukcja jego instalacji
[13] www.wsprnet.org – baza danych i mapy nasłuchów WSPR, informacje towarzyszące, dyskusje
[14] www.radioworld.co.uk – radiostacje DV-ACCESS-2 i DV-ACCESS-70
[15] http://www.george-smart.co.uk/wiki/APRS_Callpass
[16] http://apps.magicbug.co.uk/passcode/
[17] http://westerndstar.co.uk/html/downloads.html
[18] http://tech4.pl/SQ9MDD/?p=92
[19] sq9mdd.qrz.pl
[20] www.westerndstar.co.uk/Downloads
[21] svxlink.de
[22] svxlink.sourceforge.net
[23] svxlink.sourceforge.net/man/man5/svxlink.conf.5.html
[24] CQDL 10/2012 – instrukcja instalacji i uruchomienia SvxLinku
[25] www.aprs-is.net
[26] www.aprs2.net
[27] „APRS iGate mit dem Raspberry Pi”, Stefan Hüpfer, DH5FFL, CQDL 4/2013, str. 241
[28] „Auf dem Weg zum LAN-SDR“, Stefan Hüpfer, DH5FFL, CQDL 4/2013, str. 239
[29] „Aus DVB-T-/DAB-Stick wird ein Breitband-SDR-RX“, Stefan Hüpfer, DH5FFL, CQDL 11/2012, str. 780
[30] www.realvnc.com
[31] ok1hra.nagano.cz
[32] remoteQTH.com
[33] www.dl8ma.de

W serii „Biblioteka polskiego krótkofalowca” dotychczas ukazały się:

Nr 1 – „Poradnik D-STAR”
Nr 2 – „Instrukcja do programu D-RATS”
Nr 3 – „Technika słabych sygnałów” Tom 1
Nr 4 – „Technika słabych sygnałów” Tom 2
Nr 5 – „Łączności cyfrowe na falach krótkich” Tom 1
Nr 6 – „Łączności cyfrowe na falach krótkich” Tom 2
Nr 7 – „Packet radio”
Nr 8 – „APRS i D-PRS”
Nr 9 – „Poczta elektroniczna na falach krótkich” Tom 1
Nr 10 – „Poczta elektroniczna na falach krótkich” Tom 2
Nr 11 – „Słownik niemiecko-polski i angielsko-polski” Tom 1
Nr 12 – „Radiostacje i odbiorniki z cyfrową obróbką sygnałów” Tom 1
Nr 13 – „Radiostacje i odbiorniki z cyfrową obróbką sygnałów” Tom 2
Nr 14 – „Amatorska radioastronomia”
Nr 15 – „Transmisja danych w systemie D-STAR”
Nr 16 – „Amatorska radiometeorologia”
Nr 17 – „Radiolatarnie małej mocy”
Nr 18 – „Łączności na falach długich”
Nr 19 – „Poradnik Echolinku”
Nr 20 – „Arduino w krótkofalarstwie” Tom 1
Nr 21 – „Arduino w krótkofalarstwie” Tom 2
Nr 22 – „Protokół BGP w Hamnecie”
Nr 23 – „Technika słabych sygnałów” Tom 3
Nr 24 – „Raspberry Pi w krótkofalarstwie”