Projekt był opublikowany w Make: magazine przez Sama Freemana i Wyntera Woodsa. Jego anglojęzyczny opis można znaleźć pod adresem ***www.makezine.com/projects/raspberry-pirate-radio/***. Atrakcyjną prezentację wideo nadajnika można obejrzeć na ***https://goo.gl/Gk3ai7***.

Raspberry PiRate Radio

Stacja radiowa z Raspberry Pi

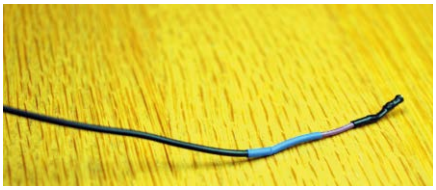
W dobie Internetu posiadanie własnej, amatorskiej stacji radiowej nie wydaje się już tak atrakcyjne, jak niegdyś. Teraz, chcąc podzielić się swoimi poglądami wystarczy założyć blog, przygotować tzw. podcast lub po prostu uruchomić kanał na YouTube. Ale by odbierać transmisje Internetowe trzeba mieć urządzenie komputerowe, podczas gdy analogowy sygnał FM można bez problemu odsłuchać z użyciem najtańszego odbiornika. I choć może wydawać się, że transmisja audio z komputera w paśmie radiowym z modulacją FM jest bezsensownym anachronizmem, w artykule pokazujemy interesujący projekt, w którym nadajnikiem radiowym jest Raspberry Pi, a przykładowe zastosowania tego rozwiązania – całkiem użyteczne.

Uwaga! Urządzenie umożliwia transmisję na częstotliwościach radiowych w zakresie od 1 MHz do 250 MHz, co pokrywa się z licencjonowanym w Polsce pasmem radiowym. Korzystanie z nadajnika może zakłócać inne sygnały radiowe, a nadawanie bez koncesji jest zabronione i może mieć konsekwencje prawne. Ten projekt ma jedynie charakter edukacyjny.

Podstawowy problem z samodzielnym budowaniem analogowego systemu radiowego jest konieczność przygotowania wszystkich potrzebnych obwodów radiowych, co dla elektroników dobrze obeznanych w domenie cyfrowej często stanowi nie lada wyzwanie. Jednakże w przypadku omawianego projektu problem ten znika. Jedyny potrzebny sprzęt to samo Raspberry Pi, karta SD i zasilacz oraz kawałek kabla. Choć może wydawać się to niewiarygodne, jest to wystarczający zestaw do niektórych aplikacji. Udowodnili to Oliver Mattos i Oskar Weigl, którzy przygotowali projekt o nazwie PiFM. Pomagał im Ryan Grassel, a wersję prezentowaną w niniejszym artykule opracował Wynter Woods, który na podstawie PiFM stworzył wygodny w użyciu odtwarzacz muzyki, transmitujący sygnał audio radiowo z modulacją FM do pobliskich odbiorników.

Montaż sprzętu

Omawiany projekt bardzo łatwo jest odtworzyć samodzielnie. Za antenę może posłużyć zwykły, jednożyłowy przewód (najlepiej miedziany, o przekroju ok. 2,5 mm^2). Jego optymalna długość zależeć będzie od częstotliwości, na której chcemy nadawać, przy czym ze względów praktycznych może okazać się że wygodniej jest by był krótszy. Dla częstotliwości 100 MHz, a więc bliskiej środka pasma, które można wykorzystać w tym projekcie, a zarazem blisko środka zakresu typowego radioodbiornika FM, przewód powinien mieć długość 75 cm. Może się jednak zdarzyć, że nie będzie



Rysunek 1. Przygotowana antena

on wystarczająco sztywny i trzeba będzie skrócić kabel. Pogorszy to parametry transmisji, ale na tyle nieznacznie, że w bliskim otoczeniu wciąż będzie można odbierać sygnał. Przykładowo, przewód o długości 40 cm będzie optymalny dla częstotliwości ok. 185 MHz.

Przycięty przewód należy podłączyć do Raspberry Pi – a dokładniej do jednego z wyprowadzonych pinów (domyślnie do GPIO4, czyli pin 7). W tym celu warto skorzystać z żeńskiego goldpinu, do którego przygotowany przewód zostanie przylutowany. Przydatne może okazać się też użycie taśmy izolacyjnej i kleju termicznego, za pomocą którego połączenie zostanie usztywnione, tak by zachować pożądany kierunek anteny. Pozostałe kroki polegają na wgraniu oprogramowania i danych na kartę oraz konfiguracji systemu, a następnie wpięciu karty i zasilania. Nic prostszego!

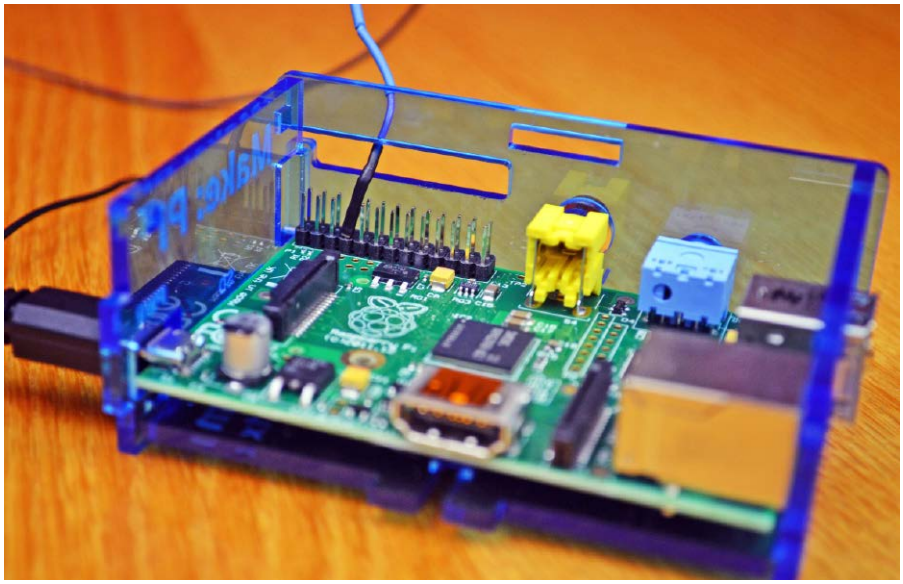
Wgranie oprogramowania

Gotowy obraz systemu z potrzebnym oprogramowaniem można pobrać z *https://goo.gl/MKMo3W*. Alternatywnie, można zainstalować standardową wersję obrazu i dograć do niej skrypty i biblioteki, dostępne na *https://goo.gl/0SXeIq*. Skrypt został zaprojektowany tak, by przestrzeń karty była podzielona na system i dane oraz by nie trzeba było wykonywać ręcznie żadnych poleceń by uruchomić odtwarzanie muzyki. Pliki w formatach takich jak MP3 czy FLAC będą automatycznie odtwarzane po włączeniu się urządzenia i transmitowane przez radio. Natomiast by wgrać pliki muzyczne na kartę należy po załadowaniu na nią obrazu i uruchomieniu z nią Raspberry Pi, połączyć się z RPI przez SSH i przekopiować dane z komputera lokalnego. Pliki można podzielić na katalogi – nie wpływa to na ich odtwarzanie. Osoby korzystające z systemu Windows, które nie są obeznane z poleceniami kopiowania danych przez SSH ze zdalnych lokalizacji mogą skorzystać z wygodnego, bezpłatnego programu WinSCP.

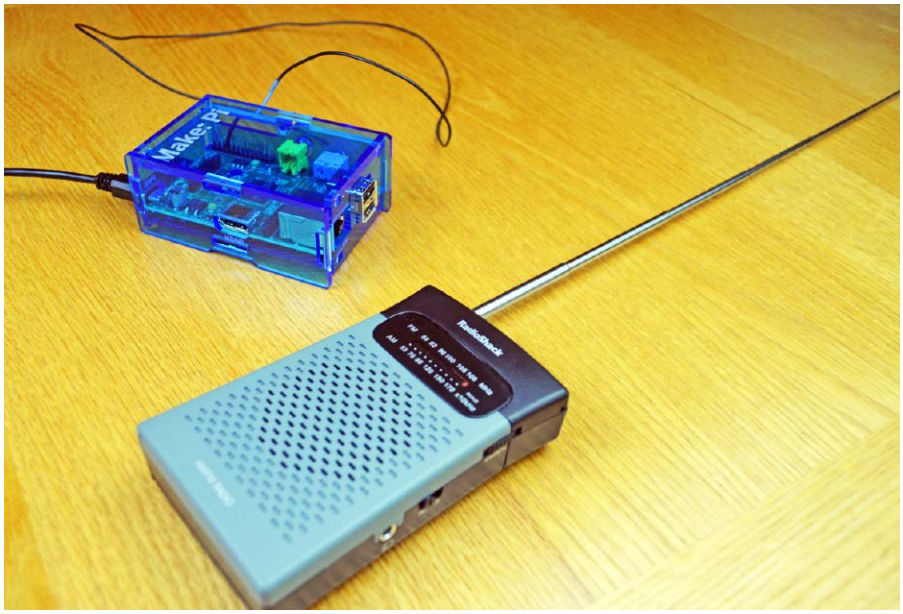
Konfiguracja

Proces konfiguracji nadajnika jest bardzo prosty i sprowadza się do wprowadzenia ustawień w kilku linijkach pliku tekstowego. Plik ten to **pirateradio.config** i zawiera domyślne ustawienia. W linijce frequency ustawiana jest częstotliwość, podana w megahercach (z kropką jako separatorem dziesiętnym) Najlepiej wybrać taką, na której w danej okolicy trudno znaleźć inne stacje radiowe, a zarazem taką, która odbierana jest przez odbiornik, z użyciem którego będziemy słuchać radia.

W linijce shuffle można określić, czy pliki mają być odtwarzane losowo (ustawienie True), czy w kolejności alfabetycznej (False). Linijka repeat_all pozwala ustawić



Rysunek 2. Zamontowana antena



Rysunek 3. Raspberry PiRate Radio w obudowie oraz radioodbiornik

powtarzanie odtwarzania plików w nieskończoność, a stereo_playback decyduje o tym, czy transmisja ma być stereofoniczna.

Uruchomienie

Urządzenie zaczyna działać natychmiast po uruchomieniu. Pierwszemu Raspberry Pi zajmuje to około 15 sekund, zanim fale radiowe zaczną być wysyłane w eter. W tym czasie można ustawić radioodbiornik na wybraną częstotliwość.

Oprogramowanie

Kod źródłowy PiRateRadio został napisany w Pythonie, ale korzysta ze skompilowanej biblioteki PiFM, która to została napisana w języku C++. W praktyce Raspberry PiRate Radio to wzorcowy przykład wykorzystania zewnętrznej biblioteki. Kod jest całkiem prosty – jego fragmenty zostały pokazane na **listingu 1**. Procedura działania sprowadza się do:

- zaimportowania bibliotek,
- ustawienia wartości domyślnych dla globalnych zmiennych,
- przełączenia wykonywania programu do trybu demona (usługi),
- wczytania i przeanalizowania pliku konfiguracyjnego, na podstawie którego zmieniane są wartości przypisane do zmiennych globalnych,
- uruchomienia biblioteki PiFM jako procesu podrzędnego o określonym strumieniu wejścia,
- wczytania listy plików poprzez rekursywne przeglądanie katalogów i przeszukiwanie ich pod kątem plików w formatach aac, mp3, wav, flac, m4a, ogg, pls i m3u,
- przekazania przygotowanej listy plików do funkcji ich odtwarzania zgodnie z zadanymi parametrami
- przeglądania kolejnych plików i jeśli są to pliki list utworów (m3u, pls), parsowania ich zawartości pod kątem

Rysunek 4. Raspberry PiRate Radio jako samochodowy nadajnik FM

ustalenia konkretnych plików audio do odtworzenia,

- wywołania kolejnej biblioteki – ffmpeg, która dekoduje wskazane pliki audio i przekazuje je na wyjście, które jest jednocześnie tym samym strumieniem, co wejście wcześniej uruchomionego podprocesu PiFM.

Biblioteka PiFM

Działanie biblioteki PiFM jest znacznie bardziej skomplikowane, choć jej kod źródłowy nie jest długi – ma jedynie ok. 650 linijek. Wykorzystuje wybrane wyjście GPIO szybko zmieniając jego stan, tak by powstawała fala zbliżona do wybranej częstotliwości nośnej, przy czym dodatkowo, dokładna częstotliwość jest nieco zmieniana w zależności od danych docierających na strumień wejściowy. Pozwala to na uzyskanie modulacji FM.

Początkowo twórcy mieli drobny problem, ponieważ jeśli procesor przełączał się na chwilę by robić coś innego niż nadawanie audio, w odbiorniku były słychać trzaski. Udało się pokonać tę trudność poprzez użycie kanałów DMA (Direct Memory Access). Z czasem wprowadzono też usprawnienie umożliwiające nadawanie dźwięku stereo.

Cały program musi być uruchamiany z uprawnienia roota, gdyż biblioteka PiFM mapuje adres szyny GPIO do pamięci wirtualnej. Następnie uruchamia generator na GPIO4, ustawiając jego częstotliwość na np. 100 MHz. Gdy na strumień wejściowy zaczynają trafiać dane, służą one do modulacji częstotliwościowej powodując, że generator emituje sygnał o częstotliwości z zakresu od 100,025 MHz do 99,975 MHz. Szybkość procesora Raspberry Pi jest wystarczająca by być w stanie nadawać na tych częstotliwościach dźwięk modulowany z 16-bitową rozdzielczością.

Podsumowanie i ocena projektu

Omawiany projekt można uznać za bardzo udany, a nawet zaskakująco łatwy w odtworzeniu. Fakt, że bez żadnych dodatkowych komponentów aktywnych, a jedynie po podpięciu niedługiego przewodu można

Listing 1. Fragmenty kodu źródłowy Raspberry PiRate Radio

```
try:       #próba importu biblioteki Pythona 3.x
    import configparser
except: #import biblioteki Pythona 2.x, jeśli operacja nie powiodła się dla Pythona 3.x
    import ConfigParser as configparser
finally:   #import pozostałych bibliotek
    import re
    import re
    import random
    import sys
    import os
    import threading
    import time
    import subprocess

#zmienne globalne, do przechowywania ustawień
fm_process = None
on_off = [„off”, „on”]
config_location = „/pirateradio/pirateradio.conf”

frequency = 87.9
shuffle = False
repeat_all = False
merge_audio_in = False
play_stereo = True
music_dir = „/pirateradio”

music_pipe_r,music_pipe_w = os.pipe()
microphone_pipe_r,microphone_pipe_w = os.pipe()

#główne polecenia programu
def main():
    daemonize()
    setup()
    files = build_file_list()
    if repeat_all == True:
        while(True):
            play_songs(files)
    else:
        play_songs(files)
    return 0

#tworzenie listy plików w oparciu o przeszukiwanie katalogu z muzyką
def build_file_list():
    file_list = []
    for root, folders, files in os.walk(music_dir):
        folders.sort()
        files.sort()
        for filename in files:
            if re.search(„.(aac|mp3|wav|flac|m4a|ogg|pls|m3u)$”, filename) != None:
                file_list.append(os.path.join(root, filename))
    return file_list

#główna funkcja odtwarzająca pliki
def play_songs(file_list):
    print(„Playing songs to frequency „, str(frequency))
    print(„Shuffle is „ + on_off[shuffle])
    print(„Repeat All is „ + on_off[repeat_all])
    # print(„Stereo playback is „ + on_off[play_stereo])

#w przypadku odtwarzania losowego, lista plików jest wstępnie mieszana
    if shuffle == True:
        random.shuffle(file_list)
    with open(os.devnull, „w”) as dev_null:
        for filename in file_list:
            print(„Playing „,filename)
#w przypadku gdy plik jest tak naprawdę listą utworów, jest on przetwarzany przed odtwarzaniem
            if re.search(„.pls$”, filename) != None:
                streamurl = parse_pls(filename, 1)
                    if streamurl != None:
                    print(„streaming radio from „ + streamurl)
                    subprocess.call([„ffmpeg”,”-i”,streamurl,”-f”,”s16le”,”-acodec”,”pcm_
s16le”,”-ac”, „2” if play_stereo else „1” ,”-ar”,”44100”,”-”],stdout=music_pipe_w, stderr=dev_null)
            elif re.search(„.m3u$”, filename) != None:
                streamurl = parse_m3u(filename, 1)
                    if streamurl != None:
                    print(„streaming radio from „ + streamurl)
                    subprocess.call([„ffmpeg”,”-i”,streamurl,”-f”,”s16le”,”-acodec”,”pcm_
s16le”,”-ac”, „2” if play_stereo else „1” ,”-ar”,”44100”,”-”],stdout=music_pipe_w, stderr=dev_null)
            else:
#niezależnie od formatu pliku jest on dekodowany programem ffmpeg, a przekonwertowany strumień jest przekazywany dalej poprzez music_pipe_w
                    subprocess.call([„ffmpeg”,”-i”,filename,”-f”,”s16le”,”-acodec”,”pcm_
s16le”,”-ac”, „2” if play_stereo else „1” ,”-ar”,”44100”,”-”],stdout=music_pipe_w, stderr=dev_null)

#wczytywanie ustawień z użyciem wygodnej biblioteki config.
def read_config():
    global frequency
    global shuffle
    global repeat_all
    global play_stereo
    global music_dir
    try:
        config = configparser.ConfigParser()
        config.read(config_location)
    except:
        print(„Error reading from config file.”)
    else:
        play_stereo = config.get(„pirateradio”, ‚stereo_playback’, fallback=True)
        frequency = config.get(„pirateradio”,’frequency’)
        shuffle = config.getboolean(„pirateradio”,’shuffle’,fallback=False)
        repeat_all = config.getboolean(„pirateradio”,’repeat_all’, fallback=False)
        music_dir = config.get(„pirateradio”, ‚music_dir’, fallback=”/pirateradio”)

#funkcja dzieląca pozycje pliku w formacie pls na pojedyncze pliki dźwiękowe (pominięta)
def parse_pls(src, titleindex):
```

```
Listing 1. cd.
#funkcja dzieląca pozycje pliku w formacie m3u na pojedyncze pliki dźwiękowe (pominięta)
def parse_m3u(src, titleindex):

#funkcja tworząca drugi proces i zamykająca pierwotny. Dzięki temu cały program działa jako
demon, uwalniając konsolę użytkownika, który go wywołał
def daemonize():
    fpid=os.fork()
    if fpid!=0: #pierwotny proces ma ustalone fpid i w ten sposób można go rozpoznać i zam-
knąć
        sys.exit(0)

#funkcja wywołująca wczytanie konfiguracji a następnie włączenie procesu PiFM w tle
def setup():
    global frequency
    read_config()
    run_pifm()

#uruchamianie biblioteki PiFM jako podrzędnego procesu, w którym strumień music_pipe_r jest
traktowany jako wejście
def run_pifm(use_audio_in=False):
    global fm_process
    with open(os.devnull, „w”) as dev_null:
        fm_process = subprocess.Popen([„/root/pifm”,”-”,str(frequency),”44100”, „stereo” if
play_stereo else „mono”], stdin=music_pipe_r, stdout=dev_null)

main()
```

Rysunek 5. Raspberry PiRate Radio z akumulatorami i usztywnioną anteną, do nadawania radia na otwartym powietrzu

stworzyć sprawnie działający, krótkozasięgowy nadajnik radiowy FM dobrej jakości nie wydaje się oczywisty. Okazuje się, że zasięg sygnału w otwartej przestrzeni sięga ok. 100 m, a w zamkniętych pomieszczeniach potrzeba przejść kilkadziesiąt metrów, by sygnał zaczął zanikać. W przypadku niepodłączenia anteny, emitowany sygnał sięga na ok. 10 cm. Natomiast już kilkadziesiąt metrów wystarcza by urządzić sobie np. ciche plenerowe kino samochodowe, czy tzw. silent disco. Bardzo niskim kosztem, gdyż z wykorzystaniem powszechnie dostępnych i instalowanych w wielu

Rysunek 6. Atrakcyjna obudowa do Raspberry PiRate Radio, wydrukowana na drukarce 3D. Jej projekt jest dostępny bezpłatnie pod adresem: *https://goo.gl/xitAl1*

urządzeniach radioodbiorników FM można nadawać dźwięk niesłyszalny dla osób postronnych. Nie ma potrzeby stosowania drogich urządzeń komputerowych ani specjalnych odtwarzaczy cyfrowych. Do tego, uzyskiwana jakość dźwięku jest bardzo dobra, dzięki obsłudze kanałów DMA i sygnału stereofonicznego.

Autorzy rozbudowali projekt także o obsługę wejścia mikrofonowego, dzięki czemu można nadawać własne audycje lub pobawić się w disc-jockeya, który mówi do uczestników na zorganizowanym silent disco.

Aby Raspberry PiRate Radio było bardziej przenośne, warto zamontować je w obudowie i podłączyć do akumulatora. Natomiast wejście mikrofonowe można też użyć do podłączenia wyjścia sygnałowego konsolety dyskotekowej, na której miksowana jest muzyka.

Marcin Karbowniczek, EP