Internet Radio for Raspberry Pi

Hamidreza Shariatmadari
Department of Communications and Networking
Aalto University, Finland
hamidreza.shariatmadari@aalto.fi

ABSTRACT
Internet based radio services has become increasingly popular over the last few years for listening to radio channels. This services offer continuous stream of audio accessible from all around the world. The wide deployment of Internet access with high reliability has made the radio streaming available for various devices. This paper describes the design and implementation of radio streaming on the Raspberry Pi platform.

Keywords
Internet Radio, Embedded System, Raspberry Pi, Linux

1. INTRODUCTION
The Internet is one of the greatest achievement of humankind. It was initially designed to allow computers to share information on research and development in scientific and military fields[1]. But soon, the Internet became available for commercial purposes and many new applications were invented over it. New Internet services have permeated and transformed all aspects of our lives. Internet radio which is also known as web radio, is a service which provides receiving and playing audio data originating from the internet. This requires a technical features which allows audio to be digitized and split into small pieces for transmission across the Internet.

While traditional radio stations are subject to extensive government regulations, Internet radio stations remain largely unregulated. In addition, establish a new channel through the online streaming is less costly compared to broadcasting in a traditional way. Internet radio services are usually accessible from anywhere in the world and the quality of these services do not degrade over distance. These features have made Internet radio service popular among radio providers. The movement from traditional radio broadcasting to the Internet radio service is also evident in cell phone industry. Most of mobile phone manufacturers no more integrate their new products with traditional radio receivers. Instead, they simply provide a program which enables receiving radio channels via the Internet connection.

Audio streaming is a continuous data transferring and receiving. To set up a such system, usually three different entities are required: encoder, compressor and server. The encoder part is responsible for converting audio content into a streaming format, utilizing a lossy audio codec. The lossy compression reduces the transmission bandwidth and storage requirement of audio data. The server part continuously transmits stream data serially over the network. The client retrieves the content and plays the audio. For a live broadcast, the encoder and streamer should operate together in real-time. An audio feed runs to the sound card of a computer running the encoder software at the broadcast location and the stream is uploaded to the streaming server. In addition, the client and servers require streaming protocols for services negotiation, data transmission and network addressing. Streaming protocols can be classified into three different categories: network-layer protocol, transport protocol and session control protocol. Network-layer protocol is responsible of address resolution and network addressing. Transport protocols, including TCP, UDP and RTC/RTCP, provide end-to-end data transmissions. Session control protocol, such as RTSP and SIP, provides mechanisms to manage the session between end-use application processes[2].

This paper demonstrates how to design and implement an Internet radio receiver using a Linux embedded system. The rest of this paper is organized as follows. Section 2 describes the platform and hardware utilized to establish the radio receiver. Section 3 explains the software installation for receiving and controlling the radio channels. The system evaluation and the summary are provided in Section 4 and Section 5.

2. HARDWARE SETUP
The Raspberry Pi (RPi) is an embedded platform, powered by the Broadcom BCM2835 System-on-Chip (SoC) which includes a 32-bit ARM1176JZF-S as a processor. The normal processor clock is 700 MHz, however it can be overclocked up to 1 GHz. The SoC includes 256MB RAM and Video-core IV GPU. The board uses an SD card reader for both booting and storage purposes. It has both HDMI and analog composite video outputs. In RPi model B, two USB ports and an analog audio port are available for connecting peripheral devices. The 10/100 Ethernet port provides network connection for the board. A General Purpose I/O (GPIO) connector allows to interface the board with other equipment. These specifications make this platform a powerful tool to establish an Internet radio. Hence, a RPi model B is used for this experiment. The Ethernet port is used to access to the Internet and the analog audio port to play audio stream. An LCD and some buttons are connected to the board using GPIO connector to display information and apply new commands. Figure 1 shows how the platform is wired to the LCD and push buttons. More information
The R/W pin to the ground to put the LCD in the reading mode. 4-bit data transmission is also utilized which entails less wire connections.

2.3 Input buttons

Five push buttons are utilized to change radio channels, set the volume and apply other commands. These buttons are labeled as Menu, Up, Down, Right, Left in the Figure 1. These buttons are connected to the Raspberry Pi using 5 GPIO pins. Each button is connected to a pin through a 1K ohm resistor. This causes that input voltage of a GPIO pin becomes high when the corresponding button is pressed. A 10K ohm resistor is used for each input pin as a pull-down resistor to ensure that the voltage on the GPIO pin becomes low when the button is not pressed, and the floating phenomena not happens. In this way, the platform is capable of distinguishing the status of buttons by checking the voltage level on GPIO pins regularly.

3. SOFTWARE INSTALLATION

This application should perform various tasks, including receiving radio streams and playing it, presenting information on the LCD and apply new commands received from push buttons. Implementation of these tasks are done by using a Linux Operating System (OS) and some additional programs. The rest of this Section provides information about the OS and programs used for this application. The additional programs are stored in a compressed file called pi_radio.tar.gz, which can be downloaded from the website[4]. This file contains the required libraries to control the LCD and GPIO, and the program to run the radio. The command shown in Appendix A.1 also allows to download and unzip this file directly from the command line.

3.1 Raspbian

In order to use all functionality of RPi efficiently, it is essential to install a proper OS. An OS gives the freedom to install additional programs and control hardware components easily. Different distributions of Linux have been modified and optimized for this platform. Raspbian is a powerful and stable OS based on Debian Linux release. This release completely supports the ARMv6 instruction sets and hard float math operations.

The Raspbian wheezy with kernel 3.6 is chosen as OS in this project. The OS should be installed on a SD card with at least 2 GB storage capability. In addition to the OS, a proper bootloader should be stored in the SD card. The full installation description of this OS can be found in[5].

3.2 Network Time Clock

The Raspberry Pi does not have an internal Real Time Clock (RTC), hence it is required to initialize the time when it is powered. An accurate way to set the is using the Network Time Protocol (NTP) which is widely used over the Internet. An accurate way to set the is using the Network Time Protocol (NTP) which is widely used over the Internet to make sure that computers have the same time[6]. In order to make the platform run the network time clock daemon at boot time, the network time daemon should be enabled. The provided command in Appendix A.2 achieves this goal.

3.3 Music Player Daemon

Music Player Daemon (MPD) is an open source application for Linux OS which allows remote access for playing music

<table>
<thead>
<tr>
<th>Pin</th>
<th>Name</th>
<th>Func</th>
<th>Pin</th>
<th>Name</th>
<th>Func</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>3.3 V</td>
<td></td>
<td>P2</td>
<td>5.0 V</td>
<td></td>
</tr>
<tr>
<td>P3</td>
<td>GPIO 2</td>
<td>I2C1 SDA</td>
<td>P4</td>
<td>5.0 V</td>
<td></td>
</tr>
<tr>
<td>P5</td>
<td>GPIO 3</td>
<td>I2C1 SCL</td>
<td>P6</td>
<td>GND</td>
<td></td>
</tr>
<tr>
<td>P7</td>
<td>GPIO 4</td>
<td>GPCLK0</td>
<td>P8</td>
<td>GPIO 14</td>
<td>UART0 TxD</td>
</tr>
<tr>
<td>P9</td>
<td>GND</td>
<td></td>
<td>P10</td>
<td>GPIO 15</td>
<td>UART0 RXD</td>
</tr>
<tr>
<td>P11</td>
<td>GPIO 17</td>
<td></td>
<td>P12</td>
<td>GPIO 18</td>
<td>PC1 CLK</td>
</tr>
<tr>
<td>P14</td>
<td>GPIO 25</td>
<td></td>
<td>P13</td>
<td>GND</td>
<td></td>
</tr>
<tr>
<td>P15</td>
<td>GPIO22</td>
<td></td>
<td>P16</td>
<td>GPIO 23</td>
<td></td>
</tr>
<tr>
<td>P17</td>
<td>3.3 V</td>
<td></td>
<td>P18</td>
<td>GPIO 24</td>
<td></td>
</tr>
<tr>
<td>P19</td>
<td>GPIO 10</td>
<td>SPI0 MOSI</td>
<td>P20</td>
<td>GND</td>
<td></td>
</tr>
<tr>
<td>P21</td>
<td>GPIO 9</td>
<td>SPI0 MISO</td>
<td>P22</td>
<td>GPIO 25</td>
<td></td>
</tr>
<tr>
<td>P23</td>
<td>GPIO 11</td>
<td>SPI0 SCLK</td>
<td>P23</td>
<td>GPIO 8</td>
<td>SPI0 CE0 N</td>
</tr>
<tr>
<td>P25</td>
<td>GND</td>
<td></td>
<td>P26</td>
<td>GPIO 7</td>
<td>SPI0 CE1 N</td>
</tr>
</tbody>
</table>

Table 1: GPIO pin description

about components and wiring shown in the Figure, beside the specifications of the platform are provided in the remaining of this section.

2.1 Raspberry Pi GPIO

GPIO is a generic pin on a chip whose behavior can be controlled through software. Peripherals and expansion boards can access to the CPU by exposing the inputs and outputs. Available GPIO pins in RPi signal on a 2x13 expansion header. Total 17 GPIO pins can be used as simple digital input/output or reconfigured to provide alternate functions. Pin descriptions are summarized in Table 1. Pin 3 and 5 are preset to be used as an I2C interface. The SPI interface can be utilized using pin 19, 21, 23, 24 and 26. By default, pins 8 and 10 are assigned to the alternate UART function for serial connection. The GPIO port provides both 3.3 and 5 V for external boards, but the RPi’s voltage level is 3.3 V and there is no over-voltage protection on the board.

2.2 LCD

In order to display information, including time and radio channel name, a peripheral character LCD is used. This module has two rows of 16 characters for displaying text. It contains the HDD44780 controller which is an industry standard interface for a variety of LCD displays[3]. This controller has two built in registers namely Command and Data registers. The Command register can be used to run a predefined task like initialization, clearing or controlling the display, setting the cursor position and etc. However, the Data registers can access to the CPU by exposing the inputs and outputs. The SPI interface is used for contrast and Vdd pins are preset to be used as an I2C interface. The SPI interface can be utilized using pin 19, 21, 23, 24 and 26. By default, pins 8 and 10 are assigned to the alternate UART function for serial connection. The GPIO port provides both 3.3 and 5 V for external boards, but the RPi’s voltage level is 3.3 V and there is no over-voltage protection on the board.

2.3 Input buttons

Five push buttons are utilized to change radio channels, set the volume and apply other commands. These buttons are labeled as Menu, Up, Down, Right, Left in the Figure 1. These buttons are connected to the Raspberry Pi using 5 GPIO pins. Each button is connected to a pin through a 1K ohm resistor. This causes that input voltage of a GPIO pin becomes high when the corresponding button is pressed. A 10K ohm resistor is used for each input pin as a pull-down resistor to ensure that the voltage on the GPIO pin becomes low when the button is not pressed, and the floating phenomena not happens. In this way, the platform is capable of distinguishing the status of buttons by checking the voltage level on GPIO pins regularly.

3. SOFTWARE INSTALLATION

This application should perform various tasks, including receiving radio stream and playing it, presenting information on the LCD and apply new commands received from push buttons. Implementation of these tasks are done by using a Linux Operating System (OS) and some additional programs. The rest of this Section provides information about the OS and programs used for this application. The additional programs are stored in a compressed file called pi_radio.tar.gz, which can be downloaded from the website[4]. This file contains the required libraries to control the LCD and GPIO, and the program to run the radio. The command shown in Appendix A.1 also allows to download and unzip this file directly from the command line.

3.1 Raspbian

In order to use all functionality of RPi efficiently, it is essential to install a proper OS. An OS gives the freedom to install additional programs and control hardware components easily. Different distributions of Linux have been modified and optimized for this platform. Raspbian is a powerful and stable OS based on Debian Linux release. This release completely supports the ARMv6 instruction sets and hard float math operations.

The Raspbian wheezy with kernel 3.6 is chosen as OS in this project. The OS should be installed on a SD card with at least 2 GB storage capability. In addition to the OS, a proper bootloader should be stored in the SD card. The full install description of this OS can be found in[5].

3.2 Network Time Clock

The Raspberry Pi does not have an internal Real Time Clock (RTC), hence it is required to initialize the time when it is powered. An accurate way to set the is using the Network Time Protocol (NTP) which is widely used over the Internet to make sure that computers have the same time[6]. In order to make the platform run the network time clock daemon at boot time, the network time daemon should be enabled. The provided command in Appendix A.2 achieves this goal.

3.3 Music Player Daemon

Music Player Daemon (MPD) is an open source application for Linux OS which allows remote access for playing music
and managing playlists[7]. The playlist can be provided by the server or as a file in PLS format. The PLS does not contain the music file, instead it points to the music file stored somewhere else. For streaming application, the playlist file should contain the URL of the streaming media. This program has additional interface, a command line interface for the player, called Media Player Client (MPC). It provides the capability for the player to be controlled through command line and other programs. The command for installing this program and the interface is provided in Appendix A.3.

3.4 GPIO library
GPIO is used to control the external character LCD and check the status of push buttons. In order to control the GPIO pins, a GPIO library is necessary. The GPIO-Python is a class, written in Python language, provides access to GPIO pins. It allows to set each GPIO pin as input or output, read the value or set the value. However, this class is only proper for applications which are not timing critical, since the timing behavior of it under the Linux kernel is not predictable. The commands provided in Appendix A.4 install the latest GPIO library on Raspbian. As previously mentioned, two GPIO 8 and 15 pins are configured for the serial interface by default, while they are used for normal operation by this design. Hence, it is necessary to disable the serial interface by removing all references to ttyAMA0 in /boot/cmdline.txt and /etc/inittab files.

3.5 LCD library
The character LCD used in this project has the HDD44780 controller. There are many libraries available to control this chip. The library which is used in this project is a standalone class written in Python and can be used for any other LCD with the same controller. To use this class properly, the wiring connections should be defined in the first part of the code. The wiring shown in Figure 1 is in compliance with the default setting, therefore additional modification is not required. The zip file contains this library with the name lcd_class.py.

3.6 Radio Controller
The Radio Controller is the main file that is run in the background and controls the device. This program is also written in Python language. When this program is executed, it first checks the network connection of the RPi. If it finds the network connection, it calls the LCD library and shows the IP on the LCD. After this step, it calls the MPD and starts to scan all channels stored in playlist. In this way, it obtains radio channels’ status and their name. The MPD then
starts to receive and play the first channel in the playlist. The main program continuously reads the status of GPIO pins connected to push buttons. If a button is pressed, the program calls the subfunction to apply the proper command. The main program is stored in the zip file with name radiod.py.

3.7 Boot manipulation
Its beneficial to run the radio program at the end of the boot procedure. In this way, the platform runs the program automatically when it is powered. The file /etc/rc.local is executed at the end of the boot process. Adding the command provided in Appendix A.5 to the end of this file, makes the radio controller to be executed. The normal boot time of the Raspberry Pi is quite high and it takes about 1 minute that the radio be launched after the platform is powered. This amount of time can be reduced by some software modifications. The first revision is to remove unnecessary packages which are already installed on Raspbian by default. The corresponding commands for this modification are provided in Appendix A.6. This results in about 10 seconds reduction in booting time. Adding kernel options like fastboot and quiet in the file /boot/cmdline.txt also saves few seconds during booting time.

4. PERFORMANCE EVALUATION
As described in previous section, this application consists of the main radio controller and some additional libraries and programs. The most of processing and memory usage for this application are associated with the controller and MPD programs. The ps aux command displays information about running applications and their resource usages. The controller roughly utilizes 2% of processor and 1% of available memory. The MPD needs more resources for execution, it almost uses 21% and 3% of processor and memory. The whole memory and processing usage of the application is quite low and it operates seamlessly.

5. SUMMARY
This paper described how the RPi platform can be used as Internet radio players. The ability of running the Linux OS on this platform allows to easily develop complex applications using other compatible programs. The interfacing of this platform with additional hardware components and employing required libraries are also explained.

6. REFERENCES

APPENDIX
A. SOFTWARE CONFIGURATIONS
This Section contains the required commands for installation and configuration of programs to establish the Internet radio player.

A.1 Enable Network Time Daemon
Download and unzip the Radio program:

Create a directory called /home/pi/radio and download the zip file using:

```
wget http://www.bobrathbone.com/raspberrypi/source/pi_radio.tar.gz
```

Then untar the file with the following command:

```
tar -xvf pi_radio.tar.gz
```

A.2 Enable Network Time Daemon
The following command enables the network time daemon which allows the platform synchronizes the time with a Network Time Protocol:

```
update-rc.d ntp enable 2 3 5
```

A.3 Install MPD
Downloads and install mpd and mpc programs:

```
apt-get install mpd mpc
```

In order to enable IPv6 and prevent start-up messages, following parameters should be amended in the configuration file /etc/mpd.conf.

```
change parameter:
bind to address "localhost" to "any"
also change parameter:
zeroconf_enabled "yes" to "no"
```

A.4 Update GPIO library
The following commands update the GPIO library.

```
dpkg -i python-rpi.gpio_0.5.2a-1_armhf.deb
dpkg -i python3-rpi.gpio_0.5.2a-1_armhf.deb
```

A.5 Start radio program automatically
The following line should be added at the end of file /etc/rc.local.

```
sudo ./home/pi/radio/radiod.py start
```

A.6 Boot manipulation
The following commands would remove some unnecessary services from Raspbian:

```
sudo apt-get -y purge libx11-6 libgtk-3-common xkb-data
lxde-icon-theme raspberrypi-artwork penguinspuzzle
apt-get -y install libfreetype6
```