An Advanced Streaming Internet Radio Player with Raspberry Pi

Mr. Jeremy Wayne Gilreath, Guilford College

Jeremy Gilreath earned his B.A. in Sociology from the University of North Carolina at Greensboro, and his B.S. in Computing Technology and Information Systems from Guilford College, both in Greensboro, NC.

Dr. Chafic BouSaba, Guilford College

An Advanced Streaming Internet Radio Player with Raspberry Pi

Abstract

Our paper describes a challenging and enjoyable undergraduate student project that details the process of configuring a Raspberry Pi into an advanced multimedia player as a headless system controllable by infrared remote or secure shell (SSH) protocol. This paper provides a methodological, step-by-step set of specific instructions on how to replicate this project. The undergraduate student applied concepts from operating systems (OS), networks, and electronics into practical steps to exploit readily available open-source software packages and highly-customizable hardware components. Electronic components were soldered to build a modified printed circuit board (PCB) with a liquid crystal display (LCD) screen and push-buttons, and were successfully combined with an external sound card, wireless universal serial bus (USB) card, and an infrared (IR) receiver into a fully assembled, affordable, and reliable device. This developed device can be controlled using any IR-capable remote, is capable of playing high-quality audio, and circumvents advertisements on the free audio streaming web service Pandora Radio using a pre-existing free or paid account. Using this device improves the user's experience of Pandora Radio in several ways while keeping this device portable and preserving all functionality of the Raspberry Pi itself. The goal of this project is to use available tools and integrate various technological fields into a deliverable consumer product. Consumer electronics continue to place increasing emphasis on supporting open-source hardware and software, and the Raspberry Pi provides an affordable, flexible, multi-purpose platform for both beginners and experts to personalize into a wide range of useful and specialized products.

Introduction

The Raspberry Pi, see Figure 1, is an inexpensive and small-sized single-board computer invented by the Raspberry Pi Foundation,[1] a registered charity in the United Kingdom whose mission is to use the Raspberry Pi in classrooms to promote affordable education in the computer sciences worldwide. Students of all ages across the world use this open-source device to learn and experiment in computer science, programming, and electronics.[2] A variety of Linux-based OS's are supported on the device, and many programming languages such as Java, C, Python, and Scratch lay a foundation for ingenuity and craftsmanship in competitions and events where groups of like-minded dabblers and inventors gather for mutual co-creation.[3]

The first section of this paper gives a full description of the Raspberry Pi, an overview of all our hardware and software modifications, and the order they will be completed in. The second section walks through all of these steps to successfully complete the project. The final section of this paper discusses the results of this project, the implications it has for engineering in education, and concludes with how to move forward with this project for future modifications. All important files created or modified during this project can be found at a stable location online[4] for users to review at any point in time, and will provide the proper configuration or source code for the project step in question. Also included is a detailed 145-step guide for the completion of all steps in this project, and is more lengthy and explicit than what is covered here.

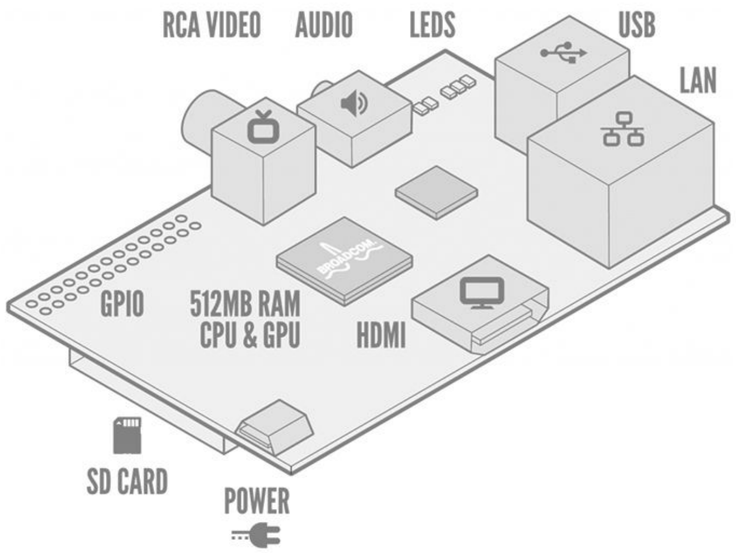

Figure 1: Diagram of Raspberry Pi Model B

Being slightly larger than a credit card and armed with an array of ports for input and output, the Raspberry Pi comes in two models: Revision 1.0 boards (Model A) and Revision 2.0 boards (Model B). Shared features include a Broadcom BCM2835 system on a chip [a 700MHz ARM11 family central processing unit (CPU) with an ARMv6 instruction set, Broadcom VideoCore IV 1080p30 graphics processing unit (GPU), digital signal processor, 256MB (megabyte) Secure Digital random access memory (SDRAM) shared with the GPU, and a single USB 2.0 port,] a camera serial interface (CSI) video input for 14 megapixel camera module, composite RCA video output, High-Definition Multimedia Interface (HDMI) video output, 3.5mm audio jack, MultiMediaCard (MMC) / SD / SD Input/Output card (SDIO) slot, several low-level peripherals [eight general-purpose input/output (GPIO) pins, a universal asynchronous receiver/transmitter (UART), an Inter-Integrated Circuit (I2C) and Serial Peripheral Interface (SPI) bus, and Inter-IC Sound (I2S) audio] as well as a MicroUSB power source. The Model B Raspberry Pi has twice the SDRAM, an additional USB 2.0 port (both of which are moved to an integrated 3-port USB hub,) and a 10/100 MBit/s Ethernet USB adapter which takes up one of these ports on the hub; the tradeoff is that the Model B takes 3.5W of power as opposed to the 1.5W required by the Model A. Both Models run on a variety of Linux distributions such as Raspbian (a Debian Wheezy port) and Pidora (a Fedora port), in addition to other OS such as OpenElec and RISC OS. The official distributions are optimized for the CPU's ARMv6 instruction set and are freely available for download, yet many more are available for download.[5] Nearly all distributions are Linux-based, with the notable exception of Plan 9 developed by Bell Labs. Schematics for Model A and Model B reveal the different electronic circuitry for both, and these documents should be referenced for correct pin-outs of the GPIO pins to be used in Model B.[6] Recently, these two Raspberry Pi models have been updated to A+ and B+ versions.

After acquiring the necessary hardware and constructing the components, the Raspbian operating system will be installed and appropriate configuration options will be set for the Raspberry Pi to run entirely without a desktop environment and eventually without a keyboard or mouse. Package lists, software versions, system software, and operating system distributions will then be updated to their most recent versions. The system name and default password will be changed for security purposes, and wireless Internet will be configured in order to keep the Raspberry Pi from depending on a wired network cable. Various dependencies, scripts, and open-source software will be downloaded to allow the Raspberry Pi to stream internet radio from Pandora.com without the use of a web browser. System files will be edited and configured along the way, and audio output will be routed from HDMI to USB ports so that headphones or speakers may be used in conjunction with an external USB 5.1 sound card. SSH will be used to perform the majority of this so that the Raspberry Pi does not need to be connected to a keyboard, mouse, or video output, and will be configured to automatically log in as the user when powered on as a headless system, launch the appropriate software, and connect to Pandora to begin streaming music. Finally we will install dependencies, scripts, and open-source software to enable infrared remote control of the streaming software and the Raspberry Pi by configuring a First-In First-Out (FIFO) file and configuring the IR remote.

Open-source software and hardware benefits consumers and producers by encouraging competition between and within organizations, fostering growth from volunteer contributors, and enhancing security by being freely auditable by any sufficiently motivated entity.[7] They also provide a permissive platform for discovery and experimentation on behalf of learners and innovators from a variety of backgrounds. A key purpose of this project is to provide students with limited experience the opportunity to implement a hands-on project that requires no prior experience working with computer programming or open-source hardware and software. The goal is to present a multidisciplinary approach to design this media tool and encourage more students and researchers from various fields to publish on the practical and theoretical applications of the Raspberry Pi. There has been a noticeable lack of scholarly publications on the Raspberry Pi and its applications in commercial, scientific, and academic settings, despite having originally been designed to stimulate academic growth in the computer sciences. The shortage of publications motivated us to document the project in a detailed technical paper.

Some of the more novel products using the Raspberry Pi include a 32-node Beowulf Cluster for a personal mini-supercomputer,[8] as well as the sending of a Raspberry Pi up to nearly 40,000 feet in Earth's atmosphere for a High-Altitude Ballooning project, resulting in some stunning images of the Earth in the process.[9] Researchers are drawn to the Raspberry Pi as well: one individual devised a way for real-time conversations in two different languages to be translated on-the-fly and displayed as subtitles on wearable computing glasses,[10] while a scientist used a Raspberry Pi to develop open-brainwave technology that allows its users to control nearly any application on their Raspberry Pi, using Electroencephalography to read their thought patterns as a hotkey.[11]

Scholarly literature on the Raspberry Pi is sparse, with online documents and printed periodicals making up the majority of literature. It is safe to assume that a fair number of people have modified their Raspberry Pi to be a media center, an audio player, a streaming web player, and/or controllable by remote. In fact, several operating systems focus on distributing the free and open-source X-Box Media Center software with the very goal of turning the Raspberry Pi into a home media center that can play audio and video from devices attached either directly to the Raspberry Pi or to the same network. Additionally, the web browsers that come standard on many operating systems for the Raspberry Pi are compatible with multiple web streaming services.[12]

There is clearly a demand for multimedia entertainment using the Raspberry Pi. Focusing on its audio capabilities while restricting the system to only a command-line interface keeps system overhead to a minimum, instead using the scrolling multicolor LCD screen to provide visual feedback to the user while allocating as much RAM to the CPU as possible. Use of an IR remote and a minimum of tethered peripherals was integral to keeping the Raspberry Pi portable. The device could also be powered by a single portable battery pack, such as the low-cost PowerAdd Pilot S, making our system completely portable. The device operates continuously for 11 hours using this 12000mAh (milliamp hours) battery pack, which has two USB outputs that provide 1.5V/1A and 2.5V/2.1A respectively, enough to fully power the Raspberry Pi and speakers.

Methods

The 16x2 Character LCD and Keypad Kit should be constructed first, since the Raspberry Pi itself requires no physical setup. A very thorough step-by-step tutorial for assembling this Kit already exists from the vendor Adafruit[13], which is a company that offers tools and online resources for learning electronics. Table 1 lists the necessary hardware and Table 2 lists the temporary hardware and tools needed for setup and construction. Figure 2 shows a flowchart for all major hardware assembly and software configuration steps in this project, while figure 3 depicts a comprehensive system diagram for all hardware components.

Table 1: Required Hardware

Required hardware:
(1) MicroUSB Cable (1) PiBow case mod[14] (1) Model B Raspberry Pi (1) TSOP38238 IR Receiver Module (1) 16x2 Character LCD + Keypad Kit (1) Extra-tall and extra-long 2x13 female stacking header[15]

(1) 90-264VAC in, 5V@1A VDC out wall plug USB Power Supply
(1) PiBow Raspberry Pi case compatible with a Model B Raspberry Pi
(1) SD / MicroSD + Adapter Card w/ minimum 4GB (gigabyte) storage capacity
(3) 28 AWG (7 strands @ 36 American Wire Gauge) Female/Female Jumper Wires
(1) C-Media Electronics, Inc. CM108/CM109 chipset compatible USB audio adapter
(1) 38KHz NEC code output, 940nm IR Light-Emitting Diode (LED) Remote Control
(1) 802.11b/g/n Edimax EW-7811Un WiFi Module or equivalent with RTl8192cu Chipset
(1) Set of headphones or 5VDC 3W 1A peak 4 Ohm impedance USB powered 3.5mm speakers
(1) Poweradd Pilot S 12000mAh Dual USB Portable Charger External Battery Power Pack

Table 2: Temporary Hardware and Tools

Temporary hardware (needed for setup):	*Required tools (needed for construction):*
HDMI cable	Soldering iron, solder
Ethernet cable	Electric drill, 1/16 drill bit
Corded mouse	Hot glue gun, hot glue sticks
Corded keyboard	Wire cutters, needle-nose pliers
HDMI compatible monitor / TV	Razor blade, rectangular rubber eraser

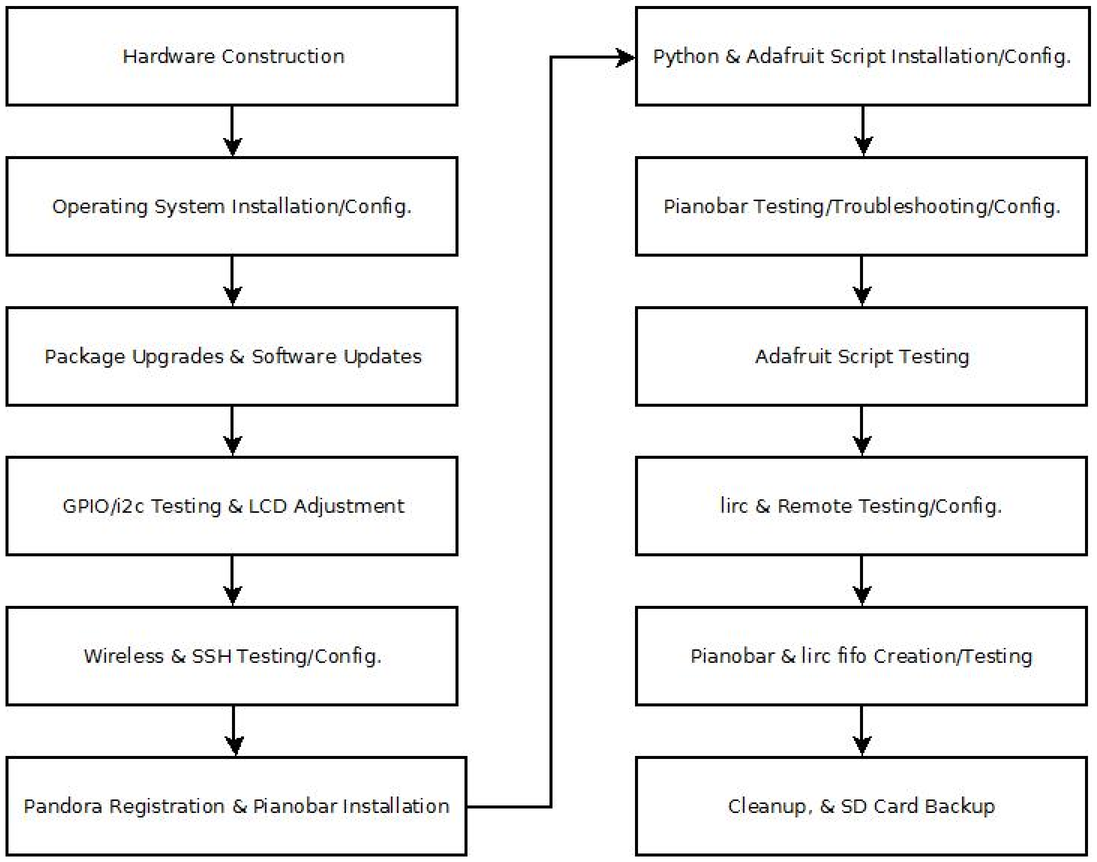
Hardware Construction
Operating System Installation/Config.
Package Upgrades & Software Updates
GPIO/i2c Testing & LCD Adjustment
Wireless & SSH Testing/Config.
Pandora Registration & Pianobar Installation
Python & Adafruit Script Installation/Config.
Pianobar Testing/Troubleshooting/Config.
Adafruit Script Testing
lirc & Remote Testing/Config.
Pianobar & lirc fifo Creation/Testing
Cleanup, & SD Card Backup

Figure 2: Flowchart for all major project steps

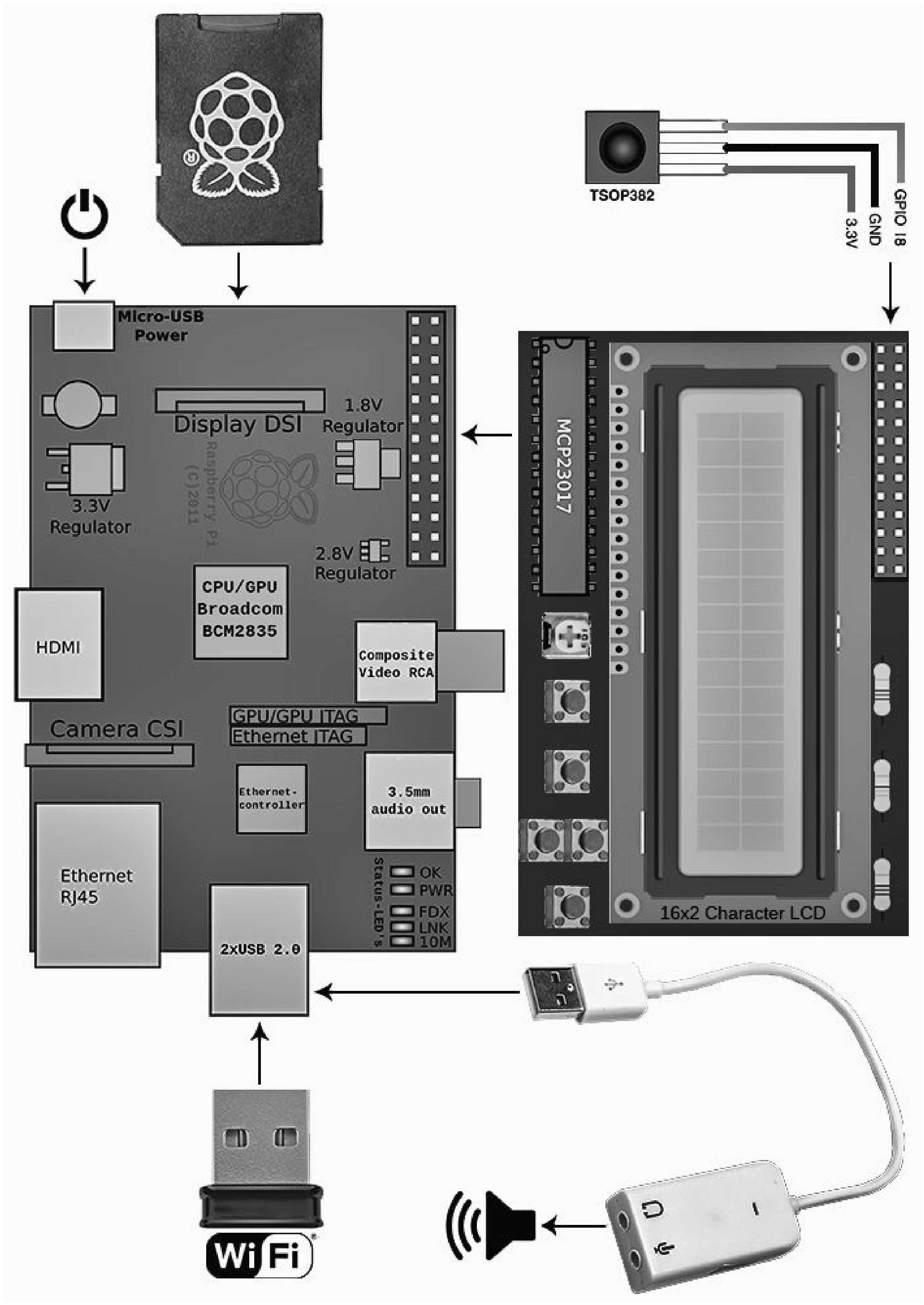

Figure 3: System diagram for all hardware components

Immediately after construction, the PCB should be turned over so that pieces of eraser carefully cut with the razor blade can be hot-glued to its underside in several key areas, as depicted in Figure 4. The single black rubber bumper provided by Adafruit is not enough to stabilize the PCB on the Raspberry Pi and it will move in undesirable ways, bending the GPIO pins and possibly damaging the board itself. The extra effort is worthwhile, as the PCB and Raspberry Pi are much more stable after this modification, as shown in Figure 5. Next, the Raspberry Pi should be enclosed in the PiBow case and case modification using the extra-long screws provided in place of the shorter PiBow ones. Many tutorials exist on online for step-by-step construction of the case.

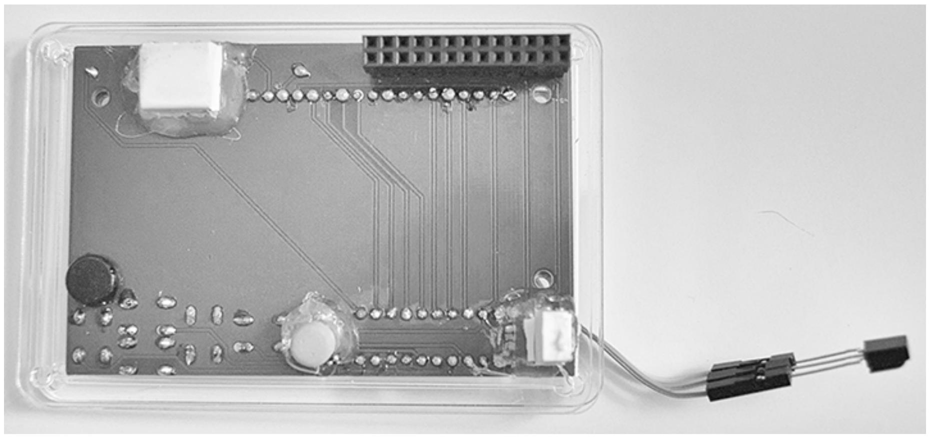
Figure 4: Rubber erasers hot-glued to key stability areas

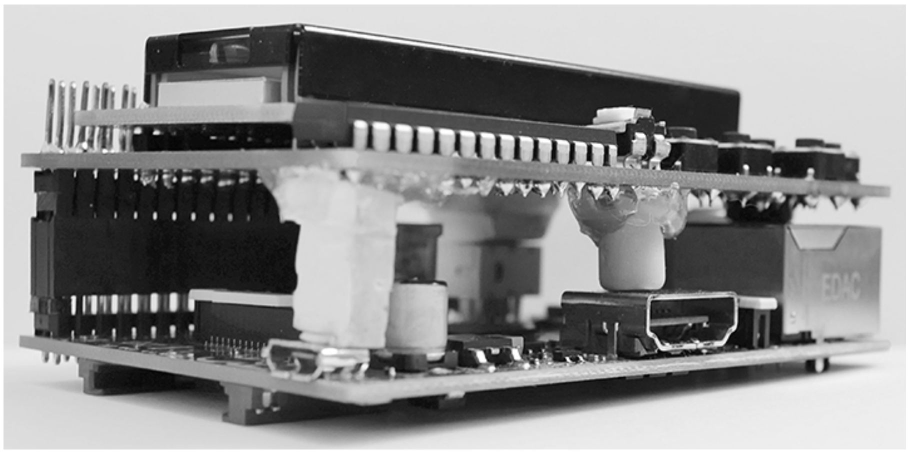

Figure 5: Raspberry Pi and PCB connected with added stability

When the last layer of the case modification is ready to be put into place, the pins of the extra-tall stacking header will contact the underside of the ceramic plate. Holes must be drilled in order to allow all pins to pass through, as this is where the female jumper wires for the IR module will connect to the GPIO pins through the ceramic layer. Without this modification, the case cannot be securely closed. Extraneous plastic around the contacts of the female/female jumper wires should be removed, as they must be as small and cylindrical as possible in order to fit snugly into the drilled holes. These wires will be connecting to pins 1, 6, and 12 (3V3, GND, GPIO18) as shown on the pin-out diagram. The wires should be just big enough to push down and secure onto their respective GPIO pins, and will look like Figure 6. This concludes the hardware construction portion of the project.

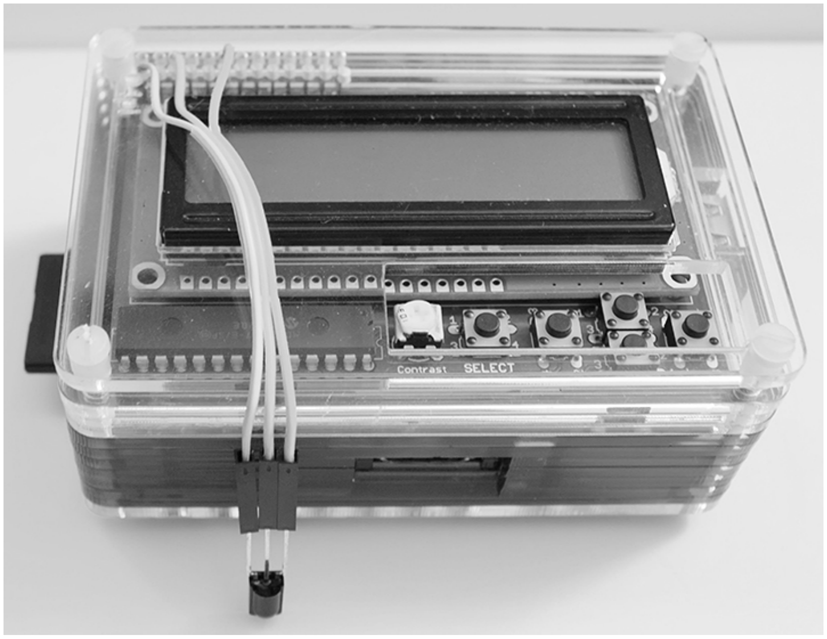

Figure 6: Fully-assembled device in modified PiBow case

The project did not require any programming when moving to the software configuration after assembling the hardware parts. Rather, a beginner's level understanding of how to use the Linux command-line interface was regularly used and could be very helpful. The SD card should now be formatted for the FAT32 (File Allocation Table) file system in preparation of the software configuration. The image for the latest distribution of the New Out Of Box Software (NOOBS)[16] should be burned onto it using the command line on Linux-based systems, or the SD Association's Formatting Tool for Windows and Mac on either of those two systems. NOOBS allows for the convenient installation, reinstallation, and recovery from several OS distributions, including Raspbian Wheezy. Many of the steps in the following overview are covered in the WiFi Radio guide[17] and the IR Remote guide[18] by Adafruit. However, these guides have fewer details, are missing interim steps to combine these projects with our OS, and contain several pieces of inaccurate information.

The Raspberry Pi does not ship with a power switch, so the cables for HDMI out, mouse, keyboard, and network should now all be connected before lastly connecting the MicroUSB power cable. Raspbian Wheezy should be chosen from the NOOBS prompt for the OS. The

Raspberry Pi will automatically restart afterward, and bring up the Raspi-Config tool which should then be updated to the latest version. The file system should then be expanded past 2GB (gigabytes) to make sure all of the SD card is available to the system, and the user password should be changed from the default password of 'raspberry' for security purposes. An appropriate locale, time zone, and keyboard layout should also be chosen, and overscanning should be enabled for HDMI-enabled TVs. It is advisable to change the hostname from 'raspberrypi' to something that is more descriptive, and does not identify the system. Memory split to the GPU should also be changed from the default of 64MB to 16MB, as this is the absolute minimum required for this headless system to display a command prompt. Finally, SSH should be enabled on startup and the Raspberry Pi should be rebooted.

Package lists, installed software, and the OS distribution should all be updated, and then 'ntpdate' should be installed since the Raspberry Pi does not come with a built-in system clock, setting it to an unprivileged port to contact the Ubuntu Network Time Protocol (NTP) server. In case it didn't come loaded with the OS, 'git-core' should be installed and then 'Hexxeh's Raspberry Pi-Update Script'[19] should be downloaded using 'wget' in the command line. The Raspberry Pi should have its firmware updated using this script and then rebooted. Upon returning to the terminal, the 'i2c-bcm2708' and 'i2c-dev' kernel modules should be set to load at boot time by editing the '/etc/modules' file appropriately, before rebooting once more. The 'python-smbus' package should be installed to add System Management Bus (SMBus) support to Python, and the 'i2c-tools' package should be installed to detect the presence of devices attached to GPIO pins. The 'snd-bcm2835' kernel module should be set to load at boot time. After rebooting the connectivity of the LCD screen and IR module can now be tested by using 'i2c-tools' to detect their presence on the system bus. If done correctly, the i2c output should show '20' for the LCD module and 'UU' for the IR module, with all other values being empty or double hyphens.

Python Developer and GPIO tools should now be installed with 'python-dev' and 'python-Raspberry Pi.gpio'. Next, clone into Adafruit's Python code repository and test the python script for the LCD to make sure it is functioning properly by adjusting the potentiometer and testing all the buttons. Wireless networking is then ready to be configured. Only one network connection is needed, but it is possible to connect to multiple networks using Wi-Fi Protected Access (WPA) Supplicant[20] to manage the connections. The '/etc/network/interfaces' file should be edited to contain the appropriate information for configuring a secured or unsecured wireless network. A good tutorial on network setup[21] is provided by Adafruit that can be referred to for a more detailed description of some aspects of this process. Reference this file from the repository for the proper configuration format.

At this point the USB sound card with headphones/speakers should be plugged in, and the Raspberry Pi rebooted. To redirect sound from HDMI to USB as the default, audio options for the system sound should be updated in '/etc/modprobe.d/alsa-base.conf.' Reference this file from the repository for the changes. After plugging in the WiFi adapter and unplugging the ethernet cord, the Raspberry Pi should be rebooted and a speaker test performed to test the sound configuration. To test the wireless connection, 'ping' google.com with a single packet to make sure the device is working properly. The Raspberry Pi now comes pre-loaded with drivers for the adapter being used. If desired, a different adapter can be picked from a list of compatible devices[22] granted that additional steps may be required. From here, record the Raspberry Pi's IP address with 'iwconfig' before shutting it down. It may now be controlled entirely via SSH.

Use another computer to create an account on Pandora.com with a few stations to test on the Raspberry Pi, as this is much simpler than using a command-line only web browser from the Raspberry Pi itself. Boot up the Raspberry Pi and install 'pianobar,'[23] the command-line tool we will use to interface with our LCD and IR module and play audio from Pandora Radio without commercial breaks. Clone into the GitHub repository for Python's 'pexpect' library and install it. Next, clone into Adafruit's Python WiFi Radio Repository, navigate to this directory and make symbolic links to all python files in the 'CharLCDPlate' folder from earlier, then reboot.

To update the username and password for the Pandora account, make a symbolic link between the Pianobar configuration file and the 'WiFi Radio' folder's configuration file, and then edit the Pianobar configuration file appropriately. In order to authenticate your login, copy the Transport Layer Security (TLS) fingerprint[24] from Pandora's tuner server to the bottom of the Pianobar configuration file. Pianobar is now ready to be tested using the appropriate commands. Pandora occasionally updates their TLS fingerprint,[25] and having a TLS fingerprint mismatch in Pianobar's configuration file will give you a 'Network Error: TLS Handshake Failed' error. The first potential solution to this issue is to clone into Pianobar's Git repository[26] and install all required dependencies for compilation. Next, edit the 'Makefile' in the Pianobar directory by choosing the appropriate 'libav' implementation, as documented by the developers. At this time of writing, 'libav10' seems to work best. Pianobar can now be compiled, but may result in a “fatal error: libavfilter/buffersink.h: No such file or directory,” in which case the 'sources.list' file must be edited to add support for Wheezy Backports from Debian Squeeze[27] so that the correct version of 'libav' may be installed. In order for this to take effect, reboot the Raspberry Pi.

After rebooting, update the software packages so that 'libavfilter-dev' will be reinstalled from this new repository, and then perform a clean make and install of Pianobar. However, yet another issue is likely to occur: a “W: GPG error” stating that “The following signatures couldn't be verified because the public key is not available:” referencing the backports release we just added support for. Get the public key from the PGP (Pretty Good Privacy) keys server at the Michigan Institute of Technology and add it to 'apt' to solve this error, so that 'libavfilter' can be successfully installed. After performing another clean make and install, Pianobar should pull the correct TLS fingerprint. Occasionally the TLS handshake error will persist despite a clean install from the source code. This is a problem with how Pianobar handles early termination of the TLS fingerprint validation from Pandora's servers. To solve this, reboot and then clone into a patch for better termination handling[28] for Pianobar. This modifies the source code to correctly perform

TLS handshakes, and requires another clean make and install to work. Pianobar should now execute without error, and may be tested using command line arguments.

A Python script by Adafruit should now be used to successfully integrate Pianobar with the PCB for a visual scrolling display of the artist, song, and album name as they transfer from Pandora to Pianobar. A good tutorial on this is already provided[17] and is largely complete and correct. First, navigate to the WiFi Radio directory containing Adafruit's Python radio script in order to test its functionality with the PCB. Testing all the buttons should reveal they function properly. In order to provide a clean method to shut down to the system, edit the PiPhy script to allow system halt on exit and enable the color on the LCD. Editing the 'rc.local' file on your system allows this script to execute on system startup, and can be referenced in the repository for the necessary changes. Finally, reboot the Raspberry Pi so that Linux IR Remote Control (LIRC) software can be integrated in order to interface with the IR receiver module.

After installing 'lirc,' the '/etc/modules' file must be edited in order to make this module load on boot. Adafruit's tutorial states that the “GPIO in” pin is 23 and the “GPIO out” pin is 22, while in actuality the “GPIO in” pin for the IR module is pin 18, and there is no out pin since it will not be transmitting. The '/etc/lirc/hardware.conf' file must now be edited to load the correct arguments, driver, device, and modules,[29] and the '/etc/lirc/lircd.conf' file must be edited to recognize the IR remote's device-specific signal attributes and assign them to specific key variables for the remote. These three files can all be referenced in the repository for the proper configuration formats.

Use 'mkfifo' in Pianobar's '.config' directory to create a FIFO file for receiving key presses from the IR remote, so that Pianobar interprets them as command-line arguments in the order received. To install 'htop' from another host, use SSH since Pianobar and the PiPhy script are loading on boot. Use 'htop' to terminate their processes temporarily. Test the FIFO file by starting up Pianobar by itself and then sending a valid command-line argument to your FIFO file using 'echo' once it begins playing a station. The letter 'p' serves as a good test because this is used to pause and play the current stream. Upon a successful test, terminate Pianobar once more and create a new '/etc/lirc/lircrc' file for the IR remote. This will send the converted remote key presses to Pianobar as command-line arguments using 'echo.' This file may be referenced in the repository.

An arbitrary number of lines for buttons may be added for the same command so that they may be pressed in quick succession to form a single chained command (e.g. button '1' and '0' for station 10), but regardless of the buttons used they must all be assigned a valid variable as defined in the remote configuration file. After this, update the package lists, installed software, and firmware again in the event an update has taken place or there are issues with LIRC, as software and firmware updates have been released in the past specifically to address timing issues with certain remotes between the Raspberry Pi and LIRC. The Raspberry Pi should be rebooted and the system should be tested for full Pandora and IR functionality. The final step is to edit the '/etc/inittab' file to automatically log the user on to the system on startup.[30] This file may be referenced in the repository for this step.

An optional last step is to back up the SD card for future imaging or recovery. While there are a variety of tools to do this on Windows and Mac systems, the most efficient method is to use a combination of two command-line tools in Linux-based systems: 'gparted' for manipulating the partitions on the SD card to shift all empty space to the end of the partition table, and the 'dd' tool to specify a specific number of megabytes to image from that SD card. The difference in this approach is dramatic, and lets you re-image a much smaller SD card than the original, as this image will only contain the data you need and a minimum of empty space. This can make the difference between a 7.9GB image file and a 16GB image file at the end of this project, assuming you are using a 16GB SD card. This can be further compressed to roughly 2.4GB using a good compression tool such as 7zip.

Results and Discussion

This end-of-semester project was done as a requirement for the computer hardware construction class. The computer hardware construction class is a sophomore/junior level course, and the project is worth 20% of the total grade. This project could possibly be a group project for three to four students, where they will not only learn technical skills but also soft and management skills. Each student could be responsible for one aspect of the project, whether it is software, hardware, or integration. The student(s) had to work independently on the project during their own time, outside of class, where the professor was only overseeing the progress of the project and providing help when necessary. The construction and configuration of this device will be user-specific, assuming the undergraduate student has no prior experience with the device or any of the components involved. Complete prior experience and flawless execution of all project steps, without unforeseen obstacles or bugs, takes approximately 12 hours. For a first-time student completing the project and with proper guidance and instruction, this project should be expected to take between 15 to 25 hours. The total cost for all required hardware is approximately $145 USD excluding shipping, handling, and applicable local taxes, assuming no prior ownership of the Raspberry Pi or other components.

The completed device, depicted in Figure 7, will automatically log the user into the system, test the configured network connections for the first one available, and attempt logging into the configured Pandora account. It will then play uninterrupted, commercial-free internet radio streams from Pandora at a 44.1kHz sampling rate, and could also be configured to play files stored on the SD card itself using the pre-installed 'omxplayer' software with a simple command-line argument. Artist, song and album information as provided by Pandora will be scrolled across the LCD screen and can be viewed all at once in the console along with the time remaining for the current song, updated at one-second intervals. The device is also controllable by IR remote from a distance of roughly fifteen feet, and an arbitrary number of commands can be configured in the appropriate files to control not only Pianobar, but any program on the Raspberry Pi that accepts command-line arguments. Additionally, the device can be safely powered off using this remote in lieu of the button on the device for this purpose.

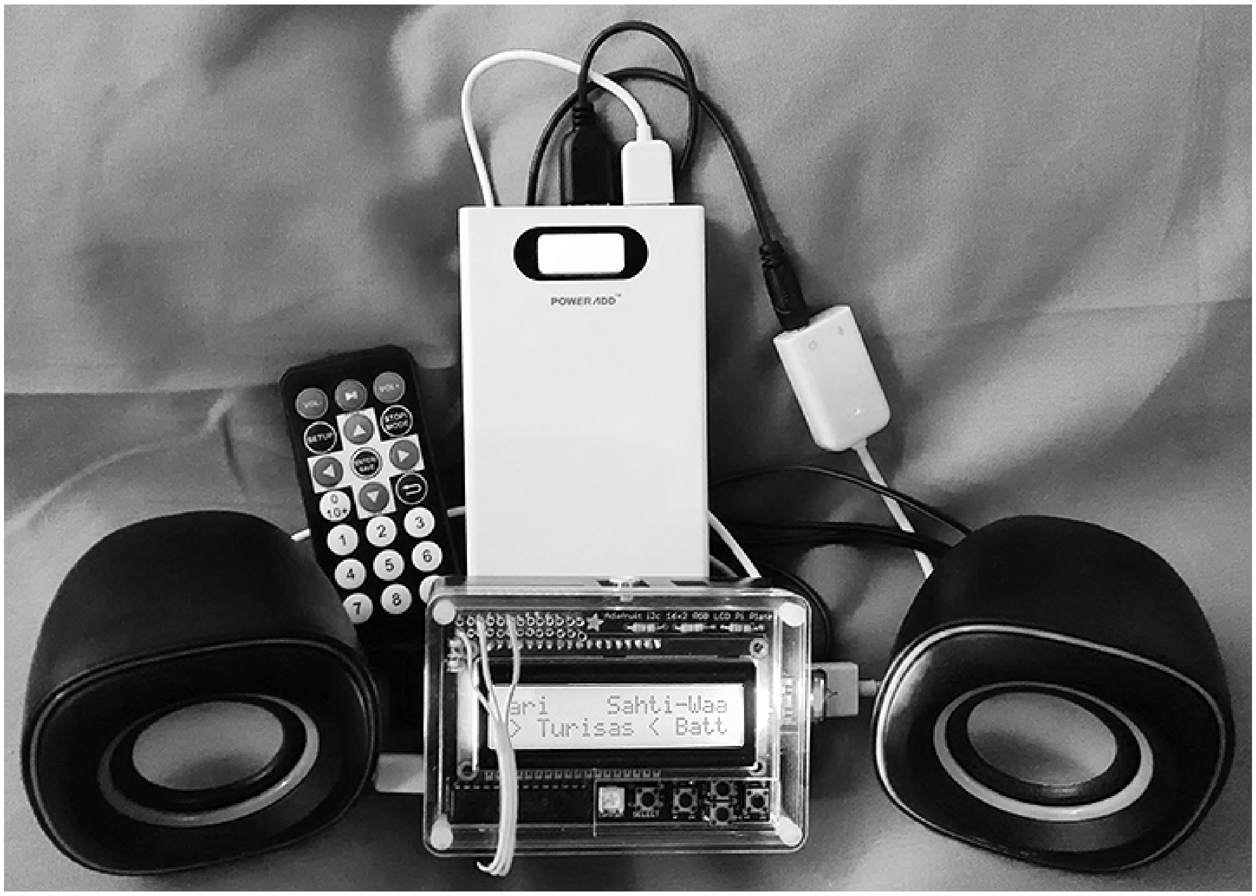

Figure 7: Complete battery-powered project with IR remote, streaming from Pandora

Challenges, Conclusion, and Future Work

While building this project, a lot of references were missing configuration steps, and thus slowed the process down. Troubleshooting these problems was not very easy for the student(s), because of lack of experience and incomplete documentation. Furthermore, coordinating these steps to make a complete project also proved difficult, first because the references used by the student(s) were published online in many unrelated places such as forums and wikis, and second because many of the resources were not originally intended to be used in the context of a comprehensive project. The errors encountered during the configuration were a network error due to a failed TLS Handshake from Pandora, a failure to compile Pianobar because of dependency packages, and a missing GPG (GNU Privacy Guard) encryption key. Resolving these errors also took some additional time.

In addition to being practically useful as a streaming media player, this device can be applied either as an end-of-semester project or for an educational setting to teach students a variety of soft and technical learning skills regarding group management, project documentation, hardware construction, software configuration, Linux OS installation and configuration, kernel modification, synaptic package modification, source compilation, troubleshooting processes, and human-computer interface design. Our experiences with the Raspberry Pi and its online community have underscored the approachability of the platform as well as the support and enthusiasm that exists within the community presence built around it. Because of this, no prior experience with the device or any of the components involved was necessary for implementing this project, enabling students with little or no computer science background to dive in head-first with a knowledgeable support base at their fingertips. Such a project could empower the creative minds of students for further exploration and learning. Raspberry Pi projects in general could be used to attract students to STEM (Science, Technology, Engineering, and Mathematics) learning.

Future work to contribute to this area of research could investigate the use of open-source DJ (Disc Jockey) software for creating custom mixes of audio files and internet radio streams; modifying a long strip of LED lighting into a custom multi-bar equalizer and attaching it to the GPIO pins to brighten and dim the equalizer along with the music; turning the Raspberry Pi into a portable radio receiver/transmitter of analog signals with an additional hardware module, or of digital signals over the internet using a program such as a High Definition Software Defined Radio (HDSDR); and the attachment of a portable solar panel in addition to the battery pack for using the device on camping or educational trips where electrical outlets are unavailable.

Bibliography

[1] Raspberry Pi home. (n.d.). Raspberry Pi Foundation. [Online]. Available: http://www.raspberrypi.org/. Accessed Jan. 6, 2015.
[2] L. Upton. (2013 Apr. 3). Raspberry Pi bringing computing to rural Cameroon. [Online]. Available: http://www.raspberrypi.org/bringing-computing-to-rural-cameroon/
[3] Raspberry Pi Jam. (n.d.). Raspberry Pi Foundation. [Online]. Available: http://www.raspberrypi.org/jam/. Accessed Jan. 6, 2015.
[4] J. Gilreath. (2014, Dec. 19). Reference files for "An Advanced Streaming Internet Radio Player with Raspberry Pi." [Online]. Available: https://github.com/elauzel/RPi
[5] RPi Distributions. (n.d.). eLinux.org. [Online]. Available: http://elinux.org/RPi_Distributions. Accessed Jan. 6, 2015.
[6] Raspberry Pi – Revision 2.0 build options/PCB layout instructions. (2012, Oct. 18). Raspberry Pi Foundation. [Online]. Available: http://www.raspberrypi.org/wp-content/uploads/2012/10/Raspberry-Pi-R2.0-Schematics-Issue2.2_027.pdf
[7] Open source case for business:advocacy. (n.d.). Opensource.org. [Online]. Available: http://opensource.org/advocacy/case_for_business.php. Accessed Jan. 6, 2015.

[8] Raspberry Pi at Southampton. (2013, Oct.). University of Southampton. [Online]. Available: http://www.southampton.ac.uk/~sjc/raspberrypi/
[9] D. Akerman. (2012, Jul. 16). PIE1 – Raspberry Pi sends live images from near space. [Online]. Available: http://www.daveakerman.com/?p=592
[10] W. Powell. (2012, Jul. 22). Project Glass: real time translation … inspired. [Online]. Available: http://www.willpowell.co.uk/blog/2012/07/22/project-glass-real-time-translation-inspired/
[11] M. Kahata. Open brain wave interface hardware. (2014, Jun. 16). [Online]. Available: http://www.instructables.com/id/open-brain-wave-interface-hardware-1/?ALLSTEPS
[12] NetFlix Hulu and other external media sources. (2013, Oct. 14). Raspberry Pi Foundation. [Online]. Available: http://www.raspberrypi.org/forums/viewtopic.php?f=35&t=58224
[13] L. Ada. (2012, Nov. 21). Adafruit 16x2 Character LCD + Keypad for Raspberry Pi. [Online]. Available: https://learn.adafruit.com/adafruit-16x2-character-lcd-plus-keypad-for-raspberry-pi/overview
[14] Pibow case mod for AdaFruit LCD. (2014, Feb. 7). Raspberry Pi Foundation. [Online]. Available: http://www.raspberrypi.org/forums/viewtopic.php?t=68892&p=502066
[15] Stacking header for Raspberry Pi – 2x13 extra tall. (n.d.). Adafruit. [Online]. Available: http://www.adafruit.com/product/1112. Accessed Jan. 6, 2015.
[16] Downloads. (n.d.). Raspberry Pi Foundation. [Online]. Available: http://www.raspberrypi.org/downloads/. Accessed Jan. 6, 2015.
[17] P. Burgess. (2013, Apr. 12). Raspberry Pi WiFi radio. [Online]. Available: https://learn.adafruit.com/pi-wifi-radio/overview
[18] S. Monk. (2013, Mar. 15). Using an IR remote with a Raspberry Pi media center. [Online]. Available: https://learn.adafruit.com/using-an-ir-remote-with-a-raspberry-pi-media-center/overview
[19] A. Bain. (2013, Jan. 3). RaspberryPi Quickstart. [Online]. Available: http://alexba.in/blog/2013/01/04/raspberrypi-quickstart/
[20] RPi edimax EW-7811Un. (n.d.). eLinux.org. [Online]. Available: http://elinux.org/RPi_edimax_EW-7811Un. Accessed Jan. 6, 2015.
[21] S. Monk. (2012, Dec. 10). Adafruit's Raspberry Pi Lesson 3. Network Setup. [Online]. Available: https://learn.adafruit.com/adafruits-raspberry-pi-lesson-3-network-setup/overview
[22] RPi USB Wi-Fi adapters. (n.d.). eLinux.org. [Online]. Available: http://elinux.org/RPi_USB_Wi-Fi_Adapters. Accessed Jan. 6, 2015.
[23] Pianobar. (2009, Apr. 13). 6xq.net. [Online]. Available: http://6xq.net/projects/pianobar/
[24] B. Saska. (2012, Nov. 29). Get Pandora TLS Fingerprint. [Online]. Available: https://gist.github.com/r35krag0th/4173333
[25] T. May. (2013, Sep. 12). Pianobar TLS Handshake woes on Raspberry Pi. [Online]. Available: http://technicaltom.wordpress.com/2013/09/12/pianobar_tls_handshake_fix/
[26] PromyLOPh. (2008, Jun. 10). Console-based pandora.com player. [Online]. Available: https://github.com/PromyLOPh/pianobar
[27] Rachael. (2014, May 2). Fun with Debian: adding wheezy-backports to /etc/apt/sources.list. [Online]. Available: http://funwithdebian.blogspot.com/2014/05/adding-wheezy-backports-to.html
[28] P. Stodghill. (2012, Nov. 15). Patch for pianobar to better handle premature termination. [Online]. Available: https://gist.github.com/pvstodghill/4078695
[29] A. Bain. (2013, Jan. 6). Setting up LIRC on the RaspberryPi. [Online]. Available: http://alexba.in/blog/2013/01/06/setting-up-lirc-on-the-raspberrypi/
[30] Raspbian users policy. (2013, May 17). Raspberry Pi Foundation. [Online]. Available: http://www.raspberrypi.org/forums/viewtopic.php?f=29&t=44082