WI-PLAY : AUDIO STREAMING OVER WI-FI

Amit Gujarathi[1], Saurabh Jagdhane[2], Rohan Gupta[3], Sanket Khedekar[4]

[1]*Department of EXTC, VIT, Mumbai,* ***amit.gujarathi@vit.edu.in***
[2]*Department of EXTC, VIT, Mumbai,* ***saurabh.jagdhane@vit.edu.in***
[3]*Department of EXTC, VIT, Mumbai,* ***rohan.gupta@vit.edu.in***
[4]*Department of EXTC, VIT, Mumbai,* ***sanket.khedekar@vit.edu.in***

Abstract

The goal of the project is to create a system which will broadcast an audio signal over Wi-Fi from computer to set of speakers. This will allow the user to play the audio files from his computer to speakers present in the range of wireless network. The project combines the use of embedded hardware, low level software programming, and the IEEE 802.11 standard protocol for wireless communication (Wi-Fi).

Keywords- Arduino, Cygwin, Microcontroller, Wi-Fi

----------***----------

I.INTRODUCTION

The initial concept for the project stemmed from the idea of wirelessly transmitting audio from a laptop in a room to a central set of speakers. The goal of this project is to achieve this without needing to connect an external dongle to the computer. A number of different options were explored, but it was agreed that the protocol of Wi-Fi made the most sense to use as a communication method since the Wi-Fi shield comes equipped in all modern laptops. After some initial research, it was discovered that no commercial products offer the service of audio over Wi-Fi, with the exception of Apple's proprietary AirPort router. Further research was conducted to determine the most suitable microcontroller to be interfaced with Wi-Shield module. This research discovered a microcontroller board Arduino containing Atmega328P microcontroller suitable to interface with Wi-Fi Shield for a certified connection via 802.11b/g networks.

II. LITERATURE REVIEW

Connectivity plays an important role in today's world. Each devices are becoming wireless to ease the human effort. Wired system has disadvantages because of their complex connection. The concept of using the wireless technology in transmitting or broadcasting the signals gave eureka for developing this project. The conventional technology of transmitting sound wirelessly using Bluetooth is failure because of its short range. Therefore there is need of devices running over Wi-Fi for long range concurrent transmission of audio.

This concept is derived from the Apple Air-Play but they have their own proprieties and is having limited accessibility. The fundamental restriction on this system is that it is proprietary. Even though it can be used with non-Apple brand speakers, it still requires the use of iTunes to stream music. If one wishes to use a different media player, the streaming will not work. The connection between the computer and the speaker is usually a very tedious process.

A device from google is Chromecast which works on the same principles but focuses more on Video Stream. In the end, both Air-Play and Chromecast require an active internet connection. But our system enables user to establish the connection easily.

The market for wireless audio device is estimated to reach $24.52 billion by 2020, at a CAGR of 25.23% from 2014 to 2020. The product market for wireless audio device is segmented into sound bar, wireless speakers system, headphone and microphone, and others which include power amplifier, radio tuner, A/V controller.

Considering all the above situations, development of a system which can transfer audio over Wi-Fi to multiple devices simultaneously without an internet connection with open source content is exciting and can have applications in variety of places like public addressing, music streaming etc.

III. OBJECTIVE

To design a system for robust playback at a minimum size of RAM while sending an audio stream at very consistent rate or audio anomalies would occur. To find a source of fault whether transmitting side (computer program) or microcontroller (receiving end) would be challenging. To design intelligent program on transmitting side which will allow for better user interface and wider array of playback options. The Apple AirPort Express system only supports proprietary apple protocol. Hence we are trying to implement an open source system which will support multiple devices within wireless network.

IV. METHDOLOGY

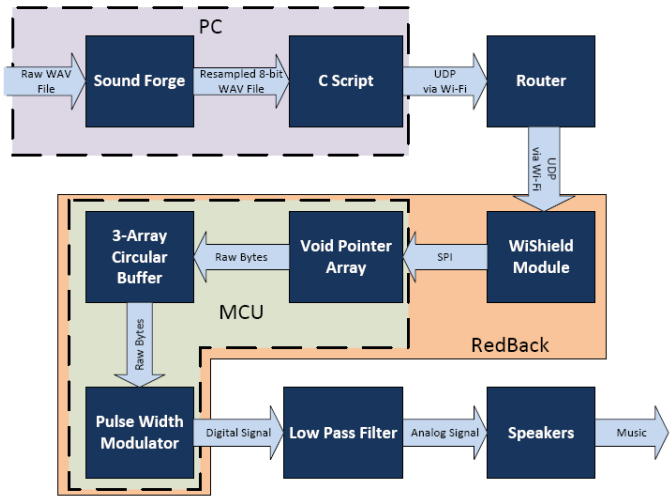
The audio streaming over Wi-Fi consists of different modules for faithful transmission of audio. These modules are categorized as follows:

A. Hardware Interfacing

B. Programming

C. Debugging

The proposed system is created where audio is (sampled) at 22 KHz from Host Computer. The WAV audio is processed into UDP packets of 325 bytes and is routed to Redback via router. The packets are stored in three array buffer cycle. The Microcontroller performs Pulse Width Modulation on received packets. The low pass attenuates unwanted High Frequencies and then it looks virtually like an audio signal. The Redback is wired with speaker from where we get the final output.



V. DISCRIPTION

A. Hardware Interfacing

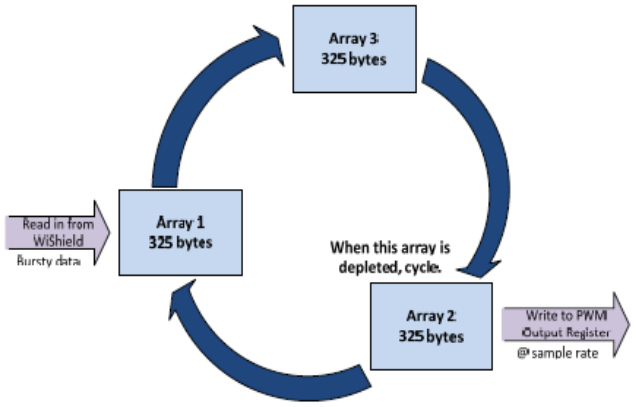
The main hardware of the project is a board called a RedBack. This board includes the microcontroller, Wi-Fi module, and the header pin hookups for I/O signals as well as power. Additionally the board has a few LEDs for power and for Wi-Fi connectivity. This board supports the use of Arduino libraries. The ATMega328p contain 2 KB of RAM. 1 KB of this was utilized in handshaking with the WiShield. This leaves 1 KB to do the buffering required for streaming. This limited buffer size is what ultimately limited the performance of the product. The WiShield connects to a wireless network as any other device would, although requires a static IP address. A Breakout Board called FT232RL USB is required to program the Microcontroller. . A Laptop can work as a Host Computer.

A second board is attached to the RedBack that includes a 5V regulator, a power plug, a few LEDS, a low pass filter, and an audio jack. Finally a router is needed to create a Wi-Fi network.

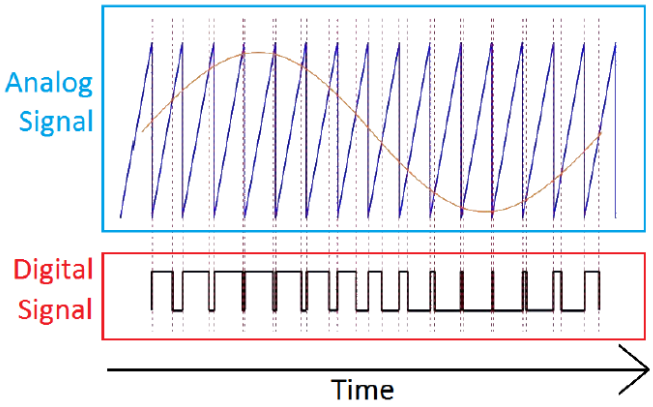
B. Programming

a) Programming the reciever(Redback)

The first step is to program the Redback to connect to the network. The necessary network parameters of the router such as the IP address, default gateway, and subnet mask had to be coded into the program memory of the microcontroller. A simple handshaking program is used to determine the maximum packet size the WiShield could receive. It is determined to be about 325 bytes. For continuous stream, a buffer cycle is developed on the microcontroller. Since there is about 1 KB of RAM available for a buffer, a three array buffer cycle is developed, with each array consisting of three 325-byte arrays. This allows 1 KB of available memory to be filled up without going over the hard packet size limit of 325 bytes. The Arduino program deals with the three 325 byte arrays that hold the audio data. These arrays are first declared as volatile char arrays, since they are read in an ISR and need to be one byte in size. Next the ISR function is specified. This function is called once every sample, or 1/(22 kHz), approximately 45 us. Within this function, one byte out of one of the three arrays is read out and placed into the OCR2B register. This register is responsible for setting the duty cycle of the timer2 PWM. Since this PWM is running at approximately 62 kHz, once this signal gets low passed filtered, it looks virtually like an audio signal. A variable named globalX is responsible for keeping track of which array should be read from and written to. Every 325 runs of the ISR, the globalX variable is updated to a new value to effectively swap the arrays' functions. As the information got depleted from one of the arrays, the next array in the sequence replaced it and the empty array was filled with new data. The following diagram helps to illustrate this process graphically:



Pulse width modulation works by encoding analog values into a digital square wave by changing its duty cycles. Thus the incoming packets are processed by PWM. The ATMega328p produces music by using this technique. Lower analog frequencies are encoded in a much higher digital frequency and then sent through a low pass filter in order to regain the desired signal.



b) *Socket programming on Transmitter*

The final program is based on a simple example given on a wiki created by AsyncLabs (the maker of the RedBack) that was written in C for Linux. Therefore a Linux environment emulator is to be used for windows, called Cygwin is installed, and a GCC compiler is added to the environment. Cygwin is then used to call upon the GCC compiler to compile the c code to then create a native windows executable file.

The Microcontroller end program initializes a bunch of variables and structures used for setting up the network connection with the Arduino RedBacks. This includes selecting the IP address of both of the RedBacks which are static. The communication ports are set up and then the socket is setup and connected using the socket function. Next the computer binds the ports on the PC that the RedBack will communicate back through. In the next section of code, the computer program runs through a large for loop that iterates over the entire WAV file in 325 byte chunks. Within this for loop, a timer is initialized to about 15ms and is used to keep the program on track in case a verification packet doesn't come back from the microcontroller on time. Next, a loop runs that loads 325 bytes into the two arrays responsible for left and right audio. This cleared up the occasional problem of a dropped packet and provided a decently reliable streaming of data. For an 8-bit WAV file. The audio is encoded as one byte left for the left channel and one byte for the right channel.

C. *Debugging*

The main part of the project was debugging as it was time consuming and was difficult to determine and fix the problems in the transmitter and receiver end. For this reason the packets sniffing program called Wireshark was used to find the bug. The packet drops and the connection loss was determined in the Wireshark and a small code was introduced to keep the connection intact.

Many a times there was loss in connection between the router and the Wishield module which was determined on Wireshark graph which introduced a noise in the circuit when connection was lost between the router and the Wishield. There was power surge problem which compelled us to design an additional board to provide a constant 5V supply to the Wishield. Additional LED's were connected to the circuit for debugging show connection between the router and the Shield.

VI. CONCLUSION

The audio was successfully transmitted from the computer end at the rate of 22.05 KHz with 8-bit Stereo. Audio quality was degraded from 16-bit Stereo CD Quality to 8-bit Stereo. Even though the audio was successfully transmitted, there was some problem in receiving side which terminated the connection between the transmitter and the receiver for a small span of time which disconnected the transmission of song producing noise.

REFERENCES

[1] http://linksprite.com/wiki/index.php5?title=Redback_WiFi_Platform_Compatible_With_Arduino_Nano
[2] http://www.cutedigi.com/wireless/wifi/wifi-redback-1-0-arduino-yellowjacket-compatible.html
[3] http://www.cutedigi.com/breakout-board/breakout-board-for-ft232rl-usb-to-ttl-5v.html
[4] https://github.com/linksprite/ZG2100BasedWiFiShield
[5] http://www.gadgetnate.com/2011/07/03/control-leds-over-the-wireless-network-using-arduino-and-wishield
[6] http://www.arduino-hacks.com/arduino-wifi-redback-yellowjacket-compatible/
[7] http://www.arduino-hacks.com/arduino-wifi-redback-webserver/
[8] http://www.arduino-hacks.com/arduino-redback-simple-client/
[9] http://learn.linksprite.com/arduino/shields/how-to-use-redback/
[10] http://forum.linksprite.com/index.php?/topic/311-redback-send-data-to-a-server-running-php/

[11] http://learn.linksprite.com/arduino/shields/redback-and-linker-temperature-experiment/
[12] http://www.rs-online.com/designspark/electronics/eng/knowledge-item/how-to-use redback?/designspark/electronics/knowledge-item/how-to-use-redback=
[13] http://www.rs-online.com/designspark/electronics/eng/knowledge-item/content-1083?/designspark/electronics/knowledge-item/content-1083=
[14] http://forum.arduino.cc/index.php?topic=145313.0
[15] http://postscapes.com/arduino-wifi.