International Journal on Recent and Innovation Trends in Computing and Communication ISSN 2321 – 8169

Volume: 1 Issue: 4 245 – 250

ARDUINO BASED WIFI ENABLED WIRELESS SPEAKER

A.Anbuselvan, D.Antony Shajan, M.Deepak, T.K.Sethuramalingam*

*B.E. (ECE) Final Year, *Assistant Professor, Dept. of Electronics & Communication Engg.*

Sri Venkateswara College of Engineering & Technology, Thirupachur, Thiruvallur, Tamilnadu, India

anbu31may1992@gmail.com, mdeepak35@gmail.com, antonyshajan073@gmail.com, tksethuramalingam@gmail.com

ABSTRACT: This project is to create a system to wirelessly broadcast an audio signal from a computer to a set of speakers using Wi-Fi. This allows one to play music files from a computer and have the sound come out of any speakers that are in range of the wireless network. The ideal use case for this product would involve the ability to have a computer in one room processing music files while multiple speakers throughout the house are actually playing the music. This would be particularly useful in a party setting where one would like to keep a computer safe in a locked room while still being able to use it to play music. Additionally, if a party is there are sets of speakers in multiple rooms, they can all be synced to the same audio source. The major components of the system are the microcontroller receiver module (an ATMega 328p) and the computer program that sends the packetized audio data. This is to create a system that uses Wi-Fi to transmit audio from a source such as a laptop to a speaker system. The final product combines the use of embedded hardware, low level software programming, and the IEEE 802.11 standard protocol for wireless communication (Wi-Fi) to create a polished end device. The hardware and software was developed using a combination of original work as well as open source code and libraries.

Key Words: Arduino, Microcontroller, Wi – Fi, Embedded

*****

I. INTRODUCTION

Embedded systems are designed to do some specific tasks, rather than be a general purpose computer for multiple tasks. Some also have real time performance constraints that must be met, for reasons such as safety and usability; others may have low or no performance requirements, allowing the system hardware to be specified to reduce the cost.

This project was to create a system to wirelessly broadcast an audio signal from a computer to a set of speakers using Wi-Fi. This allows one to play music files from a computer and have the sound come out of any speakers that are in range of the wireless network. The ideal use case for this product would involve the ability to have a computer in one room processing music files while multiple speakers throughout the house are actually playing the music. This would be particularly useful in a party setting where one would like to keep a computer safe in a locked room while still being able to use it to play music. Additionally, if a party is there are sets of speakers in multiple rooms, they can all be synced to the same audio source. The major components of the system are the microcontroller receiver module (an ATMega 328p) and the computer program that sends the packetized audio data

II.ARDUINO MICROCONTROLLER

The Arduino Uno is a microcontroller board based on the ATmega328. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started.

Arduino is the open - source single board microcontroller, designed to make a process of using electronics in multidisciplinary projects more accessible.

The hardware consists of a simple open hardware design for the Arduino board with an Atmel processor and on-board I/O support. The software consists of a standard programming language and the boot loader that runs on the board.

Arduino hardware is programmed using a wring-based language (syntax + libraries), similar

to C++ with some simplifications and modifications, and a processing-based IDE.

The Uno differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it features the Atmega16U2 (Atmega8U2 up to version R2) programmed as a USB-to-serial converter. "Uno" means one in Italian and is named to mark the upcoming release of Arduino 1.0. The Uno and version 1.0 will be the reference versions of Arduino, moving forward.

The Uno is the latest in a series of USB Arduino boards, and the reference model for the Arduino platform.

III.ARDUINO HARDWARE

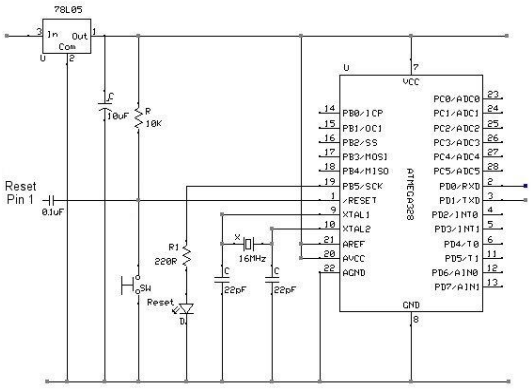

Fig 1 Arduino Uno schematic diagram

An Arduino board consists of an 8-bit Atmel AVR microcontroller with completely components to facilitate programming and incorporation into other circuits. An important aspect of the Arduino is the standard way that connectors are exposed, allowing the CPU board to be connected to a variety of interchangeable add-on modules (known as shields). Official Arduino have used the mega AVR series of chips, specially the ATmega8, ATmega168, ATmega328, and ATmega1280. A handful of other processor has been used by Arduino compatibles.

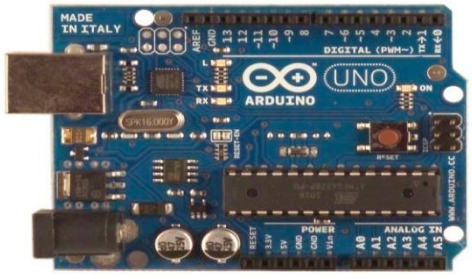

Fig 2 Arduino Uno Hardware

Most boards include a 5 volt linear regulator and a 16 MHZ crystal oscillator (or ceramic resonator in some variants), although some designs such as the Lily pad run at 8 MHZ and dispense with the onboard voltage regulator due to specific form-factor restrictions. An Arduino microcontroller is also pre-programmed with a boot loader that simplifies uploading of programs to the on-chip flash memory, compared with other devices that typically need an external chip programmer.

At a conceptual level, when using the Arduino software stack, all boards are programmed over an RS-232 serial connection, but the way this is implemented varies by hardware version. Serial Arduino boards contain a simple inverter circuit to convert between RS-232 level and TTL level signals. Current Arduino boards are programmed via USB, implemented using USB-to-serial adapter chips such as the FTDI FT232. Some variants, such as the Arduino mini and the unofficial board uno, use a detectable USB-to-serial adapter board or cable, Bluetooth or other methods. (When used with traditional microcontroller tools instead of the Arduino IDE, standard AVR ISP programming is used.) The Arduino board exposes most of the microcontroller's I/O pins for use by other circuits.

IV. FEATURES OF ATMEGA328

The controller consists of 8K bytes of In-System Programmable Flash with Read-While-Write capabilities, 512 bytes of EEPROM, 1K byte of SRAM, 23 general purpose I/O lines, 32 general purpose working registers, three flexible Timer/Counters with compare modes, internal and external interrupts, a serial programmable USART, a byte oriented Two-wire Serial Interface, a 6-channel ADC (eight channels in TQFP and QFN/MLF packages) with 10-bit accuracy, a programmable Watchdog Timer with Internal Oscillator, an SPI serial port, and five software selectable power saving modes. The Idle mode stops the CPU while allowing the SRAM, Timer/Counters, SPI port, and interrupt system to continue functioning.

The Power down mode saves the register contents but freezes the Oscillator, disabling all other chip functions until the next Interrupt or Hardware Reset. In Power-save mode, the

asynchronous timer continues to run, allowing the user to maintain a timer base while the rest of the device is sleeping. The ADC Noise Reduction mode stops the CPU and all I/O modules except asynchronous timer and ADC, to minimize switching noise during ADC conversions. In Standby mode, the crystal/resonator Oscillator is running while the rest of the device is sleeping. This allows very fast start-up combined with low-power consumption.

The device is manufactured using Atmel's high density non-volatile memory technology. The Flash Program memory can be reprogrammed In-System through an SPI serial interface, by a conventional non-volatile memory programmer, or by an On-chip boot program running on the AVR core. The boot program can use any interface to download the application program in the Application Flash memory. Software in the Boot Flash Section will continue to run while the Application Flash Section is updated, providing true Read-While-Write operation. By combining an 8-bit RISC CPU with In-System Self-Programmable Flash on a monolithic chip, the Atmel ATMEGA328 is a powerful microcontroller that provides a highly-flexible and cost-effective solution to many embedded control applications.

The ATMEGA328 AVR is supported with a full suite of program and system development tools, including C compilers, macro assemblers, program debugger/simulators, In-Circuit Emulators, and evaluation kits.

This setup has other implications. When the Uno is connected to either a computer running Mac OS X or LINUX, it resets each time a connection is made to it from software (via USB). For the following half-second or so, the boot loader is running on the Uno. While it is programmed to ignore malformed data (i.e. anything besides an upload of new code), it will intercept the first few bytes of data receives one-time configuration or other data when it first starts, make sure that the software with which it communicates waits a second after opening the connection and before sending this data.

The Uno contains a trace that can be cut to disable the auto-reset. The pads on either side of the trace can be soldered together to re-enable it. It's labeled "RESET-EN". You may also able to disable the auto-reset by connecting a 110 ohm resistor from 5V to the reset line.

V. THE PROPOSED SYSTEM

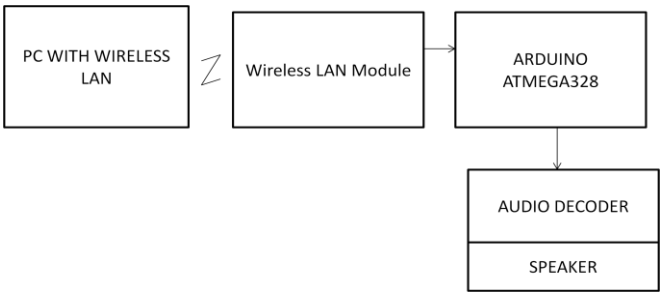

Fig 3 Block Diagram of the Proposed System

The pc with wireless LAN is used to transmit the audio over 802.11 protocols. The wireless LAN module receives the data being transmitted. The received data is passed to the arduino controller. The arduino controller passes the data to the speaker through the audio codec present in the speaker, used to get a specified audio format.

VI. Wi- Fi MODULE

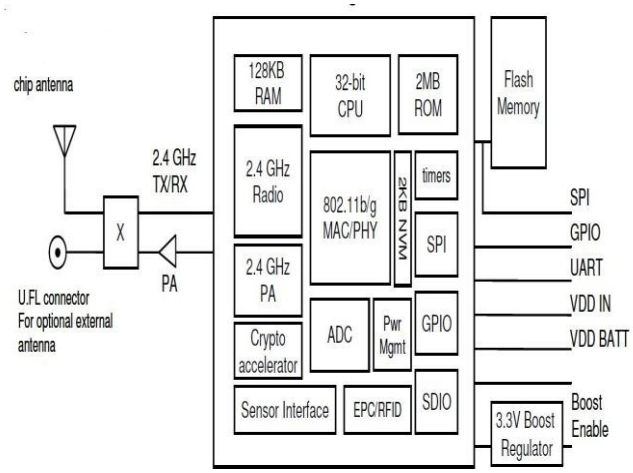

Fig 4 Block Diagram of the Wi – Fi Module

The Wi-fly GSX module is a stand alone, embedded wireless 802.11 networking module. Because of its small form factor and extremely low power consumption, the RN-131G is perfect for mobile wireless applications such as asset monitoring, GPS tracking and battery sensors. The Wi-Fly GSX module incorporates a 2.4GHz radio, processor, TCP/IP stack, real-time clock, crypto accelerator, power management and analog sensor

interfaces. This complete solution is preloaded with software to simplify integration and minimizes development of your application. In the simplest configuration the hardware only requires four connections (PWR, TX, RX, GND) to create a wireless data connection. Additionally, the sensor interface provides temperature, audio, motion, acceleration and other analog data without requiring additional hardware. The Wi-Fly GSX module is programmed and controlled with a simple ASCII command language. Once the Wi-Fly GSX is setup it can scan to find an access point, associate, authenticate and connect over any Wi-fi network.

VII. AUDIO CODEC

VS1053b is a single-chip Ogg Vorbis/MP3/AAC/-WMA/MIDI audio decoder and an IMA ADPCM and user-loadable Ogg Vorbis encoder. It contains a high-performance, proprietary low-power DSP processor core VS DSP4, working data memory, 16 KB instruction RAM and 0.5+ KB data RAM for user applications running simultaneously with any built-in decoder, serial control and input data interfaces, upto 8 general purpose I/O pins, an UART, as well as a high-quality variable-sample rate stereo ADC (mic, line, line + mic or 2×line) and stereo DAC, followed by an earphone amplifier and a common voltage buffer. VS1053b receives its input bit stream through a serial input bus, which it listens to as a system slave. The input stream is decoded and passed through a digital volume control to an 18-bit oversampling, multi-bit, sigma-delta DAC. The decoding is controlled via a serial control bus. In addition to the basic decoding, it is possible to add application specific features, like DSP effects, to the user RAM memory. Optional factory-programmable unique chip ID provides basis for digital rights management or unit identification features.

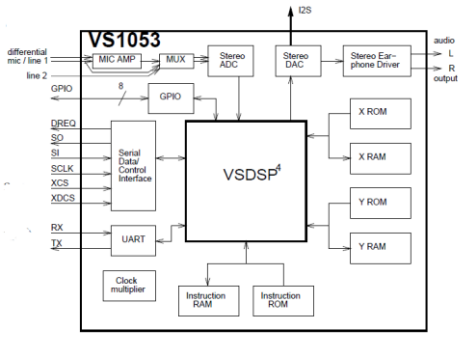

Fig 5 Block Diagram of the Audio Codec System

VIII. ARDUINO DEVELOPMENT ENVIRONMENT

The Arduino development environment contains a text editor for writing code, a message area, a text console, a toolbar with buttons for common functions, and a series of menus. It connects to the Arduino hardware to upload programs and communicate with them.

LANGUAGE EDITOR:

The Arduino 1.0.1 software environment has been translated into 30+ different languages. By default, the IDE loads in the language selected by your operating system. If you would like to change the language manually, start the Arduino software and open the Preferences window. Next to the Editor Language there is a dropdown menu of currently supported languages. Select your preferred language from the menu, and restart the software to use the selected language. If your preferred language is not supported, the IDE will default to English.

You can return Arduino to its default setting of selecting its language based on your operating system by selecting System Default from the Editor Language drop-down. This setting will take effect when you restart the Arduino software. Similarly, after changing your operating system's settings, you must restart the Arduino software to update it to the new default language.

WRITING SKETCHES:

Software written using Arduino are called sketches. These sketches are written in the text editor. Sketches are saved with the file extension .ino. It has features for cutting/pasting and for searching/replacing text. The message area gives feedback while saving and exporting and also displays errors. The console displays text output by the Arduino environment including complete error messages and other information. The bottom righthand corner of the window displays the current board and serial port. The toolbar buttons allow

you to verify and upload programs, create, open, and save sketches, and open the serial monitor.

SKETCHBOOK:

The Arduino environment uses the concept of a sketchbook - a standard place to store your programs (or sketches). The sketches in your sketchbook can be opened from the File > Sketchbook menu or from the Open button on the toolbar. The first time you run the Arduino software, it will automatically create a directory for your sketchbook. You can view or change the location of the sketchbook location from with the Preferences dialog.

LIBRARIES:

Libraries provide extra functionality for use in sketches, e.g. working with hardware or manipulating data. To use a library in a sketch, select it from the Sketch > Import Library menu. This will insert one or more #include statements at the top of the sketch and compile the library with your sketch. Because libraries are uploaded to the board with your sketch, they increase the amount of space it takes up. If a sketch no longer needs a library, simply delete its #include statements from the top of your code.

CONCLUSION

The end result from this project ended up working amazingly well considering the hard limitations in the hardware. The audio quality was very decent considering the small buffer limitation of the microcontroller being used. Furthermore the system was relatively robust and could recover from seconds of lost packets (although the audio signal would be lost for this period of time).

Although the outcome of the project was very impressive, there are a number of things that could be improved in this project. One of the largest improvements would have been to use a microcontroller with more onboard RAM, or add external RAM. This would have allowed the audio buffer to be longer, thus decreasing the receiver's sensitivity to untimely packets and other unexpected networking anomalies. Another possible improvement is the code on the computer end of the project. The user interface could have been drastically improved to allow for easier playback options. Furthermore, support for files other than uncompressed WAV could have been added. This could have even been taken a step further, and a virtual audio driver could have been created that would have streamed any audio being produced on the computer to the microcontroller receiver. This would have allowed for any audio program on the computer to be used.

A number of things have been learned throughout the course of this project. One of the most important lessons learned is that lots of research pays off in the end. More research should've been done in regards to the type of microcontroller being used as well as its amount of on board RAM. This being said, the RedBack microcontroller and Wi - Fi module was not inherently compatible with other microcontrollers. Another lesson learned was that one should not be afraid to rewrite code in more usable languages. This lesson applies directly to the use of Linux based C code on the computer end of the project. This code most likely could have been adapted for Java, which would have allowed for much better user input controls as well as possible support for more complex future additions such as MP3 decoding or integration with a virtual sound card driver.

REFERENCES

1. B. Rose , "Home Networks: a Standards Perspective",IEEE Communications Magazine, Dec. 2011, pp. 78-85.
2. T. Zahariadis, and K. Pramataris, "Multimedia Home Networks: Standards and Interfaces", Computer Standards & Interfaces, Vol. 24, Issue 5, Nov. 2009, pp. 425-435.
3. T. Saito, I. Tomoda, Y. Takabatake, K. Teramoto, and K. Fujimoto, "Wireless Gateway for Wireless Home AV Network and Its Implementation", IEEE Transactions on Consumer Electronics, Vol. 47, No, 3, Aug. 2011, pp.496-501.
4. G. Bai, and C. Williamson, "The Effects of Mobility onWireless Media Streaming Performance", Proc. IASTED Intl. Conf. on Wireless Networks and Emerging Technologies (WNET), Banff, Canada, 2009, pp. 596-601.

5. T. Kuang, and C. Williamson, "RealMedia Streaming Performance on an IEEE 802.11b Wireless LAN", Proceedings of IASTED Wireless and Optical Communications (WOC), 2012, pp. 306–311.
6. H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson, RTP: A Transport Protocol for Real-Time Applications, RFC2003,Jan. 2011.
7. H. Schulzrinne, RTP Profile for Audio and Video Conferences with Minimal Control, RFC 2007, Jan. 2010.
8. D. Hoffman, G. Fernando, V. Goyal, and M. Civanlar, RTP Payload Format for MPEG1/MPEG2 Video, RFC 2250, Jan. 2012.
9. Michael Morgolis - Arduino cookbook - II Edition 2012, O'Reilly Publications
10. http://arduino.cc/en/Main/ArduinoBoardUno
11. Tero karvinen & kimmo Karvinen - **Make a mind controlled Arduino robot** - I Edition 2011, O'Reilly Publications
12. Massimo Banzi – **Getting Started with Arduino** - II Edition 2012, O'Reilly Publications
13. Emily Gertz & Patrick Di Justo – **Environmental Monitoring with Arduino** - I Edition 2012, O'Reilly Publications
14. Alasdair Allan - **iOS sensor Apps with Arduino** - II Edition 2012, O'Reilly Publications.

A. Anbu Selvan, This author was born in Tamil nadu, in 1992 and pursuing B.E in Electronics & Communication Engg. Currently he pursues his project under the guidance of T. K. Sethuramalingam. His fields of interest are embedded systems and Networking.

D. Antony shajan, This author was born in Tamil nadu, in 1992 and pursuing B.E in Electronics and Communication Engg. Currently he pursues his project under the guidance of T. K. Sethuramalingam. His fields of interest are embedded systems and Networking.

M. Deepak, This author was born in Tamil nadu, in 1992 and pursuing B.E in Electronics and Communication Engineering. Currently he pursues his project under the guidance of T. K. Sethuramalingam. His fields of interests are embedded systems and java.

T.K.Sethuramalingam, This author was born in Tamil Nadu, India, in 1981 and received the UG. degree from the Manonmaniam Sundaranar University, India, in 2001. Further he received his PG. degree from Bharathidasan University, India, in 2003. He is currently doing his Ph.D in MEMS. He is working as an assistant professor in Sri Venkateswara college of Engineering & Technology, Chennai India. He visited foreign countries and presented his research publications. He is a Member in IEEE, ISSS, IACSIT, IETE, ISTE.His research interests and publications have been in the areas of Embedded systems, VLSI design and MEMS.