Poster Abstract: Audio

Kranthi Kumar Badagu, Arijeet Swapankumar Laga, Mithun Peddakalva
Uppsala University, Sweden

***ABSTRACT* - The project is to send audio signals wirelessly between two micro-controllers. The microphone acts as an input device that provides the required audio signal to one of the microcontrollers. This microcontroller converts the analog audio signal into a digital signal and sends it to the other microcontroller which in turn regenerates the audio signal (here speaker is the output device). This wireless communication takes place using nrf24l01+.**

I. INTRODUCTION

There are many ways of sending the audio signals wirelessly, for example, Bluetooth, FM Radio, Zigbee while we opted to use the nrf24l01+ wireless transceiver module which is customizable and operates on a frequency of 2.4 GHz and is suitable for short-range communication.

There are many features of using nrf24l01+, it can communicate with multiple nodes at a time, it consumes very low power. The signal power, data rate, packet size and node address, all are configurable which makes it an ideal solution for experimenting with various designs and techniques.

II. DESIGN OF SYSTEM

There are two Arduino/ATmega328P for data processing and communication. A microphone is connected to one of the Arduinos and the speaker is connected to the other Arduino. The output of the microphone is analog which is converted to digital by the microcontroller. The challenging part was this conversion where we have two ADC resolutions available, 8-bit and 10-bit. For the conversion, the sampling rate is a very important aspect.

For using the ADC of ATmega328P, we first set the reference voltage, in our case it is 5V. Then we choose the ADC channel out of the total 6 available channels, here we are using the ADC0 channel. We need to set the prescaler division factor for the ADC since the sampling rate increases as the prescaler division factor is reduced.

Sampling Rate = (Clock Frequency / Prescaler Division Factor) / Clock Cycle

The sampling frequency should be twice the frequency under test. We maintained the standard sampling frequency of 44100 Hz. For obtaining this value, we set the Prescaler Division Factor to 16. In the 8-bit resolution, we get 32 samples at a fast rate whereas in the 10-bit resolution we get 25 samples but with better quality.

After configuring the ADC, we configure the Timer and begin the conversion. The ADC interrupt indicates that the conversion is complete and we read the result which is stored in Lower Data Register for 8-bit resolution whereas 10-bit resolution occupies both Lower and Higher Data Register.

For initializing the communication, we need to set the channel frequency, RF power, data rate, and the addresses. With this, we also specify if it is a broadcast, multicast or unicast. Depending on the behavior, we can choose either receiving mode or transmitting mode.

The transmitter and the receiver both have unique addresses of 5 bytes that are known to each other. We can choose 125 channels with 1 MHz spacing providing a frequency selection from 2.4 GHz to 2.527 GHz. Each channel can have up to 6 addresses which allow the device to communicate with 6 different devices at the same time. The nrf24l01+ supports three data rates that one can use which is 250kbps, 1 Mbps and 2 Mbps. The channel occupies a bandwidth of less than 1 MHz at 250 kbps and 1 Mbps, and less than 2 MHz at 2 Mbps.

We have maintained one-way communication. So, the receiver radios keeps listening for packets from the desired transmitter. When the conversion is complete and the interrupt arises, the resultant data is stored in a buffer. The chip is enabled and a transmission interrupt sends/transmits the samples when the buffer is filled and if the buffer is ready to be sent then get the buffer address before allowing the nested interrupts or else we disable this interrupt vector and allow other interrupts to utilize this vector line (nested interrupts) and if the buffer is empty then we re-enable the interrupt vector.

We have again used a buffer to store the received signal (payload) at the receiver. Similar to the interrupt in transmission, there is a reception interrupt that disables other interrupts while listening and once the buffer is filled, enable the other interrupts to utilize the vector line (nested interrupts). The data in the buffer is loaded into a Timer which generates the PWM signal. This signal is then fed to the output speaker through an amplifier and a RC filter. And this results into audio as output.

III. RESULTS

The communication was not totally error-free, but the output was clearly audible with some amount of noise present. We did observe the effect on the output audio sound when the sampling rate is changed. By varying the data rate, we observed the inclusion of noise with increasing distance between the transceivers. Better RF power consumes more energy but gives good results.

Also, we didn’t observe any large delay in signal reception. We tested the system in open space where it had a good communication range, but when tested in a closed room with objects like walls in between, the communication broke very often.

IV. CONCLUSION

It is possible to successfully convert the audio signal into digital and transmit it wirelessly using an 8-bit microcontroller and a wireless transceiver module. For this, the proper configuration of the ADC and the wireless transceiver module (nrf24l01+) is very crucial.

V. REFERENCES

1. https://tmrh20.github.io/RF24Audio/index.html
2. https://github.com/nRF24/RF24Audio
3. https://cassiopeia.hk/walkietalkie/
4. https://www.instructables.com/id/NRF24L01-Wireless-Transmission-Between-Arduino/