EE318 Electronic Design Lab Project Report, EE Dept, IIT Bombay, April 2007

Secure Wireless Transmission of Data via Encryption and Frequency Hopping

Group No: B09
Sudeep Kamath (04007019) <sudeep@ee.iitb.ac.in>
Kartik Mohta (04007023) <kartikmohta@ee.iitb.ac.in>
Rohan Raj Desu (04007036) <rohanraj@ee.iitb.ac.in>
Anil Krishna N. (04007037) <anilkris@ee.iitb.ac.in>
Course Instructors:
Prof. H. Narayanan
Prof. Dinesh K. Sharma
Prof. Madhu N. Bellur
Prof. Mukul C. Chandorkar

Abstract
The project aims at developing a medium of wireless communication between two nodes, the communication being made secure by encryption and frequency hopping.

1. Introduction

The project aims to establish one-way wireless communication between two nodes. The issue of security of the communication is addressed by encryption and frequency hopping. Specifically, we look at voice transmission wherein the voice signal is sampled at a reasonable rate and the quantized samples are transmitted in digital format. At the receiver end, the signal is reconstructed from the samples.

Voice transmission over a wireless channel is an important challenge in view of the high data rate required. Wireless transfer of ordinary data, in general, does not require a high bit rate and may be achieved easily. Real-time voice transmission, on the other hand, necessitates that the rate at which bits are being generated, be equal to the rate at which they are being transmitted. Unless speech coding algorithms are used, a data rate of 32 kilobits/sec or 4kilobytes/sec is a reasonable requirement.

Our project has applications to wireless communication where security is an important necessity. Conventionally, wireless systems' claim to security is solely the encryption of data. The encrypted data stream is transmitted over the wireless channel. A possible adversary who can “hear” the data stream still has the capability to retrieve the individual bits of the encrypted stream. Often, algebraic attacks may be employed to decipher the data. When security breach is a major issue, there is a need for stronger encryption. We attempt to maintain greater security of data via the following:

- We encrypt the data by simple modulo-2 addition ($\oplus$: XOR addition) with a fixed random string.
- We attempt to implement frequency hopping in a cyclic fashion. The carrier frequency is changed in a manner, that is known only to the transmitter and the

receiver. Thus, in this case, the possible adversary would not be able to “hear” more than a certain fraction of the input data stream at intermittent intervals of time.

Thus, real-time LFSR encryption along with frequency hopping results in a secure communication link.

2. Design approach and choice of components

We establish one-way wireless voice communication between two nodes. The choice of components for the implementation was based on their properties and availability, as well as cost.

- **Wireless Module used:** We use the CYWUSB6935 (WirelessUSB(TM) LR 2.4GHz DSSS Radio SoC) chip (2 nos) which constitute a 2.4GHz radio transceiver pair. The reason for choice of this chip is primarily the ease of interfacing with a microcontroller. This chip also allows the frequency to be changed by changing the contents of a register within the chip.
- **Microcontroller used:** We use the ATmega16L microcontroller as an interface between the analog and digital components for both the transmitter and receiver side. The reasons for the choice of this microcontroller are:
  - In-built Analog to Digital Converter(ADC)
  - Serial Peripheral Interface available
  - Working voltage range includes 3.3V which is the same as the voltage required for the wireless module.

One negative point about this choice of microcontroller is the availability of four ports, not all of which are required for this application.

3. Design of circuit

The circuit design can be broken down into independent working modules.
The modules that form the basic blocks of our circuit are as follows:

a) **Conversion of sound input into electrical signal**

A microphone is installed at the input. The output of the microphone, after being fed to a second-order active low pass filter, to filter out high-frequency noise, is provided a suitable DC offset so that the signal, for the required range of sound input amplitude at the microphone, yields an output voltage value that fits into the input range of the A/D converter (0V-3.3V).

b) **Conversion into digital format**

The ATmega16L microcontroller is equipped with an in-built 8-bit analog to digital converter. We sample the signal at a rate of 2 kilosamples/sec, taking 8 bits per sample.

c) **Encryption of digital data signal**
The encryption of the sampled digital data is carried out in software by modulo-2 addition of the input data with a constant byte string. This string is chosen (quite arbitrarily) to be 95 in the hexadecimal notation, or 1001 0101 in the binary notation. The encryption key is known beforehand, to the transmitter and the receiver only, and is therefore, hidden from the possible adversary.
Encryption via modulo-2 addition with a fixed string has two major advantages:

- **A good encryption scheme for real-time encryption:** Real-time encryption of digital data necessitates availability of only a finite number of not-yet-transmitted samples. Indeed, if this number is large, it will result in excessive memory requirements at both the transmitter and receiver side in addition to producing delay due to both the processing in software required for encryption as well as the delay corresponding to the wait-period for transmitting blocks of data. Also, standard encryption methods such as RSA encryption require a large number of stored samples, to be effective enough, as, for a relatively small number of samples, RSA may be regarded as a simple look-up table. In addition, implementation of this is cumbersome, from the point of view of reproduction of a signal such as a voice signal that requires a constant rate of arrival (and departure) of samples. A trivial look-up table for every data input byte is a reasonable encryption scheme for real-time applications.
- **Easy to implement (in software or hardware):** A look-up table would cost memory. The look-up table may be simplified further by usage of XOR with a constant pre-decided byte string to generate the encrypted string. This makes encoding and decoding simple, as both require the same operation to be performed.

d) **Transmission of signal**
The encrypted data byte is sent to the wireless module by using the Serial Peripheral Interface. The chip transmits the data byte by using Direct Spectrum Spread Sequences (DSSS) and provides an interrupt to the microcontroller

e) **Additional security due to frequency hopping**
The transmitter and receiver periodically change the frequency of transmission and reception in a pattern that is hard-coded in both of them and hence, known to them beforehand. This stage could **NOT** be implemented because of issues discussed in the “” section. Frequency hopping could not be performed.

f) **Reception of signal**
The signal is received on the receiver chip which provides an interrupt to the microcontroller indicating it to read the received byte.

g) **Decryption of the digital encrypted signal**
The receiver keeps generating the random strings by the same procedure as the transmitter. As the encryption procedure is known to both, the encrypted strings received is easily decrypted. Decryption is carried by exactly the same process: modulo-2 addition with the random string.

h) **Reconstructing the transmitted audio signal**
The decrypted samples correspond to the data samples at the transmitter end. These sampled values are fed to a digital to analog converter. The output of the DAC is filtered and finally provided to a speaker which reproduces the input voice signal. We note that the output fails to resemble the exact input because of the following:

- Quantization
- Strictly band-limited nature not applicable to voice signal
- Sample-and-hold kind of reproduction at the output

We discuss in the "Tests" and "Test Results" section the experimentally observed conditions, under which the distortion due to the above are found to be negligible.

In our case, however, voice input at the transmitter end is to a large extent, unrecognisable at the receiver end because of the low sampling rate, which itself comes about due to constraint in data rate. However, the duration of the words is recognised correctly. Morse code communication through voice, for instance, can be carried out.

i) **Frequency hopping scheme**
The frequency range for transreception can be varied in the range of 2.4GHz to 2.483GHz with 78 different possible values in between the two.
We had intended to employ a simple hopping scheme wherein after every 20 samples, the frequency is chosen to be the next in the set of four frequencies arranged in cyclic order. $\{f_1=2.4GHz, f_2, f_3, f_4=2.483GHz\}$ This could not be implemented.

j) **Synchronisation Procedure**
For frequency hopping to be carried out, synchronisation becomes a very important requirement. The synchronisation method we devised was to send a known byte, say AA in hexadecimal, to the receiver after every 20 samples. This may act as a signal to the receiver to change to the next frequency. As frequency hopping could not be implemented, this is not done either.

4. Diagrams

Block diagrams:

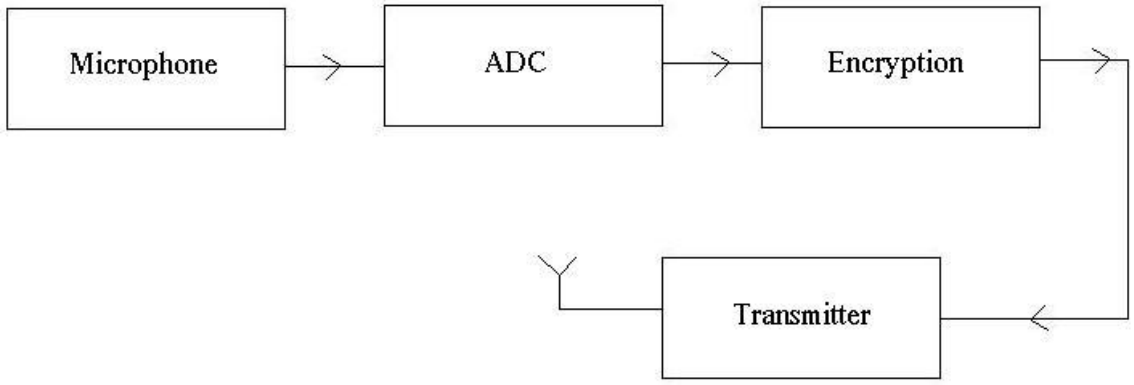

Fig 1: Transmitter side block diagram

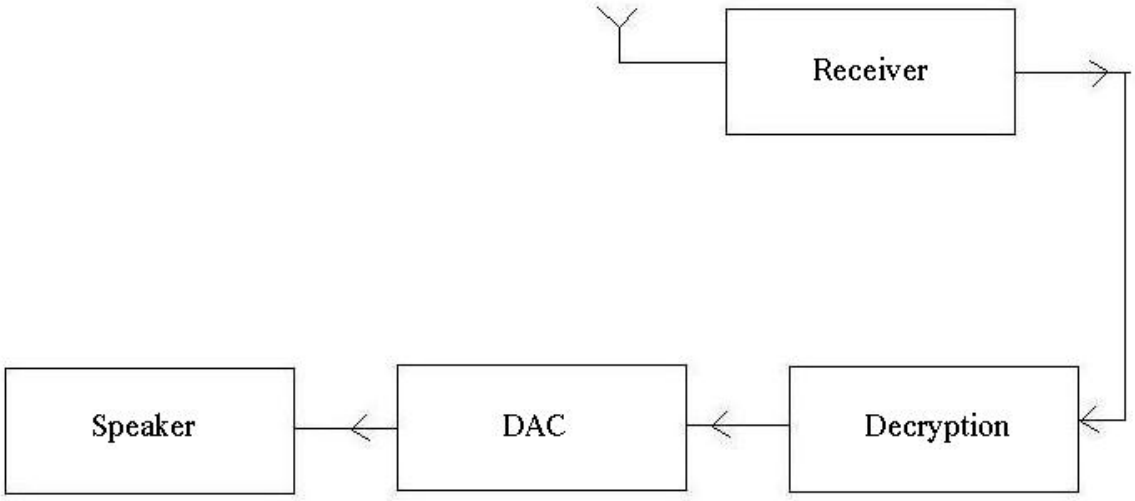

Fig 2: Receiver side block diagram

Circuit diagrams:

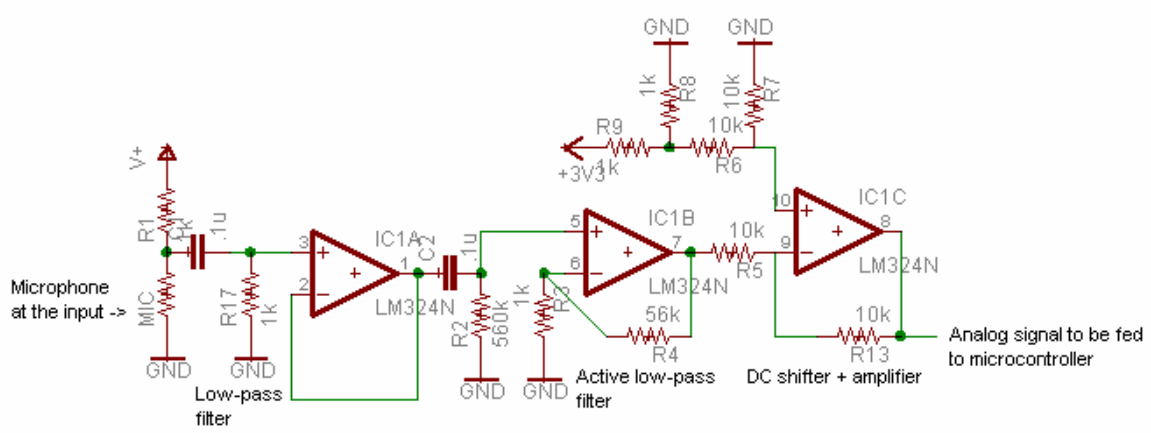

Fig 3: Tx side: Microphone converting voice to an electrical audio signal, and filtering followed by DC-shifting of the signal to fit working range of the microcontroller

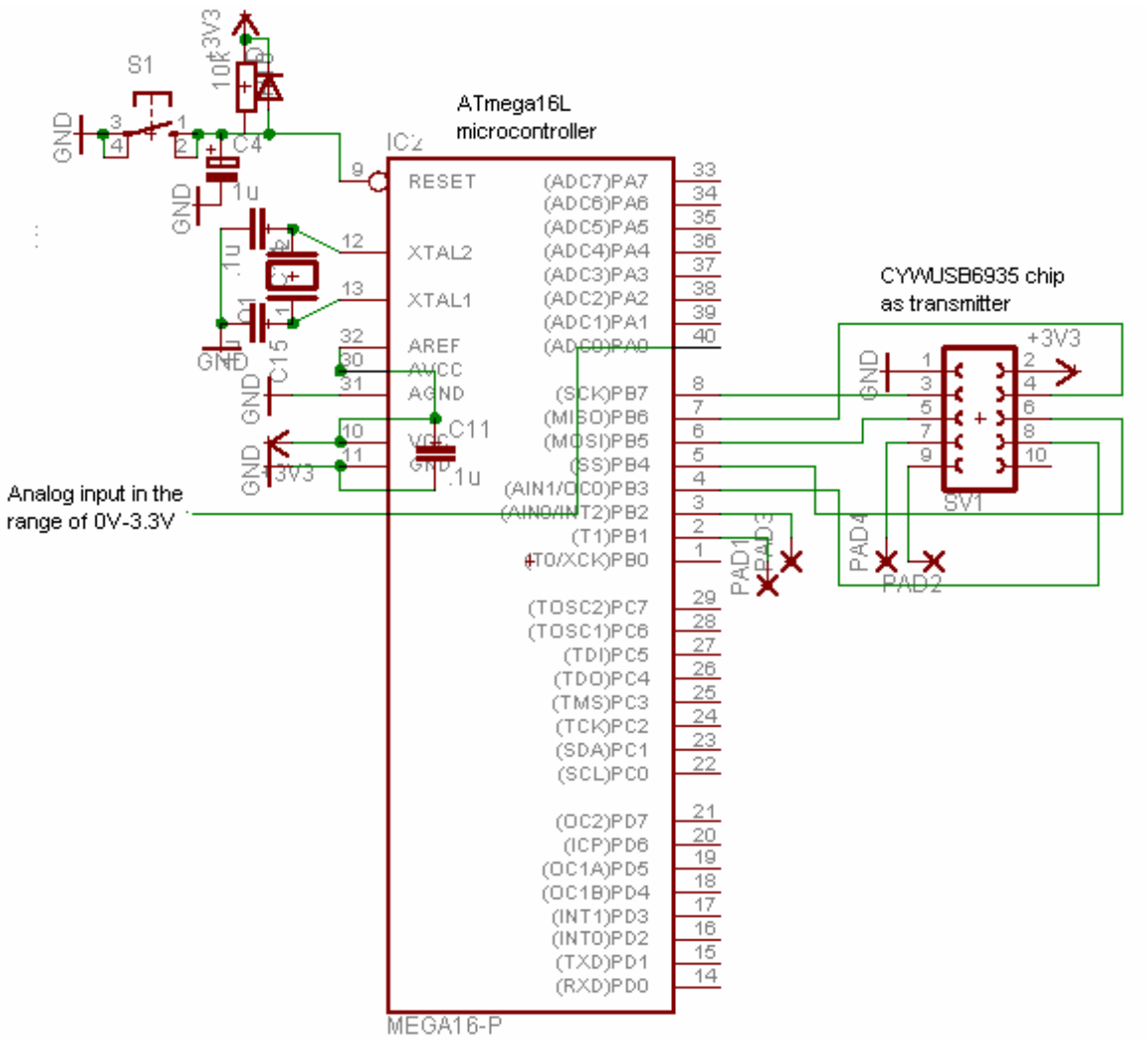

Fig 4: Tx side: The microcontroller accepting analog input, converting using internal ADC to a digital signal, providing the data byte after encryption to the wireless transmitter module via the Serial Peripheral Interface

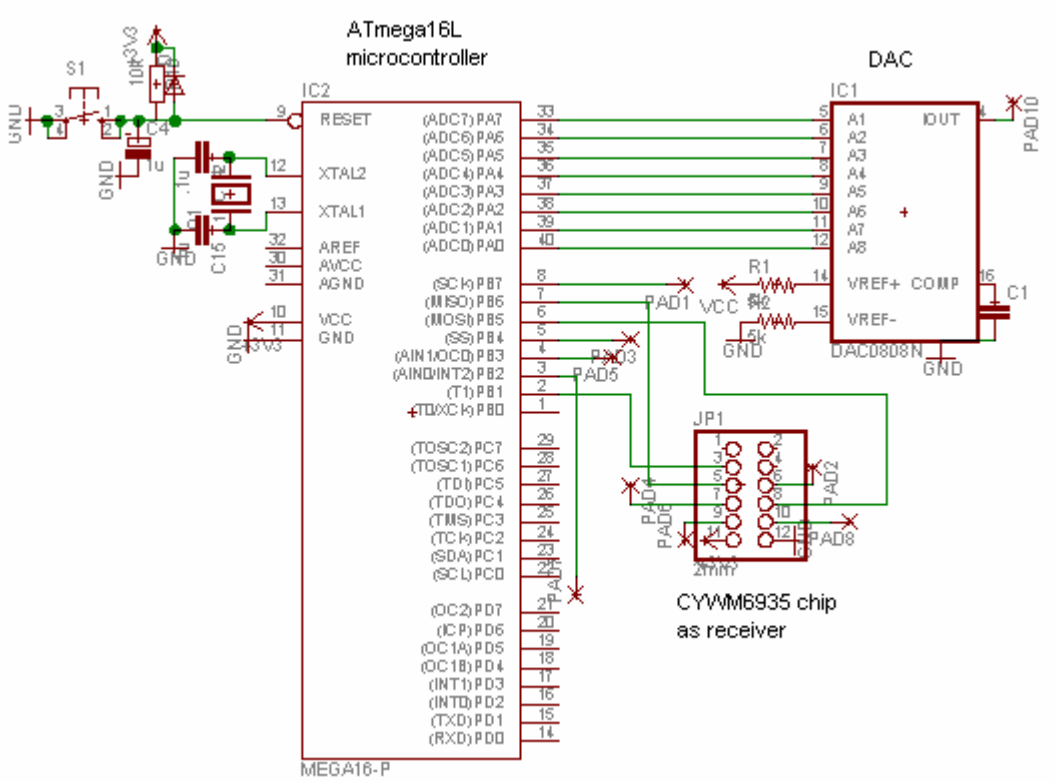

Fig 5: Rx side: The microcontroller receiving the encrypted data byte from the wireless receiver module via the Serial Peripheral Interface, decrypting the data and feeding to a DAC

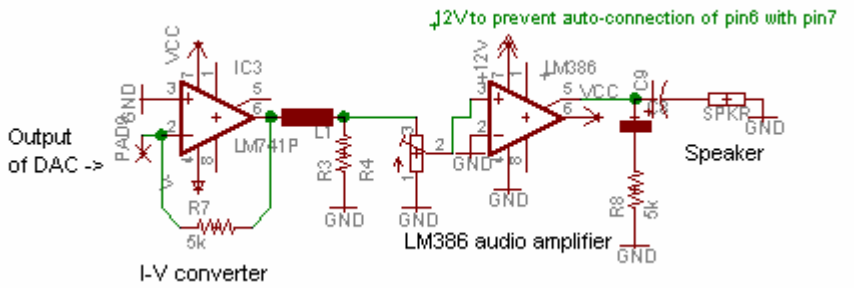

Fig 6: Rx side: The analog current output of DAC is converted to voltage via an I-V converter and then fed to an audio amplifier which is then given to speaker which reproduces the input voice signal

5. Test procedure

- The microphone is given sound input of varying amplitude levels and the maximum value at average speaking amplitude is noted.
- Calibration is done to find a reasonable position of the potentiometer to divide the output of the I-V converter after the DAC stage. This corresponds to a reasonable volume for the output at the speaker.
- Sampling the input voice signal at different rates with different number of bits per sample is tried out.
- The circuits were tested for varying physical distance between transmitter and receiver.

6. Test results

- Test results indicate that the signal peak-to-peak amplitude from the output of the microphone is close to 40mV for average speaking level. We thus, have to amplify this range to about 3V and have it shifted to a DC value of about 1.5 V so that the effective range of the signal now becomes 0V to 3.3V which is the input range of the ADC within the microcontroller.
- Test results show that upto 8 bits per sample and a sampling rate of about 8KHz give reasonable sound reproduction.
- The circuits were found to work satisfactorily for all distances (between transmitter and receiver) less than 5m.

7. Suggestions for further improvement

In hardware:

- Voltage levels of 15V and -15V are used in the project. Design may be done to remove this requirement, to make 3.3V and Ground, the only voltage levels required. This can make the transmitter and receiver circuits portable.
- One may achieve the same results as in this project by using an ATmega8 with a fewer number of ports, resulting in smaller size and lower cost.
- Wireless modules with higher data rates (such as CC1000 or CC2500), if used, can improve the quality of sound at the speaker output.

In software:

- The encryption may be carried out by modulo-2 addition of the input data with strings generated by the Linear Feedback Shift Register (LFSR) sequences. Such sequences are well-known to have pseudo-noise characteristics. They form one of the best encryption schemes for real-time encryption and are easy to implement too.
- If LFSR sequence encryption be used, a synchronising byte may be provided after regular intervals so that if the receiver goes out of range, it can still decrypt the correct input data after receiving the synchronising byte. The byte may provide information about the current random string.
- The frequency hopping may itself be done in a randomised way with the hop-frequency information provided within the synchronising byte.

8. Problems faced

Working on this project was found to be a great learning experience. Some of the problems faced for the original attempts made, as well as attempts made to tackle problems that arose are listed here:

Attempt: We tried building our own transmitter using the 74LV4046 chip as also the HCT4046 chip.
Problem faced: An effective power amplifier stage could not be built.

Attempt: We attempted to build our own transmitter-receiver pair using analog components. We were successful in building our own Colpitt's oscillator working at the desired frequency.
Problem faced: The Class C amplifier required for transmitting reasonable amount of power could not be built. In the early experiments, we were using the BC107 transistor for the Class C amplifier which failed because BC107 is not designed to work at the higher range of frequencies. After realizing this, we shifted to the BF107 transistor, however, suitable choice of inductors was not available. Upon this, we considered designing our own inductor. However, since a quick decision was impending, we decided at this stage, that we may look at transmitter-receiver pairs already available in the market in the form of wireless toy cars.

Attempt: We obtained toy cars that were remote-controlled, with the transmitter and receiver working in the citizen's band - one at 27MHz and the other at 49MHz. The idea was to use both of these transceivers simultaneously to send the bits. The security would come from the fact that an adversary would not be able to tune a receiver to both the channels simultaneously and hence, won't be able to acquire the entire bit stream. We modified the circuits for the transmitter and receiver for the toy cars to extract only the analog component of the circuit board.
Problem faced: A good data rate could not be maintained. The car circuit allowed for a maximum data rate of less than 300 bits/ sec. This is severely insufficient for real-time voice communication.

Upon availability, we had decided to use the CYWUSB6935 and CYWM6935 transceiver modules.

Problem faced: The ATmega16 microcontroller, which works at 5V does not allow a working voltage of 3.3V. To control the wireless transceiver chips at 3.3V, we constructed voltage dividers to step down the voltage to 3.3V. Moreover, we required voltage step-up to receive input (in to the microcontroller from the chip). The circuits built for the same were messy and easily allowed errors while testing.
Attempt to tackle it: This is why we turned to using the ATmega16L microcontroller.

Problem faced: It was observed that universal programmers often found an error with the ATmega16L microcontroller and reported the chip as defunct, when, in fact, it was in perfect condition.

Attempt to tackle it: We built our own programmer board which connects to the parallel port of a computer and used WinAVR software for burning the microcontrollers.

Attempt: A data rate of 32 kilobits/sec would have been ideal, for it would allow 4 kilosamples/sec with 8 bits/sample.
Problem faced: The data rate that we could manage to support was about 16kilobits/sec and hence, a sampling rate of 2 kilosamples/sec. This provided considerable distortion. Words spoken at the transmitter end are practically unrecognisable at the receiver end. However, the duration of the words is recognized correctly. Morse code communication through sound of mouth, for instance, can be carried out. The wireless modules allow for data rates upto 62.5 kilobits/sec, but we were unable to implement the same.

Attempt: Frequency hopping was tried.
Problem faced: Synchronisation was the most important hurdle for frequency hopping. Since this could not be achieved by simple transmission of "known bytes", frequency hopping could not be implemented.

9. Compromises

The one compromise that we did not choose to make, at the expense of all other costs, is the transmission of real-time voice data. All the attempts made were with the primary motivation of the challenging target of real-time voice transmission in mind.
Compromises that we did make were:

- Data rate – allowed it to fall to 16 kilobits/sec – 8bits/sample gives 2 kilosamples/sec
- Simple XOR encryption scheme instead of LFSR encryption, due to lack of synchronisation
- Frequency hopping was not implemented at all, as synchronisation could not be achieved

Voice transmission in real-time requires a high data rate. We could obtain rates of upto 16kilobits/sec, because of which, voice quality is poor.
Because of issues in synchronisation, we use simple XOR encryption. Also, due to the same issues, frequency hopping could not be implemented.

10. Conclusion

Wireless transmission of voice with security brought about by encryption is implemented. Frequency hopping was tried without success.

11. List of references

- Building a Colpitt's oscillator : http://en.wikipedia.org/wiki/Colpitts_oscillator
- Building a Class C amplifier : http://hem.passagen.se/communication/clc.html
- Phase locked loop tutorial: http://www.st-andrews.ac.uk/~jcgl/Scots_Guide/RadCom/part13/page1.html
- Datasheet for the ATmega16L microcontroller: www.datasheetcatalog.com/datasheets_pdf/A/T/M/E/ATMEGA16L.shtml

- Datasheet for CYWM6935: http://download.cypress.com.edgesuite.net/design_resources/datasheets/contents/cywm6935_8.pdf
- Datasheet for CYWUSB6935: http://download.cypress.com.edgesuite.net/design_resources/datasheets/contents/cywusb6935_8.pdf
- LFSR sequence theory: http://learn.ouhk.edu.hk/~mt888/url/Unit9/2/m_sequence_linear_feedback_shift_register_lfsr.htm