GEORG-AUGUST-UNIVERSITÄT
GÖTTINGEN

Georg-August-Universität Göttingen
Faculty of Mathematics and Computer Science
Institute of Computer Science
Telematics Group
Sensorlab

Lab 6 –Sound Recognition

Chencheng Liang – University of Goettingen – 11603960

Introduction

This lab use MTS310CB to detect the voice information, and use regular programming mote with base station program to listen to the message. The listening java commands should be modify to use a higher default baud rate, otherwise it cannot receive any information. And the regular base station listening program cannot get the voice data properly, so we need to use the capture application to receive the audio data.

In assignment 6.1 practical, I modified capture.java to process the audio data, and recognize the noise , then print it on the screen. This is the audio processing in PC. And I modified code in MicReadStreamC.nc, so the audio processing is done at the mote. Unlike in the PC(print information on the screen to should the result), in the mote, according to the noise state, I can change the LEDs and buzzers to be the feedback.

In assignment 6.1 theory, I need to consider the pros and cons for processing audio signal on the PC and on the sensor mote. Also I need to consider if the Fourier transforms is useful for analyzing audio data, and if it is feasible in real time audio data capturing.

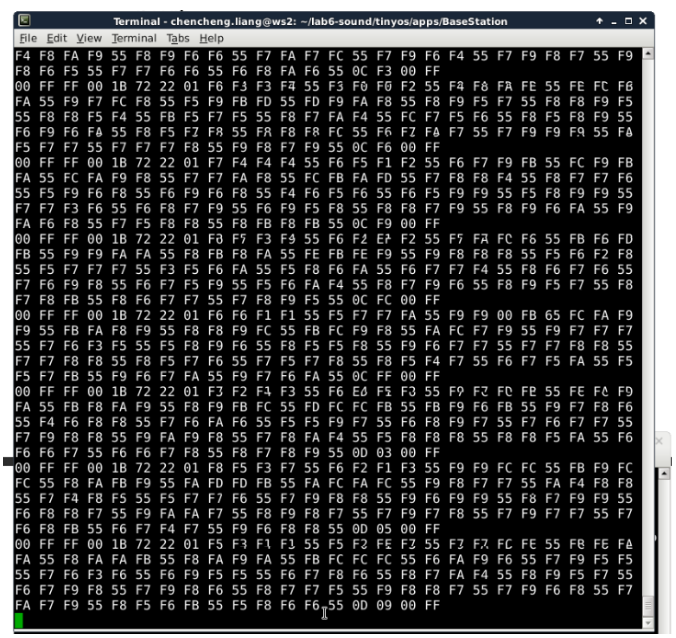
Other audio processing approaches are briefly introduced in that theory part.

Assignment 6.1

Practical:

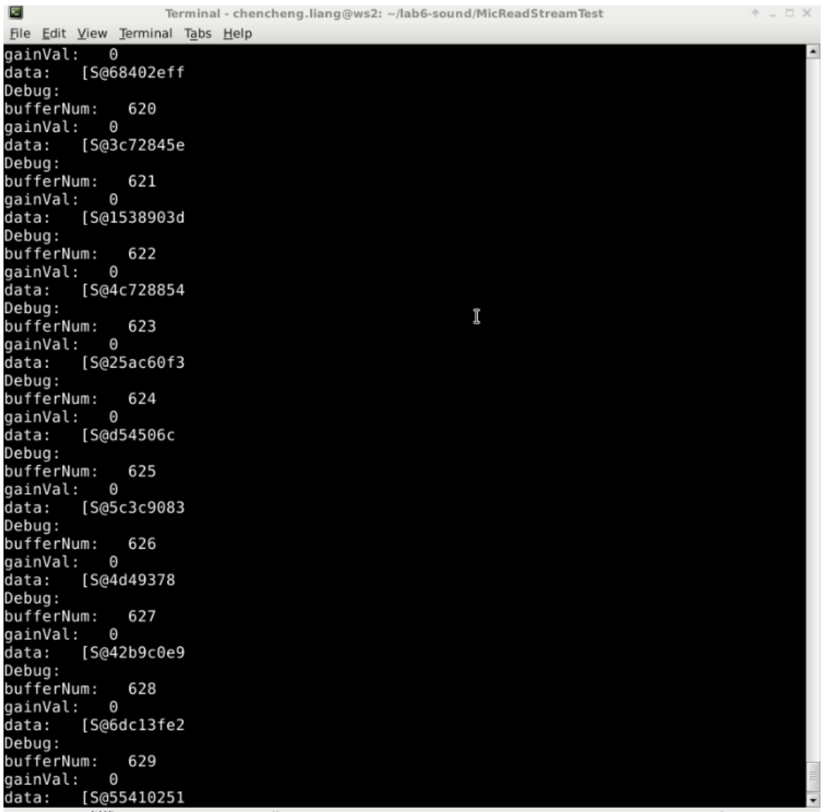
For the setup, first go to MicReadStreamTest directory, then compile the program with command “make iris”, then install it into a mote. Then go to Basestation directory, install base station program to another mote. Then connect the mote with program MicReadStreamTest to MTS310CB sensor to detect and send audio data.

If I use basic base station listening command “java net.tinyos.tools.Listen -comm serial@/dev/ttyUSB1:iris”, it will not work at all, because audio data have higher baud rate. So I need to use correct baud rate by command “java net.tinyos.tools.Listen -comm serial@/dev/ttyUSB1:230400”. But the result is like this:

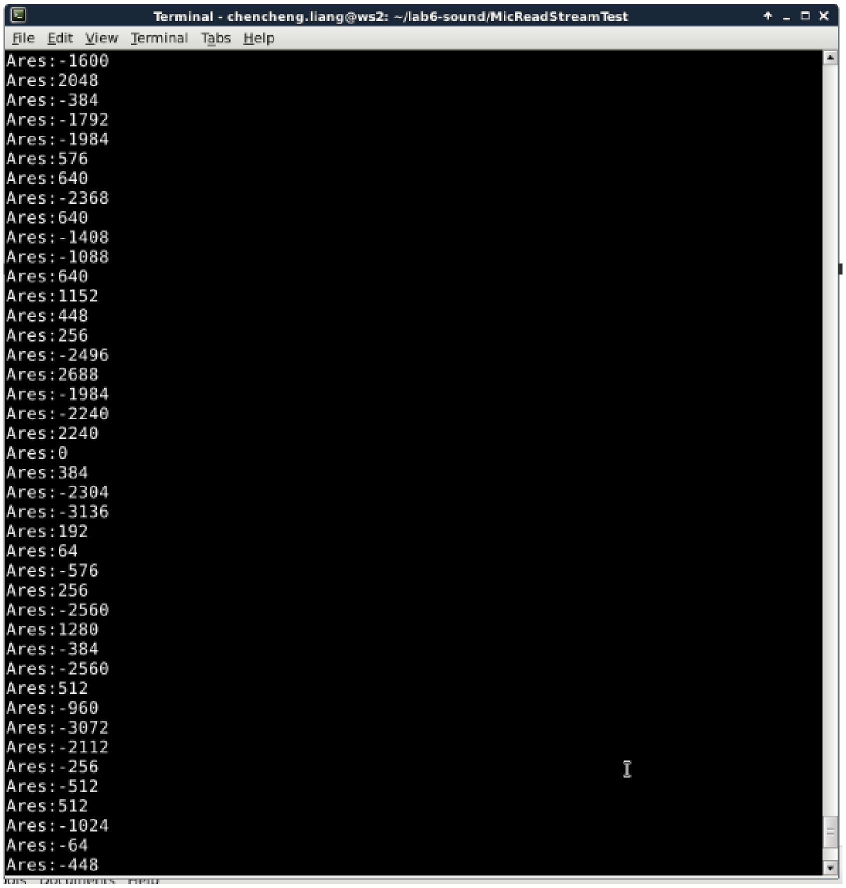


So I need to use Capture application in MicReadStreamTest to listen to the audio data in base station with the command”java Capture -comm serial@/dev/ttyUSB1:230400”. However in Capture.java, there is no proper print code, so I cannot see any information on the screen. So I add some out.println() inside of Capture.java to print something on the screen.

To finish this task, I need to know in the message which part represents the audio data, so I print all variables received in the base station on the screen:

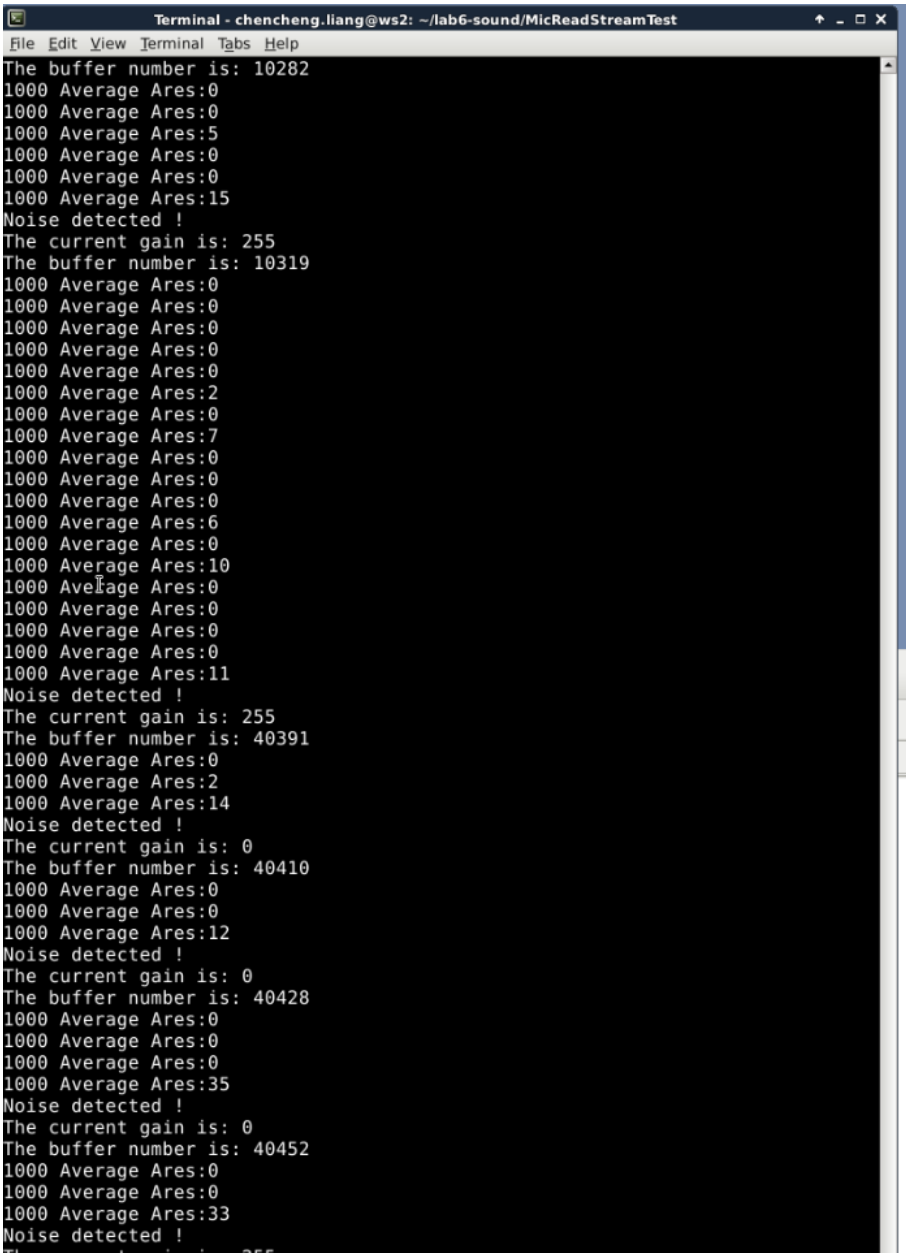


By this I know that variable bufferNum represents the data count, variable gainVal represent the gain, and variable data[] represnets the audio information. Then I found variable data[] can be processed in a function called dataCapture() inCapture.java. And for calling the function dataCapture(), I need to call function initCapture(). So in function messageReceived(), which processes the received data from mote, I call function initCapture() to initialize the processing of received data. And in function dataCapture(), I found a variable ares[] received the audio data, so I print all area[] on the screen:



The data come very fast. However I still can see that when I make some noise, this data changed dramatically. So it verifies that this variable area[] denotes the audio data. By processing this variable, the program can recognize the sudden noise.

I declare two global variables(int count, int avg) to process the audio data, the processing method is very simple. Add every 1000 ares[] together, get a average, and shrink down the range, and store it to a variable avg. If avg more than 10, it is a noise, otherwise, it is not. And receive audio data every 1000 times, print once on the screen, then I have the following result:



In the MicReadStreamC.nc, I did almost the same process on variable buf[], so accoding to the audio, the mote can response by LEDs or buzzers.

Theory:

Other approach for audio signal, we can run some sound reorganization algorithm on the PC terminal or on the motes, these algorithm can process the raw sound data in different ways.

I think audio processing should be put on the PC terminal, because audio data stream is RAM-consuming, and usually bigger than 8KB RAM which is the motes have. And processing these audio data for the motes, it needs extra computer power. But for the motes, they use battery as power supply, which is limited. But if we put the process on PC, then the motes still should send a lot of data to the PC, which also consumes a lot of power. And for the PC, if there are thousands of motes sending messages back, it should have large RAM and powerful CPU.

Fourier transforms is useful for analyzing sensed audio data. A algorithm called Fast Fourier

Transform(FFT) by decomposing an N point time domain signal into N time domain signals each composed of a single point. The second step is to calculate the N frequency spectra corresponding to these N time domain signals. Lastly, the N spectra are synthesized into a single frequency spectrum [1]. The complexity of normal Fourier is $O(N^2)$, and for FFT is O(N log N). I think this speed is considered slow in computer world, because O(N) is normal speed. So I think Fourier transform is not fast at all.

Calculating Fourier coefficients is complex, I think the motes cannot do that with such small RAM and slow CPU speed in real time.

I do not know any other audio processing approach but I read a article [2], which use time-frequency approaches to do it. In that article it mentions that a majority of audio processing techniques address the following 3 application areas: compression, classification, and security. So in the mote we can run the compression algorithm to compress the audio data, then transmit them, which can save a lot of energy, and in PC terminal we decompress the package, and use different classification algorithm to process that audio data.

Lessons Learned & Conclusion

In this lesson, I learn I can use MTS310CB to collect audio signal from the sensor, and listen to the message in the base station. And the noise application I implemented on it has many potential uses in practical, for example, at least we can put the sensor there to find out if there is sound, then record it, so we can figure out what was happening at that place in a sense.

I learned in the base station I need to use the correct baud rate then I can receive that message from motes. And by using Capture application I can analyze the received audio signal.

I learned how to deal with audio data in both theory and practical. In the beginning I do not know which data is audio data, what is the structure of audio data. In practical part of this lab, most of time I used to try to find out where I can receive the audio data, and how can I process the data(they are arrays with high sending frequency). And I just used some easy way to smooth the data, then find the high pitch, and recognize it as the noise. But actually, there are many other algorithms which can process it and event classify it.

I also learned some pros and cons about at where we should implement the audio signal program. The audio signal process can be implemented on both PC and the motes, If it is implemented by PC, then the motes need to send back a lot of raw data, which cost more power. For solving this, we can let the mote run a compression algorithm , then send the compressed data. If the audio signal process is implemented on the motes, it must be a simple program, otherwise the hardware cannot handle that, and it will get more power consuming. And because the limited resources the motes have, it may cannot send back the real time data.

Before this lab I do not know how the radio sensor works, and now I understand, and I know how to catch the audio signal, and do some simple process on it.

References

[1]http://www.dspguide.com/ch12/2.htm

[2] http://asp.eurasipjournals.springeropen.com/articles/10.1155/2010/451695