PHYSIOLOGICAL SIGNAL ACQUISITION SYSTEM FOR WIRELESS COMMUNICATION

Zhihao Cui

Bachelor of Engineering Electronic Engineering

MACQUARIE
University
SYDNEY·AUSTRALIA

Department of Electronic Engineering
Macquarie University

November 10 2016

Academic Supervisor: Doctor Oya Sevimli

Industry Supervisor: Teck, Loi and Nyuyen Cong-Van

ACKNOWLEDGMENTS

I would like to acknowledge the help of my academic supervisor Doctor Oya Sevimli and key industry engineers, Loi Teck, Qin Humphry and Nyuyen Cong-Van from National Acoustic Laboratory. Thomas Meggitt, and I worked together to achieve the prototype and computer-based software. This thesis work is aimed to upgrade an existing HEARLab System and carry through signal processing analysis. The thesis is an extension based on the work of Loi Teck is a development engineering at National Acoustic Laboratory.

STATEMENT OF CANDIDATE

I, Zhihao Cui, declare that this report, submitted as part of the requirement for the award of Bachelor of Engineering in the Department of Electronic Engineering, Macquarie University, is entirely my own work unless otherwise referenced or acknowledged. This document has not been submitted for qualification or assessment at any other academic institution.

Student's Name: Zhihao Cui

Student's Signature: ZHIHAO CUI

Date: 10 November 2016

ABSTRACT

National Acoustic Laboratory uses HEARLab System to record the brain function of the hearing loss people. Typically, the principle of the HEARLab System is by recording the patient's electroencephalography (EEG) to detect the hearing loss level of the patient. Recently, because of the wired signal transmission method, the patient has to be constrained in a particular space, which causes inconvenience to the experimenter and the subject. Thus, the aim of the project is to use a wireless connection method to replace the traditional physical line connection method and to achieve the internet of things (IoT). In this thesis, it is going to use the Texas Instruments CC3200 LanuchPad to make the wireless information transmission method. Therefore, based on the Wi-Fi environment the new HEARLab System can wirelessly transmit information between the PC and the CC3200 LaunchPad by using user datagram protocol (UDP). The UDP is written in JAVA and discussed in detail in the thesis. Further work aims to process the received EEG signals. Typically, regarding the signal processing principle, the received EEG signals are stored in the digital format on the computer. On this occasion, JAVA digital signal processing is going to be introduced in the thesis. All the related code or tools will be attached in the Dropbox.

Contents

List of Figures

Chapter 1

Introduction

National Acoustic Laboratory (NAL) [1] is a world leader in research into hearing assessment, hearing loss prevention and hearing rehabilitation. At the present, NAL has invented a hearing test system, HEARLab® System: Cortical Auditory Evoked Potential Analyzer (Figure 1.1), is related to the Cortical Threshold Estimation (CTE) and Aided Cortical Assessment (ACA) which used to help estimating hearing thresholds on patients who, for whatever reason, are unable to give reliable behavioral responses to indicate that they have heard a sound [1]. The patients could include infants and small children who have not yet developed language skills and adults who are disabled or uncooperative [2].

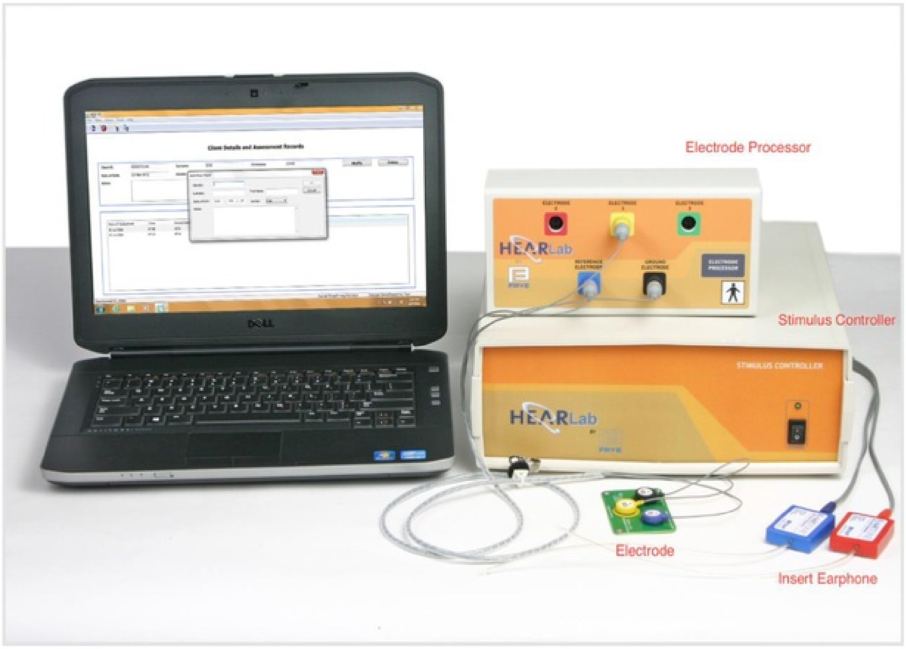


Figure 1.1: The HEARLab® System: Cortical Auditory Evoked Potential Analyzer [3]

1.1 HEARLab System Hardware Components

The HEARLab System is built around a standard PC operating under Windows. As figure 1.1 shows, the top medium box beside the computer is the electrode processor, which uses for connecting the electrodes. It connects to the jack on the stimulus controller. The bottom box is the stimulus controller, which is the HEARLab System's main hardware unit. It connects to the PC with a USB cable and functions as an Input/output device that connects to various transducers.

1.2 HEARLab System Working Process

The basic principle of the HEARLab System is that a transducer such as an insert earphone, or loudspeaker use to deliver the testing signals to the subject's ears, and electrodes placed on the subject's head to measure the cortical response (Figure 1.2) [3].

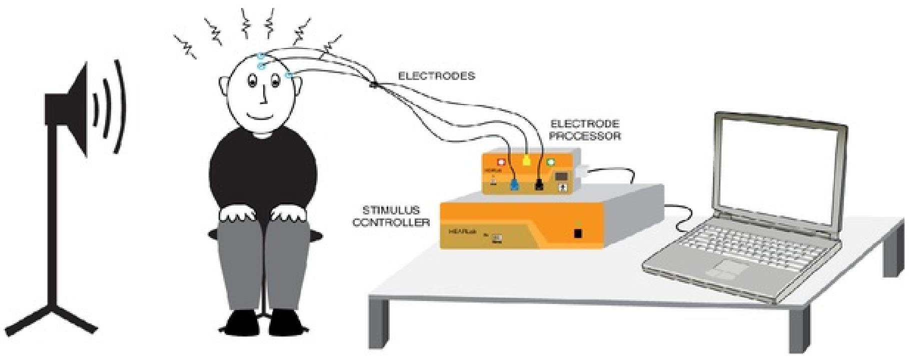


Figure 1.2: HEARLab System fundamental diagram

Typically, EEG potentials recorded at the scalp are tiny. They are measured using units such as millivolts. In order to be evaluated in computers or viewed on display, the EEG signals must be amplified to the level of volts. After the amplification, the amplified EEG signals should span as much as possible of the range that the analog-to-digital converter (ADC) without blocking. For instance, if the recorded EEG signals varies in a range of ± 20 uV and the ADC has an input range of ± 10 V, the amplification should be 500,000 times [4].

Similarly, the HEARLab System amplifies the recorded EEG and convert the analog EEG signals to digital signals, so that, the computer software is able to capture the EEG signals and process the signal processing and display the EEG signals. Meanwhile, a series complex statistical analysis: Hotelling T^2 and p-value criteria is running underneath to automatically calculate the patient's hearing threshold level in terms of Cortical Auditory Evoked Potentials (CAEPs) or auditory brainstem response (ABR). Finally, an easy understand hearing assessment report is going to present to the patients.

1.3 Next Generation of HEARLab System

At present, the HEARLab System transmits the cortical response through a USB cable between the stimulus controller, electrode processor, and the computer. For industry purpose, NAL wants to use new technology to improve the HEARLab System's usability, reliability, and performance.

In the perspective of the usability, the new generation of HEARLab System should transfer the information wirelessly between the computer and the stimulus controller to reduce the amount of the cables and minimize the size of the stimulus controller box. Therefore, a new light and handy HEARLab System is looking forward at the end of this project.

In the performance side, the new HEARLab System should able to maintain the same information transmission efficiency as the old version. For instance, if the implementation of the IoT brings down the transmission of new HEARLab System information's certainty, integrality, and definiteness. Then, it is classifying the IoT may not suitable for NAL new HEARLab System and an improvement should find in the future.

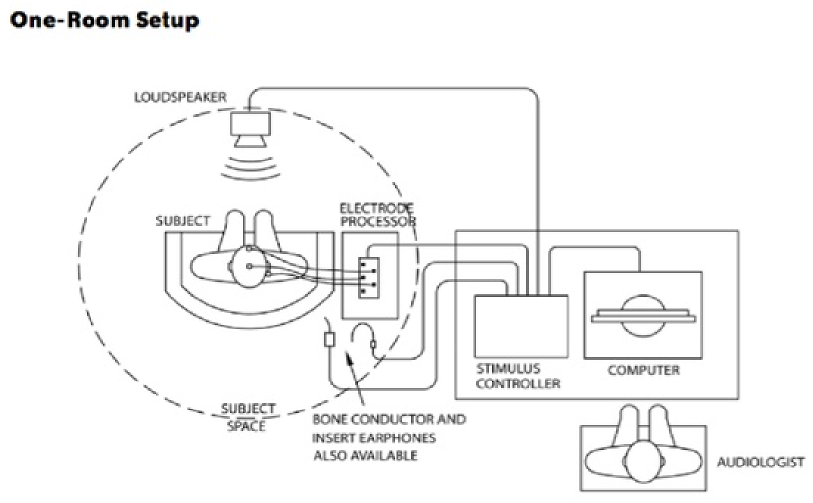


Figure 1.3: HEARLab System one room setup [4]

With the development of the technology, IoT has attracted more and more concern, NAL put faith in the wireless information transmission is the trend of the future, so putting the IoT concept into the next generation HEARLab System is the goal of this project. Figure 1.3 shows the original HEARLab System testing procedure.

1.4 Synopsis

This thesis base on the NAL HEARLab System's original function and add into the IoT concept to produce a next generation demo, which can transmit and receive information wirelessly. The current HEARLab System was invented by Loi Teck' team. Qin Humphry and Liu Kyle are the software engineering; Clinch Barrier is the Development Engineer who is a specialist in the manufacturing of the device. And Nyuyen Cong-Van, Bardy Fabrice they are both the research engineering. To achieve the wireless information transformation function, Loi Teck chooses the Texas Instruments (TI) Company's product, CC3200 Module Launchpad, as the motherboard to replace current HEARLab System's stimulus controller device.

With help from the team at NAL to prove the concept of creating a wireless CAEP testing system capable of delivering equal results to that demonstrated by the HEARLab system. Two ENGG411 students completing their research theses, Zhihao Cui the author of this thesis and Thomas Meggitt will undertake the proof of concept. The theses although separate will aim to achieve a combined end goal by investigating and undertaking individual components of the project. Each thesis paper will be individual however the thesis project will be made of 30% combined work with the remaining 70% to be individual, the project has been split the way it has to ensure the research question is thoroughly examined. Figure 1.4 shows how we separated our tasks.

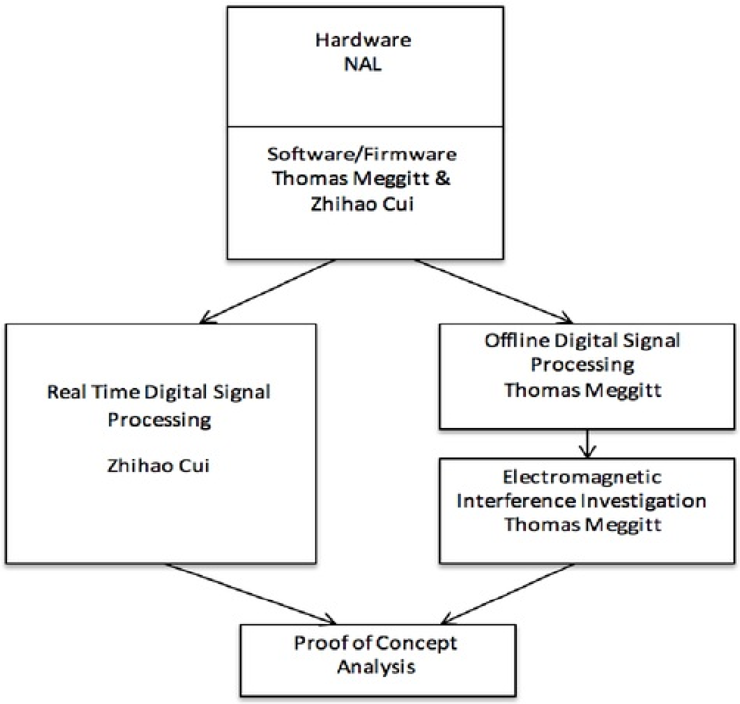

Hardware
NAL
Software/Firmware
Thomas Meggitt &
Zhihao Cui
Real Time Digital Signal Processing
Zhihao Cui
Offline Digital Signal Processing
Thomas Meggitt
Electromagnetic Interference Investigation
Thomas Meggitt
Proof of Concept Analysis

Figure 1.4 Division and cooperation in the project

Both thesis papers will attempt to determine whether the CAEP test can be successfully conducted with the design and implementation of a wireless system to transmit a stimulus signal from a PC to the patient and to transmit the recorded signals from the patient to a PC. The success of the system is based upon whether it is capable of functioning to the requirements of the test and whether the use of this system will affect the test results. The combined 30% will be made of the PC software and the test systems firmware to transmit and receive the signal, once this is achieved the thesis goals will differ such that Thomas Meggitt will investigate the digital signal processing and display of result using offline methods that occur after the recording has been performed. The goal of Thomas's section is to prove the concept in a simple, fast and effective method to allow for the further investigation of the effect of wireless transmission in the form of electromagnetic radiation caused by the system on the results and thus investigate the concept of wireless transmission for this testing use in more depth. In contrast, this thesis will investigate the NAL original HEARLab System and according to the requirements to write a demo software system, which bases on JAVA Language to achieve the real-time signal processing. For a deep understanding of the project, to deal with the CAEPs signal that is obtained by the Cortical Auditory Threshold Estimation (CATE) technique. Digital signal processing (DSP), time-locked averaging and signal to noise ratio (SNR). These concepts will be considered in the following chapter. In addition, a simple Graphical User Interface (GUI) is going to be designed to provide a user-friendly and continent user interface.

In the 10 weeks' project, the most of time, about 4 weeks was dedicated in developing the software for using the computer to detect the received EEG and cut the EEG to a single epoch (a segment of the EEG). The hardest part is to understand every steps of the current HEARLab System and then using JAVA to write a new algorithm to achieve the same functions.

This project contains a vast of digital signal processing and wireless communication. According to the HEARLab System functionality, it is a huge task to deal with the audio signal conversion between analog and digital.

I. The procedure of the original HEARLab System

The software produces a series of stimuli signals such as /mua/, /Ta/, /ga/, which present the most frequently pronounce in humans' language. Typically, the computer sends the same stimuli signals to two different channels. One is the loudspeaker or the headsets presentation channel. The other one is lead to the stimulus controller unit. Once the patient heard the stimuli sound, the patient's cortical generate a biopotential voltage in uV. The electrodes function as the detector to detect the biopotential voltage and via specially designed cable transmits to the electrode processor. The electrode processor does two tasks, the first one is to amplify the received EEG, the other one is to receive the stimuli signals, which act as a reference. The reference is used to process the onset detection and cutting the EEG epoch, which plays a significant role in the whole system (The onset detection and cutting epoch is going to describe in detail in Chapter 4). Then, the stimuli signal and the EEG response signals through the cable transmit back to the stimulus controller unit simultaneously. After the stimulus, controller unit received the stimuli signals and EEG signals in the form of analog, then the built-in codec sampling the analog signals to the digital signals and transmit the digital singles via USB cable to the computer. The final stage, the computer software analysis the digital signals and present it in a human readable format.

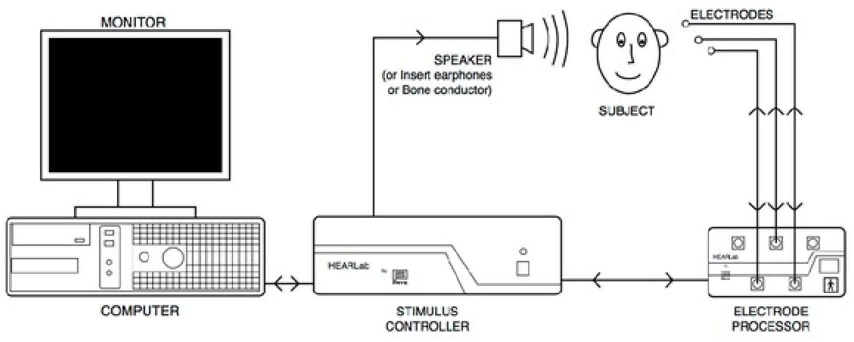


Figure 1.5 The original HEARLab System

II. The aim of the wireless HEARLab System

In this project, the stimulus controller unit is replaced by the CC3200 LanuchPad and the CC3200 Audio BoostPack (the CC3200 LanchPad and CC3200 Audio BoostPack can be seen as the stimulus controller unit with the wireless transmission function, see Figure 1.5). In this case, the USB cable is going to be abandoned. In principle, the computer generates the stimuli signals and transmit wirelessly to the CC3200 LanuchPad, according to the CC3200 Audio BoostPack has one stereo line out and loudspeakers out; therefore, using the stereo line out to loop the stimuli signals back to the electrode processor, and the loudspeaker line out used to present the stimuli sound. Meanwhile, the EEG signal, which was produced by the patient through a cable transmit to the electrode processor and merge with the stimuli signals wirelessly transmit to the computer. In the end, the computer runs the software to analyze and display the EEG response signals.

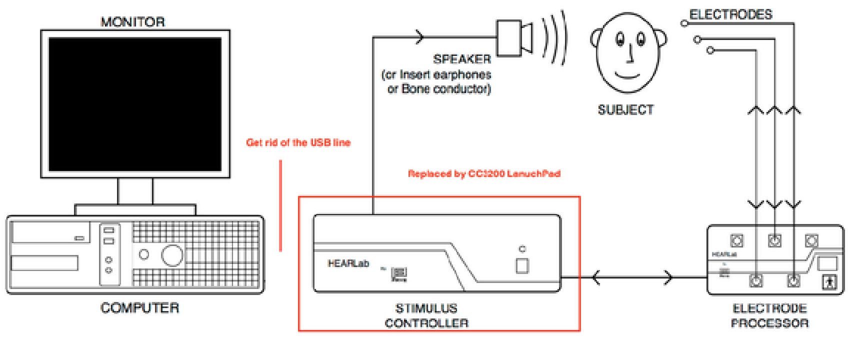


Figure 1.6 The aim of the wireless HEARLab System

–

Chapter 2

Background

Introduction

Based on a company background project, the HEARLab System involves the knowledge in different fields. Such as, the electronic engineering department designed the HEARLab System. And the software engineering department is trying to make the functions come true. The auditory engineering department analyzes the results and generates a hearing diagnosis report for the patient. The referring background knowledge is going to present in the following paragraphs.

2.1 Audiology in the HEARLab System

2.1.1 What is an electrophysiological signal?

Electroencephalography (EEG) is an electrophysiological monitoring method to record the electrical activity of the brain with the electrodes placed on the scalp. EEG measures voltage fluctuations resulting from ionic current within the neurons of the brain [5]. Hearing diagnostic applications generally focus on the spectral content of EEG, that is, the type of neural oscillations (popularly called "brain waves") that can be observed in EEG signals.

2.1.2 How to use EEG to diagnose hearing?

Once the EEG response has been captured and stored digitally for processing purposes, it is typically displayed in the time domain. Responses have the appearance of one or more peaks and troughs with major features reported by their latency [6]. There are a number of ways to classify the family of auditory evoked responses but the easiest and perhaps the most descriptive is based on their latency [6].

Typically, in terms of the latency, there are at least seven ways to diagnose the hearing threshold, however, the NAL is currently performing research in the application of CAEPs (Late latency potentials occur from 90 ms to 500 ms after the introduction of sound to the ear) to infant and adult assessments, and in the application of ABR (Early latency potentials occur within the first 10 - 20 ms after the introduction of sound at the ear) to infant assessments.

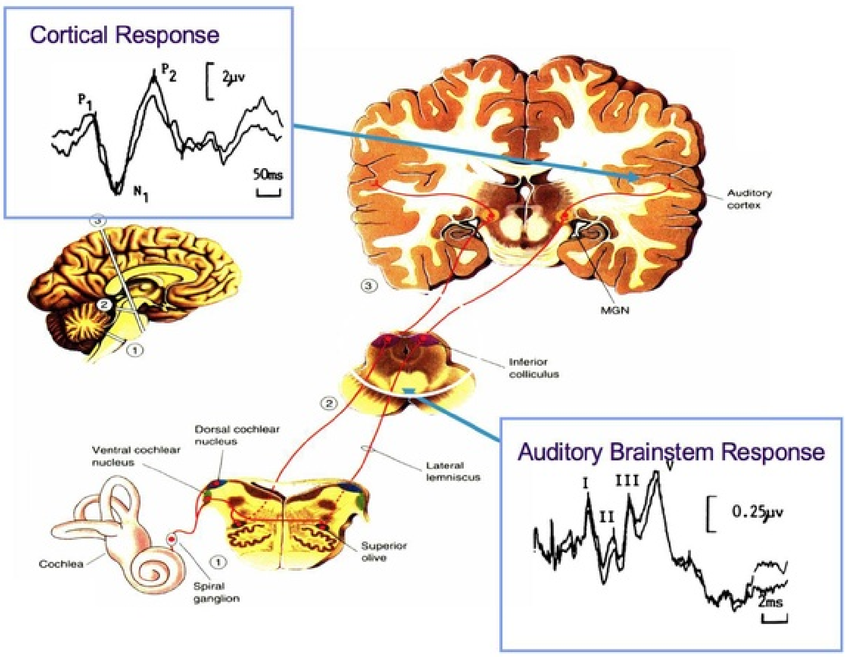


Figure 2.1: ABR and CAEPs response in different area of the humans' brain [1]

Figure 2.1 shows that the ABR and CAEPs response in different area of the humans' brain. The ABR detect at inferior colliculus (a part of the humans' brain) and the CAEPs was found in the auditory cortex. So, upon the latency of the provided stimulate signal, humans' brain is able to shows the variable reaction. Figure 2.2 shows the ABR and CAEP evoked in a captured EEG in different time.

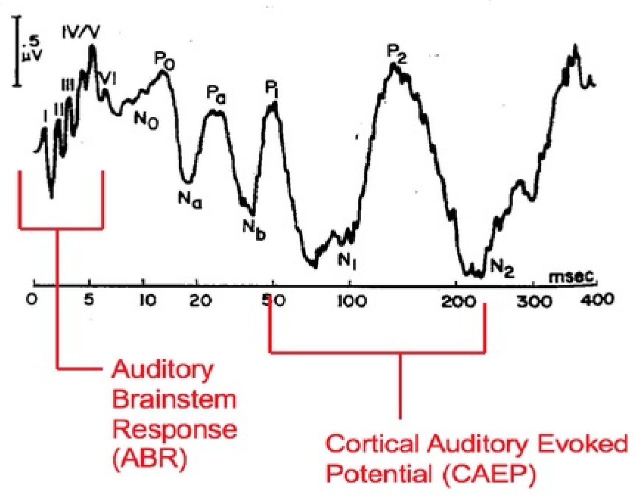


Figure 2.2: The latency of the ABR and CAEP [1]

2.1.3 What is ABR?

The auditory brainstem response (ABR) test gives information about the inner ear (cochlea) and brain pathways for hearing. This test is referred to as auditory evoked potentials (AEPs) [7]. In the clinical application the ABR was the prediction of hearing loss and determination of precise hearing threshold in infants and young children, and the person with conventional behavioral methods of hearing screening.

The first reports of the systematic recording of ABR from infants came in the mid 1970s. By the 1980s, infants at risk for hearing loss were being screened with ABR in many birthing hospitals [4, p.254]. For the infants who do not pass the birth admission screen and any subsequent re-screening begin appropriate audio logical and medical evaluations to confirm the presence of hearing loss before 3 month of age []. A test battery for children of this age “must include an EEG measure of threshold such as ABR using frequency-specific stimuli” [8]. One restriction is that the ABR test is done under anesthesia, which means that the infant need medication to help him or her sleep throughout the test.

2.1.4 What is CAEPs?

Auditory evoked potentials are the brain’s response to the presentation of an auditory stimulus. These potentials occur within the first milliseconds following stimulus onset up to 500 ms after the stimulus. The Cortical Auditory Evoked Potentials (CAEPs) is a late response occurring between 50 ms and 500 ms after stimulus onset [9]. The figure 2.2 shows the CAEPs occurs between the P1-N1-P2. In auditory filed, its called the complex P1-N1-P2.

The advantage of recording CAEPs as opposed to responses generated from ABR include the ability to measure the response when an infant is awake and the ability to use longer stimuli such as speech segments [9]. In the “Objective verification of speech perception using cortical auditory evoked potentials” it is proved that the Cortical responses were elicited by all speech sounds for all adult and infant participants. This demonstrates that CAEPs can be reliably evoked by sounds that encompass the entire speech frequency range [10].

CAEPs provide evidence of speech detection at the cortical level in the auditory system. CAEPs can be clearly observed to speech stimuli presented at conversational level in infants who are awake and have normal hearing (11, 12, 13). They may also be recorded to verify the audibility of stimuli presented at conversational level in infants fitted with hearing aids or in infants who are under evaluation for hearing aid fitting (11, 12, 13).

2.2 Texas Instruments CC3200 development board

2.2.1 TI CC3200 chip introduction

In 2014, Texas Instruments released the first microcontroller with built-in Wi-Fi – CC3200. Moreover, the TI CC3200 LanuchPad the development board (Figure 2.3) released in 2015. It worth to mention that the CC3200 means the microcontroller and the CC3200 LanuchPad means the CC3200 development board.

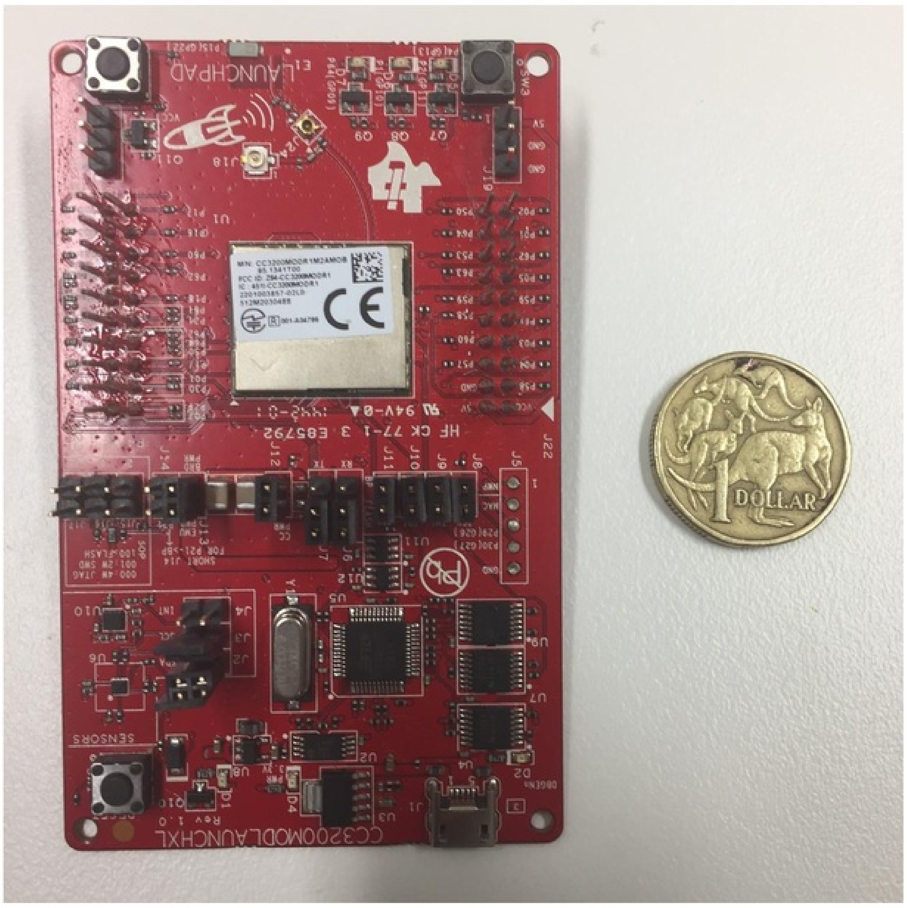


Figure 2.3: CC3200 hardware overview

The high performance CC3200 is the industry's first single-chip Microcontroller (MCU) with built-in Wi-Fi connectivity for the LaunchPad™ ecosystem. Created for the Internet of Things (IoT), the SimpleLink Wi- Fi CC3200 device is a wireless MCU that integrates a high-performance ARM® Cortex®-M4 [14] MCU allowing customers to develop an entire application with a single Integrated Chip (IC). With on-chip Wi-Fi, internet and robust security protocols, no prior Wi-Fi experience is needed for faster development [15].

The CC3200 includes three main features: Application MCU, Embedded Wi-Fi

Network Processor, and Power-Management Subsystems.

Application MCU (Figure 2.4 left section) includes an industry-standard ARM® Cortex®-M4 running at 80-MHz, and an up to 256 KB internal Random-Access Memory (RAM) to store the data and the application code, a Read-Only Memory (ROM) to store the boot loader and the peripheral deriver. And it also contains peripheral set, such as universal asynchronous receiver/transmitter (UART) [appendix B], Inter-IC Sound/Pulse-code modulation (I2S/PCM) [vide infra 2.41], Serial Peripheral Interface (SPI) [appendix C], Inter-Integrated Circuit (I2C), and ADC etc.

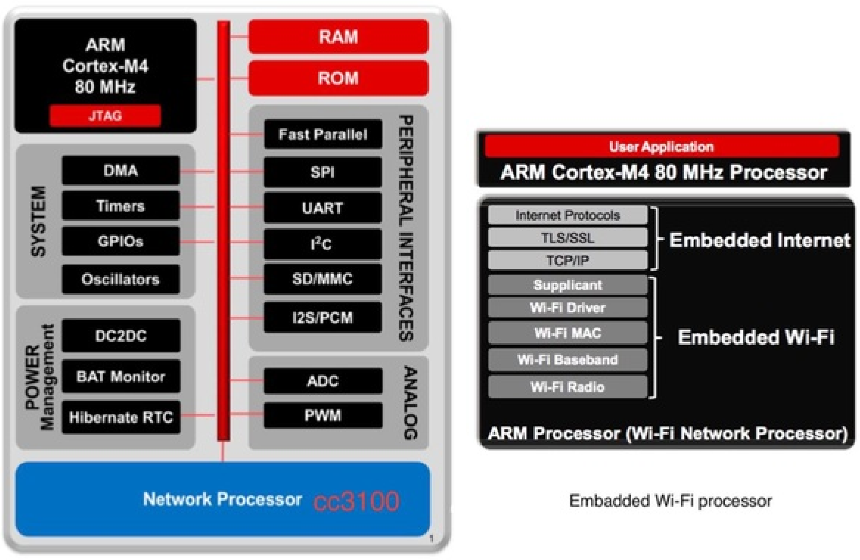


Figure 2.4: CC3200 hardware overview [3]

Embedded Wi-Fi Network Processor (Figure 2.4 right section) is TI CC3100 chip embedded on the CC3200. The CC3100 via UART and SPI to achieve the data communication with the CC3200 MCU. The CC3100 Wi-Fi network processor subsystem features a Wi-Fi Internet-on-a-Chip and contains an additional dedicated ARM MCU that completely offloads the applications MCU. This subsystem includes an 802.11 b/g/n radio, baseband, and MAC with a powerful crypto engine for fast, secure Internet connections with 256-bit encryption [16]. The CC3200 device supports Station, Access Point, and Wi-Fi Direct modes.

The power-management subsystem includes integrated DC-DC converters supporting a wide range of supply voltages. This subsystem enables low-power consumption modes, such as the hibernate with Real-Time Clock mode requiring less than 4 µA of current [16]. Also, the CC3200 LaunchPad is designed such that it can be powered by the USB connection or by external 2xAA/2xAAA batteries.

2.2.2 TI CC3200 development environment

The CC3200 development environment includes software development environment and hardware development environment. In the software development the CC3200 includes software development kit (SDK), integrated development environment (IDE), utility program, and support tools. In the hardware development environment, TI provide the CC3200 LanuchPad as the hardware development environment. The board features on-board emulation using Future Technology Devices International (FTDI) [AppendixD], UART-USB transformer, and includes temperature and acceleration sensors for a full out-of-the-box experience.

I. CC3200 LanuchPad software development environment:

To start with developing the CC3200 LanuchPad, download the TI provided SDK which include 40+ applications for Wi-Fi protocols, Internet applications and MCU peripheral examples.

Then download the CC3200 LanuchPad IDE - Code Composer Studio (CCS). CCS supports CC3200 and Embedded Processors portfolio. CCS comprises a suite of tools used to develop and debug embedded applications. It includes an optimizing C/C++ compiler, source code editor, project build environment, debugger, profiler, and many other features [17].

To load the project files to the CC3200. The Uniflash utility allows the end-users to communicate with the CC3200 LanuchPad via a standard UART interface (control the list of binary files to be Flash to the external serial Flash on the CC3200 LanuchPad) [18].

Considering that embedded Wi-Fi applications will generally lack user interfaces such as keypads or touchscreens, this process can be complex without the use of advanced I/O. To create a great user experience, TI has created SmartConfig™ technology (a mobile client application) - a one-step and one-time process to connect a CC3200-enabled device to the home wireless network [19].

II. CC3200 LanuchPad hardware development environment:

The significant configurations of the CC3200 LaunchPad hardware development environment used in the HEARLab System.

- USB interface to PC for CCS using FTDI USB driver.
- Flash update over the USB using CCS
- Standalone development platform featuring sensors, LEDs and push-buttons
- Power from USB for the CC3200 LaunchPad as well as external BoosterPack
- Operates from 2 AA alkaline batteries

2.2.3 TI CC3200 audio processing module

The stackable header of the CC3200 LaunchPad interface is used to expand the functionality of the CC3200 Launchpad. The CC3200 Audio BoosterPack (Figure 2.5) enables evaluation and development with the digital audio peripheral (I2S) present on the CC3200 Launchpad.

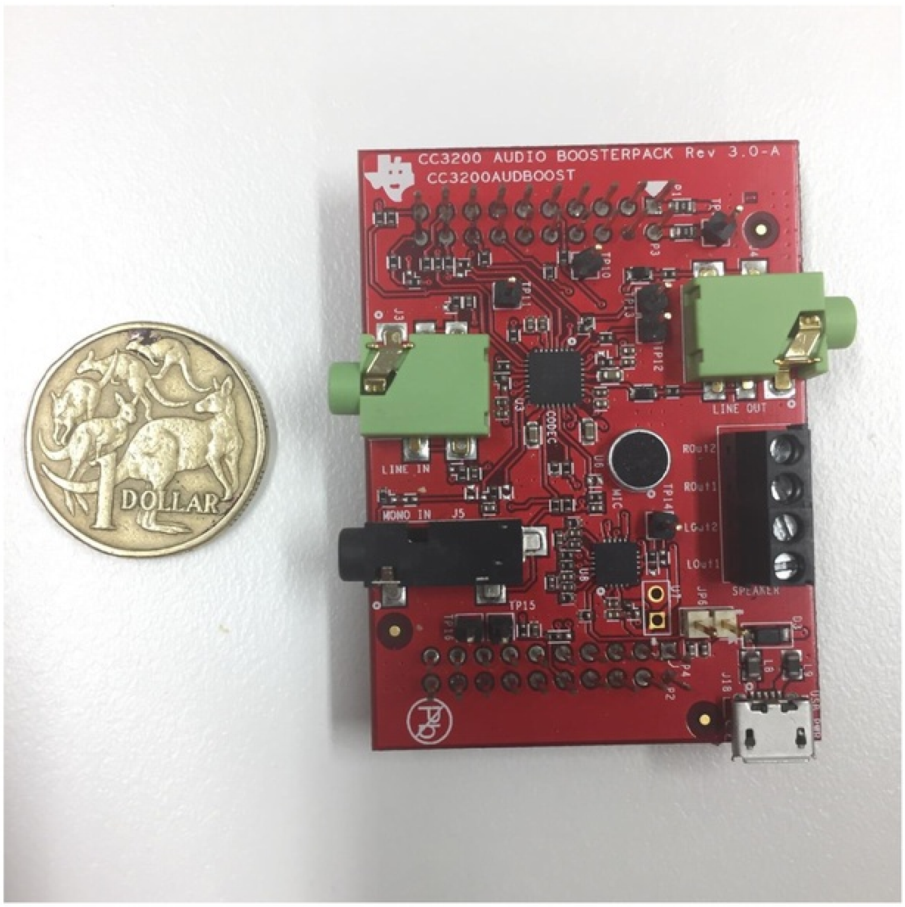


Figure 2.5: CC3200 Audio BoosterPack

The CC3200 Audio BoosterPack contains a Class-D power amplifier to drive the speakers and an ultra-low power audio codec, TLV320AIC3254, which supports programmable audio processing. Speakers, headsets, and microphones are sold separately [20]. The key features shown as below:

- 1 audio 3.5 mono jack
- 2 audio 3.5 mm stereo jack (in and out)
- Onboard audio codec (TLV320AIC3254)
- Onboard Class-D audio power amplifier (TPA2012D2)

2.3 Communication in the HEARLab System

2.3.1 What is UDP Communication?

User Datagram Protocol (UDP) is part of the Internet Protocol suite used by programs running on different computers on a network. UDP is an open system interconnection (OSI) transport layer (layer 4) protocol for client-server network applications. UDP uses a simple transmission model but does not employ handshaking dialogs for reliability, ordering and data integrity [21]. UDP is an ideal protocol for network applications in which perceived latency is critical such as gaming, voice and video communications, which can suffer some data loss without adversely affecting perceived quality. In some cases, forward error correction techniques are used to improve audio and video quality in spite of some loss [22]. UDP works in conjunction with higher level protocols to help manage data transmission services including Real Time Streaming Protocol (RTSP) which accord with the communication mode of the HEARLab System.

The UDP protocol mechanism diagram shows as below. Briefly, the client and server establish connection and send their unique IP address, port, and information between each other via the OSI. The IP address and the port can be seen as the client and server's identification which used to find the correct connection port.

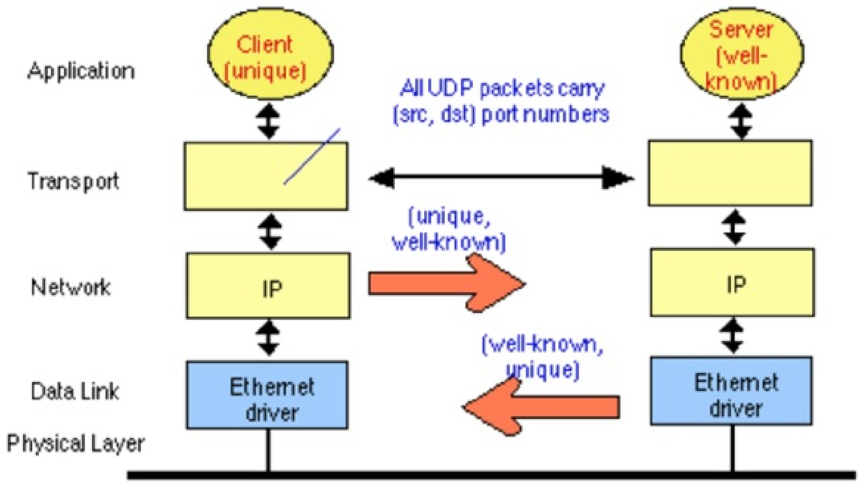


Figure 2.6: UDP protocol mechanism diagram [22]

In the communication side, the new HEARLab System based on the Wi-Fi mode to transfer the audio streaming in real time. It means the receiving end and the transmitting end should receive and send the signals simultaneously. On the other hand, UDP is an unstable communication protocol; therefore, the information packages could be lost during the transport. In the HEARLab System, the received EEG signals have a 5-10% probability of distortion. As long as the UDP packages loss probability is less or equal to 5-10%, the UDP socket method can be used in the HEARLab System.

2.3.3 What is IoT?

Internet of Things (IoT) is a big topic in nowadays, there is no an absolute definition of the IoT. A general definition is given by P. F. Fan and G. Z. Zhou, they said that IoT which bases on the Internet, uses a variety of information sensing identification device and information processing equipment, such as RFID, GPS, GIS, JIT, EDI, and other devices to combine with the Internet to form an extensive network in order to achieve information and intelligence for Entity [23]. Even though, the IoT is a big picture, the basic idea is to connect the devices to Internet. TI gives examples about how IoT used in different area.

"In the personal area network (PAN) we encounter wearable devices to enhance our health and wellness. At home we are surrounded with an ever-growing number of appliances, multimedia devices and other consumer gadgets. While on-the-go, we use private or public transportation vehicles and infrastructure to improve our mobility time utilization. In the industrial case, sensors might be introduced for production efficiency, maintenance and failure management. And at a metropolitan level, smart building management systems and infrastructure for remote management, on-going maintenance and asset tracking will be observed [24]."

As Figure 2.7, IoT utilize in a different network's: Personal Area Network (PAN), Local Area Network (LAN), Neighborhood Area Network (NAN) and Wide Area Network (WAN).

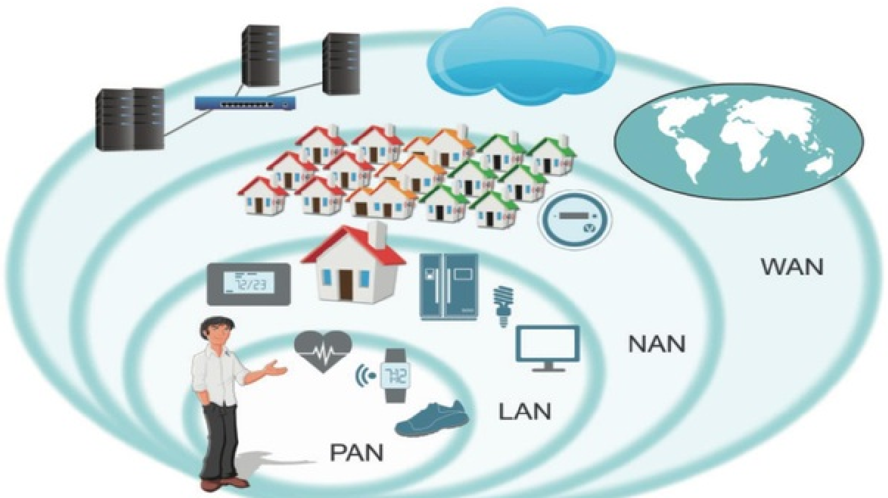


Figure 2.7: Different ranges and applications for personal, local, neighborhood and wide area networks [25].

Back to the project, the HEARLab System actually is using IoT to connect the HEARLab System to the Internet; therefore, to achieve the information wireless communication. So far, the new HEARLab System is belonging to the LAN, the information transfer in a certain local area. In the future, according to the IoT, the data and information is able to upload to the cloud and back to user's hand straightaway.

2.4 Audio Steam in the HEARLab System

2.4.1 What is PCM?

Pulse-code modulation (PCM) is a method used to digitally represent sampled analog signals. It is a standard form used in the digital audio devices, such as computer, digital telephony, and digital audio player. A PCM stream has two basic properties and three steps that determine the stream's fidelity to the original analog signal. The two properties are the sampling rate, which is the number of times per second that samples are taken; and the bit depth, which determines the number of possible digital values that can be used to represent each sample. Figure 2.8 shows the sampling rate and bit depth format combination.

Sampling rate (kHz)	bit depth (bit)	Data Size (KB/s)	
		Mono	Stereo
16	8	16	32
	16	32	64
44.1	8	44.1	88.2
	16	88.2	176.4
48	8	48	96
	16	96	192

Figure 2.8: Sampling rate and bit depth format combination

The tree steps in the PCM transmitter are: sampling, quantization, and encoding. The quantization and encoding are usually performed in the ADC.

And the tree steps in the PCM receiver are: regeneration of the impaired signals, decoding, and reconstruction of the message signal. Regeneration of the signal also occurs in intermediate point in the signal transmission path. When time division multiplexing of signals occurs it is necessary to synchronize the transmitter and receiver in time for successful multiple channel operation.

Figure 2.9 shows in the PCM stream transmission mechanism in the HEARLab System. The PCM stream has been processed by the computer's sound card and the CC3200 Audio BoosterPack's codec. In general, the Wi-Fi module uses for the information exchange. Moreover, the sound cards and the codec use as information processor.

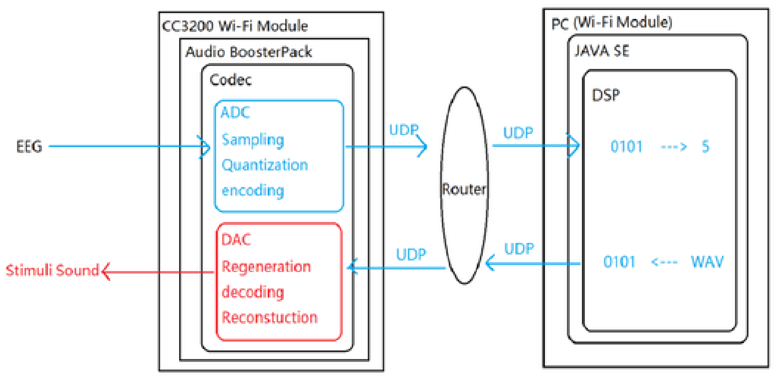


Figure 2.9: PCM stream transmission mechanism in the HEARLab System

In Figure 2.9, the codec receives the digital signals and convert it into analog signals via loudspeaker to present the sound. On the other hand, the amplified analog EEG signals go into the CC3200 Audio BoosterPack's codec and turn into the digital signals, and wirelessly transmit to the computer. Next, the computer uses the JAVA program to process the unwrap UDP packages.

2.4.2 What is DSP?

The EEG signals which received by the HEARLab System is actually the analog signal, for sake of the computer is able to processing the received signals; therefore, the DSP is necessary to be considered. DSP refers to various techniques for improving the accuracy and reliability of digital communications. The theory behind DSP is quite complex. Basically, DSP works by clarifying, or standardizing, the levels or states of a digital signal [26]. In reality, the two principal human senses are vision and hearing. Correspondingly, much of DSP is related to image and audio processing. People listen to both music and speech. DSP has made revolutionary changes in both these areas [27]. HEARLab System received the subject's brain waves by electrodes, the signals stored in the computer as the binary number. Therefore, converting the binary number to a human readable number is going to be referred in the following paper.

2.4.2 What is SNR?

In analog and digital communications, signal-to-noise ratio (SNR) is a measure of signal strength relative to background noise. SNR is given by the formula:

$$S/N = 20 \log10(V_{signal}/V_{noise})$$

Figure 2.6 shows a much-generalized experimental situation. The system could be electrical, mechanical or biological, and includes all important environmental variables such as temperature, pressure or magnetic field. The excitation is what evokes the response we wish to measure. It might be an applied voltage, a light beam or a mechanical vibration, depending on the situation. The system response to the excitation, along with some noise, is converted to a measurable form by a transducer, which may add further noise. The transducer could be something as simple as a mechanical pointer to register deflection, but in modern practice it almost always converts the system response to an electrical signal for recording and analysis [28].

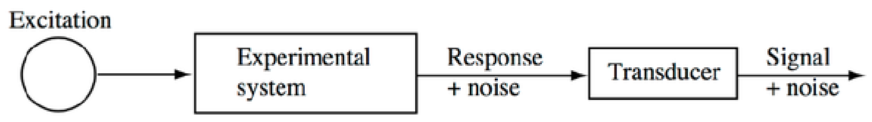


Figure 2.10: Schematic of a generalized experiment [28]

In HEARLab System, the excitation is the EEG signals. The environmental vibration or the human bodies react, such as blinking eyes, open the mouse, or move the jaw, are the noise signals. Moreover, the circuit electromagnetic (EM) interference is another factor, which causes a huge impact on the SNR.

Chapter 3

Communication between the CC3200 LaunchPad and the Computer

Introduction

On request, NAL wants to achieve the audio signals such as the received EEG and the stimuli signals is able to wirelessly transmit in real-time between the CC3200 LaunchPad and the computer. A communication plan shows as below.

1. Target the networking mode, selecting the high-efficiency networking mode.

The essential part of the new HEARLab System is the communication method, adding the CC3200 LanuchPad into a Wi-Fi environment is the way to achieve the wireless communication between the computer and the CC3200 LanuchPad. Obviously, the router is the key device to make the Wi-Fi function in a particular area, so adding the CC3200 LaunchPad to the router is the most vital step (means router dispatch a unique IP address to the CC3200 LanuchPad). After successfully adding the CC3200 LanuchPad to the Wi-Fi environment, the router becomes a bridge to connect the computer and the CC3200 LanuchPad. Figure 3.1 shows the router in this networking topology is acting as the Access Point (AP) which is called AP mode networking.

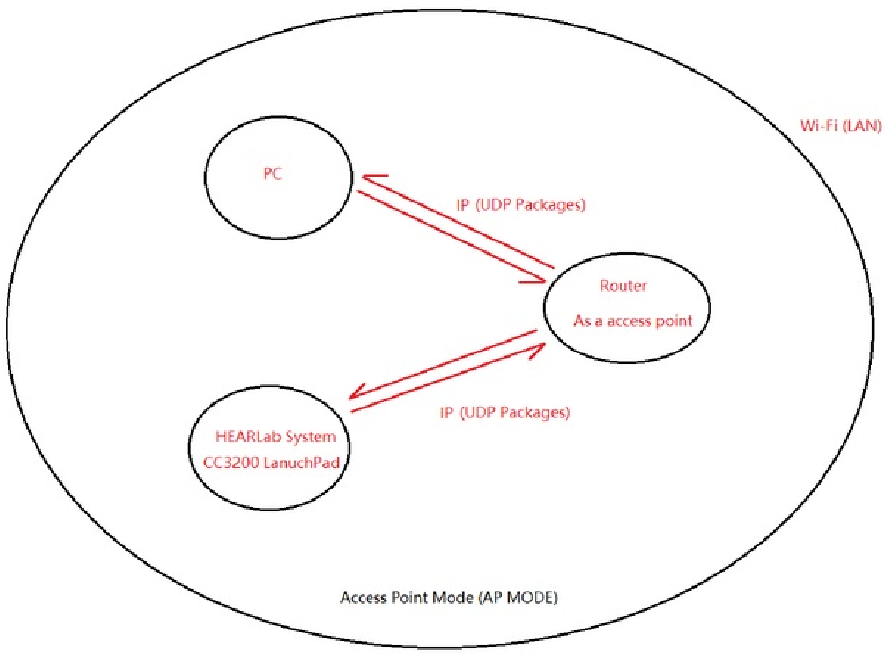


Figure 3.1: Networking topology in the HEARLab System

According to Figure 3.1, the computer sends the information (stimuli signals) wirelessly to the router; the router receives the information and sends the information wirelessly to the CC3200 LaunchPad, and vice versa.

2. Choosing the appropriate communication protocol

After settle down the AP networking mode, choosing an appropriate communication protocol becomes the second essential part. There are several popular network communication protocols such as TCP/IP, Hypertext Transfer Protocol (HTTP), Post Office Protocol (POP3), and User Datagram Protocol (UDP). Referring to the new HEARLab System, the transmit information message of the new HEARLab System is the audio stream in terms of bits; therefore, the TCP/IP and UDP cab be chosen as the communication protocol (both of them transfer the bits in the IP layer).

The HEARLab System requires the information transmission occurring in real-time, and the TCP/IP protocol is not able to achieve the real-time requirement. Hence, the UDP chose as the communication protocol, which can transfer the message with less delay, but more lost packages.

3. Selecting the user-friendly programming language

Choosing the appropriate programming language is one of the key features. For the CC3200 Launchpad, it is using the C-code as the programming language (seleted by TI). On the computer side, I decided to apply the JAVA as the development language. Figure 3.2 shows the programming language distribution.

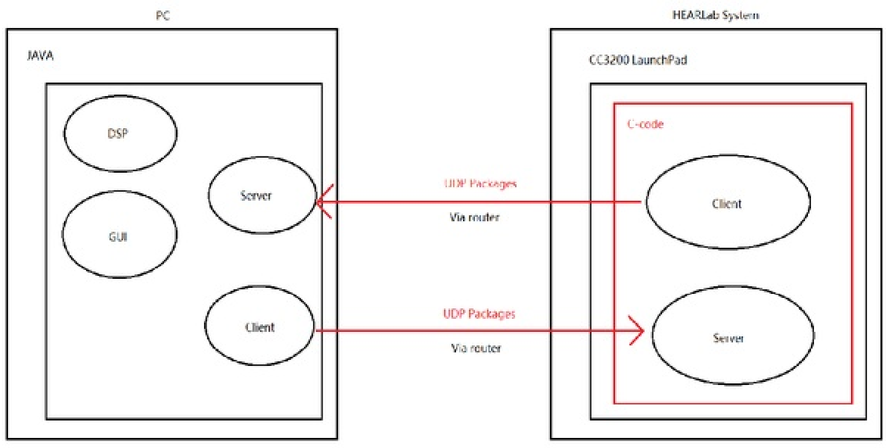


Figure 3.2: Programming language distribution in the HEARLab System

In the case, JAVA is going to use to achieve the communication, DSP, GUI and other functions, which based on this project's requirements.

In the following paragraphs, how to use the CC3200 LanchPad to transmit/receive the information, how to use java to write the computer information transmit/receive the program, and DSP process in the memory of computer are going to describe in detail.

Equipment: CC3200 Launchpad, CC3200 audio booster pack, Wi-Fi router, 3.5mm male-male jack, 3.5 mm stereo earphone, computer with JAVA SDE.

3.1 CC3200 LanuchPad C-code program

To start with, install the CCS, CC3200 SDK, and Uniflash that mentioned in the section 2.2.2, then following the TI user Handbook [29] to manipulate the CC3200 LanuchPad. The sketch of the steps shows as below:

1. Open the CCS and find the "wifi_audio_app" SDK project file (Figure 3.2), modifying the SDK project files to achieve the goal.

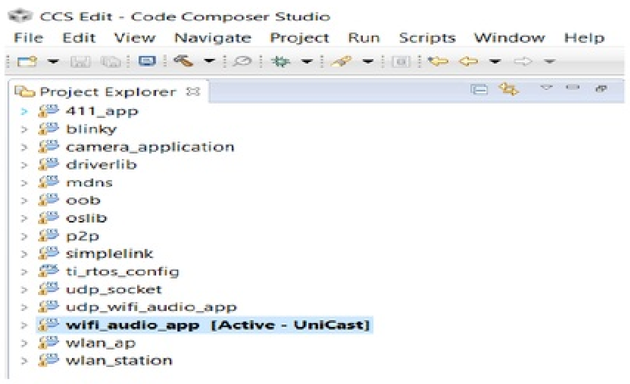
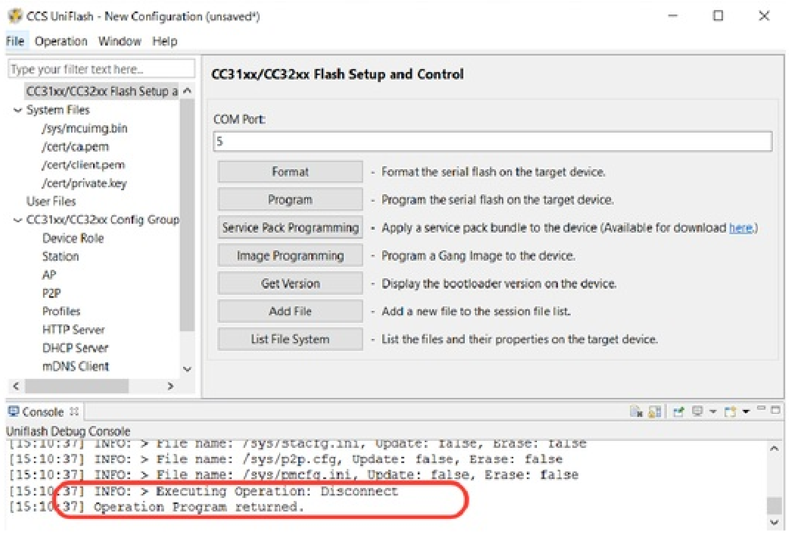


Figure 3.3: the wifi_audio_app in the Code Composer Studio

2. Build and debug the project file, then using Uniflash to upload the project file to the CC3200 MCU. The successful upload shows in Figure 3.3.



CC31xx/CC32xx Flash Setup and Control

COM Port: 5

Format	- Format the serial flash on the target device.
Program	- Program the serial flash on the target device.
Service Pack Programming	- Apply a service pack bundle to the device (Available for download here.)
Image Programming	- Program a Gang Image to the device.
Get Version	- Display the bootloader version on the device.
Add File	- Add a new file to the session file list.
List File System	- List the files and their properties on the target device.

```
[15:10:37] INFO: > File name: /sys/stacfg.ini, Update: false, Erase: false
[15:10:37] INFO: > File name: /sys/p2p.cfg, Update: false, Erase: false
[15:10:37] INFO: > File name: /sys/pmcfg.ini, Update: false, Erase: false
[15:10:37] INFO: > Executing Operation: Disconnect
[15:10:37] Operation Program returned.
```

Figure 3.4: Successful upload the code on the CC3200 LanuchPad

Mention that, owing to this project itself contains vast of contents, in here the CC3200 LanuchPad SDK modified code is not going to give unnecessary details. The modified “wifi_audio_app” SDK project file is going to upload to the Dropbox (Appendix A), providing convenience for the developers who wants to achieve the same function (communicating between the CC3200 LanuchPad and the computer).

3.2 Computer communication program

As the CC3200 LanuchPad has been well prepared, programming the Client and Server on the computer by using JAVA is going to be another task.

The Client/Server UDP JAVA program established, referring the JAVA open source code on the Internet [30], and Humphry Qin’s eager help. A brevity steps to describe how to programming a UDP Client/Server program in any programming language. Figure 3.5 provides a demonstration of the UDP communication mechanism.

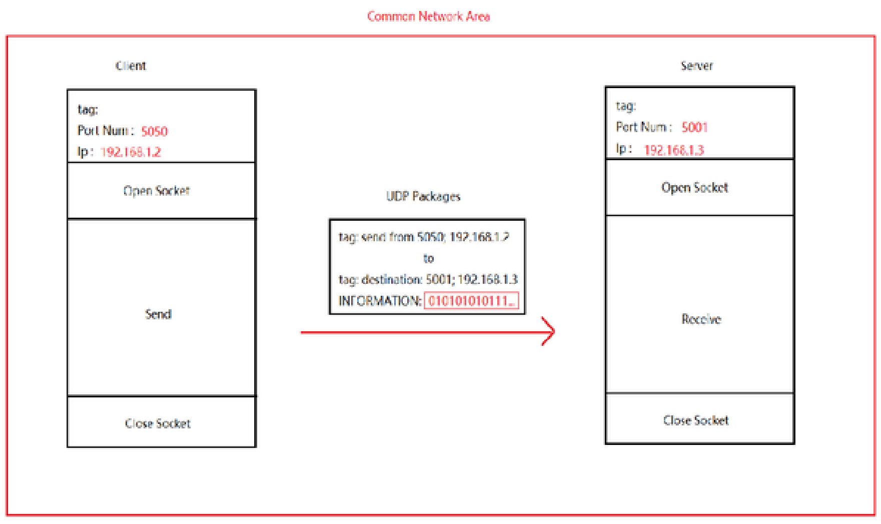
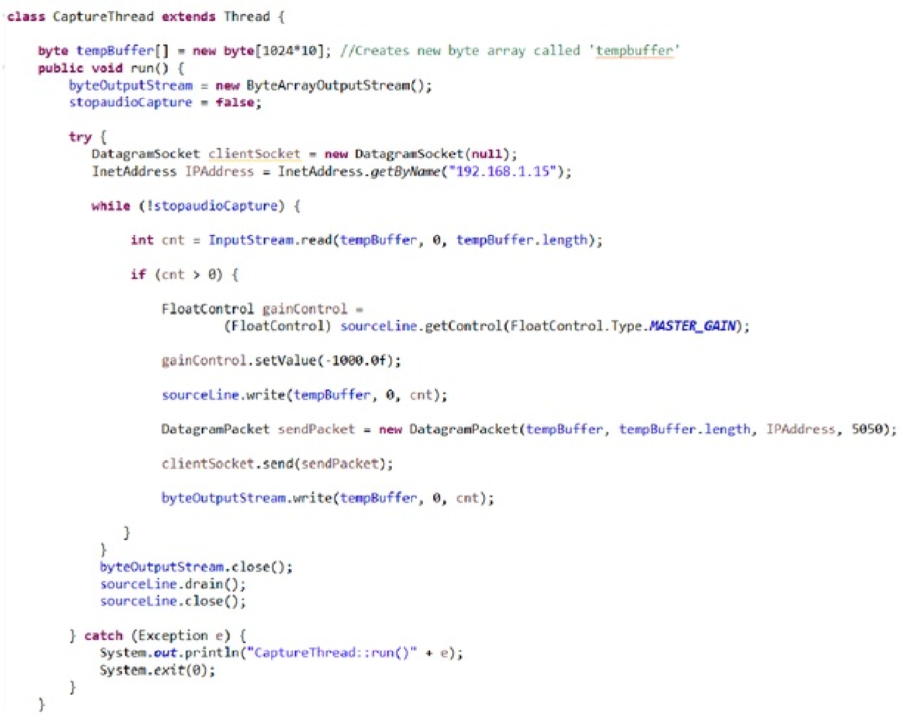


Figure 3.5: UDP communication mechanism

Create the UDP Client/Server following the four steps:

1. Open a socket. In UDP programming the socket can be seen as an invisible channel to communicate with two devices, which want to process the information exchange.
2. Create a port. The port in UDP programming closely connects with the IP address. On the Internet, if two devices want to communicate with each other through the Internet. IP address and ports are the two vital elements. For example, the socket is a highway. And the ports are the highway entrances.
3. Send/Receiver UDP package. In the computer, the information transmission is in the form of bits, whatever the sound, image or video. The sending and receiving UDP packages is like the driving cars (bits) on the highway.
4. Close the socket. This step is not always necessary; however, it plays an important role under certain circumstance.

Figure 3.6 is the UDP communication code, the integrated code upload to the Dropbox (Appendix A) either.



```java
class CaptureThread extends Thread {

    byte tempBuffer[] = new byte[1024*10]; //Creates new byte array called 'tempbuffer'
    public void run() {
        byteOutputStream = new ByteArrayOutputStream();
        stopaudioCapture = false;

        try {
           DatagramSocket clientSocket = new DatagramSocket(null);
           InetAddress IPAddress = InetAddress.getByName("192.168.1.15");

           while (!stopaudioCapture) {

                int cnt = InputStream.read(tempBuffer, 0, tempBuffer.length);

                if (cnt > 0) {

                    FloatControl gainControl =
                            (FloatControl) sourceLine.getControl(FloatControl.Type.MASTER_GAIN);

                    gainControl.setValue(-1000.0f);

                    sourceLine.write(tempBuffer, 0, cnt);

                    DatagramPacket sendPacket = new DatagramPacket(tempBuffer, tempBuffer.length, IPAddress, 5050);

                    clientSocket.send(sendPacket);

                    byteOutputStream.write(tempBuffer, 0, cnt);

               }
           }
           byteOutputStream.close();
           sourceLine.drain();
           sourceLine.close();

        } catch (Exception e) {
           System.out.println("CaptureThread::run()" + e);
           System.exit(0);
        }
    }
```

Figure 3.6: UDP communication code in JAVA

3.3 Communication mechanism

In this project, based on IoT, the communication mechanism explaining in the following paragraph.

To start with, the CC3200 LanuchPad and the computer join the same Wi-Fi. The router dispatches a unique IP address to the CC3200 LanuchPad and the computer. Then the client (whatever the CC3200 LanuchPad or the computer) send their identification message (port number and IP address) and useful information to the other device's server. It is worth mentioning that the sending and receiving UDP packages happens simultaneously on the CC3200 LanuchPand and the computer.

In this communication mechanism, the information in some circumstance is not able to transmit perfectly, in another word such as missing UDP packages, internet blocking, and interference from electronic devices could cause these problems. My partner, Thomas, is going to talk more about the Wi-Fi interference in his thesis.

Chapter 4

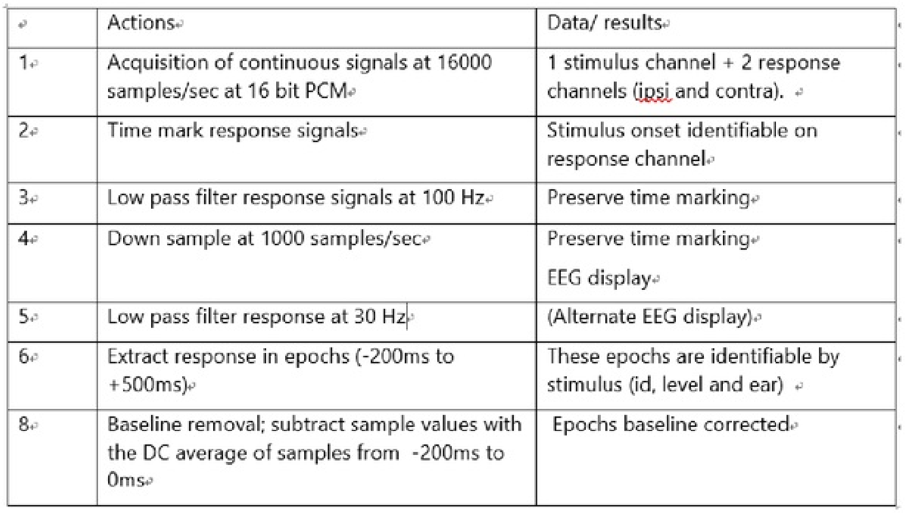
DSP, Signal Display, and GUI

Introduction

In the software side, Thomas and I decided to separate out jobs. For me, I am going to use JAVA to design a program, which able to reach the HEARLab System requirements to do the real-time EEG display. Moreover, Thomas is going to use Matlab to do the offline EEG analysis.

Typically, the software engineering uses the computer program language to achieve certain goals. In the following paragraph, using JAVA to achieve the requirements of the HEARLab System is going to be introduced.

The software requirements show in Figure 4.1:



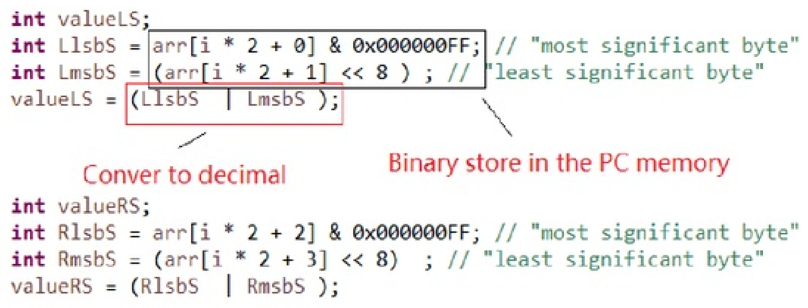
	Actions	Data/ results
1	Acquisition of continuous signals at 16000 samples/sec at 16 bit PCM	1 stimulus channel + 2 response channels (ipsi and contra).
2	Time mark response signals	Stimulus onset identifiable on response channel
3	Low pass filter response signals at 100 Hz	Preserve time marking
4	Down sample at 1000 samples/sec	Preserve time marking EEG display
5	Low pass filter response at 30 Hz	(Alternate EEG display)
6	Extract response in epochs (-200ms to +500ms)	These epochs are identifiable by stimulus (id, level and ear)
8	Baseline removal; subtract sample values with the DC average of samples from -200ms to 0ms	Epochs baseline corrected

Figure 4.1: Software requirements

4.1 EEG signals DSP in real-time

The analog EEG signals amplified by the electrode processor (section 1.1) goes into CC3200 Audio BoosterPack's codec. Then the codec via sampling, quantization, and encoding to convert the EEG signals to digital signals. The CC3200 LanuchPad Wi-Fi module wraps up the digital signals and though UDP packages send to the computer. The digital signals that received by the computer show in the form of bits. At the same time, JAVA program processes the following steps to display the signals.

1. Receive the UDP package. Create a buffer to contain the received bits. The buffer acts as a container, and the UDP packages act as the water. Sending the UDP packages to the buffer is like pouring the water into the container. If the water (UDP) fill the container (buffer) too fast, the overflowing can occur. In this case, the buffer should have a function to drain the water.
2. Convert the binary bits to decimal. The results must show in decimal. (Figure 4.2 is JAVA code converts the binary to decimal)



```
int valueLS;
int LlsbS = arr[i * 2 + 0] & 0x000000FF; // "most significant byte"
int LmsbS = (arr[i * 2 + 1] << 8 ) ; // "least significant byte"
valueLS = (LlsbS  | LmsbS );

int valueRS;
int RlsbS = arr[i * 2 + 2] & 0x000000FF; // "most significant byte"
int RmsbS = (arr[i * 2 + 3] << 8)  ; // "least significant byte"
valueRS = (RlsbS  | RmsbS );
```

Figure 4.2: JAVA code convert the binary to decimal

3. Display the EEG waveform. JAVA Swing is the lib to display the graph and make the GUI. To make the signals display in the real-time, GUI is the vital part. More detail is going to introduce in Chapter 6.

Before demonstrating the real-time EEG, step 2 needs to emphasize. In the communication system, the information of the UDP packages is the audio stream. Audio stream has different audio channels such as the stereo sound including two channels: channel one and channel two. Also, the mono sound contains one audio channel. In general, Figure 4.3 gives the localization of the channels in an audio stream.

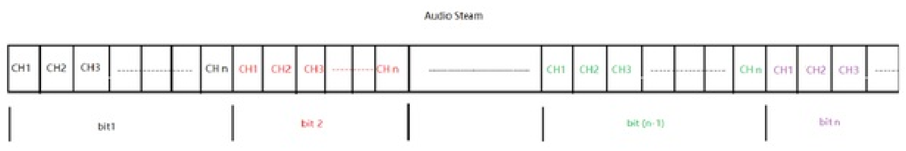


Figure 4.3: Localization of the channels in an audio stream

Therefore, in JAVA, the program needs a block of code to split the channel, making the channel in order such as Figure 4.4. Otherwise, the information is going to be disordered. For the user, it appears the user is going to hear a meaningless sound.

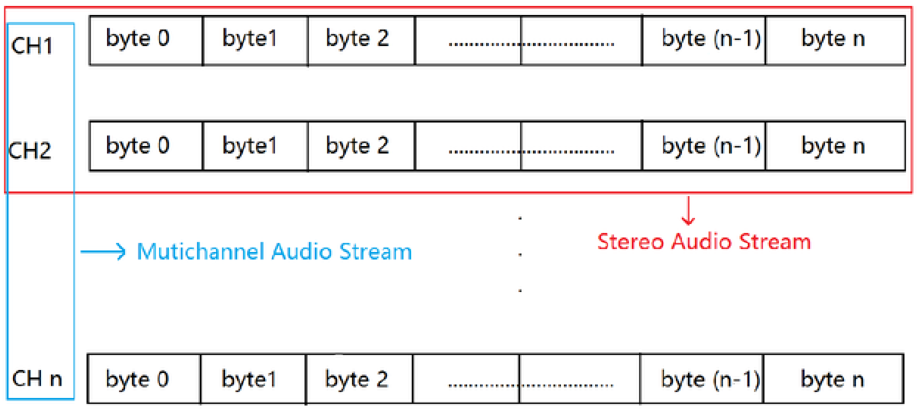


Figure 4.4: Channel and bytes distribution in the computer's memory

In step 2, Audio stream format is a key point to encode or decode an audio stream. As section 2.4.1, the audio stream has the sampling rate, such as 16 kHz, 48 kHz and 44.1 kHz (CD quality). The bit depth can be 16 bits, 32 bits, and 48bits, etc. In this project, the HEARLab System deal with the audio stream with 16 kHz sampling rate and 16 bits depth. Therefore, referring to Figure 4.4 the program converts two bytes to a PCM value. Figure 4.5 shows how the program convert the byte (binary) to PCM value (decimal).

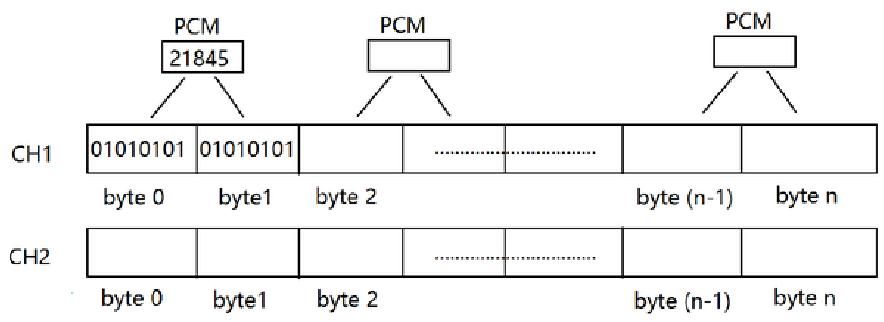


Figure 4.5: Program covet the byte (binary) to PCM value (decimal)

Step 3 is to use the JAVA Swing lib to display the audio stream. After step 2, the program got the audio stream in the form of PCM. Then, as Figure 4.6 shows the y-axis is the PCM value in decimal and the x-axis is the time in second.

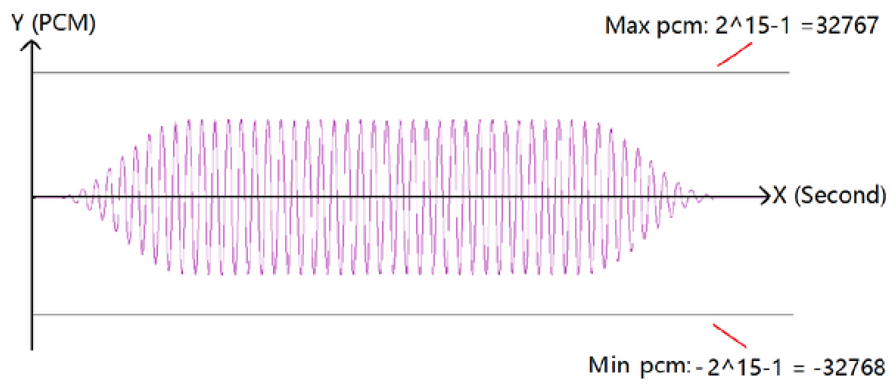


Figure 4.6: Audio stream in the form of PCM

Next, to make the JAVA dynamic plots the audio stream, and providing a friendly user interaction. Using the JAVA graphs 2D lib provide an animation, which able to “run” the audio stream. Figure 4.7 gives the idea to move the audio stream.

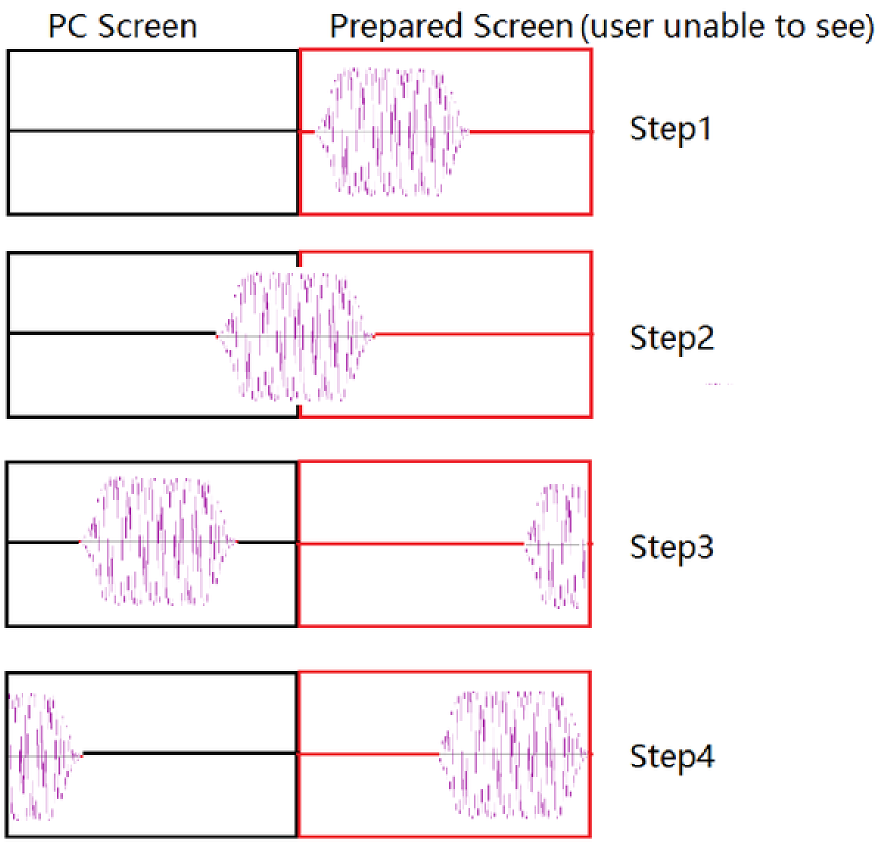


Figure 4.7: the method of dynamic plots audio stream

In Figure 4.7, it shows the audio stream plots in a prepared screen, which user unable to see it. Then, the JAVA program invokes the graph 2D function to repaint the graph on the screen in per millisecond. After that, following the buffer continuously receive the UDP packages. The user sees the audio stream dynamic displays on the screen. Moreover, the program discards the out of screen data to save the program's memory.

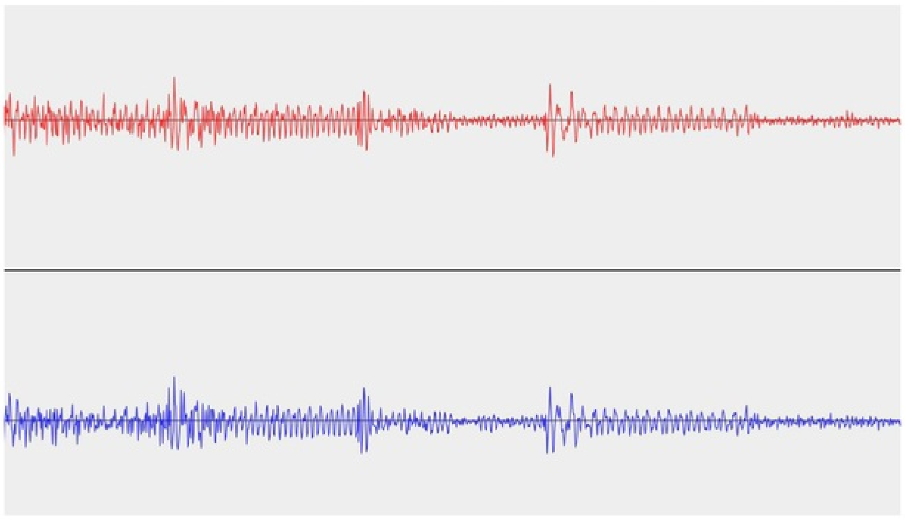


Figure 4.8 A music waveform display in the JAVA program.

Figure 4.8 shows a music waveform display in the program. The program shows the music waveform in real-time. As the project goes on, the music is going to replace by the EEG response signals and stimuli signals.

4.2 Automatic onset detection

Onset detection is the most important part of the HAERLab System without this process the whole system is not able to progress. Next, the principle of the onset detection is going to introduce in detail.

I. What is onset detection?

Note onset detection and localization is useful in some analysis and indexing techniques for signals. Every sound has its specific waveform, however, in a series audio stream the waveform displays in a continuous format. The onset detection is used to find the certain pattern from a serious audio stream.

For example, Figure 4.9 shows a series stimuli signals. The HEARLab System wants every time the software detect the stimuli signal's pattern, meanwhile, the computer cut an epoch (epoch means one of the captured EEG). The zoom in stimuli signal's pattern shows in Figure 4.9.

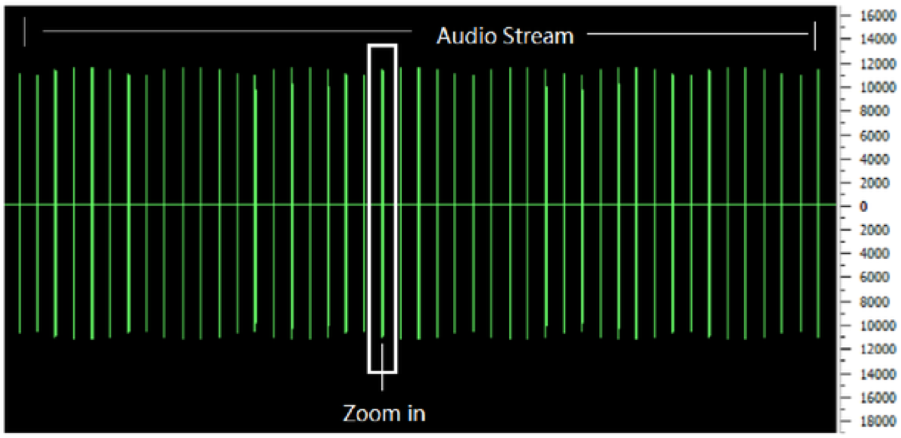


Figure 4.9: A series stimuli signals

In this case, the onset detection uses to find the beginning and ending point of every stimulus (Figure 4.10 draws the starting and ending point).

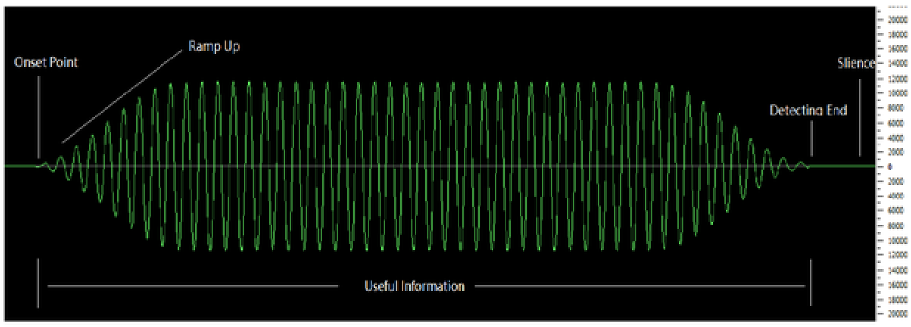


Figure 4.10: The zoom in stimuli signal's pattern

Audio stream waveform contains many noises. In this case, a serious algorithm needs to be used to detect the wanted waveform pattern and discard the unwanted noise.

II. Onset detection realizing in JAVA

To start with, building a mathematical model to detect the stimuli signal's pattern. I used three steps to build the mathematical model (Figure 4.11).

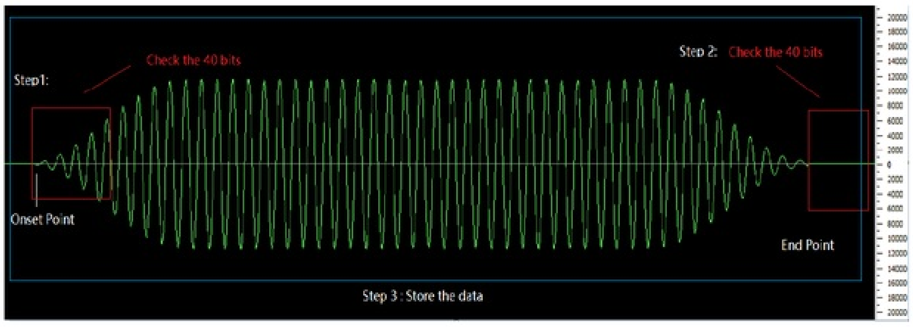


Figure 4.11: how to detect the stimuli signals

1. Find the Onset Point. In the program, a detecting point following the audio stream checks the received data byte by byte. To avoid the noise interference, the detecting point is going to check the 40 bits (Figure 4.11 front-end red square) after found the onset point. If the 40 bits match 80% of the wanted pattern, then store the detected onset point and keep checking the rest data.
2. Find the ending point. After found the onset point, the program is going to find an ending point at somewhere. Therefore, the detecting point keeps detecting the entire data until finding the ending point. Same principle, to avoid the fake ending point, the program is going to check the after 40 bits (Figure 4.11 tail red square). either. In this case, the ending point is not able to accept if the 40 bits pattern matches 80% of the noise structure.
3. Store the detected stimuli signal in a buffer. An integrated stimuli signal is the interval between the onset point and the ending point. Next, store the stimuli signal in a buffer. After, the program is going to use the stimuli signal as a reference to cut the epoch.

III. Onset detection in real-time

The most challenge part is to achieve the real-time onset detection. The idea is to create a buffer to detect the received UDP packages. In addition, the buffer is able to do the onset detection and receive the UDP packages at the same time. After onset detection, the detected data is going to be recycled by the recycling mechanism. The simulated diagram in Figure 4.12.

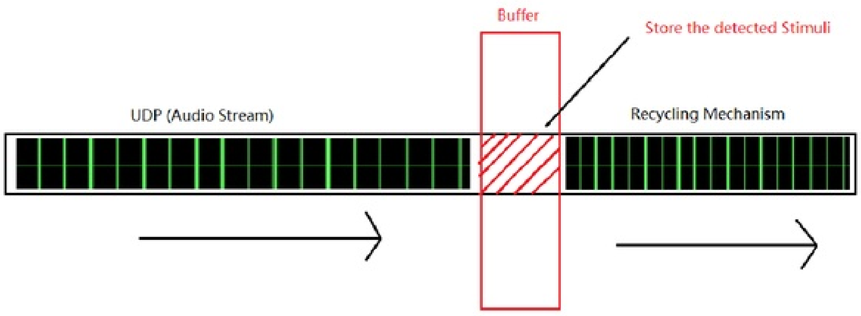


Figure 4.12: the real-time onset detection mechanism

The mechanism of the buffer is very complex. There are two main reasons:

1. The sending/receiving UDP rate. In communication, it would be best if the sender and receiver have the same sending/receiving information frequency. However, the UDP is an unstable communication mechanism; in this case, losing UDP package is able to occur the mismatched frequency between the sender and receiver. In another word, over speeding or too slow sends the UDP packages is able to cause the overflow or missing message to the receiver.
2. Time shifting. Time shifting is the stimuli signal occasionally out of the buffer, which means part of the stimuli signal is not in the buffer. To keep the accuracy of the onset detection, the buffer momentarily stores the uncompleted stimuli signal and waiting for the next part stimuli signal coming. Then merge the two parts stimuli signal together and store it into a buffer. (Figure 4.13)

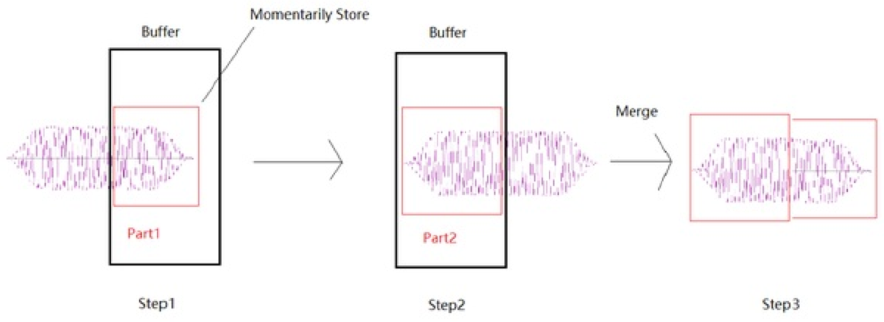


Figure 4.13: Time shifting

3. Screening wanted stimuli signal. During the testing, the noise signals misleading the program to cut the epoch. Therefore, the buffer has to get rid of the noise signals. The buffer needs to compare the detecting signal with the wanted stimuli signal's pattern. If the latency of the detecting signal is shorter than the desired stimuli signal's pattern, the buffer will not store the signal, vice versa. (Figure 4.14)

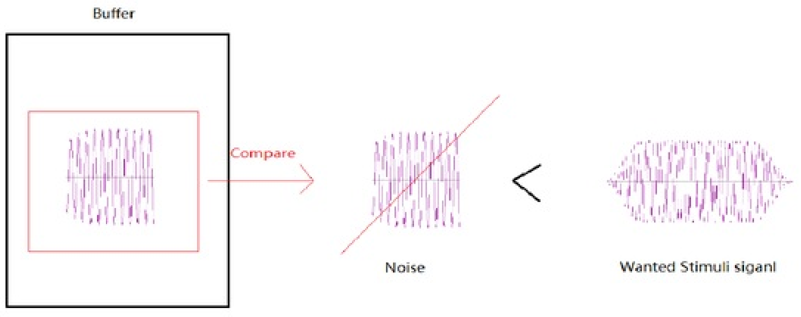


Figure 4.14: Screening wanted stimuli signal

4.3 Cutting the epochs

Cutting the epochs concurrently happens with the onset detection. In this section, how the program cut the epochs is going to be introduced.

The brain produces the EEG response signals after the subject hearing the stimuli signals. Therefore, the program using the stimuli signals as the reference to cut the recording EEG response signals. The every cutting EEG response signal is called the epoch. According to the research, human's brain hears a sound and generate the EEG signals. These EEG response signals always occur after the provided stimuli signals. In another word, the EEG response always delays a short period, which in a nanosecond, after the ear hears the stimuli sound. This phenomenon is called time-locked responses.

I. The mechanism of cutting epochs

Figure 4.15 illustrate that the program uses the stimuli signal as a reference to cut the EEG response signal. As section 4.2 mentioned, the program finds and store a stimuli signal in a buffer. In this stage, the program takes this stimuli signal's onset starting point as a benchmark point to cut a 700 ms long EEG response signal.

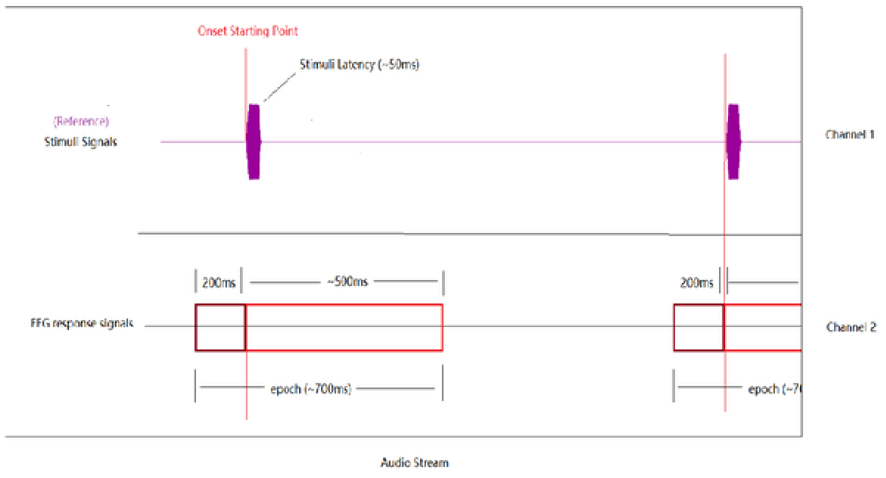


Figure 4.15: JAVA uses the stimuli signals as a reference to cut the EEG

Overview, the stimuli signals, and EEG response signals separate stored in two channels in an audio stream. Channel 1 is the reference channel, which helps the program to cut the epoch. Channel 2 is the information channel, which stores the subject's hearing status. The program dynamically detects the stimuli signal, meanwhile cutting and saving the epoch. The next stage is going to average the saved epoch to reduce the noise signals level.

4.4 Baseline correction & Averaging

I. Baseline correction

Baseline correction is the method to retain the epochs that have no peak clipping. The method is to subtract sample values with the DC average sample from -200ms. Figure 4.16 shows an EEG response signal with a certain level DC shift. The DC shift causes by the electronic devices, which giving a DC bias to the entire system. Therefore, using the program to eliminate the DC shift.

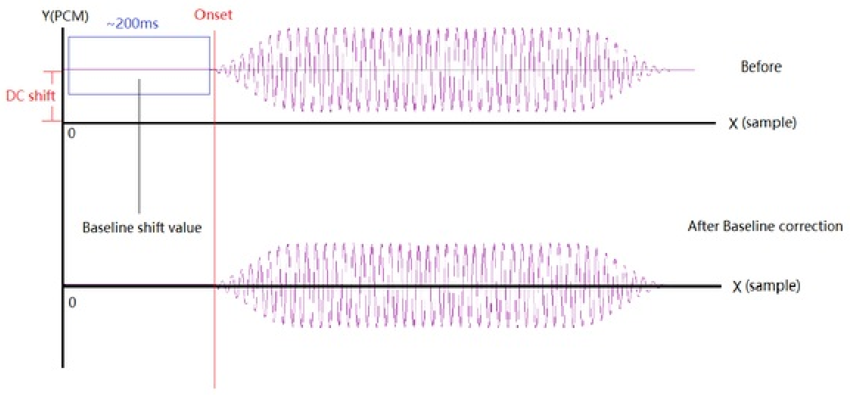


Figure 4.16: An EEG response signal with a certain level DC shift

II. Averaging the epoch

Averaging the epoch uses for eliminating the noise signals and stand out the EEG response signals. Figure 4.17 shows the EEG response with the random noise. The EEG response is a time-locked response, which arises with the similar pattern. However, the noise appears with a low amplitude and disordered pattern.

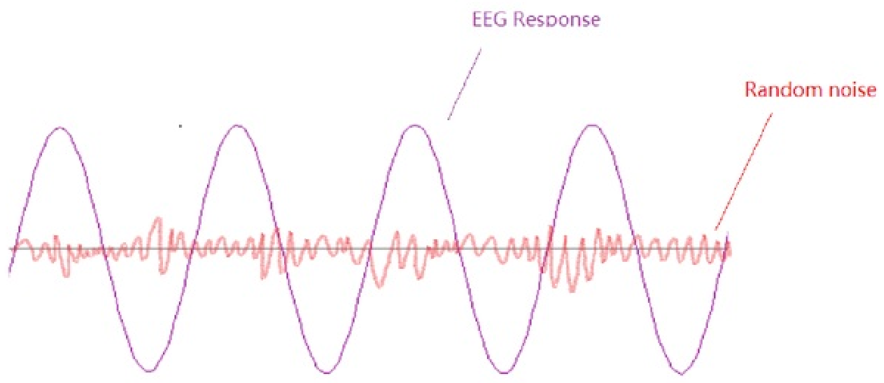


Figure 4.17: EEG response with the random noise

Adding all the epochs together and averaging the EEG response and noise. In the end, the EEG response is going to stand out, and the noise will disappear. The procedure shows in Figure 4.18.

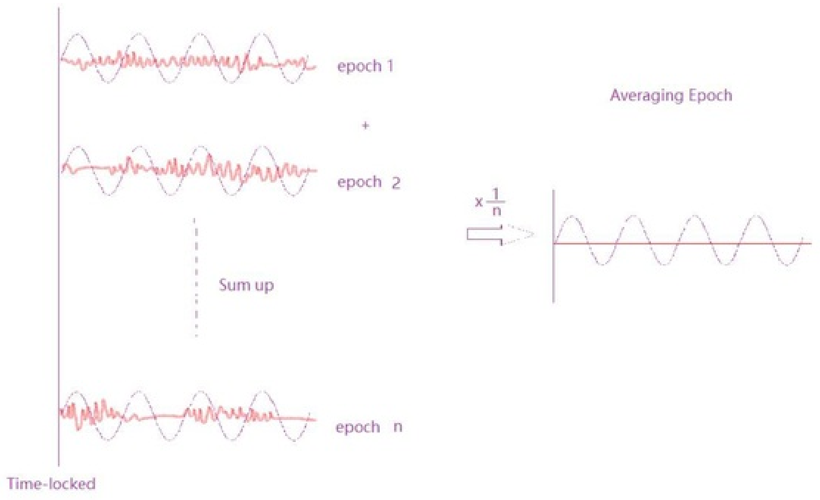


Figure 4.18: Time-locked averaging

In this project, the procedures have been refined comparing with the NAL present

HEARLab System. Moreover, the entire JAVA-code uploaded to the Dropbox [Appendix A].

Chapter 5
Wireless HEARLab System Prototype

Introduction

The prototype was designed by Loi Teck and built by Clinch, Barrier. Clinch, Barrier followed the CC3200 LaunchPad structure handbook manufacturing the printed circuit board (PCB), mounted the CC3200 LanuchPad on the PCB. The prototype shows as below Figure 5.1.

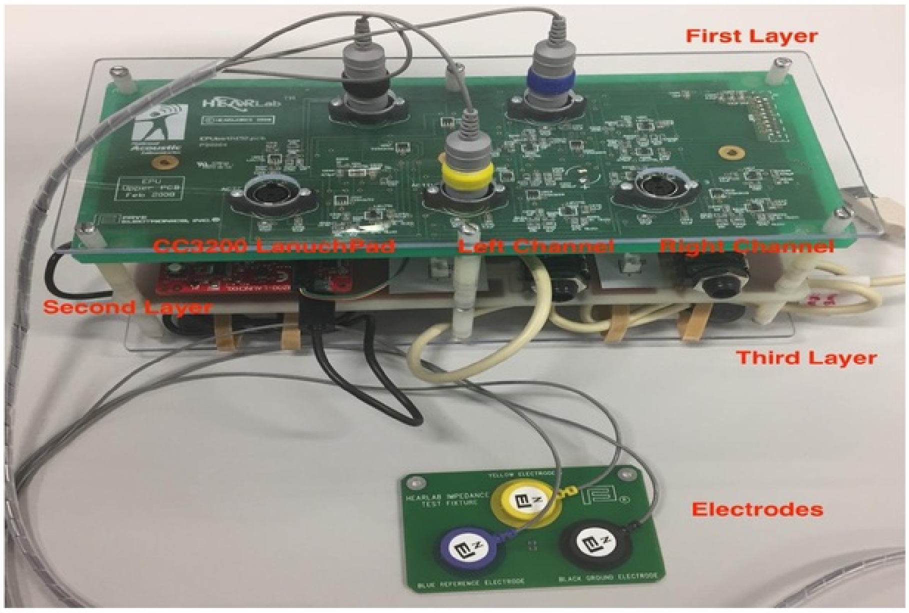


Figure 5.1: Wireless HEARLab System prototype

In Figure 5.1, the first layer is the Electrode Processor Unit (EPU), which is the inner circuit of the Figure 1.1 electrode processor. It is a great motherboard, which contains regulators, buffers, and four inverting amplifiers. The second layer is the CC3200 LaunchPad, which mounted on a PCB. On the right side of the CC3200 LanuchPad are the two loudspeakers out channels, which present the stimuli sound to the subjects. The third layer is the battery, which supplies the power to the whole device.

Below the HEARLab System is the electrodes. During the testing, the electrodes placed on the subject's scalp. The yellow node is the active channel, and the blue node is the reference channel. The black node is the ground channel.

The wireless HEARLab System prototype's three-views shows in Figure 5.2.

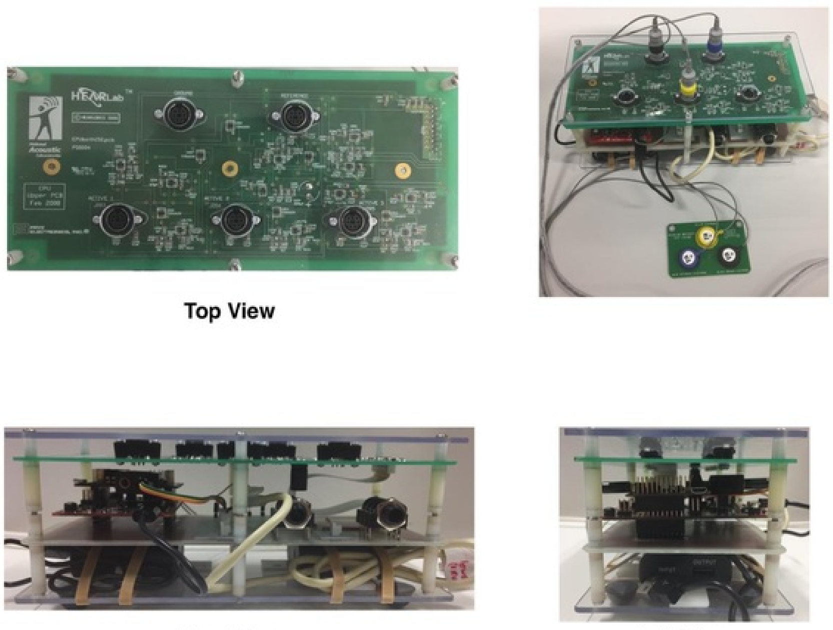


Figure 5.2: The wireless HEARLab System prototype's three-views

5.1 Working flow

Figure 5.3 shows the working flow diagram of the entire new HEARLab System prototype.

1. Start the computer JAVA program, and the computer sends the stimuli sound through the router to the CC3200 LanuchPad Wi-Fi module.
2. The CC3200 Audio BoosterPack processes the stimuli sound and separates into two branches. One of the stimuli sound goes to the loudspeaker presenting to the subject. The other though the HEARLab System second layer (PCB) goes into the first layer (EPU).
3. At the same time, the subject hears the stimuli sound and produces the EEG response signals. The EEG signals are captured by the electrodes, and with the stimuli signals merge into one audio stream. In this case, the EEG response takes the 1st channel (left channel), and the stimuli sound takes the 2nd (right channel).
4. The audio stream via line in loops back to the CC3200 Audio BoosterPack and wirelessly transmit to the computer.

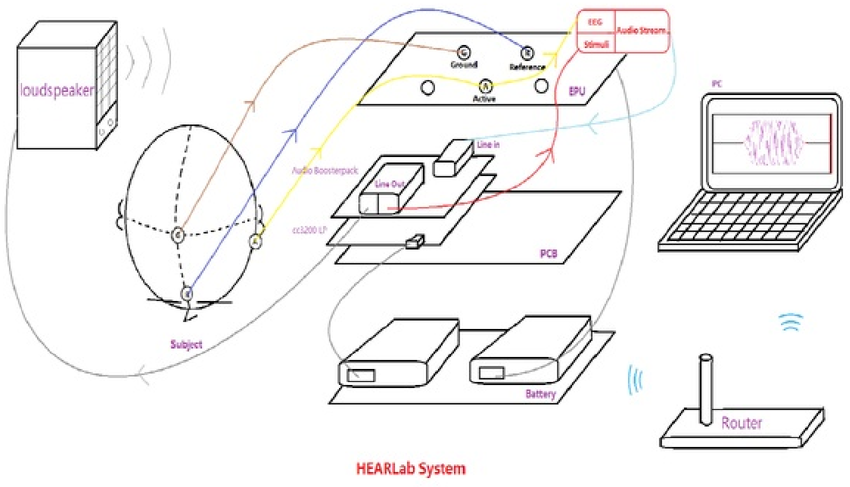


Figure 5.3: Working mechanism of the entire new HEARLab System prototype

5.2 Measurement

In Chapter 3, 4 it introduced the working principle of the new HEARLab System. After everything well prepared, the next stage aims to test the working condition of the HEARLab System. My partner, Thomas, used the Matlab to compare the NAL original HEARLab System recording with the new HEARLab System recording. Figure 5.4 shows Thomas is testing himself.

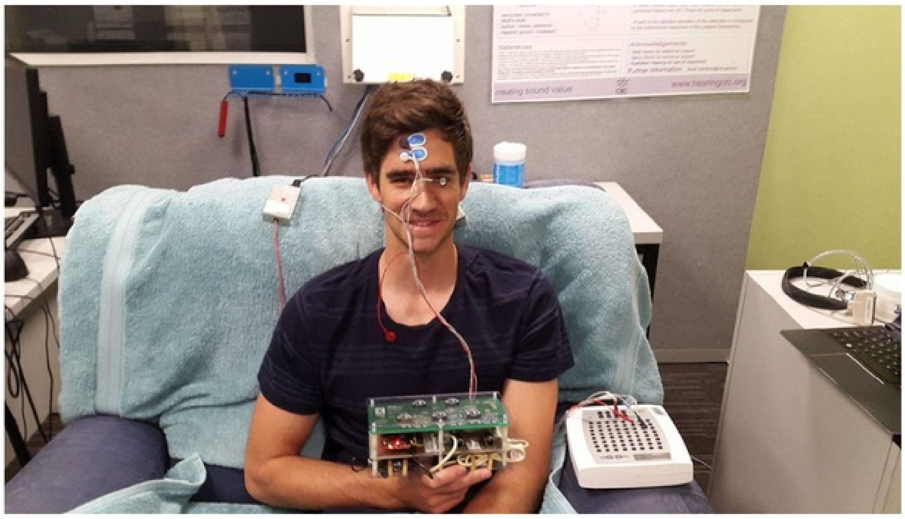


Figure 5.4: Thomas is testing the HEARLab System prototype

Figure 5.5 is Thomas's EEG response, which was captured by the original HEARLab System. Two recordings have been recorded during the experiment.

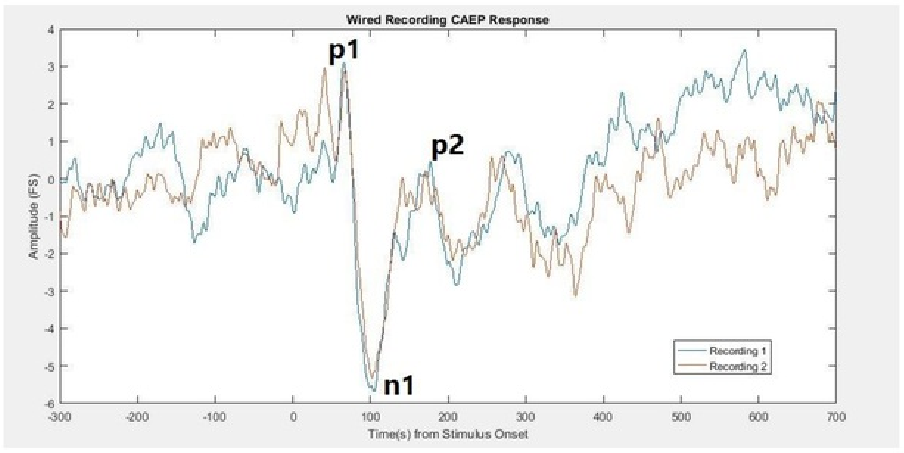


Figure 5.5: EEG response captured by the original HEARLab System

Figure 5.6 is Thomas's EEG response signals, which were captured by the wireless HEARLab System prototype. Three recordings have been recorded during this experiment.

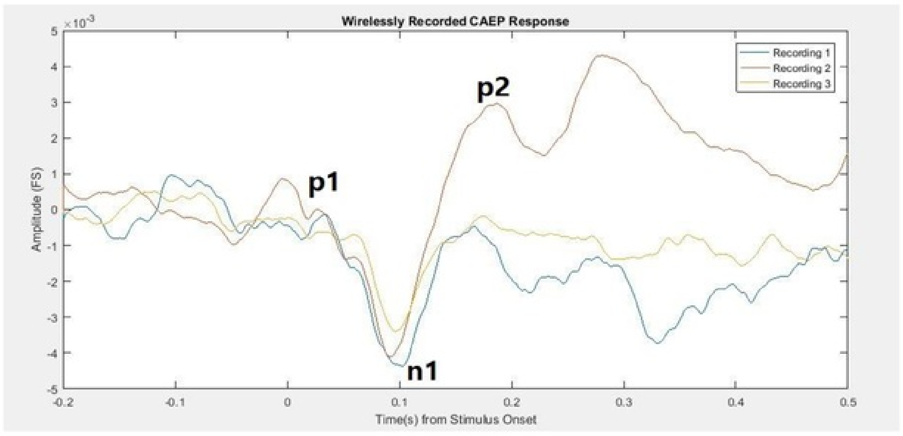


Figure 5.6: EEG response captured by the wireless HEARLab System prototype

In comparison, Figure 5.5 shows a significant EEG response signals, P1 point in Figure 5.5 is much higher than the P1 point in Figure 5.6. In addition, the two graphs share the similar n1 and p2 points.

As a result, it proved that the wireless HEARLab System prototype works properly, except certain interference. The interference reduces the intensity of the received signals such as the p1 point. Next chapter section is going to demonstrate the interference appeared in the real-time signals transmission.

Chapter 6

Real-Time Wireless HEARLab System Prototype Demonstration

In chapter 5, it has proved that the wireless HEARLab System prototype works well. Therefore, the next stage is to test the system in real-time. Following the procedure to prepare the HEALab System.

The user interface shows in Figure 6.1. The top row is the averaging EEG response display frame. The middle and bottom frames are the reference channel in red and the EEG response in blue, respectively. When the user uses the system, the first step is to follow the left column to enter the CC3200 LaunchPad’s IP address and select the different stimuli signals. The second step is to process the test. After the test finish, the user is able to stop the system or exit. The unit in the y-axis is in full scale. For example, the ADC is using 16 depth bit format, thereby 2^16 = 65536 = full scale (1.0).

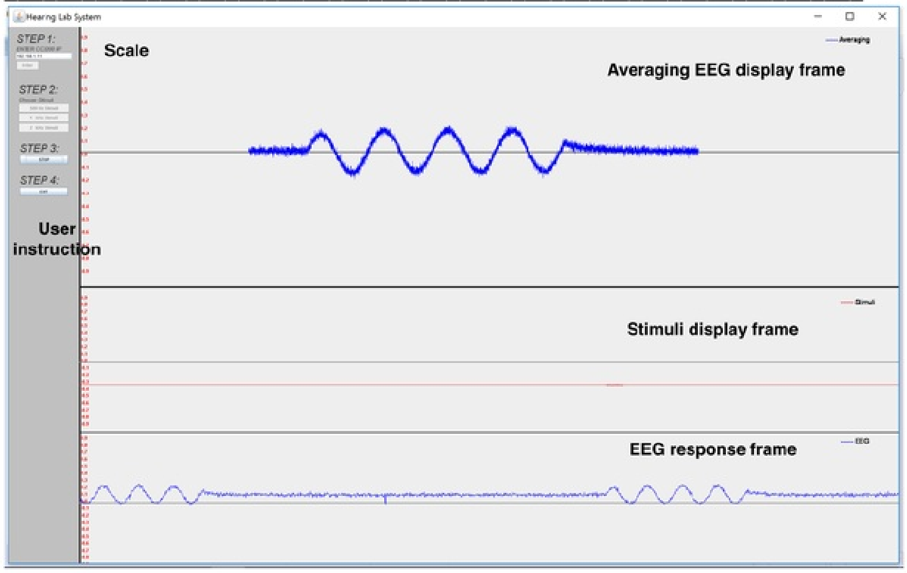


Figure 6.1: HEARLab System user interface

The next step is to test the feasibility of the real-time signal processing. In the simulation test, the main principle is to use the function generator to simulate the human brain's active signals. To set up the experiment equipment following the next four steps:

1. Connecting the CC3200 LanuchPad's louder speaker line out to the Hearlab trigger box (Figure 6.2).
2. Connecting the trigger out from the Hearlab trigger box to the function generator's trigger in port.
3. Linking the function generator's trigger out to the wireless HEARLab System prototype's electrodes.
4. Setting the function generator in N cycle mode.
5. Chang the function generator amplitude and frequency to observe the EEG response.

Figure 6.2 is the set of the equipment.

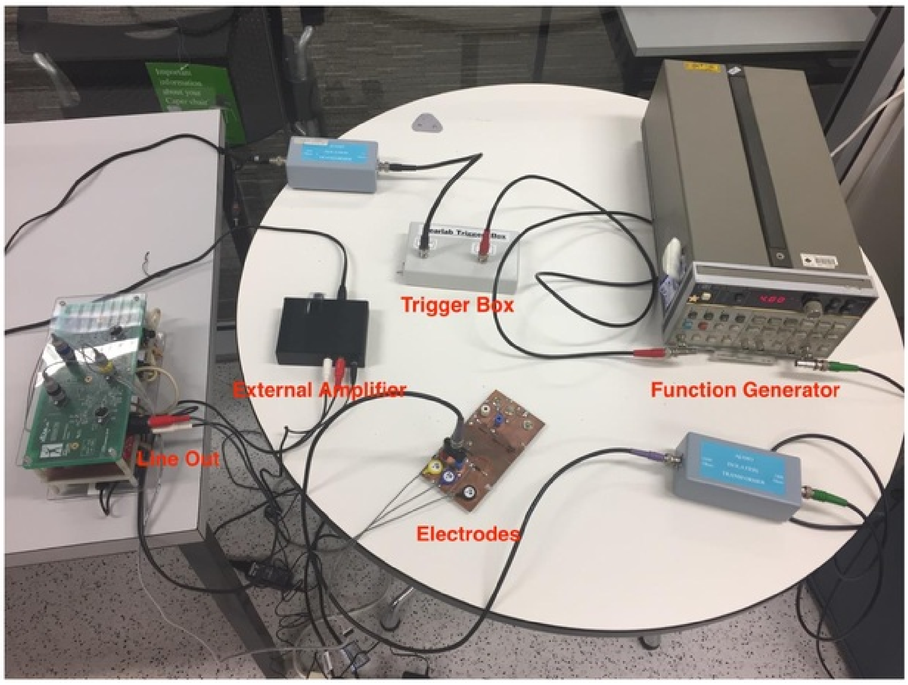


Figure 6.2: HEARLab System testing equipment set up

The result shows in Figure 6.3. The stimuli display frame shows the stimuli sound, and the EEG response frame shows the simulated brain response. The sin waves are produced by the function generator, which is triggered by the stimuli signal. The stimuli signal and the stimulate brain response are time locked, which turns up simultaneously. Comparing with the Figure 4.13 (Chapter 4), the theory and the practical software matched very well. In this case, the program is able to use to detect and process the real-time EEG response.

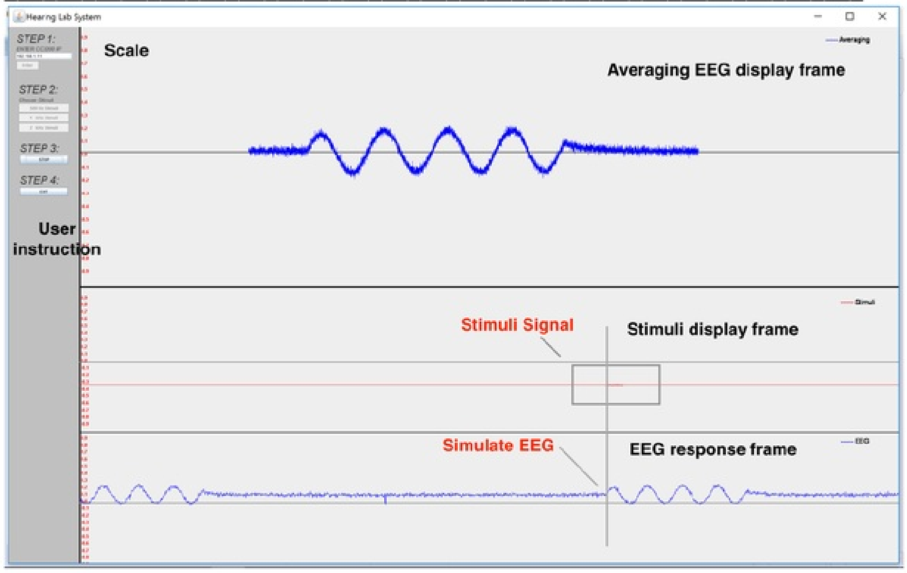


Figure 6.3: The result of simulation test

As a result, the real-time EEG signals display achieved the goal. The next step is to test the real person. However, the low pass filter has not been applied in the JAVA program due to the time restriction. Therefore, it was difficult to recognize the EEG responses on human's scalp. The reason is the human's brain responses locating in a very low range frequency, therefore, without the low pass filter, the HEARLab System cannot recognize the noise and the human's brain responses. On the other hand, the user blinks the eyes or move the jaw also influence the testing.

In the aspect of the interference, the CC3200 Audio BoosterPack contains multiple internal amplifiers. In the wireless HEARLab System prototype, the multiple amplifiers and the EPU are able to talk to each other, in this case, an artificial effect and hum causing some unexpected interference. Figure 6.4 is EEG response with the hum and artificial effect.

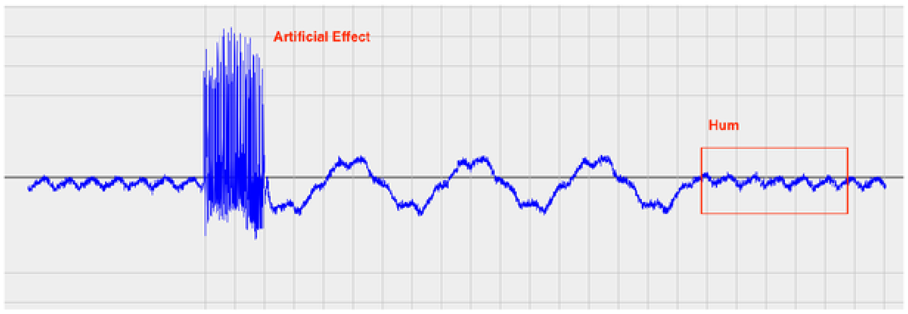


Figure 6.4 Artificial effect and hum in EEG response

To get rid of the unwanted effects, following Figure 6.2 to set up the experiment equipment, and applying an external amplifier instead of using the internal amplifier. Figure 6.3 shows the EEG response without the hum and artificial effect.

Chapter 7

Result

As a result, the wireless HEARLab System shows a positive result. The aim of this project is to test based on the IoT, in the Wi-Fi environment, the feasibility of the low frequency (5-10Hz) signals transmission. In addition, the goal is to reduce the wires in the original HEARLab System. Figure 7.1 gives the comparison of the two systems. The wireless HEARLab System prototype is able to place away from the computer, as long as within the Wi-Fi environment.

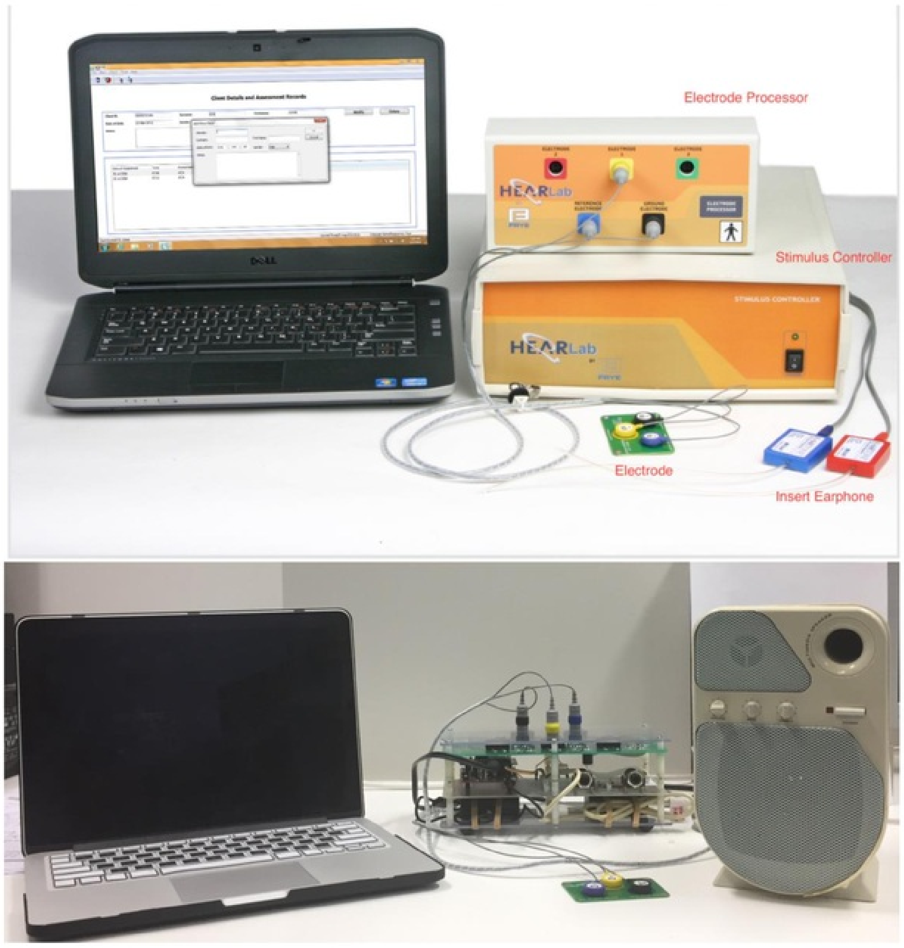


Figure 7.1: Comparison of the two systems

In Chapter 5 it proved that the wireless HEARLab System prototype is working properly, although the prototype has the weaker signals. Figure 7.2 shows the comparison between the original HEARLab System and the wireless HEARLab System prototype. It is evident that the sharp decrease is the brain cortex response while the subject heard the stimuli sound.

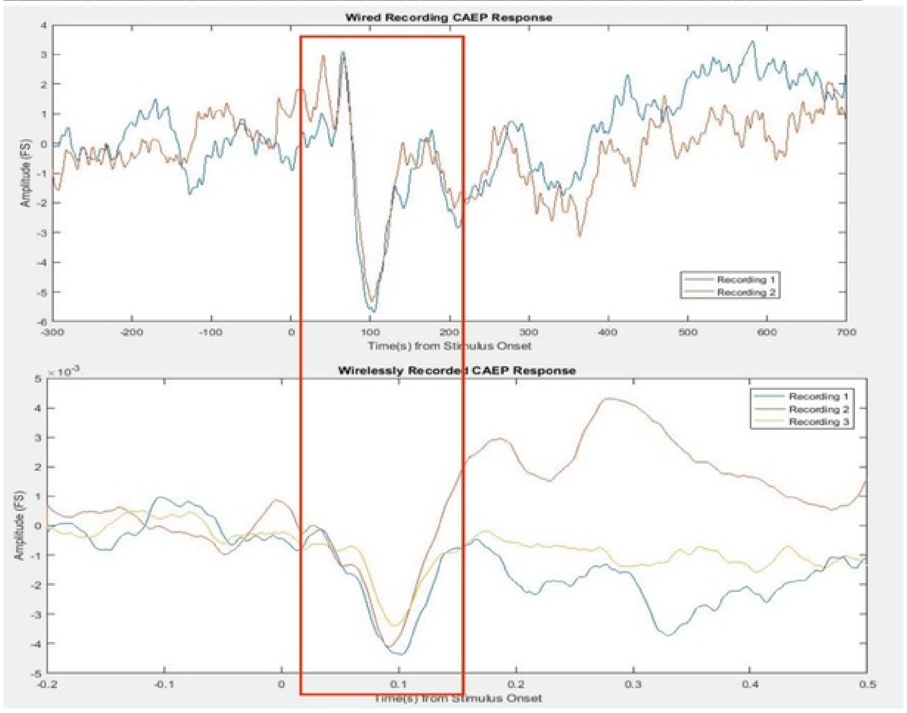


Figure 7.2: The comparison between the original HEARLab System and the wireless HEARLab System prototype

In the real-time testing, according to the Chapter 6, it shows that the program is able to detect the simulated EEG response. However, according to the NAL requirements, using the JAVA to develop a program is not able to finish by one person. In this case, there is no enough time to process everything in detail. Therefore, the low pass filter section in JAVA could not come true. In another aspect, the program met the rest of the requirements. As Figure 6.3 (Chapter 6) shows, the program is able to display the dynamic EEG response, cutting the epochs, averaging the EEG response, and also provide a GUI to help the user to operate the system.

Chapter 8 Conclusion

The goal of the project undertaken by both Thomas and myself was to prove the concept of a wireless EEG system capable of delivering accurate and informative results when used to conduct the Cortical Auditory Evoked Potentials (CAEP) assessment. The results gathered, and the methods shown by both students demonstrate this goal has been achieved to varying degrees of success as outlined in both thesis papers, more importantly; the theses show the concept is plausible if more resources such as time and money are available to develop further and refine the system.

The primary outcomes of the project found a wireless EEG system could be created to function without wires from the central unit at the patient end to the host PC at the audiologist end whilst delivering either the real time or offline recording analysis. Despite real time recording analysis, not function as desired the concept was proven. The project also found the Electromagnetic Interference generated from the wireless unit would have a negligible impact on the displayed response.

Secondary outcomes of the project stem from problems encountered throughout the development of the project. Firstly, working in a team environment on an industry project gave a great insight into the approaches used by professional Engineers to solve problems as a team; we were able to sit in on some engineering team meetings participate in product discussions. Secondly, with so much independence within the project we were forced to learn from our mistakes to ensure our project stayed on the correct path and our goals were met. Lastly, with a large period for the project, time management became an essential skill to ensure our time was used properly.

Chapter 9

Abbreviations

NAL	National Acoustic Laboratory
EEG	Electroencephalography
IoT	Internet of Things
ADC	Analog-to-Digital Converter
CAEPs	Cortical Auditory Evoked Potentials
ABR	Auditory Brainstem Response
TI	Texas Instruments
UDP	User Datagram Protocol
DSP	Digital Signal Processing
SNR	Signal to Noise ratio
MCU	Microcontroller
GUI	Graphical User Interface
IDE	Integrated Development Environment
SDK	Software Development Kit
CCS	Code Composer Studio
UART	Universal Asynchronous Receiver/Transmitter
I2S/PCM	Inter-IC Sound/Pulse-code modulation
SPI	Serial Peripheral Interface
LAN	Local Area Network
PCM	Pulse-Code Modulation
AP	Access Point
OSI	Open System Interconnection
EPU	Electrode Processor Unit

Bibliography

[1]"National Acoustic Laboratories", Nal.gov.au, 2016. [Online]. Available: https://www.nal.gov.au. [Accessed: 01- Nov- 2016].

[2] F. Bardy, B. Dun, H. Dillon, M. Seeto, H. Qin, T. Loi and R. Cowan, "The Cortical Automatic Threshold Estimation in Adults", The Hearing Journal, vol. 69, no. 6, p. 32, 2016.

[3]"The HEARLab® System–Cortical Auditory Evoked Potential Analyzer - Frye Electronics, Inc.", Frye Electronics, Inc., 2016. [Online]. Available: http://www.frye.com/wp/hearlab-system/. [Accessed: 31- Aug- 2016].

[4] T. Picton, Human Auditory Evoked Potentials. Plural Publishing, 2010, p. 31.

[5] Niedermeyer E.; da Silva F.L. (2004). Electroencephalography: Basic Principles, Clinical Applications, and Related Fields. Lippincot Williams & Wilkins.

[6]"National Acoustic Laboratories", Nal.gov.au, 2016. [Online]. Available: http://www.nal.gov.au/hearing-assessment_tab_objective-testing.shtml. [Accessed: 31- Aug- 2016].

[7]"Auditory Brainstem Response (ABR)", Asha.org, 2016. [Online]. Available: http://www.asha.org/public/hearing/Auditory-Brainstem-Response/. [Accessed: 06- Sep- 2016].

[8] Joint Committee On Infant Hearing. Year 2000 position statement. Audio Today 2000; 3-23

[9] W. Pearce and M. Golding, "The clinical use of CAEPs: a case study", National Acoustic Laboratories, Sydney, 2003.

[10] K. Agung, S. Purdy, C. McMahon, H. Dillon, R. Katsch and P. Newall, "Objective verification of speech perception using cortical auditory evoked potentials", National Acoustic Laboratory, Sydney, 2003.

[11] Cone-Wesson B, Wunderlich J. (2003). Auditory evoked potentials from the cortex: audiology applications. *Current Opinion in Otolaryngology and Head and Neck Surgery*, 11, 372-377.

[12] Gravel J S, Kurtsberg D, Stapells D R, Vaughan H G, Wallace I F. (1989). Case Studies. *Seminars in Hearing*, 10, 272-287.

[13] Rapin I, Granziani L J. (1967). Auditory-evoked responses in normal, brain-damaged, and deaf infants. *Neurology*, 17, 881-894

[14]"ARM Cortex-M4 32b MCU+FPU, 210DMIPS, up to 1MB Flash/192+4KB RAM, crypto, USB OTG HS/FS, Ethernet, 17 TIMs, 3 ADCs, 15 comm. interfaces & camera", 2016. [Online]. Available: http://www.st.com/content/ccc/resource/technical/document/datasheet/98/9f/89/73/01/b1/48/98/DM00035129.pdf/files/DM00035129.pdf/jcr:content/translations/en.DM00035129.pdf. [Accessed: 01- Nov- 2016].

[15]CC3200 SimpleLinkTM Wi-Fi® and IoT Solution with MCU LaunchPad Hardware, 2nd ed. Dallas Texas: Texas: Texas Instruments Incorporated, 2014, p. 4.

[16] "SimpleLink Wi-Fi CC3200 LaunchPad - CC3200-LAUNCHXL - TI Tool Folder", Ti.com, 2016. [Online]. Available: http://www.ti.com/tool/cc3200-launchxl. [Accessed: 31- Aug- 2016].

[17] "Code Composer Studio (CCS) Integrated Development Environment (IDE) - CCSTUDIO - TI Tool Folder", Ti.com, 2016. [Online]. Available: http://www.ti.com/tool/ccstudio. [Accessed: 01- Nov- 2016].

[18]"CCS UniFlash - CC3100/CC3200 Edition - Texas Instruments Wiki", Processors.wiki.ti.com, 2016. [Online]. Available: http://processors.wiki.ti.com/index.php/CCS_UniFlash_-_CC3100/CC3200_Edition. [Accessed: 01- Nov- 2016].

[19]"CC3000 Smart Config - Texas Instruments Wiki", Processors.wiki.ti.com, 2016. [Online]. Available: http://processors.wiki.ti.com/index.php/CC3000_Smart_Config. [Accessed: 01- Nov- 2016].

[20]"SimpleLink Wi-Fi CC3200 Audio BoosterPack - CC3200AUDBOOST - TI Tool Folder", Ti.com, 2016. [Online]. Available: http://www.ti.com/tool/cc3200audboost. [Accessed: 01- Nov- 2016].

[21] "What is User Datagram Protocol (UDP)? - Definition from Techopedia", Techopedia.com, 2016. [Online]. Available: https://www.techopedia.com/definition/13460/user-datagram-protocol-udp. [Accessed: 31- Aug- 2016].

[22]"What is UDP (User Datagram Protocol)? - Definition from WhatIs.com", SearchSOA, 2016. [Online]. Available: http://searchsoa.techtarget.com/definition/UDP. [Accessed: 01- Nov- 2016].

[23] P. f. Fan and G. z. Zhou, "Analysis of the business model innovation of the technology of internet of things in postal logistics," Industrial Engineering and Engineering Management (IE&EM), 2011 IEEE 18Th International Conference on, Changchun, 2011, pp. 532-536.

[24] A link to the Internet of Things, 1st ed. Dallas, Texas: Texas Instruments Incorporated, 2014, p. 2.

[25] Wireless connectivity for the Internet of Things, 1st ed. Dallas, Texas: Texas Instruments Incorporated, 2014, p. 4.

[26]"What is digital signal processing (DSP)? - Definition from WhatIs.com", WhatIs.com, 2016. [Online]. Available: http://whatis.techtarget.com/definition/digital-signal-processing-DSP. [Accessed: 03- Sep- 2016].

[27] S. Smith, The scientist and engineer's guide to digital signal processing. [San Diego, Calif.]: California Technical Pub., 2002, p.5.

[28] 2016. [Online]. Available: http://www.owlnet.rice.edu/~dodds/Files331/noise_notes.pdf. [Accessed: 05- Sep- 2016].

[29] CC3200 SimpleLink Wi-Fi and IoT Solution with MCU LaunchPad Getting Started Guide, 2nd ed. Dallas Texas: Texas: Texas Instruments Incorporated, 2016

[30] "Send audio over UDP", Stackoverflow.com, 2016. [Online]. Available: http://stackoverflow.com/questions/23847905/send-audio-over-udp. [Accessed: 02- Nov- 2016].

Appendix A

HEARLab System JAVA code

The HEARLab System JAVA code link shows as below.

https://www.dropbox.com/sh/fs3zc8j8ew5ye1r/AAAjMl2JGaWqB80vh6yE_Kwoa?dl=0.

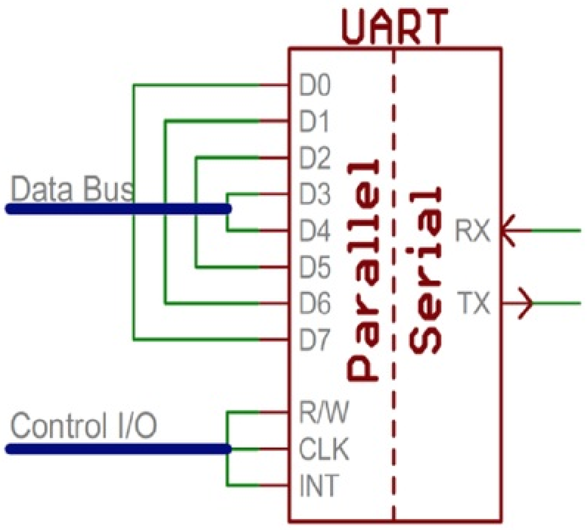
Appendix B

Universal Asynchronous Receiver/Transmitter

UARTs

The final piece to this serial puzzle is finding something to both create the serial packets and control those physical hardware lines. Enter the UART.

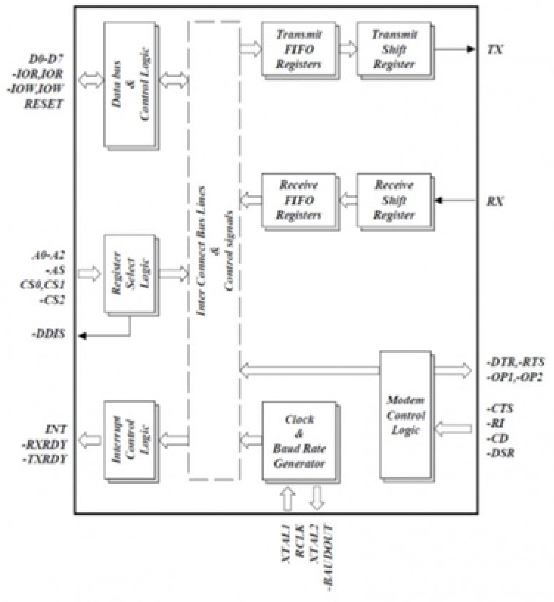
A universal asynchronous receiver/transmitter (UART) is a block of circuitry responsible for implementing serial communication. Essentially, the UART acts as an intermediary between parallel and serial interfaces. On one end of the UART is a bus of eight-or-so data lines (plus some control pins), on the other is the two serial wires - RX and TX.



Super-simplified UART interface. Parallel on one end, serial on the other.

UARTs do exist as stand-alone ICs, but they're more commonly found inside microcontrollers. You'll have to check your microcontroller's datasheet to see if it has any UARTs. Some have none, some have one, some have many. For example, the Arduino Uno - based on the "old faithful" ATmega328 - has just a single UART, while the Arduino Mega - built on an ATmega2560 - has a whopping four UARTs.

As the *R* and *T* in the acronym dictate, UARTs are responsible for both sending and receiving serial data. On the transmit side, a UART must create the data packet - appending sync and parity bits - and send that packet out the TX line with precise timing (according to the set baud rate). On the receive end, the UART has to sample the RX line at rates according to the expected baud rate, pick out the sync bits, and spit out the data.



Internal UART block diagram (courtesy of the Exar ST16C550 datasheet)

More advanced UARTs may throw their received data into a **buffer**, where it can stay until the microcontroller comes to get it. UARTs will usually release their buffered data on a first-in-first-out (FIFO) basis. Buffers can be as small as a few bits, or as large as thousands of bytes.

Software UARTs

If a microcontroller doesn't have a UART (or doesn't have enough), the serial interface can be **bit-banged** - directly controlled by the processor. This is the approach Arduino libraries like SoftwareSerial take. Bit-banging is processor-intensive, and not usually as precise as a UART, but it works in a pinch!

Appendix C

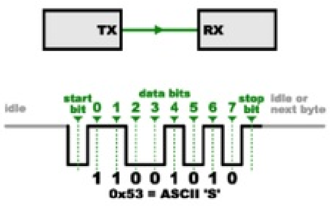
Serial Peripheral Interface

Serial Peripheral Interface (SPI) is an interface bus commonly used to send data between microcontrollers and small peripherals such as shift registers, sensors, and SD cards. It uses separate clock and data lines, along with a select line to choose the device you wish to talk to.

What's Wrong with Serial Ports?

A common serial port, the kind with TX and RX lines, is called "asynchronous" (not synchronous) because there is no control over when data is sent or any guarantee that both sides are running at precisely the same rate. Since computers normally rely on everything being synchronized to a single "clock" (the main crystal attached to a computer that drives everything), this can be a problem when two systems with slightly different clocks try to communicate with each other.

To work around this problem, asynchronous serial connections add extra start and stop bits to each byte help the receiver sync up to data as it arrives. Both sides must also agree on the transmission speed (such as 9600 bits per second) in advance. Slight differences in the transmission rate aren't a problem because the receiver re-syncs at the start of each byte.

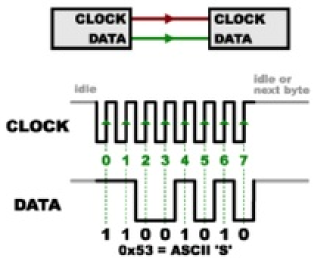


(By the way, if you noticed that "11001010" does not equal 0x53 in the above diagram, kudos to your attention to detail. Serial protocols will often send the least significant bits first, so the smallest bit is on the far left. The lower nybble is actually 0011 = 0x3, and the upper nybble is 0101 = 0x5.)

Asynchronous serial works just fine, but has a lot of overhead in both the extra start and stop bits sent with every byte, and the complex hardware required to send and receive data. And as you've probably noticed in your own projects, if both sides aren't set to the same speed, the received data will be garbage. This is because the receiver is sampling the bits at very specific times (the arrows in the above diagram). If the receiver is looking at the wrong times, it will see the wrong bits.

A Synchronous Solution

SPI works in a slightly different manner. It's a "synchronous" data bus, which means that it uses separate lines for data and a "clock" that keeps both sides in perfect sync. The clock is an oscillating signal that tells the receiver exactly when to sample the bits on the data line. This could be the rising (low to high) or falling (high to low) edge of the clock signal; the datasheet will specify which one to use. When the receiver detects that edge, it will immediately look at the data line to read the next bit (see the arrows in the below diagram). Because the clock is sent along with the data, specifying the speed isn't important, although devices will have a top speed at which they can operate (We'll discuss choosing the proper clock edge and speed in a bit).



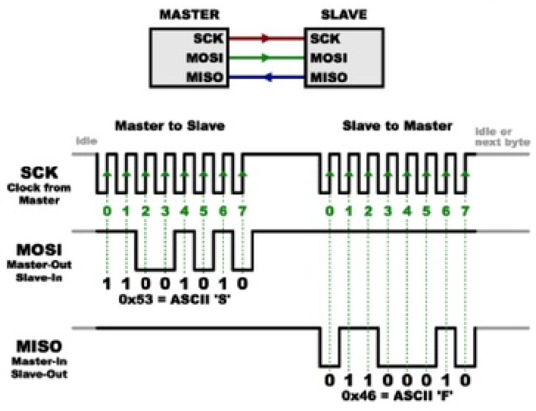
One reason that SPI is so popular is that the receiving hardware can be a simple shift register. This is a much simpler (and cheaper!) piece of hardware than the full-up UART (Universal Asynchronous Receiver / Transmitter) that asynchronous serial requires.

Receiving Data

You might be thinking to yourself, self, that sounds great for one-way communications, but how do you send data back in the opposite direction? Here's where things get slightly more complicated.

In SPI, only one side generates the clock signal (usually called CLK or SCK for Serial ClocK). The side that generates the clock is called the "master", and the other side is called the "slave". There is always only one master (which is almost always your microcontroller), but there can be multiple slaves (more on this in a bit).

When data is sent from the master to a slave, it's sent on a data line called MOSI, for "Master Out / Slave In". If the slave needs to send a response back to the master, the master will continue to generate a prearranged number of clock cycles, and the slave will put the data onto a third data line called MISO, for "Master In / Slave Out".

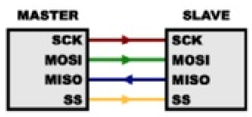


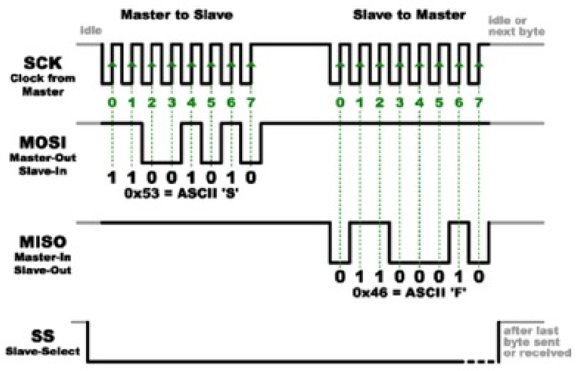
Notice we said “prearranged” in the above description. Because the master always generates the clock signal, it must know in advance when a slave needs to return data and how much data will be returned. This is very different than asynchronous serial, where random amounts of data can be sent in either direction at any time. In practice this isn't a problem, as SPI is generally used to talk to sensors that have a very specific command structure. For example, if you send the command for “read data” to a device, you know that the device will always send you, for example, two bytes in return. (In cases where you might want to return a variable amount of data, you could always return one or two bytes specifying the length of the data and then have the master retrieve the full amount.)

Note that SPI is “full duplex” (has separate send and receive lines), and, thus, in certain situations, you can transmit and receive data *at the same time* (for example, requesting a new sensor reading while retrieving the data from the previous one). Your device's datasheet will tell you if this is possible.

Slave Select (SS)

There's one last line you should be aware of, called SS for Slave Select. This tells the slave that it should wake up and receive / send data and is also used when multiple slaves are present to select the one you'd like to talk to.



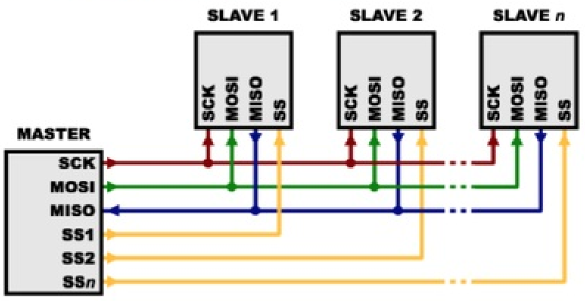


The SS line is normally held high, which disconnects the slave from the SPI bus. (This type of logic is known as “active low,” and you'll often see used it for enable and reset lines.) Just before data is sent to the slave, the line is brought low, which activates the slave. When you're done using the slave, the line is made high again. In a shift register, this corresponds to the “latch” input, which transfers the received data to the output lines.

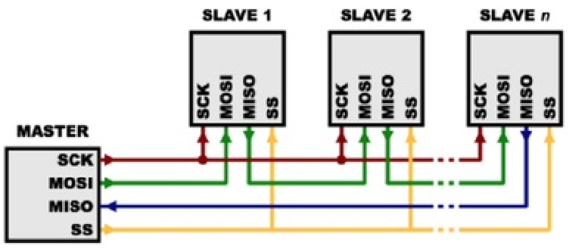
Multiple slaves

There are two ways of connecting multiple slaves to an SPI bus:

1. In general, each slave will need a separate SS line. To talk to a particular slave, you'll make that slave's SS line low and keep the rest of them high (you don't want two slaves activated at the same time, or they may both try to talk on the same MISO line resulting in garbled data). Lots of slaves will require lots of SS lines; if you're running low on outputs, there are binary decoder chips that can multiply your SS outputs.

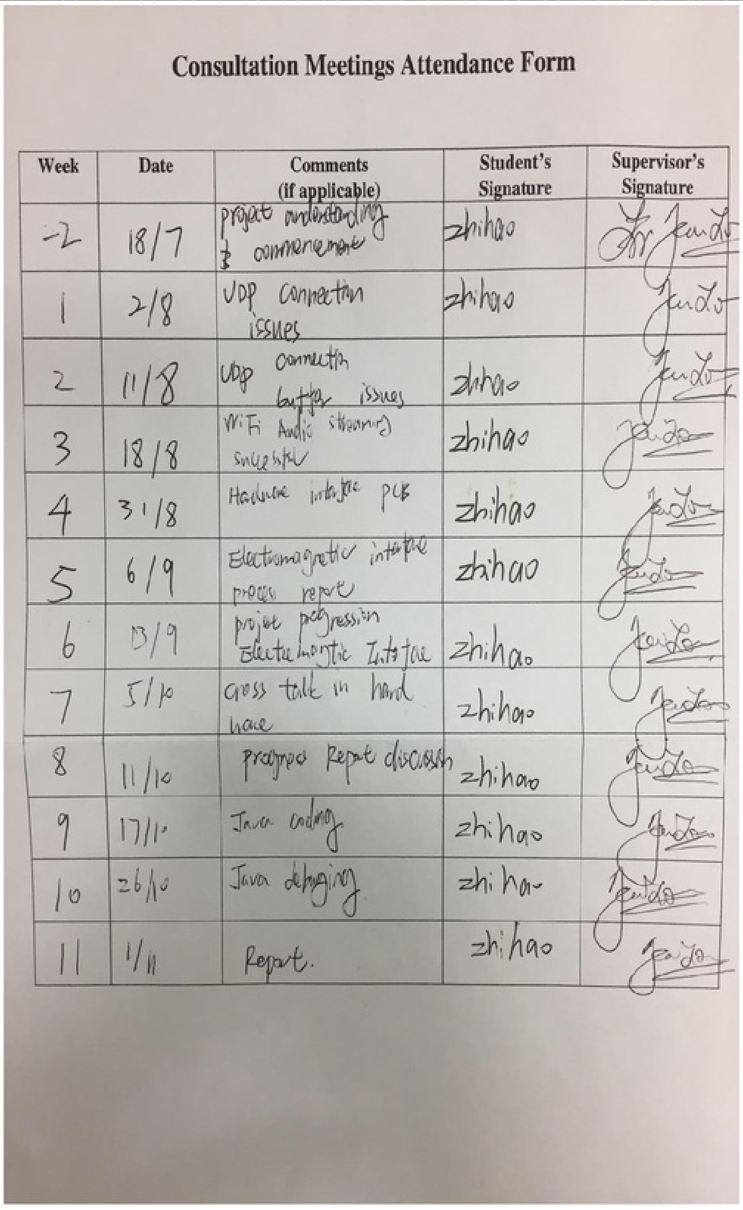


1. On the other hand, some parts prefer to be daisy-chained together, with the MISO (output) of one going to the MOSI (input) of the next. In this case, a single SS line goes to *all* the slaves. Once all the data is sent, the SS line is raised, which causes all the chips to be activated simultaneously. This is often used for daisy-chained shift registers and addressable LED drivers.



Note that, for this layout, data overflows from one slave to the next, so to send data to any *one* slave, you'll need to transmit enough data to reach *all* of them. Also, keep in mind that the *first* piece of data you transmit will end up in the *last* slave.

This type of layout is typically used in output-only situations, such as driving LEDs where you don't need to receive any data back. In these cases you can leave the master's MISO line disconnected. However, if data does need to be returned to the master, you can do this by closing the daisy-chain loop (blue wire in the above diagram). Note that if you do this, the return data from slave 1 will need to pass through *all* the slaves before getting back to the master, so be sure to send enough receive commands to get the data you need.



Consultation Meetings Attendance Form

Week	Date	Comments (if applicable)	Student's Signature	Supervisor's Signature
-2	18/7	project understanding & commencement	zhihao	[illegible]
1	2/8	UDP connection issues	zhihao	[illegible]
2	11/8	UDP connection buffer issues	zhihao	[illegible]
3	18/8	WiFi Audio streaming successful	zhihao	[illegible]
4	31/8	Hardware interface PCB	zhihao	[illegible]
5	6/9	Electromagnetic interface process report	zhihao	[illegible]
6	13/9	project progression Electromagnetic Interface	zhihao	[illegible]
7	5/10	cross talk in hard ware	zhihao	[illegible]
8	11/10	progress Report discussion	zhihao	[illegible]
9	17/10	Java coding	zhihao	[illegible]
10	26/10	Java debugging	zhi hao	[illegible]
11	1/11	Report.	zhihao	[illegible]