ISSN 2301 - 6590

# MINIMUM SYSTEM DESIGN OF ANDROID BASED PSTN PHONE

Deo Kiatama[#1], Fransiscus Ati Halim[*2], Arnold Aribowo[#3]
[1,3]Computer Systems Department, Faculty of Computer Science, Universitas Pelita Harapan
Tangerang, Indonesia
deokiatama@uph.edu,
arnold.aribowo@uph.edu
[2]Computer Engineering Department,ICT Faculty,Universitas Multimedia Nusantara
Gading Serpong, Tangerang Indonesia
Fransiscus.ati@lecturer.umn.ac.id

***Abstract*— PSTN phone is static. However, with the development of Android based device, wireless computer network and minimum system, it is possible to make a PSTN phone mobile. To add the mobility behavior in PSTN phone, a research to develop Android based PSTN phone is needed. Android based device will act as a remote control for the PSTN phone and can be used to make and accept a call from distance.**

**This paper elaborates the Android based PSTN phone design. To test the system performance, two parts of testing are performed. First is testing the Android based PSTN phone remote control system and second is testing the digital voice streaming system between Android device and minimum system using UDP protocol over wireless computer network. PSTN phone remote control system is tested for its functionality while digital voice streaming system is tested for its quality by giving a sinusoidal signal to the system and comparing the input signal with output signal using oscilloscope.**

**According to the test, Android based PSTN phone prototype is able to make a connection and disconnection of Android device (as client) with microcontroller (as server), make a connection with auto server discovery system, make a call, receive a call, and use redial and hold functions with 100% success rate in certain condition. However, it is shown that the quality of voice sent from Android phone to Arduino module is poor due to the limited size of buffer in Arduino, but on the other way around the quality of voice sent from Arduino to Android phone is good.**

***Keywords—Android based device, streaming, protocol, wireless computer network, remote control***

## INTRODUCTION

Public Service Telephone Network (PSTN) is a common telecommunication service. It is static and lack of mobility. On the other hand, mobile phone is wireless and has a higher degree of mobility. Basically mobile phone has only Short Message Service/SMS and telephone functions.

By 2013, with the development in computer technology, it is possible to put a small processor yet strong enough to do basic personal computer tasks in a mobile phone such as browsing the internet, checking emails, chatting, video call, playing or recording music and videos, making word documents, spreadsheets, or presentation slides, even playing 3D games.

With the development of smart phone, open source Operating System such as Android OS, wireless network that has been used widely, and Arduino minimum system module, it is possible to remotely control PSTN phone from a far. With the help from Arduino module that controlling PSTN phone, Android mobile phone can make a call via PSTN phone remotely. The voice data can be also transferred through wireless network between PSTN phone and Android phone. With combination of these technologies, it is possible to make PSTN phone mobile.

The main idea addressed in this research is to integrate and to combine the power of android-based smart phone and PSTN phone into one system, where the smart phone can be used as a handset and the modified PSTN phone can be functioned as a base. Integration of both technologies enhances the mobility of PSTN phone.

In this research, an Arduino UNO-Ethernet Shield is used to control a modified PSTN phone and to communicate with Android based mobile phone over wireless network using a unique protocol called PSTN Remote Simple Protocol (PRSP), which is made especially for this purpose. There is also another Arduino UNO board who work as a slave to generate audio signal. Android based mobile phone used in this research is using Android 4.0. A wireless router is also used to connect all devices together over computer network.

After several test taken, Android based PSTN phone prototype which is designed in this research is able to make a connection and disconnection of Android device (as client) with microcontroller (as server), make a connection with auto server discovery system, make a call, receive a call, and use redial and hold functions. However, it is shown that the quality of voice sent from Android phone to Arduino module is poor, but on the other way around the quality of voice sent from Arduino to Android phone is good.

## LITERATURE REVIEW

### *Minimum Computer System*

A computer is a complex system; contemporary computers contain millions of elementary electronic components [1], but

there are some minimum requirements for electronic component to be called computer. We can see it from its structure and function of its internal structure. From its function, there are four basic functions: data processing, data storage, data movement and control [1]. From its structure there are four main internal structure of the computer: Central processing unit (CPU); often simply referred to as processor, main memory, I/O which moves data between the computer and its external environment, and system interconnection [1].

*PSTN Phone*

Public Service Telephone Network describes the various equipment and interconnecting facilities that provide phone service to the public. It began in the United States in 1878, adopted and used all around the world.

This telephone network is connected with switches which switch connection between lines. User will dial phone number when making call. Those numbers will be coded in DTMF and transmitted to the network. Afterwards, switching and connection will be made.

The PSTN is well known for providing reliable communications to its subscribers. The phrase "five nines reliability," representing network availability of 99.999 percent for PSTN equipment, and has become ubiquitous within the telecommunications industry [2].

*Dual Tone Multi-Frequency (DTMF)*

*Dual Tone Multi-Frequency (DTMF)* is a modulation system which combines two frequencies. Those two frequencies symbolize a certain command code. This system is used in PSTN system to symbolize phone number which dialed by user. In a telephone device which has keypad matrix input for telephone number, every button is divided into columns and rows. Each row and column has its own frequency. There are 4 rows and 4 columns and therefore there are 16 buttons (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, *, #, A, B, C, and D). When the button is dialed, two frequency; one from its column, one from its row will be combined and produce a proper DTMF signal. Configuration of keypad matrix and its corresponding frequency for each row and column in a telephone which has keypad matrix input is depicted in the figure below:

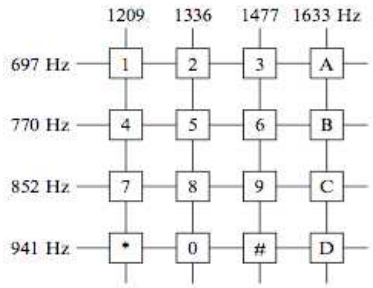


Figure 1. Keypad matrix on a telephone device [3]

For every push of a button, a DTMF signal will be generated and sent to the network. To identify which button is dialed, the signal received will be filtered and extracted, and then the button dialed will be identified by its two constituent frequencies.

*Android Operating System*

Android operating system (Android OS) is an open source operating system designed for mobile device such as mobile phone and tablet PC. Android OS is built on top of Linux. Linux is relatively easy to compile on various hardware architectures, and that means Android OS is also relatively easy to be implemented on various computer devices. Android OS is built in stacks of layer which are Linux kernel, libraries, application framework, and application layer [4].

*Analog to Digital Converter (ADC)*

Analog to Digital Converter converts an analog signal to a digital data. It converts the voltage or current value at its input into a binary word, which is the numerical representation of a quantized value closest to the input value. A quantizer defines a finite number of regions divided from a range of input value. Input signal will be sampled and then a quantizer will decide in which region it lies. The digital code of a quantized signal may be differ from one application to another, and has some varieties [5].

*Digital to Analog Converter (DAC)*

Digital to Analog Converter is the opposite of Analog to Digital Converter. It converts a digital data to an analog signal. DAC generates an analog signal based on digital data which represents a certain voltage or current value. DAC may also fill the signal values between the sampling times, which are called interpolation [5].

*Pulse Code Modulation (PCM)*

Pulse Code Modulation is a technique for the transmission of intelligence that virtually eliminates distance as a factor on quality of reception. It codes analog signal to digital data. That code symbolizes amplitude of the signal. There are two variables in PCM, which are sample rate & bit depth. Sample rate describes how many samples are taken in a second, usually measured in Hertz (Hz). Bit depth describes how many bit used for every sampling. A quantizer defines a finite number of regions divided from a range of input value. Input signal will be sampled and then quantizer will decide in which region it lies. N bit depth means a quantizer have $2^N$ regions divided from a range of input value. Higher bit depth sampling will produce more accurate representation of signal [6].

*Pulse Width Modulation (PWM) and Time Proportioning Control*

Pulse Width Modulation (PWM) is a modulation system that the output signal has a fixed potential difference but differ in pulse width. The pulse width will represent a certain value. Time proportioning control is a variation of the PWM signal that has a fixed duty cycle. Time proportioning control is widely used in control applications such as servo motor control, as well as in audio applications [7]. Both PWM signal and PWM signal with time proportioning control have a value based on time, but the time proportioning control system has a

fixed cycle time. Period of each cycle may vary, depending on the application.

## SYSTEM DESIGN

In general, this system is designed to control a PSTN phone. Initially, a user performs and receives a phone call through an android application. The call request is then transmitted in a wireless manner to an Arduino UNO-Ethernet Shield minimum system. Then the call request is forwarded to a PSTN phone. Overview of the system is shown below:

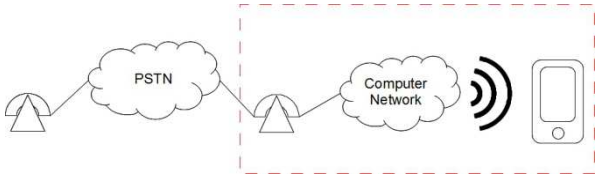


Figure 2. Overview of the system

Connectivity of the android device and the Arduino UNO-Ethernet Shield minimum computer system is implemented in both wired and wireless fashion. The android device is connected to a computer network through a wireless access point and the computer network is connected to the Arduino UNO-Ethernet Shield in a wired way.

Hardware of the system has three major components, namely: Converter module, Computer network and Android device. Converter module works as a bridge, connecting PSTN to Android device through computer network. It converts analog signal from PSTN to digital data and convert digital data from computer network back to analog signal. Converter module is built from three major components which are: A modified PSTN phone and Arduino Uno module which consist of master and slave module. Arduino Uno module is used to control PSTN phone module. It receives commands sent from Android device and executes the command received from Android device to PSTN module. The design of converter module is illustrated in the figure below:

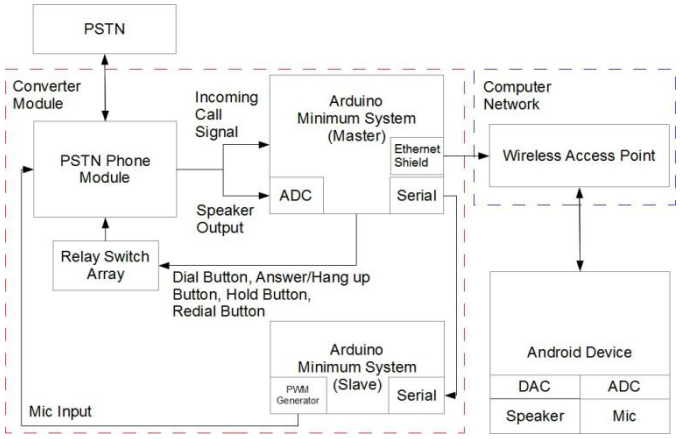


Figure 3. Hardware in detail

Phone module will be taken from the existing PSTN telephone and modified in order to communicate with the Arduino module. There are some things that will be controlled by the Arduino module, namely: Keypad matrix of 12 dial button key (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, *, and # button), hold button and hang up button.

Phone module will also provide signal notification of phone calls on the system. Incoming call notification signal will be sent to the microcontroller and will be forwarded to the Android device. To control buttons, relay switch circuit is used. Relay circuit switch is used to connect the rows and columns so that the two signals in the frequency column and row are connected and generate DTMF signals.

Minimum system module Arduino UNO-R3 used in this study has been equipped with Atmega 328 Microcontroller with 16MHz clock speed, six analog inputs with 10-bit depth, 14 pin I / O digital (six of which can be used to generate PWM signal, two pins external interrupts, two serial communication pin, four pin I2C communication), 32KB of flash memory, 2KB of SRAM, and EEPROM of 1KB. One Arduino microcontroller will act as the master and another module acts as a slave. This is because output pin that is used to generate the sound uses the same pin as I2C communication on microcontroller Atmega 328. Both microcontrollers will communicate through serial communication and communication is only done in one direction, i.e. from master to slave.

Master module's main role is to make a communication between the converter module and Android devices via a computer network. Master module has been equipped with additional Ethernet Shield devices so it can communicate through computer network. Master module also has a side job, which receives input signals for incoming calls, receives PSTN module output from the speakers, control telephone number buttons on the PSTN module, and forwards the PCM data sent by the Android device to the slave module.

Slave module produces sound with its PWM Generator. PWM signal is used to generate the sound signal. This is because the minimum system module Arduino UNO-R3 is not equipped with a DAC that is usually used to generate a signal from PCM data. The working relationship between the master module and slave modules and their tasks is shown below:

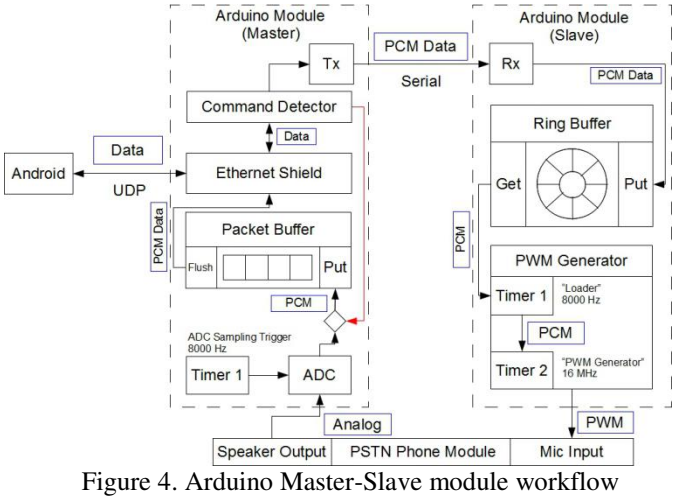


Figure 4. Arduino Master-Slave module workflow

Note:
Data = mixed data, PCM data and command

PCM data = several bytes of PCM data
PCM = one byte PCM data

Master module is responsible for checking the contents of UDP packets sent by Android device. Data may contain PCM data, as well as requests from the client. Master module is also assigned to respond to any request from the client. Data received by the master module will be forwarded to the slave module, inserted into the ring buffer and then based on this data, sound is generated by slave module. Master module is also assigned to sample sound data and send it to the Android device. Another task of the master module is to control PSTN phone module, controlling the number keys to enter the destination phone number, call hold, initiate calls and receive incoming signals.

As seen in the figure 4, the master and slave modules work with serial communications. PCM data sent from the master module will be put in a ring buffer before the data is then used to periodically generate a PWM signal. Ring buffer is a temporary data storage media based on circular array, an array in which the index system of the first array is located after the last array index.

Array system like this is very suitable for audio applications because of its very efficient buffer compared to the array of stack or queue system. If there is one data taken, removal of all existing data is not needed. It simply needs to move the start marker alone. When array is full and a new data coming, then the data which is the "oldest" will be replaced with the new data. The system is made so because in the terms of streaming audio, the new data worth more than the old data. To do that both markers (the start and the end marker) are shifted.

In the real system, data received from Android device is in 24 bytes chunks and forwarded by master module to the slave module, slave module then put them in the ring buffer. This is because 24 bytes data is the limit of data catched by Arduino's ethernet shield. Although ring buffer receives data in 24 byte chunks, data released from the ring buffer in the slave module is not in 24 bytes chunks. The data will be released from the buffer byte by byte on every 1/8000 sec triggered by timer 1 / "loader", and that data will be used by the timer 2 / "PWM Generator" to generate a PWM signal.

PSTN Remote Simple Protocol (PRSP) is a protocol which is specially designed for communication between Android device and converter module in this research. In converter module, master module is used to handle communication. Master module has command identifier function and command sender function. Command in this protocol uses the existing 24-byte, the same place with the PCM data. PRSP is designed to work in the same channel as data channel. It is designed to have a unique form of data that can be distinguished from the PCM data transmitted. This command then will be encapsulated in UDP packet and sent through computer network. There are 28 commands in this protocol with the command number 0-27 including ping, ACK, connection request and dial buttons.

The command list will be used to communicate between the server (in this case is the Arduino module master) and client (Android devices). Mostly they are used by the client as a request to the server. The order will be included in the UDP data packets and transmitted in 24 bytes data size. The command is modulated with a unique arrangement that can be distinguished with PCM data. Modulation of the command code is described as follows:

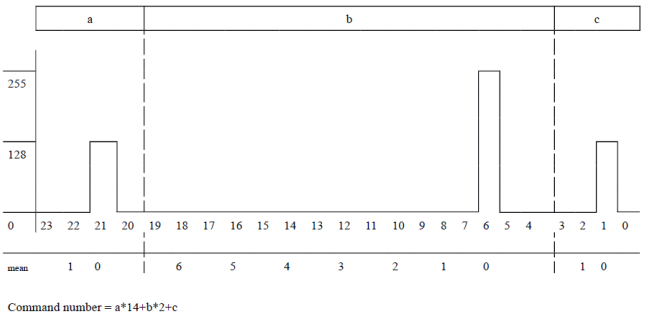


Figure 5. PSTN Remote Simple Protocol command modulation

24 bytes of data is divided into three sections, namely section a, b, and c. Section a and c are 4 bytes, while section b is of size 16 bytes. On section a and c, the only valid value is 0 and 128 while on section c, it is 0 and 255. Otherwise, it will not be detected as a command. In addition, only one non-zero on each section is allowed. More than that, it is not considered as a command. Specifically in section b data is valid if and only if the 255 value is on the even byte (6, 8, 10, 12, 14, 16, 18). If there is non-zero byte on the odd byte, then the data will not be considered as a command. The combination begins with 000 meaning a zero (0) to 161 which means that 1 * 14 + 6 * 2 + 1 = 27. Rightmost bits worth one, the bits that in the middle worth two and the leftmost bit is worth 14.

In this protocol, there are four phases (phase 0-3): Standby phase, calling phase, holding phase, incoming call phase. In addition, there are two connection conditions, which are connected or not connected and there are two conditions by the origin of orders, which are from clients who have been connected (connected to the server) or the other (not connected to the server). Phase status, connections and resources have the following relationship:

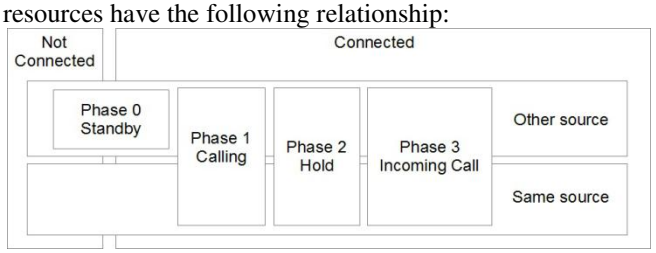


Figure 6. Relation between phase, connection, and source in PRSP

Server answers differently on each order according to the phase (at the time command arrive), connection status of the server, whether the server has been connected to a client or not, and source of the command, whether the command comes from the client that currently is connected to server or not.

Phase will change when there is a trigger such as a request from the server, or from the client. The changes into phase 0, phase 1, and phase 2 is the result of a request from the client, and the changes to the phase 3 is the result of the server request. Relationship between the client and server is one-to-one. The server is only connected to a single client. When the server has been connected with its client, the server records the address of the client and only serves requests from the paired client, as well as the client. There is one exception when server still answer to the broadcast ping from other clients. This is an auto discovery feature of the system.

The server will always respond with an answer to the client requests. The server's answers can be an ACK, which is a confirmation that the request has been received and has been processed, or NACK response which means the server can not serve the request, either because it has been connected with other client or client request can not be processed because it is in the wrong phase, and the answer to the availability of the server such as "connected" command and "not connected" command.

Software on Android devices called "PSTNremote", consists of the user interface, database, communication thread and audio thread. Interface is used as a tool for user to interact with the system. Users can input commands and receive information from the system. The communication thread connects the user interface to the converter module that is connected through a computer network. Communication thread will play a role in the exchange of audio data and commands through the PRSP. Audio thread is responsible for the audio playback, audio data capturing, and audio data buffering. The database is used to exchange data between the interface and save the state of the system.

The main menu interface is a display menu that users see when they first open the application "PSTNremote". This display menu shows the status of the connection and has two buttons, one is used to access the destination number input interface and another is used to access connection settings interface. If there is an incoming call, notification will be provided with a popup incoming call interface that will appear. User then has the option to accept or reject the call.

The connection settings interface has a server address input column and three buttons: the "disconnect" button, "search available servers" button and "connect" button. Search button will open the "search for available servers" interface. "Search for available servers" interface shows a list of existing servers, which has not been connected and which already connected. It also displays three buttons: refresh, cancel and OK. Refresh button is used to update the server list.

"Make calls" interface is used to make phone calls. In this interface, user can enter the destination number. On the top of the interface there is a column that shows the destination number (that user dialed), number buttons, *, #, redial, hold, and end button. Finally, the "incoming call" interface is the interface that is appeared when a call is coming. This interface displays two buttons: the answer and ignore buttons.

## RESULT ANALYSIS

To ensure whether the functionality of the system works, functional testings are performed by testing the disconnection and connection with the server, searching server with auto discovery, make a call by entering the destination telephone number, calling by pressing the redial button, make a call and pressing the hold button in the middle of a call, and receive incoming calls.

Connection and disconnection test is done by connecting the Android to Arduino master module through a computer network, and then a user performs a connection to server. Initially, an Android device is not connected to any server and shows "not connected" status on the main interface, and then the "connection settings" is pressed to show the connection settings interface. Server address which was initially empty is then filled with the IP address of the target server and then the "connect" button is pressed. After the Android device receives an ACK command from the server, the connection with the server is successful and the display will automatically return to the main interface. In the main menu interface's connection status, the status has been changed to "connected", that indicates the connection has been successfully done.

To test the server-client disconnection, in the "connected" state to the server, the "connection settings" button is pressed so that the display move on to the connection settings interfaces. In the connection settings interface the "disconnect" button is pressed. After receiving ACK command as an answer from server, the connection is lost and automatically returns to the main menu interface. In the main interface's connection status column, the status has been changed to "not connected" which indicates the connection has been lost.

Auto server discovery feature functional test is conducted similar to the connection test but the server address is not entered by typing the IP address of the server. The server address will be acquired from the result of auto discovery feature. The test is started from "not connected" conditions. In the connection settings interface, the "search available server" button is pressed so that the search for available servers interface appears. At the search for available servers interface, press the "refresh" button to start searching for server. As soon as the reply from server comes, the server address and its connection status are obtained. Because the server is not connected to any client, the server can be selected. After selecting the available server, press the "OK" button. The address of the selected server is appeared on the server address field. After pressing the "connect" button, the display automatically returns to the main display and the status is changed to "connected" which indicates the connection has been successfully created.

Calling by dialing destination test is done after the connection to the server has been successfully created. A call is initiated by pressing the "dial" button on the main interface so that the make calls interface shows up. On make calls interface, the destination phone number is typed. A few moments after successfully calling the destination, then the

call is terminated by pressing the "end" button. It returns to the main menu.

Calling with use of redial function test is conducted after calling by dialing destination test so the dialed number can be stored and can be used in the redial function. In this test, a call is initiated by pressing the "dial" on the main display so that the make calls interface appears. At the interface, the "redial" button is pressed and the number appears on the display before the call is started. After a few moments, the call is successfully established, then the call is terminated by pressing "end" button and the display returns to the main menu interface.

Calling with use of hold function test is similar with a test call by dialing numbers, but at this time in the middle of a conversation, the hold button is pressed to hold the call. A call is initiated by pressing the "dial" on the main display so that the make calls interface appears on screen. On the make calls interface, the destination phone number is dialed and the call begins. After a few moments later, the "hold" button is pressed, and then the "hold" button is pressed again after few seconds to continue the conversation. The call is terminated by pressing "end" button and finally it is back to the main menu.

Accepting incoming call test is done by calling to the address number using another PSTN phone and then accepting the call from Android device. When the call comes, an incoming calls interface appears on the Android device's screen and the device is ringing. By pressing the "accept" button, the call is successfully accepted. The call is stopped by pressing the "end" button and the display return to the main menu interface.

Testing the quality of sound produced by the Android device is made by giving a voice signal input of 1000 Hz on the Arduino, taking the output signal from the Android device and analyzing it with an oscilloscope. Sinusoidal signal is generated by a computer device using Audacity software. The signal is given to the Arduino device. Clamper circuit is needed to change the signal with a negative voltage on the valley sinusoidal signal into a sinusoidal signal of 0 V in the valley. This is because the DAC on the Arduino Uno minimum system can only read a positive voltage. Sinusoidal signals received is forwarded to Android devices through a computer network and produced by Android devices. The output signal from the Android is observed with the help of an oscilloscope. According the experiment, sinusoidal signals received and transmitted from the Arduino can be reproduced accurately. However, often there is a vacuum that causes the output signal to stop. This is caused by interference at the computer network so that the data is not delivered periodically. It may also be due to the limited length of data packets that can be sent by the Arduino device for one-time delivery. It is also caused by limited processing speed on the Arduino module. The small size of data that can be sent in one shipment delivery leads to more frequent transmission. It requires a longer time to transmit all the data.

Signal quality test generated by Arduino is done by providing PCM data in the form of a 1000 Hz sinusoidal signal from Android devices and then sent to Arduino minimum system through computer network. Arduino device then generates PWM signal to reproduce the sound. Output signal from the Arduino is observed using an oscilloscope.

The output signal from the Arduino device do not provide significant patterns and relatively difficult to observe. It is probably due to limited buffer size of Arduino which is only 360 bytes. With its small size, it leads the running out of data on the Arduino. When the two-way human conversational test is done, a slow pronunciation is required to enable the clear message is delivered.

## CONCLUSION

From the test and analysis done in this research, there are several points of conclusion that summarize the result of this research: Firstly, the system has been able to make a connection and disconnection with the server properly. Secondly, the system has been able to search server with “auto discovery” feature well. Thirdly, the system has been able to make a call by entering the destination number. Fourthly, the system has been able to make calls by pressing the redial button well. Fifthly, the system has been successfully able to press the hold button in the middle of a call. Sixthly, the system has been able to receive incoming calls. Finally, the system has been able to produce a good sound quality but often run out of data due to buffer limitations, the limitations of the data packet size on Arduino modules and interference with computer network.

## REFERENCES

[1] Stallings, William, “Computer Organization and Architecture Designing for Performance Eighth Edition”, Prentice Hall, 2010
[2] Hewett, Jeff and Dryburgh, Lee, “Signalling System No.7 (SS7/C7) Protocol, Architecture, and Services, Cisco Press, 2004
[3] Kuo, Sen M. and Lee, Bob H., “Real Time Digital Processing”, Wiley, 2001
[4] Gargenta, Marko, “Learning Android”, O’Reilly Media, 2011
[5] Manolakis, Dimitris G. and Ingle, Vinay K., “Applied Digital Signal Processing”, Cambridge University Press, 2011
[6] P.R.Hariharan, “Study of Pulse Modulation”, Kansas State University, 1963
[7] Montgomery, Ross and McDowall, Robert, “Fundamentals of HVAC Control Systems”, Elsevier, 2008.