PORTABLE OSCILLOSCOPE WITH WIRELESS CONNECTIVITY

BY

KHEW FAN SIN

A REPORT

SUBMITTED TO

Universiti Tunku Abdul Rahman

in partial fulfilment of the requirements

for the degree of

BACHELOR OF COMPUTER ENGINEERING (HONS)

Faculty of Information and Communication Technology

(Perak Campus)

Jan 2016

DECLARATION OF ORIGINALITY

I declare that this report entitled "**PORTABLE OSCILLOSCOPE WITH WIRELESS CONNECTIVITY**" is my own work except as cited in the references. The report has not been accepted for any degree and is not being submitted concurrently in candidature for any degree or other award.

Signature : _____

Name : _____

Date : _____

ACKNOWLEDGEMENTS

I would like to express my gratitude and appreciation to my supervisor, Mr. Lee Wai Kong who have offer me an interesting and challenging project. An opportunity is given to me for learning how to manage and complete the project. Besides, I wish to thanks again to my supervisor for his supports either in technical skills or knowledge.

Thanks to my parents and family for their love, supports, patience and encouragement when I am in preparation of final year project. Lastly, I wish to appreciate patience of my friends when I grumble to them.

ABSTRACT

This project is designed portable oscilloscope prototype running in Embedded Linux by implementing designed application which consists of oscilloscope features and wireless connectivity. Portable oscilloscope is a useful measuring instrument that capture the electronic signal and plot out the signal. Nowadays, real time data logging and remotely control device is the features required to make the portable oscilloscope become more convenient for user but the more cost is required to implementing these features. This project present how to design and implement the embedded oscilloscope application with Wi-Fi features in hardware device to become prototype and PC-based software application to receive and plot out data.

The strength of this portable oscilloscope is having the features that commonly found from oscilloscope such as measurements, trigger function, persistence mode, and storage data. Besides, user can monitor and remotely control the portable oscilloscope wirelessly with other devices. The performance of data transmission is fast and reliable. This product have cheap cost but have same features with others.

The most challenging part is that cycle time of captured signal is different with theoretical result and software trigger function. Due to limitation of hardware, algorithm is applied to tackle the encountered problem in programming way.

TABLE OF CONTENTS

TITLE **i**

DECLARATION OF ORIGINALITY ii

ACKNOWLEDGEMENTS **iii**

ABSTRACT **iv**

TABLE OF CONTENTS **v**

LIST OF FIGURES **x**

LIST OF TABLES **xi**

LIST OF ABBREVIATIONS **xii**

CHAPTER 1 INTRODUCTION **1**

1-1 Background & Problem Statement 1

1-2 Motivation 2

1-4 Proposed Study 3

1-4-1 Object-Oriented Programming 3

1-4-2 Internet Protocol Suite 3

1-5 Achievement 5

1-6 Report Organization 7

CHAPTER 2 LITERATURE REVIEW **8**

2-1 Data Acquisition through Cable 8

2-2 Data Transferring through Bluetooth Connection 8

2-3 Data Transferring over Wired LAN 10

CHAPTER 3 PORTABLE OSCILLOCOPE SPECIFICATION **12**

3-1 Hardware Specification 12

3-2 Software Specification 12

CHAPTER 4 EMBEDDED APPLICATION HARDWARE DESIGN **14**

4-1 Block Diagram 14

4-2 FriendlyARM Tiny4412 15

4-3 Wi-Fi Modules 16

CHAPTER 5 EMBEDDED APPLICATION SOFTWARE DESIGN 17

5-1 Software Design 17

5-2 Qt Creator & QCustomplot 17

5-3 Tera Term 18

5-4 Flow Chart of Oscilloscope System 19

5-5 User Interface 20

5-6 Socket 23

5-6-1 Flow Chart of socket programming 23

5-6-2 Remote Control features 24

5-6-3 Implementation 25

5-7 Time base 27

5-8 Voltage Range 28

5-9 Read ADC 28

5-9-1 Modified ADC Kernel 28

5-9-2 Access ADC Driver 29

5-9-3 Start Button 29

5-9-4 Stop Button 30

5-9-5 ADC channels checkbox 30

5-9-6 Set ADC channel 31

5-9-7 Sample ADC data 33

5-9-8 Conversion of ADC data to voltage level 33

5-9-9 Plot ADC data in graph 34

5-9-20 Flow Chart 36

5-10 Processing ADC Data 37

5-10-1 Cycle Time 37

5-10-2 Frequency 39

5-10-3 Duty Cycle 39

5-10-4 Maximum & Minimum Voltage 39

5-10-5 Peak to Peak Voltage 40

5-10-6 Root Mean Square Voltage 41

5-10-7 Average Voltage 42

5-10-8 Rise Time & Fall Time 42
5-10-9 Display measurement results 44
5-11 Trigger Function 44
5-12 Auto Scales 47
5-13 Persistence mode 47
5-14 Storage 48
5-15 Screenshot function 48
5-16 Display current Date & Time 49
5-17 Data Logging 50
5-18 Save & Load Graph 51
5-19 Vision 51
5-20 Backlight 53
5-21 Save & Load Settings 53
CHAPTER 6 PC-BASED OSCILLOSCOPE APPLICATION 54
6-1 PC-Based Application 54
6-2 Microsoft Visual Studio & C# 54
6-3 Specification 55
6-4 Software Design 55
6-4-1 Flow Chart 56
6-4-2 Graphical User Interface (GUI) 57
6-4-3 Socket 57
6-4-4 Read the incoming ADC data 59
6-4-5 Display Graph 61
6-4-6 Remote Control 61
CHAPTER 7 TESTING & RESULTS 63
7-1 Testing & Results 63
7-2 Tools 63
7-2-1 Function Generator 63
7-2-2 PicoScope 2104 64
7-3 Results 64
7-3-1 Sine wave 65

7-3-2 Square Wave 65
7-3-3 Triangle Wave 66
7-3-4 Ramp Up Wave 66
CHAPTER 8 CONCLUSION **67**
8-1 Summary 67
8-2 Encountered Problems 67
8-3 Contributions 68
8-4 Improvements 68
REFERENCES **69**
APPENDIX A **A-1**
A-1 Sine Wave A-1
A-2 Square Wave A-2
A-3 Triangle Wave A-3
A-4 Ramp Up Wave A-4

List of Figures

Figure Number	Title	Page
Figure 1-4-2-F1	TCP/IP protocol suites (ePipe, n.d.)	3
Figure 1-4-2-F2	TCP/IP model in form of Client-Server architecture	5
Figure 1-5-F1	Screenshot of designed embedded oscilloscope application	6
Figure 1-5-F2	Screenshot of PC-based application that received signal data.	6
Figure 2-2-F1	Main interface of portable oscilloscope Android application	9
Figure 2-2-F2	Settings interface of portable oscilloscope Android application	10
Figure 4-1-F1	Hardware block diagram of portable oscilloscope	14
Figure 4-2-F1	Tiny4412 board	15
Figure 4-3-F1	Wi-Fi USB Adapter GWF-3A33	16
Figure 5-2-F1	Screenshot of UI form of and objects in Qt Designer	17
Figure 5-2-F2	Screenshot of implementation source codes of QCustomPlot into project	18
Figure 5-3-F1	Screenshot of Tera Term connected with Tiny4412 through serial terminal	18
Figure 5-4-F1	Flow chart of oscilloscope system	19
Figure 5-5-F1	Screenshot of designed oscilloscope GUI	20
Figure 5-5-F2	Page 1 of designed tab widget	21
Figure 5-5-F3	Page 2 of designed tab widget.	22
Figure 5-5-F4	Settings page of designed tab widget	22
Figure 5-5-F5	Coding to move sampling ADC action and socket action to other created threads	23
Figure 5-6-1-F1	Interaction between client and server	24
Figure 5-6-3-F1	Flow chart of TCP data server	25
Figure 5-6-3-F2	Flow chart of TCP control client	26

Figure Number	Title	Page
Figure 5-7-F1	Screenshot coding of manually set the time base values and labels	28
Figure 5-9-1-F1	Coding of modified ADC kernel in sampling function	29
Figure 5-9-2-F1	Coding of open ADC device file	29
Figure 5-9-3-F1	Coding of event of started timer by clicked start button	30
Figure 5-9-4-F1	Coding of stop sampling ADC data by clicked stop button	30
Figure 5-9-5-F1	Coding of ADC channel checkboxes	31
Figure 5-9-5-F2	Coding sampling ADC data based on selected ADC channels	31
Figure 5-9-6-F1	The defined function in ADC kernel.	32
Figure 5-9-6-F2	Coding of set ADC channel before sampling data	32
Figure 5-9-7-F1	Coding of read ADC device file	33
Figure 5-9-9-F1	Example of expansion of one data to 5 points	34
Figure 5-9-9-F2	Example of shrink data in 5ms of time bases	35
Figure 5-9-9-F3	Example of shrink data that each 5 points to one data by skipped 4 data	35
Figure 5-9-9-F4	Coding of set sampled ADC into graph.	35
Figure 5-9-20-F1	Flow chart of the readADC()	36
Figure 5-10-1-F1	Stages diagram of calculate total data in one cycle	38
Figure 5-8-1-F2	Coding of the algorithm to get one cycle	38
Figure 5-10-4-F1	Coding of finding maximum and minimum voltage	40
Figure 5-10-5-F1	Explanation of maximum, minimum and peak to peak voltage in diagram	40
Figure 5-10-6-F1	Explanation of Vrms in diagram	41
Figure 5-10-6-F2	Algorithm of Vrms calculation	41
Figure 5-10-8-F1	Explanation of rise time and fall time with ratio between 10% and 90%	43
Figure 5-10-8-F2	The flow chart of calculation of rise and fall time	44
Figure 5-11-F1	Algorithm of trigger function in flow chart	46

Figure Number	Title	Page
Figure 5-13-F1	Algorithm of implementing persistence mode	48
Figure 5-15-F1	Code of screenshot function	49
Figure 5-16-F1	Code to display current date and time	50
Figure 5-17-F1	Code of data logging function that write ADC data into file respectively	50
Figure 5-18-F1	Screenshot of GUI after clicked load graph button	51
Figure 5-19-F1	Screenshot of GUI in day vision	52
Figure 5-19-F2	Screenshot of GUI in night vision.	52
Figure 6-2-F1	Screenshot of Microsoft Visual Studio	55
Figure 6-4-1-F1	Flow chart of PC-based oscilloscope application	56
Figure 6-4-2-F1	Screenshot of application GUI	57
Figure 6-4-3-F1	Code of create the socket and bind socket to defined port number in C#	58
Figure 6-4-3-F2	Coding of disconnect socket connection event	58
Figure 6-4-4-F1	The flow chart diagram of receiving incoming data	60
Figure 6-4-5-F1	Coding screenshot of bind array data to graph	61
Figure 6-4-6-F1	Flow chart of sending command when event occurred	62
Figure 6-4-6-F2	Flow chart of receiving command from embedded oscilloscope	62
Figure 7-3-1-F1	Agilent U2761A USB Modular Function Generator	63
Figure 7-3-2-F1	PicoScope2104	64
Figure 7-3-F1	Screenshot of PicoScope 6 oscilloscope software	65

List of Tables

Table Number	Title	Page
Table 1-4-2-T1	Description 4 layered architectures of TCP/IP model	4
Table 5-7-T1	Table of time bases and its total data respectively	27
Table 5-12-T1	Appropriate time base according to frequency	47
Table A-1-T1	Results of sine wave by PicoScope 2104	A-1
Table A-1-T2	Results of sine wave by designed oscilloscope application	A-1
Table A-2-T1	Results of square wave by PicoScope 2104	A-2
Table A-2-T2	Results of square wave by designed oscilloscope application	A-2
Table A-3-T1	Results of triangle wave by PicoScope 2104	A-3
Table A-3-T2	Results of triangle wave by designed oscilloscope application	A-3
Table A-4-T1	Results of ramp up wave by PicoScope 2104	A-4
Table A-4-T2	Results of ramp up wave by designed oscilloscope application	A-4

LIST OF ABBREVIATIONS

GUI	Graphical User Interface
OOP	Object-Oriented Programming
ADC	Analog Digital Converter
A/D	Analog to Digital
USB	Universal Serial Bus
ms	Millisecond
JPEG	Joint Photographic Expert Group
PNG	Portable Network Graphic
Hz	Hertz
kHz	Kilo Hertz

CHAPTER 1 INTRODUCTION

1-1 Background & Problem Statement

Portable oscilloscope is a type of electronic test instrument that used to display and analyse the waveform of electronic signal over the time. Most of the hardware developer and industries use the portable oscilloscope to troubleshoot electric equipment. Nowadays it is necessary to acquire data from the portable oscilloscope in order to record or process the data in processor computer (PC). Using a cable to build a connection between oscilloscopes with PC is the most common ways for data transferring. Although this method achieve the goal, but there are still many constraints with this method.

Sometimes, oscilloscope may be spoiled due to overcurrent or overvoltage when troubleshooting electric equipment. PC connected with oscilloscope through a simple cable may be affected and damaged if the oscilloscope get the problem of overcurrent or overvoltage. Industries and developers require solutions to this problem in order to reduce the expense of PC replacement and cost development.

Troubleshooting electric equipment can be a short time process or long time process. The process can be completed with few hours or a day if it is a short time process. However, some of the troubleshooting process can be done with few days or few weeks. There is no ways to let the developers or testers stay with the PC to monitor the data without leaving. If developers have to work in outstation, they will be worry about their equipment's condition. These worries affect their mind when they are working. By implementing wireless connection with portable oscilloscope, the users can monitor the data of oscilloscope by internet in anywhere including other countries. In addition, users can remote control the oscilloscope in order to use function of oscilloscope. This solution is convenient to the user who have distance problem with oscilloscope.

The oscilloscope may be sold to some of the countries such as Iraq, Saudi Arabia, Russia and Greenland. The average temperature of Iraq and Saudi Arabia can reach 50 degree Celsius or above when summer. In winter season, average temperature of Russia and Greenland can fall minus 9 degree Celsius. Therefore, it is necessary to do temperature test to oscilloscope. For example, oscilloscope is put into oven for temperature test. It is not convenient that put the oscilloscope into oven with cable to acquire data. The wireless

connection is able to solve this problem. Besides, this solution is convenient for other test such as humidity test and others.

Some of the users monitor their electric equipment by oscilloscope all the times to ensure the electric equipment is in operating mode. They desire data acquired from oscilloscope are quasi simultaneous with real time data of oscilloscope. Therefore, a high data transferring speed is required to make data acquired from oscilloscope acting as quasi simultaneous.

1-2 Motivation

The motivation of this project is to build a portable oscilloscope which is low cost, low power consumption and transmit oscilloscope data wirelessly. The designed product can tackle and solve the problem statement mentioned above such as limitation of cable connection and slow data acquisition rate when transmitting data wirelessly. Another application is need to be developed to connect and receive the sent data from the oscilloscope system.

1-3 Objective

The main objective of this project is to build a prototype which consists of the common oscilloscope functionality and remote access features via wireless connection. The main objective is divided into three sub-objectives.

The first of sub-objectives of this project is develop an embedded application that consists of oscilloscope features and run the application in an embedded board device.

The next sub-objective is to implement the features of remote access between embedded devices and PC via Wi-Fi connection. The embedded application is developed to be able to send ADC and status data from embedded device to PC through wireless connection.

Last of the sub-objective is to develop a PC-based application capable to receive the data sent by the embedded device and display the oscilloscope information on PC. This application have the similar interface and display configuration of the embedded oscilloscope application. The user can also control the oscilloscope form PC wirelessly.

1-4 Proposed Study

1-4-1 Object-Oriented Programming

Object-oriented programming (OOP) skill is necessary to build a Graphical User Interface (GUI) application. OOP is a programming language model which organized around objects rather than "actions" and data rather than logic. The relationship between one object and another can be created. Example of the relationship is that object can inherit characteristics from other objects.

1-4-2 Internet Protocol Suite

In order to communicate between two processes or applications over a network, internet protocol suite is needed to be implemented. TCP/IP protocol suite is widely used protocol suite. TCP/IP networking model is built by 4 layered architectures. Each layer perform some functionality with each protocol but each layer consists of more than one protocol options. Figure 1-5-F1 shows the protocol suites of TCP/IP model and Table 1-5-T1 describes the 4 layered architectures of TCP/IP model.

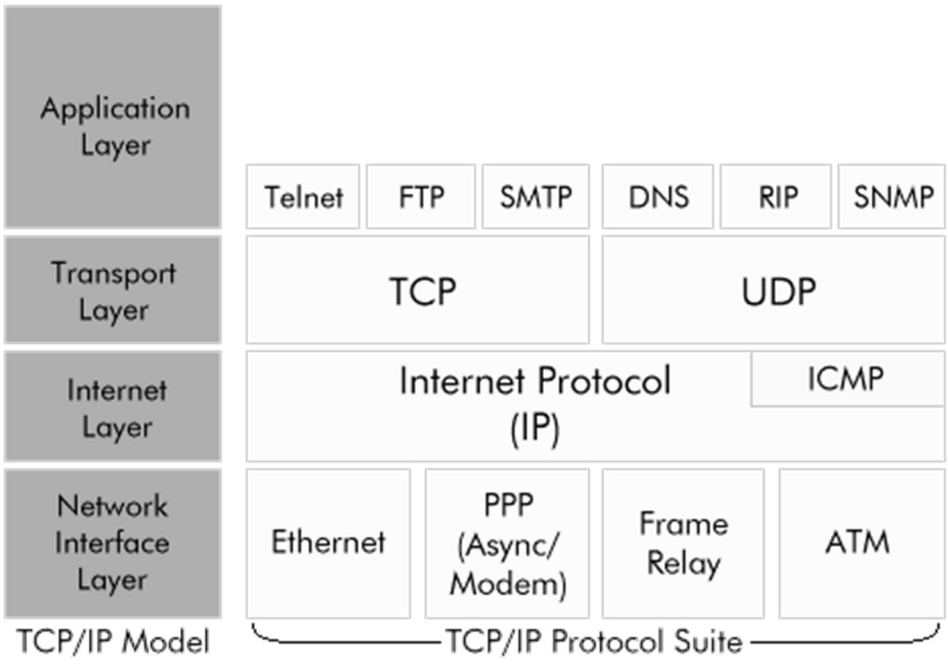

Figure 1-4-2-F1: TCP/IP protocol suites (ePipe, n.d.)

Layer	Description	Protocols
Network Interface	This layer specifies how data is physically transmitted across the network and includes specifications for the hardware that connects the device to the cable, the cable media itself and all the connecting hardware.	Ethernet, Serial, Frame Relay, ATM, etc.
Internet Layer	Sometimes called the network layer as the protocols in this layer are responsible for the sending and receiving the data on the network and for deciding how packets of data get from one network to another (called routing).	IP, ICMP, ARP
Transport Layer	This layer is responsible for ensuring reliable delivery of data from end to end and for other connection management activities.	TCP, UDP
Application Layer	This layer defines how TCP/IP applications talk to or use the transport layer in order to effectively communicate across the network.	HTTP, Telnet, FTP, SNMP, DNS, SMTP, HTTP and many others

Table 1-4-2-T1: Description 4 layered architectures of TCP/IP model (ePipe, n.d.)

Client-Server architecture is used as the form of TCP/IP model for interaction of two application over network. Figure 1-5-F2 shows the TCP/IP model in form of Client-Server architecture.

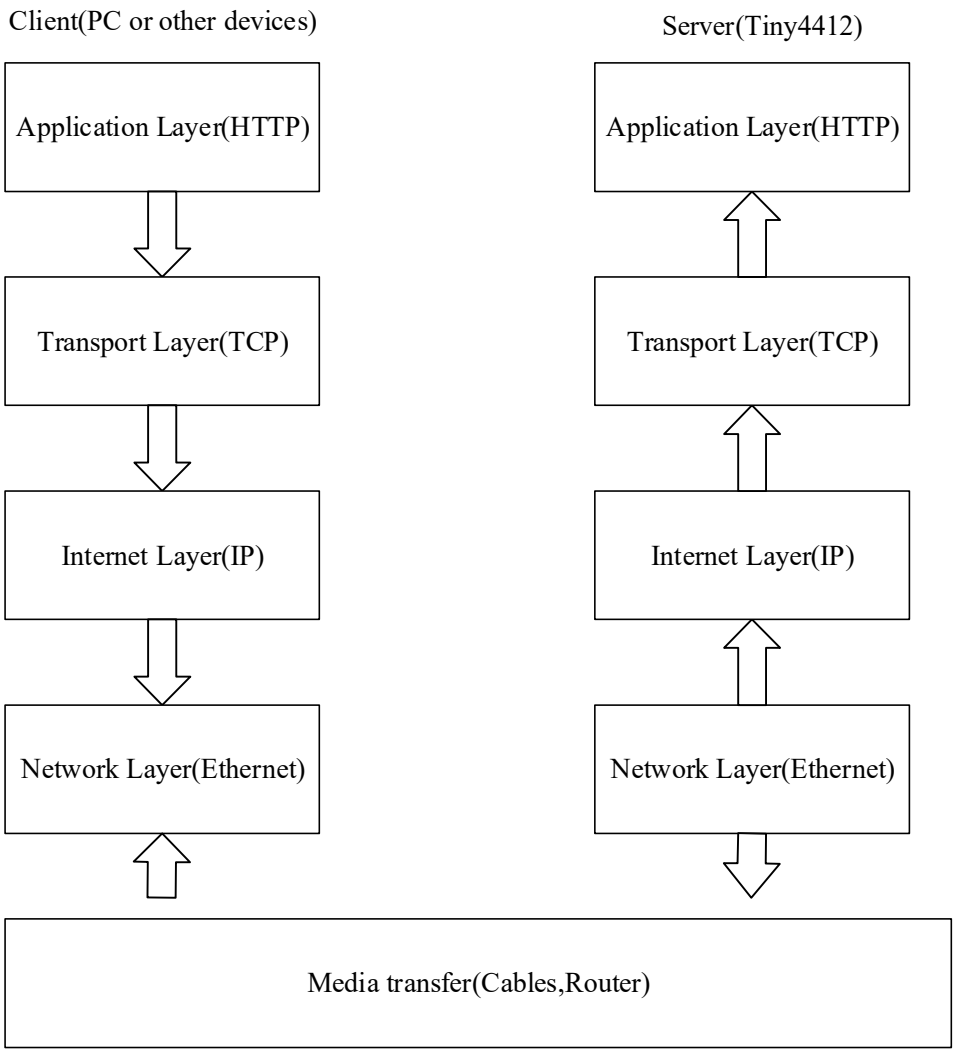

Figure 1-4-2-F2: TCP/IP model in form of Client-Server architecture

1-5 Achievement

An embedded application which consists most of the oscilloscope features is successfully designed and developed. Furthermore, the embedded application is successfully send ADC and data status to the connected device. The connected device is able to receive ADC and data status from the embedded application and plot the ADC data in graph. Figure 1-5-F1 shows screenshot of designed embedded oscilloscope application Figure 1-5-F2 shows the screenshot of PC-based application that received signal data.

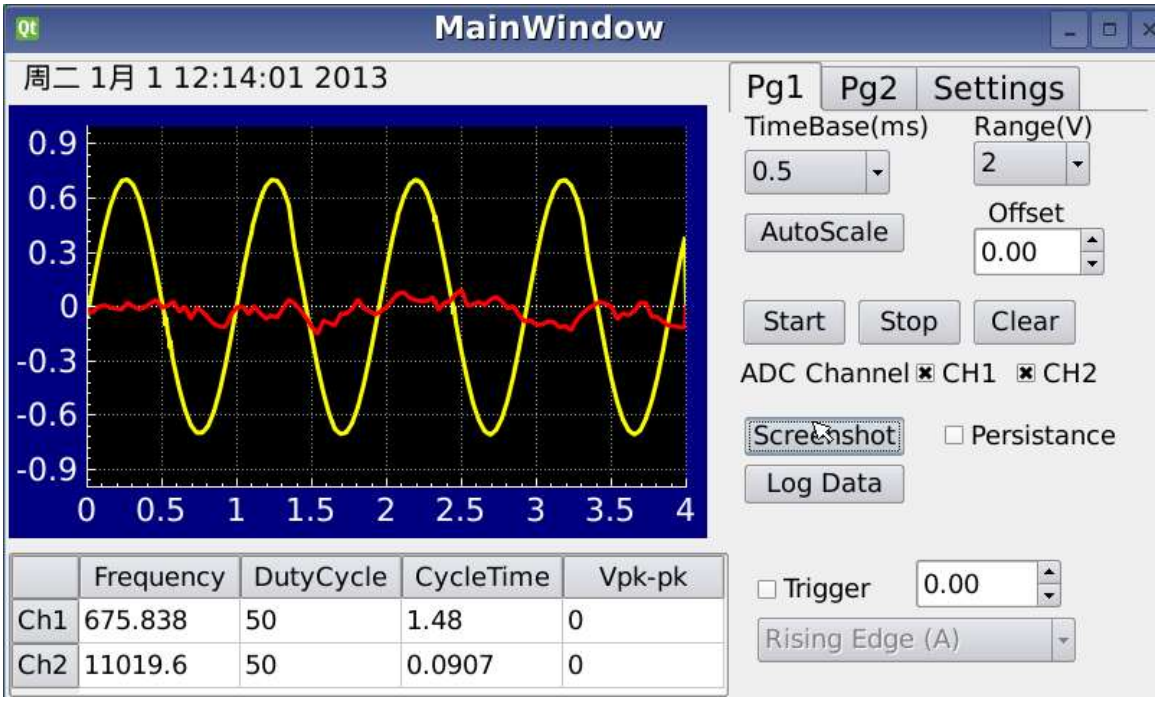

Figure 1-5-F1: Screenshot of designed embedded oscilloscope application

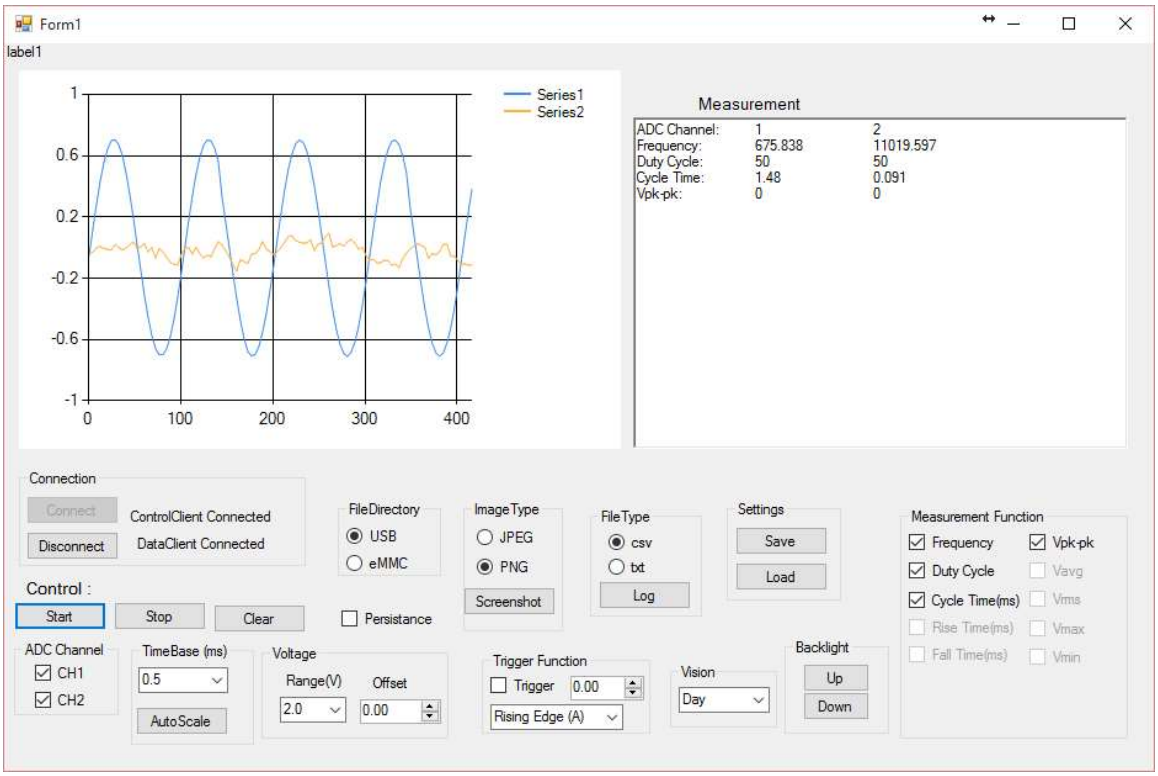

Figure 1-5-F2: Screenshot of PC-based application that received signal data.

1-6 Report Organization

The report of this project has 6 chapter which are introduction, literature review, embedded oscilloscope application, PC-based oscilloscope application and conclusion.

In the chapter 1, the introduction is mainly briefing about background and problem statement of the project, objective, achievement and organization of report. In chapter 2, there are 3 relevant previous work discussed and included into literature review.

In chapter 3, this chapter is briefly explained about hardware and software specification. In chapter 4, hardware design is discussed about block diagram of hardware and used hardware tools. The used software and software design of embedded oscilloscope application is discussed in chapter 5. In software design part, the user interface and algorithm for each part of function is explained. The flow chart diagrams or screenshots of coding are shown to have easier understanding.

In chapter 6, PC-based application is explained in this chapter. This chapter has included the specification of application, software design. The design of user interface and defined functions of application is included in the software design part.

The testing results from both applications is included into chapter 7. In the part, the theoretical result is calculated and compared with the actual result from the application.

In the last chapter, chapter 8 is the conclusion of the report which has the summary of the project, project contributions, limitation and improvement of project.

CHAPTER 2 LITERATURE REVIEW

2-1 Data Acquisition through Cable

There are several solutions that other developers have invented to solve the problem of data acquisition from oscilloscope. Emelia, Christian and Bernard (2002), they have invented solution that is using the standard interfaces RS232 and General Purpose Interface Bus(GPIB) to collect data from oscilloscope. A hardware independent software is developed by them to control device and collect data (Emelia, Christian and Bernard, 2002). The software can support data acquisition from maximum 13 devices and 52 channels over the GPIB interface and maximum four oscilloscope through serial ports (Emelia, Christian and Bernard, 2002). Simultaneous data acquisition is allowed from more than one oscilloscope because each oscilloscope is addressed individually (Emelia, Christian and Bernard, 2002). The software interface is design as the oscilloscope control panel and provide several function such as data processing, device control and interface configuration (Emelia, Christian and Bernard, 2002). The GPIB interface have high speed of data collection and easy to be used. However, the row data cannot be processed with higher resolution due to byte format of digital data. In addition, the data acquisition cannot be done in real time because the writing action in data memory is blocked by every read command. In my project, wireless connectivity is used instead of cable. Data rates can reach approximately 54Mbps by implementing internet protocol suite into oscilloscope.

2-2 Data Transferring through Bluetooth Connection

Another solution to acquire data from oscilloscope is implementation Bluetooth module into oscilloscope. According to Seneviratne and Abhayasinghe(2013, p.103), input voltage signals are captured and transmitted by Bluetooth embedded device to external devices such as an Android smartphone. An Android application is developed for processing and displaying data because the smartphone which has built in Bluetooth function can be connected with Bluetooth embedded device in wireless connection. Host Controller Interface (HCI) has been chosen as the mode of Bluetooth module in order to fully utilize Bluetooth bandwidth due to not able achieving 2Mbps of data rates

(Seneviratne and Abhayasinghe, 2013, p.104). Standard mode of Bluetooth embedded device stand for on-board stack running on Bluetooth module. While HCI mode is chosen, the stack is not running on Bluetooth module but implemented on host processor. The host processor need to interface with module by either UART or USB. Obtaining maximum throughput and implementing custom profile on Bluetooth module is the strength of HCI mode. The Android application main interfaces is design as Figure 2-2-F1 and provide oscilloscope functions and indicate connection status. Users are able to switching channel mode and changing voltage and time division in the settings interface. Figure 2-2-F2 shows the settings interface of portable oscilloscope Android application.

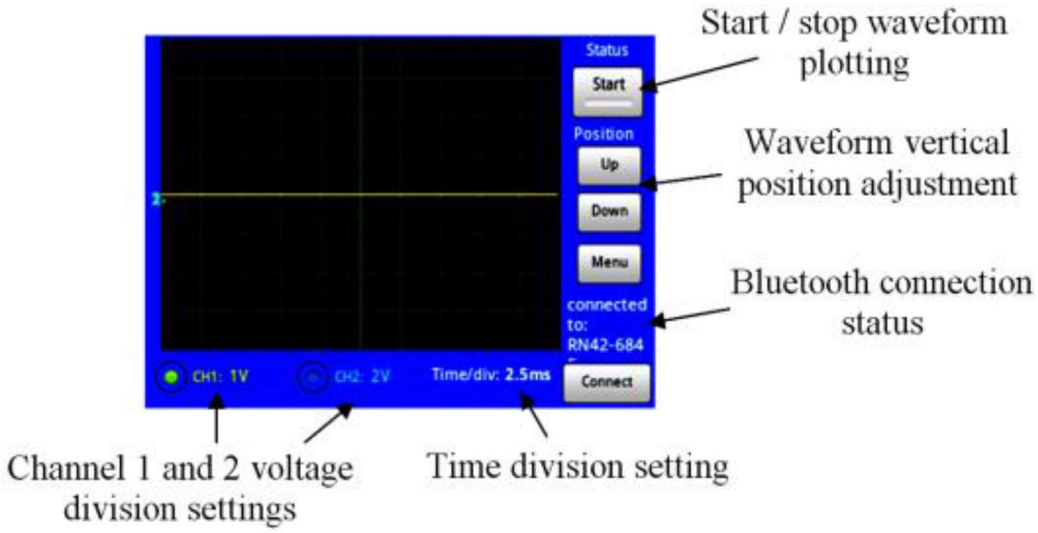

Figure 2-2-F1: Main interface of portable oscilloscope Android application (Seneviratne and Abhayasinghe, 2013, p.105)

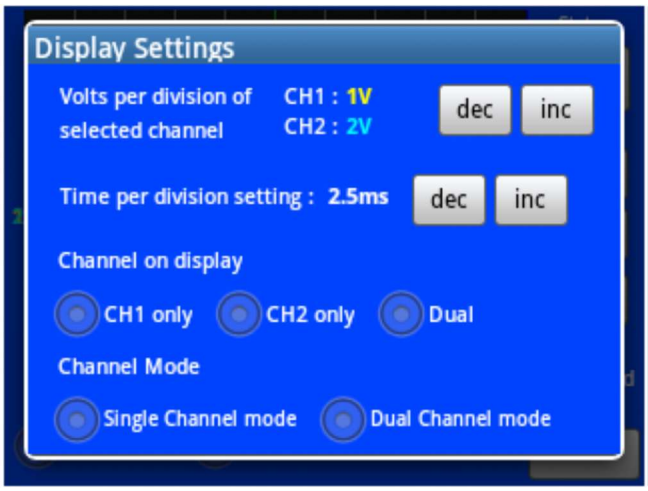

Figure 2-2-F2: Settings interface of portable oscilloscope Android application (Seneviratne and Abhayasinghe, 2013, p.106)

The strength of this solution is that the android application can be updated to implement more functions without changing the hardware (Seneviratne and Abhayasinghe, 2013, p.104). The wireless connectivity of Bluetooth embedded device solve the problem of product testing which is mention above. Unfortunately, the Bluetooth embedded device has short range of connectivity. For example, the users who have out of range of Bluetooth connection are not able to collect the data from oscilloscope. Furthermore, the data rates is not enough fast to acquire data in real time. Within my solution, the users can collect data from oscilloscope without limited range with oscilloscope as long as the internet connection is available.

2-3 Data Transferring over Wired LAN

TCP/IPv4 and Ethernet 10BASE-T/100BASE-TX Debugging with the MSO/DPO4000B Series Oscilloscopes (TestEquity, 2010) defines a solution that implement Internet Protocol Suite into oscilloscope using wired local area network(LAN). This solution is quite similar to my solution that using the internet protocol but the only difference is the hardware connection and transport layer of internet protocol. The current solution is using TCP protocol as in transport layer. A Category 5(CAT5) cable is used as Ethernet cable for wired LAN between oscilloscope and internet in this solution

(TestEquity, 2010). The wired LAN has a more reliable and secure connection. Besides, it is less costs and less power consumption compared with wireless LAN. The maximum speed of the Ethernet network can theoretical reach 100Mbps (TestEquity, 2010). However, it is not convenient that using the cable for connection. The problem of product test which is mentioned before cannot be solve within cable connection. In addition, many of modern product such as laptop, tablets, smartphone and others do not have Ethernet port and cannot get the connection. Most of the modern product do have the Wi-Fi features that are able to connect with wireless LAN. The transfer speed of 802.11a and 802.11g wireless LAN still can reach approximately 54Mbps (Bradley, n.d.). Although the transfer speed of wireless LAN is slower than wired LAN, but it is enough fast to collect data from oscilloscope. TCP is a connection-oriented protocol which is able to exchange streams of data with reliable and full duplex. TCP guarantee the delivery of packets, arranging sequence of packets and no duplicate packets. Data loss will never be found with TCP because of this. However, the data rates is slower than UDP. UDP is a connectionless protocol and send packets of data continuously without guarantee delivery of packets. Therefore, UDP is faster than TCP but data loss may happen with UDP.

CHAPTER 3 PORTABLE OSCILLOCOPE SPECIFICATION

3-1 Hardware Specification

The following specification is established based on a basic digital oscilloscope hardware specification.

- Sampling Frequency $\leq$ 2kHz
- Resolution $\geq$ 12bit
- Input Voltage Range $\geq$ ±1.8V
- Two channels ADC input
- 4GB eMMC Flash Memory
- USB Flash Drive Port
- 7 inches LCD Touch Screen (800x480)
- Wi-Fi modules

From this basic hardware requirements, an embedded oscilloscope application can be developed.

3-2 Software Specification

This software has two major functions. The first function is sampling ADC data from input signal and display in graph form. The second function is sending the sampled ADC data to other devices through socket.

The others functions of this software is listed as below:

- User friendly interface
- Display data in form of graph
- Different time bases (0.5ms, 1ms, 2ms, 5ms, 10ms, 20ms)
- Different division of voltage ranges (0.1V, 0.5V, 1V, 2V)
- Two ADC channels selection
- Compute measurement functions
    - Frequency
    - Duty Cycle
    - Cycle Time
    - Rise Time
    - Fall Time
    - Maximum voltage
    - Minimum voltage
    - Peak to Peak voltage
    - Root Mean Square voltage
    - Average voltage
- Software trigger system (rising and falling edges)
- Persistence mode

- Set voltage offset
- Auto scale in time bases based on data frequency
- Select internal or external memory as file saved directory
- Data logging in CSV or TXT format
- Screenshot in JPEG or PNG format
- Display current date and time
- Control back light of LCD
- Save and recall last saved settings
- Save and load last saved graph inside the oscilloscope for comparing purposes
- Send sampled ADC to other connected devices
- Remote control features

The detail of the functions mentioned above will be explained in chapter, depends on the hardware specification.

CHAPTER 4 EMBEDDED APPLICATION HARDWARE DESIGN

4-1 Block Diagram

Based on the hardware requirement, a block diagram is designed and implemented the oscilloscope application. Figure 4-1-F1 shows the hardware block diagram of portable oscilloscope.

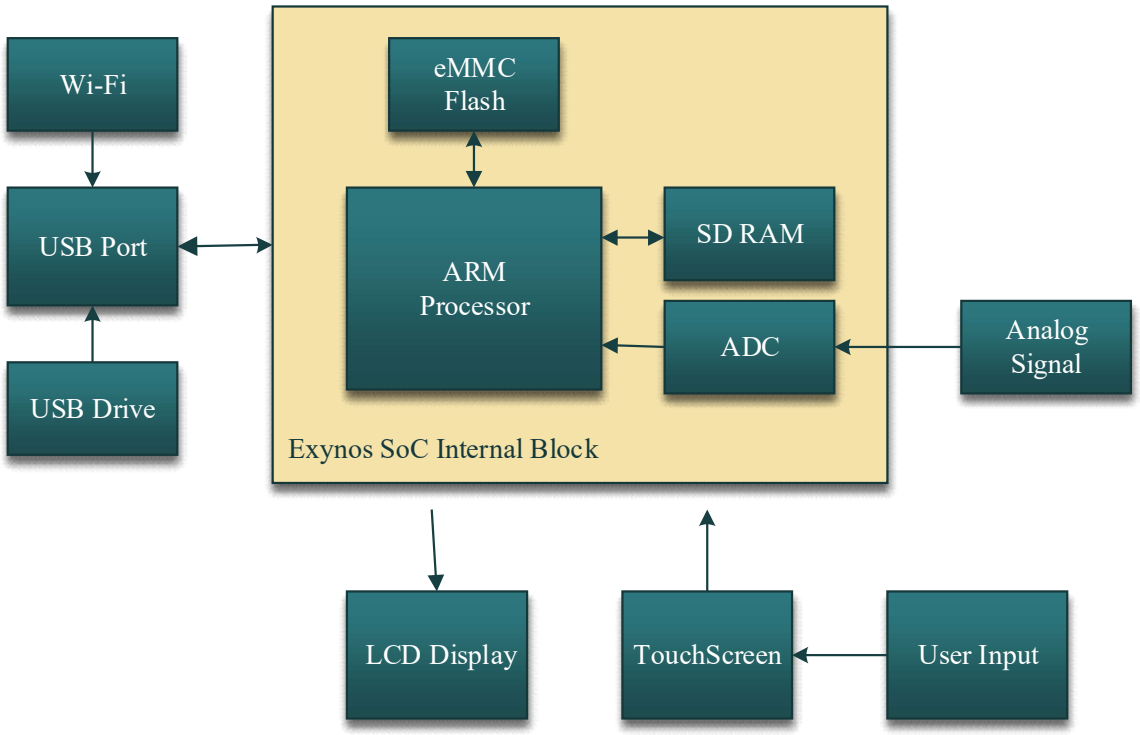

Figure 4-1-F1: Hardware block diagram of portable oscilloscope.

4-2 FriendlyARM Tiny4412

In this project, Tiny4412 board is chosen as hardware device to integrate with embedded oscilloscope application. The specification of Tiny 4412 board meets the hardware requirements for this project. Tiny4412 board is Samsung Exynos4412 ARM Cortex-A9 core board with high performance. Samsung Exynos4412 consists of quad core and used as the main processor. Exynos4412 is integrated by Mali-400 MP as GPU. Therefore, it support 3D graphic acceleration and high definition 1080p. Besides, it has built in 1G DDR3 RAM and 4G eMMC Flash memory. Tiny 4412 is suitable to develop high technology embedded products. The board is built in with internal Analog Digital Converter (ADC) and the ADC is shared by 4 input channels to get analog signals and convert to digital values at maximum conversion rate in 1MSPS with 5MHz with A/D converter clock. The internal ADC supports the 10bit or 12bit in resolution. The input voltage ranges of ADC is 0 to 1.8V. In addition, it consists of three USB Host 2.0 drive ports. Figure 3-1-2-F1 shows the Tiny4412 board. Figure 4-2-F1 shows the Tiny4412 board.

Figure 4-2-F1: Tiny4412 board (Tiny4412, n.d.)

4-3 Wi-Fi Modules

Wi-Fi USB Adapter GWF-3A33 is used as the Wi-Fi module for wireless connection between Tiny4412 boards with switch. The adapter is using the chipset of Ralink RT3070 and standard protocol which is 802.11n in 2.4GHz. The antenna is built in PCB board. The adapter supports in different operating systems which are Window, Linux, and Mac OS. Figure 4-3-F1 shows the image of Wi-Fi USB Adapter GWF-3A33.

Figure 4-3-F1: Wi-Fi USB Adapter GWF-3A33

CHAPTER 5 EMBEDDED APPLICATION SOFTWARE DESIGN

5-1 Software Design

An oscilloscope application is designed to integrate with hardware in order to build a digital oscilloscope system. Embedded Linux is used as the operating system in tiny4412 board. The oscilloscope software is designed to run as an application in Embedded Linux. The software and IDE used to develop the software are Qt Creator and Tera Term.

5-2 Qt Creator & QCustomplot

The embedded application is developed with Qtopia (Qt) GUI that runs on Embedded Linux OS. Qt Creator is used as IDE to develop the embedded application. Users can design their GUI, write codes, profile program, compile and debug application.

Qt Designer, features of Qt Creator is used for editing GUI. First, a form is created and objects such as text label, check box push button and others can be dragged and droppedon the form in form design view. Figure 5-2-F1 shows the screenshot of UI form and objects in Qt Designer.

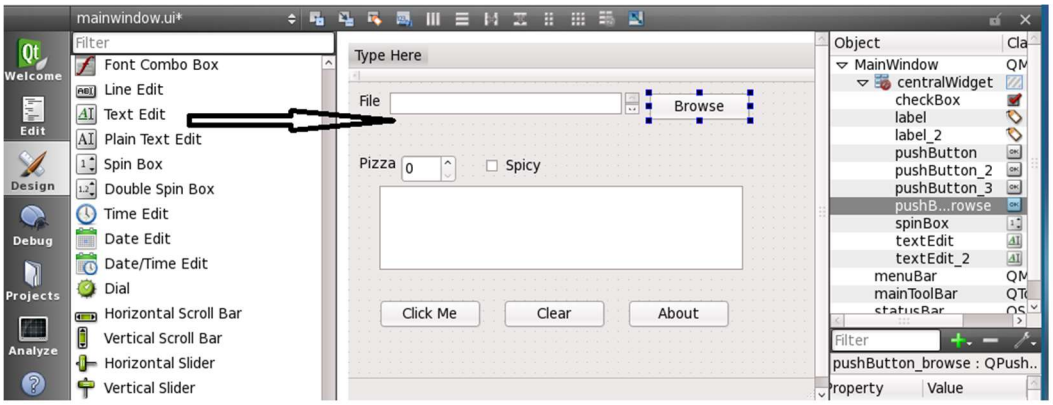

Figure 5-2-F1: Screenshot of UI form of and objects in Qt Designer

In the GUI of oscilloscope embedded application, a graph plot is required to display the waveform of voltage signals. Therefore, QCustomPlot is used to perform the graph plot because it is a Qt C++ widget to perform data visualisation and plotting graph. Several steps is need to follow in order to use QCustomPlot. The source code of QCustomPlot need to be implemented into project and a widget is placed on form with sufficient space to plot a graph. Figure 5-2-F2 shows screenshot of implementation source codes of QCustomPlot into project. Then, a function codes for plotting graph is need to be developed. Tiny 4412

has an ADC feature that is able to convert analogue data to digital data or convert digital data to analogue data. An ADC function codes is implemented into the project to read the ADC values.

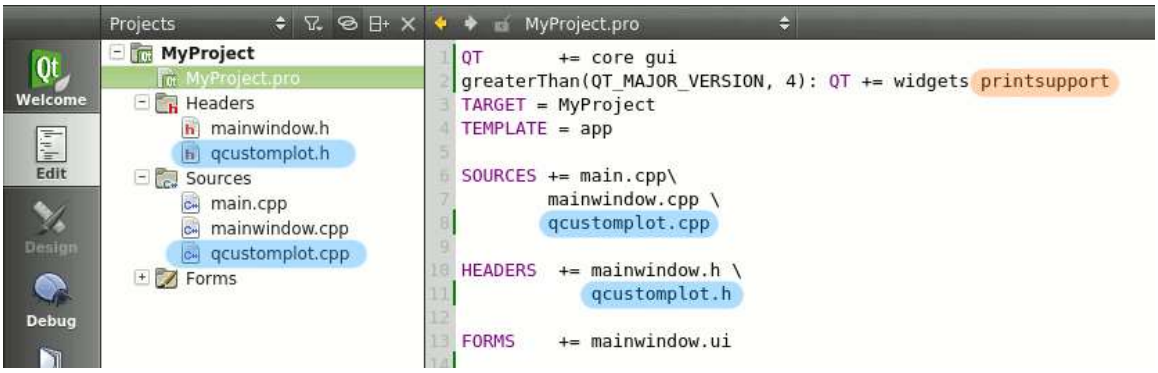

Figure 5-2-F2: Screenshot of implementation source codes of QCustomPlot into project

5-3 Tera Term

In embedded system based on Embedded Linux, serial port is the way for troubleshooting and debugging tools. Developers can send the execution command to embedded devices through serial terminal in order to debug or monitor the system. Besides, developers can control over the embedded device through serial terminal. Therefore, Tera Term is used in this project to control Tiny4412. Figure 5-3-F1 shows the screenshot of Tera Term connected with Tiny4412 through serial terminal.

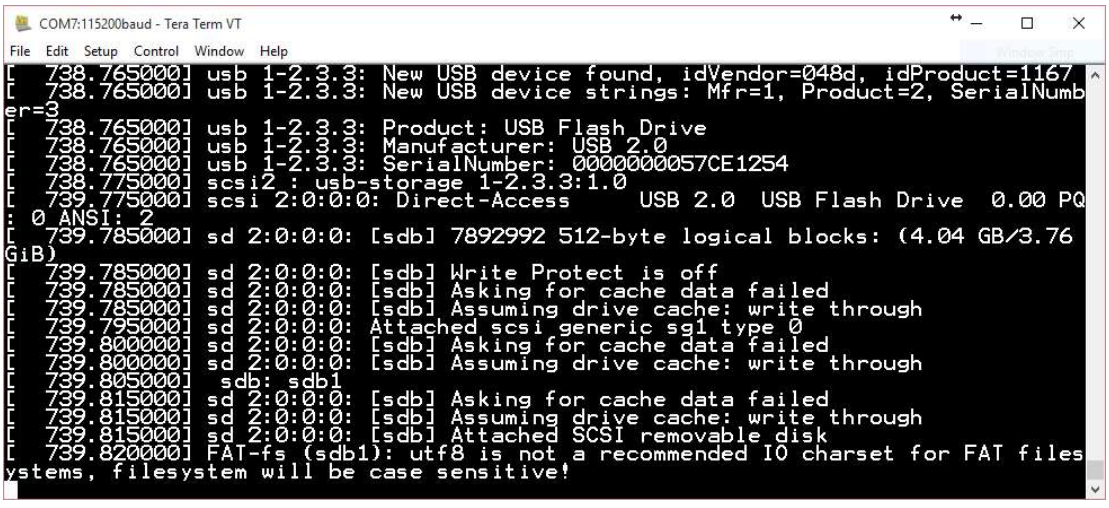

Figure 5-3-F1: Screenshot of Tera Term connected with Tiny4412 through serial terminal

5-4 Flow Chart of Oscilloscope System

The flow chart of the oscilloscope application is designed and shown in Figure 5-4-F1.

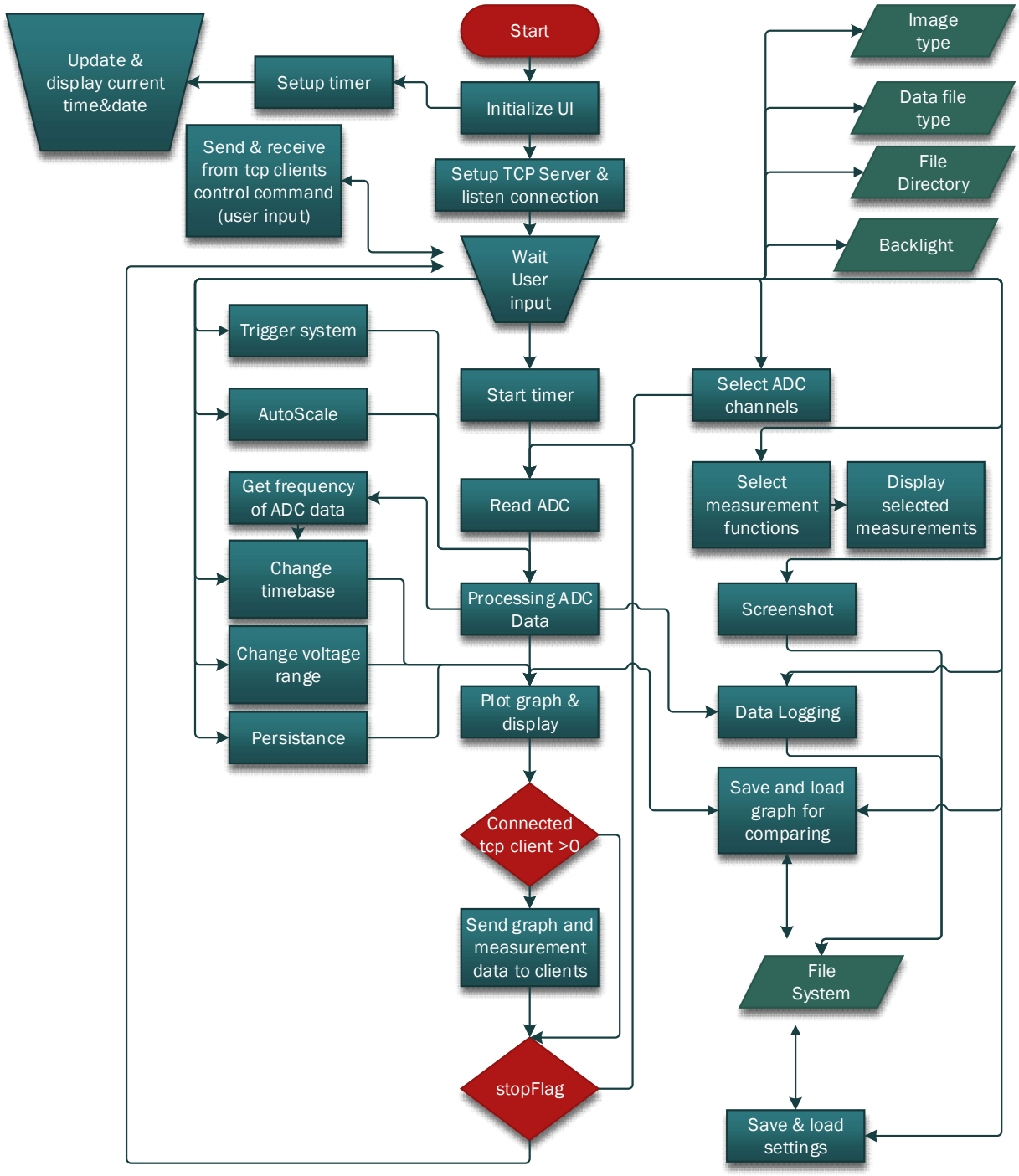

Figure 5-4-F1: Flow chart of oscilloscope system

5-5 User Interface

When the oscilloscope application is started, the GUI of application is initialized and waited for user inputs to do actions respectively when the event is occurred. Figure 5-5-F1 shows the designed GUI of oscilloscope application.

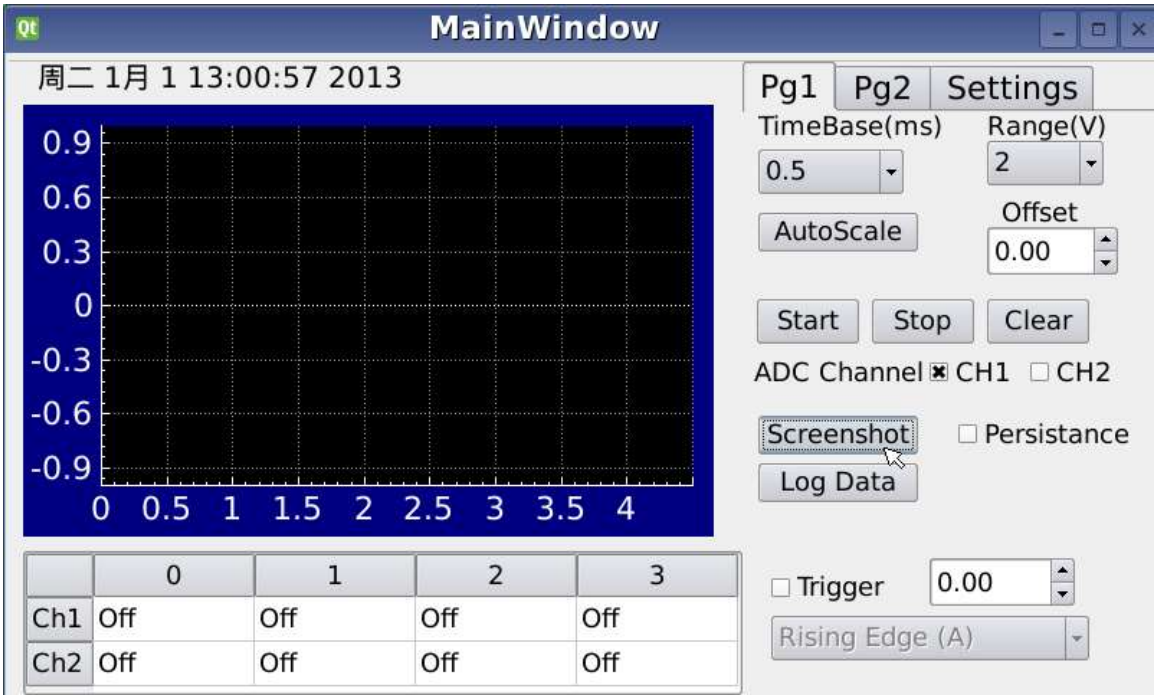

Figure 5-5-F1: Screenshot of designed oscilloscope GUI

The top left of GUI shows the current date and time. The middle left is the graph to show the sampled ADC data and the bottom left is a table that shows the measurement output values.

Due to the constraint of screen size that cannot designed the all the buttons and checkboxes in one page, QTabWidget is used to tackle this issue. QTabWidget is a class that can contains stack of pages. There is only one page is shown in GUI and the other pages is hidden until the tab of pages are selected. As the designed GUI above shown, there are 3 pages in the QTabWidget that are named as Pg1, Pg2 and Settings

In Pg1, most of the control functions such as start, stop, clear graph, changing time base and voltage range, selecting ADC channels, triggering system, persistence mode, auto

scale function, screenshot and data logging. . Figure 5-5-F2 shows the page 1 of designed tab widget.

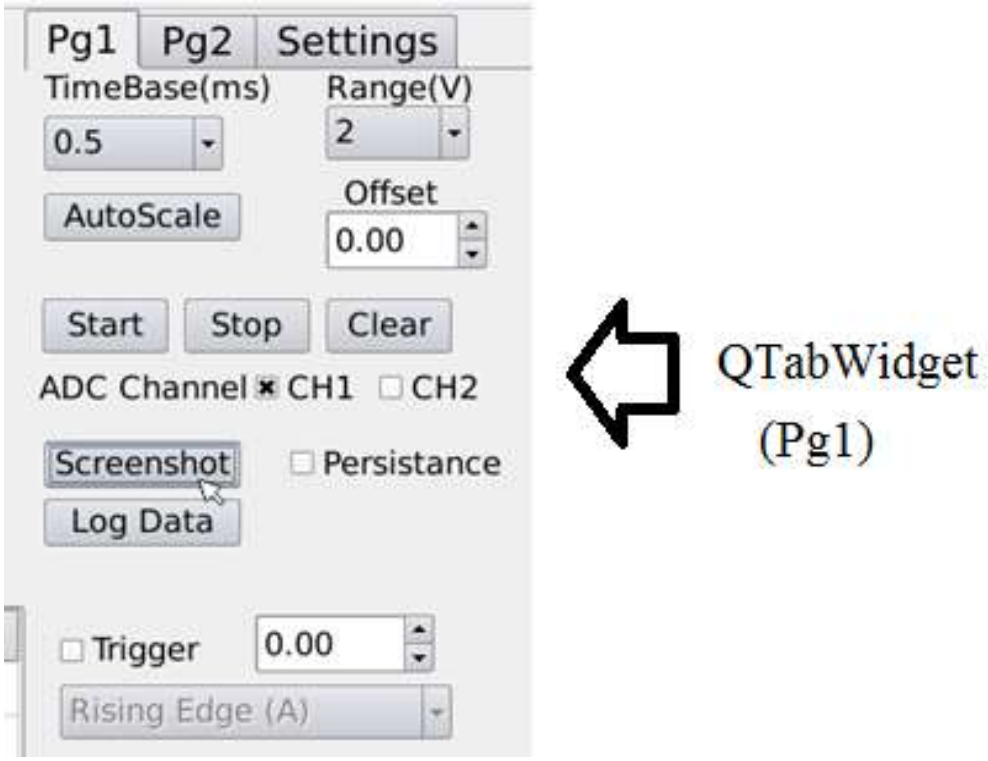

Figure 5-5-F2: Page 1 of designed tab widget

In second page, Pg2, the measurement functions of the oscilloscope application is shown inside. Users may select the measurement functions by clicking the checkbox to display the measurement respectively. The maximum selections of the measurement functions is 4 only. If the maximum selections is reached, the other measurement checkboxes will be disabled until the user uncheck the selected measurement checkbox. The measurement functions include frequency, duty cycle, cycle time (ms), rise time (ms), fall time (ms), Vpk-pk, Vavg, Vrms, Vmax and Vmin. Figure 5-5-F3 shows the page 2 of designed tab widget.

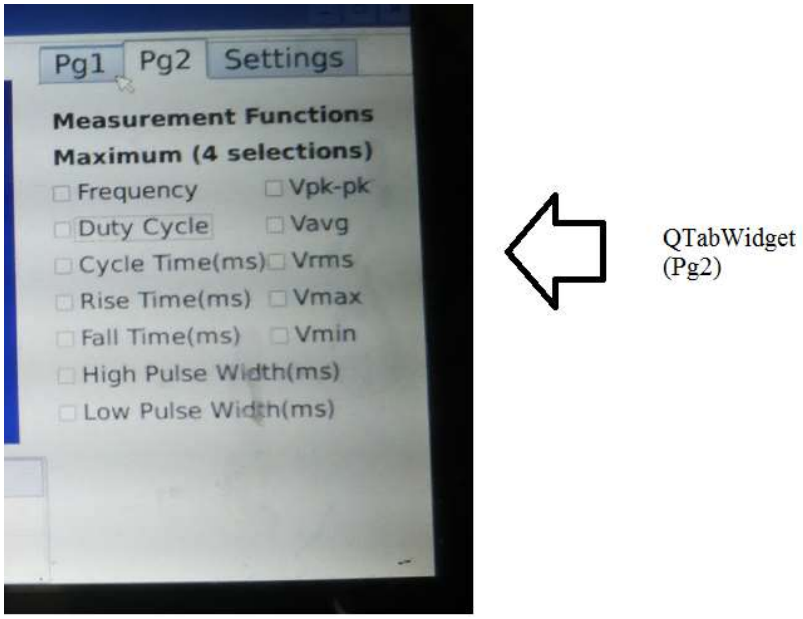

Figure 5-5-F3: Page 2 of designed tab widget.

In third page which named as Settings, the save file directory, types of image, types of data file, visions, brightness of LCD display can be determined. Furthermore, settings can be saved and loaded for next time running. For the buttons in bottom Settings page, user can saves the current graph of ADC channel respectively and loads in next time for comparing purpose. Figure 5-5-F4 shows the settings page of designed tab widget.

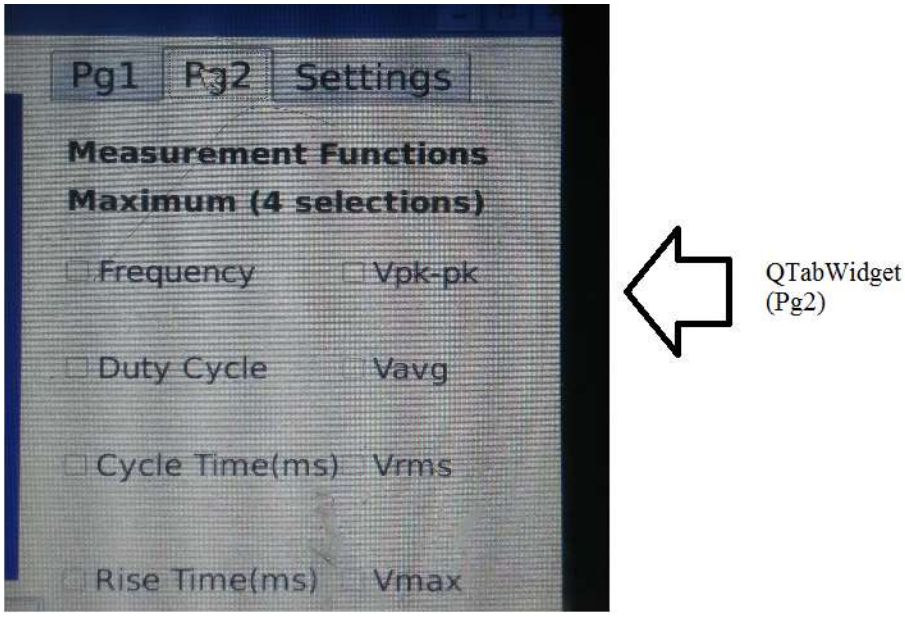

Figure 5-5-F4: Settings page of designed tab widget

In order to keep GUI in more responsive, the event of sampling ADC data and sending ADC data to other devices through socket is move to created thread. Due to quad core processor is used the tasks can be run in parallel. Figure 5-5-F5 shows the coding to move sampling ADC action and socket action to other created threads. The BackgroundWorker class contains the sampling ADC actions and Socket class has the action of creating socket and other socket actions.

```
QThread *t = new QThread;
bw->moveToThread(t);
t->start();
soc = new Socket();
QThread *t2 = new QThread;
soc->moveToThread(t2);
t2->start();
```

Figure 5-5-F5: Coding to move sampling ADC action and socket action to other created threads

5-6 Socket

One of the features of this oscilloscope application is able to send sampled ADC data and measurement values to other devices through socket programming. Therefore, a TCP server is created to send the data to the connected clients. After initialized GUI of the application, TCP server is initialized and listens to the incoming clients' connection. The oscilloscope will send sampled data and measurement values to connected clients if there are any clients connected. Therefore, this TCP server will be called as TCP data server. The client-server architecture is used as communication architecture.

5-6-1 Flow Chart of socket programming

Client socket and server socket is created and bounded to specific port number by calling socket functions. Server is listening to the socket to make connection request. Client can identify the running server and listening port number. Client then sends connection request to server and waits for its respond. Once the server accepted the connection request, data is ready to be sent or received between clients and server. Figure 5-6-1-F1 shows the interaction between clients and server in flow chart.

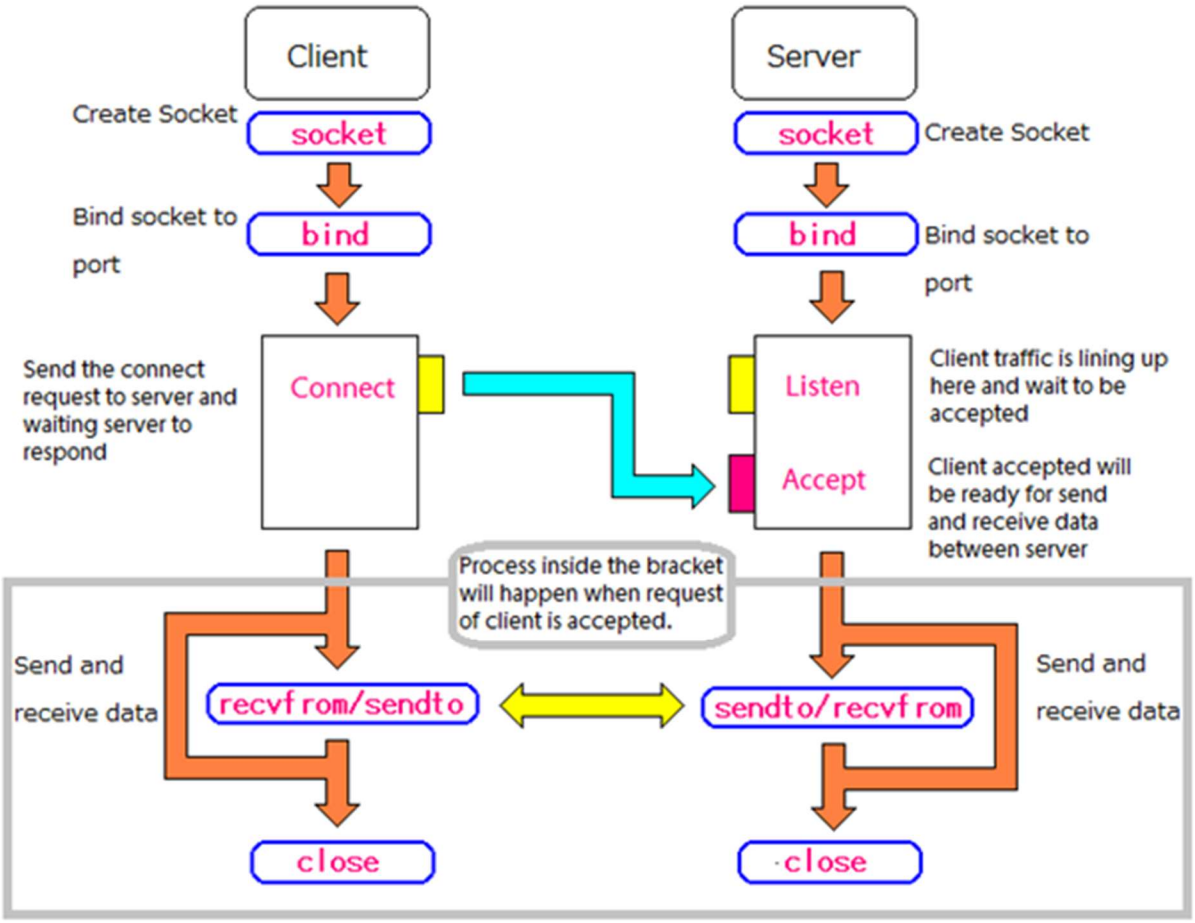

Figure 5-6-1-F1: Interaction between client and server (Edison, 2014)

5-6-2 Remote Control features

One more TCP server is created to handle the remote control functions between the oscilloscope application and other devices. This TCP server will be named as TCP control server. TCP control server is created and initialized with different port number of TCP data server. When there are any incoming client's connection, TCP data server will process the incoming connection followed by TCP control server. The TCP control server will send the command when there is any event occurred. The command is included with the details of event occurs. At the same time, TCP control server is ready to received command that sent by other devices to invoke the specific events occurred.

5-6-3 Implementation

The algorithm used to implement socket into oscilloscope application is shown in flow chart below. Port number for TCP data server is 9999 while port number for TCP control server is 8888. Figure 5-6-3-F1 shows the flow chart of TCP data server. Figure 5-6-3-F2 shows the flow chart of TCP control server.

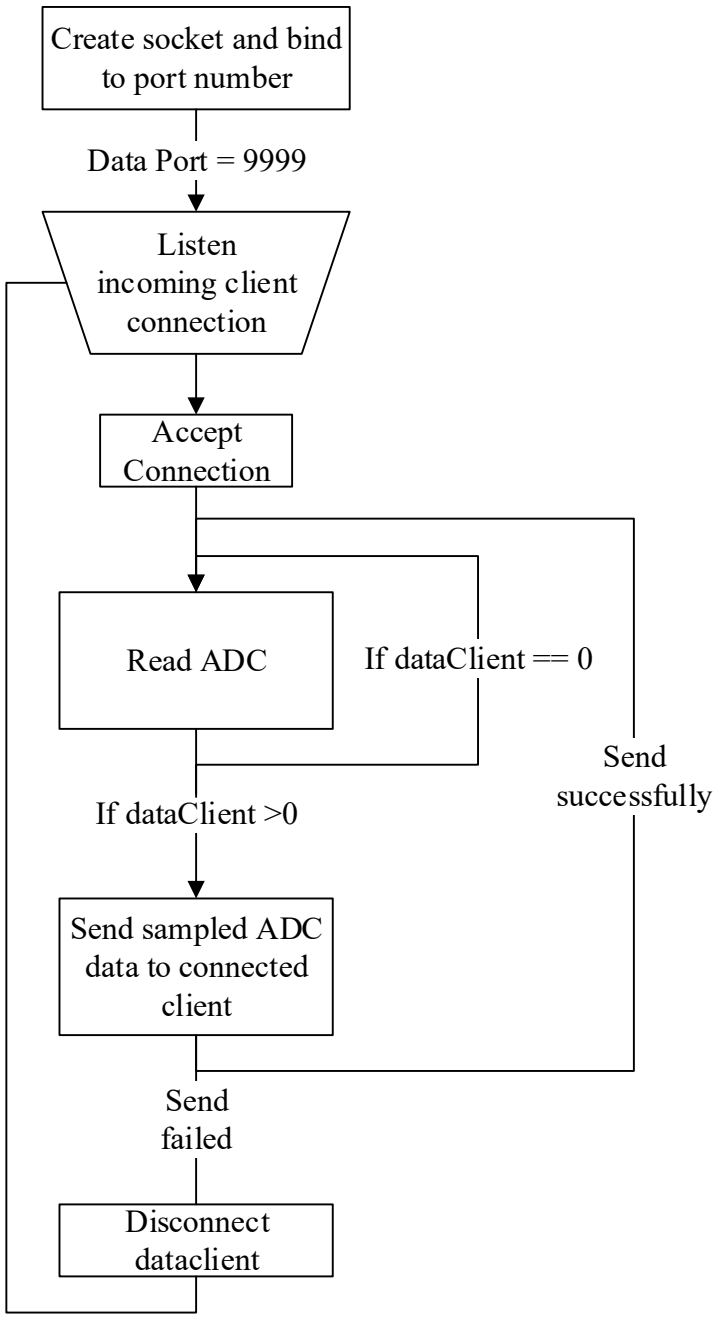

Figure 5-6-3-F1: Flow chart of TCP data server

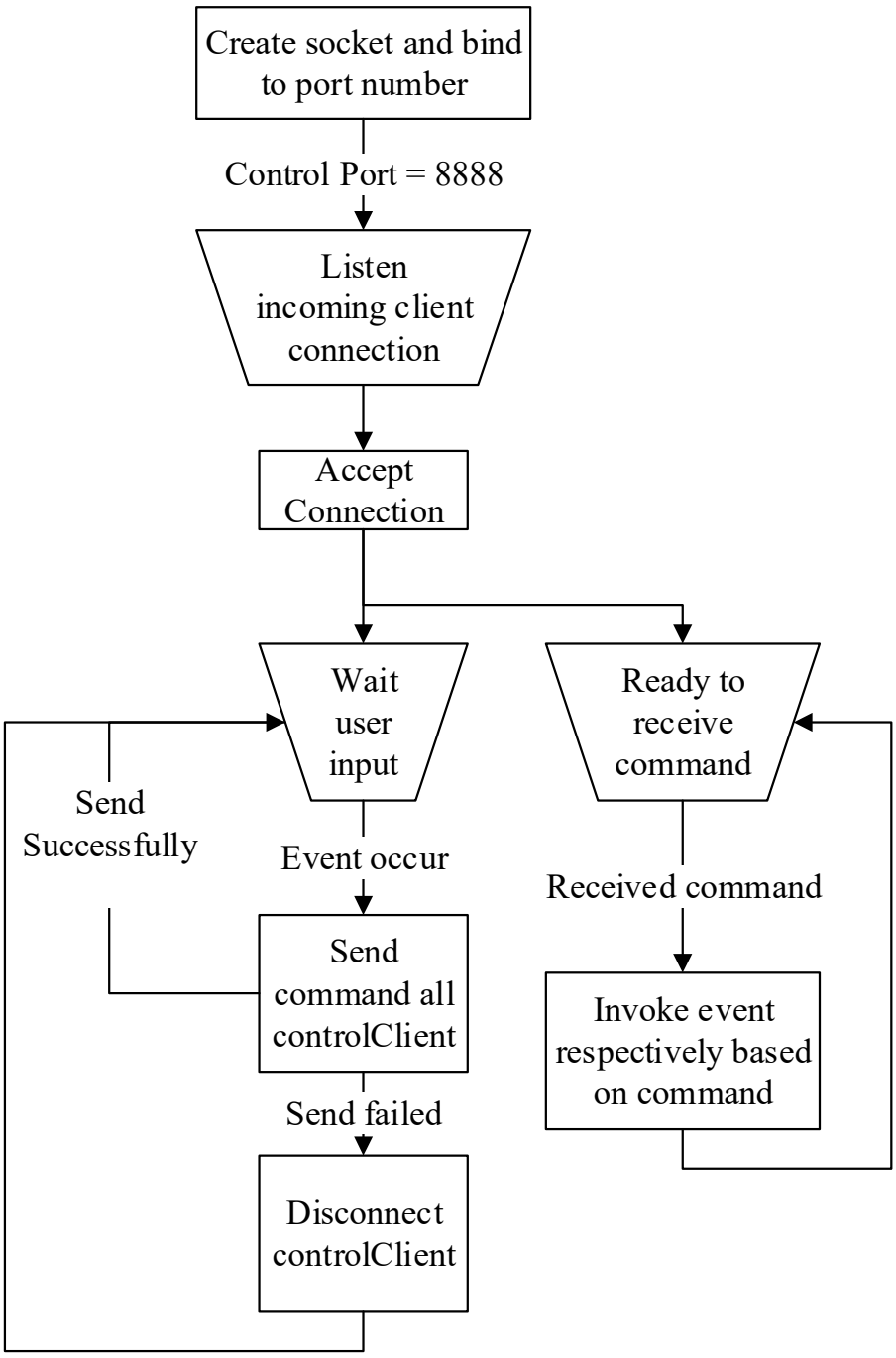

Figure 5-6-3-F2: Flow chart of TCP control client

5-7 Time base

Time base is the x-axis of graph and represented as time. It is controlled by Seconds-per-Division (Sec/Div). The selections of time base are 0.5ms/div, 1ms/div, 2ms/div, 5ms/div, 10ms/div and 20ms/div. By selecting the more time in one division, the more data are needed to be displayed. A formula is figured by expecting 1 kHz of periodic waveform has 26 counts of data.

$$total\ data\ (each\ division) = 26 * selected\ timebase$$

In the graph settings, there is 8 divisions in the x-axis of the graph. Therefore, total data for display are 8 times of the total data (each division). Table 5-7-T1 show the table of time bases and its total data respectively.

Time base (ms)/div	Total data of each division	Total data of 8 divisions
0.5	13	104
1.0	26	208
2.0	52	416
5.0	130	1040
10.0	260	2080
20.0	520	4160

Table 5-7-T1: Table of time bases and its total data respectively

In order to implement the customized time base value, the tick label and value of graph must be set manually. The required labels and values is saved into an array then call setTickVector() and setTickVectorLabels(), that are functions of QCustomPlot to set values and labels of time base in graph. Figure 5-7-F1 shows the screenshot coding of manually set the time base values and labels.

```
double dataUnit = (val/1000)/(0.001/26);
for(int i = 0; i < 8; i++)
{
    n += val;
    totalDispUnit += dataUnit;
    graphUnit += 52;
    tickValue<<graphUnit;
    tickLabel<<QString::number(n);
}
ui->customPlot->xAxis->setTickVector(tickValue);
ui->customPlot->xAxis->setTickVectorLabels(tickLabel);
```

Figure 5-7-F1: Screenshot coding of manually set the time base values and labels

5-8 Voltage Range

The voltage range is the y-axis of the graph and representing the voltage level that display in graph. The combo box of voltage ranges has the selection of 2V, 1V, 0.5V and 0.1V. After changed the voltage range, the graph will be refreshed.

5-9 Read ADC

5-9-1 Modify ADC kernel

By default, the ADC driver will return a value when user space application read ADC device file. This behavior may cost much time in returning the value from ADC driver to user space application and slow down the performance. Therefore, the ADC kernel is modified to return 500 ADC values each times instead of return a value each times. The ADC kernel store the ADC values into char buffer array and send to user space application. Figure 5-9-1-F1 shows coding of modified ADC kernel in sampling function.

```
int i;
for(i=0;i<arr_size;i++)
{
        value = exynos_adc_read_ch();
        valueArr[i] = value;
        sprintf(singleStr,"%d",valueArr[i]);
        memcpy(str+pos,&singleStr, value_size);
        pos += 5;
}

if (count >= sizeof(str)) {
        int r = copy_to_user(buffer, str, sizeof(str));
        return r ? r : sizeof(str);
} else {
        return -EINVAL;
}
```

Figure 5-9-1-F1: Coding of modified ADC kernel in sampling function

5-9-2 Access ADC Driver

In order to read ADC values, ADC driver that installed in tiny4412 is accessed by oscilloscope application. Firstly, the ADC device file is read as normal file to get the ADC values. Figure 5-9-2-F1 show coding of open ADC device file.

```
file = new QFile("/dev/adc");
if(!file->open(QIODevice::ReadOnly))
    return;
fd = file->handle();
if (fd < 0) {
    perror("open ADC device:");
    return;
}
```

Figure 5-9-2-F1: Coding of open ADC device file

5-9-3 Start Button

A timer is required to invoke this readADC() function. The timer is connected to the readADC() functions when timeout signal is emitted by the timer. The timer will be started after clicked event of start button occurred. The readADC() will be invoked when every time the timeout signal is emitted. Figure 5-9-3-F1 shows coding of event of started timer.

```
void MainWindow::setupTimer()
{
    dataTimer = new QTimer(this);
    // setup a timer that repeatedly calls MainWindow::realtimeDataSlot:
    connect(dataTimer, SIGNAL(timeout()), this, SLOT(readAdc()));


//Start sampling ADC data by start timer
void MainWindow::start()
{
    timerStart();
}
```

Figure 5-9-3-F1: Coding of event of started timer by clicked start button

5-9-4 Stop Button

A stop flag will be checked before read ADC values every times. When the stop button is clicked, the occurred event will turn the stop flag become true. Therefore, the stop flag will be checked before start sampling ADC data. If the stop flag is true, then the timer which emits timeout signal to invoke readADC() to stop and the stop flag will be turned false then finally exit the readADC(). Figure 5-9-4-F1 show coding of stop sampling ADC data by clicked stop button.

```
void MainWindow::readAdc()
{
    QMutex mutex;
    mutex.lock();   //prevent other threads from changing the "Stop" value
    if(stopFlag){     // stop sample ADC data when stop flag is set
        dataTimer->stop();
        stopFlag = false;
        return;
    }
    mutex.unlock();
```

Figure 5-9-4-F1: Coding of stop sampling ADC data by clicked stop button

5-9-5 ADC channels checkbox

The readADC() will start sampling ADC data based on the selected ADC channels. When the state of any ADC channel checkboxes is changed, the flag that indicates the state of the ADC channels will be toggled based on state of the checkbox. This flag is used to be checked whether should sample the ADC data from the current ADC channels. Figure 5-

9-5-F1 shows the coding of ADC channel checkbox. Figure 5-9-5-F2 shows the coding sampling ADC data based on selected ADC channels.

```
void MainWindow::adcCh1(int state)
{
    if(state == 2) // Checked
    {
        flagAdcCh1 = true;
        ui->customPlot->graph(0)->setVisible(true);
        controlClientSendCmd("CH1 true;");
    }
    else if(state == 0)
    {
        flagAdcCh1 = false;
        ui->customPlot->graph(0)->setVisible(false);
        controlClientSendCmd("CH1 false;");
    }
}
```

Figure 5-9-5-F1: Coding of ADC channel checkboxes

```
if(flagAdcCh1)
{

    int adcChannel = 1;
    setADCChannel(adcChannel,fd);
    myTimer1.start();
    for(int i=0;i<DATA_SIZE;i++){
        y1[i] = in.readLine().toDouble();
    }
    nMsec1 = myTimer1.restart();
```

Figure 5-9-5-F2: Coding sampling ADC data based on selected ADC channels.

5-9-6 Set ADC channel

In order achieve two channel ADC inputs, the input channels need to be set before start sampling ADC data. To set the ADC channel, a function that has been defined in ADC kernel is used. This function is ioctl() that required the parameter of file description to device file, command, ADC channels that need to be set. Figure below shows the functions that define in the ADC kernel. Figure 5-9-6-F1 shows the defined function in ADC kernel.

```
static long exynos_adc_ioctl(struct file *file,
        unsigned int cmd, unsigned long arg)
{
#define ADC_SET_CHANNEL     0xc000fa01
#define ADC_SET_ADCTSC      0xc000fa02

    switch (cmd) {
        case ADC_SET_CHANNEL:
            exynos_adc_set_channel(arg);
            break;
        case ADC_SET_ADCTSC:
            /* do nothing */
            break;
        default:
            return -EINVAL;
    }

    return 0;
}

static inline void exynos_adc_set_channel(int channel) {
    if (channel < 0 || channel > 3)
        return;

    adcdev.channel = channel;
}
```

Figure 5-9-6-F1: The defined function in ADC kernel.

Due to internal ADC is shared by the four of ADC input pins, different file descriptions to ADC channels respectively are required to sample the correct ADC data according to the ADC input channels. Figure 5-9-6-F2 shows coding of set ADC channel before sampling data.

```
if(flagAdcCh1)
{

    int adcChannel = 1;
    setADCChannel(adcChannel,fd);

// ADC Channel 2
if(flagAdcCh2)
{
    setADCChannel(2,fd2);
```

```
void MainWindow::setADCChannel(int adcChannel, int fd)
{
    if (ioctl(fd, ADC_SET_CHANNEL, adcChannel) < 0) {
        perror("Can't set channel for /dev/adc!");
        return;
    }
}
```

Figure 5-9-6-F2: Coding of set ADC channel before sampling data.

5-9-7 Sample ADC data

After set the correct ADC channels, the application is start to sample ADC data. A for loop is used to read the ADC value and the ADC value is changed to double format. After sampled ADC data, some data processing on ADC data are done based on the selected measurement functions or trigger system. Figure 5-9-7-F1 shows coding of read ADC device file.

```
for(int i=0;i<DATA_SIZE;i++){
    y1[i] = in.readLine().toDouble();
}
```

Figure 5-9-7-F1: Coding of read ADC device file.

5-9-8 Conversion of ADC data to voltage level

The data that acquired from the ADC driver is 0 to 4095. The resolutions of ADC data is 12bit. Therefore, the data need to be converted to voltage level before plotting in graph. A formula is figured out to do this conversion.

$$v[i] =((y[i] / 4095)* 1.8) – 0.9 + offset$$

Due to limitation of hardware specification, the ADC cannot receive the full waveform from voltage input when zero offset of input voltage. Therefore, it is necessary to have 0.9V offset value from input voltage. In the application, 0.9V value will be minus back to act as receiving 0V of offset value.

The last variable of the formula, “offset” is the DC voltage. The default value of the offset is 0.

5-9-9 Plot ADC data in graph

After processing the ADC data, the ADC data is copy to an array that for display purpose only. If trigger system is not invoked, only middle part of ADC data but not all sampled ADC data are plot in the graph. If using the original array of ADC data, the x-axis will be shaken frequently.

To display a graph efficiently, one pixel of graph show one point only. As the width of oscilloscope graph is set as 416, the x-axis of graph should have 416 points only. There are different amount of data to display in graph based on time base. Therefore, different amount of data is shrink or expanded into 416 points in order to display in graph.

Referring table 5-11-T1 that show the total data for displaying according to time base, the total data for 2ms in time base is 416 which means there are no any shrink or expand actions to display the data.

For 0.5ms and 1.0ms of time bases, the total data for displaying is less than 416 points. Therefore, the total data for displaying is expanded by adding average of two points in between of the two points. In 0.5ms of time bases, 416 divided by total data for displaying, 104 is 4. It means that each data is expanded to 4 points. The first point is remained as same with data and the followed 3 points is average of two data and added after each data. The algorithm is also applied to 1.0ms of time bases. Figure 5-9-9-F1 shows example of expansion of one data to 5 points.

```
Before expanded
10 15 20 25 30

After expanded
10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29
```

Figure 5-9-9-F1: Example of expansion of one data to 5 points

For 5ms of time bases, the total data for displaying is 2080. Due to total data for displaying, 2080 divided by 416 is 2.5, it means that 2.5 points representing each data. In order to

shrink this array, 1.5 of points is skipped for each data. Figure 5-9-9-F2 shows example of shrink data in 5ms of time bases

```
Before expand
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

After expanded
1 3.5 6 8.5 11 13.5 16 18.5
```

Figure 5-9-9-F2: Example of shrink data in 5ms of time bases

For 10ms and 20ms of time bases, the total data for displaying is 1040 and 2080. As the result of 2080 and 4160 divided by 416 is 5 and 10. It means that each five points is consider as one data and the first point is act as the data and the following 4 points will be skipped. This algorithm is same applied to 20ms of time bases. Figure 5-9-9-F3 shows example of shrink data that each 5 points to one data by skipped 4 data.

```
Before expand
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27
 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50
After expanded
1 6 11 16 21 26 31 36 41 46
```

Figure 5-9-9-F3: Example of shrink data that each 5 points to one data by skipped 4 data

After doing expansion of shrink of data, the graph is plotted and displayed. The x-axis is represented as time while the y-axis is represented as the voltage. The x-axis is controlled by the time bases. The y-axis can be adjusted by voltage range. Figure 5-9-9-F1 shows the coding of set sampled ADC into graph.

```
    int s = halfDataSize - totalDispUnit/2;
    int e = halfDataSize + totalDispUnit/2;
    for(int i = s; i < e; i++)
        resizeArrGraph(y,dispY,s,e);
}
ui->customPlot->graph(graph)->setData(dispX,dispY);
ui->customPlot->replot();
```

Figure 5-9-9-F1: Coding of set sampled ADC into graph.

5-9-20 Flow Chart

Figure 5-9-20-F1 shows the flow chart of the readADC()

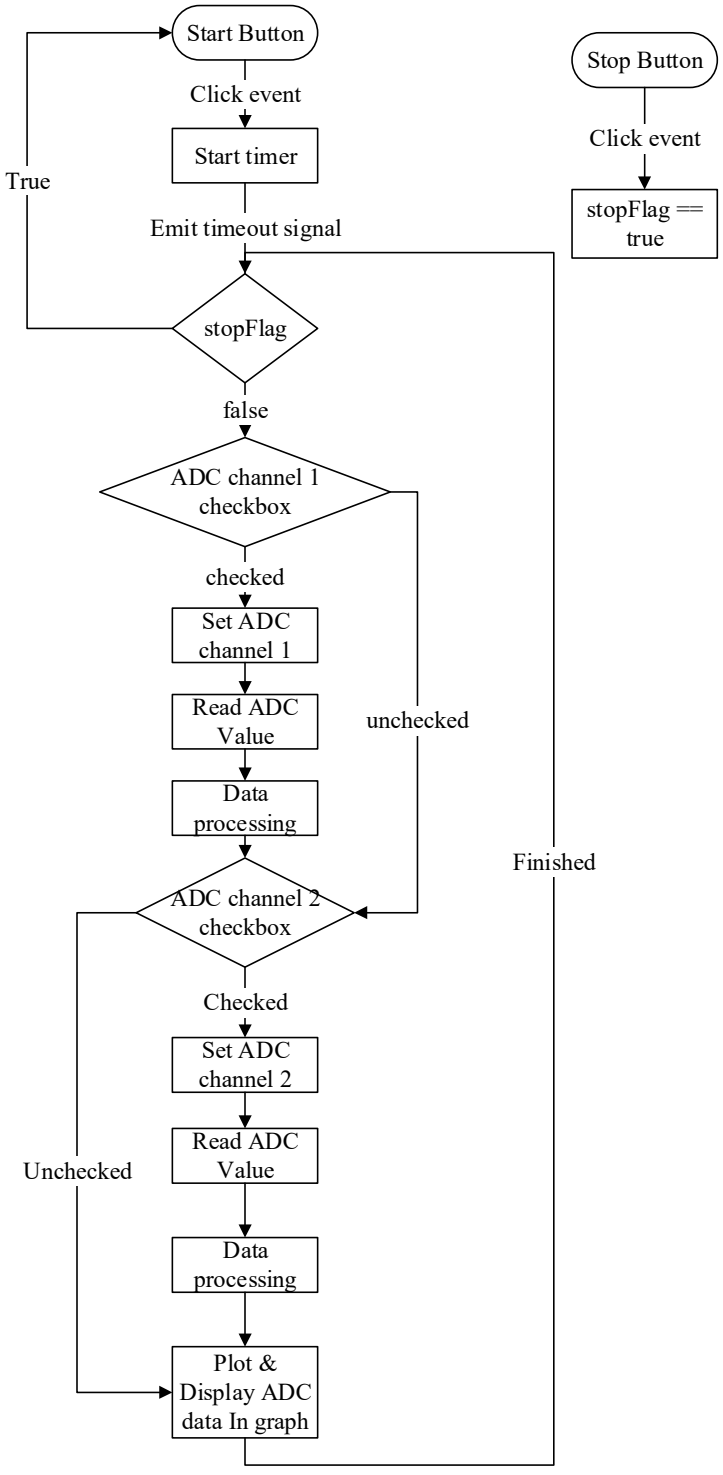

Figure 5-9-20-F1: Flow chart of the readADC()

5-10 Processing ADC Data

The data processing is done depends on the selected measurement functions and trigger system. There are 10 measurement functions defined in this oscilloscope application.

- Frequency
- Duty cycle
- Cycle time
- Rise time
- Fall time
- Peak to peak voltage, Vpk-pk
- Average voltage, Vavg
- Root Mean Square voltage, Vrms
- Maximum voltage, Vmax
- Minimum voltage, Vmin

5-10-1 Cycle Time

Cycle Time is the times in seconds that waveform complete itself from starting until it finished. This also be called as Period, *T* of the waveform. This value can be obtained by the total counts of data in one cycle times with the times of sampled each data from ADC.

Five stages are declared to get the total counts of data in one cycle. The first, second and third stages ensure the data is start from the middle point. The fourth stage will start counting the data which is in positive half cycle. Then the last stage will count the following data which is in negative half cycle. The FOR loop will be stopped after finished counting data in one cycle. If the starting data is in positive half cycle, the first stage will be skipped to second stage. Figure 5-10-1-F1 shows stages diagram of calculate total data in one cycle. Figure 5-10-1-F2 shows coding of the algorithm to get one cycle.

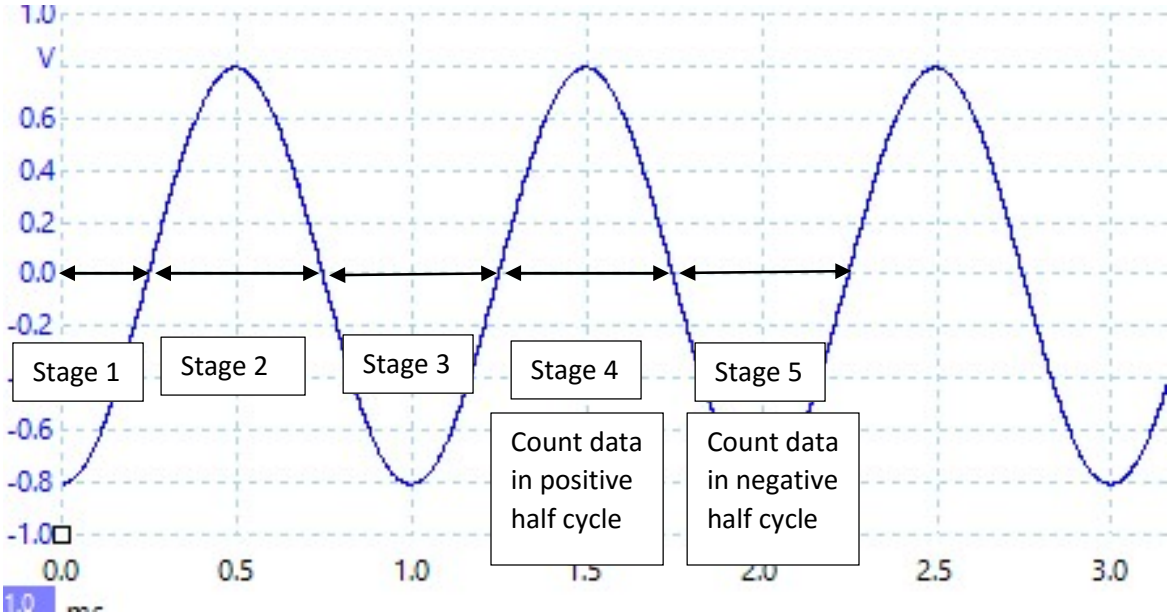

Figure 5-10-1-F1: Stages diagram of calculate total data in one cycle

```
for(int i=sp;i<ep;i++)
{
    if(!flag1[0]){
        if(y1[i] >= 2048)
            flag1[index1++] = true;
    }
    else if(!flag1[1]){
        if(y1[i] < 2048)
            flag1[index1++] = true;
    }
    else if(!flag1[2]){
        if(y1[i] >= 2048){
            pos1++;
            fcycleSP = i;
            flag1[index1++] = true;
        }
    }
    else if(!flag1[3]){
        if(y1[i] < 2048){
            neg1++;
            flag1[index1++] = true;
        }
        else
            pos1++;
    }
    else if(!flag1[4]){
        if(y1[i] >= 2048){
            fcycleEP = i;
            flag1[index1] = true;
            break;
        }
        else
            neg1++;
    }

}
```

Figure 5-8-1-F2: Coding of the algorithm to get one cycle.

After obtained the total counts of data in one cycle, the duration to sample each unit of data is required.

QElapsedTimer is used to calculate the duration of time for sampling ADC data. Before start sampling ADC data, a timer is started and the time after finished sampling ADC data is recorded. The duration for sampling each unit of data can be obtained by the formula below.

Duration for each unit data = Total duration for all data / total counts of data

The duration for the one cycle of sine wave in 1 kHz should be 1ms but the duration obtained from the algorithm is 1.387ms. The calculated sampled time is slower by 1.387 than the theoretical sampled time. This is due to the limitation of hardware specification. To solve this constraint, each calculated unit time is divided by 1.387 in order to get the correct duration.

5-10-2 Frequency

Frequency is the number of cycles occurring per second in waveform. Period, *T* is representing the duration of one cycle and is the reciprocal of frequency, *f*. The standard unit of frequency is in *Hertz, Hz.*

$$f = \frac{1}{T}.$$

In order to the frequency of waveform, the cycle time must be calculated and the formula is implemented to calculate the frequency.

5-10-3 Duty Cycle

Duty cycle is the percentage of one cycle in which the signal is active or on. A formula is used to calculate this percentage.

Duty cycle (%) = (total of positive data / total data) * 100%

5-10-4 Maximum & Minimum Voltage

Maximum voltage is also called as peak voltage. Peak voltage is the value of highest voltage level in one cycle of signal. In the opposite, the minimum voltage is the value of

lowest voltage level. It also be called as the maximum negative amplitude. The algorithm for this function is that the default maximum and minimum voltage is offset value. The maximum voltage will be updated when a higher voltage value is detected. These steps are same for the minimum voltage. Figure 5-10-4-F1 shows the coding of finding maximum and minimum voltage.

```
vMin1 = vMax1 = ui->offsetBox->value;
// Find max & min Voltage
for(int i=sp; i<ep;i++)
{
    if(vMax1 < v1[i])
        vMax1 = v1[i];
    else if(vMin1 > v1[i])
        vMin1 = v1[i];
}
```

Figure 5-10-4-F1: Coding of finding maximum and minimum voltage.

5-10-5 Peak to Peak Voltage

Peak to peak voltage, Vpk-pk is the amplitude change from the maximum positive amplitude to maximum negative amplitude, which mean the difference between maximum voltage and minimum voltage. The algorithm for this calculation is the maximum voltage minus the minimum voltage. Therefore, the maximum and minimum voltage are needed to be calculated before proceed this functions. Figure 5-10-5-F1 shows the explanation of maximum, minimum and peak to peak voltage in diagram.

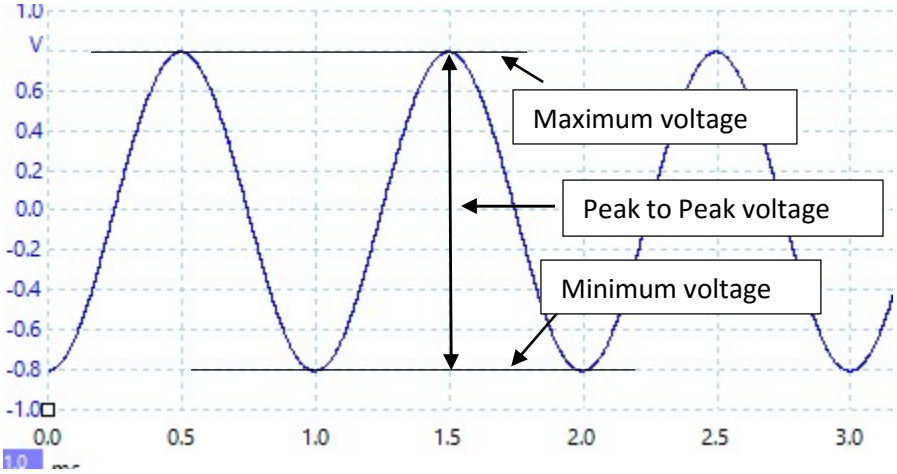

Figure 5-10-5-F1: Explanation of maximum, minimum and peak to peak voltage in diagram

5-10-6 Root Mean Square Voltage

The symbol of Root Mean Square (RMS) voltage is Vrms. The meaning of Vrms is the amount of alternating current power that produces the same heating effect as an equivalent DC power. Figure 5-10-6-F1 shows the explanation of Vrms in diagram.

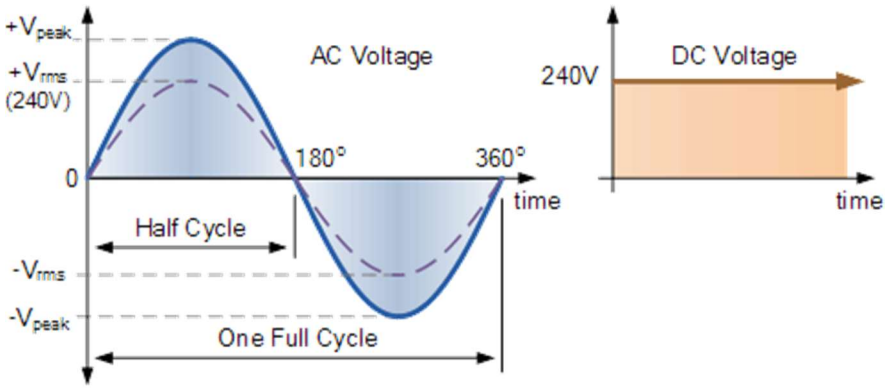

Figure 5-10-6-F1: Explanation of Vrms in diagram

The RMS voltage is the square root of the average or mean voltage of squared function of instantaneous values. The positive cycle of the periodic wave is used for calculation method. The positive cycle is divide to multiple portion which are same portion width with each other. The more portion is divided, the result will be more accurate. Figure 5-10-6-F2 show algorithm of Vrms calculation.

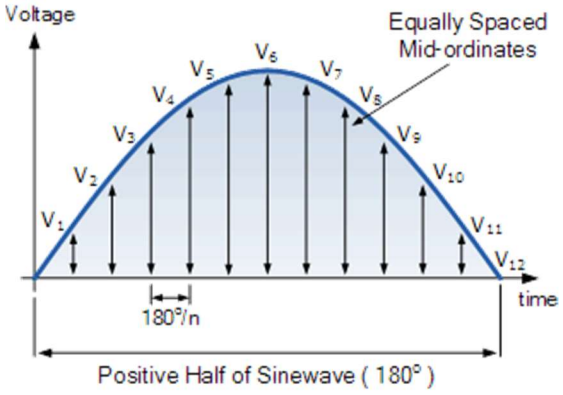

Figure 5-10-6-F2: Algorithm of Vrms calculation

The formula of RMS voltage is square root of total sum of squared function of instantaneous values divided by total number of instantaneous.

$$Vrms = \sqrt{\frac{{V_1}^2 + {V_2}^2 + \ldots + {V_n}^2}{n}}$$

5-10-7 Average Voltage

The average voltage in of a periodic waveform is defined that is "the quotient of the area under the waveform with respect to time". The method to calculate average voltage is similar to finding Vrms but the difference is the instantaneous value is not in squared function. The symbol of average voltage is defined as Vavg.

$$Vavg = \sqrt{\frac{V_1 + V_2 + \ldots + V_n}{n}}$$

5-10-8 Rise Time & Fall Time

Rise time indicates that the time of a signal change from specific low voltage level to specific high voltage level. The voltage level is usually expressed in percentage. The output step height for the rise time is from 10% to 90% of voltage level.

Fall time is the opposite of the rise time that is the time of a signal change from specific high voltage level to specify low voltage level. The output step height for the fall time is from 90% to 10% of voltage level. Figure 5-10-8-F1 shows the explanation of rise time and fall time with ratio between 10% and 90%.

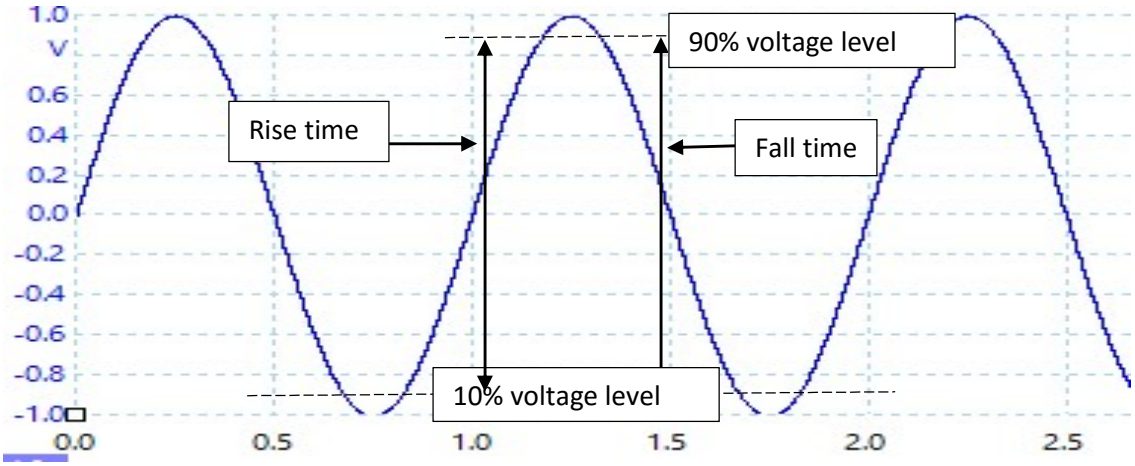

Figure 5-10-8-F1: Explanation of rise time and fall time with ratio between 10% and 90%.

The total counts of data in the rise time and the time of each sampled data is required to calculate the rise time of signal.

First, peak to peak voltage is divided by 10 to get 10% of Vpk-pk. Then, specified 10% voltage level is calculated by minimum voltage adding with 10% of Vpk-pk. The specified 90% voltage level is calculated by using maximum voltage minus the 10% of Vpk-pk.

After that, the rise time data will be counted from the specific low voltage level until the specified high voltage level. Then, the fall time data will be counted when the voltage values are in between the specified high and low voltage level. Lastly, the total counts of rise and fall data are multipled with time for each sampled unit data. Figure 5-10-8-F2 shows the flow chart of calculation of rise and fall time.

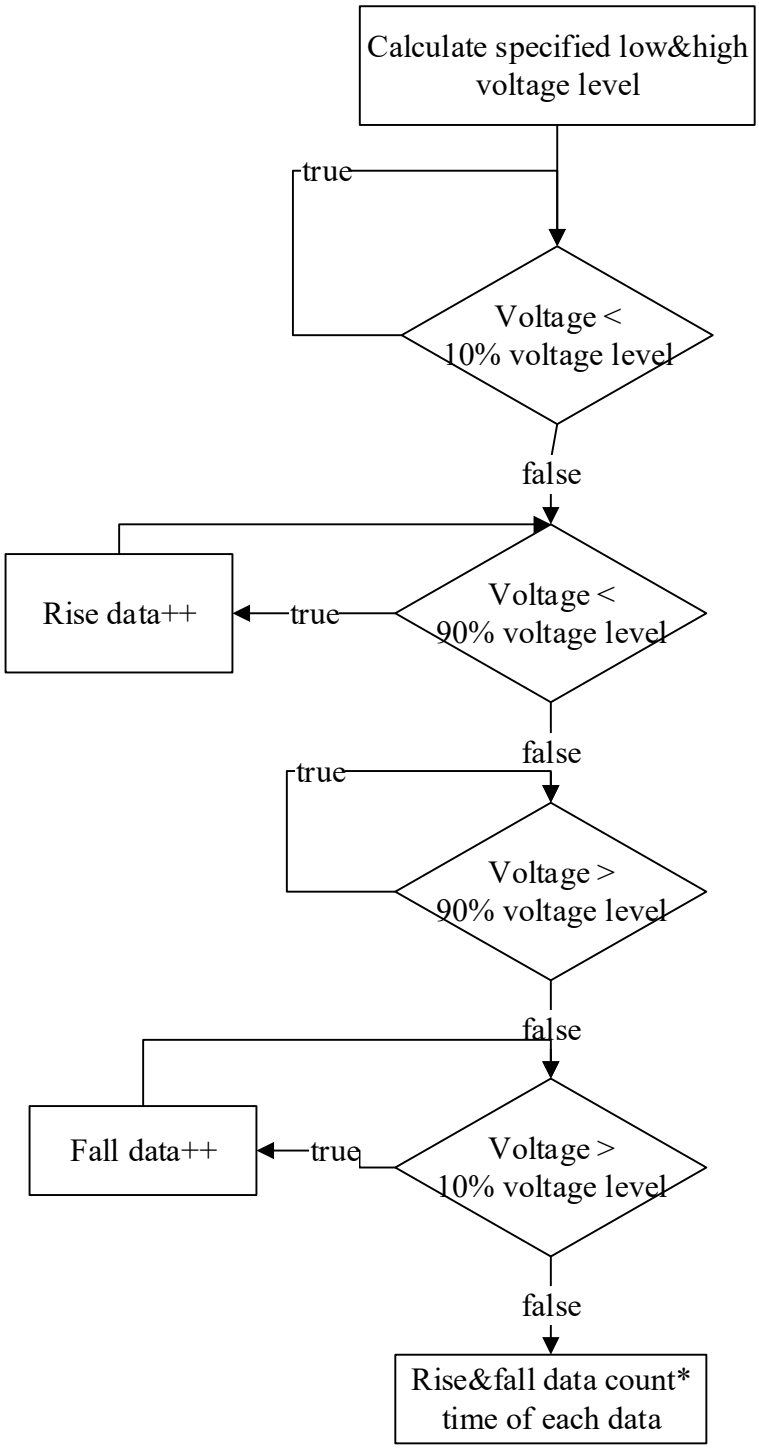

Figure 5-10-8-F2: The flow chart of calculation of rise and fall time.

5-10-9 Display measurement results

Due to limitation of hardware specifications causing the results are not so accurate, the results of selected measurement functions is calculated by taking the average of total values of selected measurements in *n* times. Then the results will be displayed in GUI. Besides, users have enough time to verify the measurement result without keep watching to the changing of measurement result.

5-11 Trigger Function

The trigger function of oscilloscope is used to stabilize the repetitive waveform. It synchronize the horizontal sweep of the oscilloscope to the threshold value which set by user. The waveform will be display repeatedly with similar looks. This is very important and useful function to achieve clearly displaying signal.

The threshold value is limited to specified low and high voltage level which same with the rise and fall time. The threshold value will be reset when range of specified voltage level is out.

A simple trigger function is developed that can trigger the rising edge and falling edge of signal. The algorithm of calculate total counts of rise and fall data is reused and position of threshold value is recorded based on selected edge in periodic cycle of waveform for display purpose later. Figure 5-11-F1 shows the algorithm of trigger function in flow chart.

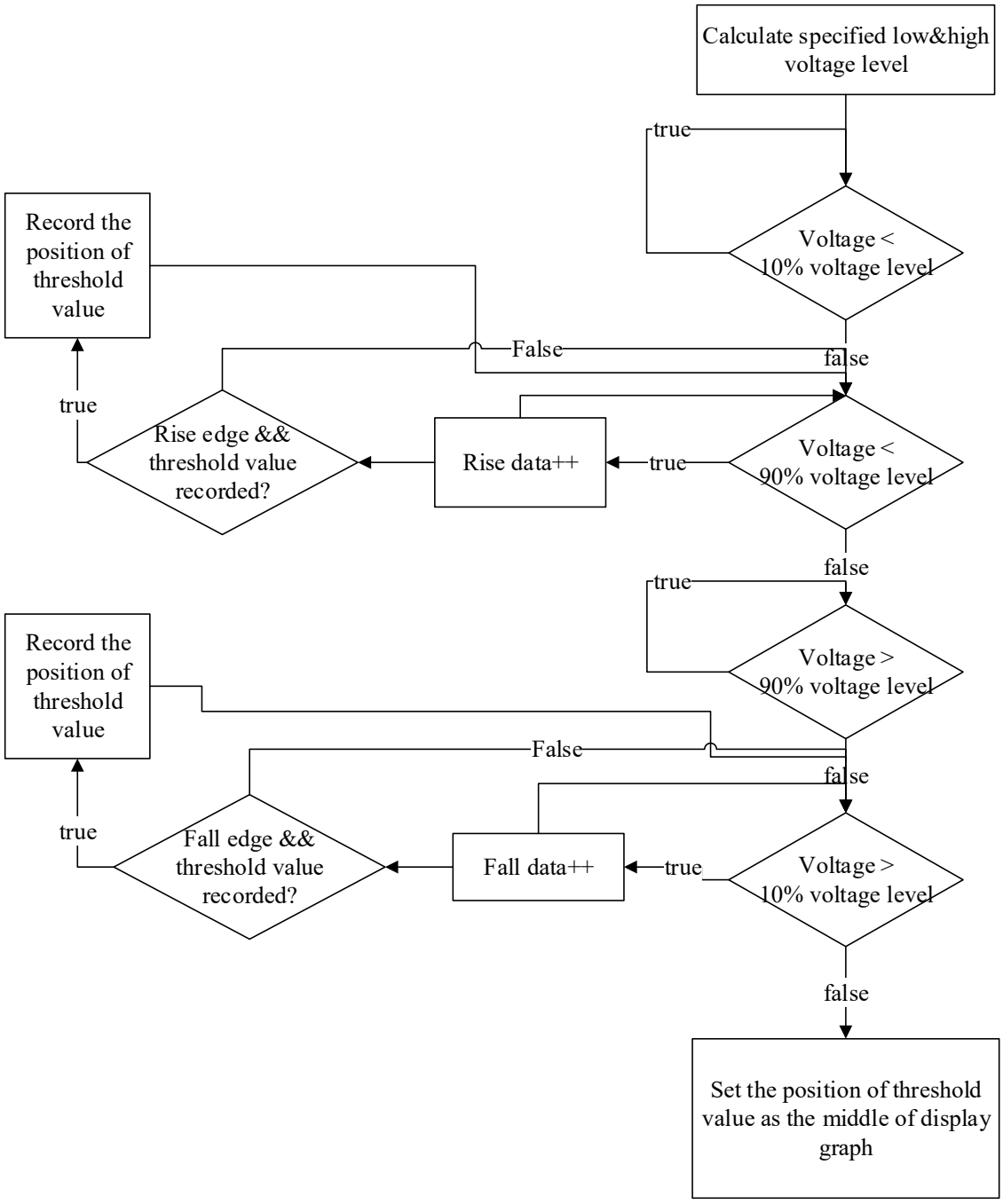

Figure 5-11-F1: Algorithm of trigger function in flow chart

After found the position of the threshold value on selected edge, the position is set as the middle point of the graph and display in graph.

5-12 Auto Scales

Auto scale is a useful function that change the time base to appropriate time base based on the frequency of periodic waveform. The graph can be displayed clearly by change to appropriate time bases. A table is formed that the frequency value should be suitable for which time base. Table 5-12-T1 shows the appropriate time base according to frequency.

Time base	Frequency
0.5ms	>= 700
1.0ms	< 700
2.0ms	< 400
5.0ms	< 200
10.0ms	< 70
20.0ms	< 40

Table 5-12-T1: Appropriate time base according to frequency

5-13 Persistence mode

Persistence mode is another useful function to troubleshoot the occasional glitches or some other problems. In most of the cases, a stable trigger point is required to display the graph in persistence mode. If not, there will be a jumble of traces.

The algorithm of persistence mode is doing screenshot the graph parts after plot the latest graph then set the screenshot of the graph as the background image. This action is repeatedly completed and become as persistence mode. QPixmap class that is the representation of off-screen image for painting purpose. Therefore, QPixmap is used to handle image data in QtCreator. Figure 5-13-F1 shows the algorithm of implementing persistence mode.

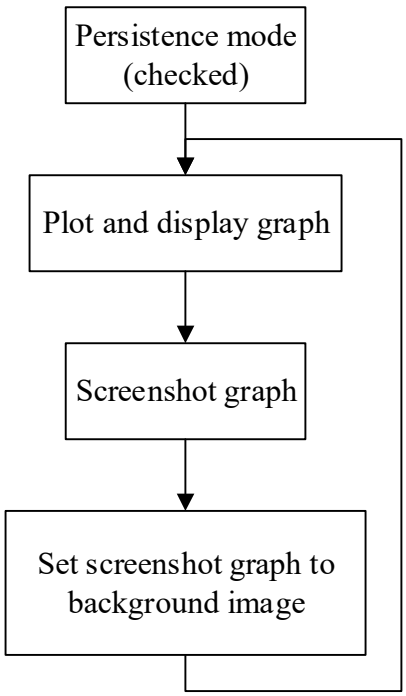

Figure 5-13-F1: Algorithm of implementing persistence mode

5-14 Storage

In this oscilloscope application, user can choose which storage to access files that required for functions respectively. The storage that define in the application is internal storage, eMMC flash and thumb drive of user that connected through USB Host 2.0. Users can select the storage in the Settings page.

5-15 Screenshot function

Screenshot function is implemented to oscilloscope application. The purpose for implementing this function is to record the graph in the form of graphics file. The whole screen of application is captured and saved into file system with selected image format.

The options for image format in this oscilloscope application is Joint Photographic Expert Group (JPEG) and Portable Network Graphic (PNG). JPEG format is an image format that

have standardized compression mechanism. It has the features that can compress digital image in either full color (24bit) of gray level scales. PNG format is lossless compression format. Therefore, it is commonly used in web pages.

QPixmap class is used to grab pixel data in whole of the screen. The name of image file is created by a sequence of the storage location, current count of screenshots and image format. Then, the screenshot image is saved into selected storage locations. Figure 5-15-F1 shows code of screenshot function.

```
void MainWindow::on_screenshotButton_clicked()
{
    QPixmap p;
    p = QPixmap::grabWindow(QApplication::desktop()->winId());

    QString filename = QString("screenshot_%2.").arg(screenShotCount);
    filename.append(imageType);
    QString name = fileDir;
    name.append(filename);
    QString format = imageType;
    p.save(name,format.toAscii());
    screenShotCount++;
}
```

Figure 5-15-F1: Code of screenshot function.

5-16 Display current Date & Time

The current date and time is shown in the top left of GUI. The purpose of this function is used to display information of date and time. The screenshot images cannot be identified that when the image is captured if this function is not involved.

The information of current date and time are obtained from system time of operating system. A timer is started and emits timeout signal in every 0.1 second to update the current time. Figure 5-16-F1 shows code to display current date and time.

```
systemDateTimer = new QTimer(this);
connect(systemDateTimer,SIGNAL(timeout()),this,SLOT(showDataTime()));
systemDateTimer->start(100);

void MainWindow::showDataTime()
{
    // Display current Date & Time
    ui->dataTimelabel->setText(QDateTime::currentDateTime().toString());
}
```

Figure 5-16-F1: Code to display current date and time.

5-17 Data Logging

This is an important function to record the data for some process or calculation purpose. The algorithm of this function is same as writing data in a file. . Besides, there are two different files for data logging according to two voltage input channel. This function is invoked based on the selected channels.

There are two types of file format can be selected which is comma-separated value (CSV) and text (TXT) format. CSV file store the data in a table structured format to easy for data processing. CSV format is supported by Microsoft Excel. Text file is the electronic text in structure of line sequence. There are many users still using the TXT format. Figure 5-17-F1 shows code of data logging function that write ADC data into file respectively.

```
QString filename = QString("%1data_Ch1.").arg(fileDir);
filename.append(dataFileType);
QFile f(filename);
f.open(QIODevice::WriteOnly | QIODevice::Text);
QTextStream output(&f);
output<<"Samples\tVoltage(V)\n";
for(int i=0;i<DATA_SIZE;i++)
    output<<x1[i]<<"\t"<<v1[i]<<endl;
f.close();
```

Figure 5-17-F1: Code of data logging function that write ADC data into file respectively.

5-18 Save & Load Graph

This function is used to compare between a saved graph and current graph. The data of graph are written into a file according the selected channels when the save graph button is clicked. When the load graph button is clicked, the data are read from the file and plotted with another graph in green color line. Figure 5-18-F1 shows screenshot of GUI after clicked load graph button. The green line is represented as the loaded graph.

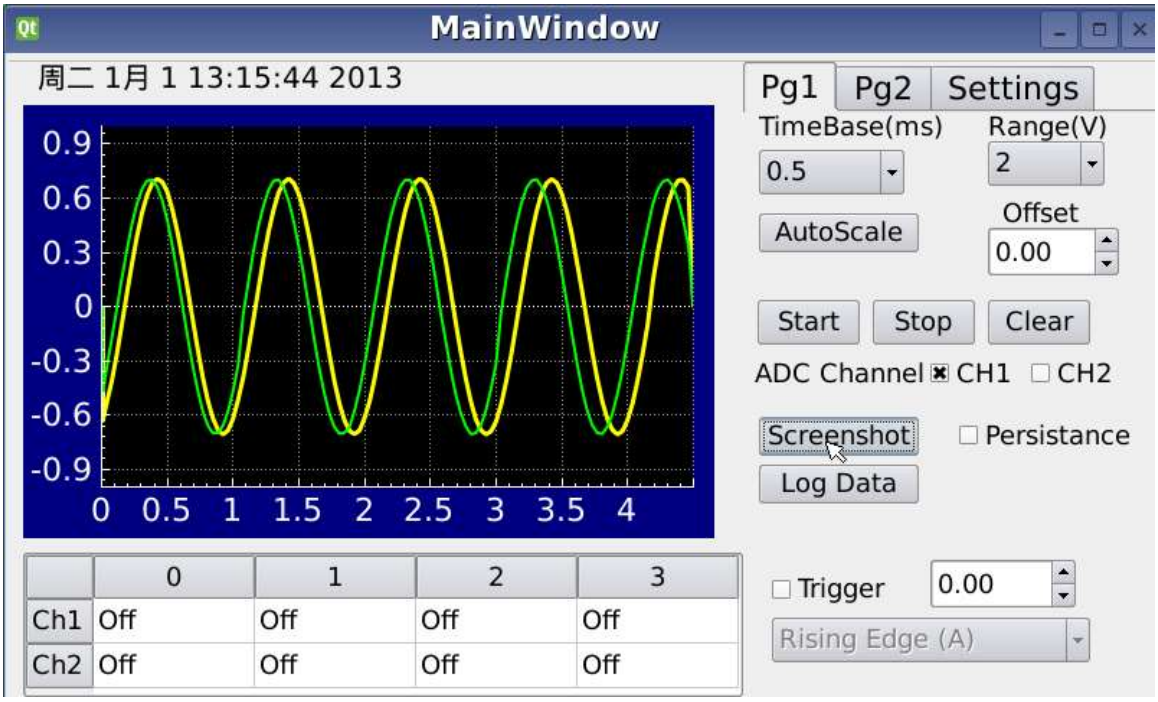

Figure 5-18-F1: Screenshot of GUI after clicked load graph button

5-19 Vision

The vision function is implemented to set the background color of graph. There are two visions that are day vision and night vision is designed. The purpose of this function is to let the user have more clearly vision when monitor the graph. Figure 5-19-F1 shows the screenshot of GUI in day vision while Figure 5-19-F2 shows the screenshot of GUI in night vision.

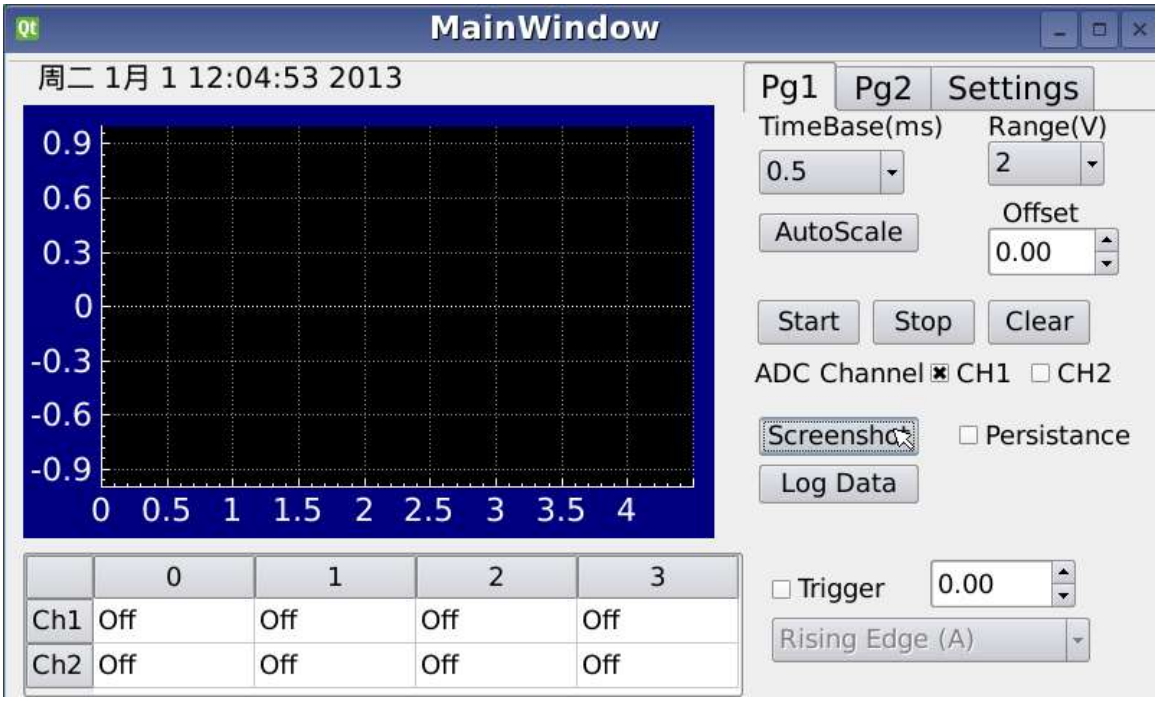

Figure 5-19-F1: Screenshot of GUI in day vision

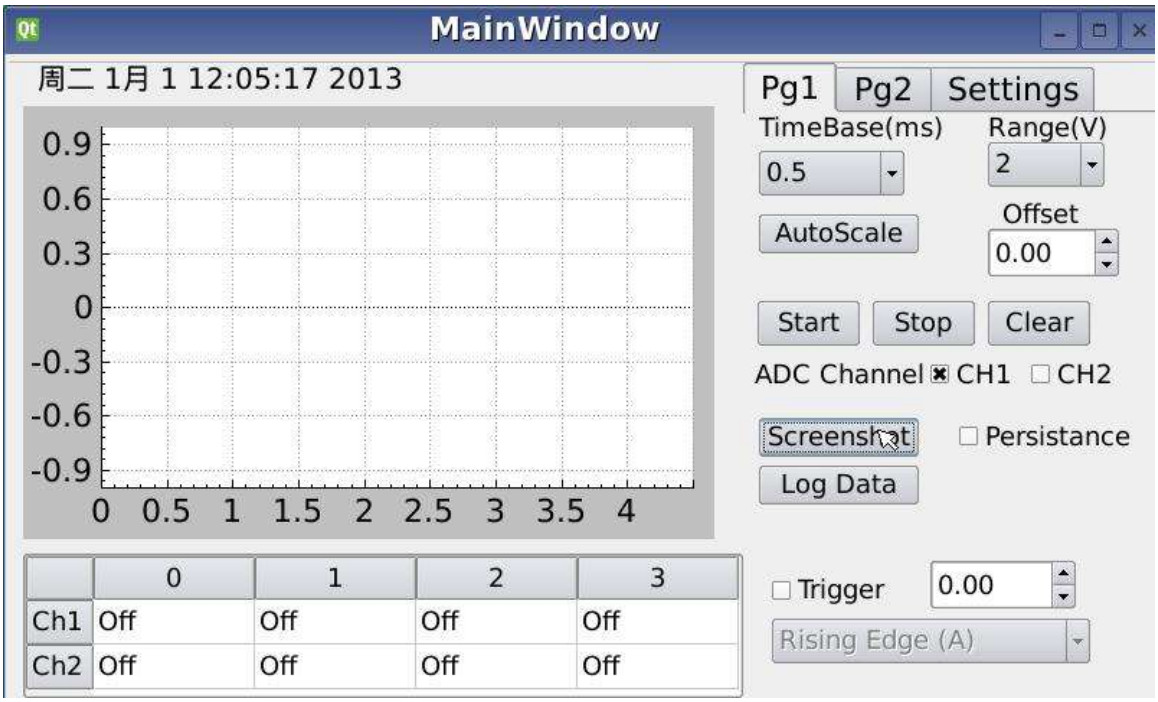

Figure 5-19-F2: Screenshot of GUI in night vision.

5-20 Backlight

The backlight of LCD display is controlled to increase or decrease the brightness of display. The brightness of LCD display should be lower when the environment is dark. In opposite, the LCD display will show brighter to show clearly when the environment is bright. In order to control the backlight, the backlight LCD driver is needed to be accessed. The backlight intensity value is written into backlight device file ("/dev/backlight-1wire"). The ranges of backlight intensity value are from 0 to 127 while 0 is the darkest and 127 brightest.

5-21 Save & Load Settings

All the selections in oscilloscope application such as the image formats, data file formats, selected ADC channels, storage locations, time bases, voltage ranges, offset values and others are saved into a file when save signal is emitted.

The settings file can be loaded in next time operation so the last configuration settings can be recovered without users configure manually again.

CHAPTER 6 PC-BASED OSCILLOSCOPE APPLICATION

6-1 PC-Based Application

This application is designed in window form application to receive the ADC data from designed embedded oscilloscope and display it in graph and the result of measurement calculation. This application is developed in C# language and used Microsoft Visual Studio as the Integrate Development Environment (IDE). Besides, remote control feature between embedded oscilloscope and this application is implemented to let users can control the board wirelessly and convenient. Asynchronous programming is implemented into this window form application to keep the GUI more responsive.

6-2 Microsoft Visual Studio & C#

For software part, Microsoft Visual Studio is chosen as the IDE to develop the PC-based application. Microsoft Visual Studio have many tools and services to support projects having any size and complexity. Besides, Microsoft Visual Studio supports many programming languages such C++, C#, VB, Python, Node.js, HTML and JavaScript. Microsoft Visual Studio consists of some features such as advanced debugging, profiling and manual testing.

C# is used as the programming language in development of the application. C# is a multi-paradigm programming language that having several principles of simple, modern, strong type checking, functional, general purpose, object-oriented and component-oriented. (Ecma International, 2006). Figure 6-2-F1 shows the screenshot of Microsoft Visual Studio.

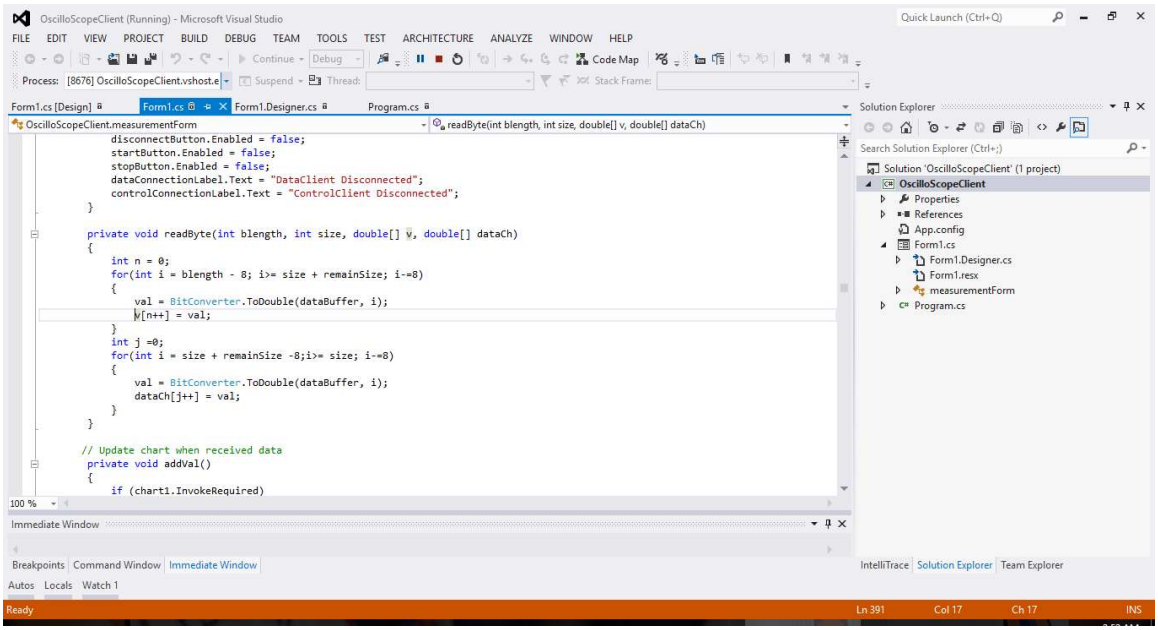

Figure 6-2-F1: Screenshot of Microsoft Visual Studio

6-3 Specification

The features of PC-based oscilloscope application are listed as below:

- Able to receive the ADC data from embedded oscilloscope application
- Plot data into graph and display
- Receive calculated measurement results
- Remote control features between both application

6-4 Software Design

The application must have to connect to the embedded oscilloscope application through socket before start any function. After connected, the application will be ready to receive the ADC data from the embedded oscilloscope application. The another socket is created and connected to send the command to invoke the GUI event and ready to receive the command that contain changed settings in the embedded oscilloscope application. In addition, the users can disconnect the socket from the embedded oscilloscope application.

6-4-1 Flow Chart

The flow chart of whole application is designed and shown in Figure 6-4-1-F1.

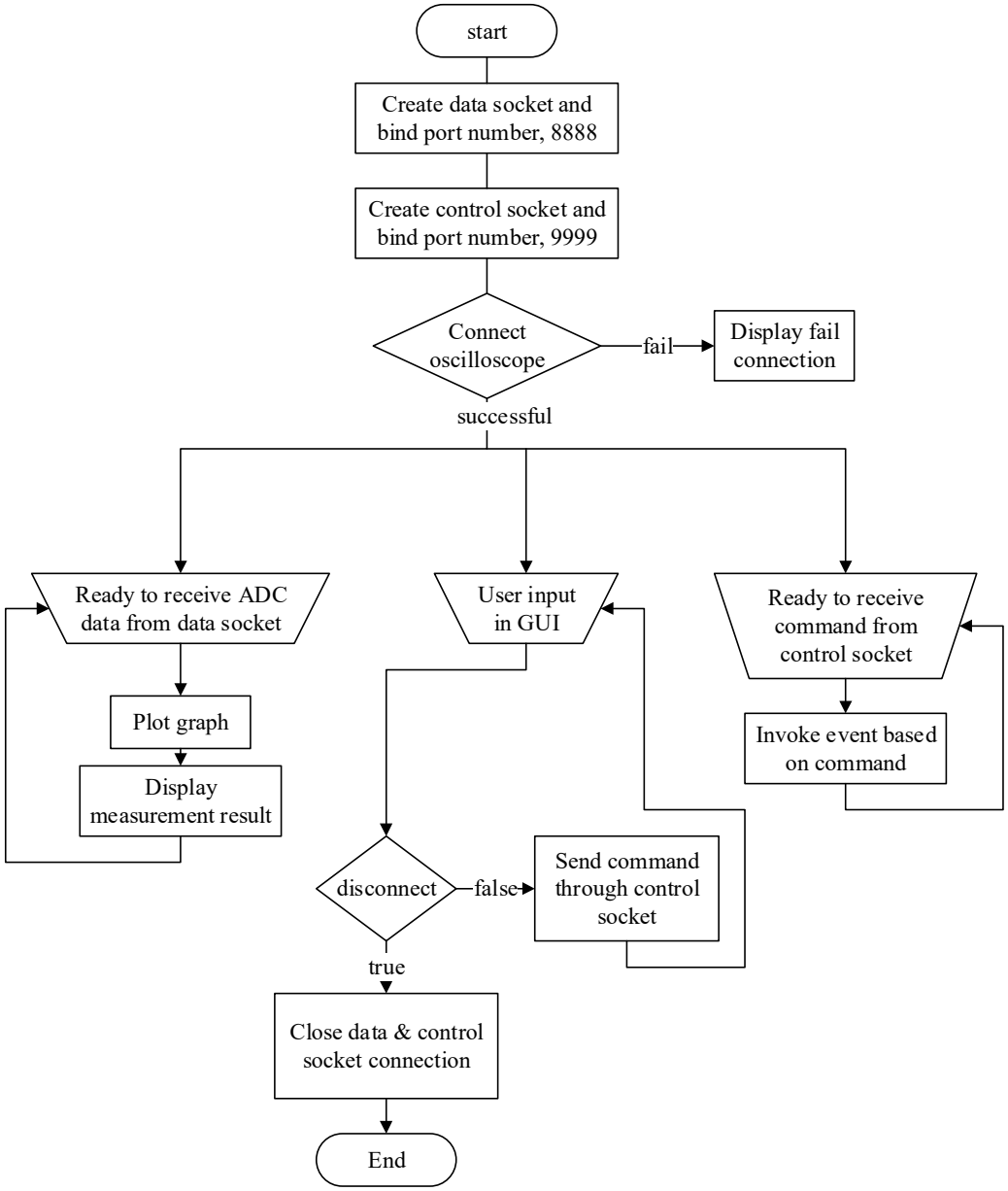

Figure 6-4-1-F1: Flow chart of PC-based oscilloscope application

6-4-2 Graphical User Interface (GUI)

The GUI of the application is designed as same as the embedded oscilloscope application. All the button and selection are inserted to application GUI. Figure 6-4-2-F1 shows screenshot of application GUI.

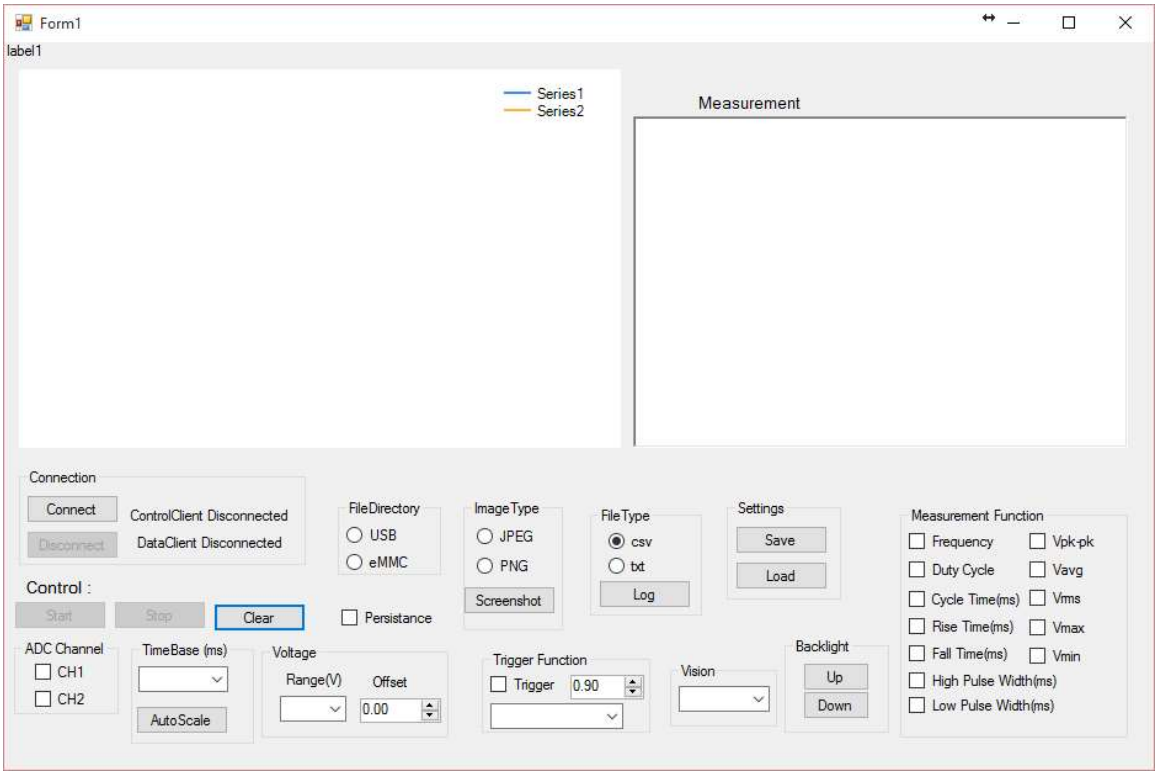

Figure 6-4-2-F1: Screenshot of application GUI.

6-4-3 Socket

The algorithm used in this application to implement the socket programming is same the embedded oscilloscope application. Figure 5-6-1-F1 show the flow chart of the socket programming. The difference in between this two application is that this PC-based oscilloscope application is using the asynchronous programming. The data and control port are set to 9999 and 8888 which are same with the embedded oscilloscope application to enable the connection with each other. After the connection is established, the data connection is ready to read the incoming data and decode it then display in graph. The control connection is ready to receive the control command to change GUI settings or send

control command respectively when GUI event occurs. Figure 6-4-3-F1 shows the coding of create the socket and bind socket to defined port number in C#.

```
private void connectButton_Click(object sender, EventArgs e)
{
    ////////////////////////////////////////////////////////////////////////////
    //Create Data socket
    Socket dataSock = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);
    //connect server host address
    IPEndPoint dataIep = new IPEndPoint(IPAddress.Parse(ipAddress), dataPort);
    //Bind socket to port and send connection request to server and wait server respond
    dataSock.BeginConnect(dataIep, new AsyncCallback(DataConnected), dataSock);

    ////////////////////////////////////////////////////////////////////////////
    //Create Control socket
    Socket controlSock = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);
    //connect server host address
    IPEndPoint controlIep = new IPEndPoint(IPAddress.Parse(ipAddress), controlPort);
    //Bind socket to port and send connection request to server and wait server respond
    controlSock.BeginConnect(controlIep, new AsyncCallback(ControlConnected), controlSock);
}
```

Figure 6-4-3-F1: Code of create the socket and bind socket to defined port number in C#.

The connected client can be disconnected by close the connection. This event can be invoked by clicked disconnect button in application GUI. Figure 6-4-3-F2 shows the coding of disconnect socket connection event.

```
private void disconnectButton_Click(object sender, EventArgs e)
{
    dataClient.Close();
    controlClient.Close();
    connectButton.Enabled = true;
    disconnectButton.Enabled = false;
    startButton.Enabled = false;
    stopButton.Enabled = false;
    dataConnectionLabel.Text = "DataClient Disconnected";
    controlConnectionLabel.Text = "ControlClient Disconnected";
}
```

Figure 6-4-3-F2: Coding of disconnect socket connection event.

6-4-4 Read the incoming ADC data

The ADC data received from embedded oscilloscope application are packed in byte array. The received data contains total data displaying in the graph of embedded oscilloscope plus some measurement results. The length of each data is 8bit long. Therefore, the length of the byte array received is total amount of data times with eight. If the length of byte array is twice of normal length, it means the incoming data contain two channels voltage data in the byte array. After decoded the incoming data, the graph is refreshed by adding the decoded data. The received measurement results are same with the embedded oscilloscope application. Figure 6-4-4-F1 shows the flow chart diagram of receiving incoming data.

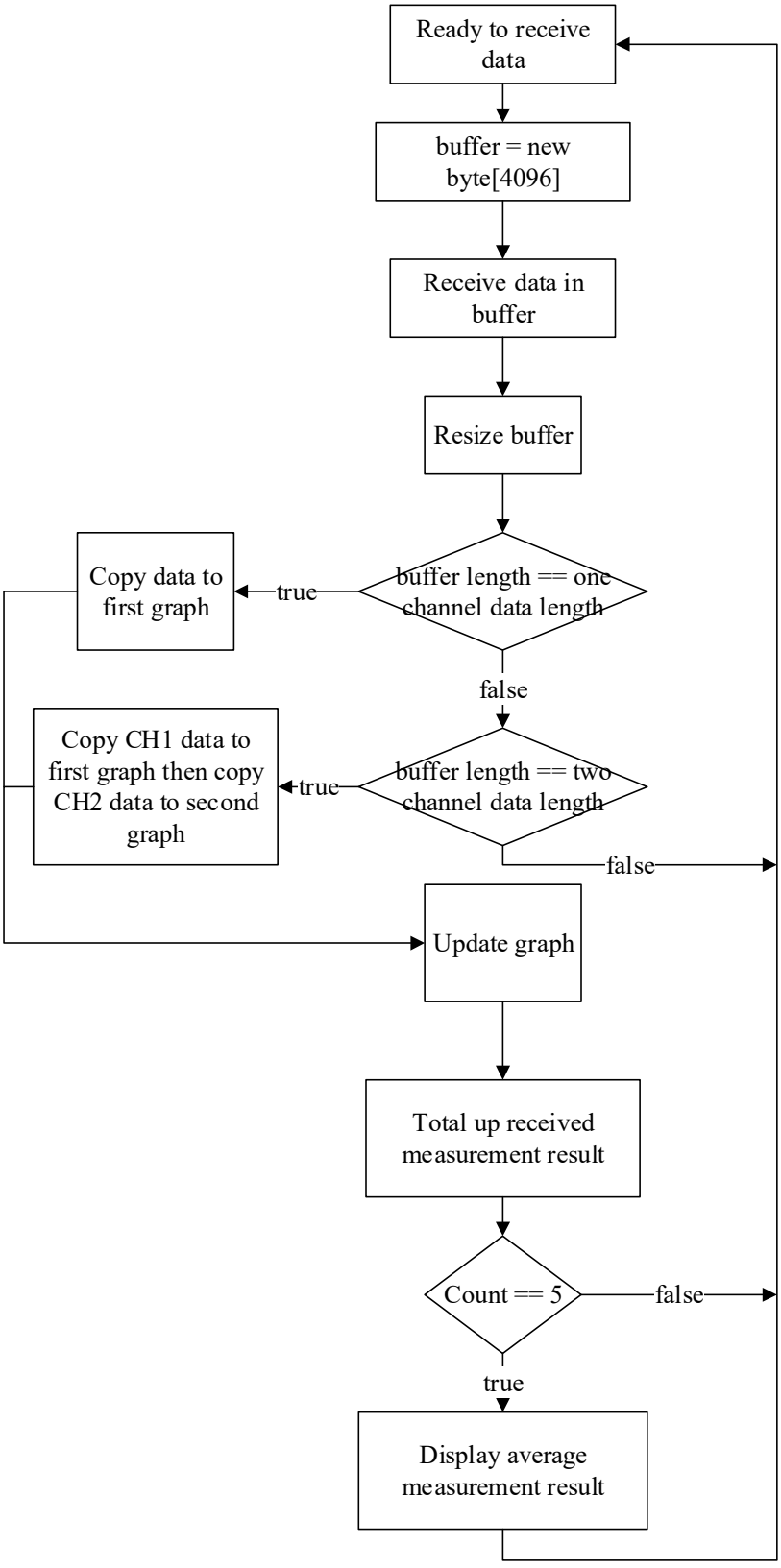

Figure 6-4-4-F1: The flow chart diagram of receiving incoming data.

6-4-5 Display Graph

Microsoft Chart control is used to display the graph in this application. The Microsoft chart control is easy to be used and have many defined useful functions. The array that contain the ADC data are bind to graph for displaying purpose. Figure 6-4-5-F1 shows the coding screenshot of bind array data to graph.

```
// Update chart when received data
 private void addVal(){
     if (chart1.InvokeRequired){
         addValCallBack d = new addValCallBack(addVal);
         this.Invoke(d);
     }
     else{
         if(ch1Box.Checked && !ch2Box.Checked)
             chart1.Series["Series1"].Points.DataBindY(v1);
         else if (!ch1Box.Checked && ch2Box.Checked){
             chart1.Series["Series2"].Points.DataBindY(v1);
         }
         else if (ch1Box.Checked && ch2Box.Checked){
             chart1.Series["Series1"].Points.DataBindY(v1);
             chart1.Series["Series2"].Points.DataBindY(v2);
         }
     }
 }
```

Figure 6-4-5-F1: Coding screenshot of bind array data to graph.

6-4-6 Remote Control

As mentioned above, a control socket is created and established the connection to same IP address of client who connected to data socket but bind with different port number, 8888. The application will send the commands which included description of event occurred to embedded oscilloscope. Besides, the application is ready to receive the command which included the descriptions of changed settings from embedded oscilloscope. Figure 6-4-6-F1 shows the flow chart of sending command when event occurred. Figure 6-4-6-F2 shows s\the flow chart of receiving command from embedded oscilloscope.

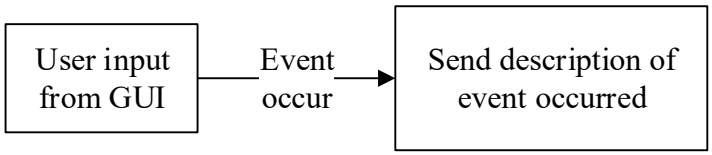

Figure 6-4-6-F1: Flow chart of sending command when event occurred

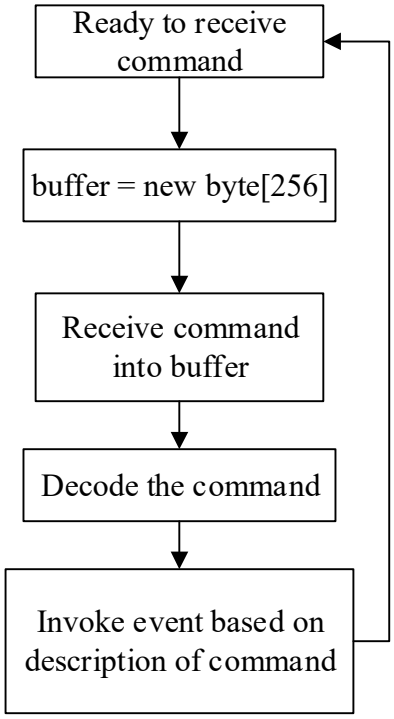

Figure 6-4-6-F2: Flow chart of receiving command from embedded oscilloscope.

CHAPTER 7 TESTING AND RESULTS

7-1 Testing & Results

In order to do the testing to embedded oscilloscope, a function generator is required to generate voltage signal to ADC input. An oscilloscope product is act as reference to measurement results of embedded oscilloscope.

7-2 Tools

7-3-1 Function Generator

Agilent U2761A USB Modular Function Generator is used as function generator to produce voltage signal to ADC input of Tiny4412. It is able to emit 20MHz of sine wave, square wave, ramp wave, DC wave and arbitrary wave. Besides, it is compatible with Microsoft Windows operating system. Figure 7-3-1 shows the Agilent U2761A USB Modular Function Generator.

Figure 7-3-1-F1: Agilent U2761A USB Modular Function Generator.

7-3-2 PicoScope2104

PicoScope2104 is a handheld oscilloscope that used to verify the validity calculated result of portable oscilloscope prototype. The bandwidth of this handheld oscilloscopes is up to 25MHz bandwidth while the maximum sample rate is 100Ms/s. Besides, it contains 24000 sample buffer memory. This product is connected and powered by USB. Figure 7-3-2 shows the PicoScope2104

Figure 7-3-2-F1: PicoScope2104

7-3 Results

In this chapter, PicoScope 6 oscilloscope software is used to obtain the reference result to compare with actual result that obtained from the embedded application oscilloscope. Sine wave, square wave, triangle wave and ramp up wave is the waveform signal used for the testing application. Figure 7-3-F1 shows screenshot of PicoScope 6 oscilloscope software.

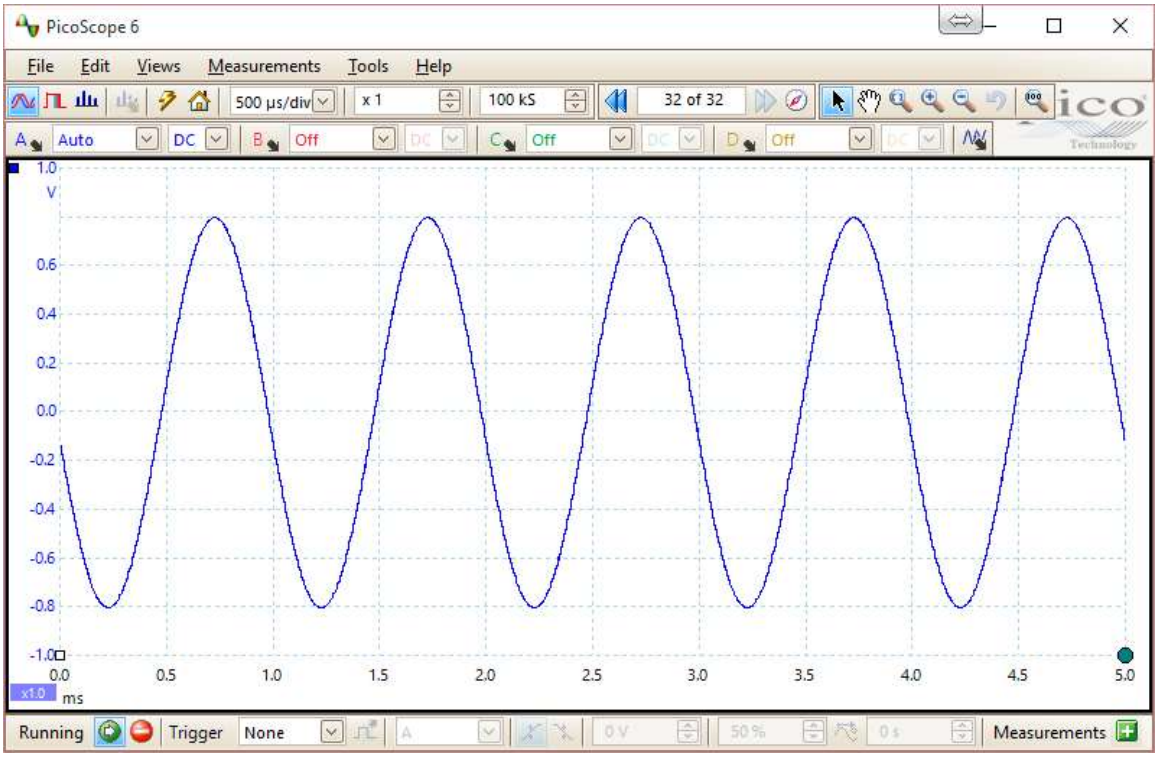

Figure 7-3-F1: Screenshot of PicoScope 6 oscilloscope software.

7-3-1 Sine Wave

A signal in sine waveform is generated from function generator. Peak voltage of the sine wave is set to 0.9V. The result of measurements functions is test with the lowest frequency, 1 Hz until the highest frequency, 2 kHz by PicoScope2104 and designed oscilloscope application. In Appendix A-1, Table A-1-T1 shows the results of sine wave by PicoScope2104 and Table A-1-T2 shows results of sine wave by designed oscilloscope application.

7-3-2 Square Wave

Square wave signal is used in this testing plan. . Peak voltage of the square wave is set to 0.2V and duty cycle is set to 80%. The result of measurements functions is test with the lowest frequency, 1 Hz until the highest frequency, 2 kHz by PicoScope2104 and designed oscilloscope application. In Appendix A-2, Table A-2-T1 shows the results square wave of

by PicoScope2104 and Table A-2-T2 shows results of square wave by designed oscilloscope application.

7-3-3 Triangle Wave

Triangle wave signal is used in this testing plan. . Peak voltage of the triangle wave is set to 0.7V. The result of measurements functions is test with the lowest frequency, 1 Hz until the highest frequency, 2 kHz by PicoScope2104 and designed oscilloscope application. In Appendix A-3, Table A-3-T1 shows the results of triangle wave by PicoScope2104 and Table A-3-T2 shows results of triangle wave by designed oscilloscope application.

7-3-4 Ramp Up Wave

Ramp up wave signal is used in this testing plan. Peak voltage of the ramp up wave is set to 0.45V. The result of measurements functions is test with the lowest frequency, 1 Hz until the highest frequency, 2 kHz by PicoScope2104 and designed oscilloscope application. In Appendix A-4, Table A-4-T1 shows the results of ramp up wave by PicoScope2104 and Table A-4-T2 shows results of ramp up wave by designed oscilloscope application.

CHAPTER 8 CONCLUSION

8-1 Summary

In this project, a portable oscilloscope prototype is designed and built. The sampling frequency of the oscilloscope prototype less than 2 kHz. The input voltage ranges is between 0 to 1.8V and the resolution of sampled ADC data is in 12bit. The main objective of this project has been achieved that an embedded oscilloscope application is successfully designed and developed and run in the Tiny4412 board. The application is able to sample ADC data and plot it into graph. The measurement result is calculated after sampled the ADC data.

Besides, the ADC data can be sent to other devices through socket in wireless connection. A PC-based oscilloscope application is designed and developed to receive the ADC data. Furthermore, remote control features is implemented between embedded oscilloscope and the PC-based application. Therefore, the second main objective that sent ADC data through socket is successfully achieved.

8-2 Encountered Problems

There are some problems that the time when sampled data is not accurate to the theoretical consumed time and some minor difference of ADC data when sampling ADC data every time. This is due to the limitations of hardware. The programming algorithm is figured out to solve the constraints of hardware. Although the results are not so accurate to the theoretical results but almost close to it. In addition, some useful oscilloscope features such as trigger function and persistence mode are implemented. Most of the oscilloscope are using hardware for triggering. By using hardware triggering, the results will be more accurate but Tiny4412 have no this hardware module. Therefore, the way to implement trigger function is software algorithm. The persistence modes is useful to troubleshoot the glitches of signal by using the trigger function together.

CHAPTER 8 CONCLUSION

8-3 Contributions

The contributions in this project is that a portable oscilloscope is successfully designed and developed that the prototype have the basic features of an oscilloscope. Besides, the data transmission through wireless connection is successfully implemented and get better performance in data transmission. This achievement solves the cable constraint and low rates of data transmission of Bluetooth and RS232 cable. Remote control feature between the prototype and PC-based application is one of the contributions in this project that users can remotely control the prototype wirelessly.

8-4 Improvements

There are some future works that can do some improvements to this project. By implementing some suitable hardware such as FPGA, the more accurate signal waveform and time of sampled data can be obtained. The results from trigger function also can be improved by using hardware triggering. The optimization of the application of portable oscilloscope is required to increase performance of oscilloscope system by split the work to idle processor core.

REFERENCES

Emilia.B, Chirstian.S, Bernhard.V (2002), 'Oscilloscope Data-Acquisition over IEEE-488 without Overhead', 30 December, Available from: http://fahrzeugtechnik.fh-joanneum.at/publikationen/Oscilloscope_Data-Acquisition_over_IEEE-488_Dez2002_BRATSCHITSCH_SCHERMAIER_VONIER.pdf

H.M.D.B. Seneviratne, K.N. Abhayasinghe (2013), 'Bluetooth Embedded Portable Oscilloscope', *SAITM Research Symposium on Engineering Advancements 2013,* p.103-p.106. Available from: http://www.saitm.edu.lk/fac_of_eng/RSEA/SAITM_RSEA_2013/imagenesweb/24.pdf

TestEquity (2010), 'TCP/IPv4 and Ethernet 10BASE-T/100BASE-TX Debugging with the MSO/DPO4000B Series Oscilloscopes', Available from: http://www.testequity.com/documents/pdf/ethernet-debugging.pdf

Tiny4412 (n.d.), Available from: http://www.armdevs.com/image/Tiny4412/Tiny4412.jpg (Accessed from 30 March 2015)

ePipe (n.d.), *TCP/IP Networking Primer* Available from: **http://www.ml-ip.com/html/documentation/vpn-ug-key-concepts-1.html** (Accessed 31 March 2015)

Edison.H (2014), Networking and Socket programming tutorial in C, 21 August 2014 Available from http://www.codeproject.com/Articles/586000/Networking-and-Socket-programming-tutorial-in-C (Accessed from 30 March 2015)

Ecma International (June 2006), C# Language Specification Available from: http://www.ecma-international.org/publications/files/ECMA-ST/Ecma-334.pdf

Average Voltage Tutorial, AC circuits, Available from: http://www.electronics-tutorials.ws/accircuits/average-voltage.html (Accessed from 27 February 2016)

REFERENCES

RMS Voltage Tutorial, AC circuits, Available from: <http://www.electronics-tutorials.ws/accircuits/rms-voltage.html> (Accessed from 27 February 2016)

Maya Posch, "How To Really,Truly Use QThreads; The Full Explanation", November 2011, Available from <https://mayaposch.wordpress.com/2011/11/01/how-to-really-truly-use-qthreads-the-full-explanation/> (Accessed from 30 March 2016)

APPENDIX A

A-1 Sine Wave

Signal : Sine wave
Peak voltage : 0.9V

Measurement	Frequency of signal							
	10Hz	**20Hz**	**50Hz**	**100Hz**	**500Hz**	**1000Hz**	**1500Hz**	**2000Hz**
Frequency (Hz)	10	20	49.99	100	500.2	1000	1500	2000
Cycle Time (ms)	99.98	50.01	20	9.998	2.001	0.9999	0.6667	0.5
Duty Cycle (%)	50.06	50	50.08	50.2	50.11	50.13	50.1	50.13
Rise Time (ms)	29.86	14.89	5.955	2.975	0.5965	0.2984	0.1989	0.1495
Fall Time (ms)	29.82	14.94	5.965	2.99	0.5987	0.2991	0.1998	0.1497
Vmax (V)	0.9075	0.9075	0.9075	0.9075	0.9075	0.9075	0.9075	0.9075
Vmin (V)	-0.9077	-0.9077	-0.9077	-0.9077	-0.9077	-0.9077	-0.9077	-0.9077
Vpk-pk (V)	1.815	1.815	1.815	1.815	1.815	1.815	1.815	1.815
Vavg (V)	0.5781	0.5781	0.5781	0.5781	0.5781	0.5781	0.5781	0.5781
Vrms (V)	0.6385	0.6385	0.6302	0.6386	0.6331	0.6381	0.6365	0.6374

Table A-1-T1: Results of sine wave by PicoScope 2104

Measurement	Frequency of signal							
	10Hz	**20Hz**	**50Hz**	**100Hz**	**500Hz**	**1000Hz**	**1500Hz**	**2000Hz**
Frequency (Hz)	13.388	19.912	48.241	100.614	489.529	943.623	1446.6	1869.608
Cycle Time (ms)	74.694	50.222	20.729	9.939	2.043	1.06	0.691	0.535
Duty Cycle (%)	78.851	44.103	49.901	51.429	50	50	47.059	53.846
Rise Time (ms)	32.502	14.604	6.075	2.961	0.613	0.326	0.203	0.165
Fall Time (ms)	30.876	16.265	5.919	3.03	0.619	0.327	0.245	0.163
Vmax (V)	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9
Vmin (V)	-0.9	-0.9	-0.9	-0.9	-0.9	-0.9	-0.9	-0.9
Vpk-pk (V)	1.8	1.8	1.8	1.8	1.8	1.8	1.8	1.8
Vavg (V)	0.548	0.589	0.599	0.576	0.572	0.556	0.554	0.622
Vrms (V)	0.616	0.652	0.657	0.64	0.64	0.619	0.629	0.664

Table A-1-T2: Results of sine wave by designed oscilloscope application

A-2 Square Wave

Signal : Square wave Duty Cycle: 80%
Peak voltage : 0.2V

Measurement	Frequency of signal							
	10Hz	**20Hz**	**50Hz**	**100Hz**	**500Hz**	**1000Hz**	**1500Hz**	**2000Hz**
Frequency (Hz)	10	20	50	99.99	500	1000	1500	2000
Cycle Time (ms)	99.98	49.99	20	10	2	1	0.6667	0.5
Duty Cycle (%)	79.19	79.18	79.3	79.84	79.26	79.49	79.62	79.9
Rise Time (ms)	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
Fall Time (ms)	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
Vmax (V)	0.2058	0.2058	0.2058	0.2058	0.2058	0.2058	0.2058	0.2058
Vmin (V)	-0.2025	-0.207	-0.2025	-0.2025	-0.2025	-0.2025	-0.2025	-0.2025
Vpk-pk (V)	0.4082	0.4127	0.4082	0.4082	0.4082	0.4082	0.4082	0.4082
Vavg (V)	0.2005	0.2005	0.2005	0.2006	0.2005	0.2005	0.2005	0.2004
Vrms (V)	0.2005	0.2005	0.2005	0.2006	0.2005	0.2005	0.2005	0.2004

Table A-2-T1: Results of square wave by PicoScope 2104

Measurement	Frequency of signal							
	10Hz	**20Hz**	**50Hz**	**100Hz**	**500Hz**	**1000Hz**	**1500Hz**	**2000Hz**
Frequency (Hz)	10.724	23.334	49.966	100.414	489.991	1080.779	1501.082	2094.744
Cycle Time (ms)	93.247	42.856	20.013	9.959	2.041	0.925	0.666	0.477
Duty Cycle (%)	78.84	79.21	80.55	80.899	76.364	80	77.778	76.923
Rise Time (ms)	0	0	0	0	0	0	0	0
Fall Time (ms)	0	0	0	0	0	0	0	0
Vmax (V)	0.215	0.215	0.215	0.226	0.217	0.214	0.22	-0.226
Vmin (V)	-0.216	-0.215	-0.219	-0.22	-0.215	-0.213	-0.221	0.217
Vpk-pk (V)	0.432	0.43	0.434	0.447	0.432	0.427	0.441	0.443
Vavg (V)	0.199	0.199	0.199	0.198	0.199	0.199	0.198	0.198
Vrms (V)	0.199	0.199	0.199	0.198	0.199	0.199	0.198	0.198

Table A-2-T2: Results of square wave by designed oscilloscope application

APPENDIX A

A-3 Triangle Wave

Signal : Triangle wave
Peak voltage : 0.7V

Measurement	Frequency of signal							
	10Hz	**20Hz**	**50Hz**	**100Hz**	**500Hz**	**1000Hz**	**1500Hz**	**2000Hz**
Frequency (Hz)	10	19.99	49.99	99.99	499.9	1000	1500	2000
Cycle Time (ms)	99.98	50.03	20	10	2	0.9999	0.6667	0.4999
Duty Cycle (%)	49.55	49.24	50.05	50.31	49.49	50.26	50.58	50.52
Rise Time (ms)	40.26	20.11	8.044	3.941	0.8038	0.3958	0.2637	0.198
Fall Time (ms)	40.3	20.09	8.024	3.957	0.801	0.397.5	0.2645	0.1987
Vmax (V)	0.2013	0.2013	0.2013	0.1968	0.2013	0.1968	0.1968	0.1968
Vmin (V)	-0.2025	-0.2025	-0.2025	-0.2025	-0.2025	-0.2025	-0.2025	-0.2025
Vpk-pk (V)	0.4038	0.4038	0.4038	0.3993	0.4038	0.3993	0.3993	0.3993
Vavg (V)	0.35	0.351	0.35	0.35	0.349	0.35	0.35	0.351
Vrms (V)	0.1159	0.116	0.1159	0.1159	0.116	0.1158	0.1158	0.1157

Table A-3-T1: Results of triangle wave by PicoScope 2104

Measurement	Frequency of Signal							
	10Hz	**20Hz**	**50Hz**	**100Hz**	**500Hz**	**1000Hz**	**1500Hz**	**2000Hz**
Frequency (Hz)	12.462	20.706	48.519	103.766	502.391	1089.039	1537.467	2151.055
Cycle Time (ms)	80.241	48.296	52.985	9.637	1.99	0.918	0.65	0.465
Duty Cycle (%)	62.524	49.363	20.61	49.2	50.98	50	52.941	50
Rise Time (ms)	38.047	20.495	7.767	3.893	0.781	0.383	0.268	0.155
Fall Time (ms)	28.58	19.36	7.786	3.864	0.773	0.499	0.266	0.193
Vmax (V)	0.708	0.707	0.708	0.709	0.708	0.709	0.712	0.709
Vmin (V)	-0.707	-0.707	-0.706	-0.707	-0.707	-0.706	-0.706	-0.707
Vpk-pk (V)	1.415	1.415	1.414	1.416	1.415	1.415	1.418	1.416
Vavg (V)	0.264	0.362	0.352	0.355	0.349	0.349	0.374	0.352
Vrms (V)	0.303	0.421	0.406	0.409	0.406	0.406	0.418	0.412

Table A-3-T2: Results of triangle wave by designed oscilloscope application

APPENDIX A

A-4 Ramp Up Wave

Signal : Ramp up wave
Peak voltage : 0.45V

Measurement	Frequency of signal							
	10Hz	**20Hz**	**50Hz**	**100Hz**	**500Hz**	**1000Hz**	**1500Hz**	**2000Hz**
Frequency (Hz)	10	20	49.99	99.96	500	999.8	1500	2
Cycle Time (ms)	99.96	50	20	10	2	1	0.6667	0.5001
Duty Cycle (%)	44.92	44.99	45.27	44.98	45.36	44.94	44.96	45
Rise Time (ms)	79.63	39.79	15.9	7.921	1.583	0.7974	0.5314	0.3976
Fall Time (ms)	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
Vmax (V)	0.4525	0.4525	0.4525	0.448	0.448	0.4525	0.4525	0.448
Vmin (V)	-0.4492	-0.4492	-0.4492	-0.4492	-0.4492	-0.4492	-0.4492	-0.4492
Vpk-pk (V)	0.9018	0.9018	0.9018	0.8973	0.8973	0.9018	0.9018	0.8973
Vavg (V)	0.2204	0.2204	0.2204	0.2204	0.2204	0.2204	0.2204	0.2204
Vrms (V)	0.26	0.2601	0.26	0.2599	0.2599	0.2598	0.2657	0.2595

Table A-4-T1: Results of ramp up wave by PicoScope 2104

Measurement	Frequency of signal							
	10Hz	**20Hz**	**50Hz**	**100Hz**	**500Hz**	**1000Hz**	**1500Hz**	**2000Hz**
Frequency (Hz)	10.853	22.72	53.61	212.752	538.99	1044.637	1585.264	1940.041
Cycle Time (ms)	92.141	44.015	18.653	4.7	1.855	0.957	0.631	0.515
Duty Cycle (%)	48.701	53.373	50	49.606	50	50	52.941	50
Rise Time (ms)	62.674	39.88	15.1	3.701	1.484	0.736	0.557	0.405
Fall Time (ms)	0	0.037	0	0	0	0	0	0
Vmax (V)	0.454	0.453	0.453	0.455	0.453	0.453	0.456	0.454
Vmin (V)	-0.454	-0.456	-0.458	-0.455	-0.46	-0.456	-0.456	-0.455
Vpk-pk (V)	0.909	0.91	0.91	0.91	0.913	0.909	0.912	0.909
Vavg (V)	0.226	0.227	0.226	0.244	0.226	0.233	0.226	0.219
Vrms (V)	0.26	0.259	0.261	0.275	0.262	0.268	0.264	0.251

Table A-4-T2: Results of ramp up wave by designed oscilloscope application