COMPUTER-BASED INSTRUMENTATION SYSTEM FOR TEMPERATURE MEASUREMANT USING THERMOCOUPLE IN VISUAL BASIC APPLICATION
(C.I.S.T.V.A)

MUHAMAD AKMAL BIN ISHAK

UNIVERSITY MALAYSIA PAHANG

UNIVERSITI MALAYSIA PAHANG

BORANG PENGESAHAN STATUS TESIS♦

JUDUL: **DEVELOPMENT OF THE COMPUTER-AIDED DIAGRAM (CAD) FOR ELECTRICAL MACHINE FOR UNDERGRADUATE PURPOSE LEARNING**

SESI PENGAJIAN: **2008/2009**

Saya **MUHAMAD AKMAL BIN ISHAK (860213-06-5423)**
(HURUF BESAR)

mengaku membenarkan tesis (Sarjana Muda/~~Sarjana~~ /~~Doktor Falsafah~~)* ini disimpan di Perpustakaan dengan syarat-syarat kegunaan seperti berikut:

1. Tesis adalah hakmilik Universiti Malaysia Pahang (UMP).
2. Perpustakaan dibenarkan membuat salinan untuk tujuan pengajian sahaja.
3. Perpustakaan dibenarkan membuat salinan tesis ini sebagai bahan pertukaran antara institusi pengajian tinggi.
4. **Sila tandakan (√)

☐	**SULIT**	(Mengandungi maklumat yang berdarjah keselamatan atau kepentingan Malaysia seperti yang termaktub di dalam AKTA RAHSIA RASMI 1972)
☐	**TERHAD**	(Mengandungi maklumat TERHAD yang telah ditentukan oleh organisasi/badan di mana penyelidikan dijalankan)
√	**TIDAK TERHAD**	

Disahkan oleh:

_____ (TANDATANGAN PENULIS)

_____ (TANDATANGAN PENYELIA)

Alamat Tetap:

KM 11, JLN PADANG TENGKU,
27200, KUALA LIPIS,
PAHANG DARUL MAKMUR,

MR. MOHD ANWAR ZAWAWI
(Nama Penyelia)

Tarikh: **11 NOVEMBER 2008**

Tarikh: : **11 NOVEMBER 2008**

CATATAN:
- * Potong yang tidak berkenaan.
- ** Jika tesis ini SULIT atau TERHAD, sila lampirkan surat daripada pihak berkuasa/organisasi berkenaan dengan menyatakan sekali tempoh tesis ini perlu dikelaskan sebagai atau TERHAD.
- ♦ Tesis dimaksudkan sebagai tesis bagi Ijazah doktor Falsafah dan Sarjana secara Penyelidikan, atau disertasi bagi pengajian secara kerja kursus dan penyelidikan, atau Laporan Projek Sarjana Muda (PSM).

"I hereby acknowledge that the scope and quality of this thesis is qualified for the award of the Bachelor Degree of Electrical Engineering (Electronic)"

Signature : _____

Name : MR. MOHD ANWAR ZAWAWI

Date : 11 NOVEMBER 2008

COMPUTER-BASED INSTRUMENTATION SYSTEM FOR TEMPERATURE MEASUREMANT USING THERMOCOUPLE IN VISUAL BASIC APPLICATION
(C.I.S.T.V.A)

MUHAMAD AKMAL BIN ISHAK

This thesis is submitted as partial fulfillment of the requirements for the award of the Bachelor of Electrical Engineering (Electronic)

Faculty of Electrical & Electronics Engineering
University Malaysia Pahang

OCTOBER 2008

Signature : _____

Author : MUHAMAD AKMAL BIN ISHAK

Date : 11 NOVEMBER 2008

Specially dedicated to

My beloved parent

ACKNOWLEDGEMENT

Alhamdulillah, a lot of praise and ‘syukur’ to ALLAH. I wish to express my sincere gratitude and appreciation to Ms. Najidah Binti Hambali as my 1st supervisor and Mr. Anwar Zawawi as my 2nd supervisor for encouragement, guidance and motivation. Without their never ending guidance, patience and encouragement throughout this project, I would never finish this project as it is. Thank you very much! Not forget to the laboratory assistant, Mr. Hamka, who spends time to help my researches in the lab.

My fellow friends and colleagues should also be recognized for their support. Without them, I do not think that I can get through this. Their tips and views are very useful in completing this project. I would also like to thanks to panels during FKEE R&D exhibition.

Last but not least, I would like to use this opportunity to say thank you to my beloved parents, Ishak bin Sat and Noriah bt Abdul Rahman. Finally, I would like to express my appreciation to all my friends, especially to Izhan, Syamil, Mino and Faris, thanks for your love and constant moral support.

Thank you to all of you

Assalamualaikum.

ABSTRACT

In this project, Visual Basic is used as a main programming language to develop a GUI (Graphical User Interface) application. This application is developed to help student in studying the industrial instrumentation subject. For this project, thermocouple sensor type K will be used as an input device to detect temperature changes. The input will be converted into current signal between 4 - 20mA. Then, a DAQ card will be used to interface between the instrument and computer. The software is divided into 4 sections, data preview, data control, application and setting. Data preview section is used to preview live data from DAQ card. When the data is completely recorded, the data is manipulated to get appropriate result and graph, using data control section. In application section, some useful applications have been added for student such as converter, uncertainty calculation, graph generator, live graph, export data, data recorder and data plotter. And the setting section has function for software's setting.

ABSTRAK

Dalam projek ini, Visual Basic digunakan sebagai bahasa pengaturcaraan utama untuk membina aplikasi AGP (Antaramuka Grafik Pengguna). Aplikasi ini dibangunkan untuk membantu pelajar dalam subjek Industrial Instrumentation. Untuk projek ini thermocouple jenis K akan digunakan sebagai alat untuk mengesan perubahan suhu. Input akan ditukar kepada signal arus dalam 4-20 mA. Kemudian, DAQ akan digunakan untuk mengantaramuka antara komputer dan peralatan. Perisian ini dibahagiakan kepada 4 bahagian, iaitu data pratonton, data kawalan, aplikasi and tetapan. Bahagian data pratonton digunakan untuk melihat data secara langsung dari kad DAQ. Apabila data telah sepenuhnya direkodkan, data akan dimanipulasi untuk mendapatkan keputusan dan graf yang sepatutnya menggunakan bahagian data kawalan. Dalam bahagian aplikasi, aplikasi yang berguna telah ditambah untuk pelajar seperti pengubah, kiraan ketikpastian, penjana graf, graf langsung, data ekspot, perakam data dan pemplot data. Dan bahagian tetapan mempunyai fungsi untuk tetapan perisian.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABREVIATIONS

GUI	-	Graphical User Interface
VB	-	Visual Basic
SIMULATION	-	Simulation and Capture
DAQ	-	Data Acquisition
DAO	-	Data Access Object
RDO	-	Remote Data Objects
DOS	-	Disk Operating System
BASIC	-	Beginners' All-purpose Symbolic Instruction Code

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION

1.1 Background

Nowadays, the major change occurring at the present is the increasing number of user friendly software that make it possible for student to experience new and fast ways of learning. In minutes, simulation, controller and real world interfacing can be created instantly. The software is developed to help students to learn and explore the experiment with an interesting and interactive way.

In this project, Visual Basic will be used as a main programming language to develop a GUI (Graphical User Interface) application. This application is developed to help student in studying the industrial instrumentation subject.

For this project, thermocouple sensor type K will be used as an input device to detect temperature changes. The input will be converted into current signal between 4 - 20mA. Then, a DAQ card will be used to interface between the instrument and computer.

The software will be developed using Visual Basic to calculate and analyse the output. Visual basic software does all the manipulation, analysis and report generation.

1.2 Objectives

There are three main objectives of the project which are:-

1. To understand basic measurement principal of temperature instrumentation. For this project, the instruments that will be used consist of several parts. This part is important because it is an input for the system and required deeper knowledge and understanding.

2. Interface the temperature transmitter output to software application. The interface process is done with Data Acquisition process (DAQ). DAQ card will be used to interface between instrument and computer.

3. Develop software application to help in student learning process. Visual Basic 2008 Express Editon will be used as a main programming language. The software is developed to be interactive and user friendly to the user. And the software will be used during lab session of Industrial Instrumentation subject (BEE4523) to help in student learning process.

1.3 Scopes

This project involves designing the software application to analysis the data using Microsoft Visual Basic 2008 Express Edition. USB Data Acquisition (USB-4716) is used to interface between the computer and temperature instrument such as temperature transmitter, hart communicator and digital manometer. Thermocouple type K is used as a primary transducer to detect temperature changes in Isotech Jupiter temperature bath.

1.4 Research Methodology

i) Literature review to understand the concept and identify the problems and techniques.
ii) Understand the whole system especially how to communicate between PC and instrument.
iii) Design and writing program according to lecturer needs and understanding of the system.
iv) Interface between computer and thermocouple through USB-4716DAQ card
v) Test the software and verify the result with manual calculation.

Design step of work methodology can be simplified as shown in Figure 1.1

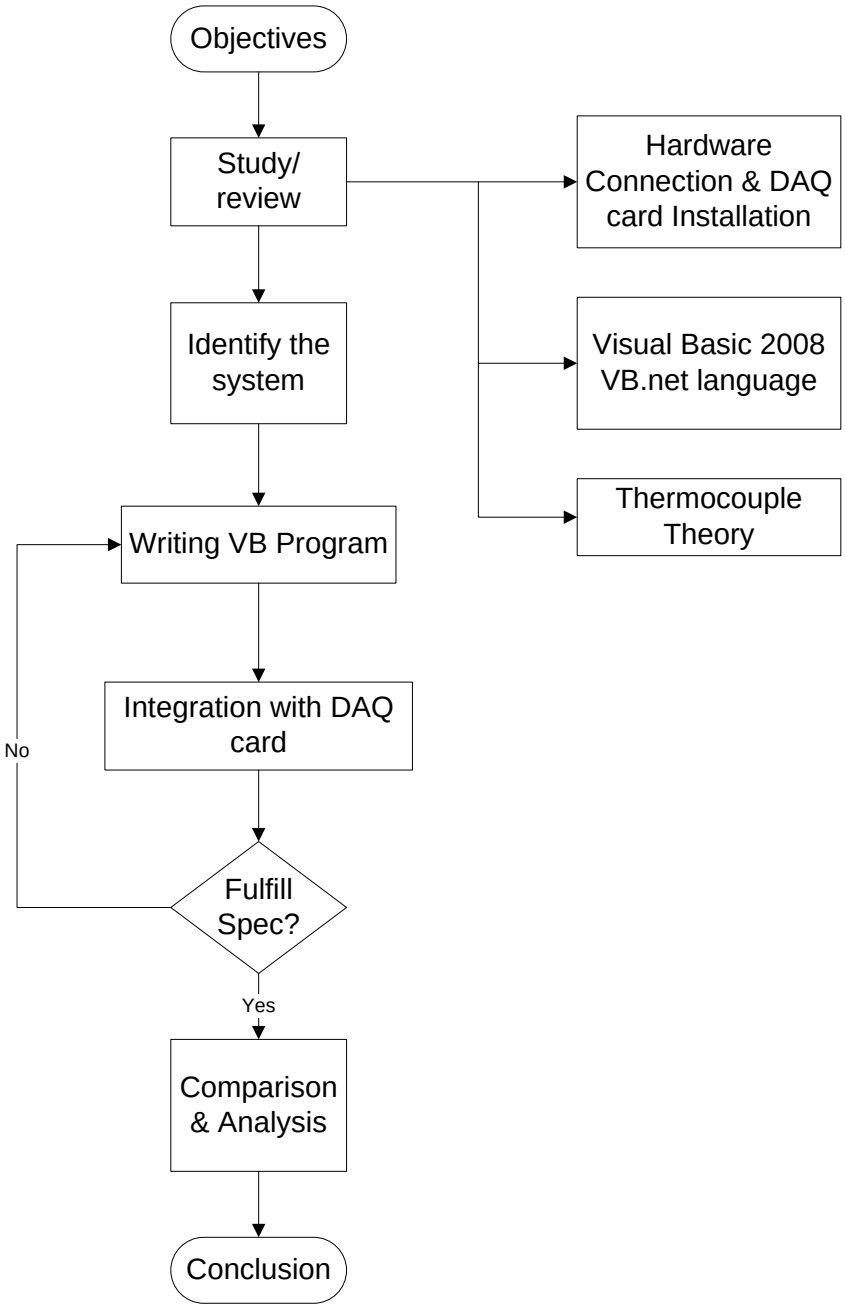

Figure 1.1: Design Flowcharts.

CHAPTER 2

LITERATURE REVIEW

2.1 Evaluation of the Freezing Point of Zinc for Pt/Pd Thermocouple Calibration

This research was conducted by H.Narushima*, H.Ogura, M.Izuchi and M.Arai from National Metrology Institute of Japan, National Institute of Advanced Industrial Science and Technology (NMIWIST). In this research, a new apparatus was developed to calibrate thermocouple at zinc freezing point. The stability of the temperature repeatability as well as effect of surroundings temperature was also investigated as uncertainty components.

With the rapid globalization of economic activities and borderlessness of industry, many kinds of products are assembled by parts from several countries. To this concern, measuring instruments traceable to national standards for these products and parts are essential. In the field of measuring instruments, a lot of thermocouples are widely used. To calibrate these thermocouples accurately, several fixed points (Cu, Ag, AI) apparatuses have been developed”’ since 2000 at NMIJ. The zinc fixed-point apparatus was needed to calibrate thermocouples from 0°C to 1100°C [5].

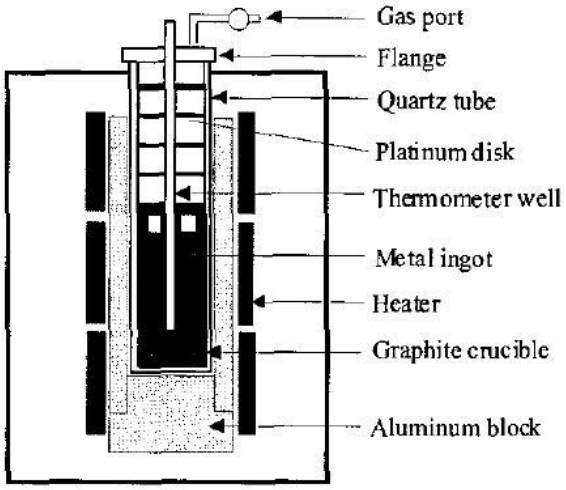

Figure 2.1: Zinc Freezing Point Furnace

Schematic view of the zinc freezing point fumace is shown in Fig.1. The zinc freezing point cell installed in the aluminum isothermal block is heated hy a three-zone heater. To reduce the electromagnetic noise during measurement, the sheathed heaters are wound noninductively and supplied by DC power. The zinc freezing point cell is an open type filled with pressurized argon gas whose pressure is measurable.[5] To measure the emf of thermocouples, reference junction is generally maintain at 0°C in an ice bath.

At the start of estimating zinc freezing point realization apparatus, an automatically operating ice-point device was used to maintain the temperature of the reference junction. But, on and off of the electronically cooling in the automatically operating ice-point device induced the temperature up and down to the reference junction [5].

In this experiment Pt/Pd thermocouples were used, because they show outstanding stability compared to the conventional thermocouples.

2.2 An Automated Thermocouple Calibration System

An Automated Thermocouple Calibration System was invented by Mark D. Bethea and Bruce N. Rosenthal. It was developed for calibration type K thermocouple.

This system operates from room temperature to 650°C and has been used for calibration of thermocouples in an eight-zone furnace system which may employ as many as 60 thermocouples simultaneously. It is highly efficient, allowing for the calibration of large numbers of thermocouples in significantly less time than required for manual calibrations. The system consists of a personal computer, a data acquisition/ control unit, and a laboratory calibration furnace. The calibration furnace is a microprocessor-controlled multipurpose temperature calibrator with an accuracy of f.7”C. The accuracy of the calibration furnace is traceable to the National Institute of Standards and Technology (NIST). The computer software is menu-based to give the user flexibility and ease of use. The user needs no programming experience to operate the systems [6]. The purpose of calibration is to determine if the thermocouple being tested are within the standard. If the thermocouple is outside the standard, a proper calibration must be made.

The calibration of a thermocouple consists of the determination of its electromotive force (emf) at a sufficient number of known temperatures so that with some means of interpolation, its emf will be known over the temperature range in which it is to be used. This process requires a standard thermometer to indicate temperatures on a standard scale, a means for measuring the emf of the thermocouple, and a controlled environment in which the thermocouple and the standard can be brought to the same temperature [6]. The standard thermometer can be considered as MSU (Master Standard Unit). Usually digital thermometer and RTD is used as MSU because it has high accuracy and fast response.

To calibrate thermocouples by the temperature comparison method, the thermocouples are placed in a special calibration furnace which provides a stable, repeatable environment. The heating block inside the furnace has a high thermal

conductivity and is heated by resistance elements. The temperature is controlled by a precision platinum resistance thermometer (RTD sensor) and a closed-loop circuit [6].

Using the ATCS, the experimenter needs only to be present at the beginning and end of a calibration session. If a thermocouple is to be calibrated at five different temperatures, the experimenter simply specifies those temperatures at the start of the calibration and then returns when all five temperatures have been reached and the calibration session is complete [6]. This benefit of this system is it can save a lot of time for calibrating thermocouples and reduce the cost. This system can be applied in industry that uses a lot of thermocouples.

2.3 High Speed PC-based Data Acquisition Systems

Until recently, within the last three years, high speed data acquisition systems which guaranteed data integrity could only be implemented on high end platforms such as VAX/VMS or U N I X systems. These provided sophisticated multitasking operating systems and enough horsepower to not only provide data collection of tens of thousands of U 0 points, but also provide graphical interfaces to users. These systems were, and still are, expensive to buy and maintain [7].

Data collection on PCs has been, and continues to be, rather disappointing for users who demand high speed sampling, data integrity, and data storage. This has been due to a combination of the lack of CPU power, the lack of sophistication of PC operating systems, and the slow speed of peripheral devices such as disk drives. Therefore, many users are reluctant to leave their high end systems and look at what can be done on the PC platform [7]. Recent availability of powerful PC hardware and software has now made it possible for the PC platform to match the data acquisition performance of traditional high end systems.

The first step in designing and implementing a flexible, high speed data acquisition system on the PC platform is to specify a set of goals which the system must achieve. This section lists these goals, and also how most other PC systems fail to meet them. The failure of these existing systems to meet these goals is what separates them from the PC system being designed [7]. Thus, the following goals are discussed.

2.3.1 Guaranteed high speed data acquisition

First, the system must support high speed data acquisition. High speed processes, such as variable speed drive control, and steel rolling, demand high speed data acquisition. Sample intervals of 20 milliseconds to 50 milliseconds are the goal of the system being discussed here. Also, to meet the data demands of modern production machinery, the system must be able to provide this sampling for up to 10,000 analog and digital values from the monitored system [7].

2.3.2 Minimized channel skew

Skew refers to drifting in time from the specified sampling interval. For example, reading a 20ms sample at 18ms since the last sample or at 22ms since the last sample means that a 2ms skew has been introduced into the data collection. When skew approaches the sample interval, detecting cause and effect relationships using trended and exception data becomes impossible [7]. You can think of the channel skew as the time it takes the analog input subsystem to sample a single channel.

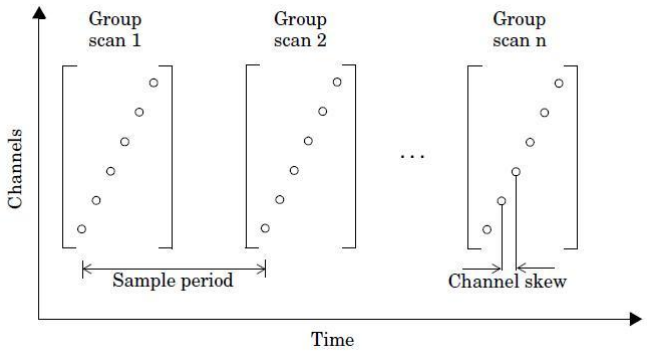

Figure 2.2: Channel Skew

Not only must skew be minimized from sample to sample, but also within samples. For example, skew is introduced into a sample if the system interrupts its sampling to perform an operator task such as responding to a mouse click. Introducing skew within a sample makes the data collected in that sample questionable, and there is no way to tell that the skew was introduced [7].

2.3.3 High Speed Response

In order to monitor high speed processes, the PC platform must be powerful enough to support the running software. This includes components such as the microprocessor, peripheral devices like the disk drive and video accelerator, and memory capacity. Powerful PC platforms have become available only within past several years. Also, the operating software must be sophisticated enough to take advantage of provided platform [7].

This goal must be met in order to provide a useful operator interface. Graphical elements on the screen should update fast enough to emulate analog display devices, such as bar graphs. If these graphical elements cannot update this quickly, problems in the monitored process can be masked [7].

2.3.4 Data Acquisition Functionality

The system must provide all expected data acquisition functionality, such as fault and alarm management, and trending. This functionality must be provided at the sampling rates being supported, and for suitable lengths of time, for example, storing trended data for several days [7].

Indeed, one of the goals of the system is to provide trending for up to 1,000 digital and analog values continuously and simultaneously [7]. This will allow maintenance personnel to diagnose a problem without having to specify which values to trend beforehand.

2.3.5 Hardware

The hardware of the PC system is the foundation of high speed data acquisition and storage. The most sophisticated operating software cannot provide the required functionality if the hardware cannot run it fast enough. As mentioned previously, it is only in recent years that hardware powerful enough for this system has become available [7]. A very powerful PC platform can be put together today for less than RM 2000. The base platform for the PC system being designed here is a desktop PC. The minimum requirement for Data Acquisition System is:

- 2 GHz microprocessor
- 1 GB RAM
- 60GB Hard Drive
- Graphic Card

2.3.6 Operating System

The operating system provides the environment in which all of the PC system software runs. It controls which competing software tasks can have access to the underlying hardware, including the CPU. It also controls when tasks can access the hardware. Different operating systems control access differently. In the PC environment, either “cooperative” multitasking or “preemptive” multitasking is provided [7]. Nowadays, most PC use Microsoft Windows as their operating system. Engineering software such as MatLab, LabView only support Microsoft Windows as their operating system and sometimes we can only find driver for Microsoft Windows.

2.4 Thermocouple

2.4.1 Thermocouple

The basic theory of a thermocouple is found from a consideration of the electrical and thermal transport. When the temperature changes at the junction formed by joining two unlike conductors, its electron configuration changes due to the resulting heat transfer. This electron reconfiguration produces a voltage (emf or electromotive force), and is known as Seeback effect. Two junctions or more of a thermocouple are made with two unlike conductors such as iron and constant an, copper and constantan, chrome and alumel, and so on. One junction is place in a reference source (cold junction) and the other in the temperature.[1]

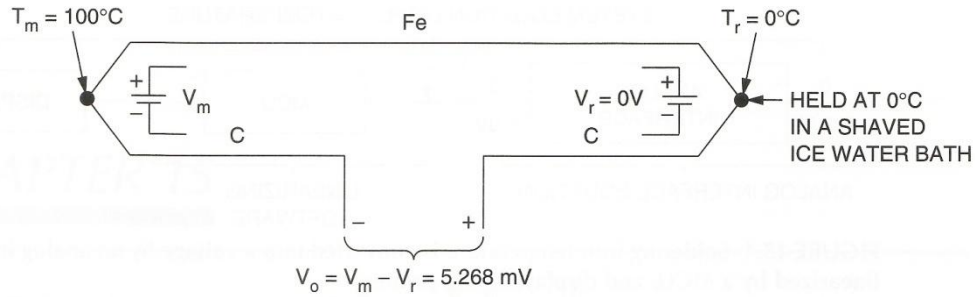

Figure 2.3: Thermocouple Junction

One junction in Figure 1.0 is placed in a shave ice bath and the temperature is 0°C. This junction is reference or cold junction and its thermoelectric reference voltage is defined as V_r = 0V. Temperatures are measured with respect to T_r by the remaining measuring junction, T_m. If T_m is placed in boiling water at T_m 100°C, V_o will equal 5.268mV with the polarity shown in figure 1.0.[1]

2.4.2 Thermocouple types

Thermocouples consist of many types, such as type J, K, T, E, S, and R. Each type has its particular features, such as range, linearity, inertness to hostile environment and sensitivity. In each type, various sizes of conductors may be employed for specific cases such as oven measurement.[3] Thermocouple is considered as transducer. Transducers and sensors convert a real world signal into measurable electrical signal, such as voltage or current. A thermocouple will produce a voltage difference that increases as the temperature increases. For this project thermocouple type K will be used as it is the most commonly used thermocouple.

Table 2.4: Thermocouple type

Type	Materials	Normal Range
J	Iron-constantan	-190°C to760°C
T	Copper-constantan	-200°C to 371°C

K	Chromel-alumel	-190°C to 1260°C
E	Chromel-constantan	-100°C to 1260°C
S	90% platinum+ 10% rhodium-platinum	0°C to 1482°C
R	87% platinum + 13% rhodium-platinum	0°C to 1482°C

2.5 Data Acquisition

Data acquisition is a process of gathering and sampling signals from real world to generate data which can be stored, analyzed, and presented by a PC. Generation of these signals from the real world is through instruments and sensors, and each type of signal needs its own special consideration. Simple ways to store data use a strip-chart recorder or an x-y plotter. But for modern instruments, digitized data is essential for later analysis and storage purposes. Digital system samples all data collected from the instruments, takes average of the data and gives an improved value of the measured signal. Many instruments digitize and store data in their inbuilt memory which can later be transferred to a computer. Other instruments use certain software for data acquisition and control. [2] Data acquisition is a process used to convert a signal in analog to digital so the computer can understand thus process the signal in computer language.

A transducer is an electronic device that converts energy from one form to another. Common examples include microphones, loudspeakers, thermometers, position and pressure sensors, and antenna. Although not generally thought of as transducers, photocells, LEDs (light-emitting diodes), and even common light bulbs are transducers [6].The ability of a data acquisition system to measure different phenomena depends on the transducers to convert the physical phenomena into signals measurable by the data acquisition hardware. Transducers are synonymous with sensors in DAQ systems. There are specific transducers for many different applications, such as measuring temperature, pressure, or fluid flow. DAQ also

deploy various Signal Conditioning techniques to adequately modify various different electrical signals into voltage that can then be digitized using ADCs. [3]

Signal conditioning may be necessary if the signal from the transducer is not suitable for the DAQ hardware to be used. The signal may be amplified or deamplified, or may require filtering, or a lock-in amplifier is included to perform demodulation. Various other examples of signal conditioning might be bridge completion, providing current or voltage excitation to the sensor, isolation, linearization, etc.[3]

2.6 Visual Basic

Visual Basic (VB) is a third-generation event-driven programming language and associated development environment (IDE) from Microsoft for its COM programming model. [9]

Visual Basic was designed to be easy to learn and use. The language not only allows programmers to create simple GUI applications, but can also develop complex applications as well. Programming in VB is a combination of visually arranging components or controls on a form, specifying attributes and actions of those components, and writing additional lines of code for more functionality. Since default attributes and actions are defined for the components, a simple program can be created without the programmer having to write many lines of code. Performance problems were experienced by earlier versions, but with faster computers and native code compilation this has become less of an issue [9].

A programmer can put together an application using the components provided with Visual Basic itself. Programs written in Visual Basic can also use the Windows API, but doing so requires external function declarations.

The latest version of Visual Basic is Microsoft Visual Basic 2008 Express Edition.

2.7 Calibration

Calibration is the process of establishing the relationship between a measuring device and the units of measure. This is done by comparing a device or the output of an instrument to a standard having known measurement characteristics. For example the length of a stick can be calibrated by comparing it to a standard that has a known length. Once the relationship of the stick to the standard is known the stick is calibrated and can be used to measure the length of other things.[9]

2.8 Standard Deviation

The standard deviation is a measure of the dispersion of a set of values. It can apply to a probability distribution, a random variable, a population or a multiset. The standard deviation is usually denoted with the letter σ (lowercase sigma). It is defined as the root-mean-square (RMS) deviation of the values from their mean, or as the square root of the variance. [9]

This calculation is described by the following formula:

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \overline{x})^2}$$

Where the mean of X is define as:

$$\overline{x} = \frac{x_1 + x_2 + \cdots + x_N}{N} = \frac{1}{N}\sum_{i=1}^{N} x_i.$$

CHAPTER 3

INSTRUMENTS AND HARDWARES

3.1 Overall System Connection

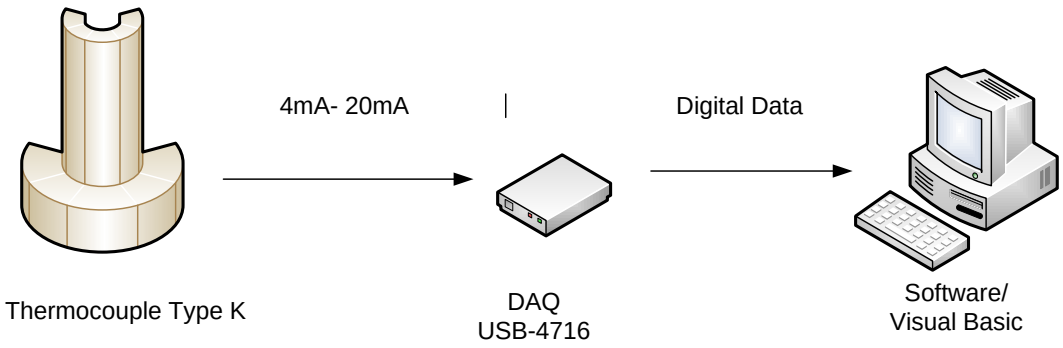

Figure 3.1: Overall system connection

Figure above show the overall system connection for this project. There are three major parts:

i. Thermocouple Type K- Thermocouple will detect temperature changes and then transmit the value in voltage form to DAQ.

ii) DAQ USB-4716 – DAQ is used to convert analogue data to digital data.

iii) Software – Visual basic is used to stored, saved and analysis the data.

Thermocouple and DAQ card is considered as hardware part in this project.

3.1.1 Basic instrument connection

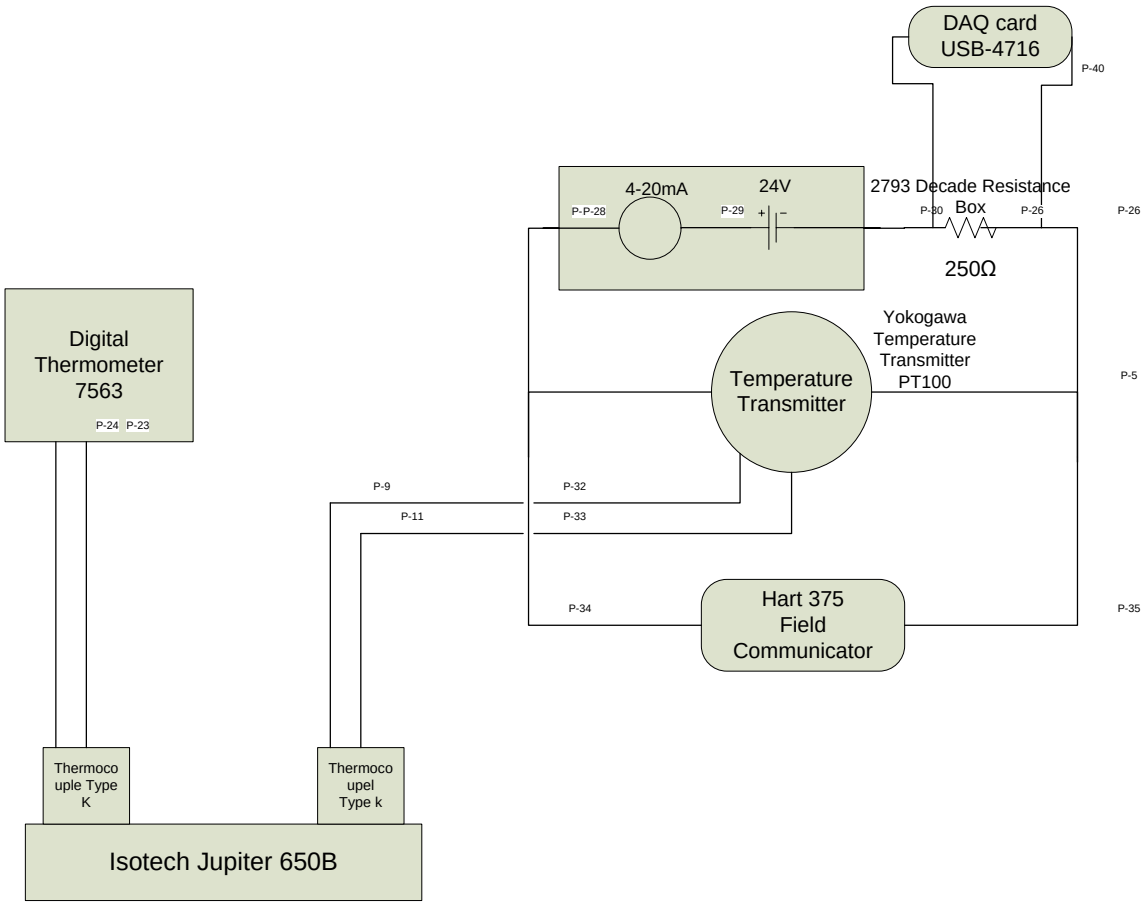

Figure 3.2: Basic Instrument Connection

Instrument part as in Figure 2 consists of Digital Thermometer 7563, RTD, Isotech Jupiter 650B, Yokogawa Digital Manometer MT220, Decade Resistance Box, Yokogawa Temperature Transmitter PT100, and Hart 375 Field Communicator. Thermocouple type K is used as an input device to detect temperature changes. Type K (chromel–alumel) is the most commonly used general purpose thermocouple. It is inexpensive and, owing to its popularity, available in a wide variety of probes. They

are available in the −200 °C to +1350 °C range. Isotech Jupiter 650B is an artificial heater used to simulate temperature changes. Thermocouple will detect the temperature changes and transmit the signal to Yokagawa Temperature Transmitter. Then, Yokogawa Temperature Transmitter will convert the signal to current value and the current can be read by Digital Manometer or Hart 375 Field Communicator.

Figure 3.3: Instruments

3.2 Instruments

3.2.1 Thermocouple

Thermocouple is a transducer used to convert temperature value into equivalent value of voltage or current. In this project, there are two thermocouple types K used. One Master Standard Unit (MSU), for reference point and another one for Unit Under Test (UUT). These thermocouples produce output between 0 to 44μV

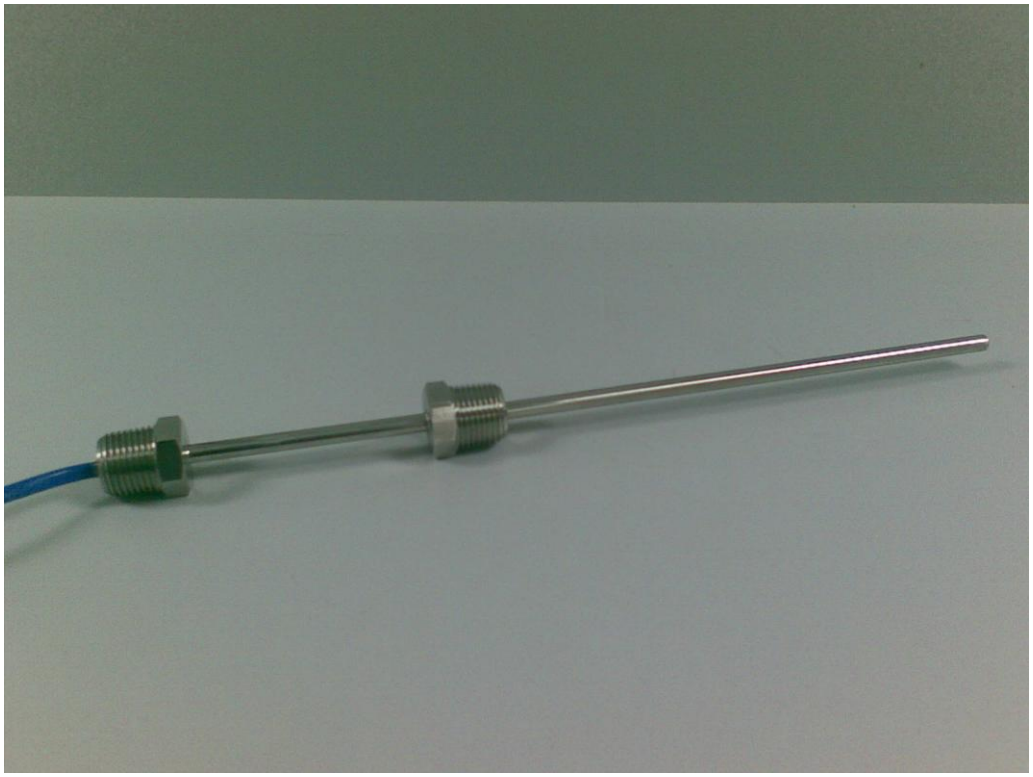

Figure 3.4: Thermocouple

Thermocouple properties:

1) Sensor type : TC type K
2) Lower Range value: -270°C
3) Upper Range value : 1372°C
4) Minimum Range : 50°C
5) Accuracy : ± 0.45°C

3.2.2 Hart 375 Field Communicator.

Hard communicator has many functions, and one of them is to calibrate the thermocouple current with specific temperature ranges. For instance, if we want to read value between 50°C - 200°C we need to set the high PV value at 200°C and the lowest PV value at 50°C using hart communicator. The calculation how to calibrate the device is shown in chapter analysis and result. Hart communicator also used to read present value such as current and temperature from instrument.

Figure 3.5: Hart Field Communicator

3.2.3 Yokogawa Temperature Transmitter

Temperature transmitter converts the temperature dependent change in resistance or voltage of the instrument into a load independent current standard signal. In other words, it converts low level voltage from thermocouple to current value. Current is use to transmits signal because it is much less affected by environmental noise. A 40-20mA signal has the advantage that even at minimum signal current value, there should be a detectable current flowing. The absence of current signal indicates a wiring problem.

Figure 3.6: Temperature Transmitter

3.2.4 Isotech Jupiter

Figure 3.7: Isotech Jupiter Temperature Bath

Isotech Jupiter is a heater used to heat up the thermocouple to a specific temperature. The Isotech Jupiter's controller has dual display, the upper display indicates the current temperature and the lower display indicates the setpoint or desired temperature. To change the temperature set point, simply use the UP and DOWN keys to raise and lower the setpoint. The temperature changes very quick to desired temperature in heating up process but it will take time to cool down to desired temperature.

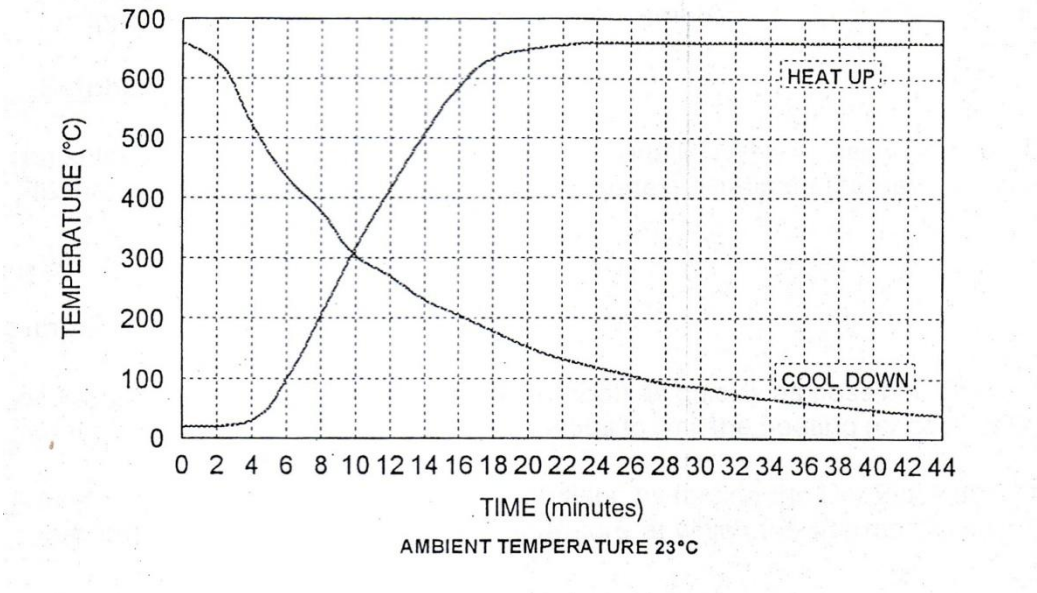

Figure 3.8: Isotech Graph

The graph above shows a theoretical value of temperature heating up process and cool down process. The actual value taken from real experiment shows a slightly different result. All experiment results are shown in chapter 4.

3.2.5 Decade Resistance Box

Figure 3.9: Decade Resistance Box

Decade resistance box is adjustable resistor used to give minimum load to Hart Communicator. Without a minimum load, the hart communicator won't work.

3.2.6 Yokogawa MT220- Digital Manometer

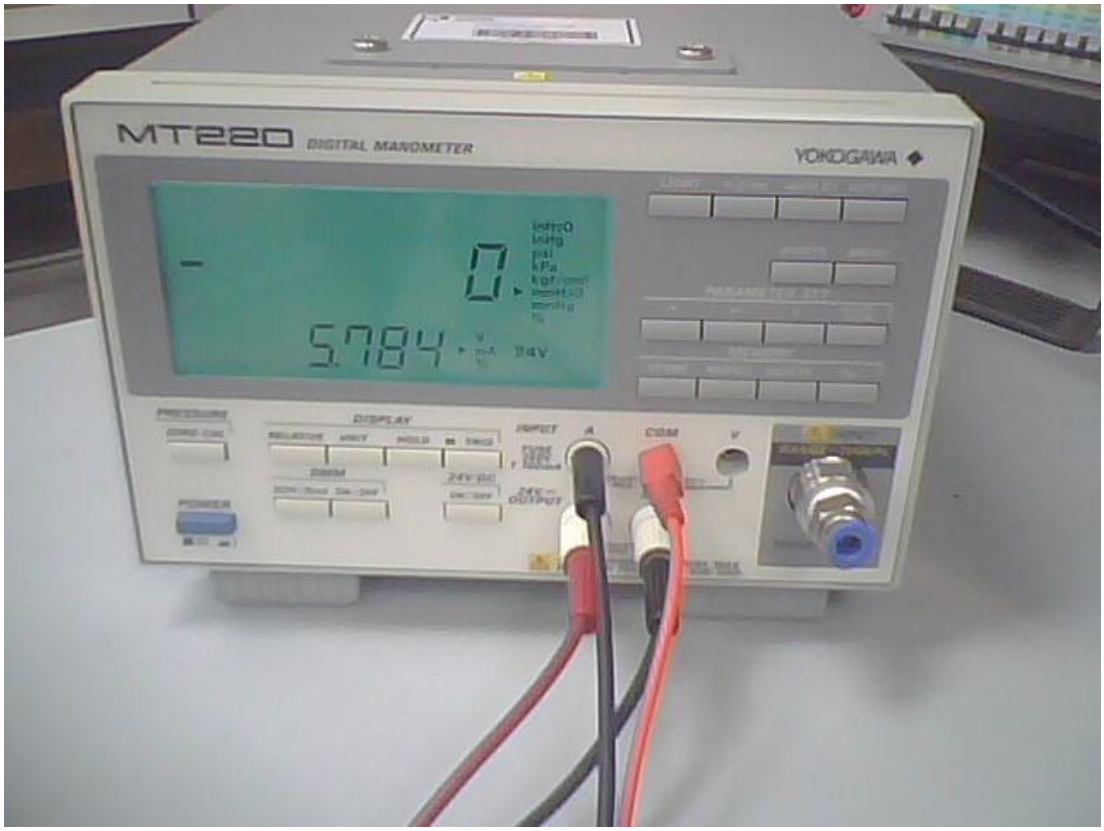

Figure 3.10: Yokogawa MT220- Digital Manometer

Yokogawa MT220- Digital Manometer used as digital ammeter as well as power supply for the circuit.

3.2.7 Digital thermometer 7563

Digital thermometer is used as a Master Standard Unit (MSU). It has high sensitivity measurement. This instrument is capable measuring DC voltages, resistance and temperature (Thermocouple and RTD). This model also provides a thermocouple temperature measurement resolution of 0.1°C, and temperature 0.01°C when using RTD.

Figure 3.11: Digital thermometer 7563

3.3 Data Acquisition Hardware

Data acquisition is achieved by using USB-4716 DAQ Card which is manufactured by Advantech. USB-4716 DAQ Card use USB (Universal Serial Bus) to connect to computer. The Advantech Data Acquisition driver is designed to support programming language such as Visual Basic. Figure 3.1 shows the data acquisition card that used in this project. This daq card has analog input, analog output, digital input and digital output channel.

Figure 3.12: USB-4716 DAQ Card

3.3.1 Analog input

Analog input subsystems convert real-world analog input signals from a sensor into bits that can be read by your computer. USB-4716 is 16bits resolution and analog input subsystems are also referred to as AI subsystems

3.3.2 Analog output

Analog output subsystems convert digital data stored on your computer to a real-world analog signal. These subsystems perform the inverse conversion of analog input subsystems. USB-4716 offers two output channels with 16 bits of resolution,

with special hardware available to support multiple channel analog output operations. Analog output subsystems are also referred to as AO subsystems.

3.3.3 Digital input/output

Digital input/output (DIO) subsystems are designed to input and output digital values (0 and 1) from and to hardware. These values are typically handled either as single bits or lines, or as a port, which typically consists of eight lines.
USB-4716 provides 8 channel digital inputs and 8 channel digital outputs.

3.3.4 Counter

Counter/timer (C/T) subsystems are used for event counting, frequency and period measurement, and pulse train generation. USB-4716 has 1 channel for event counter.

3.3.5 I/O Connectors

USB-4716 is equipped with plug-in screw-terminal connectors that facilitate the connection to the module without cables or terminal boards.

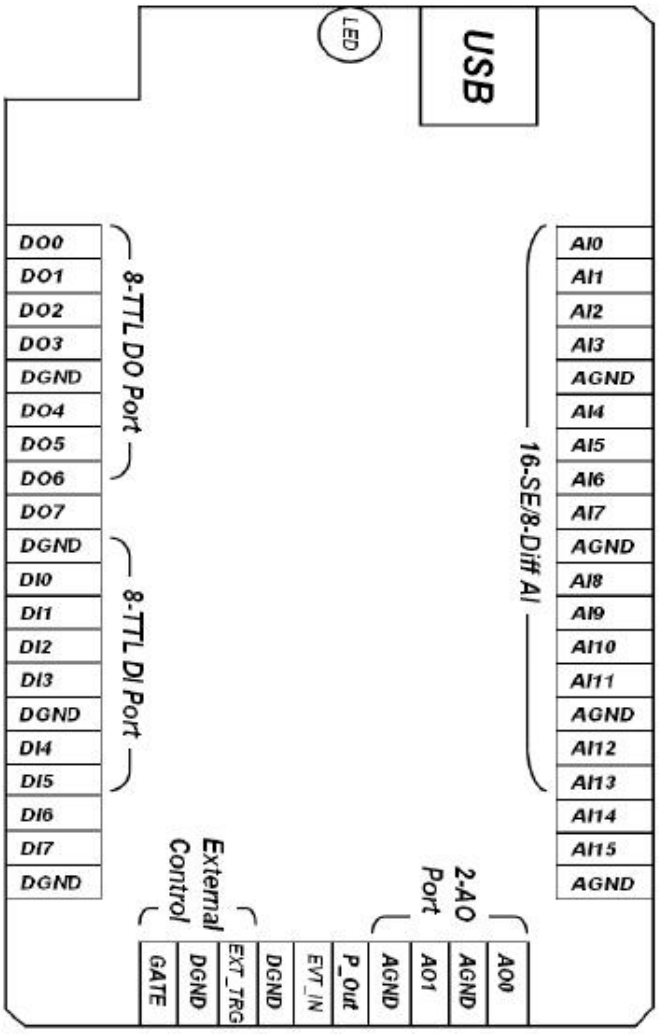

Figure 3.13

Table 3.1: I/O Connector Signal Description

Signal Name	Name Reference	Direction	Description
AI<0…15>	AGND	Input	Analog Input Channels 0 through 15.
AIGND	-	-	Analog Input Ground.
AO0 AO1	AGND	Output	Analog Output Channels 0/1.
AOGND	-	-	Analog Output Ground. The analog output voltages are referenced to these nodes.
DI<0..7>	DGND	Input	Digital Input channels.
DO<0..7>	DGND	Output	Digital Output channels.
DGND	-	-	Digital Ground. This pin supplies the reference for the digital

			Channels at the I/O connector.
GATE	DGND	Input	A/D External Trigger Gate. When GATE is connected to +5 V, it will disable the external Trigger signal to input.
EXT _TRG	DGND	Input	A/D External Trigger. This pin is external trigger signal input for the A/D conversion. A low-tohigh edge triggers A/D conversion to start.
EVT_IN	DGND	Input	External events input channel.
P_OUT	DGND	Output	Pulse output channel

3.3.6 Noise

Noise must be considered before using the DAQ card because DAQ card is very sensitive to noise. Noise can be divided into two components. The first one is internal noise and the second is external noise.

Internal noise arises from thermal effects in the amplifier. Usually, amplifier generates a few microvolt of internal noise. The amount of noise added to the signal depends on the bandwidth of the input amplifier. To reduce internal noise, an amplifier with a bandwidth that closely matches the bandwidth of the input signal must be selected.

External noise arises from many sources. A common external noise is fluorescent lightning. These lights generate an arc at twice the power of line frequency (120Hz). The noise is added to the acquisition circuit because every wire in the circuit acts as aerials picking up environmental electrical activity. To remove this noise the input channel should be configured in differential mode. Beside that, the signal wire must be twisted together rather than separate. And the signal wire must be kept as short as possible and far away from environmental electrical activity.

3.3.7 Input Configuration

Input configuration can be divided into 2 categories. First, single-ended input and second differential input.

When single-ended input is used, there is one signal wire associated with each input signal, and each input signal is connected to the same ground. This type of input configuration is more susceptible to noise than the differential measurements because of differences in signal paths. It is advised to use this configuration when the input signal is greater than 1 volt and the wire connecting the signal is are less than 10 feet.

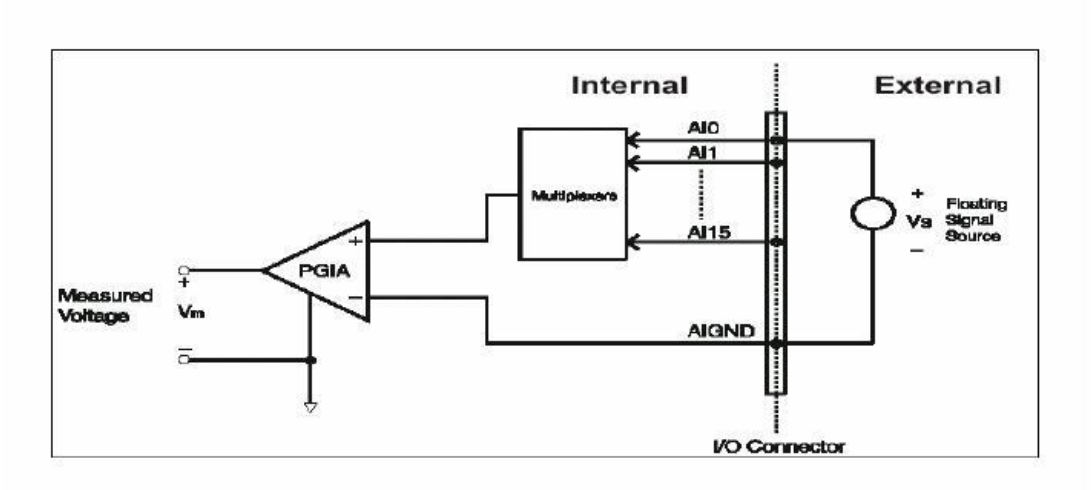

Figure 3.13: Single-ended input connection

When differential input is configured, there are two signal wires associated with each input signal, one for input signal and one for reference signal. The measurement is the difference between the two wires. This configuration type helps reduce noise. It is recommends to use this configuration when the input signal is less than 1 Volt, the wires connecting the signal are greater than 10 feet and the signal wires travel through noisy environment.

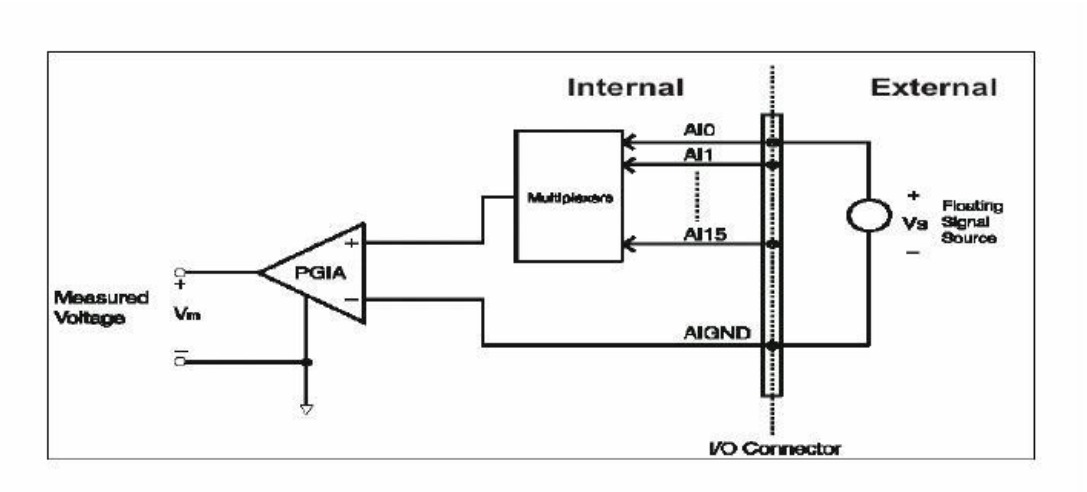

Figure 3.14: Differential input channel connection - ground reference signal source

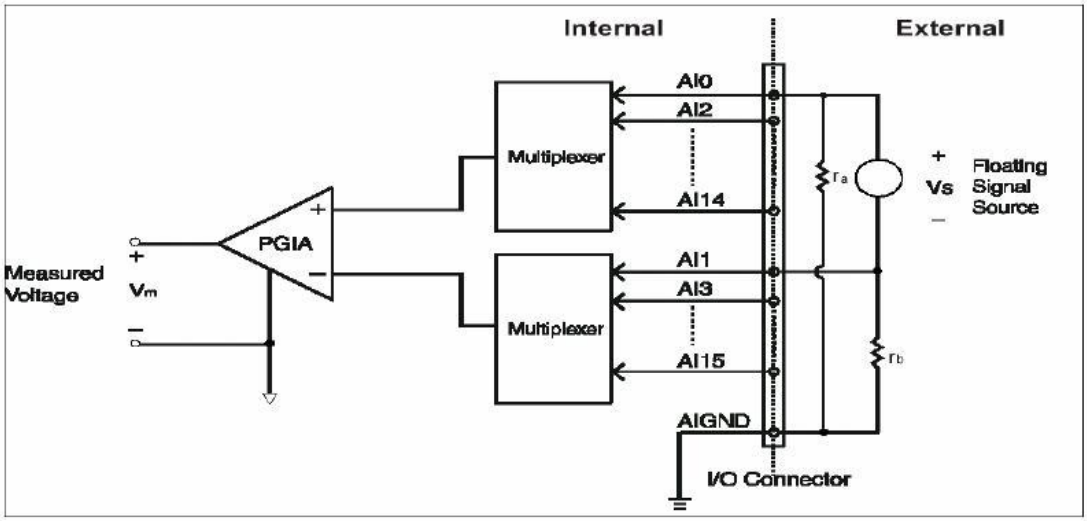

Figure 3.15: Differential input channel connection - floating signal source

CHAPTER 4

SOFTWARE

4.1 Software Development

Microsoft Visual Basic 2008 Express Edition is chosen because it is free software and hosted a VB.Net platform. In addition VB.Net is the most efficient programming language and consumes low CPU utilization compare to other high level programming languages.

4.1.1 Device Driver Installation

Before start programming using Visual Basic, device driver for DAQ USB-4716 must be installed. If not, Visual Basic won't detect the DAQ USB-4716 hardware. To install the device driver the following steps are required.

i) Insert the DAQ CD.
ii) Install Device Manager.
iii) Install USB-4716 driver.

4.1.2 General Software Flow Cart

The flow chart below shows the basic construction of the software

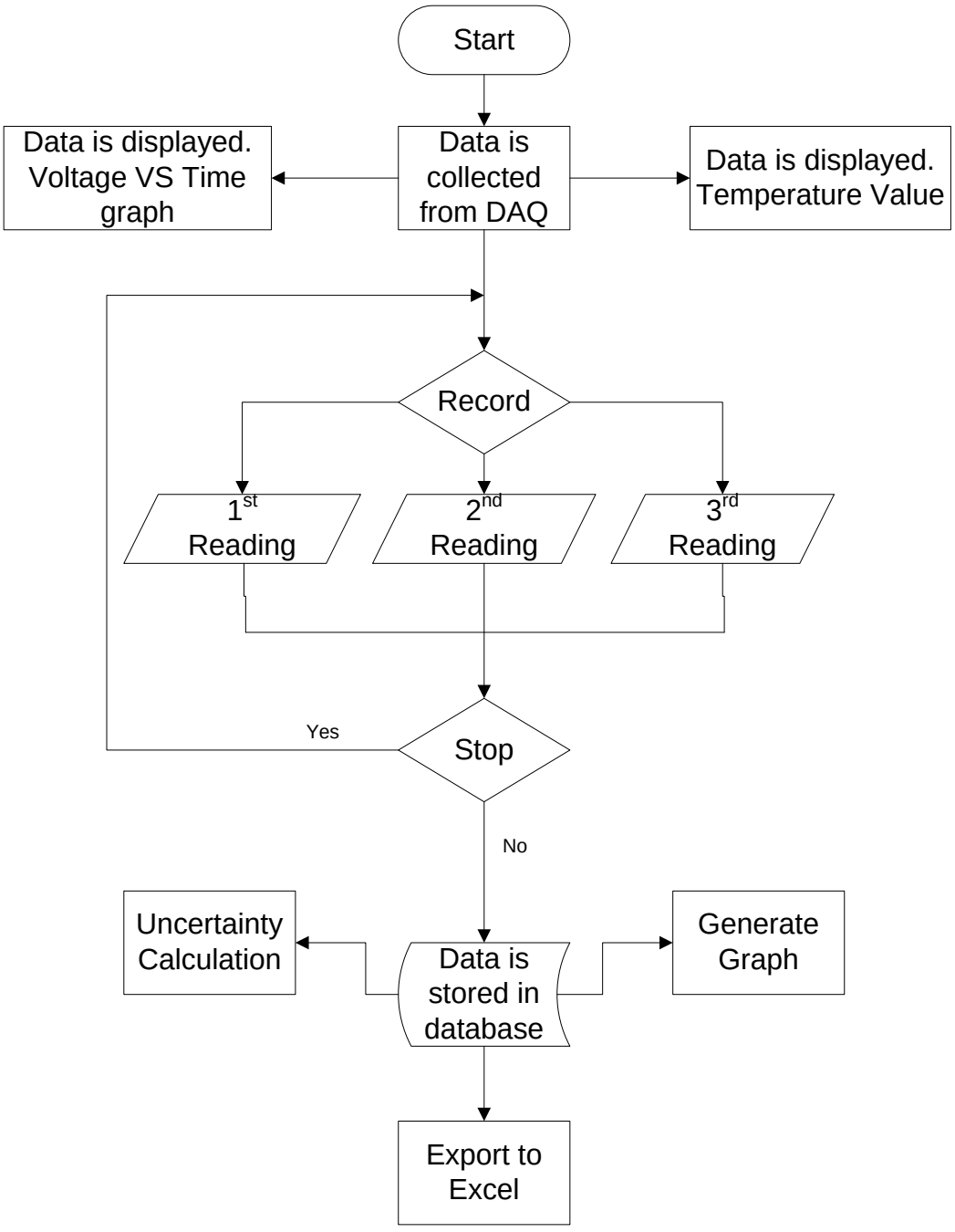

Figure 4.1: General Flowchart

4.2 Creating Graphical User Interface (GUI)

This project used Object oriented-programming (OOP) in order to implement the Data Acquisition. OOP is the process of developing well-defined design model [4]. Visual Basic is one of the languages supported by the Advantech DAQ USB-4716.

Before the USB-DAQ can be communicate with the application software, all dll(Dynamic Link Library) driver is installed in every applications. The dll driver acts as reference for VB software. The dll driver is provided in the installation CD.

The software used Microsoft Access database as data storage. To access data that has been saved, user can directly open Microsoft Access and make any changes of the data. When user opens the software, VB will automatically convert Microsoft Access database to dataset. Dataset is an internal database for VB.

To display graph, ZedGraph is used because it is more advance compared to other VB components. ZedGraph is a set of classes, written in C#, for creating 2D line and bar graphs of arbitrary datasets. The classes provide a high degree of flexibility -- almost every aspect of the graph can be user-modified [5].
In order to use ZedGraph class, user must add dll reference and component for VB.

4.2.1 Uncertainty calculation

Uncertainty calculation is calculated using formula:

$$s(x_k) = \sqrt{\frac{1}{(n-1)}\sum(x_k - \bar{x})^2}$$

To apply this formula in programming language for loop is used. First, all data is extracted from database and located in variable form. Then, the formula is applied in each row. To continue calculate the next row, for loop is used.

```
For i = 0 To MaxRows - 1

            a = ds.Tables("proto").Rows(inc).Item(3)
            b = ds.Tables("proto").Rows(inc).Item(4)
            c = ds.Tables("proto").Rows(inc).Item(5)
            ave = (a + b + c) / 3
            sig = ((a - ave) * (a - ave)) + ((b - ave) * (b - ave))
+ ((c - ave) * (c - ave))
            sq = Math.Sqrt(0.5)
            std = sq * sig

            stdx(i) = sq * sig
            y = Math.Round(stdx(i), 10)
            inc = inc + 1

Next i
```

To calculate uncertainty u1, each standard deviation is put in array form, then each array value is compared with each other. After the comparison complete, the highhest and the lowest value is located in variable high grade and low grade.

```
            highgrade = stdx(0)
            lowgrade = stdx(0)

        For i = 1 To 5
            If stdx(i) < lowgrade Then
                lowgrade = stdx(i)
            End If
        Next i

        For i = 1 To 5
            If stdx(i) > highgrade Then
                highgrade = stdx(i)
            End If
        Next i
```

4.2.2 Sample rate / Frequency sampling

Sample rate formula is calculate using this formula :

$$S = \frac{1}{time\ interval}$$

This formula is applied using timer control in VB. A timer is another one of those useful controls within VB that allows the developer to make things happen after a certain time, or on a certain event. Time in VB is calculated in mili seconds. Let say time interval is 500ms. We will divide 1 second over timer interval to get the sample rate.

$$S = \frac{1}{5m}$$

$$S = 200\ Sample/second$$

$$S = 200\ Hz$$

In order to get a smooth graph, a higher sample rate or lowest time interval is needed.

4.2.3 Temperature Calibration

Temperature calculation is achieved using formula

$$y = mx + c$$

y = Temperature in celcius

x = input voltage from daq card

m = gradient

c = intersection at y axis

With this formula we are able to determine the exact temperature value by calibrating the temperature and the voltage. Voltage value is used rather than current because DAQ card can only read voltage. To calculate the current value, formula V=IR is used and as we know from the formula the voltage increase proportional

with current. So when the current increase, the voltage also increase. In this calculation we used a fix 250Ω resistor.

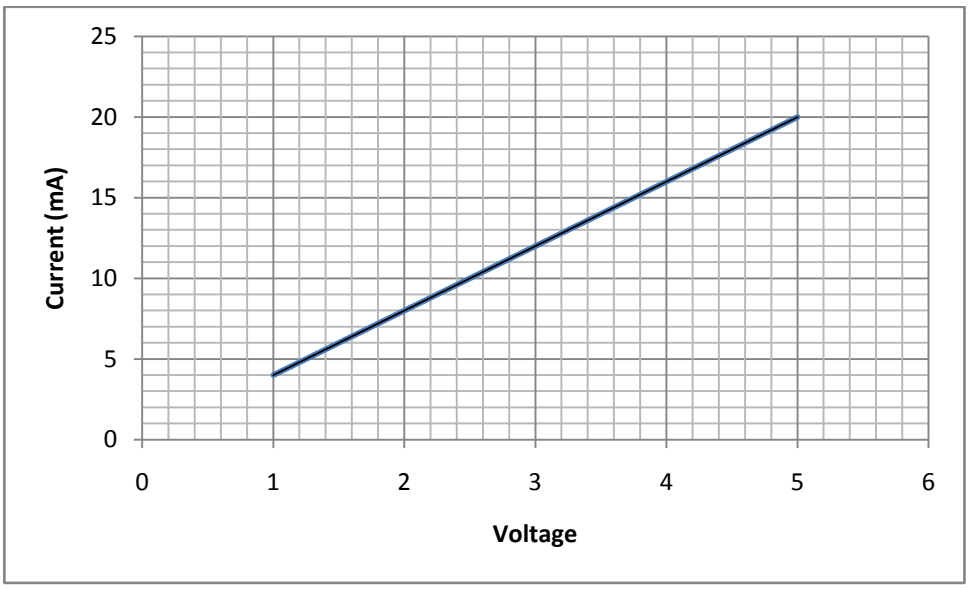

Figure 4.2: Current Vs Voltage

4.2.4 Trigger modes

A trigger is defined as an event that initiates data logging to memory or to a disk file. Triger event is important as it will determine the frequency sampling. Some trigger mode might not give a maximum sampling frequency because it will depends on other factors. Trigger can be divided into 3 parts which is software trigger, internal pacer trigger and external trigger.

This software use software trigger mode. User can set the sampling frequency by adjusting the Time interval value and the value is in miliseconds unit.

```
Timer1.Interval = 500
```

Software trigger is less accurate to get the sampling frequency. The sampling frequency will depends on the computer hardware arcitechture and software environtment. In addition, the sampling frequency may vary due to programming

language, code efficiency and CPU utilization. For instance, the sampling frequency may reduce if the CPU utilization is high while you logging the data. To maximize the frequency sampling, close all unnessary apllication when logging the data and try to increase the code efficiency.

Internal pacer trigger also known as onboard trigger or hardware trigger. This trigger mode is the most accurate trigger mode to get the maximum sampling frequency. In hardware trigger mode, the hardware clock and hardware memory will be used. All dat a will be transferred to software when the hardware memory is full. USB-4716 DAQ card support to 200k sample per second (200k HZ). This sampling frequency can only be achieved if hardware trigger mode is used.

External trigger used an external clock to trigger the events to initiates data logging to memory. This mode usually used when involving external hardware and needs a synchronous sampling with the harware.

4.3 Connecting USB-4716 DAQ with computer.

Step 1: Touch the metal part of the surface of your computer to neutralize the static electricity that might be in your body.
Step 2: Plug the USB modul into the selected USB Port.
Step 3: Goto Advantech Device Manager and see wether the device is connected or not.
Windows > All Programs > Advantech Automation > Device Manger > Advantech Device Manager

Figure 4.3: Advantech Form

Step 4: Press the test button on Advantech Device Manager. Now you should have Advantech Device Test window. If the DAQ card is connected with voltage source, you should see the analog input voltage reading.

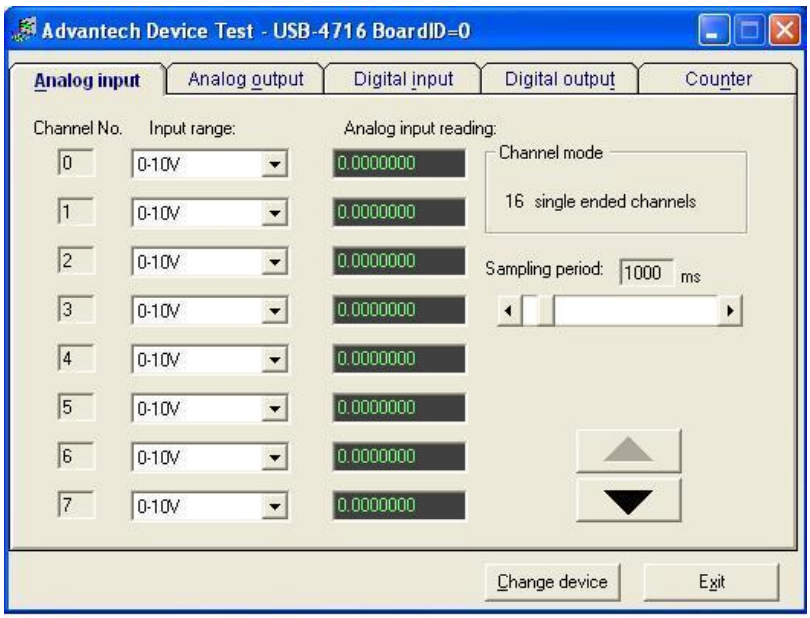

Figure 4.4: Advantech Device Test

4.4 GUI interface

The GUI user interface has been developed using Microsoft Visual Basic 2008 Express Edition. The GUI has been designed so that it becomes "user friendly" as shown in figure below. In this GUI, there have four section or tab.

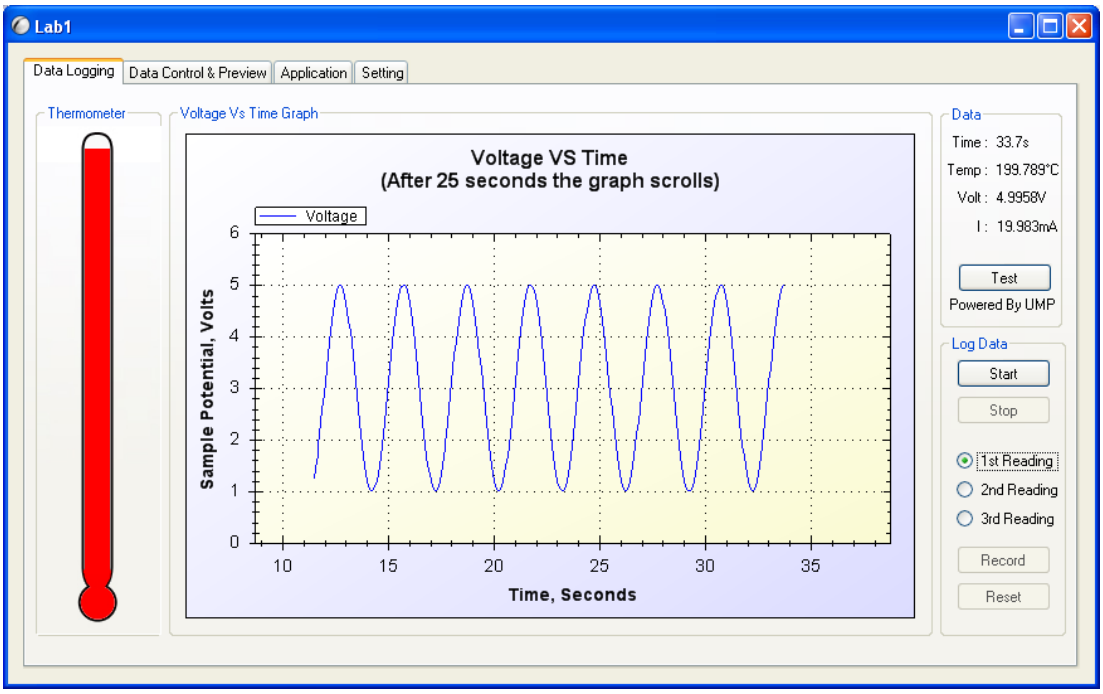

Figure 4.5: Data logging tab

The first tab shows Temperature VS Time graph. If the computer is connected to the USB DAQ card, the user can view live data from the instrument. To start viewing the data stream, user must press start button. Before start logging the data, user must select which reading he likes to record into database, 1st, 2nd or 3rd reading. The graph only shows Voltage VS Time data, but the thermometer panel will show the equivalent value of temperature, for instance 1mV equal 50°C. And the Voltage VS Time graph data is not stored in database means that the value is not recorded. In order to start recording data from instrument, user must press record button. User must be alert when he wants to record the data because once he presses the button he cannot get the previous data at the specific time he wanted. This happen because, as the time increases, the temperature also increases. Obviously, we cannot stop the temperature from

rising and temperature reading is not stable. If the data is recorded earlier than it should be, it can still be deleted.

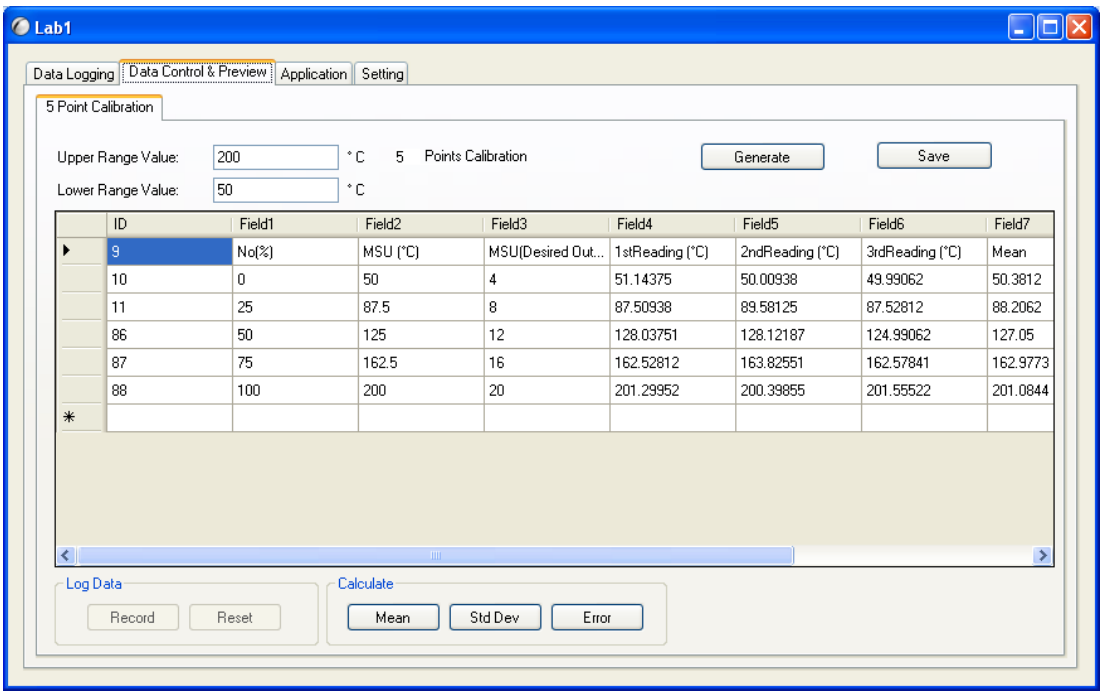

Figure 4.6: Data control & Preview tab

The second tab shows data that has been recorded. If no data has been recorded yet, the software will be loaded with default data from database. From this tab, user can scroll through recorded data, add new data, update data or delete data. After enough data has been recorded, user can calculate the mean, standard deviation and error. All result will be calculated automatically by the software.

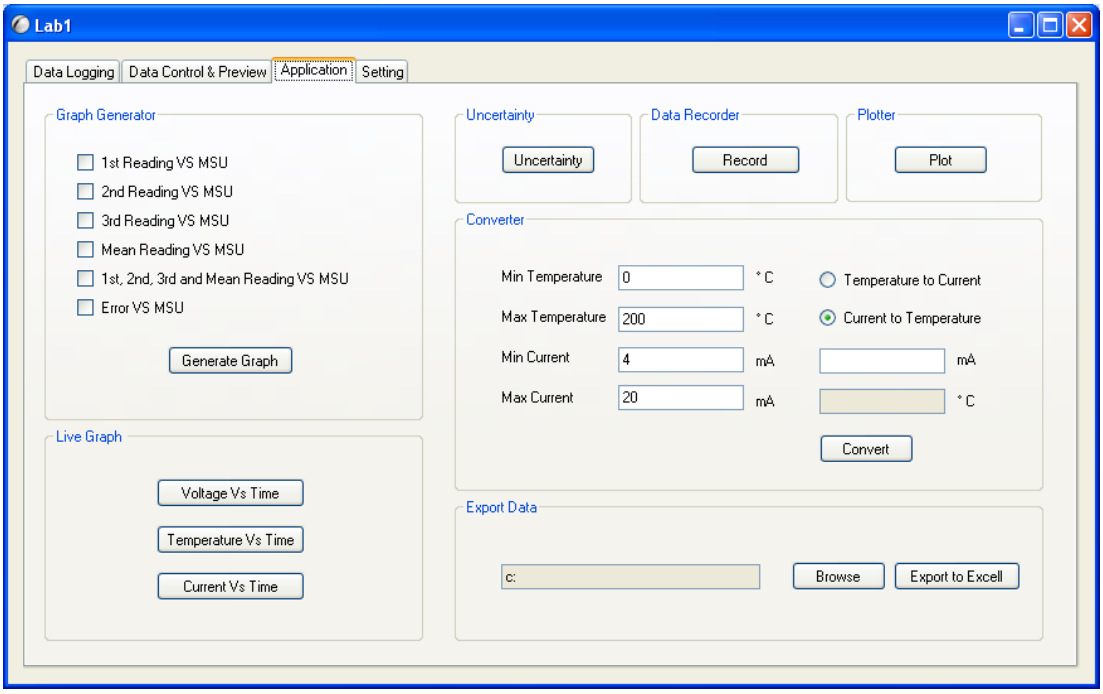

Figure 4.7: Application tab

In application tab, user can generate several of graphs such as 1st reading VS MSU, Mean Reading VS MSU and Error VS MSU. The graph can be saved in several types of format such as gif, jpeg, png, bmp, emf and tif. Zooming option also included in the graph properties. With this interface also, the displayed graph can be printed. Uncertainty calculation is generated automatically by this software when data is loaded, but if certain data is not available for analysis the software will assume 0 for the data field. For instance, if user still not records the 3rd reading, the software will assume the value for 3rd reading is 0. Data Recorder function is used to record live data and the recorded data is stored in separate database. After the data has been saved, the data can be plotted with plotter function. Converter function is used to convert value from temperature to current or current to temperature. In addition, this software has the function to export all data into Microsoft Excel format.

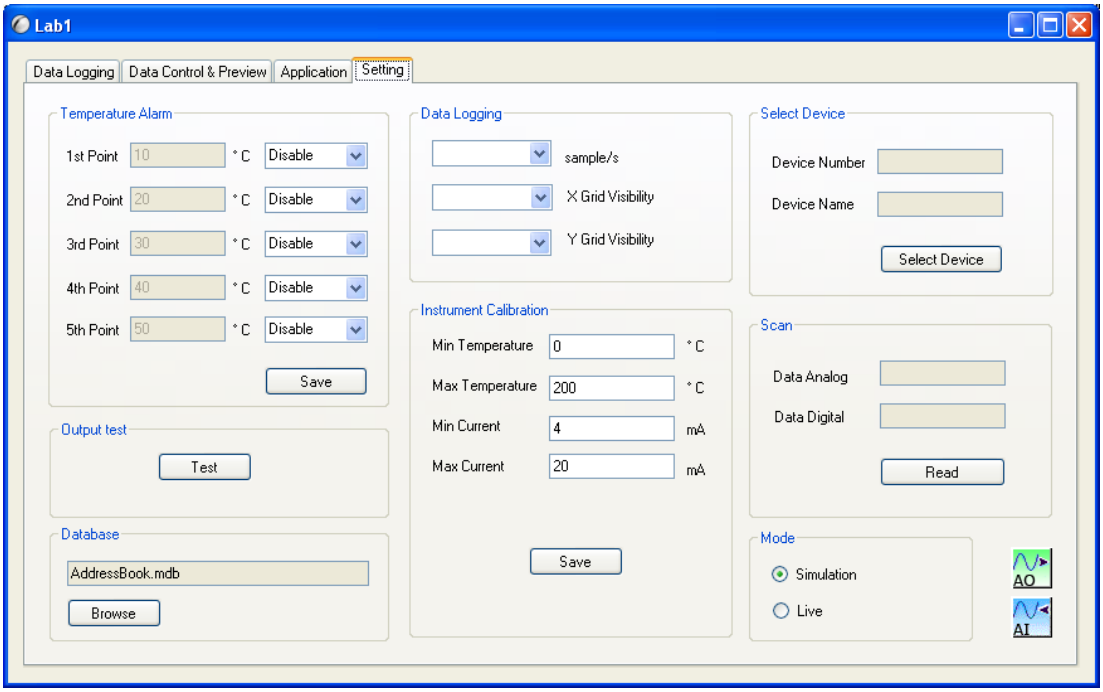

Figure 4.8: Setting tab

In setting tab, user can change setting or properties of the software. User can change how many data to log per second in data logging panel. The higher the value we choose, the smother the graph will be. The maximum value that can be recorded by this software is 32 sample/s. User can also set the temperature alarm. A message will be prompt when the desired temperatures reach. There are points that can be set by user and each point can be disable depends on the user needs.

Before any experiment can be made, user needs to fill the exact instrument calibration data in the Instrument Calibration panel. If not, the temperature value will not be accurate.

Select device panel is necessary for software to connect with hardware and the scan panel is used to determine the current data from DAQ card in analog and digital form.

This software is built in with the function to load external database. The database must be in Microsoft Access format.

4.5 General Procedure How to Use the Software

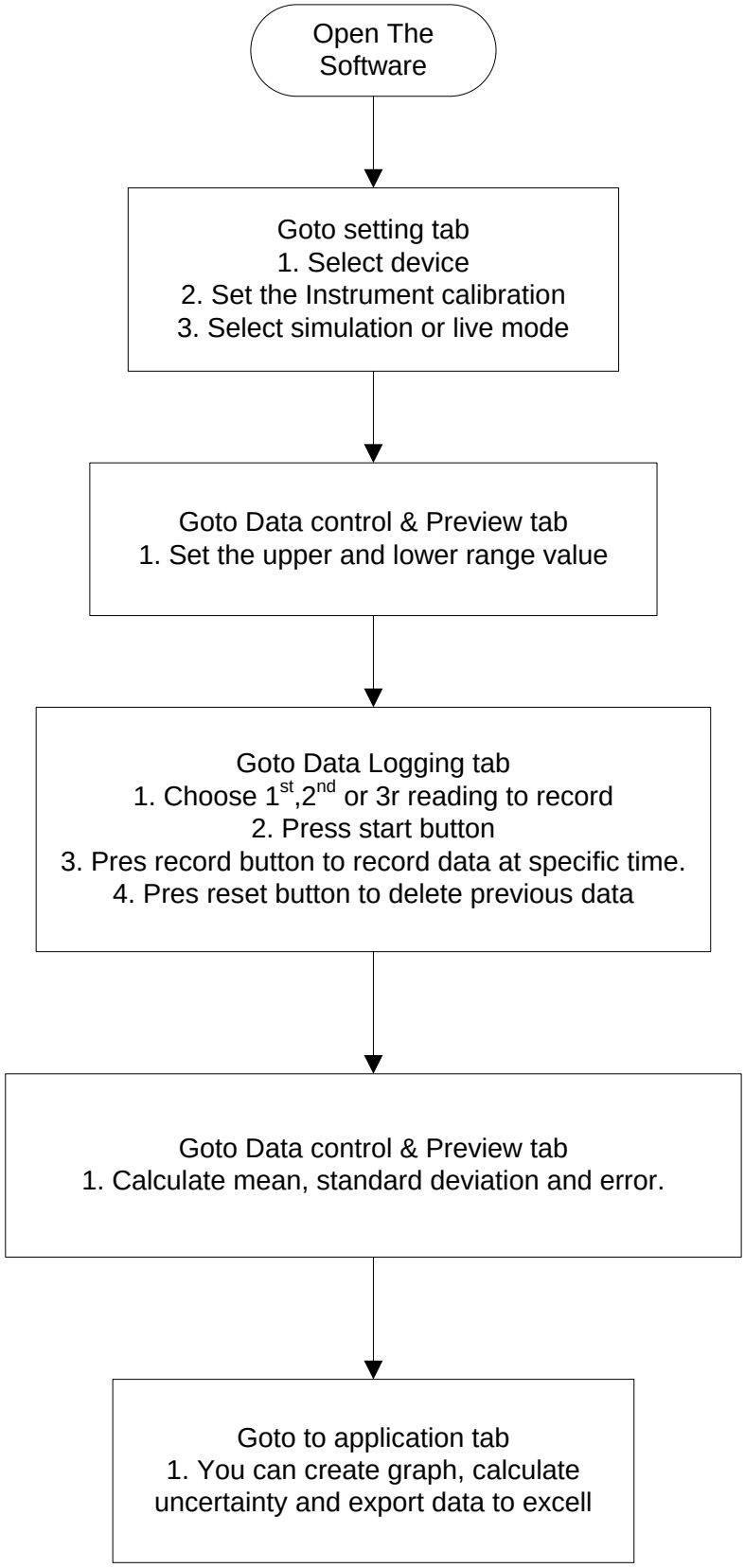

Figure 4.9: General Procedure How to Use the Software

CHAPTER 5

RESULT AND ANALYSIS

5.1 Introduction.

The main objective of this project is to develop a software application to help in student learning process and the software will be used during lab session of Industrial Instrumentation subject (BEE4523). This software has successfully developed using Microsoft Visual Basic 2008 Express Edition (VB) and it can be concluded that VB is a simple programming language to develop a user friendly application.

5.2 Experiment

5.2.1 Experiment 1: Five point calibration of temperature transmitter

For the Five-point calibration of the instruments, the span of the UUT is divided into five equal parts with the first point at the low range and the top pint at the high range. For example the temperature transmitter has the range 50°C -200°C. Therefore the span is 200-50=150°C. Dividing the span by four we get 37.5°C. Hence the five equal points are 50, 87.5, 125.0, 162.5 and 200°C based. The desired output for 4-20mA range is calculated based on the 50 -200°C ranges using below equation;

$$Desired\ output: \frac{x}{100}(URV)LRV + LRV$$

Where;

X = ith point

URV = Upper range value

LRV = Lower range value

Objectives: Determine the calibration of temperature transmitter.

Figure 5.0 shows the connection for this experiment:

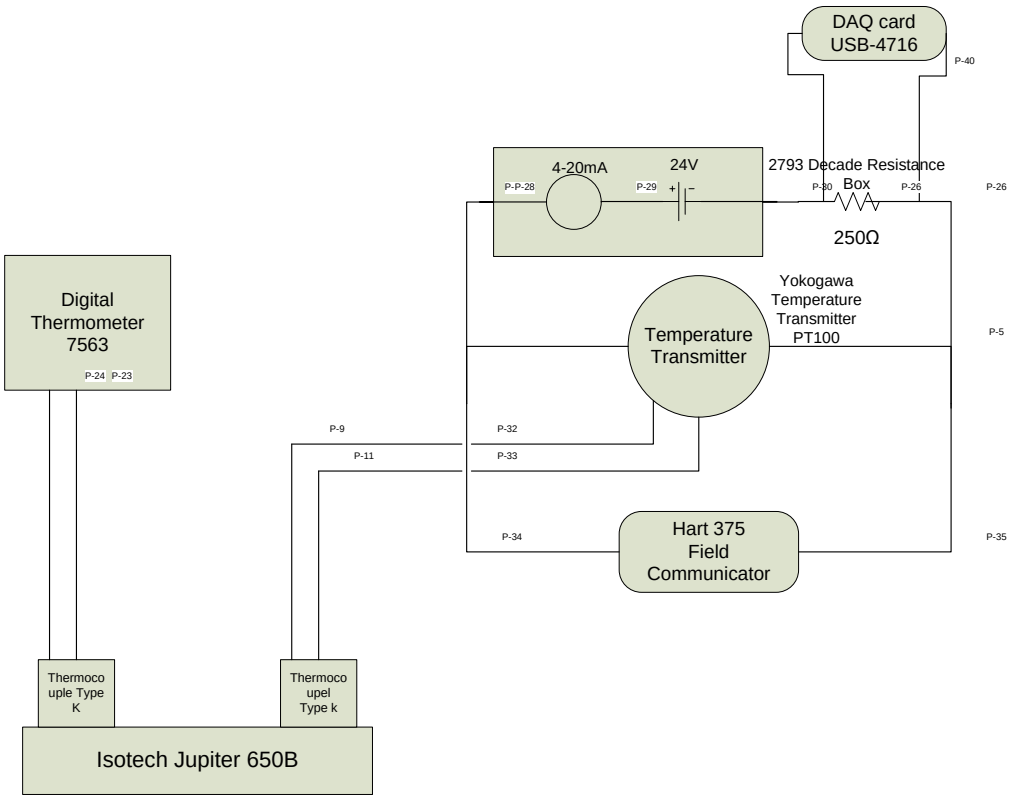

Figure 5.0: Connection for Experiment 1

Procedures:

1) The equipment is connected as shown in Figure 5.0.
2) The desired temperature range is calibrated using Hart Communicator. In this experiment the lowest value is set to 50°C and the highest value to 200°C.
3) USB DAQ card is connected parallel to decade resistance box. Once the DAQ card has been connected, the software for this experiment is activated and the calibration value is inserted in software's setting.

4) To record the desired value using the software, make sure the input is stable. In order to get a stable input, the Hart 375 Field Communicator must be removed from the circuit during the experiment. If the input is still not stable, try to reconnect the DAQ card with decade resistance box.
5) When the desired value has reached, the button is pressed immediately. The experiment is continued till the maximum value.
6) Three readings are recorded using this software.
7) The mean, standard deviation and error curve is automatically generated by the software.

Results:

Results for experiment 1 was put in Table 5.1 (a) & (b) and plotted in Figure 5.2.

Table 5.1 (a): Result from experiment

No(%)	MSU (°C)	MSU(Output mA)	1stReading (°C)	2ndReading (°C)	3rdReading (°C)
0	50	4	51.14375	50.00938	49.99062
25	87.5	8	87.50938	89.58125	87.52812
50	125	12	128.03751	128.12187	124.99062
75	162.5	16	162.52812	163.82551	162.57841
100	200	20	201.29952	200.39855	201.55522

Table 5.1 (b): Mean, Standard Deviation and Error

Mean	Std	Error
50.3812	0.6168	0.7625
88.2062	2.0054	0.8071
127.05	4.5008	1.64
162.9773	0.7639	0.2938
201.0844	0.5221	0.5422

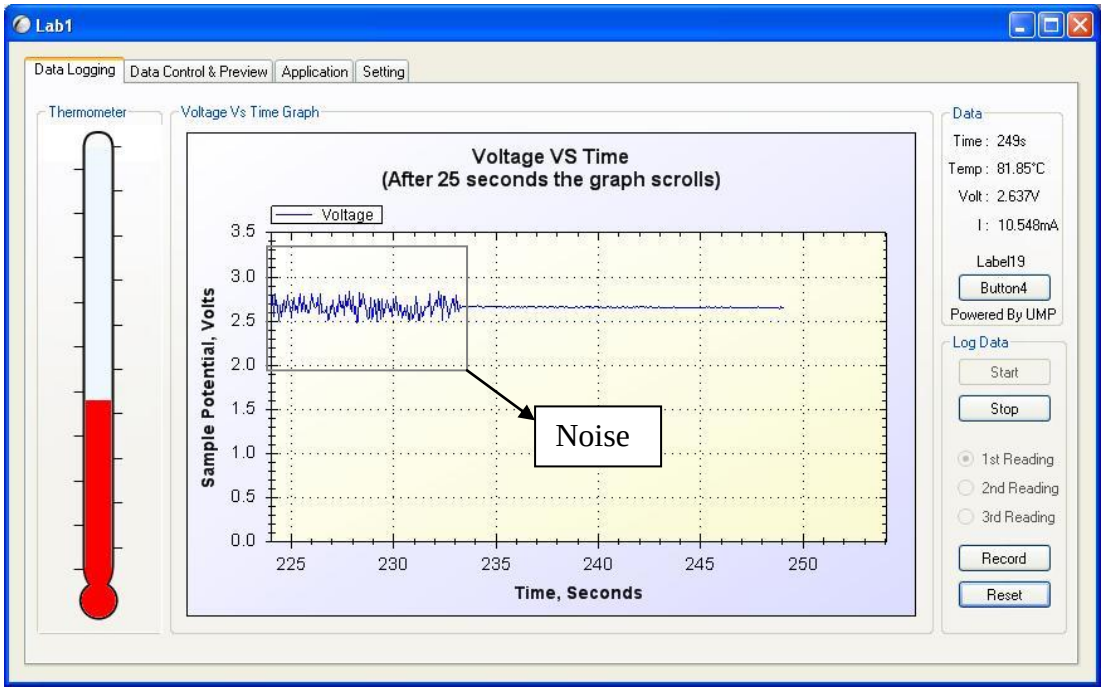

Figure 5.1: Temperature VS MSU Applied value

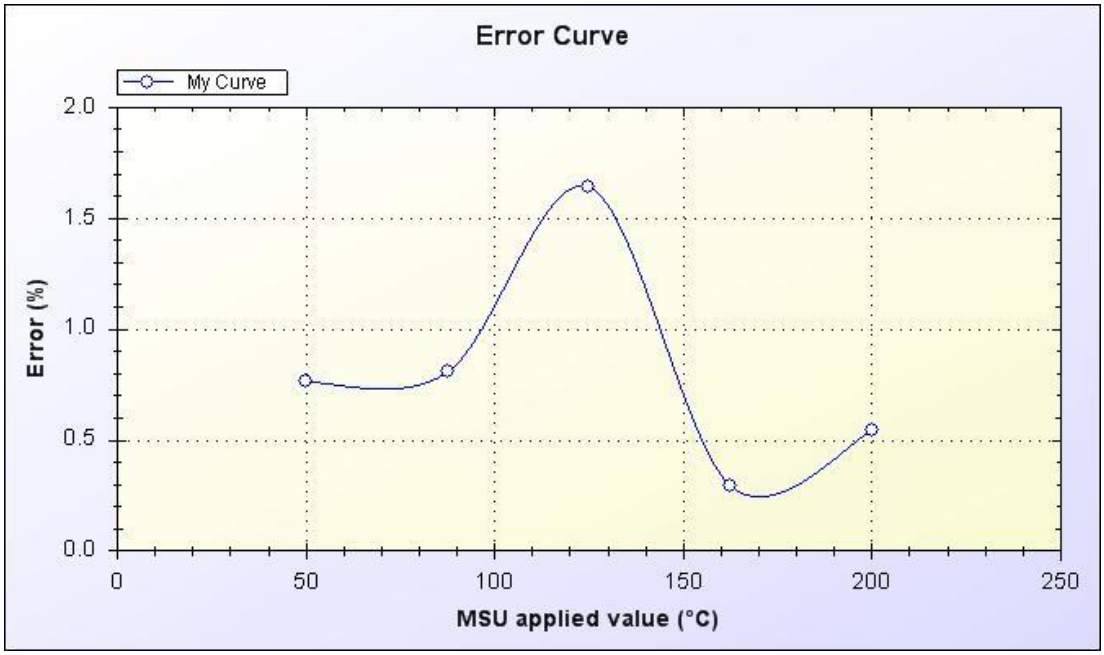

Figure 5.2: Error Curve

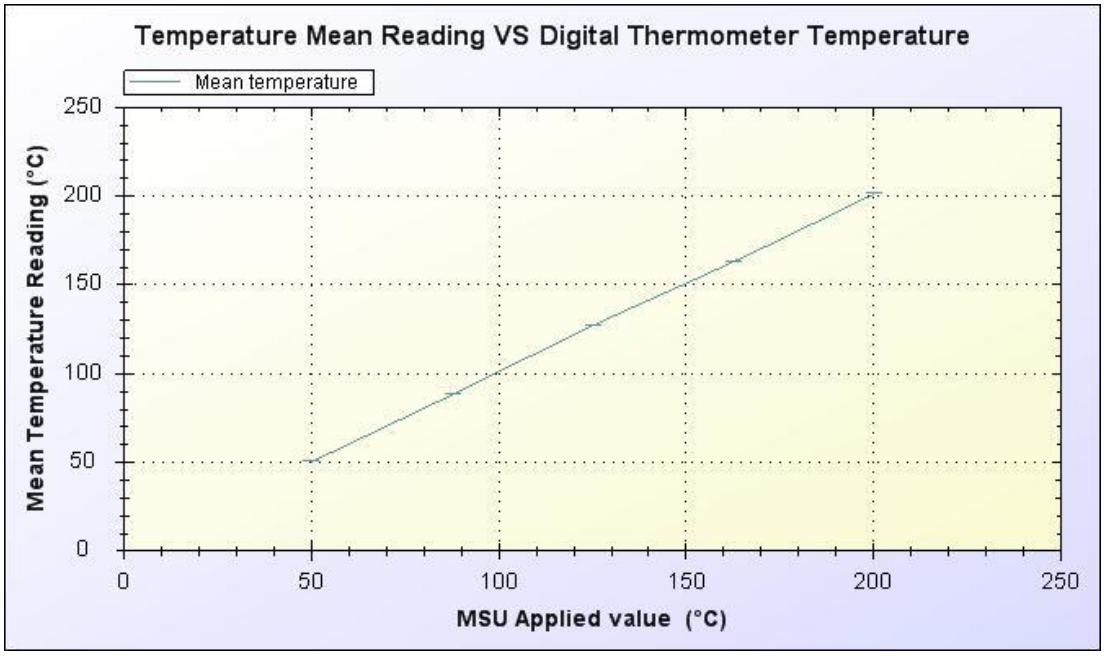

Figure 5.3: Mean VS Digital Thermometer Temperature

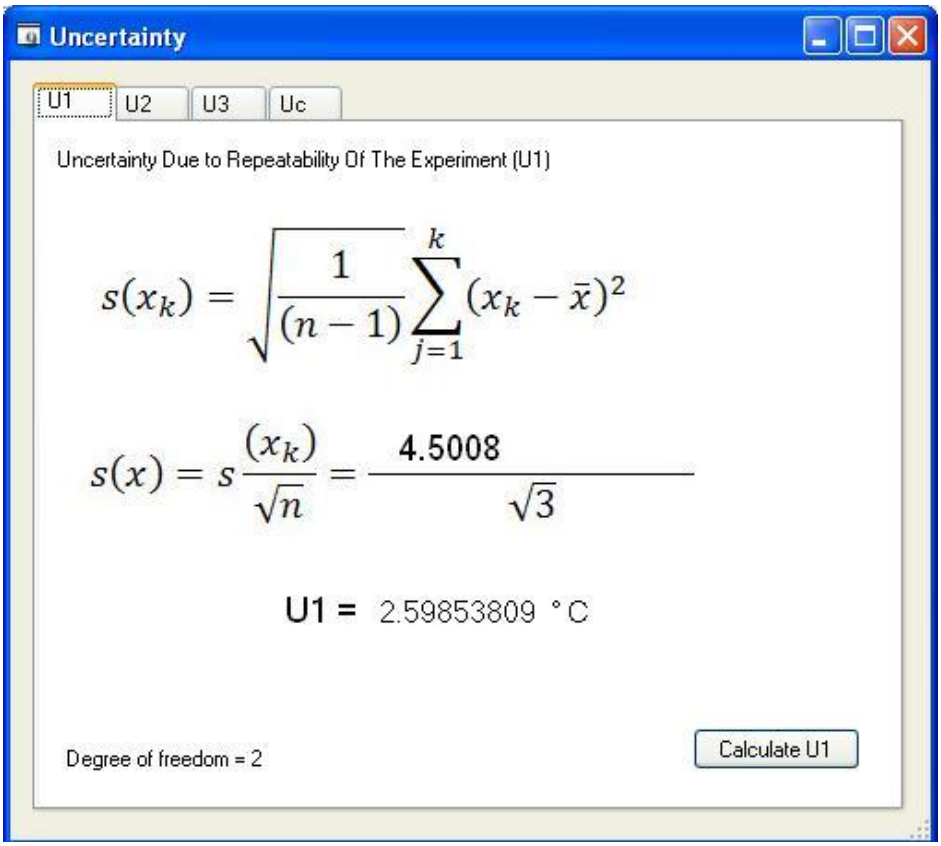

Figure 5.4: Uncertainty due to Repeatability of the Experiment (U_1)

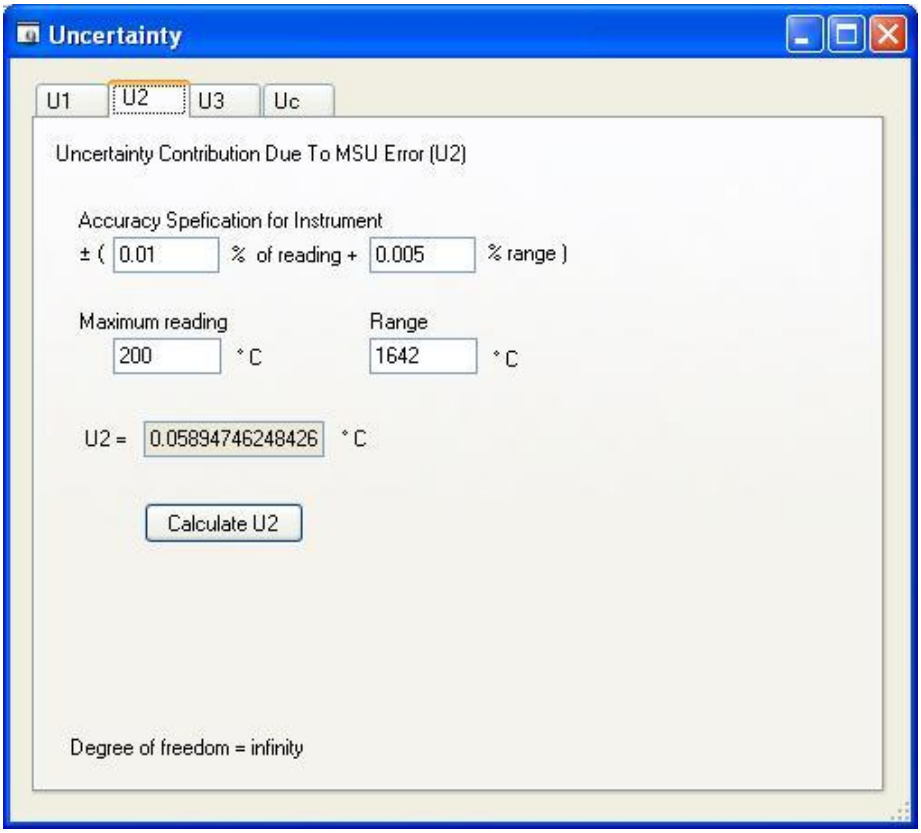

Figure 5.5: Uncertainty Contribution due to MSU Error (U_2)

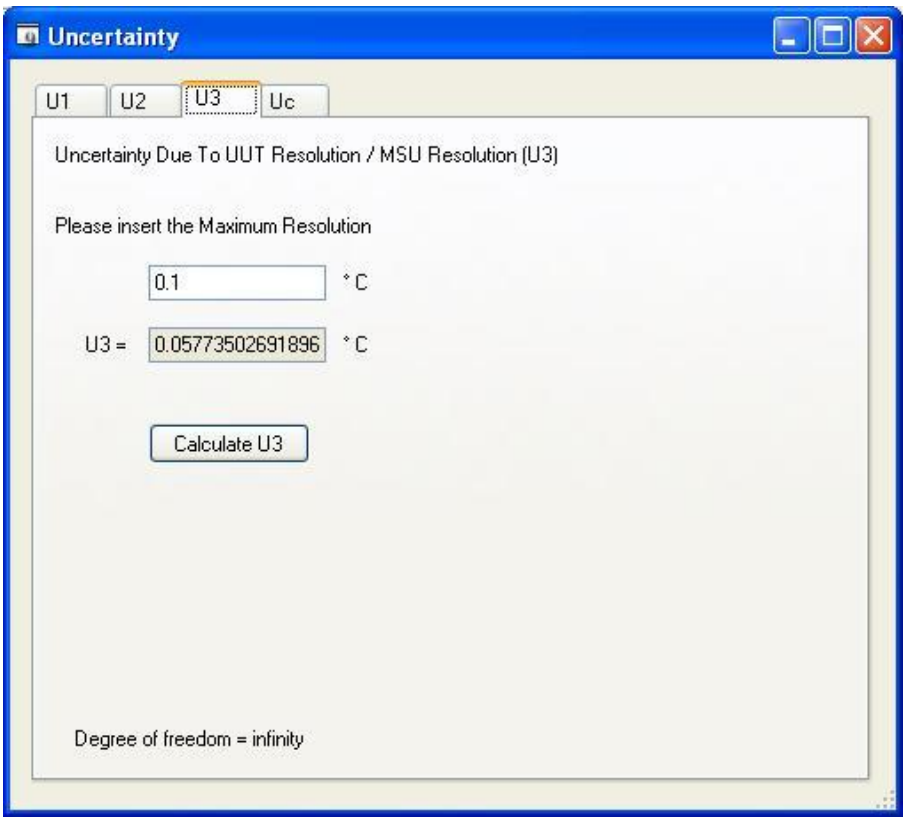

Figure 5.6: Uncertainty Due to UUT Resolution (U_3)

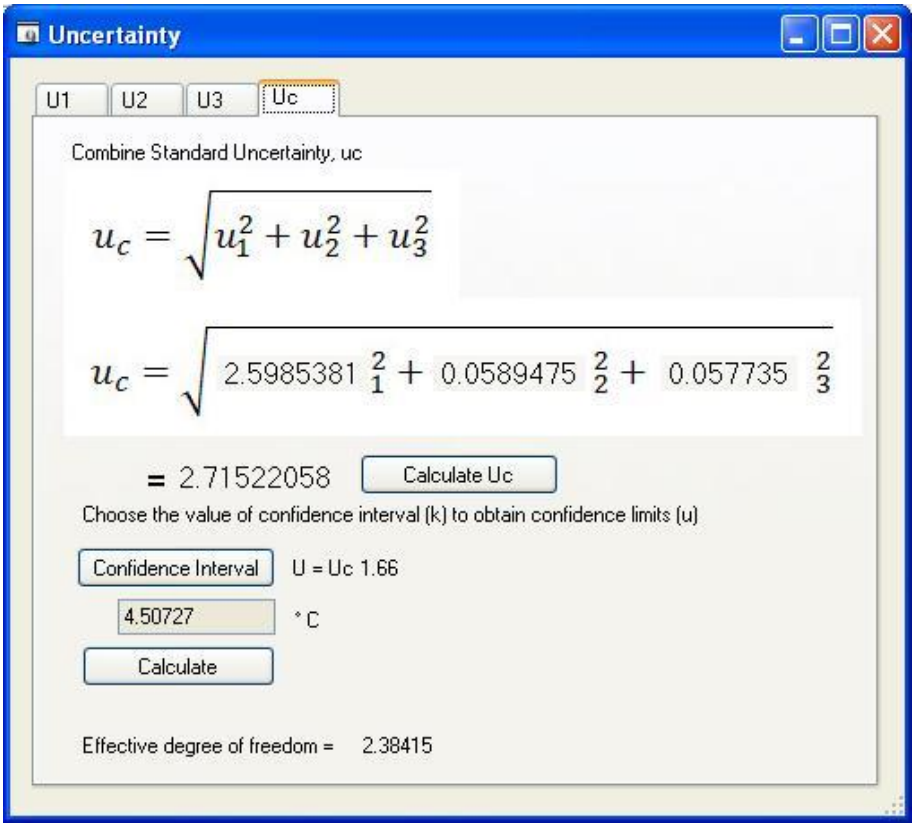

Figure 5.7: Combined Standard Uncertainty (U_c)

Calculation:

1) The error calculation:

$$error = \frac{Mean - Desired}{Desired} \times 100$$

$$error = \frac{50.3812°C - 50°C}{50°C} \times 100$$

$$error = 0.7624$$

2) Mean

$$\overline{x} = \frac{1st\ run + 2nd\ run + 3rd\ run}{3}$$

$$\overline{x} = \frac{51.14375°C + 50.00938°C + 49.99062°C}{3}$$

$$\overline{x} = 50.3812°C$$

3) Uncertainty

$$S(x_k) = \sqrt{\frac{1}{(n-1)}} \sum_{j=1}^{k} (x_k - \overline{x})^2$$

$$S(x_k) = \sqrt{\frac{1}{(3-1)}} (51.14375°C - 50.3812°C)^2 + (50.00938°C - 50.3812°C)^2 + (49.99062°C - 50.3812°C)^2$$

$$S(x_k) = 0.6168$$

4) Uncertainty due to Repeatability of the Experiment (u_1).

$$S(x_k) = \frac{Highest\ standard\ value}{\sqrt{n}}$$

$$S(x_k) = \frac{4.5008}{\sqrt{3}}$$

$$S(x_k) = 2.59853809°C$$

$$u_1 = 2.59853809°C$$

$$Degree\ of\ freedom = \gamma_1 = 3 - 1 = 2$$

5) Uncertainty Contribution Due to MSU Error (u_2).

Accuracy specification for this instrument is

$$\pm(0.01\%\ of\ reading + 0.005\%\ range)$$

Hence the error in MSU is

$$MSU = \pm(0.01\%\ of\ reading + 0.005\%\ range)$$

$$MSU = \pm(0.01\% \times 200 + 0.005\% \times 1642)$$

$$MSU = \pm 0.1021°C$$

$$u_2 = \frac{MSU}{\sqrt{3}}$$
$$u_2 = \frac{0.058948}{\sqrt{3}}$$
$$u_2 = 0.058948°C$$

$$Degree\ of\ freedom = \gamma_2 = \infty$$

Degree of freedom is infinity since the manufacturer is expected to provide the error data after a large number of tests.

6) Uncertainty due to UUT resolution (u_3).

From the user manual we can get the maximum resolution for MSU when using thermocouple is 0.1°C.

$$u_3 = \frac{0.1°C}{\sqrt{3}}$$
$$u_3 = 0.057735°C$$

$$Degree\ of\ freedom = \gamma_3 = \infty$$

Degree of freedom is infinity since the manufacturer is expected to provide the error data after a large number of tests.

7) Combined Standard Uncertainty (u_c).

$$u_c = \sqrt{(u_1^2 + u_2^2 + u_3^2)}$$
$$u_c = \sqrt{(2.59853809^2 + 0.058948^2 + 0.057735^2)}$$
$$u_c = 2.71522$$

The effective degrees of freedom v_e is given by

$$v_e = \frac{u_c^4}{\frac{u_1^4}{v_1} + \frac{u_2^4}{v_2} + \frac{u_3^4}{v_3}}$$

$$v_e = \frac{2.71522^4}{\frac{2.59853809^4}{2} + \frac{0.058948^4}{\infty} + \frac{0.057735^4}{\infty}}$$

$$v_e = 2.38415$$

The confidence limits are obtained by the formula. We choose the coverage factor k from table. Refer to Appendix A, for a value of v=2 and 95%, confidence interval k is 4.3

$$u = u_c k$$

$$u = 2.71522 \times 4.3$$

$$u = 11.67545$$

Analysis:

1) From Table 5.1(b) we could see that, the highest standard deviation is 4.5008 and produce an error of 1.68%. Standard deviation shows us how much the recorded value deviate from desired value. If the standard deviation equal to 0, that means the recorded value is equal to desired value. Thus, no error produces in the reading.

2) From the formula, we know that if the standard deviation is high, the uncertainty is also high. This would mean that the reading or the thermocouple is not properly calibrated. The sample taken "n" should also be considered in calculation, because if more samples are taken, the lower uncertainty value will become and if fewer samples are taken, the uncertainty value will increase. As a conclusion, if the uncertainty value is high, the thermocouple is not reading the value correctly and it needs to be calibrated later.

3) The data must be recorded in noise free environment, if not the error percentage will increase and the data is not valid for calculation. The noise occurs when connecting to Hart Communicator parallel with temperature transmitter. This causes, the current produced by temperature transmitter

diverted to 2 junctions. Thus, decreasing the actual current value. When the current changed, the voltage is also changed according to ohm's law. And this causes the instability to the system. To remove the noise, simply remove the Hart Communicator and reconnect the DAQ card with decade resistance box.

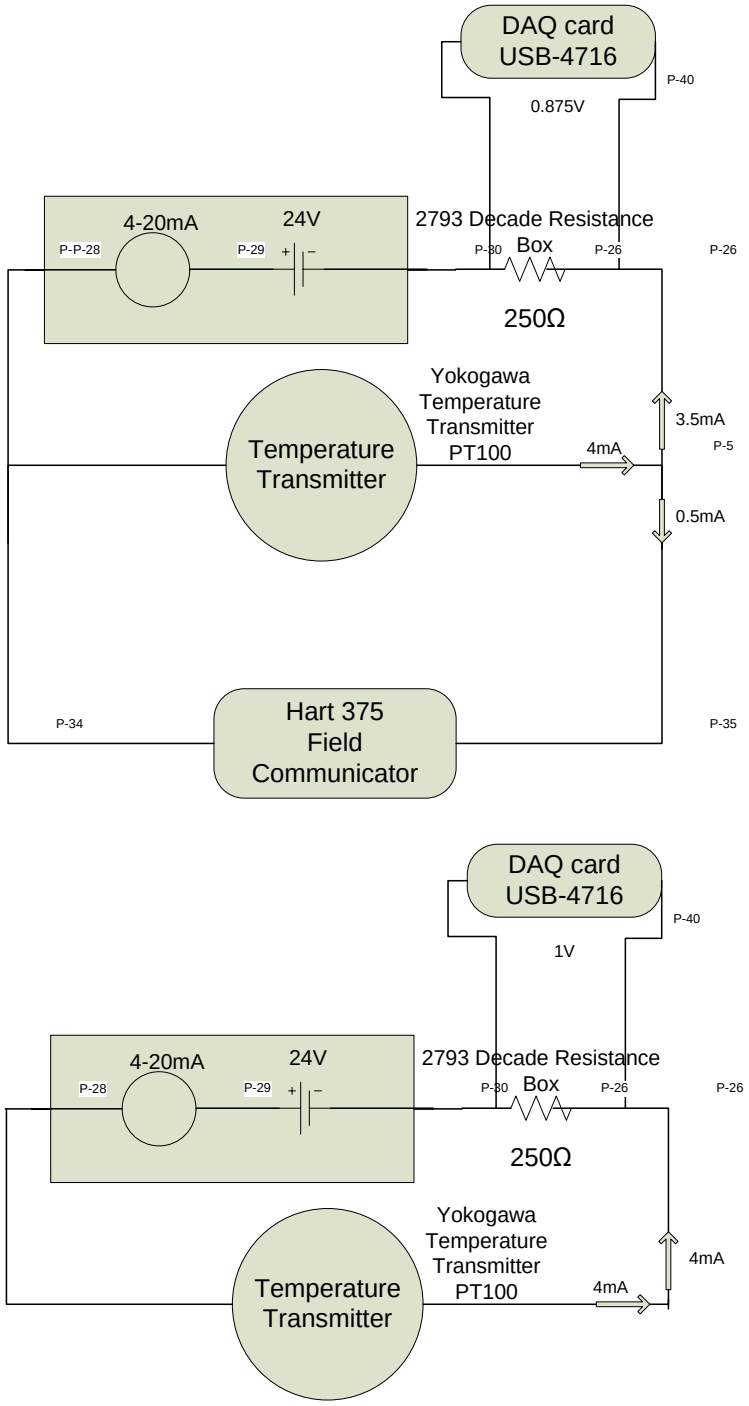

Figure 5.8: Removing Noise

4) Another potential error rises from thermocouple cold junction. The signal produced by a thermocouple is a function of the difference in temperature between the probe tip (hot junction) and the other end of the thermocouple wire (cold junction). The room temperature can affect the cold junction temperature and thus change the voltage output. This error is cannot be 100% eliminated but can be reduced. To minimize the cold junction error, we must perform thermocouple calibration a few minutes after the measuring instrument is powered up, allowing the cold junction to stabilize after warm up.

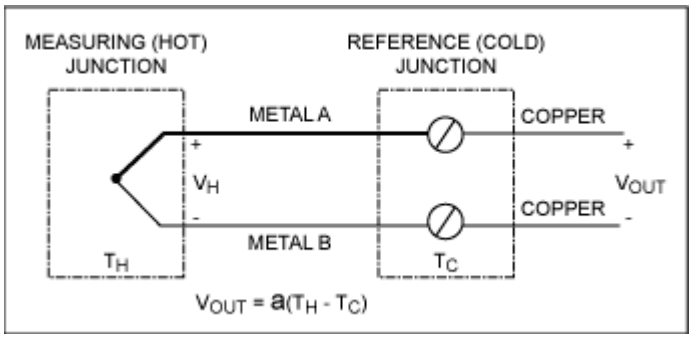

Figure 5.9: Thermocouple cold juntion

5.2.2 Experiment 2: Isotech Jupiter Heat up and Cool down process.

This experiment is conducted to determine the temperature increase rate and decrease rate of Isotech Jupiter. Data recorder function is used in this experiment to record data and plot the data with plotter function. The recorded data is compared with theoretical value as in Figure 3.08.

Objectives: Determine the time for Isotech Jupiter to heat up and cool down.

Procedures:

1) The equipment is connected as shown in Figure 5.0.
2) The desired temperature range is calibrated using Hart Communicator. In this experiment the lowest value is set to 0°C and the highest value to 600°C.

3) USB DAQ card is connected parallel to decade resistance box. Once the DAQ card has been connected, the software for this experiment is activated and the calibration value is inserted in software's setting.
4) To record the desired value using the software, make sure the input is stable. In order to get a stable input, the Hart 375 Field Communicator must be removed from the circuit during the experiment. If the input is still not stable, try to reconnect the DAQ card with decade resistance box.
5) Record the temperature data using data recorder function.
6) Increase the Isotech Jupiter temperature to 600°C.
7) Stop recording when the temperature reaches 600°C and press the save button. The graph temperature vs time can be generated automatically using the generate button. Then, save the graph.
8) Decrease the Isotech Jupiter temperature to 0°C.
9) Press the record button.
10) Stop the recording when temperature reaches 50°C and press the save button.
11) Generate the temperature vs time graph and save the graph.

Results:

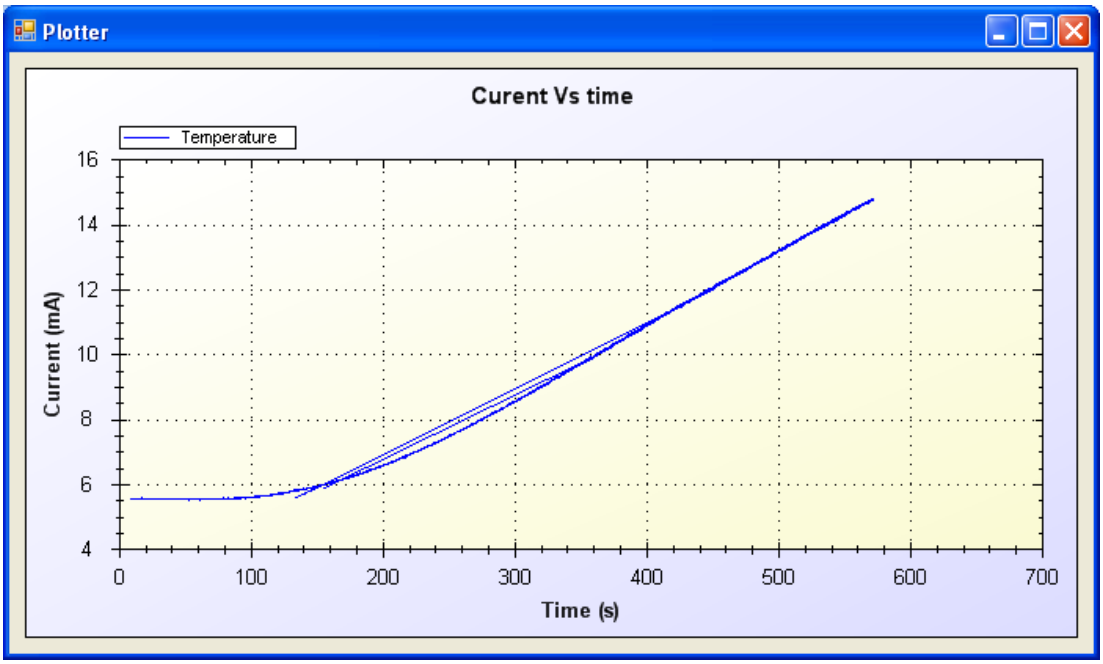

Figure 5.10: Temperature increase rate

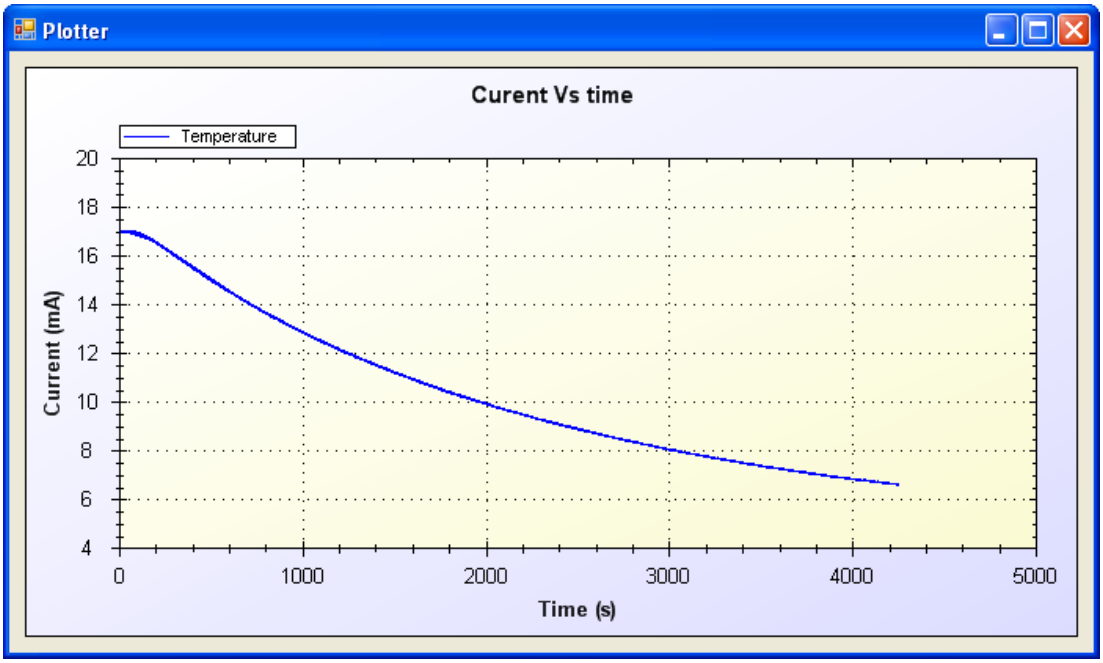

Figure 5.11: Temperature decrease rate

Calculation:

1) Temperature increase rate.

From the graph 5.2, we get

$T_1 = 534.1s$

$T_2 = 274.0s$

$I_1 = 14mA$

$I_2 = 8mA$

By using the calibration value, the current is converted to temperature.

$T = mI + C$

$$m = \frac{600℃ - 0℃}{20 - 4}$$

$m = 37.5$

$C = T - mI$

$C = 600 - 37.5(20)$

$C = -150$

$$T = m37.5 - 150$$
$$T_1 = (14)37.5 - 150$$
$$T_1 = 375℃$$

$$T_2 = (8)37.5 - 150$$
$$T_2 = 150℃$$

Temperature increase rate is calculated using the formula:

$$\frac{\Delta T}{\Delta t} = \frac{375℃ - 150℃}{535s - 274s} = 0.86℃/s \approx 1℃/s$$

2) Temperature decrease rate.

From the graph 5.2, we get

$$T_1 = 300.5s$$
$$T_2 = 3050.016s$$
$$I_1 = 16.83mA$$
$$I_2 = 7.998mA$$

By using the calibration value, the current is converted to temperature.

$$T = mI + C$$

$$m = \frac{600℃ - 0℃}{20 - 4}$$
$$m = 37.5$$

$$C = T - mI$$
$$C = 600 - 37.5(20)$$
$$C = -150$$
$$T = m37.5 - 150$$

$$T_1 = (16.83)37.5 - 150$$
$$T_1 = 481.125℃$$

$$T_2 = (7.998)37.5 - 150$$
$$T_2 = 150℃$$

Temperature decrease rate is calculated using the formula:

$$\frac{\Delta T}{\Delta t} = \frac{481.125℃ - 150℃}{3050.016s - 300.5s} = 0.12℃/s$$

Analysis:

1) From the calculation, we could say that temperature increase linearly but decrease in quadratic form. Thus, it is hard to calculate the exact temperature decrease rate.

2) From the graph 5.2 we can conclude that the temperature increase rapidly but it takes time to cool down. Theoretical value from Figure 3.08 shows that it will take about 20 minutes for temperature to decrease from 400°C-100°C. But by experiment, it takes about 4000s = 66.6 minute to cool down from 400°C to 100°C.

3) To gain a precise data, the temperature must be increase step by step. For instance, if student want to record 50°C, the temperature must be set to 100°C first. When the temperature has already pass the desired point (50°C), increase the temperature to 150°C. This process is done to ensure that the temperature do not increase rapidly and to give time to the temperature to stabilize.

CHAPTER 6

CONCLUSION AND RECOMMENDATIONS

6.1 Summary of the Work

The objective of this project has been achieved and successfully completed. This point leads to the idea of using the Data Acquisition card that reacts as a connection between the instrument hardware and computer. Since the computer can be used in various ways, such as act as real time oscilloscope, data collection, database, data analysis and manipulation etc. It is beneficial if the computer is fully utilized for this purpose.

Studying the noise characteristic is very important in this project because it will determine the efficiency of the system and reduce the error percentage. Various noise patterns can be detected by using this system for instance electromagnetic noise and electrostatic noise. To reduce this noise or completely eliminate this noise, a proper step must be taken.

As a conclusion, it is concluded that Visual Basic is a good platform to develop user friendly software to calculate and analyze the temperature data. Beside that, the systems offer a new learning experience for student. In addition, student can save a lot of time using this software instead of using manual calculation.

6.2 Recommendations For the Future Work

For the future development and enhancement, there are some suggestions that seem can be implemented to improve the system. Here is some enhancement of this project that should be added in future:-

1. This software application is too complex so, in order to reduce the CPU utilization, the code efficiency must be maximize and hardware trigger mode must be used if we want to get maximum sampling frequency.

2. Upgrade the program for more function and not limited for analysis the data only. Use the software as a controller to control an actuator such as heater. Various controllers can be implemented such as discontinuous controller or continues controller mode. For instance, P, PD and PID controller can be used to control plant temperature.

3. Upgrade the system including save function. The output from the temperature analysis can be saved in various formats such as doc and txt.

4. Write the programming using other language such as LabView, C and C++. This can allow the program can be executed in other platform instead Microsoft Windows.

5. Build a fully automatic temperature calibration system. By implementing this system it could save a lot of time in calibrating the thermocouple and in addition it will reduce cost.

6.3 Costing & Commercialization

This software application is suitable for commercialization especially to student and industrial application. Most of the application in this software has been build to replace manual calculation. Manual calculation requires a lot of time, concentration and focus. Thus, by using this software student, lecturer and engineer personnel can minimize their time in doing analysis instead of using manual calculation which require more times. In addition, this software can increase productivity of works and make job easier.

REFERENCES

1. Clarence W. de Silva (2007). Sensors and Actuators – Control System Instrumentation. CRC Press

2. Frederick F. Driscoll, Robert F. Coughlin and Robert S. Villanucci (2000). Data Acquisition and Process Control with the MC68HC11 Microcontroller. Prentice Hall, Inc

3. Curtis d. Jonson. Process Control Instrumentation Technology. Pearson International Technology

4. Deitel, T.R Neito. *Visual Bsic 6 – How to Program*. Prentice Hall, 1999

5. H.Narushima*, H.Ogura, M.Izuchi, M.Arai.Evaluation of the Freezing Point of Zinc for Pt/Pd. Fukui University, Japan. August 4-6.2003 http://ieeexplore.ieee.org/iel5/9240/29302/01324246.pdf?arnumber=1324246

6. Mark D. Bethea and Bruce N. Rosenthal. An Automated Thermocouple Calibration System. NASA Lewis Research Center, Cleveland.April 6, 1992. http://ieeexplore.ieee.org/iel1/19/4476/00177346.pdf?arnumber=177346

7. Jeffrey R. Payne Bradford A. Menz et al.High Speed PC-based Data Acquisition Systems. 1995 IEEE http://ieeexplore.ieee.org/iel3/4010/11527/00530575.pdf

8. Zedgraph http://zedgraph.org/wiki/index.php?title=Main_Page

9. Wikipedia http://en.wikipedia.com

10. Microsoft Visual Basic URL
http://www.devdos.com/vb/wanttobe.shtml

APPENDICES

APPENDIX A

Program Design

```
Imports System.Data
Imports ZedGraph
Imports System.Drawing.Drawing2D
Imports System.Data.OleDb

Public Class Form1
    Dim inc As Integer
    Dim MaxRows As Integer
    Dim con As New OleDb.OleDbConnection
    Dim sql As String
    Dim tickStart As Integer = 0
    Dim x As Integer
    Dim paint1 As Integer
    Dim tempIns As Double
    Dim volt As Double
    Dim incr As Integer
    Dim alrm1 As Integer
    Dim alrm2 As Integer
    Dim alrm3 As Integer
    Dim alrm4 As Integer
    Dim alrm5 As Integer
    Const DATA_FILE_EXTENSION As String = ".mdb"

    Private da As OleDbDataAdapter
    Private ds As New DataSet()

    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
        timer()

        btnRecord.Enabled = False
        btnReset.Enabled = False
        btnStop.Enabled = False
        btnRecord2.Enabled = False
        btnReset2.Enabled = False

        database()

    End Sub
    Private Sub Load_Excel_Details()
        'Extracting from database
        Dim filename As String

        Try
            'ds.Reset()
            da.Fill(ds, "proto")
            If ds.Tables.Count < 0 Or ds.Tables(0).Rows.Count <= 0
Then
                Exit Sub
            End If
        Catch ex As Exception
            MsgBox(ex.Message)
        End Try
        Dim Excel As Object = CreateObject("Excel.Application")
        If Excel Is Nothing Then
```

```
            MsgBox("It appears that Excel is not installed on this
machine. This operation requires MS Excel to be installed on this
machine.", MsgBoxStyle.Critical)
            Return
        End If


        'Export to Excel process
        Try
            With Excel
                .SheetsInNewWorkbook = 1
                .Workbooks.Add()
                .Worksheets(1).Select()

                Dim i As Integer = 1
                For col = 0 To ds.Tables(0).Columns.Count - 1
                    .cells(1, i).value =
ds.Tables(0).Columns(col).ColumnName
                    .cells(1, i).EntireRow.Font.Bold = True
                    i += 1
                Next
                i = 2
                Dim k As Integer = 1
                For col = 0 To ds.Tables(0).Columns.Count - 1
                    i = 2
                    For row = 0 To ds.Tables(0).Rows.Count - 1
                        .Cells(i, k).Value =
ds.Tables(0).Rows(row).ItemArray(col)
                        i += 1
                    Next
                    k += 1
                Next
                filename = txtPath.Text & "\" & Format(Now(), "dd-
MM-yyyy_hh-mm-ss") & ".xls"
                .ActiveCell.Worksheet.SaveAs(filename)
            End With

System.Runtime.InteropServices.Marshal.ReleaseComObject(Excel)
            Excel = Nothing
            MsgBox("Data's are exported to Excel Succesfully in '" &
filename & "'", MsgBoxStyle.Information)

        Catch ex As Exception
            MsgBox(ex.Message)
        End Try

        Dim pro() As Process =
System.Diagnostics.Process.GetProcessesByName("EXCEL")
        For Each i As Process In pro
            i.Kill()
        Next

    End Sub

    Private Sub timer()
        Dim myPane As GraphPane = ZedGraphControl1.GraphPane
        myPane.Title.Text = "Voltage VS Time" & Chr(10) & _
          "(After 25 seconds the graph scrolls)"
        myPane.XAxis.Title.Text = "Time, Seconds"
        myPane.YAxis.Title.Text = "Sample Potential, Volts"
```

```
        Dim list As New RollingPointPairList(1200)

        Dim curve As LineItem = myPane.AddCurve("Voltage", list,
Color.Blue, SymbolType.None)

        Timer1.Interval = 50

        myPane.XAxis.Scale.Min = 0
        myPane.XAxis.Scale.Max = 30
        myPane.YAxis.Scale.Min = 0
        myPane.YAxis.Scale.Max = 6
        myPane.XAxis.Scale.MinorStep = 1
        myPane.XAxis.Scale.MajorStep = 5

        myPane.Chart.Fill = New Fill(Color.White,
Color.LightGoldenrodYellow, 45.0F)

        myPane.Fill = New Fill(Color.White, Color.FromArgb(220, 220,
255), 45.0F)

        myPane.XAxis.MajorGrid.IsVisible = True
        myPane.YAxis.MajorGrid.IsVisible = True

        ZedGraphControl1.AxisChange()
        tickStart = Environment.TickCount
    End Sub

    Private Sub Button5_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs)
        Dim y As Double

        inc = 0
        y = 0
        For i = 0 To MaxRows - 1
            y = y + ds.Tables("proto").Rows(inc).Item(2)
            inc = inc + 1
        Next i
        MsgBox(y)

    End Sub

    Private Sub Button9_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles Button9.Click


        If CheckBox1.Checked Then
            Graph1.Visible = True
        End If
        If CheckBox2.Checked Then
            Graph2.Visible = True
        End If
        If CheckBox3.Checked Then
            Graph3.Visible = True
        End If
        If CheckBox4.Checked Then
            Graph4.Visible = True
        End If
        If CheckBox5.Checked Then
            Graph5.Visible = True
        End If
        If CheckBox6.Checked Then
```

```
            Graph6.Visible = True
        End If

    End Sub
    Private _drawInsidePanel As Boolean
    Private Sub Timer1_Tick(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles Timer1.Tick
        Dim textA1 As Integer
        Dim textA2 As Integer
        Dim textA3 As Integer
        Dim textA4 As Integer
        Dim textA5 As Integer

        If ZedGraphControl1.GraphPane.CurveList.Count <= 0 Then
Return

        Dim curve As LineItem =
ZedGraphControl1.GraphPane.CurveList(0)
        If curve Is Nothing Then Return

        Dim list As IPointListEdit = curve.Points
        If list Is Nothing Then Return

        Dim time As Double = (Environment.TickCount - tickStart) /
1000.0

        If rbtnSimulation.Checked Then
            ' 3 seconds per cycle
            ' Produce dummy data range 1-5V
            volt = (Math.Sin(2 * Math.PI * time / 3.0)) * 2 + 3
            'volt = (Math.Sin(0.1 * Math.PI * time / 3.0)) * 2 + 3
        Else
            'volt = Math.Round(AxAdvAI1.DataAnalog, 2)
            volt = AxAdvAI1.DataAnalog
        End If

        list.Add(time, volt)
        Dim xScale As Scale = ZedGraphControl1.GraphPane.XAxis.Scale
        If time > xScale.Max - xScale.MajorStep Then
            xScale.Max = time + xScale.MajorStep
            xScale.Min = xScale.Max - 30.0
        End If

        ZedGraphControl1.AxisChange()
        ZedGraphControl1.Invalidate()


        InstrumentCalibration()

        txtDataTime.Text = Math.Round(time, 1) & "s"
        txtDataTemp.Text = Math.Round(tempIns, 3) & "°C"
        txtDataVolt.Text = Math.Round(volt, 4) & "V"
        txtDataCurrent.Text = Math.Round(((volt / 250) * 10 ^ (3)),
3) & "mA"


        textA1 = txtA1.Text
        textA2 = txtA2.Text
        textA3 = txtA3.Text
        textA4 = txtA4.Text
        textA5 = txtA5.Text
```

```
        If Math.Round(textA1, 0) > Math.Round(tempIns, 0) And
cmbA1.SelectedItem = "Enable" And alrm1 = 1 Then
            alrm1 = 0
            MsgBox("Temperature is recorded in database.")
        ElseIf Math.Round(textA2, 0) > Math.Round(tempIns, 0) And
cmbA2.SelectedItem = "Enable" And alrm2 = 1 Then
            alrm2 = 0
            MsgBox("Temperature is recorded in database.")
        ElseIf Math.Round(textA3, 0) > Math.Round(tempIns, 0) And
cmbA3.SelectedItem = "Enable" And alrm3 = 1 Then
            alrm3 = 0
            MsgBox("Temperature is recorded in database.")
        ElseIf Math.Round(textA4, 0) > Math.Round(tempIns, 0) And
cmbA4.SelectedItem = "Enable" And alrm4 = 1 Then
            alrm4 = 0
            MsgBox("Temperature is recorded in database.")
        ElseIf Math.Round(textA5, 0) > Math.Round(tempIns, 0) And
cmbA5.SelectedItem = "Enable" And alrm5 = 1 Then
            alrm5 = 0
            MsgBox("Temperature is recorded in database.")
        End If


        paint1 = Math.Round(time, 0)
        _drawInsidePanel = True
        Panel1.Invalidate() ' force to redraw the Panel1

    End Sub


    Private Sub Panel1_Paint(ByVal sender As System.Object, ByVal e
As System.Windows.Forms.PaintEventArgs) Handles Panel1.Paint
        Dim g As Graphics = e.Graphics
        Dim rect As New Rectangle
        Dim y As Integer

        y = 85 * volt - 85

        g.FillRectangle(Brushes.Red, 0, 0, 20, 340)
        If _drawInsidePanel Then
            ' Draw inside the panel
            rect = New Rectangle(0, 0, 80, 340 - y)
            g.FillRectangle(Brushes.AliceBlue, 0, 0, 80, 340 - y)
        End If
    End Sub

    Private Sub Button3_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles btnStart.Click
        'ZedGraphControl1.GraphPane.CurveList.Clear()
        Timer1.Enabled = True
        Timer1.Start()
        RadioButton1.Enabled = False
        RadioButton2.Enabled = False
        RadioButton3.Enabled = False
        btnRecord.Enabled = True
        btnReset.Enabled = True
        btnStart.Enabled = False
        btnStop.Enabled = True
        btnRecord2.Enabled = True
```

```
        btnReset2.Enabled = True
        My.Forms.Voltage.Timer1.Enabled = True
        My.Forms.Voltage.Timer1.Start()

        incr = 1
    End Sub

    Private Sub Button4_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles btnStop.Click
        Timer1.Enabled = False
        Timer1.Stop()
        RadioButton1.Enabled = True
        RadioButton2.Enabled = True
        RadioButton3.Enabled = True
        btnRecord.Enabled = False
        btnReset.Enabled = False
        btnStart.Enabled = True
        btnStop.Enabled = False
        btnRecord2.Enabled = False
        btnReset2.Enabled = False
        My.Forms.Voltage.Timer1.Enabled = False
        My.Forms.Voltage.Timer1.Stop()

        incr = 1
    End Sub

    Private Sub Button10_Click(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles Button10.Click
        Form2.ShowDialog()
    End Sub

    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles btnRecord.Click
        MaxRows = ds.Tables("proto").Rows.Count

        InstrumentCalibration()
        If RadioButton1.Checked And MaxRows <> incr Then
            ds.Tables("proto").Rows(incr).Item(4) =
Math.Round(tempIns, 3)
            MsgBox("Data " & incr & " is recorded in 1st sample")
            incr = incr + 1
        ElseIf RadioButton2.Checked And MaxRows <> incr Then
            ds.Tables("proto").Rows(incr).Item(5) =
Math.Round(tempIns, 3)
            MsgBox("Data " & incr & " is recorded in 2nd sample")
            incr = incr + 1
        ElseIf RadioButton3.Checked And MaxRows <> incr Then
            ds.Tables("proto").Rows(incr).Item(6) =
Math.Round(tempIns, 3)
            MsgBox("Data " & incr & " is recorded in 3rd sample")
            incr = incr + 1
        End If
    End Sub

    Private Sub ComboBox1_SelectedIndexChanged(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
ComboBox1.SelectedIndexChanged
        Dim myPane As GraphPane = ZedGraphControl1.GraphPane
        myPane.XAxis.MajorGrid.IsVisible = ComboBox1.Text
    End Sub
```

```
    Private Sub ComboBox2_SelectedIndexChanged(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
ComboBox2.SelectedIndexChanged
        Dim myPane As GraphPane = ZedGraphControl1.GraphPane
        myPane.YAxis.MajorGrid.IsVisible = ComboBox2.Text
    End Sub

    Private Sub ComboBox3_SelectedIndexChanged(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
cmbA1.SelectedIndexChanged
        If cmbA1.SelectedIndex = 1 Then
            txtA1.Enabled = False
            alrm1 = 1
        Else
            txtA1.Enabled = True
        End If
    End Sub

    Private Sub cmbA2_SelectedIndexChanged(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
cmbA2.SelectedIndexChanged
        If cmbA2.SelectedIndex = 1 Then
            txtA2.Enabled = False
            alrm2 = 1
        Else
            txtA2.Enabled = True
        End If
    End Sub

    Private Sub cmbA3_SelectedIndexChanged(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
cmbA3.SelectedIndexChanged
        If cmbA3.SelectedIndex = 1 Then
            txtA3.Enabled = False
            alrm3 = 1
        Else
            txtA3.Enabled = True
        End If
    End Sub

    Private Sub cmbA4_SelectedIndexChanged(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
cmbA4.SelectedIndexChanged
        If cmbA4.SelectedIndex = 1 Then
            txtA4.Enabled = False
            alrm4 = 1
        Else
            txtA4.Enabled = True
        End If
    End Sub

    Private Sub cmbA5_SelectedIndexChanged(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
cmbA5.SelectedIndexChanged
        If cmbA5.SelectedIndex = 1 Then
            txtA5.Enabled = False
            alrm5 = 1
        Else
            txtA5.Enabled = True
        End If
    End Sub
```

```
    Private Sub cmbSample_SelectedIndexChanged(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
cmbSample.SelectedIndexChanged
        If cmbSample.SelectedItem = "1" Then
            Timer1.Interval = 1000
        ElseIf cmbSample.SelectedItem = "2" Then
            Timer1.Interval = 500
        ElseIf cmbSample.SelectedItem = "4" Then
            Timer1.Interval = 250
        ElseIf cmbSample.SelectedItem = "8" Then
            Timer1.Interval = 125
        ElseIf cmbSample.SelectedItem = "16" Then
            Timer1.Interval = 62.5
        ElseIf cmbSample.SelectedItem = "32" Then
            Timer1.Interval = 31.25
        End If
    End Sub

    Private Sub Button2_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles btnExport.Click
        Load_Excel_Details()
    End Sub

    Private Sub cmdSelectDevice_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles cmdSelectDevice.Click
        AxAdvAI1.SelectDevice()
        txtDeviceNumber.Text = AxAdvAI1.DeviceNumber
        txtDeviceName.Text = AxAdvAI1.DeviceName

    End Sub

    Private Sub cmdRead_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles cmdRead.Click
        txtDataDigital.Text = Hex(AxAdvAI1.DataDigital)
        txtDataAnalog.Text = Format(AxAdvAI1.DataAnalog,
"0.######0")

    End Sub
    Private Sub InstrumentCalibration()
        Dim y1 As Double
        Dim y2 As Double
        Dim x1 As Double
        Dim x2 As Double
        Dim m As Double
        Dim c As Double
        Dim yIns As Double

        y1 = txtTempLR.Text
        y2 = txtTempUR.Text
        x1 = (txtTcLR.Text) * (1 * 10 ^ (-3))
        x2 = (txtTcUR.Text) * (1 * 10 ^ (-3))

        m = (y2 - y1) / (x2 - x1)
        c = y2 - (x2 * m)
        yIns = m * (volt / 250) + c
        tempIns = yIns

    End Sub
```

```
    Private Sub btnMean_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles btnMean.Click
        Dim y As Double
        Dim a As Double
        Dim b As Double
        Dim c As Double
        Dim obj As Object
        Dim obj2 As Object
        Dim obj3 As Object
        Dim g As Integer

        For i = 1 To MaxRows - 1

            obj = ds.Tables("proto").Rows(i).Item(4)
            obj2 = ds.Tables("proto").Rows(i).Item(5)
            obj3 = ds.Tables("proto").Rows(i).Item(6)

            If IsDBNull(obj) Or IsDBNull(obj2) Or IsDBNull(obj3)
Then
                g = 1
            Else
                a = ds.Tables("proto").Rows(i).Item(4)
                b = ds.Tables("proto").Rows(i).Item(5)
                c = ds.Tables("proto").Rows(i).Item(6)
                y = (a + b + c) / 3
                ds.Tables("proto").Rows(i).Item(7) = Math.Round(y,
4)
            End If

            If g = 1 And i = MaxRows - 1 Then
                MsgBox("Not Enough Data")
            End If

        Next i
    End Sub


    Private Sub btnStd_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles btnStd.Click
        Dim avg As Double
        Dim a As Double
        Dim b As Double
        Dim c As Double
        Dim sum As Double
        Dim obj As Object
        Dim obj2 As Object
        Dim obj3 As Object
        Dim g As Integer



        For i = 1 To MaxRows - 1

            obj = ds.Tables("proto").Rows(i).Item(4)
            obj2 = ds.Tables("proto").Rows(i).Item(5)
            obj3 = ds.Tables("proto").Rows(i).Item(6)

            If IsDBNull(obj) Or IsDBNull(obj2) Or IsDBNull(obj3)
Then
                g = 1
            Else
```

```
                a = ds.Tables("proto").Rows(i).Item(4)
                b = ds.Tables("proto").Rows(i).Item(5)
                c = ds.Tables("proto").Rows(i).Item(6)
                avg = (a + b + c) / 3
                sum = Math.Sqrt(0.5) * ((a - avg) ^ 2 + (b - avg) ^
2 + (c - avg) ^ 2)

                ds.Tables("proto").Rows(i).Item(8) = Math.Round(sum,
4)
            End If

            If g = 1 And i = MaxRows - 1 Then
                MsgBox("Not Enough Data")
            End If
        Next i
    End Sub

    Private Sub btnError_Click(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles btnError.Click
        Dim avg As Double
        Dim a As Double
        Dim b As Double
        Dim c As Double
        Dim x As Double
        Dim sum As Double
        Dim obj As Object
        Dim obj2 As Object
        Dim obj3 As Object
        Dim obj4 As Object
        Dim g As Integer

        For i = 1 To MaxRows - 1

            obj = ds.Tables("proto").Rows(i).Item(4)
            obj2 = ds.Tables("proto").Rows(i).Item(5)
            obj3 = ds.Tables("proto").Rows(i).Item(6)
            obj4 = ds.Tables("proto").Rows(i).Item(2)

            If IsDBNull(obj) Or IsDBNull(obj2) Or IsDBNull(obj3) Or
IsDBNull(obj4) Then
                g = 1
            Else
                a = ds.Tables("proto").Rows(i).Item(4)
                b = ds.Tables("proto").Rows(i).Item(5)
                c = ds.Tables("proto").Rows(i).Item(6)
                x = ds.Tables("proto").Rows(i).Item(2)
                avg = (a + b + c) / 3

                If x > avg Then
                    sum = ((avg - x) / x) * 100
                    ds.Tables("proto").Rows(i).Item(9) =
Math.Round(sum, 4)
                ElseIf avg > x Then
                    sum = ((avg - x) / x) * 100
                    ds.Tables("proto").Rows(i).Item(9) =
Math.Round(sum, 4)
                ElseIf x = avg Then
                    sum = 0
                    ds.Tables("proto").Rows(i).Item(9) =
Math.Round(sum, 4)
                End If
```

```
            End If

            If g = 1 And i = MaxRows - 1 Then
                MsgBox("Not Enough Data")
            End If

        Next i
    End Sub

    Private Sub btnReset_Click(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles btnReset.Click
        If RadioButton1.Checked And incr <> 1 Then
            incr = incr - 1
            ds.Tables("proto").Rows(incr).Item(4) = 0
            MsgBox("Data " & incr & " is deleted from 1st sample")
        ElseIf RadioButton2.Checked And incr <> 1 Then
            incr = incr - 1
            ds.Tables("proto").Rows(incr).Item(5) = 0
            MsgBox("Data " & incr & " is deleted from 1st sample")
        ElseIf RadioButton3.Checked And incr <> 1 Then
            incr = incr - 1
            ds.Tables("proto").Rows(incr).Item(6) = 0
            MsgBox("Data " & incr & " is deleted from 1st sample")
        End If
    End Sub


    Private Sub btnAlarmSave_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles btnAlarmSave.Click
        Dim lowRange As Double
        Dim highRange As Double

        lowRange = txtTempLR.Text
        highRange = txtTempUR.Text

        If cmbA1.SelectedItem = "Enable" Then
            alrm1 = 1
        ElseIf cmbA1.SelectedItem = "Disable" Then
            alrm1 = 0
        End If

        If cmbA2.SelectedItem = "Enable" Then
            alrm2 = 1
        ElseIf cmbA2.SelectedItem = "Disable" Then
            alrm2 = 0
        End If

        If cmbA3.SelectedItem = "Enable" Then
            alrm3 = 1
        ElseIf cmbA3.SelectedItem = "Disable" Then
            alrm3 = 0
        End If

        If cmbA4.SelectedItem = "Enable" Then
            alrm4 = 1
        ElseIf cmbA4.SelectedItem = "Disable" Then
            alrm4 = 0
        End If

        If cmbA5.SelectedItem = "Enable" Then
            alrm5 = 1
```

```
        ElseIf cmbA5.SelectedItem = "Disable" Then
            alrm5 = 0
        End If

        If txtA1.Text < lowRange Or txtA1.Text > highRange Then
            MsgBox("Out of range.The range is between " & lowRange &
"°C" & " to " & highRange & "°C")
            txtA1.Text = lowRange
            alrm1 = 0
        End If

        If txtA2.Text < lowRange Or txtA2.Text > highRange Then
            MsgBox("Out of range.The range is between " & lowRange &
"°C" & " to " & highRange & "°C")
            txtA2.Text = lowRange
            alrm2 = 0
        End If

        If txtA3.Text < lowRange Or txtA3.Text > highRange Then
            MsgBox("Out of range.The range is between " & lowRange &
"°C" & " to " & highRange & "°C")
            txtA3.Text = lowRange
            alrm3 = 0
        End If

        If txtA4.Text < lowRange Or txtA4.Text > highRange Then
            MsgBox("Out of range.The range is between " & lowRange &
"°C" & " to " & highRange & "°C")
            txtA4.Text = lowRange
            alrm4 = 0
        End If

        If txtA5.Text < lowRange Or txtA5.Text > highRange Then
            MsgBox("Out of range.The range is between " & lowRange &
"°C" & " to " & highRange & "°C")
            txtA5.Text = lowRange
            alrm5 = 0
        End If


    End Sub



    Private Sub Button5_Click_1(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles Button5.Click
        Voltage.Visible = True
    End Sub


    Private Sub btnBrowse_Click(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles btnBrowse.Click
        Dim objFolderDialog As New FolderBrowserDialog()

        txtPath.Text = GetNetworkFolders(objFolderDialog)
    End Sub
    Public Shared Function GetNetworkFolders(ByVal
oFolderBrowserDialog _
      As FolderBrowserDialog) As String

        If oFolderBrowserDialog.ShowDialog() = DialogResult.OK Then
```

```
            Return oFolderBrowserDialog.SelectedPath
        Else
            Return ""
        End If
    End Function

    Private Sub txtTempLR_TextChanged(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles txtTempLR.TextChanged

    End Sub

    Private Sub btnSave_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles btnSave.Click
        Dim command_builder As New OleDbCommandBuilder(da)
        da.Update(ds, "proto")
        MsgBox("Data has been saved")
    End Sub


    Private Sub Button7_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles Button7.Click
        vCurrent.Visible = True
    End Sub

    Private Sub Button6_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles Button6.Click
        vTemperature.Visible = True
    End Sub

    Private Sub btnRecord2_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles btnRecord2.Click
        MaxRows = ds.Tables("proto").Rows.Count

        If RadioButton1.Checked And MaxRows <> incr Then
            ds.Tables("proto").Rows(incr).Item(4) = Math.Round(volt
/ 0.25, 5)
            MsgBox("Data " & incr & " is recorded in 1st sample")
            incr = incr + 1
        ElseIf RadioButton2.Checked And MaxRows <> incr Then
            ds.Tables("proto").Rows(incr).Item(5) = Math.Round(volt
/ 0.25, 5)
            MsgBox("Data " & incr & " is recorded in 2nd sample")
            incr = incr + 1
        ElseIf RadioButton3.Checked And MaxRows <> incr Then
            ds.Tables("proto").Rows(incr).Item(6) = Math.Round(volt
/ 0.25, 5)
            MsgBox("Data " & incr & " is recorded in 3rd sample")
            incr = incr + 1
        End If
    End Sub

    Private Sub btnReset2_Click(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles btnReset2.Click
        If RadioButton1.Checked And incr <> 1 Then
            incr = incr - 1
            ds.Tables("proto").Rows(incr).Item(4) = 0
            MsgBox("Data " & incr & " is deleted from 1st sample")
        ElseIf RadioButton2.Checked And incr <> 1 Then
            incr = incr - 1
            ds.Tables("proto").Rows(incr).Item(5) = 0
            MsgBox("Data " & incr & " is deleted from 1st sample")
```

```
        ElseIf RadioButton3.Checked And incr <> 1 Then
            incr = incr - 1
            ds.Tables("proto").Rows(incr).Item(6) = 0
            MsgBox("Data " & incr & " is deleted from 1st sample")
        End If
    End Sub

    Private Sub btnConvert_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles btnConvert.Click
        Dim y1 As Double
        Dim y2 As Double
        Dim x1 As Double
        Dim x2 As Double
        Dim m As Double
        Dim c As Double
        Dim x As Double
        Dim y As Double

        If txtMinTemp.Text = "" Or txtMaxTemp.Text = "" Or
txtMaxCur.Text = "" Or txtMinCur.Text = "" Then
            MsgBox("Please fill in the required value")
        ElseIf rbCurtoTemp.Checked And txtValCur.Text = "" Or
rbTemptoCur.Checked And txtValTemp.Text = "" Then
            MsgBox("Please fill in the required value")
        Else
            y1 = txtMinTemp.Text
            y2 = txtMaxTemp.Text
            x1 = (txtMinCur.Text)
            x2 = (txtMaxCur.Text)

            m = (y2 - y1) / (x2 - x1)
            c = y2 - (x2 * m)

            If rbCurtoTemp.Checked Then
                x = txtValCur.Text
                y = m * x + c
                txtValTemp.Text = Math.Round(y, 5)
            ElseIf rbTemptoCur.Checked Then
                y = txtValTemp.Text
                x = (y - c) / m
                txtValCur.Text = Math.Round(x, 5)
            End If
        End If
    End Sub

    Private Sub rbCurtoTemp_CheckedChanged(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
rbCurtoTemp.CheckedChanged
        If rbCurtoTemp.Checked Then
            txtValTemp.ReadOnly = True
            txtValCur.ReadOnly = False
        ElseIf rbTemptoCur.Checked Then
            txtValTemp.ReadOnly = False
            txtValCur.ReadOnly = True
        End If
    End Sub


    Private Sub btnGenerate_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles btnGenerate.Click
        Dim command_builder As New OleDbCommandBuilder(da)
```

```
        Dim x As Integer
        Dim percent As Single
        Dim a As Single
        Dim incrx As Integer
        Dim temp As Single
        Dim upper As Single
        Dim lower As Single
        Dim uppCur As Single
        Dim lowCur As Single
        Dim current As Single
        Dim more As Integer
        Dim less As Integer


        If txtUpperVal.Text = "" Or txtLowerVal.Text = "" Then
            MsgBox("Please enter a correct value")
        Else
            upper = txtUpperVal.Text
            lower = txtLowerVal.Text
            uppCur = txtTcUR.Text
            lowCur = txtTcLR.Text
            x = txtPoint.Text - 1
            percent = 100 / x

            MaxRows = ds.Tables("proto").Rows.Count

            If txtPoint.Text > (MaxRows - 1) Then
                more = txtPoint.Text - (MaxRows - 1)
                For i = 1 To more
                    ds.Tables("proto").Rows.Add()
                    da.Update(ds, "proto")
                Next i

            ElseIf txtPoint.Text < (MaxRows - 1) Then
                less = (MaxRows - 1) - txtPoint.Text
                For i = 1 To less
                    MaxRows = ds.Tables("proto").Rows.Count
                    ds.Tables("proto").Rows(MaxRows - 1).Delete()
                    da.Update(ds, "proto")
                Next i

            End If

            MaxRows = ds.Tables("proto").Rows.Count
            a = 0
            For i = 1 To MaxRows - 1
                incrx = incrx + 1
                ds.Tables("proto").Rows(incrx).Item(1) = a
                temp = (a / 100) * (upper - lower) + lower
                ds.Tables("proto").Rows(incrx).Item(2) = temp
                current = (a / 100) * (uppCur - lowCur) + lowCur
                ds.Tables("proto").Rows(incrx).Item(3) = current
                a = a + percent
            Next i
        End If


    End Sub

    Private Sub btnDel_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs)
```

```
        Dim command_builder As New OleDbCommandBuilder(da)
        'Dim dsNewRow As DataRow
        MaxRows = ds.Tables("proto").Rows.Count
        ds.Tables("proto").Rows(MaxRows - 1).Delete()
        'da.Update(ds, "proto")
    End Sub
    Private Sub btnSaveData_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles btnSaveData.Click
        Data_Recorder.Visible = True
    End Sub

    Private Sub btnPlotter_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles btnPlotter.Click
        OpenFileDialog1.Filter = DATA_FILE_EXTENSION & _
            " files (*" & DATA_FILE_EXTENSION & "|*" &
DATA_FILE_EXTENSION
        OpenFileDialog1.FilterIndex = 1
        OpenFileDialog1.RestoreDirectory = True
        OpenFileDialog1.ShowDialog()
    End Sub


    Private Sub OpenFileDialog1_FileOk(ByVal sender As
System.Object, ByVal e As System.ComponentModel.CancelEventArgs)
Handles OpenFileDialog1.FileOk
        Plotter.Visible = True
    End Sub

    Private Sub btnBrowseDatabse_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
btnBrowseDatabse.Click
        OpenFileDialog2.Filter = DATA_FILE_EXTENSION & _
            " files (*" & DATA_FILE_EXTENSION & "|*" &
DATA_FILE_EXTENSION
        OpenFileDialog2.FilterIndex = 1
        OpenFileDialog2.RestoreDirectory = True
        OpenFileDialog2.ShowDialog()
    End Sub

    Private Sub OpenFileDialog2_FileOk(ByVal sender As
System.Object, ByVal e As System.ComponentModel.CancelEventArgs)
Handles OpenFileDialog2.FileOk
        txtDatabase.Text = OpenFileDialog2.FileName
        database()
    End Sub
    Private Sub database()

        con.ConnectionString =
"PROVIDER=Microsoft.Jet.OLEDB.4.0;Data Source =" & txtDatabase.Text

        con.Open()

        sql = " select*from tblContacts"
        da = New OleDb.OleDbDataAdapter(sql, con)
        da.Fill(ds, "proto")

        con.Close()

        MaxRows = ds.Tables("proto").Rows.Count
        inc = -1
        'NavigateRecords()
```

```
        x = 0
        Try
            ds.Reset()
            da.Fill(ds, "proto")
            DataGridView1.DataSource = ds.Tables(0)
        Catch ex As Exception
            MsgBox(ex.Message)
        End Try
    End Sub

    Private Sub RadioButton5_CheckedChanged(ByVal sender As
System.Object, ByVal e As System.EventArgs)
        Dim y As Double
        Dim a As Double
        Dim b As Double
        Dim c As Double
        Dim obj As Object
        Dim obj2 As Object
        Dim obj3 As Object
        Dim g As Integer
        Dim y1 As Double
        Dim y2 As Double
        Dim x1 As Double
        Dim x2 As Double
        Dim m As Double
        Dim co As Double
        Dim x As Double
        Dim yo As Double

        For i = 1 To MaxRows - 1

            obj = ds.Tables("proto").Rows(i).Item(4)
            obj2 = ds.Tables("proto").Rows(i).Item(5)
            obj3 = ds.Tables("proto").Rows(i).Item(6)

            If IsDBNull(obj) Or IsDBNull(obj2) Or IsDBNull(obj3)
Then
                g = 1
            Else
                a = ds.Tables("proto").Rows(i).Item(4)
                b = ds.Tables("proto").Rows(i).Item(5)
                c = ds.Tables("proto").Rows(i).Item(6)
                y = (a + b + c) / 3

                y1 = txtTempLR.Text
                y2 = txtTempUR.Text
                x1 = (txtTcLR.Text)
                x2 = (txtTcUR.Text)

                m = (y2 - y1) / (x2 - x1)
                co = y2 - (x2 * m)

                yo = m * a + co
                ds.Tables("proto").Rows(i).Item(4) = yo


            End If

            If g = 1 And i = MaxRows - 1 Then
                MsgBox("Not Enough Data")
            End If
```

```
        Next i
    End Sub

    Private Sub RadioButton4_CheckedChanged(ByVal sender As
System.Object, ByVal e As System.EventArgs)
        Dim y As Double
        Dim a As Double
        Dim b As Double
        Dim c As Double
        Dim obj As Object
        Dim obj2 As Object
        Dim obj3 As Object
        Dim g As Integer
        Dim y1 As Double
        Dim y2 As Double
        Dim x1 As Double
        Dim x2 As Double
        Dim m As Double
        Dim co As Double
        Dim x As Double
        Dim yo As Double

        For i = 1 To MaxRows - 1

            obj = ds.Tables("proto").Rows(i).Item(4)
            obj2 = ds.Tables("proto").Rows(i).Item(5)
            obj3 = ds.Tables("proto").Rows(i).Item(6)

            If IsDBNull(obj) Or IsDBNull(obj2) Or IsDBNull(obj3)
Then
                g = 1
            Else
                a = ds.Tables("proto").Rows(i).Item(4)
                b = ds.Tables("proto").Rows(i).Item(5)
                c = ds.Tables("proto").Rows(i).Item(6)
                y = (a + b + c) / 3

                y1 = txtTempLR.Text
                y2 = txtTempUR.Text
                x1 = (txtTcLR.Text)
                x2 = (txtTcUR.Text)

                m = (y2 - y1) / (x2 - x1)
                co = y2 - (x2 * m)

                yo = a
                x = (yo - co) / m
                ds.Tables("proto").Rows(i).Item(4) = x

                yo = b
                x = (yo - co) / m
                ds.Tables("proto").Rows(i).Item(5) = x

                yo = c
                x = (yo - co) / m
                ds.Tables("proto").Rows(i).Item(6) = x

            End If

            If g = 1 And i = MaxRows - 1 Then
```

```
                MsgBox("Not Enough Data")
            End If

        Next i
    End Sub


    Private Sub btnSaveINst_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles btnSaveINst.Click
        If txtTempLR.Text = "" Or txtTempUR.Text = "" Or
txtTcLR.Text = "" Or txtTcUR.Text = "" Then
            MsgBox("Please insert the value")
            txtTempLR.Text = 0
            txtTempUR.Text = 200
            txtTcLR.Text = 4
            txtTcUR.Text = 20
        End If
    End Sub

    Private Sub AxAdvAO1_OnTimeOut(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles AxAdvAO1.OnTimeOut

    End Sub
End Class
```

APPENDIX B