

**DESIGN, DEVELOP AND PERFORM ROBUSTNESS TEST OF
ANALYZER²_{ver2}**

By

KHAIRUL AZRI BIN JAMALUDDIN

FINAL PROJECT REPORT

**Submitted to the Electrical & Electronics Engineering Programme
in Partial Fulfillment of the Requirements
for the Degree
Bachelor of Engineering (Hons)
(Electrical & Electronics Engineering)**

**Universiti Teknologi Petronas
Bandar Seri Iskandar
31750 Tronoh
Perak Darul Ridzuan**

**© Copyright 2010
by
Khairul Azri Jamaluddin, 2010**

CERTIFICATION OF APPROVAL

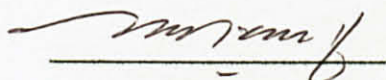
DESIGN, DEVELOP AND PERFORM ROBUSTNESS TEST OF ANALYZER²_{ver2}

by

Khairul Azri Bin Jamaluddin

A project dissertation submitted to the
Electrical & Electronics Engineering Programme
Universiti Teknologi PETRONAS
in partial fulfilment of the requirement for the
Bachelor of Engineering (Hons)
(Electrical & Electronics Engineering)

Approved by,



Dr Rosdiazli bin Ibrahim

Project Supervisor

**UNIVERSITI TEKNOLOGI PETRONAS
TRONOH, PERAK**

June 2010

CERTIFICATION OF ORIGINALITY

This is to certify that I am responsible for the work submitted in this project, that the original work is my own except as specified in the references and acknowledgements, and that the original work contained herein have not been undertaken or done by unspecified sources or persons.



Khairul Azri bin Jamaluddin

ABSTRACT

Software known as Analyzer² was already been designed to facilitate predictive maintenance and asset management system in PETRONAS Penapisan Terengganu (PPTSB). The idea is to design and develop a new version of Analyzer² software and test it so that it can be shared among other PETRONAS Operating Unit. The objective of this project is mainly to verify the 6 algorithms used in old version of Analyzer² Software through research and discussion. Next objective is to design, develop and enhance Analyzer² Software version 2 by using Visual Basic Software. Last objective is to perform Robustness Test on the software. For this test, a list of input is created for particular characteristic. The output from the input will be compared to the expected output. The output should be consistent with the theory of algorithm. This is to ensure that the software has good validation and ready to be used anywhere with high degree of accuracy.

ACKNOWLEDGEMENTS

I would like to pay my highest gratitude to my supervisor, Dr Rosdiazli Bin Ibrahim for teaching, advising, and assisting me throughout this project and to Mr Azrin Sani(former Engineer from PETRONAS Penapisan Terengganu) and Ms Wahidah Adzman (Group Technology Solution) who always there for guidance in times of uncertainties. Also special mention is Mr Jafreezal Bin Jaafar from Computer Information Science Department. Their commitment and dedication in this project is beyond university's required obligations. Without their expertise, experience and advice, the project would not have been successfully completed.

Also high on the list acknowledgement is to Ms Siti Hawa the coordinator for Electrical and Electronics Final Year Project (FYP). Her endurance in advising me and everyone else on the right procedure of logbooks and reports are irreplaceable.

To my friends Lim Hwei Ping who had helped me so much in this topic; assisting and always being there to answer problems face throughout the development of software, thank you so much.

Finally, I would like to acknowledge all the others whose name was not mention on this page, but has one way or another contributed to the accomplishment of this project.

TABLE OF CONTENTS

LIST OF FIGURES	ix
LIST OF TABLES.....	x
CHAPTER 1 INTRODUCTION.....	1
1.1 Background of Study	1
1.2 Problem Statement.....	2
1.3 Objectives & Scope of Study.....	3
1.3.1 Objectives.....	2
1.3.2 Scope of Study.....	3
CHAPTER 2 LITERATURE REVIEW.....	4
2.1 Analytical Instrumentation.....	4
2.1.1 Analyzer Application.....	6
2.2 Gas Chromatography.....	8
2.2.1 Measurement Principles.....	8
2.2.2 Common GC Application.....	11
2.2.3 Ranges of Variable.....	12
2.3 Analyzer ² Software.....	13
2.3.1 Analyzer ² Algorithm.....	14
2.4 Introduction to VB.net.....	14
2.4.1 Class and Object.....	15
2.5 Software and Robustness Test.....	17
2.5.1 Objectives of Software Test.....	18
CHAPTER 3 METHODOLOGY.....	21
3.1 Procedure Identification.....	21

CHAPTER 4 RESULTS AND DISCUSSION.....	23
4.1 Design and Develop Analyzer2 Software Version 2.....	23
4.2 Results of Robustness Test.....	25
4.2.1 Average Value Test.....	26
4.2.2 High & Low Fluctuation Level Test.....	28
4.2.3 High & Low Fluctuation Period Test.....	33
4.2.4 Spike Test.....	38
4.2.5 Average Deviation Test	41
4.2.6 Moment Correlation Test.....	44
4.2.7 System Test.....	48
 CHAPTER 5 CONCLUSIONS AND RECOMMENDATIONS.....	 49
5.1 Conclusions.....	49
5.2 Recommendations.....	49
 REFERENCES.....	 50
 APPENDICES.....	 52
APPENDIX A Gantt Chart for Final Year Project I.....	53
APPENDIX B Gantt Chart for Final Year project II	54
APPENDIX C: Analyzer ² Software ver2 Interface.....	55

LIST OF FIGURES

Figure 2.1	Typical basic process GC System.....	9
Figure 2.4	An alternative representation of class and object.....	16
Figure 3.1	Flow Chart of Project.....	21
Figure 4.1	Unified Modeling Language (UML) Model.....	23

LIST OF TABLES

Table 1.1	Applications of Physical Property Analysis in Plant.....	7
Table 2.2	Process GC Applications by Industry Sector.....	11

CHAPTER 1

INTRODUCTION

1.1 Background Of Study

Fault prediction monitoring and detection system has been the new way to overcome the schedule maintenance. This system will give all information required to the people to make an estimation of the failure before it happens. By doing so, schedule maintenance can be avoided and only doing the maintenance when it is required. This will save a lot of maintenance cost, reduce the downtime time and achieve efficient process control [2].

A software known as Analyzer² was already been designed to facilitate predictive maintenance and asset management system in PETRONAS Penapisan Terengganu (PPTSB). This project aims to design, develop and perform robustness test on the next generation of Analyzer² software. It is hope that this system can be used not just for refinery only but to all other PETRONAS Operating Units such as petrochemical, LNG (Liquid and Gas) etc. The idea is before an analyzer gets broken, it will demonstrate certain trend behaviour. The software is designed to track the behaviour changes and anticipate it before it fail. 6 pattern recognition algorithms will be identified and applied in this software.

1.2 Problem Statement

Analyzer or Analytical Instruments is one of the most important elements in plant process industry. These instruments are fully utilized in many interesting application for example worker safety, process safety and environment monitoring [3]. Reliability by definition is the probability that an item (component, equipment or system) will operate without failure for a stated period of time under specified conditions [9]. Faulty Analyzer will result in decrease of the reliability of the plant thus will contribute to the downturn of profit and margin for any company. Analyzer² software has been developed and currently being used in PETRONAS Penapisan Terengganu Sdn Bhd (PPTSB). This software has the capability to predict the state of instrument based on its trend behaviour. Although Analyzer² has been working very well, but there's **no test has been conducted to validate the robustness of the software**. Since this software has been identified to be shared throughout the PETRONAS Operating Unit, there's no guarantee that Analyzer² software will be perfectly fit into the system of other plant.

1.3 Objectives and Scope of Study

1.3.1 Objectives

The major goals of this project are:

1. To verify the 6 algorithms used in old version software.
2. To design, develop and enhance Analyzer² Software version 2 by using Visual Basic software.
3. To perform robustness test on Analyzer² software version 2.

1.3.2 Scope of Study

Throughout this project, knowledge and theory learned is much involved.

The scope of study includes:

- Understanding 6 pattern algorithms that are Average Value Algorithm, High Fluctuation Level Algorithm, Low Fluctuation Level Algorithm, High Fluctuation Period Algorithm, Low Fluctuation Period Algorithm, Spike Algorithm, Average Deviation Algorithm and Moment Correlation Algorithm.
- Understanding of using Visual Basic software and related coding used.
- Concept and knowledge in performing robustness test.

CHAPTER 2

LITERATURE REVIEW

2.1 Analytical Instrumentation (Analyzer)

Process measurement is regarded as a system which takes a measured variable from a measuring object as a signal and converts it to another form that is understandable to an observer. The most common variable in the process are pressure, temperature, flow and level variable but there are lots of quality measuring variable such as to measure the pH, conductivity, type of gas etc. This type of measuring device is called analyzer.

Although the development of instrumental techniques for qualitative and quantitative chemical analysis began earlier in the 20th century, this activity was restricted to scientific research laboratories. Early analytical instruments were designed and fabricated by the researches themselves for their own used only. The early, traditional approach to implementing Analyzer was to modify selected laboratory analytical technology to create instrumentation suitable to production plant environment.

Significant efforts were made to obtain a representative sample from the process in such a manner that it would interface with the analytical instrumentation. The history of process Analyzers is best represented by process gas chromatography (GC). The first generation of process GCs were relatively accurate and maintainable, and normally gave analytical results in 10 to 60 min. The recent generation GCs, as with various other process

analyzers today have improved accuracy and increased speed. The next generation GC is expected to approach real-time response, almost like a sensor, with expected high level of accuracy and reliability [6].

Despite of the above progress, analysis instruments in general still require much more care and attention than flow, pressure, temperature and level instruments and few, if any are sufficiently reliable for automatic control purposes. The emergence of on-line process control computers in the 1960s accelerated the application of on-stream analyzers as process analytical data became more crucial to process control strategies.

Analyzer can be divided into 3 categories:

1. Instruments that can measure physical properties of the chain compounds.
2. Instrument that can automate procedures of analytical chemistry
3. Instrument that can measure the quantitative separation of mixture [1].

Instruments that fall in the first two parts are the one that can only measure single component analyzer and the measurement takes place continuously all the time. The third type of analyzer is to measure certain quality in multiple components of mixture but discontinuous in term of its operation.

Some of criteria that been taken into consideration for an analyzer are the Packaging, Safety, Utilities, Ambient Temperature, Analyzer Temperature, Materials of Construction, Standardization, Maintenance and Operability, Sensitivity, Repeatability of reading, Accuracy, Reliability and Data Handling System.

2.1.1 Analyzer Application

Analyzer has played an important role for the safety of the plant. Analyzer is commonly used to monitor the levels of contaminants in a process stream, the environment or the ambient air. In most cases, the leakage of a small amount of a certain contaminant to the process stream can result in a catastrophic explosion. Apart from using Analyzer to protect the plant, it is also widely used for personnel protection. One of the easy examples is H₂S detector that can provide 24 hours protection and alarm capability.

Analyzer also can improve process control of the plant. Objectively, composition is not as difficult to control as flow, for example but the specifications placed on product quality are so stringent that ordinary performance is seldom acceptable. The impurity of a product stream leaving a fractionators, for example may be specified at $1.0 \pm 0.2\%$. It is virtually impossible to regulate flow within $\pm 1\%$ in the unsteady state, yet the composition control is asked to perform five times as well. This is perhaps the greatest single reason why composition control has the distinction of being problem area. Because quality can be measured to 0.1% is apparently reason enough to expect composition to be controlled to the same tolerance [4]. A control loop is only as good as its weakest element. If the final control element is accurate to only $\pm 1\%$, you will never control to $\pm 0.2\%$.

Economic incentive for improved control is most likely to be found where consistent quality of valuable product is important. Many unit operations are conducted at reduced capacity to assure that quality will surpass specifications, even with poor control. Naturally, the rate of payout from these sources will vary directly with the production rate of the plant. Large plants therefore encounter reduced risks [5].

Analyzers have been working very well for different application in industry. Some common application is at Steel Industry, Petrochemical Industry, Pharmaceutical Industry, Electronic Industry, Food and Beverage Industry, Refinery etc. Table 1 below summarizes typical applications for physical property analysis used in a refinery operation [7].

Table 1.1: Applications for Physical Property Analysis in a plant

Measurement	ASTM Method	Common Application
Viscosity		
Capillary	D446	Asphalts
Saybolt	D88	Oils
Distillation		
Atmospheric	D86	Diesel fuel and lighter
Reduced pressure	D1160	Lube oils
Flash		
Tag closed cup (TCC)	D56	Gasoline
Cleveland open cup (COC)	D92	Gasoline
Pensky-Martens closed cup (P-M)	D93	Fuel Oils
Vapor Pressure		
Reid	D323	Gasoline
LPG	D1267	LPG
Absolute	D2551	Heavy Oils
Pour point	D97	Fuel Oils
Freeze Point	D1659	Jet Fuels
Cloud Point	D2500	Clear distillates

Among all the Analyzer, Gas chromatography has been one of the important analytical measurements in plant. So next section will focus more on Gas Chromatography overview and how it works.

2.2 Gas Chromatography

Gas Chromatography (GC) is a common technique employed to separate and measure components of process streams. Since it has thousands of application, this technology is widely used for manufacturing operations of almost every industry.

There are four major areas that have advances in recent years: more substrates available for analytical columns; improvement in sample and column switching valve technology; new and improved detector types and capabilities; and greater utilization of microprocessors (control of the GC hardware, software programming capability and increased data reduction capability). A simple example is data storage and computations for Continuous Emission Monitoring System (CEMS) applications.

2.2.1 Measurement Principles

The main advantage of GC is because of its resolving power, speed and small sample size requirement. The technique used is for separating substances in the vapour state and then quantifying the separated substances. GC combine both quantitative (how much) method and semi qualitative method (what to a limited degree) method. The sample must be vaporizable at the temperature of the GC analyzer oven and in either the liquid or gaseous state when injected into the analyzer.

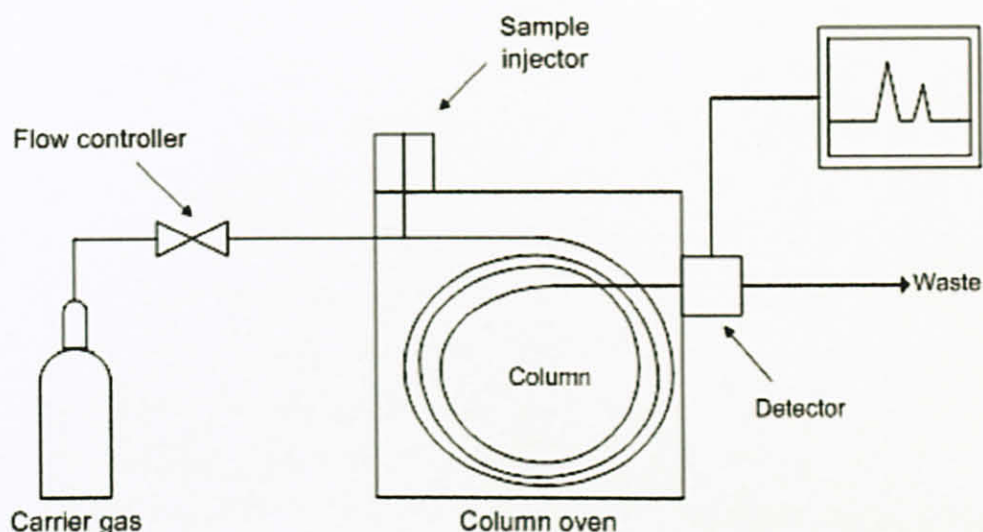


Figure 2.1: Typical basic process GC system.

The main purpose of the sample components prior to the sample injector valve is to supply the suitable homogeneous sample to the sample injector valve. The carrier gas system usually consists of a carrier gas supply (normally gas cylinders of He, N₂, Ar or H₂), single or multiple pressure regulators and also flow controllers. The sample injector valve is purposely used to repeatedly inject the same physical volume (normally 0.25 to 5 μL of liquid or 0.1 to 2 cm^3 of vapour) of sample into the carrier gas stream and onto the column.

The separation column is used to separate the desired components that make up the sample. A GC column does not only separate components by their boiling point like a normal distillation column does. An appropriate column selection can impart a variety of separation criteria (polarity, reactivity, absorptivity, boiling point, etc.). These additional separation criteria are especially useful when trying to separate similar boiling compounds and/or trying to do a fast analysis.

The most basic GC separation system consists of a single column between the inject valve and detector. It is quite common for this single column to be replaced by several columns and valves, and sometimes multiple detectors are used in more complex systems.

The column oven is an insulated and thermally-heated box that consists of two purposes. First, is to assist by providing the stability and/or repeatability for the separation mechanism and allows control of the column system at an optimum (isothermal) temperature to effect the desired separation. In some cases, multiple temperature zones and/or temperature programming are utilized. Normal practise requires a separate isothermal column oven used to house and thermally-stabilize the detector.

The detector allows the concentration of the component and components of interest to be measured. The separation column has separated the desired components from one another and presents them to the detector as isolated "batches" of the carrier and also the component. Several separate type of detectors are available for use in GC work [e.g thermal conductivity (TCD), flame ionization (FID), flame photometric (FPD), electron capture (ECD), hall effect (HED), thermionic conductivity (TIC), flame infrared (FIE) and others]. The GC controller directs the analyzer operation and receives the analogue output of detectors. It converts the analogue output to a digital value that is mathematically reduced to sample concentration by one of many mathematical algorithms, and then outputs the result [10].

2.2.2 Common GC application

Process GC is suitable for any application where the sample components of interest are vaporizable, sufficiently separable on a GC column and measurable on a compatible detector. This table below list several well known applications.

Table 2.2 : Process GC Applications by Industry Sector [11]

Industry	Applications
Refining	Gas fractionation, Catalytic Cracker, Distillation control, Aromatics BTX unit, Catalytic reforming, Butane and pentane isomerisation, Hydrogen Plant, Coker VRU, Gasoline Blanding, Flare Gas recovery, Natural gas BTU, Sulphur in tail gas
Chemicals	Ethylene production, Alcohol production, Distillation control, Reactor product Analysis, Ammonia production, Vinyl Chloride production, Chlorine purity, Polypropylene production, Methyl chloride production, Styrene production, Chlorofluorocarbon production, Silicone production
Safety and Environmental Monitoring	Ambient air (OSHA) monitoring (primary regulated species), Emissions vent monitoring (primarily air toxics), Water pollutants (volatile effluent species), Fenceline air monitors (air toxics and fugitive emissions)
Biotechnology Industries	Fermenter off-gas monitoring
Steel industry	Blast furnace top-gas analysis
Foods	Foods and beverages
Pharmaceutical	Solvent concentration, Ethylene oxide sterilization

2.2.3 *Ranges of variable*

The basic output of a process GC is concentration. The physical output can be of a current, voltage, or serial format. Since GC can measure multiple components, it can be represented a concentration of any value from parts per billion to 100%.

The sample size, the column and the detector can limit the range of output. The detector needs to see a sample concentration in the carrier gas that exceeds its level of detection and is less than the level that results in detector saturation. If appropriate sample is obtained and one component is measured, sample size need to be varied until have enough molecules of the desired components to meet detectors needs. The sample size that normally ideal to trace the components may be too large for major components. Also, Injecting too large a sample into the column can overload it, preventing the normal interaction between the mobile and the stationary phases from taking place. The result is a bad separation.

2.3 Analyzer² Software

Analyzer² is software originally designed by Mr Azrin Sani, Analyzer Engineer from PETRONAS Penapisan Terengganu Sdn Bhd (PPTSB). The main purpose of this software is to facilitate predictive maintenance and also act as a good asset management tool to manage plant measuring instrument.

The Analyzer² algorithm is able to predict the state of an instrument by analyzing the trend behaviour. This is because before a measuring instrument fails, it will exhibit some distinct behaviour such as sudden increase in reading fluctuations, spikes in reading, slow response to plant process and etc. So by detecting a pre-failure behaviour of instruments, it is possible to anticipate a failure before it actually happens.

The final output of Analyzer² algorithm is confidence level indicator that shows how confidence that the analyzer is healthy. The range is between 0% to 100%. If let say the indicator reads 80%, it means that only operator only 80% confidence that the instrument is healthy. In term of predictive maintenance, the confidence level will highlight the performance of instrument in a certain period of time and it should prompt maintenance team to check the instrument if the indicator drops below 95%.

2.3.1 Analyzer² Software Algorithm

Analyzer² are going to detect 6 pattern recognition algorithms that will lead to the behaviour of particular instrument. The 6 patterns recognition are:

- a) Moment Correlation Algorithm
- b) High & Low Fluctuation Level Algorithm
- c) High & Low Fluctuation Period Algorithm
- d) Spike Detection Algorithm
- e) Moving Average Algorithm
- f) Average Deviation Algorithm

2.4 Introductions to Microsoft Visual Basic.NET Programming

The programming language chosen to design and develop of Analyzer² software is Visual Basic.NET Programming. Visual Basic is categorized as Object Oriented Programming (OOP). Programming consists of designing a set of objects that provide advantages at both the management and technical level. Software objects in the program can represent real or abstract entities in the problem domain. It will make the design of the program more natural and it's also easier to get right and easier to understand.

Timothy Budd in his book, Introduction to Object Oriented Programming define that Object Oriented Programming (OOP) is a set of tools and methods that enable software engineers to build reliable, user friendly, maintainable, well documented, reusable software systems that fulfils the requirements of its users[14]. It is claimed that object-orientation provides software developers with new mind tools to use in solving a wide variety of problems.

An object-oriented programming language provides support for the following concepts:

- Classes and Objects
- Inheritance
- Polymorphism and Dynamic binding

2.4.1 Class and Object

Class and object are the main elements in Object Oriented-Programming. Both of them are related to each other. A class can contain variables and methods.

The object encapsulates data (the attribute value that define the object), operations (the actions that are applied to change the attributed of object), other objects, constant (set values), and other related information). Encapsulation here means that all of the information is grouped under same name and can be used as one specification or program component [12].

However, object can be defined as the subset of a class. Class is a generalized description (e.g a template, pattern or blueprint) that describes a collection of similar objects. All objects that exist within a class inherit its attributes and the operations that are available to manipulate the attributes. A superclass is a collection of classes and a sub class is an instance of a class. Taylor uses the notation below to describe a class (and objects derived from a class).

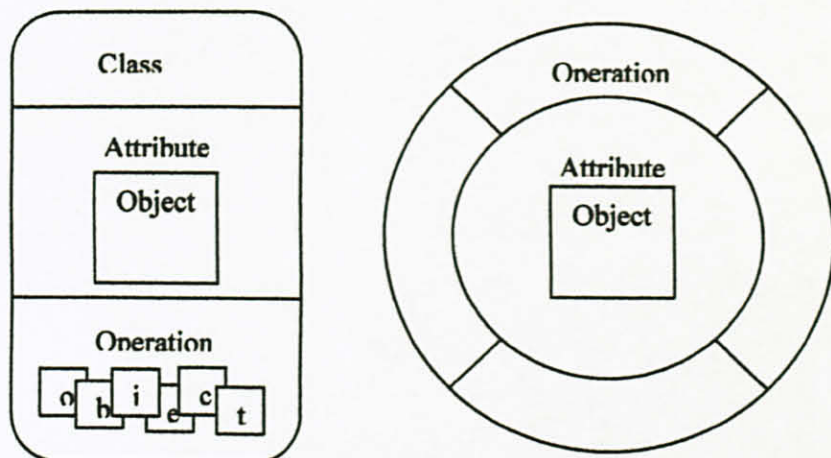


Figure 2.4: An alternative representation of class and object

Taylor explained that “the data abstractions (attributes) that describe the class are enclosed by a “wall” of procedural abstractions (called operations, methods, or services) that are capable of manipulating the data in some way. The only way to reach the attributes (and operate on them) is to go through one of the methods that comprise the wall. Therefore, the class encapsulates data (inside) the wall and the processing that manipulates the data (the method that make up the wall). This achieves information hiding and reduces the impact of side effects associated with change. Since the methods tend to manipulate a limited number of attributes, they are cohesive and because communication occurs only through the methods that comprise “the wall”, the class tends to be decoupled from other elements of a system” [13].

2.5 Software & Robustness Test

Robustness Test is a part of software test as a validation to Analyzer² software that can prove that this software can maintain a consistent results if it is implemented by other PETRONAS Operating Units.

Hetzel and William C. define software testing as any activity aimed at evaluating an attribute or capability of a program or system and determining that it meets its required results [15]. Software testing can be difficult due to the complexity algorithm of the software. Software testing is very crucial to verify that the software meets the technical requirement that have been planned during design and development session. It's also to ensure that the software meet the software designer and user's expectation.

Glen Myers states some rules while conducting the software test as per below [16]:

- Testing is a process of executing a program with the intent of finding an error.
- A good test case is one that has high probability of finding an as-yet undiscovered error.
- A successful test is one that uncovers an as-yet undiscovered error.

Testing principle is to find error but not to show that there's no defect in the software. If testing is conducted successfully, it will uncover certain errors in the software. Testing also demonstrates that software functions appear to be working according to the specification and that performance requirements appear to have been met. In addition, data collected as testing is conducted provides a good indication of software reliability and some indication of software quality [17].

One part of method validation is a robustness test that is performed during method optimization. The function of robustness test is to evaluate the influence of a number of method parameters (factors) on the response prior to the transfer to another testing plant. [8]

2.5.1 Objectives of Software Test

There are 3 main objectives for Software Test:

- a) To improve quality

Quality here means the conformance to the specified design requirement. Being correct, the minimum requirement of quality, means performing as required under specified circumstances. Debugging, a narrow view of software testing, is performed heavily to find out design defects by the programmer. The imperfection of human nature makes it almost impossible to make a moderately complex program correct the first time. Finding the problems and get them fixed, is the purpose of debugging in programming phase.

b) For Verification and Validation [17].

Testing can serve as metrics and heavily used as a tool in the Verification & Validation process. Testers can make claims based on interpretations of the testing results, which either the product works under certain situations, or it does not work. Comparing the quality among different products can be done under the same specification, based on results from the same test.

Quality cannot be tested directly. However, factors that make quality visible still can be tested. Quality has three sets of factors - functionality, engineering, and adaptability. Good testing provides measures for all relevant factors. In the typical business system usability and maintainability are the key factors.

Tests with the purpose of validating the software are called clean tests, or positive tests but it can only validate that the software works for the specified test cases. A finite number of tests cannot validate that the software works for all situations. Only one failed test is sufficient enough to show that the software does not work.

The second is the dirty tests, or negative tests, refer to the tests aiming to show that the software does not work. Very good software must have enough capabilities to survive a significant level of dirty tests. A testable design is a design that can be easily validated, falsified and maintained.

Design for testability is also an important design rule for software development simply because testing is a rigorous effort and requires significant time and cost,

c) For Reliability Estimation [17].

Software reliability has important relations with many aspects of software, including the structure, and the amount of testing it has been subjected to testing can serve as a statistical sampling method to gain failure data for reliability estimation. Software testing can be costly, but not testing software is even more expensive, especially in places that human lives are at stake. Analyzer² software should have a very high reliability because it will be widely used to the other PETRONAS operating unit.

CHAPTER 3

METHODOLOGY

3.1 PROCEDURE IDENTIFICATION

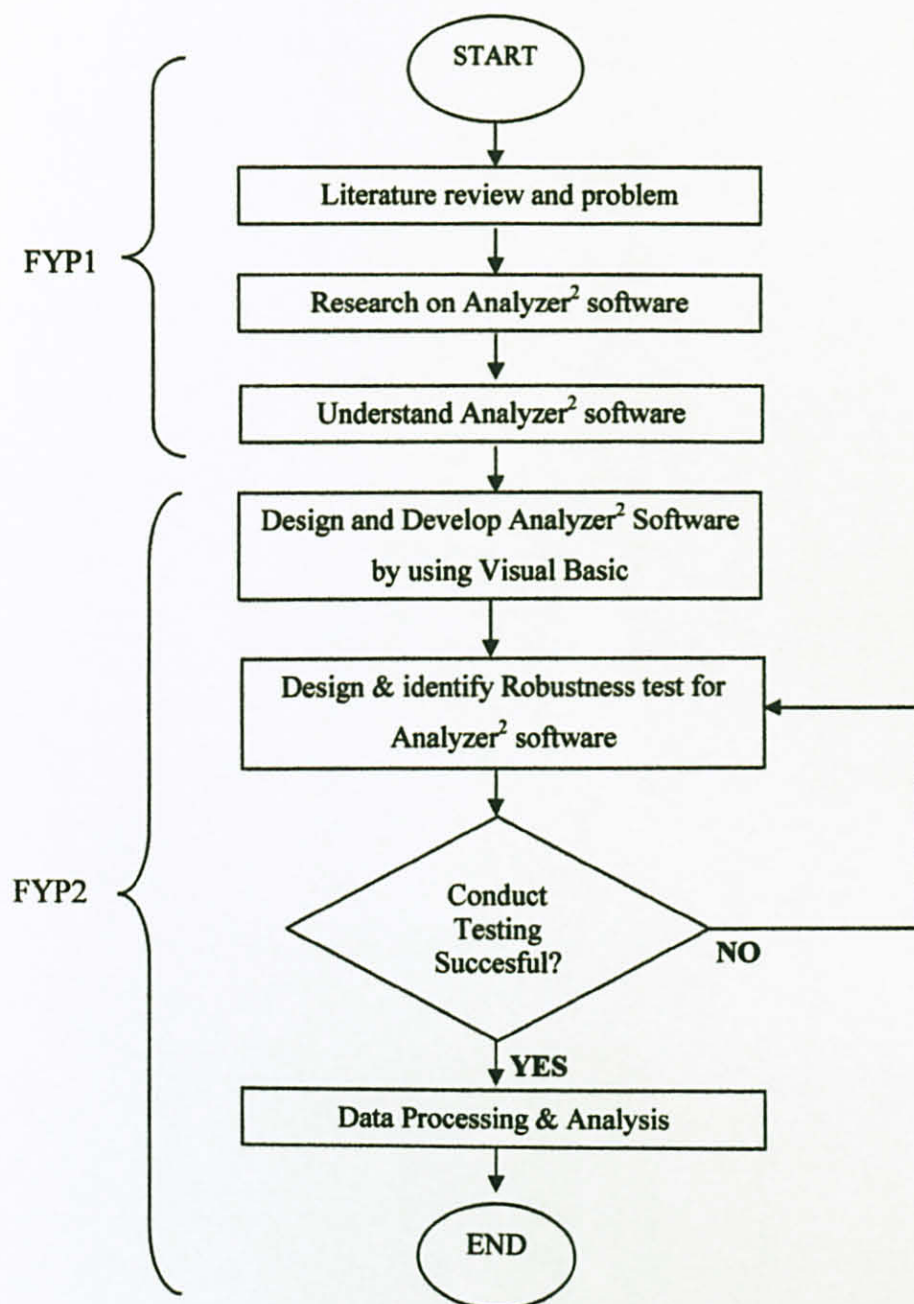


Figure 3.1 : Flow chart of project

The research methodology for the final year project is meant to be conducted in two semesters. In the first semester, literature reviewed and understanding the Analyzer² software. The design and development as well as robustness test for Analyzer² is conducted in second semester.

The workflow of this project started with a research done via library, books and journals for the literature review section. Next is to understand and familiarize with the algorithm of Analyzer² software. Mr Azrin Sani, designer of Analyzer² software has been very helpful for this project. Group Technology Solutions (GTS) has been provided with the older version of Analyzer² software and also some documents related to it. The documents given were very important to understand the Analyzer² software in more details.

Next step is to develop the algorithm in Visual Basic Software Programming. This powerful software has been chosen because it is easier to learn than other language. Besides that, this software suits more for application developing. One whole week has been dedicated to this task which requires a little bit of knowledge in Structured Programming subject and has been successfully completed with the assistance and guidance from both of the engineers.

Next step is to design and conduct robustness test for the software. For this test, a list of input is created for that particular characteristic. The output of from the input will be compared to the expected output. Output should be consistent with the theory of algorithm. Observation will be recorded and discussed.

CHAPTER 4

RESULTS AND DISCUSSION

4.1 Design and Develop Analyzer² Software version 2

Analyzer² software version 2 have been designed and developed with the help from Mr Azrin Sani from PETRONAS Penapisan Terengganu (PPTSB), Ms Wahidah Adzman from Group Technology Solution (GTS) and Lim Hwei Ping. It takes 1 week to complete the class engine for Analyzer² software version 2. Besides that, the other objective of In-House Development is to complete the documentation part and also to discuss regarding the related issues of the software.

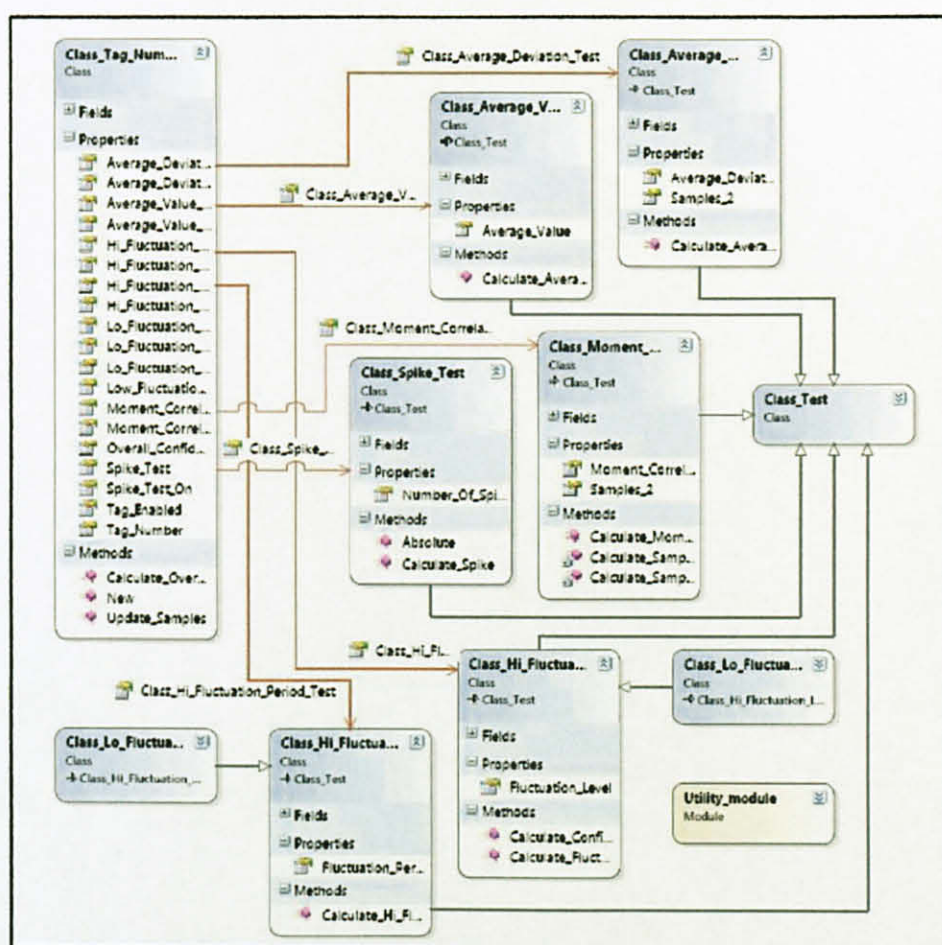


Figure 4.1: Unified Modelling Language (UML) Model for Analyzer² Software

All the class involved in the development can be summarized in Unified Modelling Language (UML). UML is a standardized general-purpose modelling language in the field of software engineering. UML is a notation (graphical language with rules for creating analysis and design methods). UML is a supporting tool for the project. UML will give more understanding on how each class is related to each other. Each class consist of the properties together with the method of execution.

The documentation is done after finishing the design of class and UML. The documentation is basically to give an overview regarding the source code of the class and act as a user manual for this software. This documentation is beneficial to the user who wants to use the software as well as important to the programmer to understand the algorithm of the software.

Meeting with Dr Jafreezal from IT Department has been conducted to discuss regarding the architecture of Analyzer² Software and performing the test.

Some test that can be conducted including:

- a) Functionality test,
- b) Performance test
- c) Load and stress test
- d) Usability test and many more test.

4.2 Results of Robustness Test

2 types of test suggested by Dr Jaffreezal as per below:

a) Validation Test

Validation can be defined in many ways, but a simple definition is that validation succeeds when software functions in a manner that can be reasonably expected by the user. Reasonable expectations are defined in the software requirements specification [17]. In this project, validation test means that each class of the software must be tested according to meet expected output.

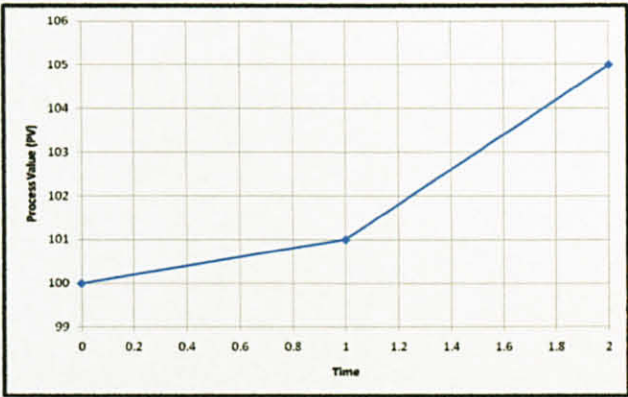
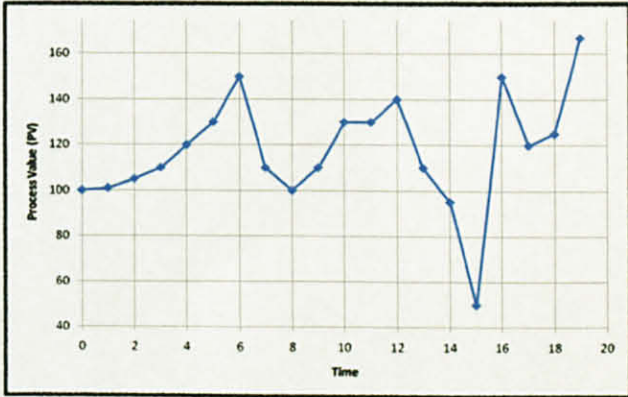
b) System Test

System testing is actually a series of different tests whose primary purpose is to fully exercise the computer based-system. Although each class has different purpose, all work is to verify that all system elements have been properly integrated and perform allocated functions.

Time constrain has limited the testing. It is advised to focus more on Validation Test and continue with System Test if there's enough time left.

4.2.1 Average Value Test

On a stable plant process, a healthy instrument is expected to give reading within certain values. If the reading does not fit into the expected value, the instrument might be faulty.

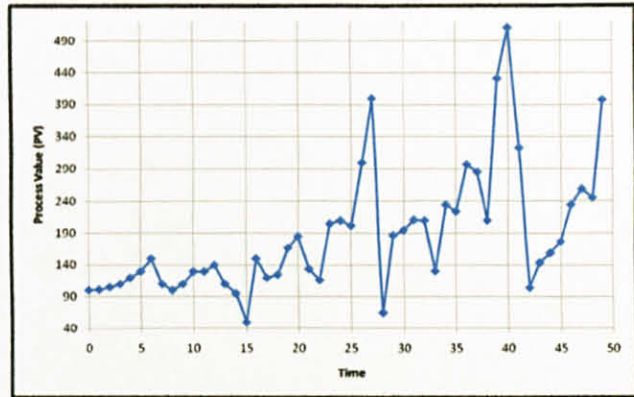
Input Samples	Discussion
<p>Samples1 = {100, 101, 105}</p> <p>Expected Output = 102</p> <p>Actual Output = 102</p>	<ul style="list-style-type: none"> ■ This test is performed to observe the results for a few samples (3 samples size). ■ The algorithm produces the expected result without any problem. 
<p>Samples1 = {100, 101, 105, 110, 120, 130, 150, 110, 100, 110, 130, 130, 140, 110, 95, 50, 150, 120, 125, 167}</p> <p>Expected Output = 117.65</p> <p>Actual Output = 117.65</p>	<ul style="list-style-type: none"> ■ This test is performed to observe the results for a medium samples (20 samples size). ■ The algorithm produces the expected result without any problem. 

Samples1={100, 101,
105, 110, 120, 130, 150,
110, 100, 110, 130, 130,
140, 110, 95, 50, 150,
120, 125, 167, 185, 134,
116, 205, 210, 202, 300,
400, 65, 187, 195, 211,
210, 131, 234, 224, 297,
285, 210, 432, 511, 323,
104, 144, 159, 177, 234,
259, 245, 398}

Expected Output = 186.8

Actual Output = 186.8

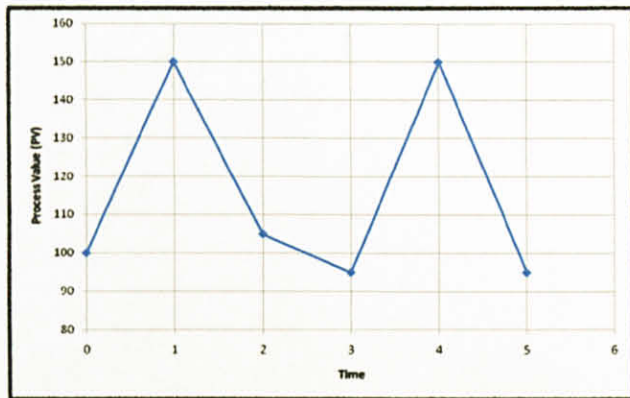
- This test is performed to observe the results for large samples (50 samples size).
- The algorithm produces the expected result without any problem.



4.2.2 High & Low Fluctuation Level Test

Healthy instrument should fluctuate between certain levels. Too heavy or too small fluctuations may give indication that the instrument faulty.

Input Samples	Discussion
<p>Samples1={100, 150, 105, 95, 150, 95}</p> <p>Expected Output = 1.25</p> <p>Actual Output = 53.75</p>	<ul style="list-style-type: none"> ▪ This test is performed to observe the results for few samples (6 samples size). ▪ From the finding, there is a different between expected output and actual output. ▪ After checking the coding, error has been detected inside the High & Low Fluctuation Level Class as per below: <pre> If Total_Peak <= 1 Then In_Class_Fluctuation_Level = 0 Else In_Class_Fluctuation_Level = ((Total_Up_Move - Total_Down_Move) / (Total_Peak)) End If </pre> <ul style="list-style-type: none"> ▪ The modified coding is per below: <pre> If Total_Peak <= 1 Then In_Class_Fluctuation_Level = 0 Else In_Class_Fluctuation_Level = ((Absolute(Total_Up_Move + Total_Down_Move)) / (Total_Peak)) End If </pre>

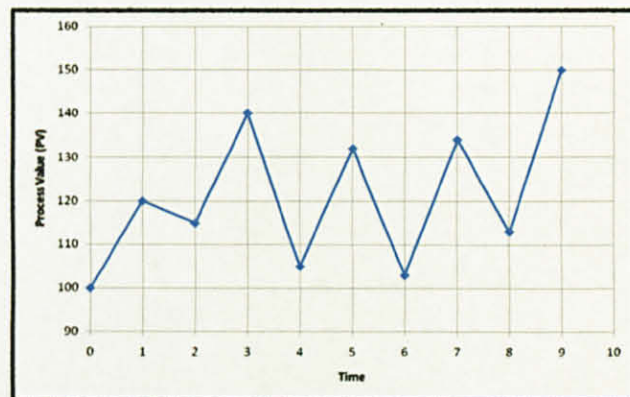


Samples1={100, 120, 115, 140, 105, 132, 103, 134, 113, 150}

Expected Output = 5.555

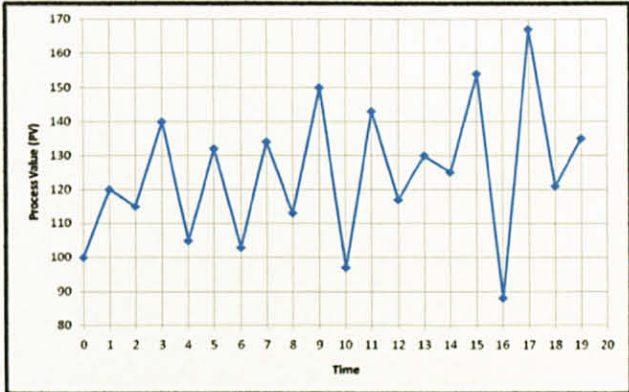
Actual Output = 5.555

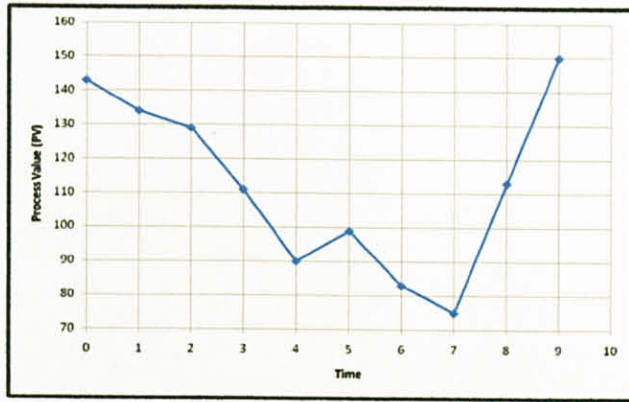
- This test is performed to observe the results for medium samples (10 samples size).
- Heavy fluctuation will results in higher number of output.
- The algorithm produces the expected result without any problem.
- The value of 5.555 indicates a heavy fluctuation level.



Samples1={100, 120, 115, 140, 105, 132, 103, 134, 113, 150, 97, 143, 117, 130, 125, 154, 88, 167,

- This test is performed to observe the results for long samples (20 samples size).
- The algorithm produces the expected result without any problem.

<p>121, 135}</p> <p>Expected Output = 1.842</p> <p>Actual Output = 1.842</p>	<ul style="list-style-type: none"> Long samples (20 samples size) contribute to the smaller fluctuation level output value compared to the same test with 10 samples. The value of 1.842 indicates a normal fluctuation level. 
<p>Samples1={143, 134, 129, 111, 90, 99, 83, 75, 113, 150}</p> <p>Expected Output = 1.75</p> <p>Actual Output = 1.75</p>	<ul style="list-style-type: none"> This test is performed to observe the results when there is a consecutive decrease in samples When there's a consecutive decrease or increase in samples, it will count as one movement only. The value of 1.842 indicates a normal fluctuation level. The algorithm produces the expected results without any problem.

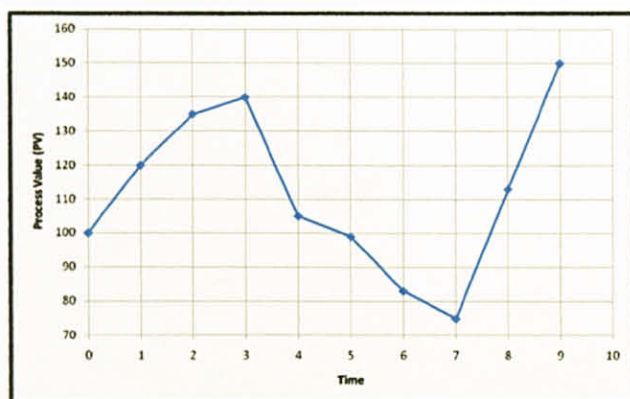


Samples1={100, 120, 135, 140, 105, 99, 83, 75, 113, 150}

Expected Output = 16.667

Actual Output = 16.667

- This test is performed to observe the results when there is a consecutive increase and decrease in samples.
- When there's a **consecutive decrease or increase in samples**, it will count as **one movement only**.
- This test produces the **highest output value** because there are only **3 total movements** resulting from the consecutive increase and decrease.
- The value of 16.667 indicates a very heavy fluctuation level.
- The algorithm produces the expected results without any problem.

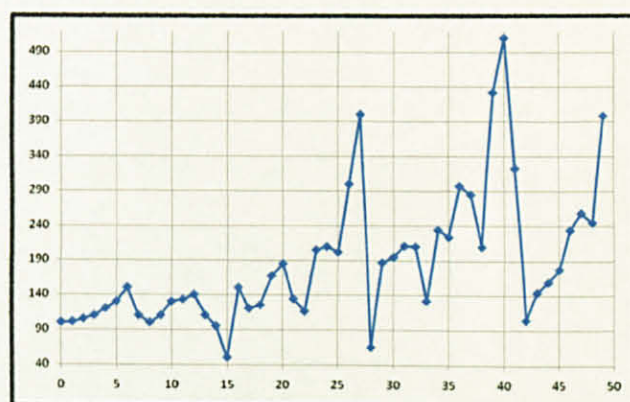


Samples1={100, 101, 105, 110, 120, 130, 150, 110, 100, 110, 130, 133, 140, 110, 95, 50, 150, 120, 125, 167, 185, 134, 116, 205, 210, 202, 300, 400, 65, 187, 195, 211, 210, 131, 234, 224, 297, 285, 210, 432, 511, 323, 104, 144, 159, 177, 234, 259, 245, 398}

Expected Output = 12.95

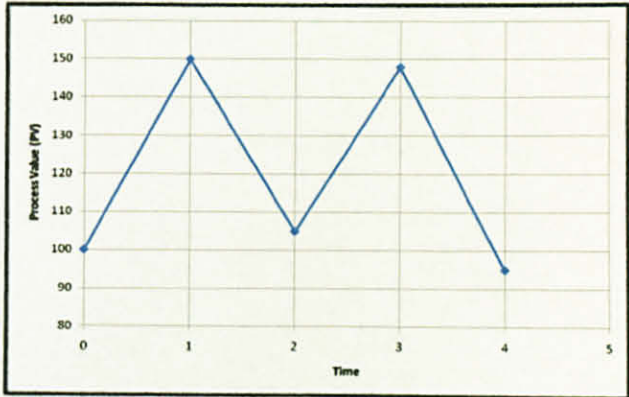
Actual Output = 12.95

- This test is performed to observe the results when there is a consecutive increase and decrease in very long samples (50 samples size).
- This test is a combination of **many consecutive increases and decreases in samples**.
- The value of 12.95 indicates a very heavy fluctuation level.
- The algorithm produces the expected results without any problem.

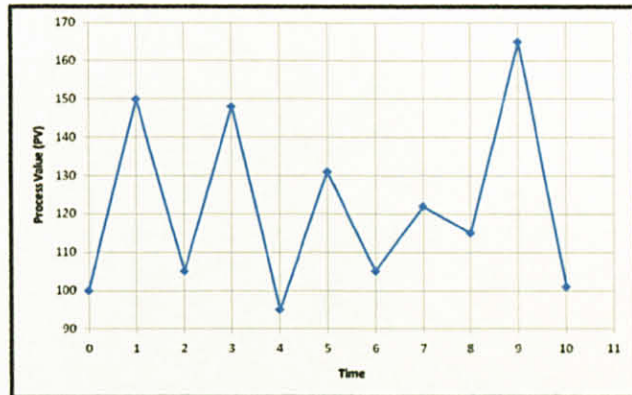


4.2.3 High & Low Fluctuation Period Test

Similar to fluctuation level algorithm, the fluctuation period algorithm analyzes the fluctuation period of instruments. Healthy instruments are expected to fluctuate between certain periods. Too heavy or too small fluctuation periods may indicate that unhealthy instruments.

Input Samples	Discussion
<p>Samples1={100, 150, 105, 148, 95}</p> <p>Expected Output = 2</p> <p>Actual Output = 2</p>	<ul style="list-style-type: none"> ▪ This test is performed to observe the results for few samples (5 samples size). ▪ The value of 2 indicates 2 cycles in one period for this signal. It shows a normal fluctuation period. ▪ The algorithm produces the expected result without any problem. 
<p>Samples1={100, 150, 105, 148, 95, 131, 105, 122, 115, 165, 101}</p> <p>Expected Output = 2</p> <p>Actual Output = 2</p>	<ul style="list-style-type: none"> ▪ This test is performed to observe the results for few samples (11 samples size). ▪ The value of 2 indicates 2 cycles in one period for this signal. It shows a normal fluctuation period.

- The algorithm produces the expected result without any problem.

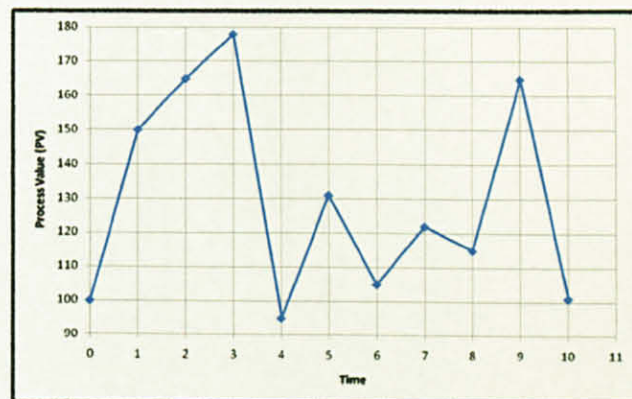


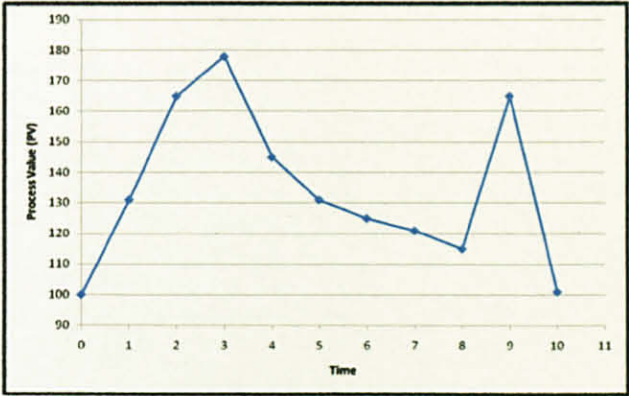
Samples1={100, 150, 165,
178, 95, 131, 105, 122,
115, 165, 101}

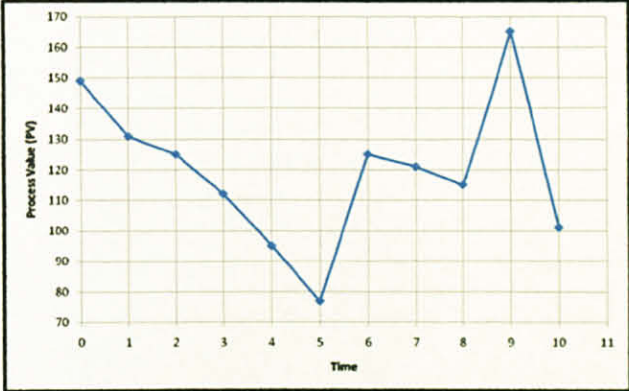
Expected Output = 2.5

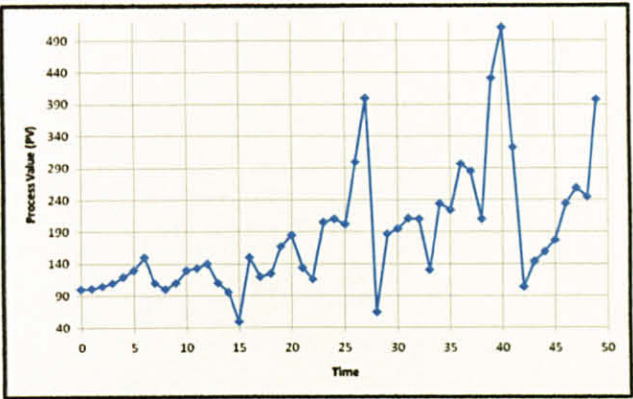
Actual Output = 2.5

- This test is performed to observe the results when there is a consecutive increase in samples.
- When there's a **consecutive increase in samples**, it will count as **one movement only**.
- The value of 2.5 indicates 2.5 cycles in one period for this signal. It shows a normal fluctuation period.
- The algorithm produces the expected results without any problem.



<p>Samples1={100, 131, 165, 178, 145, 131, 125, 121, 115, 165, 101}</p> <p>Expected Output = 5</p> <p>Actual Output = 5</p>	<ul style="list-style-type: none"> ▪ This test is performed to observe the results when there is a consecutive increase and decrease in samples. ▪ When there's a consecutive decrease or increase in samples, it will count as one movement only. ▪ This test produces the highest output value because there are only 4 total movements resulting from the consecutive increase and decrease. ▪ The value of 5 indicates 5 cycles in one period for this signal. It shows a heavy fluctuation period. ▪ The algorithm produces the expected results without any problem. 
<p>Samples1={149, 131, 125, 112, 95, 77, 125, 121, 115, 165, 101}</p> <p>Expected Output = 4</p> <p>Actual Output = 4</p>	<ul style="list-style-type: none"> ▪ This test is performed to observe the results when there is a consecutive decrease in samples. ▪ When there's a consecutive decrease in samples, it will count as one movement only. ▪ The value of 4 indicates 4 cycles in one period

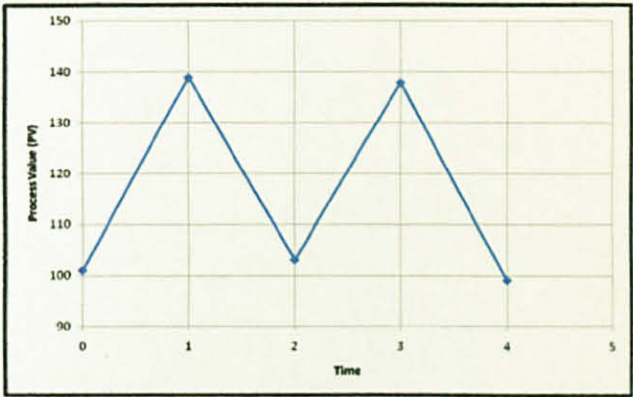
	<p>for this signal. It shows a heavy fluctuation period.</p> <ul style="list-style-type: none"> ▪ The algorithm produces the expected results without any problem. 
<p>Samples1={100, 101, 105, 110, 120, 130, 150, 110, 100, 110, 130, 133, 140, 110, 95, 50, 150, 120, 125, 167, 185, 134, 116, 205, 210, 202, 300, 400, 65, 187, 195, 211, 210, 131, 234, 224, 297, 285, 210, 432, 511, 323, 104, 144, 159, 177, 234, 259, 245, 398}</p> <p>Expected Output = 4.26</p> <p>Actual Output = 4.26</p>	<ul style="list-style-type: none"> ▪ This test is performed to observe the results when there is a consecutive increase and decrease in very long samples (50 samples size). ▪ This test is a combination of many consecutive increases and decreases in samples. ▪ The value of 4.26 indicates 4.26 cycles in one period for this signal. It shows a heavy fluctuation period. ▪ The algorithm produces the expected results without any problem.

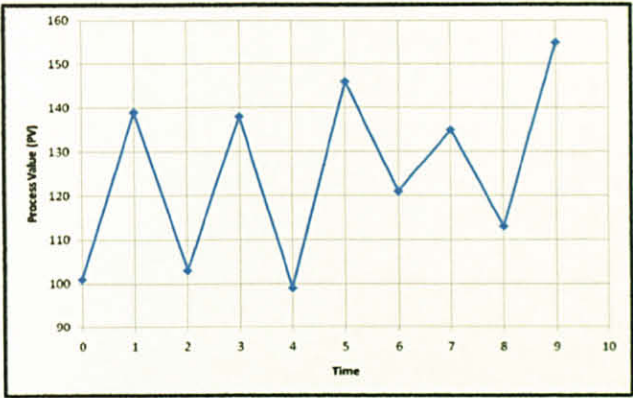


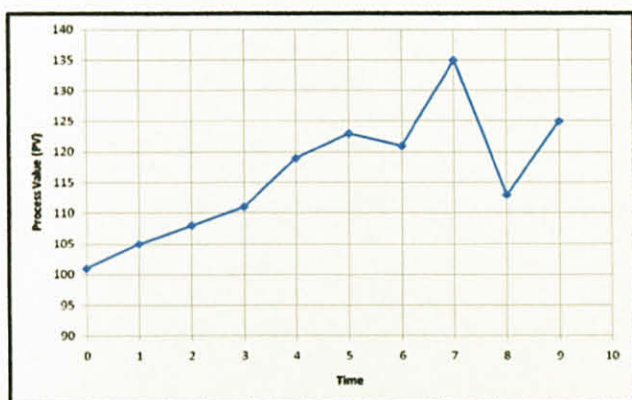
4.2.4 Spike Test

A spike is a sudden increase in an instrument reading. It is a good indication that an instrument is faulty. The algorithm only identified the reading as a spike if the samples do fulfil the 2 conditions:

1. The instrument reading jumps higher than 4 times of long fluctuation level
2. The jump takes place within 0.5 a long fluctuation period.

Input Samples	Discussion
<p>Samples1={101, 139, 103, 138, 99}</p> <p>Expected Output = 4</p> <p>Actual Output = 4</p>	<ul style="list-style-type: none"> ▪ This test is performed to observe the spike results for few samples (5 samples size). ▪ Spike Limit=2 ▪ The value of 4 indicates there are 4 spikes detected in this signal. ▪ The algorithm produces the expected results without any problem. 
<p>Samples1={101, 139, 103, 138, 99, 146, 121, 135, 113, 155}</p> <p>Expected Output = 7</p> <p>Actual Output = 7</p>	<ul style="list-style-type: none"> ▪ This test is performed to observe the spike results for medium samples (10 samples size). ▪ Spike Limit = 24 ▪ Since there's only 7 spike that exceed the spike limit, so the results only give 7 output

	<ul style="list-style-type: none"> ▪ The algorithm produces the expected results without any problem. 
<p>Samples1={101, 105, 108, 111, 119, 123, 121, 135, 113, 125}</p> <p>Expected Output = 1</p> <p>Actual Output = 1</p>	<ul style="list-style-type: none"> ▪ This test is performed to observe the spike results when there is a consecutive increase in samples. ▪ When there's a consecutive increase in samples, it will count as one spike only. ▪ The algorithm produces the expected results without any problem. ▪ Spike Limit = 19.2 ▪ Since there's only 1 spike that exceed the spike limit, so the results yield only 1 spike detected. ▪ The algorithm produces the expected results without any problem.

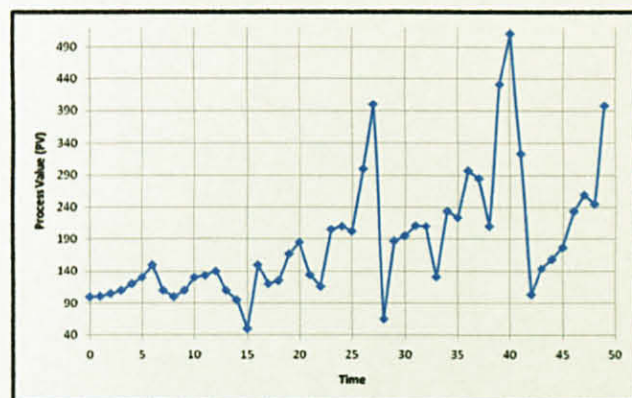


= {100, 101, 105, 110, 120, 130, 150, 110, 100, 110, 130, 133, 140, 110, 95, 50, 150, 120, 125, 167, 185, 134, 116, 205, 210, 202, 300, 400, 65, 187, 195, 211, 210, 131, 234, 224, 297, 285, 210, 432, 511, 323, 104, 144, 159, 177, 234, 259, 245, 398}

Expected Output = 20

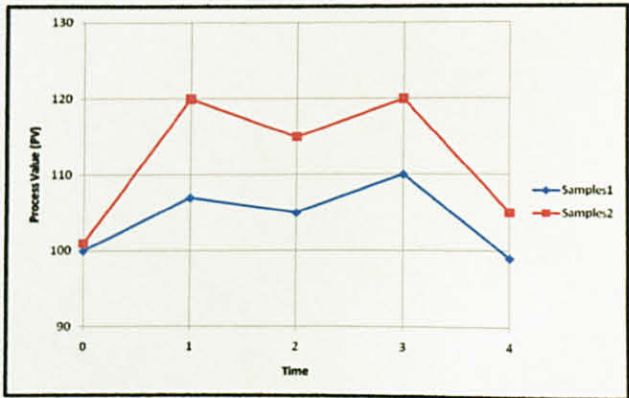
Actual Output = 20

- This test is performed to observe the spike results when there is a **consecutive increase and decrease** in very long samples (50 samples size).
- This test is a combination of **many consecutive increases and decreases in samples**.
- Spike Limit = 51.8
- Since there are only 20 spikes that exceed the spike limit, so the results only give 20 outputs.
- The algorithm produces the expected results without any problem.



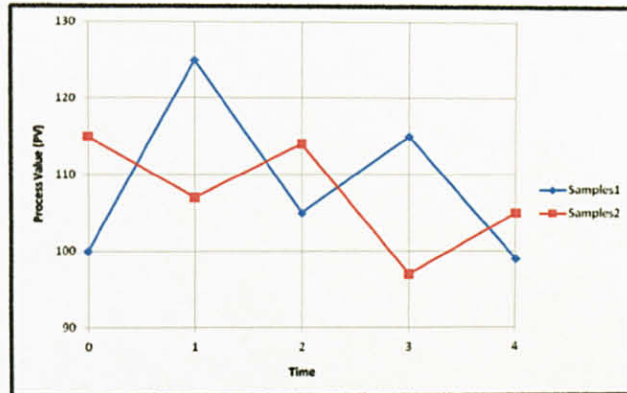
4.2.5 Average Deviation Test

Average Deviation Test calculates the average deviation between 2 identical instruments. 2 similar instruments measuring the same point should always have a standard allowable average deviation.

Input Samples	Discussion																		
<p>Samples1 (blue line) =$\{100, 107, 105, 110, 99\}$</p> <p>Samples2(red line)=$\{101, 120, 115, 120, 105\}$</p> <p>Expected Output = 8</p> <p>Actual Output = 8</p>	<ul style="list-style-type: none">▪ This test is performed to observe the average deviation results for 2 type of samples▪ Since both samples are not so far from each other, the average deviation output value is small.▪ The value of 8 indicates an average different value deviation between these 2 signals.▪ The algorithm produces the expected results without any problem.  <table><caption>Data for Average Deviation Test Graph 1</caption><thead><tr><th>Time</th><th>Samples1 (PV)</th><th>Samples2 (PV)</th></tr></thead><tbody><tr><td>0</td><td>100</td><td>101</td></tr><tr><td>1</td><td>107</td><td>120</td></tr><tr><td>2</td><td>105</td><td>115</td></tr><tr><td>3</td><td>110</td><td>120</td></tr><tr><td>4</td><td>99</td><td>105</td></tr></tbody></table>	Time	Samples1 (PV)	Samples2 (PV)	0	100	101	1	107	120	2	105	115	3	110	120	4	99	105
Time	Samples1 (PV)	Samples2 (PV)																	
0	100	101																	
1	107	120																	
2	105	115																	
3	110	120																	
4	99	105																	
<p>Samples1 (blue line)=$\{100, 125, 105, 115, 99\}$</p> <p>Samples2(red line)=$\{115, 107, 114, 97, 105\}$</p> <p>Expected Output = 13.2</p>	<ul style="list-style-type: none">▪ This test is performed to observe the average deviation results for 2 type of samples▪ The output value is still small since both samples are overlapping to each other▪ The value of 13.2 indicates an average different value deviation between these 2 signals.																		

Actual Output = 13.2

- The algorithm produces the expected results without any problem.



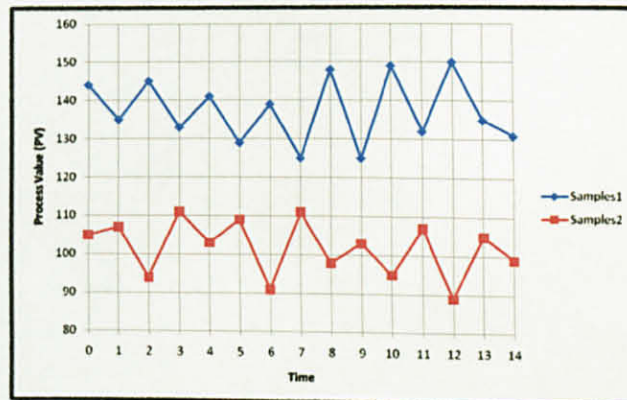
Samples1 (blue line)={144, 135, 145, 133, 141, 129, 139, 125, 148, 125, 149, 132, 150, 135, 131}

Samples2(red line)={105, 107, 94, 111, 103, 109, 91, 111, 98, 103, 95, 107, 89, 105, 99}

Expected Output = 35.6

Actual Output = 35.6

- This test is performed to observe the average deviation results for 2 types of samples with medium samples size and deviate from each sample.
- Both samples are far from each other, resulting in higher output of average deviation value.
- The value of 35.6 indicates a big different average value deviation between these 2 signals.
- The algorithm produces the expected results without any problem.



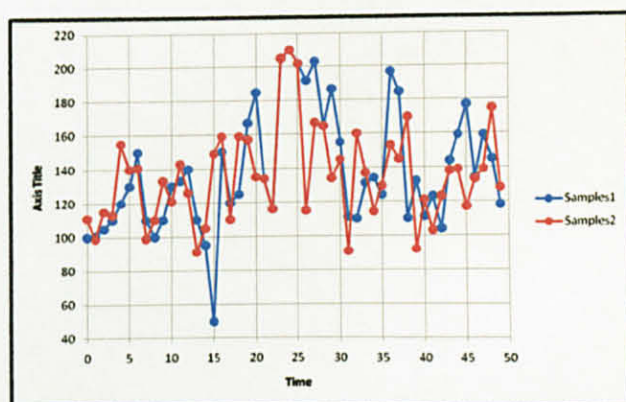
Samples1(blue line)={100, 101, 105, 110, 120, 130, 150, 110, 100, 110, 130, 133, 140, 110, 95, 50, 150, 120, 125, 167, 185, 134, 116, 205, 210, 202, 192, 203, 165, 187, 155, 111, 110, 131, 134, 124, 197, 185, 110, 132, 111, 123, 104, 144, 159, 177, 134, 159, 145, 118}

Samples2(red line)={111, 99, 115, 113, 155, 140, 141, 99, 110, 133, 121, 143, 126, 91, 105, 149, 159, 110, 159, 157, 135, 134, 116, 205, 210, 202, 115, 167, 165, 134, 145, 91, 160, 137, 114, 129, 153, 145, 170, 92, 121, 103, 123, 138, 139, 117, 133, 139, 175, 128}

Expected Output = 21.1

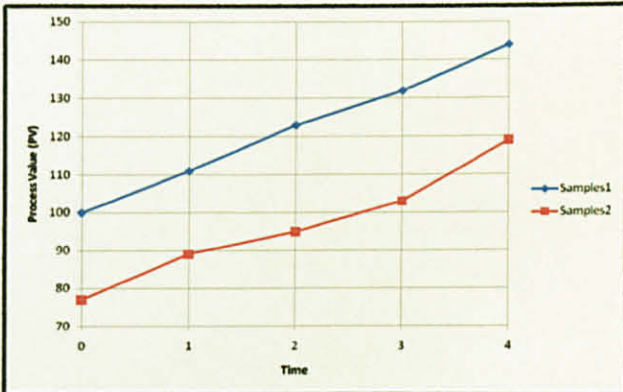
Actual Output = 21.1

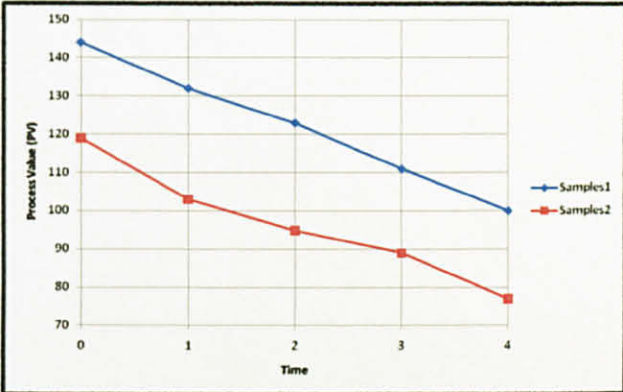
- This test is performed to observe the deviation results in very long samples (50 samples size).
- The output value is still lower than the previous test because each sample are overlapping to each other.
- This test is a combination of **many consecutive increases and decreases in samples**.
- Consecutive increases and decreases in samples will not affect the output.
- The value of 21.1 indicates a big different average value deviation between these 2 signals.
- The algorithm produces the expected results without any problem.

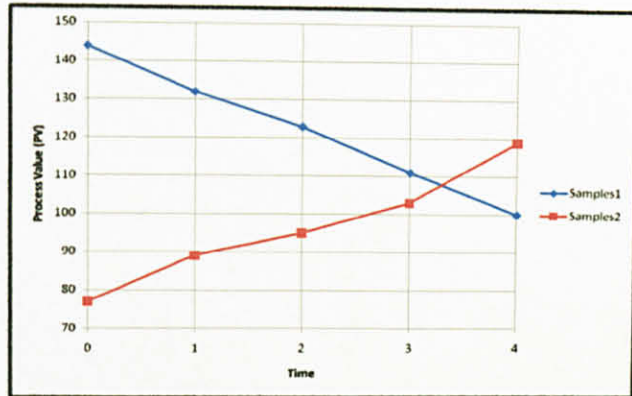


4.2.6 Moment Correlation Test

This test measures the movement correlation between a particular instrument and other process variables.

Input Samples	Discussion																		
<p>Samples1(blue line)= {100, 111, 123, 132, 144}</p> <p>Samples2(red line)= {77, 89, 95, 103, 119}</p> <p>Expected Output = 1.234</p> <p>Actual Output = 1.234</p>	<ul style="list-style-type: none">▪ This test is performed to observe the moment correlation results between 2 sample that increasing from first until last sample.▪ Increase value of samples1 correlate with the increase value of samples2▪ Positive output shows positive linear relationship between samples1 and samples2.▪ The algorithm produces the expected results without any problem.  <table><caption>Data for Moment Correlation Test (Positive)</caption><thead><tr><th>Time</th><th>Samples1 (Blue line)</th><th>Samples2 (Red line)</th></tr></thead><tbody><tr><td>0</td><td>100</td><td>77</td></tr><tr><td>1</td><td>111</td><td>89</td></tr><tr><td>2</td><td>123</td><td>95</td></tr><tr><td>3</td><td>132</td><td>103</td></tr><tr><td>4</td><td>144</td><td>119</td></tr></tbody></table>	Time	Samples1 (Blue line)	Samples2 (Red line)	0	100	77	1	111	89	2	123	95	3	132	103	4	144	119
Time	Samples1 (Blue line)	Samples2 (Red line)																	
0	100	77																	
1	111	89																	
2	123	95																	
3	132	103																	
4	144	119																	
<p>Samples1(blue line)= {144, 132, 123, 111, 100}</p> <p>Samples2(red line)= {119, 103, 95, 89, 77}</p> <p>Expected Output = 1.234</p>	<ul style="list-style-type: none">▪ This test is performed to observe the moment correlation results between 2 samples that decreasing from first sample until last sample.▪ Decrease value of samples1 correlate with the decrease value of samples2																		

<p>Actual Output = 1.234</p>	<ul style="list-style-type: none">Positive output shows positive linear relationship between samples1 and samples2.Positive means that when samples1 changes to certain behaviour, samples1 also follows that same pattern.The algorithm produces the expected results without any problem.  <table><caption>Data for Samples1 and Samples2</caption><tr><th>Time</th><th>Samples1 (PV)</th><th>Samples2 (PV)</th></tr><tr><td>0</td><td>144</td><td>119</td></tr><tr><td>1</td><td>132</td><td>103</td></tr><tr><td>2</td><td>123</td><td>95</td></tr><tr><td>3</td><td>111</td><td>89</td></tr><tr><td>4</td><td>100</td><td>77</td></tr></table>	Time	Samples1 (PV)	Samples2 (PV)	0	144	119	1	132	103	2	123	95	3	111	89	4	100	77
Time	Samples1 (PV)	Samples2 (PV)																	
0	144	119																	
1	132	103																	
2	123	95																	
3	111	89																	
4	100	77																	
<p>Samples1(blue line)= {144, 132, 123, 111, 100}</p> <p>Samples2(red line)= {77, 89, 95, 103, 119}</p> <p>Expected Output = -1.234</p> <p>Actual Output = -1.234</p>	<ul style="list-style-type: none">This test is performed to observe the moment correlation results between 2 samples that is different in trending.Samples1 is decreasing while samples2 is increasing.Decrease value of samples1 does not correlate with the increase value of samples2Negative output shows negative linear relationship between samples1 and samples2.The algorithm produces the expected results without any problem.																		



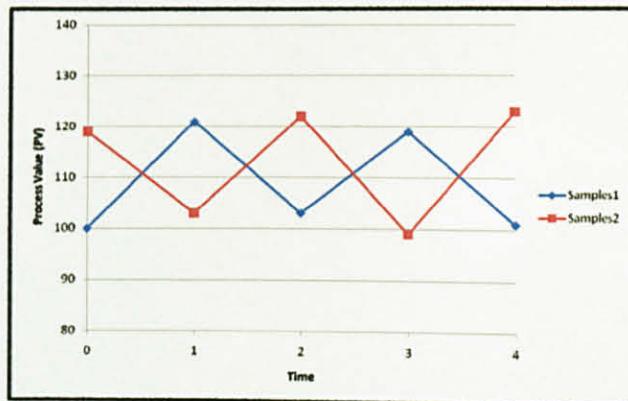
samples1(blue line)=
{100, 121, 103, 119, 101,
120}

samples2(red line)= {119,
103, 122, 99, 123, 101}

Expected Output = -1.16

Actual Output = -1.16

- This test is performed to observe the moment correlation results between 2 samples that fluctuate to each other.
- Negative output shows negative linear relationship between samples1 and samples2 although it is not obvious from the input graph.
- The algorithm produces the expected results without any problem.



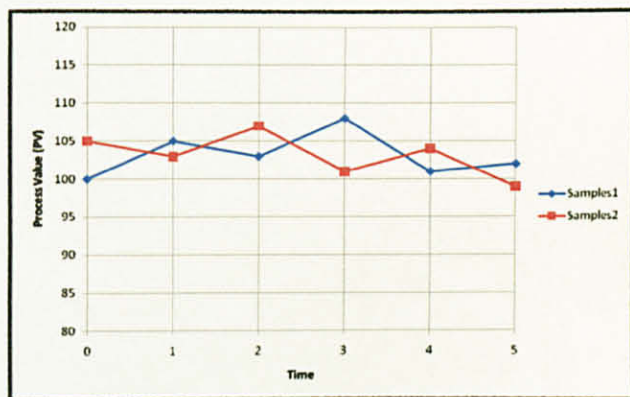
Samples1(blue line)=
{100, 105, 103, 108, 101,
102}

Samples2(red line)= {105,
103, 107, 101, 104, 99}

Expected Output = -0.40

Actual Output = -0.40

- This test is performed to observe the moment correlation results between 2 samples that fluctuate to each other but in small different of values.
- Negative output shows negative linear relationship between samples1 and samples2.
- The negative output value is very small and close to zero because there's no linear relationship between samples1 and samples2.
- The algorithm produces the expected results without any problem.

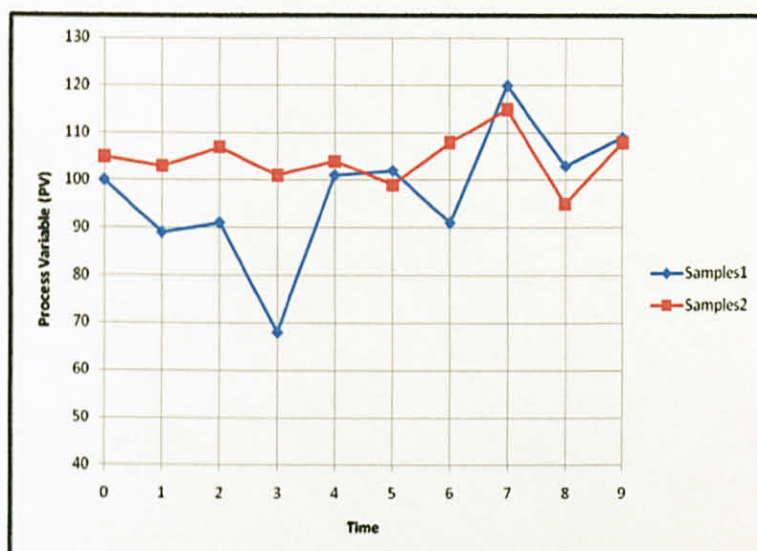


4.2.7 System Test

Input Samples

Samples1 (blue line) = {100, 89, 91, 68, 101, 102, 91, 120, 103, 109}

Samples2 (red line) = {105, 103, 107, 101, 104, 99, 108, 115, 95, 108}



Type of Test	Expected Output	Actual Output	Condition
Average Value	97.4	97.4	Good
High & Low Fluctuation Level	1.125	1.125	Good
High & Low Fluctuation Period	2.25	2.25	Good
Average Deviation	10.5	10.5	Good
Spike	7	7	Good
Moment Correlation	0.442	0.442	Good

CHAPTER 5

CONCLUSIONS AND RECOMMENDATION

5.1 Conclusions

In conclusion, it is fair to say that the project is almost completed. All the three objectives of this project that is to verify the 6 algorithms used in the old version of Analyzer² software, to design, develop and enhance Analyzer² software version 2 by using Visual Basic software and to perform Robustness Test on the software has been achieved.

The first objective was achieved after all 6 algorithms were verified through research from books, journal and discussion with Mr Azrin Sani, designer from PPTSB. The second objective was reached through designing, developing, and enhancing into a new version of Analyzer² Software in Visual Basic. The last objective was attained after performing the Robustness Test of the software. Validation type method has been chosen and each class has been tested with certain condition. It is confirmed that Analyzer² Software version 2 is currently working as expected and can be shared throughout the other PETRONAS Operating Unit.

5.2 Recommendations

Interface of the Analyzer² software version 2 can be improved further to be more user-friendly and easy to be used for all ages of people. More test identification is strongly encouraged to improve the accuracy of Robustness Test. MIMOS certification for this software is highly welcomed because this software needs to be verified by certain official body to increase its credibility and reliability.

REFERENCES

- [1] Helmut J.Maier, 1962, *Analytical Instrumentation for Process Monitoring and Control*, IRE Transactions of Industrial Electronics
- [2] N. Bolf, M. Ivandic, G. Galinec, 2008, *Soft Sensors for Crude Distillation Unit Product Properties Estimation and Control*, 16th Mediterranean Conference on Control and Automation
- [3] L.D. Goettsche, 2005, *Maintenance of Instruments & Systems*, ISA – The Instrumentation, Systems, and Automation Society
- [4] Shinskey, F.G., *Process Control Systems Application/Design/Adjustment*, 2nded. New York: McGraw Hill, 1979, p. 78-79
- [5] Shinskey, F.G., *Process Control Systems Application/Design/Adjustment*, 2nded. New York: McGraw Hill, 1979, p. 191.
- [6] M. V .Koch, K.M VandenBussche and R. W.Chrisman, *Micro Instrumentation for High Throughput Experimentation and Process Intensification – a Tool for PAT*, 2007 Wiley-VCH Verlag GmbH & Co. KGaA, Weinheim
- [7] Sherman, R.E., “Process Analyzers : Vital Control System Components,” *Chemical Processing*, p. 70-78, June 1988

- [8] Bieke Dejaegher, Yvan Vander Heyden, Robustness Test, July 1 2006
- [9] J.D Andrews and T.R.Moss, Reliability and Risk Assessment, Longman Scientific & Technical, 19
- [10] Sherman, R.E and L.J Rhodes, "Analytical Instrumentation", p. 619-631, 1996.
- [11] Graham Currell, Analytical Instrumentation : Performance Characteristics and Quality, John Wiley & Sons Ltd, 2000
- [12] Roger S. Pressman, Software Engineering: A Practitioner's Approach (4th Edition, pp555-556) McGraw Hill. 1997
- [13] Taylor, D.A., Object-Oriented Technology: A Manager's Guide, Addison Wesley, 1990.
- [14] Timothy Budd, Introduction to Object-Oriented Programming, (3rd Edition), 2002
- [15] Hetzel, William C., The Complete Guide to Software Testing, 2nd ed. Publication info: Wellesley, Mass. : QED Information Sciences, 1988.
- [16] Myers, G., The Art of Software Testing, Wiley, 1979
- [17] Roger S. Pressman, Software Engineering: A Practitioner's Approach (4th Edition, pp 450) McGraw Hill. 1997

APPENDICES

APPENDIX A

Suggested Milestone for Final Year Project 1

No.	Detail/ Week	1	2	3	4	5	6	7		8	9	10	11	12	13	14
1	Selection of Project Topic															
2	Preliminary Research Work															
3	Submission of Preliminary Report															
4	Seminar 1 (optional)															
5	Project Work															
6	Submission of Progress Report															
7	Seminar 2 (compulsory)															
8	Project work continues															
9	Submission of Interim Report Final Draft															
10	Oral Presentation (30/11- 4/12)															

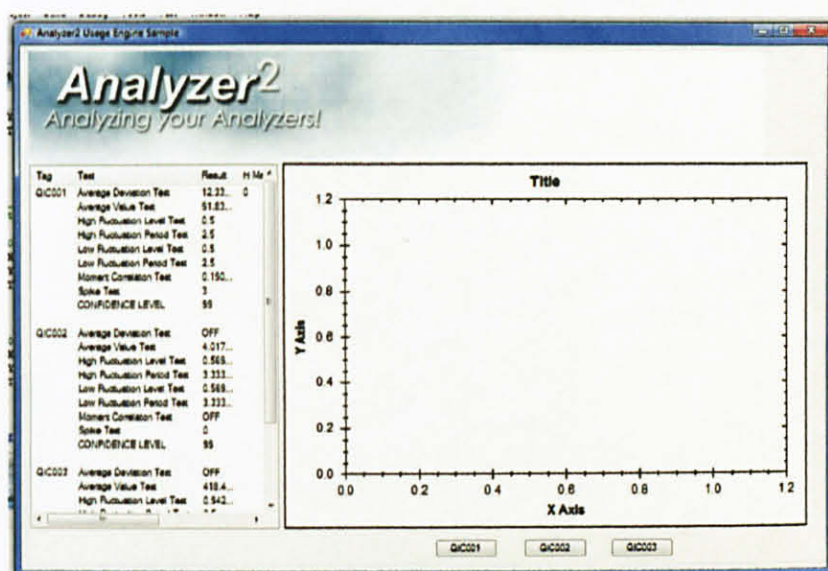
APPENDIX B

Suggested Milestone for Final Year Project 2

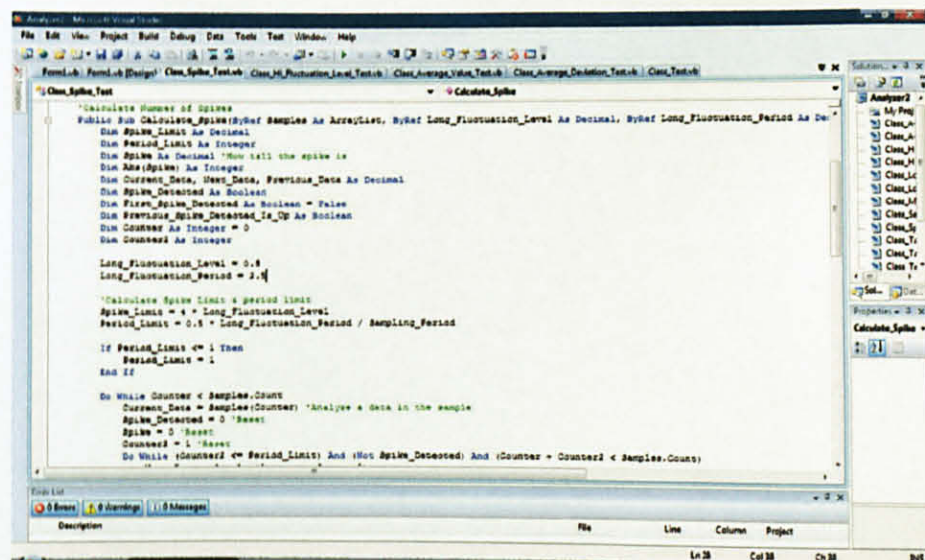
No.	Detail/ Week	1	2	3	4	5	6	7		8	9	10	11	12	13	14
1	Project Work Continue															
2	Submission of Progress Report 1															
3	Project Work Continue															
4	Submission of Progress Report 2															
5	Seminar (compulsory)															
5	Project work continue															
6	Poster Exhibition															
7	Submission of Dissertation (soft bound)															
8	Oral Presentation															
9	Submission of Project Dissertation (Hard Bound)															

APPENDIX C

ANALYZER² INTERFACE



Interface in Analyzer² Software ver2



Example of coding inside Analyzer² Software ver2