UNIVERSITI TEKNIKAL MALAYSIA MELAKA

SYSTEM DEVELOPMENT OF SOFTWARE – SENSOR BASED FORCE AND TORQUE DETECTION

This report submitted in accordance with requirement of the Universiti Teknikal Malaysia Melaka (UTeM) for the Bachelor Degree of Manufacturing Engineering (Robotic and Automation) with Honours.

by

NOR HAFIZAH BT HASSAN

FACULTY OF MANUFACTURING ENGINEERING

2009

DECLARATION

I hereby, declared this report entitled “System Development of Software – Sensor Based Force and Torque Detection” is the results of my own research except as cited in references.

Signature	:	…….
Author’s Name	:	……
Date	:	……

APPROVAL

This report is submitted to the Faculty of Manufacturing Engineering of UTeM as a partial fulfillment of the requirements for the degree of Bachelor of Manufacturing Engineering (Robotic and Automation) with Honours. The member of the supervisory committee is as follow:

(Signature of Supervisor)

……

(Official Stamp of Supervisor)

ABSTRACT

System Development of Software - Sensor Based Force and Torque Detection is the systems that measure the force and torque at one joint. The systems are integrated between computer and electrical signal through a microprocessor module to record and process the data. The input and output for this system can be visualized on a graphic user interface. Besides that, two types of sensor which is Flexiforce sensor and Piezo Vibra Tab Mass sensor that use to detect the force and torque at one joint and the reading of the measurement will record and process at this system. These systems use the database to storage the data and provide several ways to view the data depending upon the need of the user. Visual Basic programming is use to do the user interface of this system and the BASIC Stamp Editor Software use to program the microcontroller. For this system, the data was showing in real-time and the data that had recorded will be portrayed as graph automatically.

ABSTRAK

System Development of Software - Sensor Based Torque and Force Detection ialah sistem yang mengukur kuasa dan daya pada satu sendi. Sistem itu akan menyatupadukan antara komputer dan isyarat elektrik terus ke mikropemproses untuk juan merekod dan memproses data. Input dan output untuk sistem ini boleh dilihat pada antara muka pengguna (user interface). Dua jenis penderia digunakan iaitu Flexiforce dan Piezo Vibra Tab Mass yang digunakan untuk mengesan kuasa dan daya yang terhasih pada satu sendi dan ukuran bacaan akan dicatatkan dan diproses di dalam sistem. Sistem ini menggunakan pangkalan data untuk menyimpan data dan menyediakan ia menggunakan beberapa cara untuk melihat data bergantung pada keperluan pengguna. Pengaturcaraan Visual Basic digunakan untuk membuat antara muka pengguna dan perisian BASIC Stamp Editor digunakan untuk merancang mikropengawal. Untuk sistem ini, data ditunjukkan secara *real-time* dan data yang telah direkod akan dipaparkan sebagai graf secara automatik.

DEDICATION

I would like to dedicate this report to my family and friends, as they have been my source of motivation throughout my “Projek Sarjana Muda (PSM)” period. Although far from home, my parents never failed to keep in touch. They kept me on track, and encouraged me to do my best. Their advice and motivation stopped me from giving up easily and kept me focused while I was done my project. I owe the success of completing my PSM to them and with that I would like to end my dedication with heartfelt thanks for all that they have done to help me.

ACKNOWLEDGEMENTS

I would like to express profound gratitude to Mr. Ahmad Yusairi Bani Hashim as a PSM Supervisor for her invaluable support, encouragement, supervision, ideas, opinions and suggestions throughout this PSM. Her moral support and continuous guidance enabled me to complete my work successfully.

My deepest grace and thank goes to my beloved parents, Mr. Hassan B Salim and Mrs. Hasnah Bt Mat for their love and who gave me an endless supports and patience toward the completion of the dissertation. Their supports and advices have boosted up my spirit to do the best in my final year project. I also wish to thank to my siblings for their support and understanding during my final year project.

Lastly, my greatest gratitude to Faculty of Manufacturing Engineering through the knowledge benefited and to all my friends who contributed in giving me toughness spirit and support for me in completing my final year project whether it is directly or indirectly. The kindness, corporation and support from all of the above mentioned people would always be remembered.

TABLE OF CONTENT

TABLE OF CONTENT

CHAPTER 1

INTRODUCTION

1.1 Introduction

Today system software is very useful tool to allow the parts of a computer to work together. Without the system software the computer cannot operate as a single unit. In the case of measure force and torque, sensors are use to detect the measurement and will operate with software to record and process the data when the reading is taken by the sensor at one joint. It uses the database to storage the data and provides several ways to view the data depending upon the need of the user. In this project of System Development of Software – Sensor: Based Force and Torque Detection, the system is integrated between computer and electrical signal through a microprocessor module to record and process the data. The input and output for this system can be visualized on a graphic user interface. This system is developed using Visual Basic program and the BASIC Stamp Editor Software. For this system, the data was showing in real-time and the data that had recorded will be portrayed as graph automatically.

1.2 Objective

The objectives of this project are:

- To identify the system that can measure the force and torque.
- To design the system integration and programming for user interface, database and microcontroller.
- To develop an interface system and database where the information about the torque and force data were located.
- To take the measurement at one joint using the sensor and display the graphical result at Force and Torque Detection System.

1.3 Scope of Work

Project scopes are important in order to help in the development and the progress of the project. Scope that is covered for this project:

1. This project wills only focusing to make the database and interface from computer to real time.
2. Connect the sensor with Force and Torque Detection System to take the force and torque reading at one joint.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

Literature review discusses published information in a particular subject area, and sometimes information in a particular subject area within a certain time period. A literature review can be just a simple summary of the sources, but it usually has an organizational pattern and combines both summary and synthesis [1].

This chapter will present the literature reviews relevant to this project. It includes software, control circuit by transistor and receiver, electronic components, sensor, and microcontroller. The entire thing in this chapter must be considered to develop this project.

2.2 What is Software Development?

Software development is the set of activities that results in software products. Software development may include research, new development, modification, reuse, re-engineering, maintenance, or any other activities that result in software products [2]. Especially the first phase in the software development process may involve many departments, including marketing, engineering, research and development and general management [3]. The term software development may also refer to computer programming, the process of writing and maintaining the source code.

2.2.1 History of software development

Since the beginning of software development which was somewhere in the late 1940s, various software types have gone through many stages of evolution. It has evolved steadily throughout the ages and applications of various software types have reached heights that were not thought to be possible.

One important factor that boosted the development of software throughout the ages was that new and improved computers were coming out into the market at an unprecedented rate. The development of computer hardware technology demanded that the software be as good, fast and reliable as the hardware itself. During the early ages, a computer was not built to sit on top of a desk; rather they were huge machines and did not have the speed or the reliability of modern computers. The need for faster, more reliable and smaller computers was felt throughout the computing world. As time went on computers began to shrink in size but expanded in speed, reliability and performance. With the introduction of better hardware technology, software developed into new heights. These new software were not only reliable and fast, they were user-friendly too. Hardware vendors gave away system software for free, as hardware could not be sold without the proper software.

Software engineering is a rather relative term when we consider the word "engineering". The first appearance of the two words came about in the 1950s. Many have heard the words civil, computer and electrical engineering, but how engineering actually related to software was a mystery.

The basic problem software engineers had was that one could not see a physical development in the software. It was all done virtually or on paper. So it was a little difficult to develop software without a proper model. The NATO Science Committee sponsored two major software conferences, one in 1968 and the other in 1969. These conferences gave the initial boost required for software engineering and many mark these events as the "official birth period" of software engineering.

The 1960s, 1970s and 1980s brought about the so called software crisis. This time period highlighted many of the problems in software development. Many software projects ran over budget or over schedule. Many software companies either closed down or were bought over by other software companies. Initially the software crisis was defined in terms of productivity, but later it turned out to be defined in terms of quality. This time frame became a very bumpy road for software developers and engineers. Eventually the hard work of many software companies paid off and reviled the path towards a brighter future of software engineering.

With the start of the nineteen-hundred came as never before seen phenomenon called the Internet. The World Wide Web brought out opportunities like never before. Many programmers were hired to maintain websites, which involved handling illustrations, maps, photographs, simple animations, creating and maintaining personal accounts and web spaces, and so many other requirements which needed an alarming amount of programmers. The 21st century brought out some of the best programmers and programs of all time. Software became user friendly and easy to use. Programmers were looking

for easier and better ways to write down codes. Life for both the software engineer and the end user became much, much easier.

Software engineering has come a long way since it was first introduced. Many software engineers might even smell like smoke because they have been through fire and come out into the world to make a bang. Software engineering has truly changed the way how we, as end users and the developers themselves, see and experience the world.[4]

2.2.2 What is Software Development Process?

A software development process is a structure imposed on the development of a software product. Synonyms include software life cycle and software process. There are several models for such processes, each describing approaches to a variety of tasks or activities that take place during the process [5].

The largely growing body of software development organizations implements process methodologies. Many of them are in the defense industry, which in the U.S. requires a rating based on 'process models' to obtain contracts. The international standard for describing the method of selecting, implementing and monitoring the life cycle for software is ISO 12207.

A decades-long goal has been to find repeatable, predictable processes that improve productivity and quality. Some try to systematize or formalize the seemingly unruly task of writing software. Others apply project management techniques to writing software. Without project management, software projects can easily be delivered late or over budget. With large numbers of software projects not meeting their expectations in terms of functionality, cost, or delivery schedule, effective project management appears to be lacking.

2.2.3 Software Development Activity

Software development activities are an activity to develop the software such as planning, design, specification, architecture, implementation, testing and development and deployment and maintenance.

2.2.3.1 Planning

The important task in creating a software product is extracting the requirements or requirement analysis. Customers typically have an abstract idea of what they want as an end result, but not what software should do. Incomplete, ambiguous, or even contradictory requirements are recognized by skilled and experienced software engineers at this point. Frequently demonstrating live code may help reduce the risk that the requirements are incorrect.

Once the general requirements are gleaned from the client, an analysis of the scope of the development should be determined and clearly stated. This is often called a scope document. Certain functionality may be out of scope of the project as a function of cost or as a result of unclear requirements at the start of development. If the development is done externally, this document can be considered a legal document so that if there are ever disputes, any ambiguity of what was promised to the client can be clarified

2.2.3.2 Design

Domain Analysis is often the first step in attempting to design a new piece of software, whether it is an addition to existing software, a new application, a new subsystem or a whole new system. Assuming that the developers (including the analysts) are not sufficiently knowledgeable in the subject area of the new software, the first task is to

investigate the so-called "domain" of the software. The more knowledgeable they are about the domain already, the less work required. Another objective of this work is to make the analysts, who will later try to elicit and gather the requirements from the area experts, speak with them in the domain's own terminology, facilitating a better understanding of what is being said by these experts. If the analyst does not use the proper terminology it is likely that they will not be taken seriously, thus this phase is an important prelude to extracting and gathering the requirements.

2.2.3.3 Specification

Specification is the task of precisely describing the software to be written, possibly in a rigorous way. In practice, most successful specifications are written to understand and fine-tune applications that were already well-developed, although safety-critical software systems are often carefully specified prior to application development. Specifications are most important for external interfaces that must remain stable. A good way to determine whether the specifications are sufficiently precise is to have a third party review the documents making sure that the requirements and Use Cases are logically sound.

2.2.3.4 Architecture

The architecture of a software system or software architecture refers to an abstract representation of that system. Architecture is concerned with making sure the software system will meet the requirements of the product, as well as ensuring that future requirements can be addressed. The architecture step also addresses interfaces between the software system and other software products, as well as the underlying hardware or the host operating system.

2.2.3.5 Implementation, testing and documenting

Implementation is the part of the process where software engineers actually program the code for the project. Software testing is an integral and important part of the software development process. This part of the process ensures that bugs are recognized as early as possible.

Documenting the internal design of software for the purpose of future maintenance and enhancement is done throughout development. This may also include the authoring of an API, be it external or internal.

2.2.3.6 Deployment and maintenance

Deployment starts after the code is appropriately tested, is approved for release and sold or otherwise distributed into a production environment. Software Training and Support is important because a large percentage of software projects fail because the developers fail to realize that it doesn't matter how much time and planning a development team puts into creating software if nobody in an organization ends up using it. People are often resistant to change and avoid venturing into an unfamiliar area, so as a part of the deployment phase, it is very important to have training classes for new clients of your software.

Maintenance and enhancing software to cope with newly discovered problems or new requirements can take far more time than the initial development of the software. It may be necessary to add code that does not fit the original design to correct an unforeseen problem or it may be that a customer is requesting more functionality and code can be added to accommodate their requests. It is during this phase that customer calls come in and you see whether your testing was extensive enough to uncover the problems before customers do. If the labor cost of the maintenance phase exceeds 25% of the prior-

phases' labor cost, then it is likely that the overall quality, of at least one prior phase, is poor. In that case, management should consider the option of rebuilding the system (or portions) before maintenance cost is out of control.

Bug Tracking System tools are often deployed at this stage of the process to allow development teams to interface with customer/field teams testing the software to identify any real or perceived issues. These software tools, both open source and commercially licensed, provide a customizable process to acquire, review, acknowledge, and respond to reported issues.

2.3 Data Acquisition

Data acquisition is concerned with taking one or more analogue signals and converting them to digital form with sufficient accuracy and speed to be ready for processing by a computer. The increasing use of computers makes this an expending field, and it is important that the conversion process is done correctly because information lost at this stage can never be regained, no matter how good the computation [6].

Data acquisition typically involves the conversion of analog signals and waveforms into digital values and processing the values to obtain desired information. The components of data acquisition systems include:

- Sensors that convert physical parameters to electrical signals.
- Signal conditioning circuitry to coerce sensor signals into a form that can be converted to digital values.
- Analog-to-digital converters, which convert conditioned sensor signals to digital values.

Depending on the application, acquired data may be displayed, analyzed, or recorded, or some combination thereof. Data acquisition applications may be controlled by commercial DAQ software or by custom programs developed using various general purpose programming languages such as BASIC or C. Specialized programming languages used for data acquisition include EPICS for building large scale data acquisition systems, LabVIEW, which offers a graphical programming environment, and MATLAB which provides graphical tools and libraries for data acquisition and analysis.

2.3.1 Data Acquisition System

Data acquisition systems, as the name implies, are products and/or processes used to collect information to document or analyze some phenomenon. In the simplest form, a technician logging the temperature of an oven on a piece of paper is performing data acquisition. As technology has progressed, this type of process has been simplified and made more accurate, versatile, and reliable through electronic equipment. Equipment ranges from simple recorders to sophisticated computer systems. Data acquisition products serve as a focal point in a system, tying together a wide variety of products, such as sensors that indicate temperature, flow, level, or pressure. Some common data acquisition terms are shown at Table 2.1 below

Table 2.1: Data Acquisition Terms.

Data Acquisition System	Description
Analog-to-digital converter (ADC)	An electronic device that converts analog signals to an equivalent digital form. The analog-to-digital converter is the heart of most data acquisition systems.
Digital-to-Analog Converter (D/A)	An electronic component found in many data acquisition devices that produce an analog

Digital Input/Output (DIO)	Refers to a type of data acquisition signal. Digital I/O is discrete signals which are either one of two states. These states may be on/off, high/low, 1/0, etc. Digital I/O are also referred to as binary I/O.
Differential Input	Refers to the way a signal is wired to a data acquisition device. Differential inputs have a unique high and unique low connection for each channel. Data acquisition devices have either single-ended or differential inputs, many devices support both configurations.
General Purpose Interface Bus (GPIB)	Synonymous with HPIB (for Hewlett-Packard), the standard bus used for controlling electronic instruments with a computer. Also called IEEE 488 in reference to defining ANSI/IEEE standards.
Resolution	The smallest signal increment that can be detected by a data acquisition system. Resolution can be expressed in bits, in proportions, or in percent of full scale. For example, a system has 12-bit resolution, one part in 4,096 resolutions, and 0.0244 percent of full scale.
RS232	A standard for serial communications found in many data acquisition systems. RS232 is the most common serial communication, however, it is somewhat limited in that it only supports communication to one device connected to the bus at a time and it only supports transmission distances up to 50 feet.
RS485	A standard for serial communications found in many data acquisition systems. RS485 is not as popular as RS232, however, it is more flexible in

	that it supports communication to more than one device on the bus at a time and supports transmission distances of approximately 5,000 feet.
Sample Rate	The speed at which a data acquisition system collects data. The speed is normally expressed in samples per second. For multi-channel data acquisition devices the sample rate is typically given as the speed of the analog-to-digital converter (A/D). To obtain individual channel sample rate, you need to divide the speed of the A/D by the number of channels being sampled.
Single-ended Input (SE):	Refers to the way a signal is wired to a data acquisition device. In single-ended wiring, each analog input has a unique high connection but all channels share a common ground connection. Data acquisition devices have either single-ended or differential inputs. Many support both configurations.

2.4 Programming

Computer programming is the process of writing, testing, debugging/troubleshooting, and maintaining the source code of computer programs. This source code is written in a programming language. The code may be a modification of an existing source or something completely new. The purpose of programming is to create a program that exhibits a certain desired behavior (customization). The process of writing source code often requires expertise in many different subjects, including knowledge of the application domain, specialized algorithms and formal logic [7].

Hartree (1950) explained the process of preparing a calculation for a machine can be broken down into two parts, ‘programming’ and ‘coding’. Programming is the process of drawing up the schedule of the sequence of individual operations required to carry out the calculation [8]. Coding was of course extremely time-consuming before the development of assembler languages, but programming soon became the predominant activity.

Programming is basically a process of translating from the language convenient to human beings to the language convenient to the computer [9].

By the end of the first decade of computing research, the notions of a programmed as an automatic sequence, programming as mathematical specification, and linguistic translation between the two were firmly established. “This sequence of basic operations is called the program and the process of preparing it is called programming” [10].