DEVELOPMENT OF SENSOR USING GRAPHICAL USER INTERFACE

MUHAMAD ZULFIKRI BIN SAIDIN

UNIVERSITY MALAYSIA PAHANG

UNIVERSITI MALAYSIA PAHANG

BORANG PENGESAHAN STATUS TESIS♦

JUDUL: **DEVELOPMENT OF SENSOR USING GHAPHICAL USER INTERFACE**

SESI PENGAJIAN: **2007/2008**

Saya **MUHAMAD ZULFIKRI BIN SAIDIN (850426-07-5351)**
(HURUF BESAR)

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

1. Tesis adalah hakmilik Kolej Universiti Kejuruteraan & Teknologi Malaysia.
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:

78, JLN SERULING EMAS 2,
TMN SERULING EMAS,
14200 SUNGAI BAKAP,
PULAU PINANG.

MR. MUHAMMAD SHARFI BIN NAJIB
(Nama Penyelia)

Tarikh: **26 NOVEMBER 2007**

Tarikh: : **26 NOVEMBER 2007**

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

(Power System)”

Signature: ______________________________

Name: MUHAMMAD SHARFI BIN NAJIB

Date: 26 NOVEMBER 2007

DEVELOPMENT OF SENSOR USING GRAPHICAL USER INTERFACE

MUHAMAD ZULFIKRI BIN SAIDIN

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

Faculty of Electrical & Electronics Engineering

University Malaysia Pahang

NOVEMBER 2007

"All the trademark and copyrights use here in are property of their respective owner. References of information from other sources are quoted accordingly; otherwise the information presented in this report is solely work of the author."

Signature	: ________________________________
Author	: MUHAMAD ZULFIKRI BIN SAIDIN
Date	: 26 NOVEMBER 2007

Specially dedicated to

my beloved family and those people who have guided and inspired me

throughout my journey of education.

ACKNOWLEDGEMENT

First and foremost, I am very grateful to the almighty ALLAH S.W.T for giving me this opportunity to accomplish my Final Year Project.

Firstly, I wish to express my deep gratitude to my supervisor, Mr. Muhammad Sharfi bin Najib for all his valuable guidance, assistance and support all through this work.

Secondly, I wish to thank lecturers and technicians, for their suggestions and support on this project. Their comments on this project are greatly appreciated. My thanks are also to all my friends who have involved and helped me in this project.

Most importantly I extend my gratitude to my parents who have encouraged me throughout my education and I will always be grateful for their sacrifice, generosity and love.

ABSTRACT

A sensor is a device that measures or detects a real-world condition, such as motion, heat or light. When flow sensors are devices used for measuring the flow rate or quantity of a moving fluid or gas. The key in selecting correctly between the many available flow sensors and flow meters is one of the requirements of the particular application. The purpose for this project is to interface the flow sensor with MATLAB GUI. The MATLAB GUI will display the result and the data that will get from the flow sensor. To interface between them the PIC 16F877 and MAX232 will be use. The PIC will convert the analog data to digital data and MAX232 will connect the PIC to serial port at computer. This is to make sure the computer (GUI) will be able to read the data. As a result, flow measurement using GUI is able to display generated signal from the developed flow sensor.

ABSTRAK

Sensor ialah alat yang dapat mengukur keadaan dalam dunia yang nyata ini. Contohnya seperti pergerakan, haba dan cahaya. Manakala sensor aliran ialah sensor yang digunakan untuk mengukur kadar aliran ataupun kuantiti bendalir/cecair atau gas yang melaluinya. Keputusan yang dibuat di dalam memilih diantara banyak sensor aliran mestilah difahami sebetul-betulnya tentang penggunaan yang diperlukannya di dalam aplikasi yang dikhususkan. Tujuan projek ini adalah untuk menghubungkan sensor aliran dengan perisian MATLAB GUI. MATLAB GUI akan menunjukkan hasil dan data yang boleh diperolehi daripada sensor aliran. Untuk menyambungkan sensor dengan perisian GUI, PIC 16F877 dan MAX 232 akan digunakan. Tujuan PIC adalah untuk menukarkan data analog kepada data digital manakala MAX 232 akan menyambungkan PIC kepada serial port(komputer). Ini adalah untuk memastikan computer yakni perisian GUI boleh untuk membaca data bacaan. Secara kesimpulannya, ukuran aliran dengan mengunakn GUI ini mampu untuk mununjukkan data yang dihasilkan daripada pembanggunan sensor ukuran.

TABLE OF CONTENTS

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION

1.1 Overview

This is project about flow sensor with using MATLAB GUI. This project will use few devices that need to be taken into consideration to successfully accomplish this project. The devices that are need to be considered are flow sensor (movement of air or liquid), analog to digital converter (ADC), Peripheral interface controller and graphical user interface using MATLAB GUI.

Flow sensors are devices used for measuring the flow rate or quantity of a moving fluid or gas. The key to selecting correctly between the many available flow sensors and flow meters is a clear understanding of the requirements of the particular application. Measuring the flow of liquids is a critical need in many industrial plants. In some operations, the ability to conduct accurate flow measurements is so important that it can make the difference between making a profit and taking a loss.

A PIC microcontroller chip combines the function of microprocessor, ROM program memory, some RAM memory and input/output interface in one single package which is economical and easy to use. The PIC-Logicator system is designed to be used to program a range of 8, 18, 28 pin reprogrammable PIC microcontrollers which provide a variety of output, digital input and analogue input option to suit school project uses.

A graphical user interface (GUI) is a pictorial interface to a program. A good GUI can make programs easier to use by providing them with a consistent appearance and with intuitive controls like pushbuttons, list boxes, sliders, menus, and so forth. The GUI should behave in an understandable and predictable manner, so that a user knows what to expect when he or she performs an action.

MATLAB is viewed by many users not only as a high-performance language for technical computing but also as a convenient environment for building graphical user interfaces (GUI). Data visualization and GUI design in MATLAB are based on the Handle Graphics System in which the objects organized in a Graphics Object Hierarchy can be manipulated by various high and low level commands. If using MATLAB7 the GUI design more flexible and versatile, they also increase the complexity of the Handle Graphics System and require some effort to adapt to.

1.2 Objective

i. Design MATLAB GUI for flow sensor GUI

Able to create and design GUI using GUIDE in MATLAB software package to make an easier for the user to use. The design in GUI must be user-friendly to make the user understand to use it.

ii. To display a signal that generated by flow sensor through PIC to GUI

To be able display the actual signal that needed for movement liquid or air in MATLAB GUI. The signal that display in MATLAB GUI must be the correct one to make sure the project successfully done.

1.3 Scope of Project

The first element need to be considered for scope of this project is hardware. The main contribution for hardware in this project is Peripheral Interface Controller (PIC). This PIC use to interface between sensor and computer. For the PIC, must design the appropriate program and coding for the PIC and the circuit design to interface with computer using serial port.RS232.

The second element is software that becomes the main part of this project. The software that use in this project is Graphical User Interface Development Environment (GUIDE) in MATLAB software package. This software is to design and create the GUI layout to make a user-friendly for user. For this GUIDE software is divide into two, first is GUI layout design with a consistent appearance and with intuitive controls like pushbuttons, list boxes, sliders, menus, and so forth. And second is for the program M-File, must design and use the right coding to make sure the design in GUI layout is work properly like what is needed.

1.4 Problem Statement

The sensor is able to detect any movement that through it but it's difficult to get the value that had been measure by the sensor. Many sensors have been created to detect any movement but it will not show the value directly. In this development country, the flow meter has been created to show and display the value that had been measured by the sensor. Same for this project, but the different with flow meter is the flow meter show the value at the gauges but with this project the measurement that has been made by the sensor will able to display at the MATLAB program that is GUIDE. The advantages of this GUIDE is it will not only display the value but it will also able to explain the purpose of this program with interesting button and figure and can guide the users to use this program.

1.5 Thesis Organization

This thesis consists of five chapters including this chapter. The contents of each chapter are outlined as follows. Chapter 2 contains a detailed description on the GUI, PIC and the sensor. It will explain the detail about what is GUI and function of PIC and what sensor that had been used. Chapter 3 includes the project methodology. This will explain how the project is organized and the flow of the process in completing this project. Chapter 4 presents the result of the sensor. It will show the result and display that data at MATLAB GUI and also with comparison with oscilloscope. Finally the conclusions for this project are presented in Chapter 5. This chapter also included the future recommendation, costing and commercialization of this project.

CHAPTER 2

LITERATURE REVIEW

2.1 Graphical User Interface (GUI)

2.1.1 Definition of GUI

A graphical user interface (GUI) is a human-computer interface (i.e., a way for humans to interact with computers) that uses windows, icons and menus and which can be manipulated by a mouse (and often to a limited extent by a keyboard as well) [1] [2] [3] [19] [20].

GUIs stand in sharp contrast to command line interfaces (CLIs), which use only text and are accessed solely by a keyboard. The most familiar example of a CLI too many people is MS-DOS. Another example is Linux when it is used in console mode (i.e., the entire screen shows text only) [1].

An icon is a small picture or symbol in a GUI that represents a program (or command), a file, a directory or a device (such as a hard disk or floppy). Icons are used both on the desktop and within application programs. Examples include small rectangles (to represent files), file folders (to represent directories), a trash can (to indicate a place to dispose of unwanted files and directories) and buttons on web browsers (for navigating to previous pages, for reloading the current page, etc.) [1].

Commands are issued in the GUI by using a mouse, trackball or touchpad to first move a pointer on the screen to, or on top of, the icon, menu item or window of interest in order to select that object [1] [2] [3]. Then, for example, icons and windows can be moved by dragging (moving the mouse with the held down) and objects or programs can be opened by clicking on their icons [1] [2] [19].

2.2 MATLAB GUI

2.2.1 Introduction

A graphical user interface (GUI) is a pictorial interface to a program. A good GUI can make programs easier to use by providing them with a consistent appearance and with intuitive controls like pushbuttons, list boxes, sliders, menus, and so forth [2] [3] [4] [20]. The GUI should behave in an understandable and predictable manner, so that a user knows what to expect when he or she performs an action. For example, when a mouse click occurs on pushbutton, the GUI should initiate the action described on the label of the button. This chapter introduces the basic elements of the MATLAB GUIs [2] [3] [4]. The chapter does not contain a complete description of components or GUI features, but it does provide the basics required to create functional GUIs for your programs [2] [19] [20].

Applications that provide GUIs are generally easier to learn and use since the person using the application does not need to know what commands are available or how they work [3] [4] [20]. The action that results from a particular user action can be made clear by the design of the interface [2] [3] [4] [20].

2.2.2 Operation in GUI

A graphical user interface provides the user with a familiar environment in which to work. This environment contains pushbuttons, toggle buttons, lists, menus, text boxes, and so forth [1] [2] [3] [4]. All of which are already familiar to the user, so that he or she can concentrate on using the application rather than on the mechanics involved in doing things. However, GUIs are harder for the programmer because a GUI-based program must be prepared for mouse clicks (or possibly keyboard input) for any GUI element at any time [1] [2] [3]. Such inputs are known as events, and a program that responds to events is said to be *event driven.* The three principal elements required to create a MATLAB Graphical User Interfaces are [2]:-

1. Components. Each item on a MATLAB GUI (pushbuttons, labels, edit boxes, etc.) is a graphical component. The types of components include graphical controls (pushbuttons, edit boxes, lists, sliders, etc.), static elements (frames and text strings), menus, and axes. Graphical controls and static elements are created by the function uicontrol, and menus are created by the functions uimenu and uicontextmenu. Axes, which are used to display graphical data, are created by the function axes [1] [2] [3] [4].

2. Figures. The components of a GUI must be arranged within a figure, which is a window on the computer screen. In the past, figures have been created automatically whenever we have plotted data. However, empty figures can be created with the function figure and can be used to hold any combination of components [2].

3. Callbacks. Finally, there must be some way to perform an action if a user clicks a mouse on a button or types information on a keyboard. A mouse click or a key press is an event, and the MATLAB program must respond to each event if the program is to perform its function. For example, if a user clicks on a button, that event must cause the MATLAB code that implements the function of the button to be executed. The code executed in response to an event is known as a call back. There must be a callback to implement the function of each graphical component on the GUI [2] [3].

2.3 Analog Digital Converter (ADC)

2.3.1 Introduction

An analog-to-digital converter (abbreviated ADC, A/D or A to D) is an electronic circuit that converts continuous signals to discrete digital numbers. The reverse operation is performed by a digital-to-analog converter (DAC) [17]. Typically, an ADC is an electronic device that converts an input analog voltage to a digital number. The digital output may be using different coding schemes, such as binary and two's complement binary. However, some non-electronic or only partially electronic devices, such as rotary encoders, can also be considered ADCs [17].

The resolution of the converter indicates the number of discrete values it can produce over the range of voltage values. It is usually expressed in bits. For example, an ADC that encodes an analog input to one of 256 discrete values (0.255) has a resolution of eight bits, since $2^8 = 256$ [17].

Resolution can also be defined electrically, and expressed in volts. The voltage resolution of an ADC is equal to its overall voltage measurement range divided by the number of discrete values [17].

2.3.2 Flash ADC

This is one of the most common ways of implementing an electronic ADC that is direct conversion ADC. A direct conversion ADC or flash ADC has a comparator that fires for each decoded voltage range. The comparator bank feeds a logic circuit that generates a code for each voltage range. Direct conversion is very fast, but usually has only 8 bits of resolution (256 comparators) or fewer, as it needs a large, expensive circuit. ADCs of this type have a large die size, a high input capacitance, and are prone to produce glitches on the output (by outputting an out-of-sequence code). They are often used for video or other fast signals [17].

2.4 Peripheral Interface Controller (PIC)

2.4.1 Introduction

PIC is a family of Harvard architecture microcontrollers made by Microchip Technology, derived from the PIC1650 originally developed by General Instrument's Microelectronics Division. PICs are popular with developers due to their low cost, wide availability, large user base, extensive collection of application notes, availability of low cost or free development tools, and serial programming (and re-programming with flash memory) capability [5].

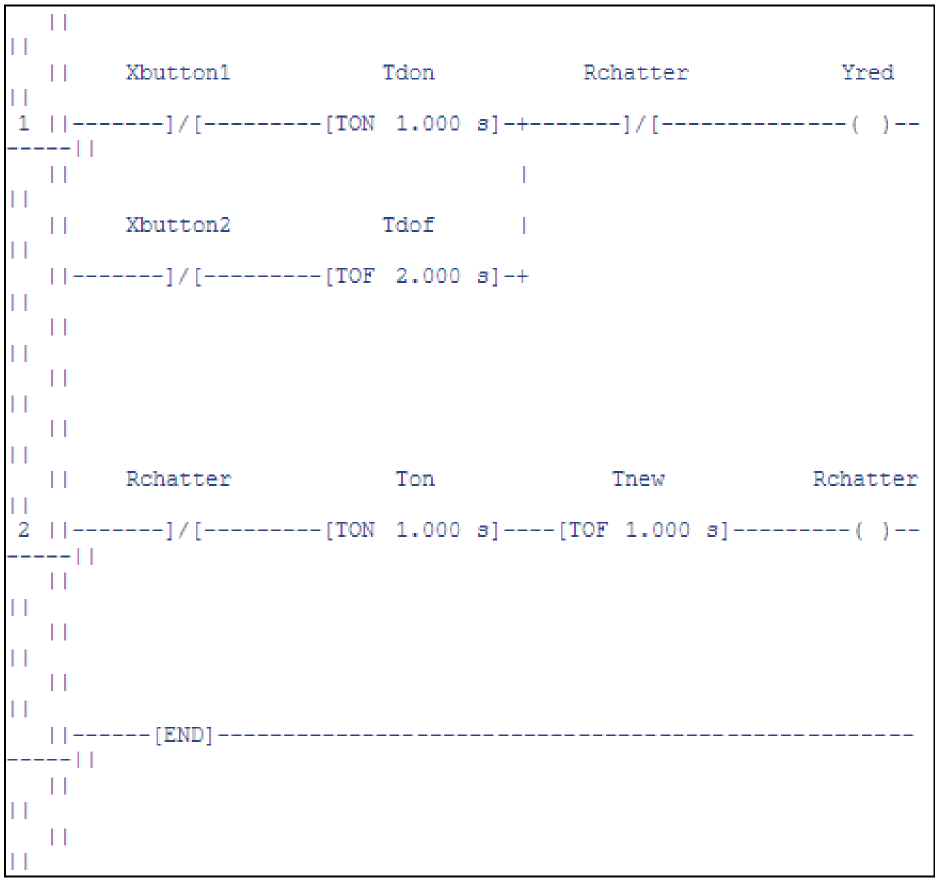
The original PIC was built to be used with GI's new 16-bit CPU, the CP1600. While generally a good CPU, the CP1600 had poor I/O performance, and the 8-bit PIC was developed in 1975 to improve performance of the overall system by offloading I/O tasks from the CPU. The PIC used simple microcode stored in ROM to perform its tasks, and although the term wasn't used at the time, it is a RISC design that runs one instruction per cycle (4 oscillator cycles) [5].

In 1985 General Instruments spun off their microelectronics division, and the new ownership cancelled almost everything — which by this time was mostly out-of-date. The PIC, however, was upgraded with EPROM to produce a programmable channel controller, and today a huge variety of PICs are available with various on-board peripherals (serial communication modules, UARTs, motor control kernels, etc.) and program memory from 512 words to 32k words and more (a "word" is one assembly language instruction, varying from 12, 14 or 16 bits depending on the specific PIC micro family) [5].

Microchip Technology does not use PIC as an acronym; in fact the brand name is PICmicro. It is generally regarded that PIC stands for Peripheral Interface Controller, although General Instruments' original acronym for the PIC1650 was "Programmable Intelligent Computer" [5].]

2.4.2 Programmer PIC

There is much method use to program the PIC. One of those methods is using ladder logic diagram (LDmicro). The LDmicro generates native code for certain Microchip PIC16 and Atmel AVR microcontrollers. Usually software for these microcontrollers is written in a programming language like assembler, C, or BASIC. A program in one of these languages comprises a list of statements [18]. These languages are powerful and well-suited to the architecture of the processor, which internally executes a list of instructions. PLCs, on the other hand, are often programmed in `ladder logic.' A simple program might look like this [18]:



```
   ||
||
   ||    Xbutton1          Tdon          Rchatter          Yred
||
 1 ||-------]/[---------[TON  1.000 s]-+-------]/[--------------(  )--
-----||
   ||                                  |
||
   ||    Xbutton2          Tdof        |
||
   ||-------]/[---------[TOF  2.000 s]-+
||
   ||
||
   ||
||
   ||
||
   ||    Rchatter           Ton            Tnew            Rchatter
||
 2 ||-------]/[---------[TON  1.000 s]----[TOF  1.000 s]---------(  )--
-----||
   ||
||
   ||
||
   ||
||
   ||------[END]---------------------------------------------------
-----||
   ||
||
   ||
||
```

Figure 2.1: Example of Simple Program

TON is a turn-on delay; TOF is a turn-off delay. The --] [--statements are inputs, which behave sort of like the contacts on a relay. The --( )-- statements are outputs, which behave sort of like the coil of a relay [18].

2.5 Sensor

2.5.1 Definition of Sensor

A detector [6]. A device that measures or detects a real-world condition, such as motion, heat or light and converts the condition into an analog or digital representation. An optical sensor detects the intensity or brightness of light, or the intensity of red, green and blue for color systems [7] [8] [9] [10]. Also means sensing element, the basic element that usually changes some physical parameter to an electrical signal [7].

Sensors are normally components of some larger electronic system such as a computer control and/or measurement system. Analog sensors most often produce a voltage proportional to the measured quantity [10]. The signal must be converted to digital form with a {ADC} before the CPU can process it. Digital sensors most often use serial communication such as {EIA-232} to return information directly to the controller or computer through a {serial port} [10].

A sensor is a technological device or biological organ that detects, or senses, a signal or physical condition and chemical compounds [11]. A device that converts physical conditions into information so that the control system can understand the commands and turns it into a signal which can be measured or recorded [12]. An instrument, usually consisting of optics, detectors, and electronics, that collects radiation and converts it into some other form suitable for obtaining information. This may be a certain pattern (an image, a profile, etc.), a warning, a control signal, or some other signal [13].

2.5.2 Flow Sensor

A flow sensor is a device for sensing the rate or quantity of fluid flow whether it be a gas, steam , liquid or solid [14] [15] [16]. The flow sensor directory will enable you to source single-point sensors as well as multi-point sensors [16]. Flow sensor configurations are available for use in liquids or gases with flow rates from ultra low flow sensing to fast transient flow sensors [14] [15] [16]. The flow sensor directory prides itself by the fact it tries to list only quality products, from well known flow sensor manufacturers with worldwide sales support [16].

The key to selecting correctly between the many available flow sensors and flow meters is a clear understanding of the requirements of the particular application. Measuring the flow of liquids is a critical need in many industrial plants. In some operations, the ability to conduct accurate flow measurements is so important that it can make the difference between making a profit and taking a loss [16].

With most fluid flow sensors, the flow rate is determined directly or inferentially by measuring the liquid's velocity or the change in kinetic energy. Velocity depends on the pressure differential that is forcing the liquid through a pipe or conduit. Because the pipe's cross-sectional area is known and remains constant, the average velocity is an indication of the flow rate [16].

Normally a flow sensor is the sensing element used in a flow meter, or flow logger or a flow data logging device to record the flow of fluids. [14] [15]. The flow sensor can normally measure whether velocity, flow rate or totalized flow of fluids flowing through them [14] [15] [16]. Flow sensors are sometimes related to sensors called velocimeters that measure speed of fluids flowing through them, these use units like ft/sec [14] [15] [16]. A very basic relationship for determining the fluid's flow rate in such cases is [16]:

$Q = V \, X \, A$; Where

Q = liquid flow through the pipe; V = average velocity of the flow;
A = cross-sectional area of the pipe.

Other factors that affect flow rate include the liquid's viscosity, density and temperature. Some other factors may be considered such as frictional forces and pipe configurations [16].

There are three basic types of flow sensors and flow meters. Mass flow sensors measure flow rate in terms of the mass of the fluid substance and have units such as lbs/min. Volumetric flow sensors measure flow rate in terms of how much of the material is flowing and use units like mol/min [16]. Velocity flow sensors measure flow rate as in terms of how fast the material is moving. These use units like ft/sec [14] [15] [16]. Critical specifications for flow sensors and flow meters are the measuring range, what type of medium and measurement is to be used, and the operating temperature and pressure ranges [14] [16].

The most common types of Flow sensors are designed to measure the flow of media through pipes, hoses and systems. They can be classified into three categories [16]:

I. Mass flow sensors

   - Measure flow rate in units of mass flow, for example, lbs/min [16].

II. Velocity flow sensors

   - Measure flow rate as in units of velocity, for example, ft/sec [14] [15] [16].

III. Volumetric flow sensors.

   - Measure flow rate in units of volumetric flow, for example, mL/min [16].

Most flow sensors are designed to handle a single style of media, while a few are designed to provide multimedia measurements. Specific are designed as air flow sensors and other gas flow sensors, water flow sensors and other liquid flow sensors, or solid flow sensors [16].

In addition to the main classification, the flow sensor technology can be based on such things as light, heat, electromagnetic properties, ultrasonic and many other technologies in a wide spectrum. Some of the most common types of flow sensor technologies are magnetic flow sensors, turbine flow sensors and ultrasonic flow sensors. [15] [16]. Ultrasonic flow sensors use sound frequencies above audible pitch to determine flow rates. They can be either Doppler Effect sensors or Time-of-Flight sensors [14] [16].

Doppler flow sensors measure the frequency shifts caused by fluid flow [14] [16]. The frequency shift is proportional to the liquid's velocity. Time of flight sensors use the speed of the signal traveling between two transducers that increases or decreases with the direction of transmission and the velocity of the fluid being measured [16].

Turbine flow sensors measure the rate of flow in a pipe or process line via a rotor that spins as the media passes through its blades. The rotational speed is a direct function of flow rate and can be sensed by magnetic pick-up, photoelectric cell, or gears [16].

Magnetic flow sensors apply Farraday's law to measure liquid flow. The sensor contains two electrodes that produce a magnetic field when energized. When a conductive liquid passes through the electrodes in the flow meter, a voltage is induced. The voltage is proportional to the electric field strength, diameter of the pipe, and flow velocity [16].

A fluid dynamics problem is easily solved (especially in non-compressible fluids) by knowing the flow at all nodes in a network [14]. Alternatively, pressure sensors can be placed at each node, and the fluid network can be solved by knowing the pressure at every node [14] [16]. These two situations are analogous to knowing the voltages or knowing the currents at every node (noncompressible fluid being conserved in the same manner as Kirchoff's current or voltage laws, in which conservation of fluid is analogous to conservation of electrons in a circuit).Flow meters generally cost more than pressure sensors, so it is often more economical to solve a fluid dynamics network monitoring problem by way of pressure sensors, than to use flow meters [14] [16].

CHAPTER 3

METHODOLOGY

3.1 Introduction

This chapter presents the methodology of this project. It describes on how the project is organized and the flow of the steps in order to complete this project. The methodology is diverged in two parts, first is software with the main part using MATLAB GUI. Design the layout GUI to display the result from the sensor. The second part is software with using peripheral interface controller (PIC) to interface with computer.

3.2 Methodology

There is few mains method in order to develop this project. Before interface the main parts hardware and software. The PIC can be simulating with using PIC simulator IDE program. This program can simulate the hardware part before the real project is developing to interface with MATLAB GUI. The ladder logic diagram (Ldmicro) is use to program the PIC. This program also can simulate before burn to real PIC. The figure below show the diagram and flow chart for this project.

3.2.1 Project Diagram

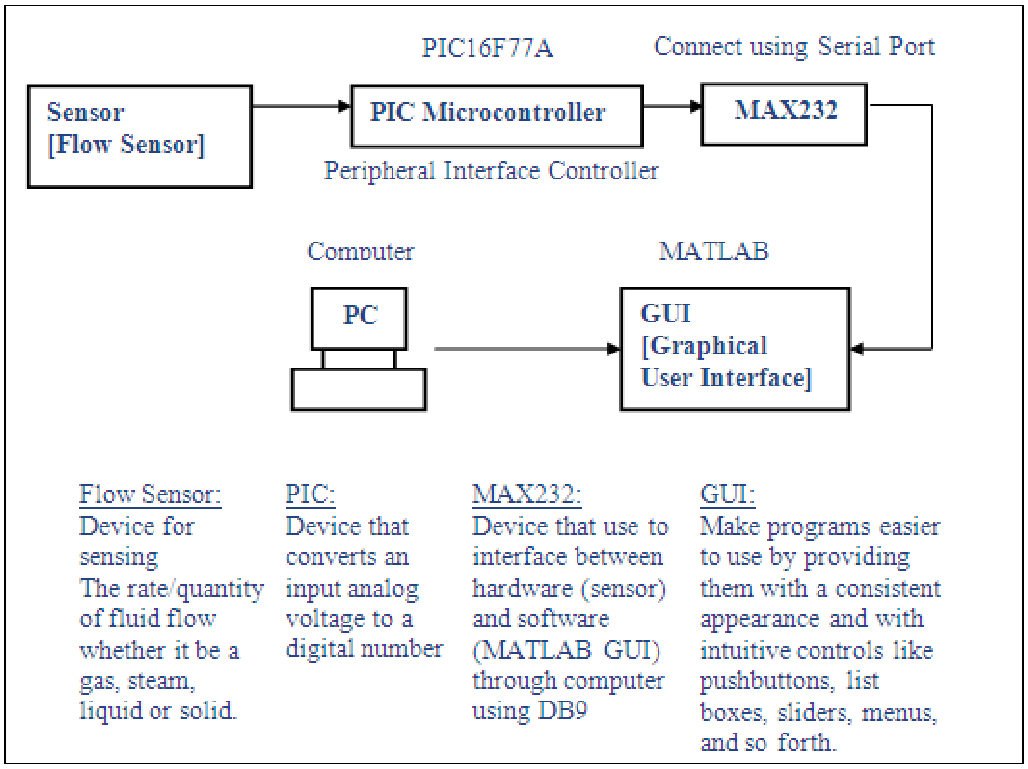

PIC16F77A
Connect using Serial Port
Sensor [Flow Sensor]
PIC Microcontroller
MAX232
Peripheral Interface Controller
Computer
MATLAB
PC
GUI [Graphical User Interface]
Flow Sensor: Device for sensing The rate/quantity of fluid flow whether it be a gas, steam, liquid or solid.
PIC: Device that converts an input analog voltage to a digital number
MAX232: Device that use to interface between hardware (sensor) and software (MATLAB GUI) through computer using DB9
GUI: Make programs easier to use by providing them with a consistent appearance and with intuitive controls like pushbuttons, list boxes, sliders, menus, and so forth.

Figure 3.1: Simplified Block Diagram

From this project, the flow sensor will be an input and the output it will show at MATLAB GUI. The flow sensor will detect movement of liquid or air and from that the sensor will capture the data about that movement and produced in analog signal. And it will go through PIC. The PIC is to convert the input analog to digital number. From that the PIC will connect to MAX232 and the MAX232 will connect to the computer with communication port using DB9. It will connect to MATLAB GUI through serial port of computer. The PIC will transfer the data that get from the sensor via ADC to MATLAB GUI and from that the data will show in MATLAB GUI. It will show the data that we need from the movement of air or liquid like waveform, quantity, velocity and so forth.

3.2.2 Flow Chart of Project

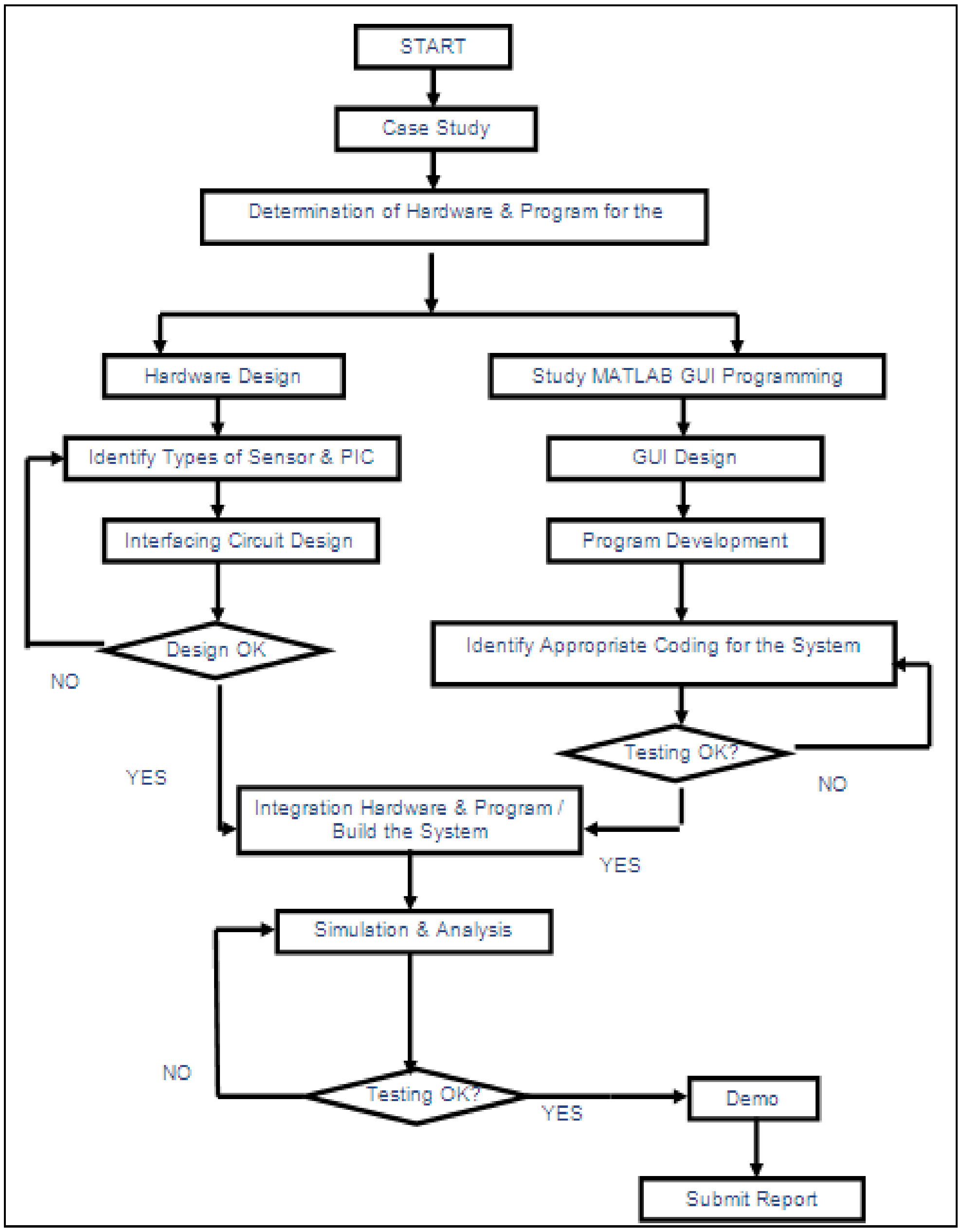

START
Case Study
Determination of Hardware & Program for the
Hardware Design
Study MATLAB GUI Programming
Identify Types of Sensor & PIC
GUI Design
Interfacing Circuit Design
Program Development
Design OK
NO
YES
Identify Appropriate Coding for the System
Testing OK?
NO
YES
Integration Hardware & Program / Build the System
Simulation & Analysis
NO
Testing OK?
YES
Demo
Submit Report

Figure 3.2: Flow Chart

From the flow chart above after get the topic of the project and go to case study to find more related information and to deep knowledge about the project. Find the information whether at internet, book or anything else that is related to the topic. After that, define the part of the project and divide it into two parts. The first part is about the hardware. First is defining the hardware that want to use after that design the circuit for this hardware and interfacing it. After done with interfacing the circuits that have design test it whether is okay or not okay. If not okay redesign the circuit and try to troubleshoot the circuit until the circuit have function correctly.

The second part for this project is about the software. For this project the software that has to use is MATLAB GUI. First, study about the software programming and understand how to use it. For this software has divide by two parts, first is GUI layout design with a consistent appearance and with intuitive controls like pushbuttons, list boxes, sliders, menus, and so forth. And second is for the program M-File, must design and use the right coding to make sure the design in GUI layout is work properly like what is needed. After the two parts have done, test it to make sure the software that has been design is work properly. Is not, identify the problem and overcome it.

After the hardware and software part have work properly, interface the two of this part. Simulate and testing it whether is okay or not. And troubleshoot this part if not okay until get the satisfied result. After the testing is work properly and correctly, finally these projects have done and submit the thesis about this project.

3.2.3 Creating GUI with Guide

This is the main part of this project. GUIDE, the MATLAB graphical user interface development environment, provides a set of tools for creating graphical user interfaces (GUIs). These tools simplify the process of laying out and programming GUIs.

This tool allows a programmer to layout the GUI, selecting and aligning the GUI components to be placed in it. Once the components are in place, the programmer can edit their properties: name, color, size, font, text to display, and so forth. When guide saves the GUI, it creates working program including skeleton functions that the programmer can modify to implement the behavior of the GUI. When guide is executed, it creates the Layout Editor as shown in Figure 3.3. The large white area with grid lines is the layout area, where a programmer can layout the GUI. A user can create any number of GUI components by first clicking on the desired component, and then dragging its outline in the layout area. The top of the window has a toolbar with a series of useful tools that allow the user to distribute and align GUI components, modify the properties of GUI components, add menus to GUIs, and so on. There are few basic steps required to create a MATLAB GUI.

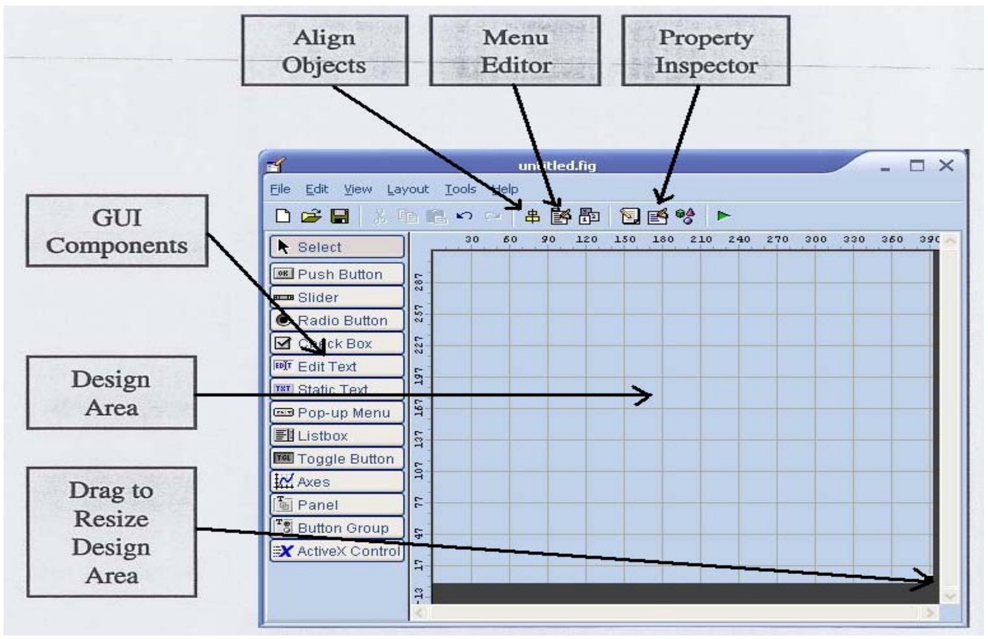
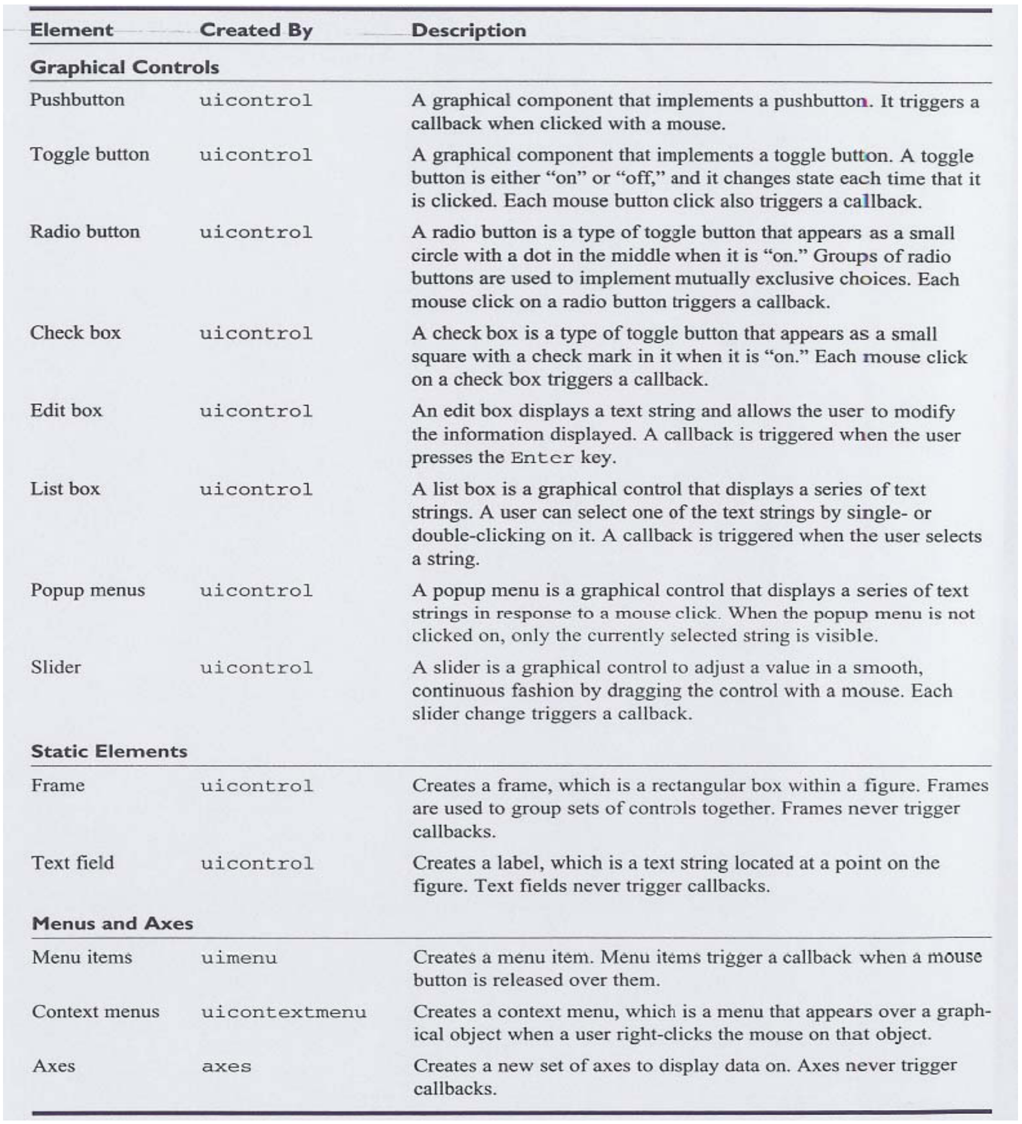


Figure 3.3: The GUIDE Tool Window

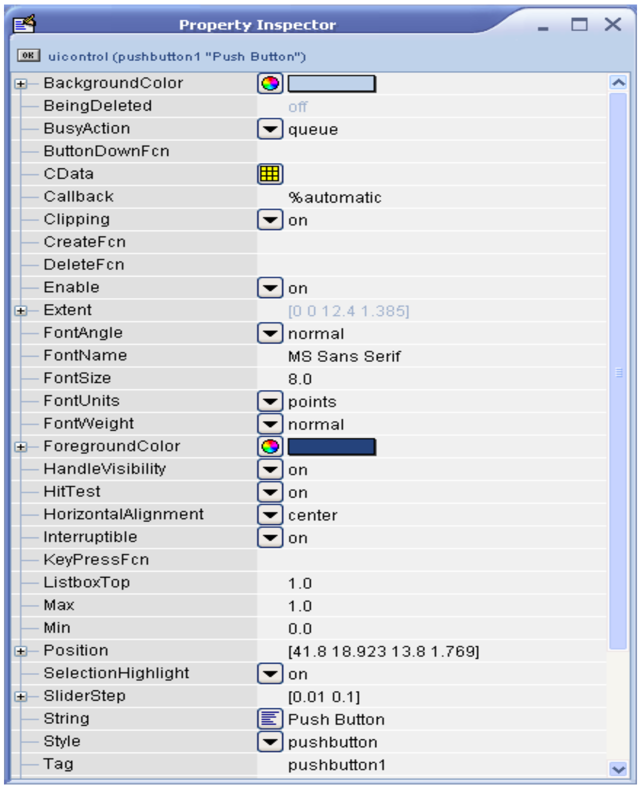
Firstly, decide what elements are required for the GUI and what the function of each element will be and then make a rough layout of the components by hand on a piece of paper. Then, after that use a MATLAB tool called guide (GUI Development Environment) to layout the Components on a figure. The size of the figure and the alignment and spacing of components on the figure can be adjusted using the tools built into guide. This figure below show some basic component of GUI that can be use to design the layout GUI.

Table 3.1: Some Basic GUI Component [1]



Element	Created By	Description
Graphical Controls		
Pushbutton	`uicontrol`	A graphical component that implements a pushbutton. It triggers a callback when clicked with a mouse.
Toggle button	`uicontrol`	A graphical component that implements a toggle button. A toggle button is either "on" or "off," and it changes state each time that it is clicked. Each mouse button click also triggers a callback.
Radio button	`uicontrol`	A radio button is a type of toggle button that appears as a small circle with a dot in the middle when it is "on." Groups of radio buttons are used to implement mutually exclusive choices. Each mouse click on a radio button triggers a callback.
Check box	`uicontrol`	A check box is a type of toggle button that appears as a small square with a check mark in it when it is "on." Each mouse click on a check box triggers a callback.
Edit box	`uicontrol`	An edit box displays a text string and allows the user to modify the information displayed. A callback is triggered when the user presses the `Enter` key.
List box	`uicontrol`	A list box is a graphical control that displays a series of text strings. A user can select one of the text strings by single- or double-clicking on it. A callback is triggered when the user selects a string.
Popup menus	`uicontrol`	A popup menu is a graphical control that displays a series of text strings in response to a mouse click. When the popup menu is not clicked on, only the currently selected string is visible.
Slider	`uicontrol`	A slider is a graphical control to adjust a value in a smooth, continuous fashion by dragging the control with a mouse. Each slider change triggers a callback.
Static Elements		
Frame	`uicontrol`	Creates a frame, which is a rectangular box within a figure. Frames are used to group sets of controls together. Frames never trigger callbacks.
Text field	`uicontrol`	Creates a label, which is a text string located at a point on the figure. Text fields never trigger callbacks.
Menus and Axes		
Menu items	`uimenu`	Creates a menu item. Menu items trigger a callback when a mouse button is released over them.
Context menus	`uicontextmenu`	Creates a context menu, which is a menu that appears over a graphical object when a user right-clicks the mouse on that object.
Axes	`axes`	Creates a new set of axes to display data on. Axes never trigger callbacks.

After design the layout, MATLAB tool called the Property Inspector (built into guide) is use to give each component a name (a "tag") and to set the characteristics of each component, such as its color, the text it displays, and so on. This figure below had shown the example of property inspector on push button.



Property Inspector

uicontrol (pushbutton1 "Push Button")

BackgroundColor	⬜
BeingDeleted	off
BusyAction	queue
ButtonDownFcn	
CData	
Callback	%automatic
Clipping	on
CreateFcn	
DeleteFcn	
Enable	on
Extent	[0 0 12.4 1.385]
FontAngle	normal
FontName	MS Sans Serif
FontSize	8.0
FontUnits	points
FontWeight	normal
ForegroundColor	🟦
HandleVisibility	on
HitTest	on
HorizontalAlignment	center
Interruptible	on
KeyPressFcn	
ListboxTop	1.0
Max	1.0
Min	0.0
Position	[41.8 18.923 13.8 1.769]
SelectionHighlight	on
SliderStep	[0.01 0.1]
String	Push Button
Style	pushbutton
Tag	pushbutton1

Figure 3.4: Property Inspector

After done with setting the property inspector in all components and then save the figure to a file. When the figure is saved, two files will be created on disk with the same name but different extents. The fig file contains the actual GUI that has been created, and the M-file contains the code to load the figure and skeleton call backs for each GUI element. Figure 3.5 and Figure 3.6 show the layout GUI after done with the designation with few basic components that had been used like push button and axes.

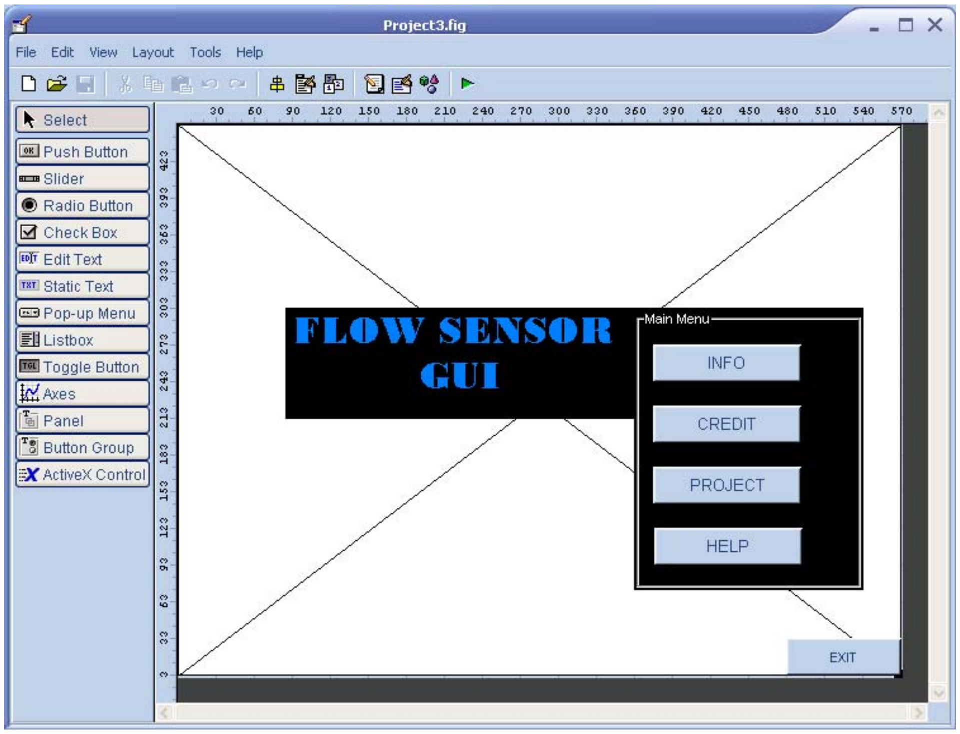


Figure 3.5: Example Layout GUI (Main Window)

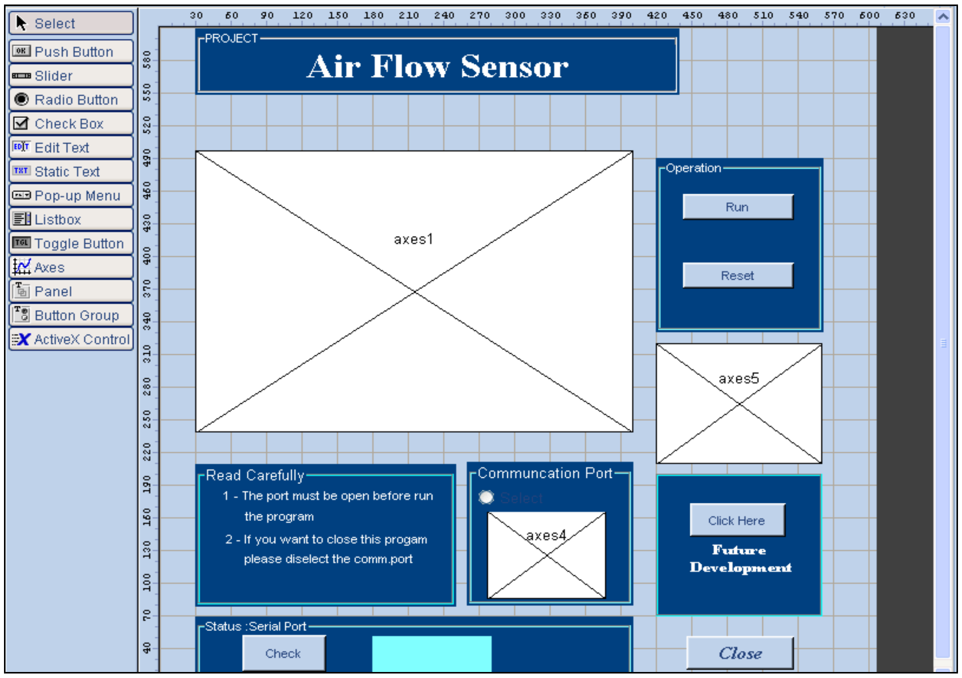
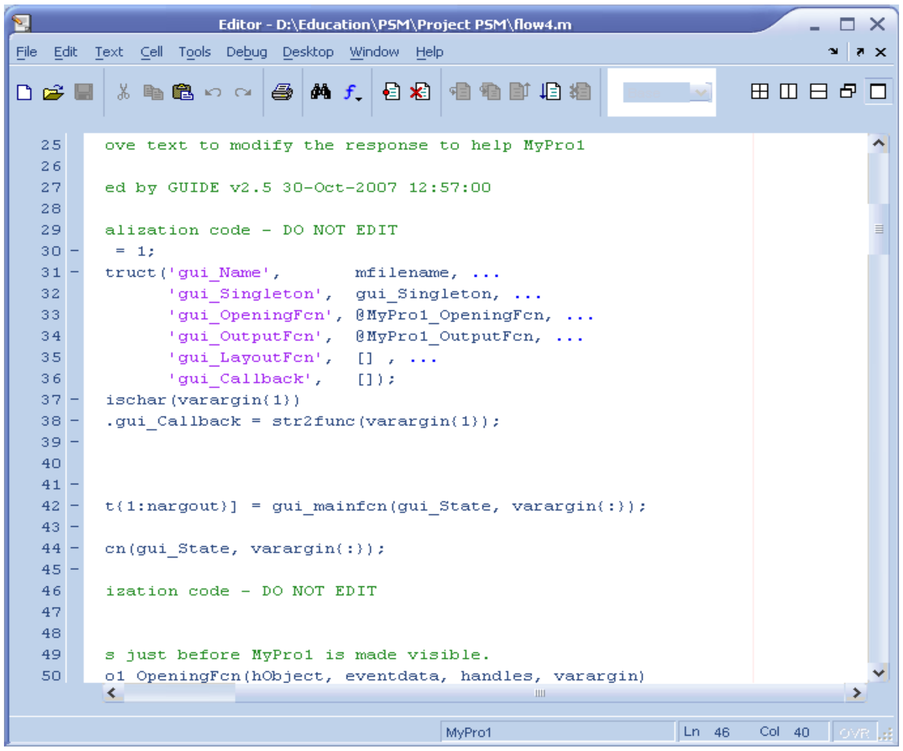


Figure 3.6: Example Layout GUI (Air Flow Sensor)

Finally, when save GUI layout, it will automatically generate an M-file and then write code to implement the behavior associated with each callback function in M-file as shown in Figure 3.7. This last step is the difficult part in GUIDE. This part is where the programmer can add code to the callbacks to perform the functions that what we want. If the coding is not correct then it cannot perform the function that we want.


Editor - D:\Education\PSM\Project PSM\flow4.m
File Edit Text Cell Tools Debug Desktop Window Help
MyPro1
Ln 46
Col 40
OVR

```
ove text to modify the response to help MyPro1

ed by GUIDE v2.5 30-Oct-2007 12:57:00

alization code - DO NOT EDIT
 = 1;
truct('gui_Name',       mfilename, ...
      'gui_Singleton',  gui_Singleton, ...
      'gui_OpeningFcn', @MyPro1_OpeningFcn, ...
      'gui_OutputFcn',  @MyPro1_OutputFcn, ...
      'gui_LayoutFcn',  [] , ...
      'gui_Callback',   []);
ischar(varargin{1})
.gui_Callback = str2func(varargin{1});



t{1:nargout}] = gui_mainfcn(gui_State, varargin{:});

cn(gui_State, varargin{:});

ization code - DO NOT EDIT


s just before MyPro1 is made visible.
o1_OpeningFcn(hObject, eventdata, handles, varargin)
```

Figure 3.7: Example of M-file

Unlike GUI objects, MATLAB does not automatically create callback strings and stub functions for menu items. The programmer must perform this function manually. Only the Label, Tag, Callback, Checked, and Separator properties of a menu item can be set from the Menu Editor. If want to set any of the other properties, use the Property Editor (propedit) on the figure, and select the appropriate menu item to edit.

3.2.4 Programming the GUI

After the layer out GUI is been created then need to program its behavior. The code that had been write, control how the GUI responds to events such as button clicks, slider movement, menu item selection, or the creation and deletion of components. This programming takes the form of a set of functions, called callbacks, for each component and for the GUI figure itself.

A callback is a function that writes and associates with a specific GUI component or with the GUI figure. It controls GUI or component behavior by performing some action in response to an event for its component. This kind of programming is often called event-driven programming. When an event occurs for a component, MATLAB invokes the component's callback that is triggered by that event. As an example, suppose a GUI has a button that triggers the plotting of some data. When the user clicks the button, MATLAB calls the callback that associated with clicking that button, and the callback, which have programmed, then gets the data and plots it.

The GUI figure and each type of component have specific kinds of callbacks with which it can be associated. The callbacks those are available for each component is defined as properties of that component. For example, a push button has five callback properties: Callback, CreateFcn, DeleteFcn, ButtonDownFcn, and KeyPressFcn as shown in Figure 3.8 after right click at the push button. The programmer can, but are not required to, create a callback function for each of these properties. The GUI itself, which is a figure, also has certain kinds of callbacks with which it can be associated. Each kind of callback has a triggering mechanism or event that causes it to be called. The following Table 3.2 is lists the callback properties that GUIDE makes available, their triggering events, and the components to which they apply.

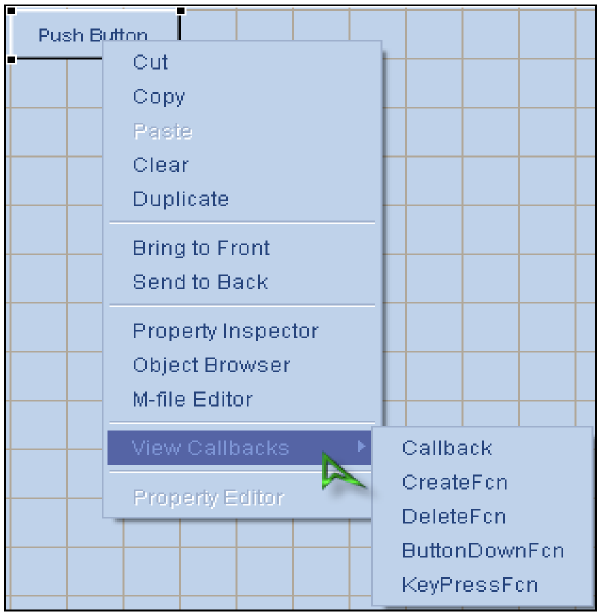


Figure 3.8: Push Button with Callback

Table 3.2: Lists the Callback Properties [3]

Callback Property	**Triggering Event**	**Components**
`ButtonDownFcn`	Executes when the user presses a mouse button while the pointer is on or within five pixels of a component or figure. If the component is a user interface control, its `Enable` property must be `on`.	Axes, figure, button group, panel, user interface controls
`Callback`	Component action. Executes, for example, when a user clicks a push button or selects a menu item.	Context menu, menu, user interface controls
`CloseRequestFcn`	Executes before the figure closes.	Figure
`CreateFcn`	Component creation. It can be use to initialize the component when it is created. It executes after the component or figure is created, but before it is displayed.	Axes, figure, button group, context menu, menu, panel, user interface controls

Callback Property	Triggering Event	Components
`DeleteFcn`	Component deletion. It can be used to perform cleanup operations just before the component or figure is destroyed.	Axes, figure, button group, context menu, menu, panel, user interface controls
`KeyPressFcn`	Executes when the user presses a keyboard key and the callback's component or figure has focus.	Figure, user interface controls
`ResizeFcn`	Executes when a user resizes a panel, button group, or figure whose figure `Resize` property is set to `On`.	Button group, figure, panel
`SelectionChangeFcn`	Executes when a user selects a different radio button or toggle button in a button group component.	Button group
`WindowButtonDownFcn`	Executes when you press a mouse button while the pointer is in the figure window.	Figure
`WindowButtonMotionFcn`	Executes when you move the pointer within the figure window.	Figure
`WindowButtonUpFcn`	Executes when you release a mouse button.	Figure

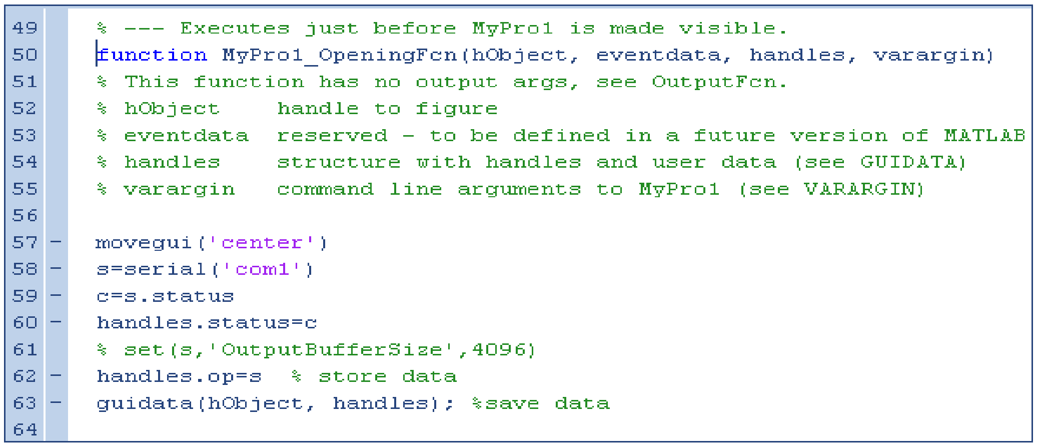
User interface controls include push buttons, sliders, radio buttons, check boxes, editable text boxes, static text boxes, list boxes, and toggle buttons. They are sometimes referred to as uicontrols.

The GUI M-file that GUIDE generates is a function file. The name of the main function is the same as the name of the M-file. For example, if the name of the M-file is mypro.m, then the name of the main function is mypro. Each callback in the file is a subfunction of the main function. When GUIDE generates an M-file, it automatically includes templates for the most commonly used callbacks for each component. The M-file also contains initialization code, as well as an opening function callback and an output function callback. The code must add to the component callbacks for the GUI to work as we want. The programmer may also want to add code to the opening function callback and the output function callback. The major sections of the GUI M-file are ordered as shown in the following table.

Table 3.3: Major Sections of The GUI M-file [3]

Section	**Description**
Comments	Displayed at the command line in response to the `help` command. Edit these as necessary for your GUI.
Initialization	GUIDE initialization tasks. *Do not edit this code.*
Opening function	Performs your initialization tasks before the user has access to the GUI.
Output function	Returns outputs to the MATLAB command line after the opening function returns control and before control returns to the command line.
Component and figure callbacks	Control the behavior of the GUI figure and of individual components. MATLAB calls a callback in response to a particular event for a component or for the figure itself.
Utility/helper functions	Perform miscellaneous functions not directly associated with an event for the figure or a component.

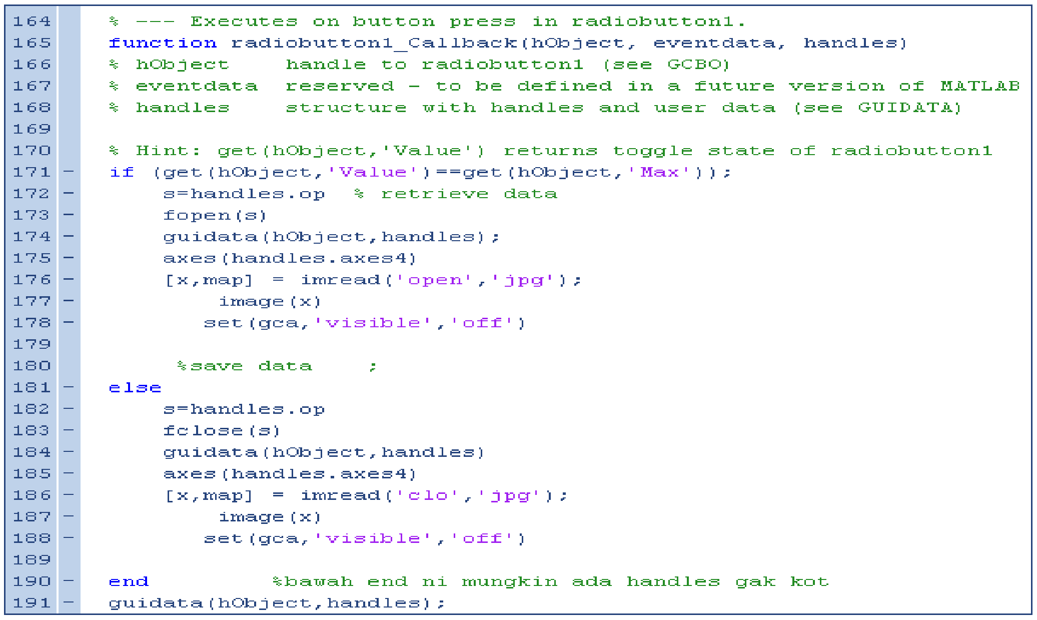
The opening is so important to the people who want to create the M-file in this GUIDE. For this project the important for opening function is to initialize the communication port in the computer. This is important before want to interface with the hardware using DB9 through communication port. First, we have to know the communication port name with located at my computer < properties < hardware < device manager < communication port. Then setting the communication port as shown in Figure 3.9 at the opening function.



```
% --- Executes just before MyPro1 is made visible.
function MyPro1_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to MyPro1 (see VARARGIN)

movegui('center')
s=serial('com1')
c=s.status
handles.status=c
% set(s,'OutputBufferSize',4096)
handles.op=s   % store data
guidata(hObject, handles); %save data

```

Figure 3.9: Initialize the Communication Port

Then after done with the setting, the port that had been initializing must be open to read the data from MAX232. If want to read the data the port must be open and when want to close the program the port must close to avoid the error and problem that can be occur. This Figure 3.10 shown the coding to open and close the port using radio button.

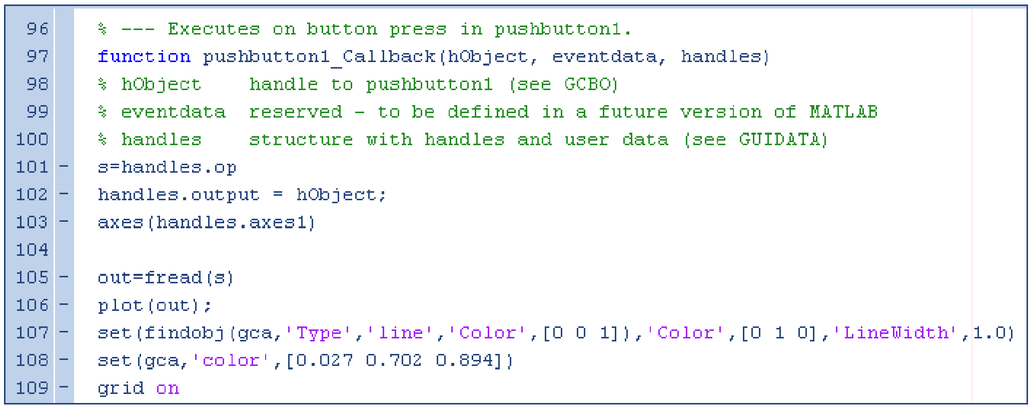


```
% --- Executes on button press in radiobutton1.
function radiobutton1_Callback(hObject, eventdata, handles)
% hObject    handle to radiobutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of radiobutton1
if (get(hObject,'Value')==get(hObject,'Max'));
    s=handles.op  % retrieve data
    fopen(s)
    guidata(hObject,handles);
    axes(handles.axes4)
    [x,map] = imread('open','jpg');
        image(x)
       set(gca,'visible','off')

     %save data    ;
else
    s=handles.op
    fclose(s)
    guidata(hObject,handles)
    axes(handles.axes4)
    [x,map] = imread('clo','jpg');
        image(x)
       set(gca,'visible','off')

end          %bawah end ni mungkin ada handles gak kot
guidata(hObject,handles);
```

Figure 3.10: Open and Close Communication Port

After the communication port has opened, to read the data that come from the sensor through PIC and MAX232 using DB9 then the coding as shown in Figure 3.11 had been used. The function fread(serial) is read binary data from the device.

Then set the function with out = fread(s) reads binary data from the device connected to obj, and returns the data to S. The maximum number of values to read is specified by size. After read the data then, use function plot (out) to display the signal data at axes in GUI.



```
% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
s=handles.op
handles.output = hObject;
axes(handles.axes1)

out=fread(s)
plot(out);
set(findobj(gca,'Type','line','Color',[0 0 1]),'Color',[0 1 0],'LineWidth',1.0)
set(gca,'color',[0.027 0.702 0.894])
grid on
```

Figure 3.11: Read Data from MAX232

3.2.5 Programming the PIC

There is much method use to program the PIC, such as P Basic and assembly language. But for this project, at the PIC part the ladder logic diagram (LDmicro) had been used. The ladder logic is much more like programming the PLC. Only the content and command will be different.

The program is presented in graphical format, not as a textual list of statements. Many people will initially find this easier to understand. At the most basic level, programs look like circuit diagrams, with relay contacts (inputs) and coils (outputs). This is intuitive to programmers with knowledge of electric circuit theory. The ladder logic compiler takes care of what gets calculated where. Don’t have to write code to determine when the outputs have to get recalculated based on a change in the inputs or a timer event, and don’t have to specify the order in which these calculations must take place. The PLC tools do that for you.

3.2.6 Getting Started with LDmicro

Using LDmicro, the user can draw a ladder diagram for their program. This program can simulate the logic in real time on your PC. Then when users are convinced that it is correct the user can assign pins on the microcontroller to the program inputs and outputs. Once users have assigned the pins, they can compile PIC or AVR code for their program. The compiler output is a .hex file that anyone can program into their microcontroller using any PIC/AVR programmer.

LDmicro is designed to be somewhat similar to most commercial PLC programming systems. There are some exceptions, and a lot of things aren't standard in industry anyways. The description of each instruction in the help must read carefully, even if it looks familiar. This part will assume basic knowledge of ladder logic and of the structure of PLC software (the execution cycle: read inputs, compute, and write outputs).

3.2.6.1 Command Line Options

Ldmicro.exe is typically run with no command line options. That means that the users can just make a shortcut to the program, or save it to their desktop and double-click the icon when they want to run it, and then they can do everything from within the GUI.

3.2.6.2 Basics

If LDmicro is run with no arguments then it starts with an empty program. If the LDmicro run with the name of a ladder program (xxx.ld) on the command line then it will try to load that program at startup. LDmicro uses its own internal format for the program it cannot import logic from any other tool.

If users did not load an existing program then users will be given a program with one empty rung. The users could add an instruction to it; for example they could add a set of contacts (Instruction -> Insert Contacts) named `Xnew'. `X' means that the contacts will be tied to an input pin on the microcontroller. This program could assign a pin to it later, after choosing a microcontroller and renaming the contacts. The first letter of a name indicates what kind of object it is. For example:

* Xname -- tied to an input pin on the microcontroller
* Yname -- tied to an output pin on the microcontroller
* Rname -- `internal relay': a bit in memory
* Tname -- a timer; turn-on delay, turn-off delay, or retentive
* Cname -- a counter, either count-up or count-down
* Aname -- an integer read from an A/D converter
* name -- a general-purpose (integer) variable

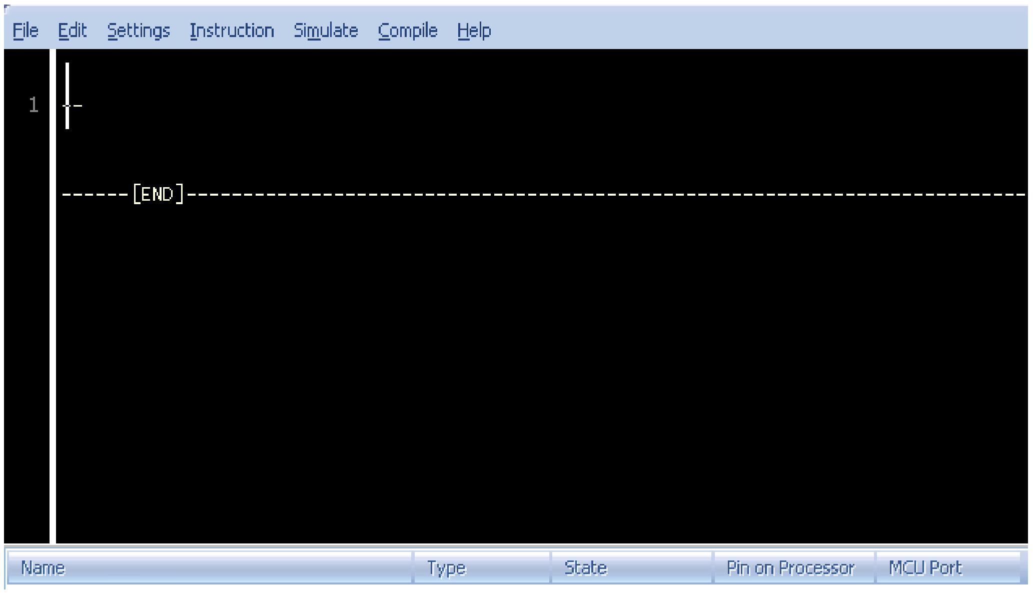


Figure 3.12: Start the program with one empty rung

Choose the rest of the name so that it describes what the object does, and so that it is unique within the program. The same name always refers to the same object within the program. Variable names can consist of letters, numbers, and underscores (_). A variable name must not start with a number. Variable names are case-sensitive.

At the bottom of the screen it will show a list of all the objects in the program. This list is automatically generated from the program; there is no need to keep it up to date by hand. Most objects do not need any configuration. `Xname', `Yname', and `Aname' objects must be assigned to a pin on the microcontroller, however. First choose which microcontroller you are using (Settings -> Microcontroller). Then assign your I/O pins by double-clicking them on the list.

The users can modify the program by inserting or deleting instructions. The cursor in the program display blinks to indicate the currently selected instruction and the current insertion point. If it is not blinking then press <Tab> or click on an instruction. Now the users can delete the current instruction, or users can insert a new instruction to the right or left (in series with) or above or below (in parallel with) the selected instruction. Some operations are not allowed. For example, no instructions are allowed to the right of a coil.

Once the program has been written, the program can test it in simulation, and then the program can be compile it to a HEX file for the target microcontroller.

3.2.6.3 Simulation

To enter simulation mode, choose Simulate -> Simulation Mode or press <Ctrl+M>. The program is shown differently in simulation mode. There is no longer a cursor. The instructions that are energized show up bright red; the instructions that are not appear greyed. Press the space bar to run the PLC one cycle. To cycle continuously in real time, choose Simulate -> Start Real-Time Simulation, or press <Ctrl+R>. The display of the program will be updated in real time as the program state changes.

The state of the inputs to the program can be set by double-clicking them in the list at the bottom of the screen, or by double-clicking an `Xname' contacts instruction in the program. When change the state of an input pin then that change will not be reflected in how the program is displayed until the PLC cycles, this will happen automatically if real time simulation is running, or when the space bar is press.

3.2.6.4 Compiling to Native code

Ultimately the point is to generate a .hex file that can be program into any microcontroller. Firstly, select the part number of the microcontroller, under the Settings -> Microcontroller menu. Then assign an I/O pin to each `Xname' or `Yname' object. This can be done by double-clicking the object name in the list at the bottom of the screen. A dialog will pop up where an unallocated pin from a list can be chosen.

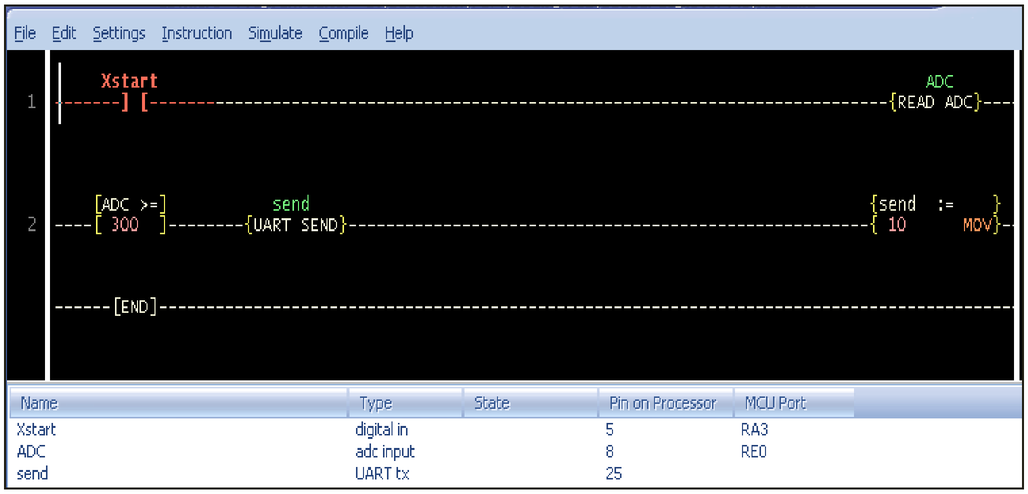
Then choose the cycle time that users will run with, and set the compiler what clock speed the micro will be running at. These are set under the Settings -> MCU Parameters... menu. In general there not need to change the cycle time; 10 ms is a good value for most applications. Type in the frequency of the crystal that this value will use with the microcontroller (or the ceramic resonator, etc.) and click okay.

Now with this it can generate code from the program that had been made. Choose Compile -> Compile, or Compile -> Compile As... if the users have previously compiled this program and want to change the specify a different output file name. If there are no errors then LDmicro will generate an Intel IHEX file ready for programming into any chip.

Use whatever programming software and hardware that want to load the hex file into the microcontroller. Remember to set the configuration bits (fuses)! For PIC16 processors, the configuration bits are included in the hex file, and most programming software will look there automatically. For AVR processors there must set the configuration bits by hand.

3.2.7 Programming PIC for This Project

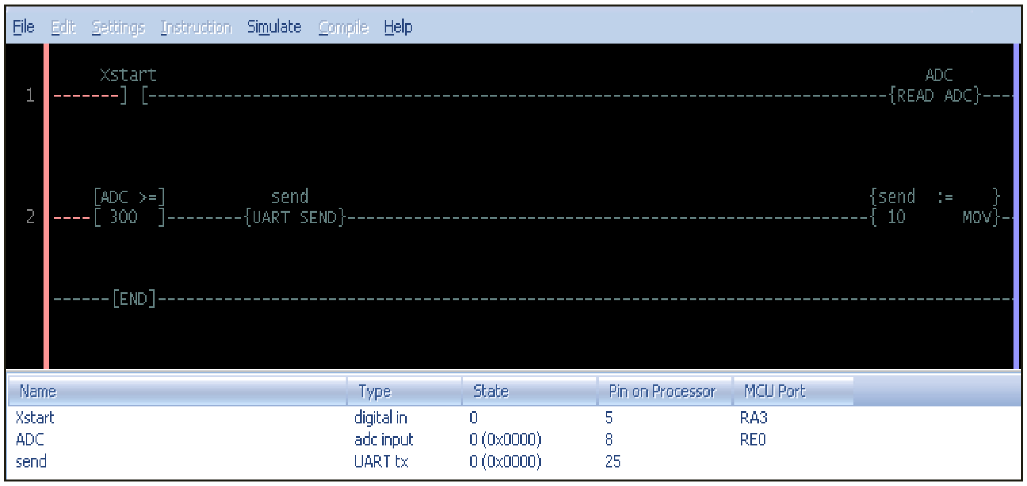
This below figure is show the programming of the PIC using this ladder logic diagram (LDmicro) for this project. The instruction that used in this program is contact, ADC, compare (greater than or equal), UART send and move.



```
     Xstart                                                                         ADC
1 ------] [---------------------------------------------------------------------{READ ADC}---

     [ADC >=]          send                                                 {send  :=     }
2 ----[ 300  ]--------{UART SEND}--------------------------------------------------{ 10      MOV}--

  ------[END]---------------------------------------------------------------------------------
```

Name	Type	State	Pin on Processor	MCU Port
Xstart	digital in		5	RA3
ADC	adc input		8	RE0
send	UART tx		25	

Figure 3.13: Programming the PIC

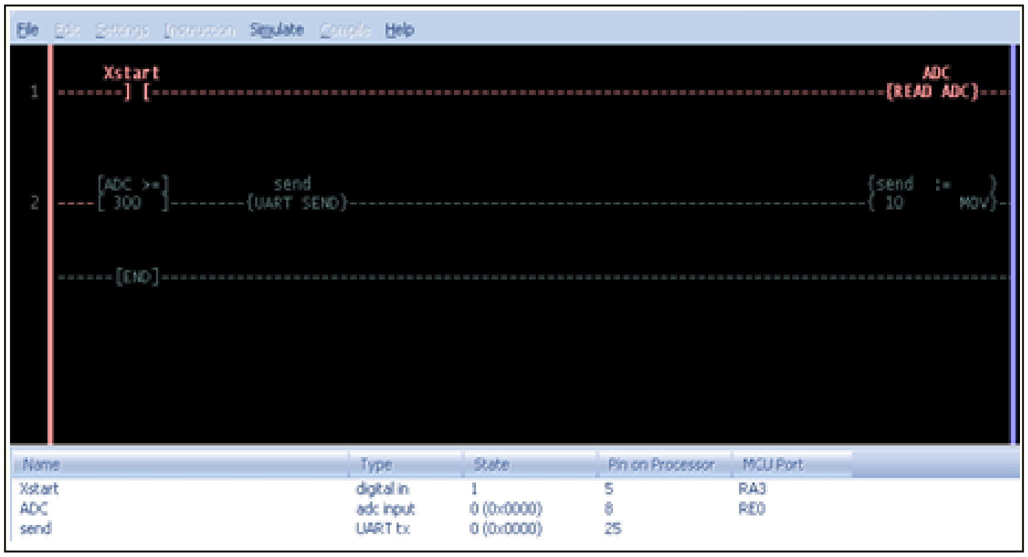


```
     Xstart                                                                         ADC
1 -------] [---------------------------------------------------------------------{READ ADC}---

     [ADC >=]          send                                                 {send  :=     }
2 ----[ 300  ]--------{UART SEND}--------------------------------------------------{ 10      MOV}--

  ------[END]---------------------------------------------------------------------------------
```

Name	Type	State	Pin on Processor	MCU Port
Xstart	digital in	0	5	RA3
ADC	adc input	0 (0x0000)	8	RE0
send	UART tx	0 (0x0000)	25	

Figure 3.14: In Simulation Mode

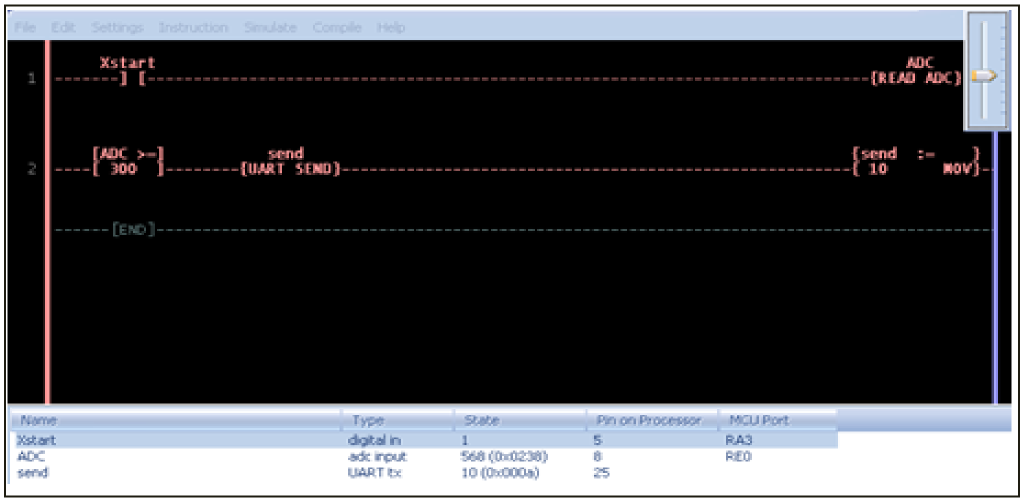
The above Figure 3.14 show the programming that had been made is in the simulation mode. There is no longer a cursor. The instructions that are energized show up bright red, the instructions that are not appear greyed.



```
        Xstart                                                              ADC
1  -------] [------------------------------------------------------------{READ ADC}---

       [ADC >=]          send                                          {send  :=    }
2  ----[ 300  ]--------{UART SEND}----------------------------------------{ 10    MOV}-

   ------[END]-----------------------------------------------------------------------
```

Name	Type	State	Pin on Processor	MCU Port
Xstart	digital in	1	5	RA3
ADC	adc input	0 (0x0000)	8	RE0
send	UART tx	0 (0x0000)	25	

Figure 3.15(a): In the Real-Time Simulation



```
        Xstart                                                              ADC
1  -------] [------------------------------------------------------------{READ ADC}

       [ADC >=]          send                                          {send  :=    }
2  ----[ 300  ]--------{UART SEND}----------------------------------------{ 10    MOV}-

   ------[END]-----------------------------------------------------------------------
```

Name	Type	State	Pin on Processor	MCU Port
Xstart	digital in	1	5	RA3
ADC	adc input	568 (0x0238)	8	RE0
send	UART tx	10 (0x000a)	25	

Figure 3.15(b): In the Real-Time Simulation

This figure as shown as above is when the real-time simulation is started. When the real-time simulation is started, firstly double click at the Xstart and the current (red line) will go through the ADC. And then it will go to the compare. The instruction compare is compare when the value of the ADC is greater than or equal 300 that had been setting. When the value is true then the instruction compare will allow the current go through the next instruction. The next instruction is UART Send. This instruction will just to transmit the data that had been converted to connect with the MAX232. This instruction must be with move instruction. When no error or warning occurs at this simulation then the programming can be compiling in .hex file that can be program into any microcontroller that supported with this ladder logic diagram (LDmicro).

3.2.7.1 Instructions Reference

> CONTACT, NORMALLY OPEN

```
   Xname          Rname          Yname
----] [----    ----] [----    ----] [----
```

If the signal going into the instruction is false, then the output signal is false. If the signal going into the instruction is true, then the output signal is true if and only if the given input pin, output pin, or internal relay is true, else it is false. This instruction can examine the state of an input pin, an output pin, or an internal relay.

> A/D CONVERTER READ

```
     Aname
--{READ ADC}--
```

LDmicro can generate code to use the A/D converters built in to certain microcontrollers. If the input condition to this instruction is true, then a single sample from the A/D converter is acquired and stored in the variable `Aname'. This variable can subsequently be manipulated with general variable operations (less than, greater than, arithmetic, and so on). Assign a pin to the `Axxx' variable in the same way that can be assign a pin to a digital input or output, by double-clicking it in the list at the bottom of the screen. If the input condition to this rung is false then the variable `Aname' is left unchanged.

For all currently-supported devices, 0 volts input corresponds to an ADC reading of 0, and an input equal to Vdd (the supply voltage) corresponds to an ADC reading of 1023. If using an AVR, then connect AREF to Vdd. There can use arithmetic operations to scale the reading to more convenient units afterwards, but remember that this instruction are using integer math. In general not all pins will be available for use with the A/D converter. The software will not allow the users to assign non-A/D pins to an analog input. This instruction must be the rightmost instruction in its rung.

> COMPARE

```
  [var ==]        [var >]        [1 >=]
-[ var2 ]-      -[ 1   ]-      -[ Ton]-

 [var /=]       [-4 <   ]       [1 <=]
-[ var2 ]-     -[ vartwo]-     -[ Cup]-
```

If the input to this instruction is false then the output is false. If the input is true then the output is true if and only if the given condition is true. This instruction can be used to compare (equals, is greater than, is greater than or equal to, does not equal, is less than, is less than or equal to) a variable to a variable, or to compare a variable to a 16-bit signed constant.

> MOVE

```
  {destvar :=  }      {Tret :=     }
-{ 123     MOV}-    -{ srcvar  MOV}-
```

When the input to this instruction is true, it sets the given destination variable equal to the given source variable or constant. When the input to this instruction is false nothing happens. This instruction can assign to any variable with the move instruction; this includes timer and counter state variables, which can be distinguished by the leading `T' or `C'. For example, an instruction moving 0 into `Tretentive' is equivalent to a reset (RES) instruction for that timer. This instruction must be the rightmost instruction in its rung.

> UART (SERIAL) SEND

```
              var
          --{UART SEND}--
```

LDmicro can generate code to use the UARTs built in to certain microcontrollers. On AVRS with multiple UARTs only UART1 (not UART0) is supported. Configure the baud rate using Settings -> MCU Parameters. Certain baud rates may not be achievable with certain crystal frequencies; LDmicro will warn if this is the case.

If the input condition to this instruction is false, then nothing happens. If the input condition is true then this instruction writes a single character to the UART. The ASCII value of the character to send must previously have been stored in `var'. The output condition of the rung is true if the UART is busy (currently transmitting a character), and false otherwise.

These characters take some time to transmit. The output condition of this instruction must be checked to ensure that the first character has been transmitted before trying to send a second character, or use a timer to insert a delay between characters. There must only bring the input condition true (try to send a character) when the output condition is false (UART is not busy). The formatted string instruction (next) must be investigate before using this instruction. The formatted string instruction is much easier to use, and it is almost certainly capable of doing what users want.

3.2.8 Hardware Installation

For this part in hardware installation design, firstly is design the power supply module using voltage regulator 7805. This is to supply 5V fixed to PIC and MAX232 IC. This voltage regulator is so important to prevent the higher input supply to the device which can bring damage to PIC and MAX232 IC. The schematic diagram for the voltage regulator 7805 as shown in Figure 3.16. Input to the power supply must greater than 7V to achieve the 5V output supply to PIC and max232.

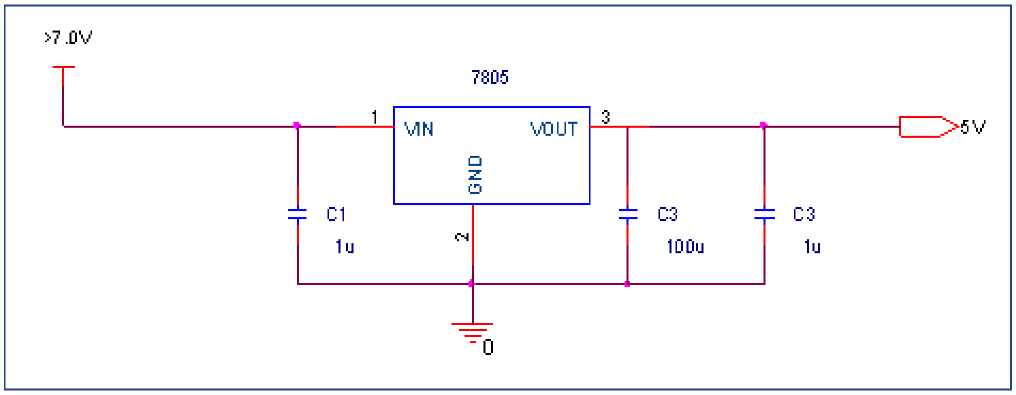


Figure 3.16: Voltage Regulator 7805

The second part is the flow sensor. For this project the air flow sensor had been designed and created as shown in Figure 3.17. The sensor used photo infrared sensor. The brown wire has connected to power supply that setting 12.5V and the blue wire is connecting to the ground. The air flow sensor works when there a movement (fan) cut light from the device transmit and receive at the sensor. When sensor detect the movement then its will give 10V to black wire. The black wire connects to two resistances in series to decrease the voltage to 5V. Then from the two resistance in series (voltage divider), its will connect to PIC to convert into digital data.

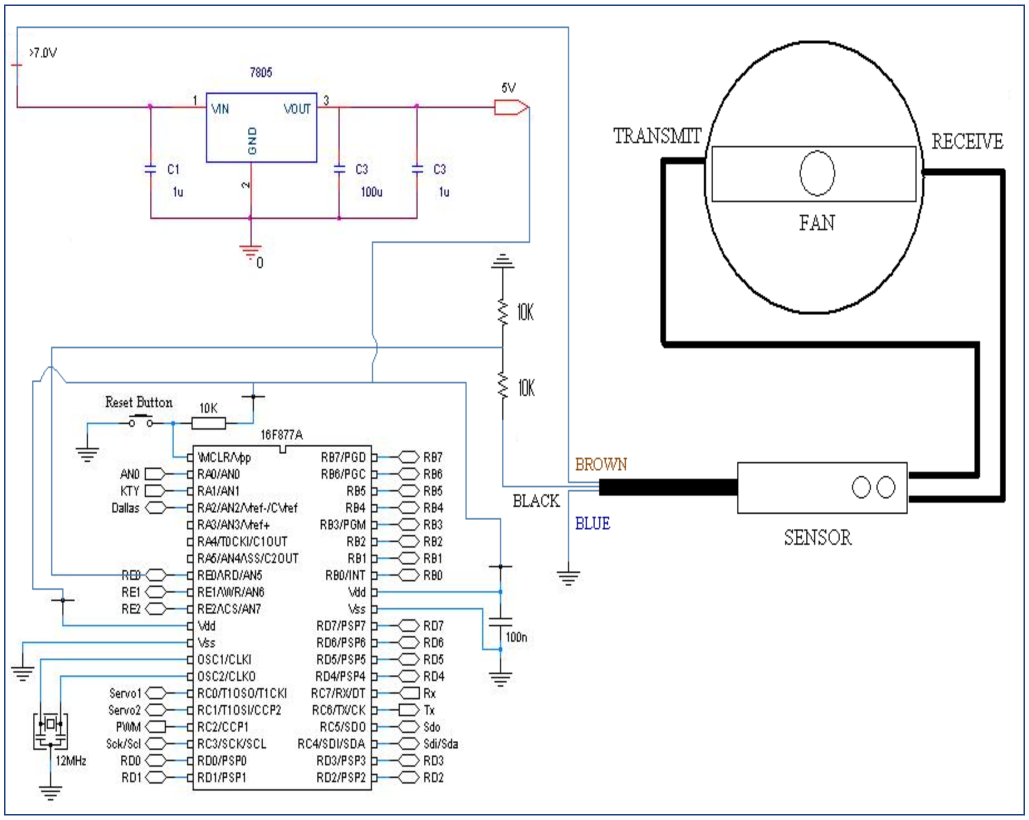


Figure 3.17: Air Flow Sensor with PIC and Voltage Regulator

Then the third part is building the connection between PIC16F877A and MAX232. After get the data from the sensor, the program in PIC will convert the data into digital data then transmit the data to MAX232. Then the MAX232 will send the data to computer through communication port with using DB9. The Figure 3.18 is show the schematics diagram for PIC 16F877A and MAX232.

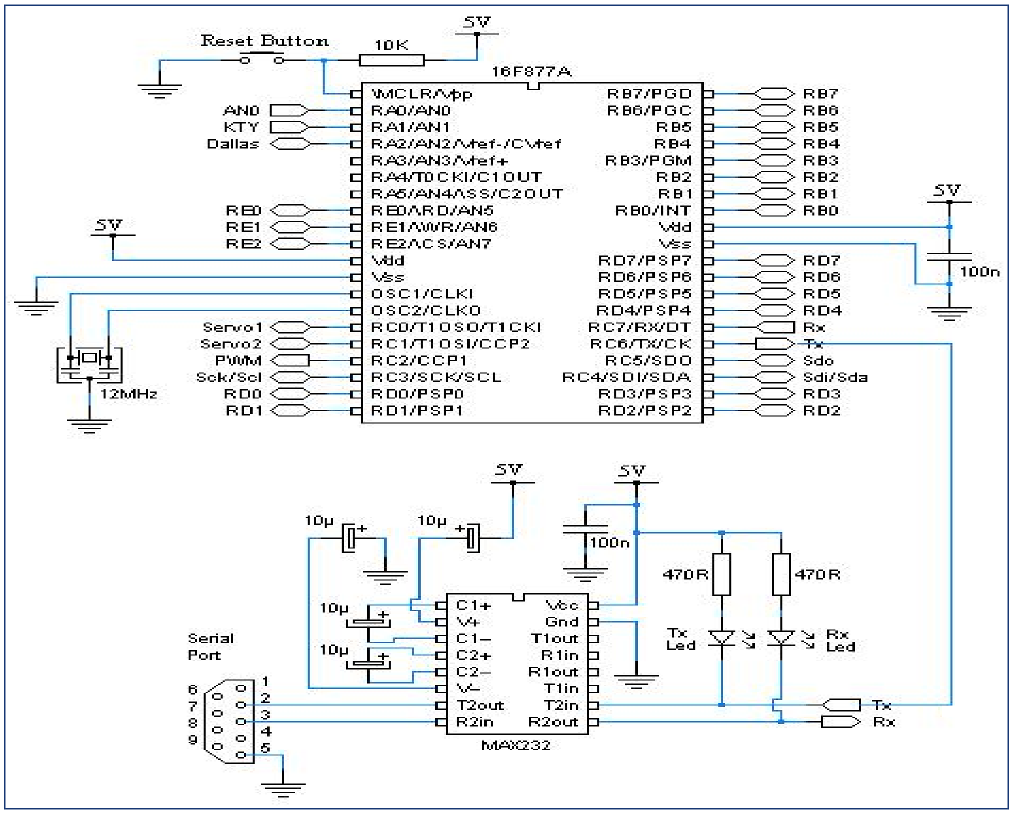


Figure 3.18: PIC16F77A and MAX232

The last part is to design the connection from communication port (DB9 female connection) from computer to the device. The pin assignment is shown in Table 3.4 below and the figure of RS 232 communication port shown on Figure 3.19. Only three pins are required for serial port communications: one for receiving data, one for transmitting data, and another one for the signal ground. The connection only on pin 2, 3 and pin 5.

Table 3.4: Serial Port Pin and Signal Assignments

Pin	Label	Signal Name	Signal Type
1	CD	Carrier	Detect Control
2	RD	Received Data	Data
3	TD	Transmitted Data	Data
4	DTR	Data Terminal Ready	Control
5	GND	Signal Ground	Ground
6	DSR	Data Set Ready	Control
7	RTS	Request to Send	Control
8	CTS	Clear to Send	Control
9	RI	Ring Indicator	Control

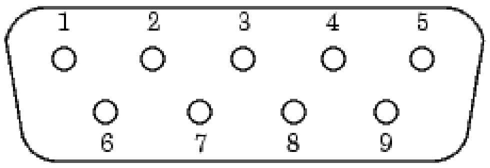


Figure 3.19: Pins and Signals Associated With the 9-pin Connector

CHAPTER 4

RESULT DISCUSSION

4.1 Introduction

This chapter consists of the discussions on the results from the simulations using the MATLAB GUI that the results come from flow sensor. This chapter shows the result that can display to MATLAB GUI with the information come from the sensor. All the figures of the layout GUI is 100 percent working. The GUI has performed the task that had been given. This part will explain the main menu of GUI, interface with hardware, advance development of GUI, user guide (help)

4.2 Main Menu

This part is show the main menu for the flow sensor. This main menu consists 5 push buttons that had been named with info, credit, project, help and lastly the button exit as shown is Figure 4.1. The part for info button will explain about the detail for this project. For part credit button it will show the detail about GUI programmer and also the supervisor that support and help to make sure this project is successful. The project button will show the subtopic for this project. The part for help button is to help the users more understanding this project and also try to guide the users to use this software. The last part is button exit confirmation, the user will ask whether want to close or not. When click yes it will close the main menu and all figure.

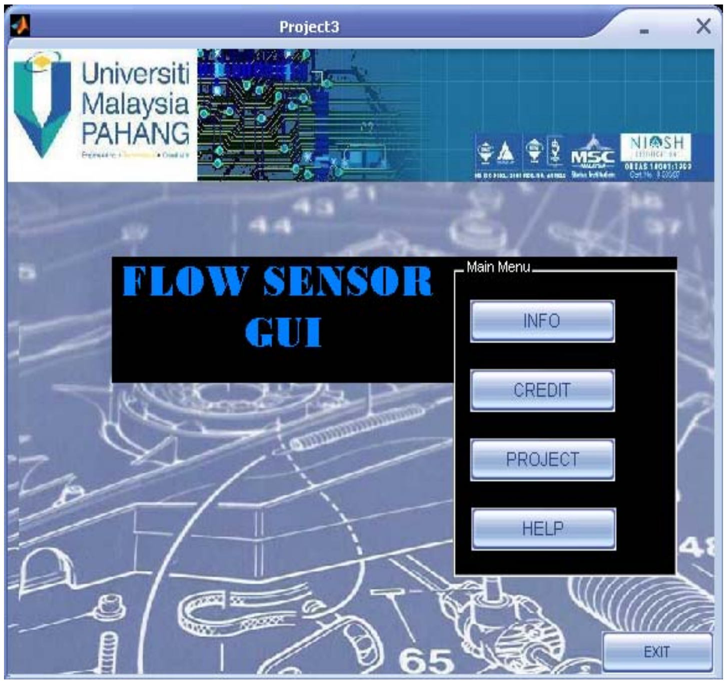
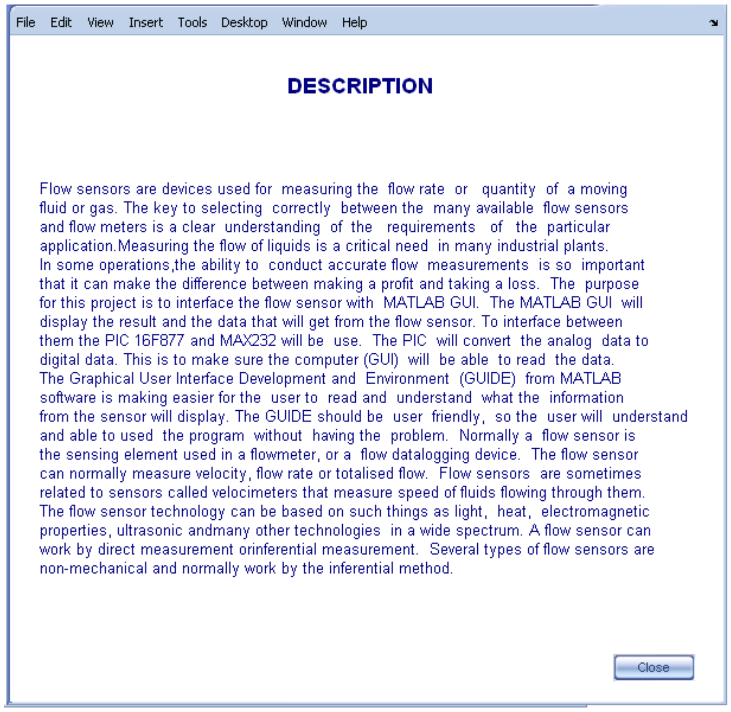


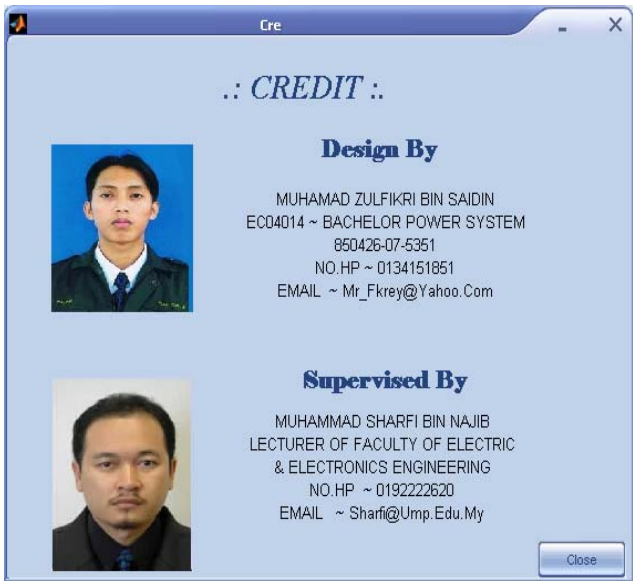
Figure 4.1: Main Menu of GUI



DESCRIPTION

Flow sensors are devices used for measuring the flow rate or quantity of a moving fluid or gas. The key to selecting correctly between the many available flow sensors and flow meters is a clear understanding of the requirements of the particular application.Measuring the flow of liquids is a critical need in many industrial plants. In some operations,the ability to conduct accurate flow measurements is so important that it can make the difference between making a profit and taking a loss. The purpose for this project is to interface the flow sensor with MATLAB GUI. The MATLAB GUI will display the result and the data that will get from the flow sensor. To interface between them the PIC 16F877 and MAX232 will be use. The PIC will convert the analog data to digital data. This is to make sure the computer (GUI) will be able to read the data. The Graphical User Interface Development and Environment (GUIDE) from MATLAB software is making easier for the user to read and understand what the information from the sensor will display. The GUIDE should be user friendly, so the user will understand and able to used the program without having the problem. Normally a flow sensor is the sensing element used in a flowmeter, or a flow datalogging device. The flow sensor can normally measure velocity, flow rate or totalised flow. Flow sensors are sometimes related to sensors called velocimeters that measure speed of fluids flowing through them. The flow sensor technology can be based on such things as light, heat, electromagnetic properties, ultrasonic andmany other technologies in a wide spectrum. A flow sensor can work by direct measurement orinferential measurement. Several types of flow sensors are non-mechanical and normally work by the inferential method.

Figure 4.2: Info


Cre

.: CREDIT :.

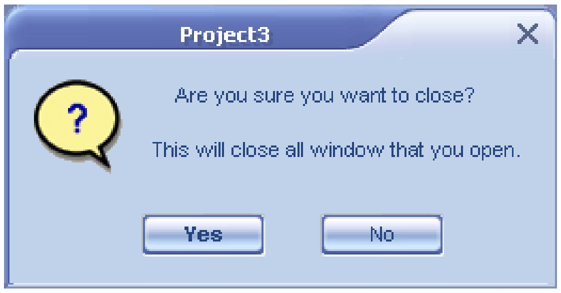
Design By

MUHAMAD ZULFIKRI BIN SAIDIN
EC04014 ~ BACHELOR POWER SYSTEM
850426-07-5351
NO.HP ~ 0134151851
EMAIL ~ Mr_Fkrey@Yahoo.Com

Supervised By

MUHAMMAD SHARFI BIN NAJIB
LECTURER OF FACULTY OF ELECTRIC
& ELECTRONICS ENGINEERING
NO.HP ~ 0192222620
EMAIL ~ Sharfi@Ump.Edu.My

Figure 4.3: Credit


Project3

Are you sure you want to close?

This will close all window that you open.

Figure 4.4: Button exit confirmation

4.3 Interfaces with the Air Flow Sensor

After select the project button, it will show the new window as show in Figure 4.5. Then to go to air flow sensor figure, click at button air flow sensor then the figure as shown in Figure 4.6 will display. To run the program, first is needed to open the port with select the communication port (Radio Button). Then port will open as show in Figure 4.7. The status for this port can be check with click at button check. Then if the port had been open it will show ‘opened’ as shown also in Figure 4.7. Then for the close port as shown in Figure 4.8.

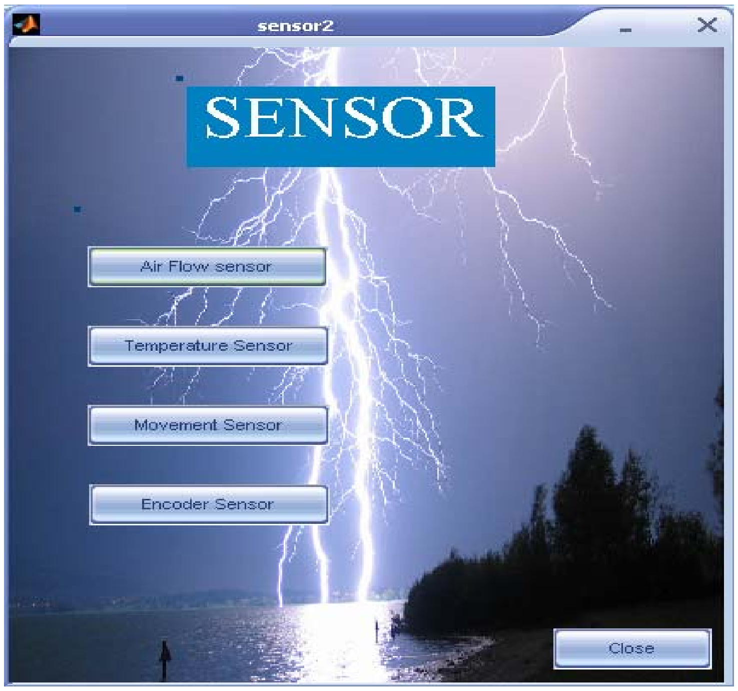
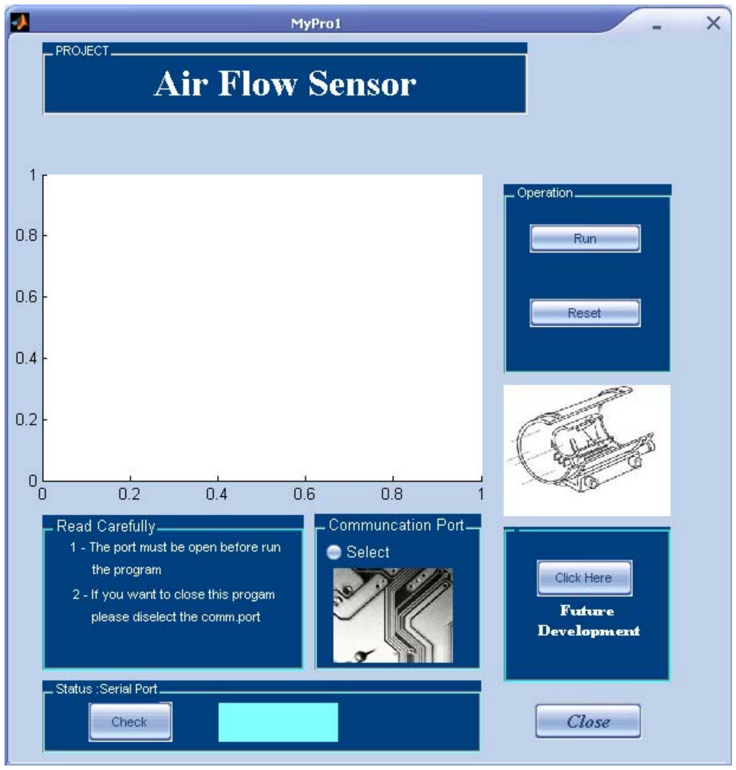


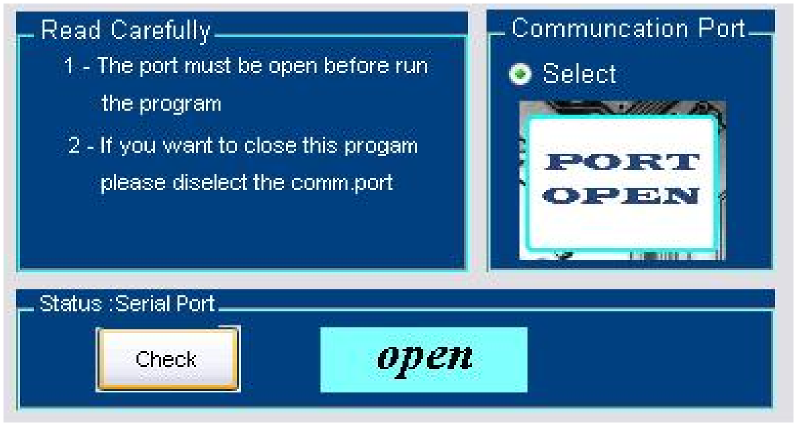
Figure 4.5: Sensor



1 - The port must be open before run the program

2 - If you want to close this progam please diselect the comm.port

Figure 4.6: Air Flow sensor



Read Carefully

1 - The port must be open before run the program

2 - If you want to close this progam please diselect the comm.port

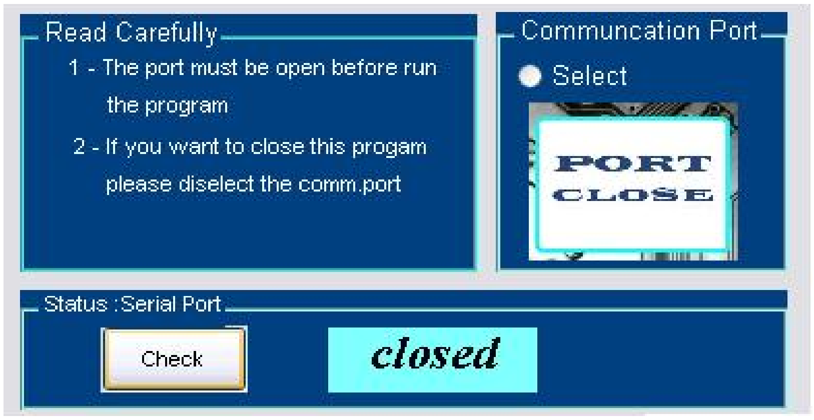
Communcation Port

Select [✓]

Status :Serial Port

open

Figure 4.7: Port is Open



Read Carefully

1 - The port must be open before run the program

2 - If you want to close this progam please diselect the comm.port

Communcation Port

Select

Status :Serial Port

closed

Figure 4.8: Port is closed

After the port has opened, then the program will be run with click at run button. When the hardware is ready the result will display at the GUI. At the air flow sensor when the air through this with high velocity then the fan will rotate fast and it will give more input as shown in Figure 4.9. But if the air coming through this sensor with low velocity then the fan will rotate slow then less input will display as shown in Figure 4.10.

But the problem with this result is not consistent. The time and voltage show in GUI figure also not consistent. It only show the time that had been detected. The problem came from this sensor because the sensor can’t detect a very fast movement of fan. It will effective if fan rotate in slow movement.

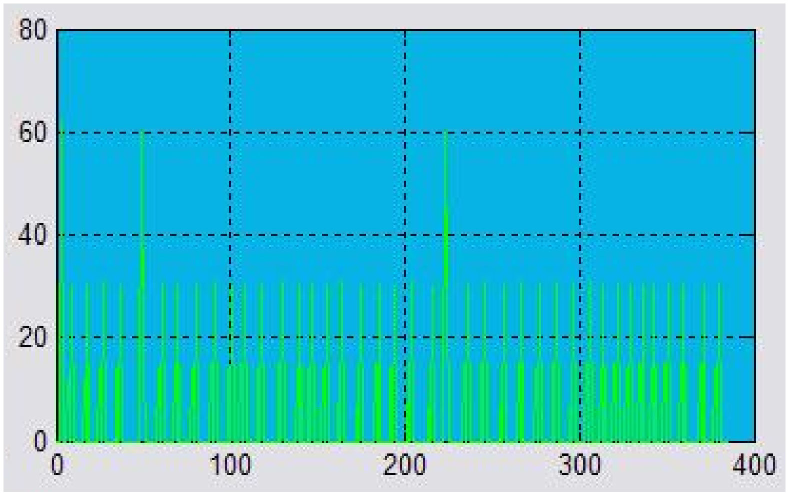


Figure 4.9: Result with High Velocity

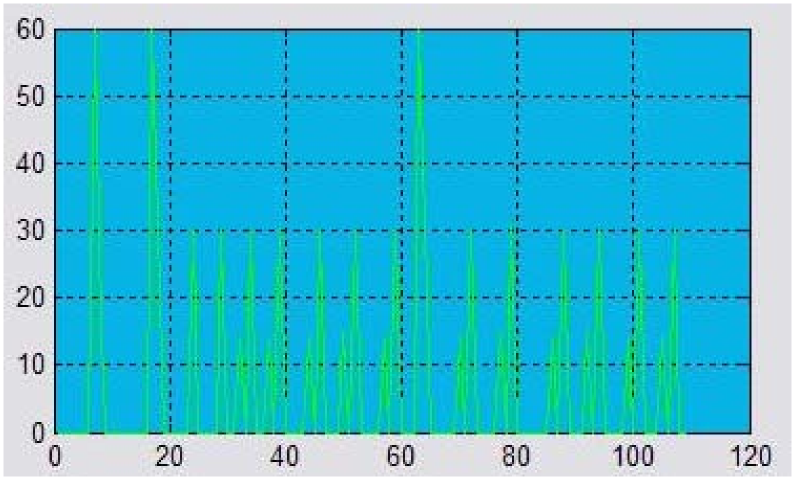


Figure 4.10: Result with Low Velocity

From above figure, we can see the difference for both figures. The both signal waveform has a little bit difference which Figure 4.9 give more input than Figure 4.10. That because Figure 4.9 is shows the result when the air coming through the sensor with high velocity but for Figure 4.10 is shows the result when the air coming through the sensor with low velocity. For the normal condition, when click at button run it will take 10s to display at GUI. The time is measure in milliseconds. The both figure also shows the difference at the time and amplitude of signal. The difference in time for both signals it is because the GUI will display the signal when there are inputs at that time. That why when no input is detected by the sensor the GUI will not display the signal. For the amplitude it shows the amplitude in voltage after been converted in binary. Than the value for amplitudes is display in binary number.

The comparison with this result that display at GUI and oscilloscope had been done. The oscilloscope is connect to the pin at MAX232 that at pin T2out. The comparison is show in Figure 4.11 and 4.12. Figure 4.11 show the result with high velocity of air that because at that time the sensor is detect more input when the fan moving fast and the result will display as shown in figure. Figure 4.12 shows the result with low velocity of air when the sensor is detecting the fan moving slowly via transmits and receive of sensor. When that happens it will provide less input and show in oscilloscope as shown in Figure 4.12. The result in oscilloscope can display signal in positive and negative side. But the result show in GUI is only display on positive side. The oscilloscope can display the signal continuously but for the MATLAB GUI it will display signal when clicking at button run and the signal that display on GUI is only at that time.

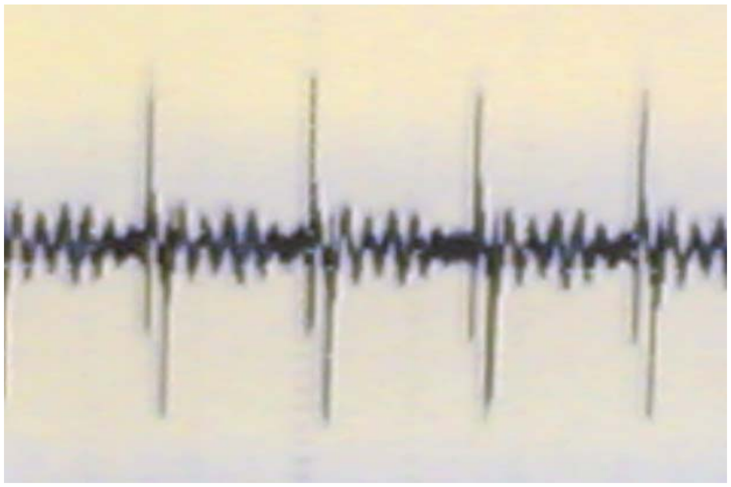


Figure 4.11: Result from oscilloscope (More Input)

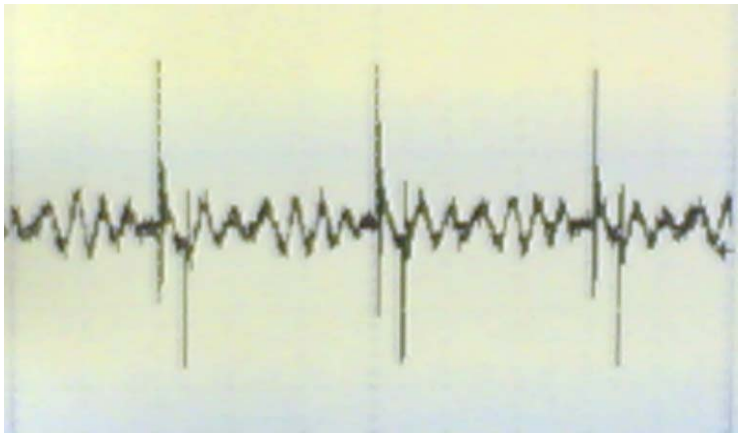


Figure 4.12: Result from oscilloscope (Less Input)

From this window of air flow sensor when click at button ‘click here’ it will display the future development of this sensor. For the future development the flow sensor not only can detect movement of air but also this flow sensor can detect solid, liquid and also steam/gas flow sensor. Figure 4.13 show the future development of flow sensor.

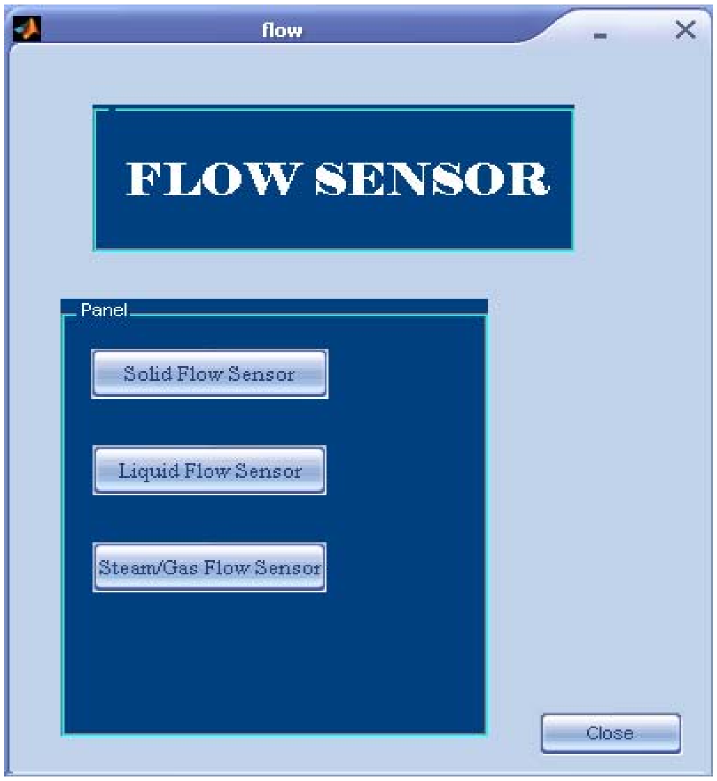
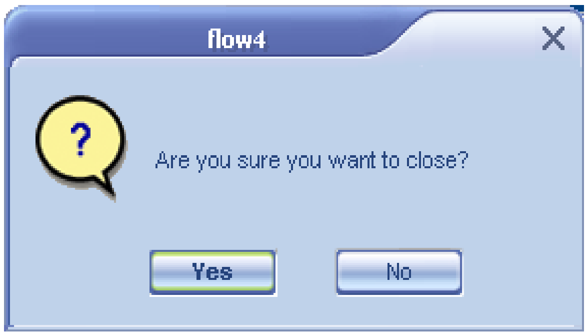


Figure 4.13: Future Development of Flow Sensor

After the project had been done then this program will close with click at close button. The figure close confirmation will appear an ask user whether to close or not as shown in Figure 4.14. This close confirmation is to reminder the user to close the port when want to close this GUI.



Are you sure you want to close?

Figure 4.14: Close Confirmation

4.4 Advanced Development of GUI

For this part will explain about advanced development that had been designed for this project. This part is just to show not only flow sensor can be used to interface with this software but temperature sensor, movement sensor and encoder sensor also can interfaces using this software program. Figure 4.15 show the sensor in advanced development.

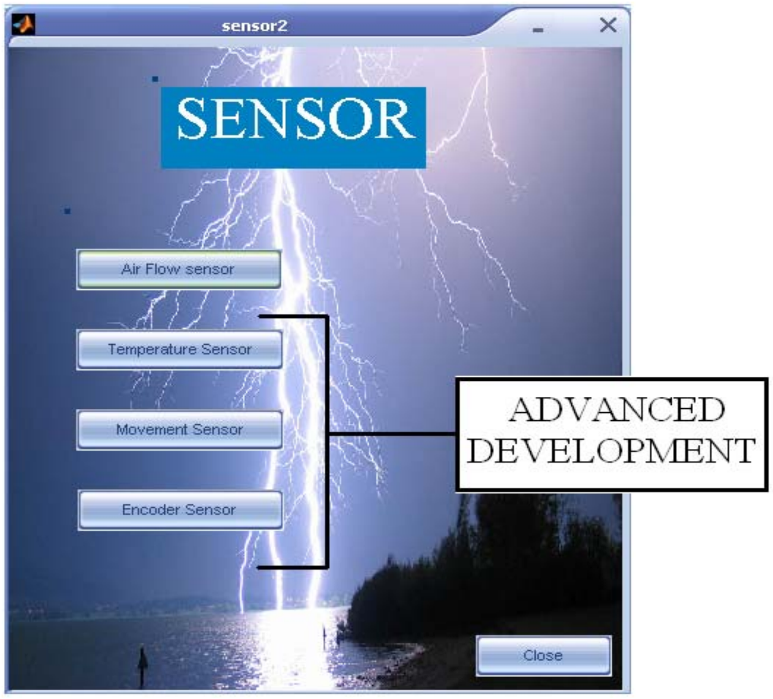
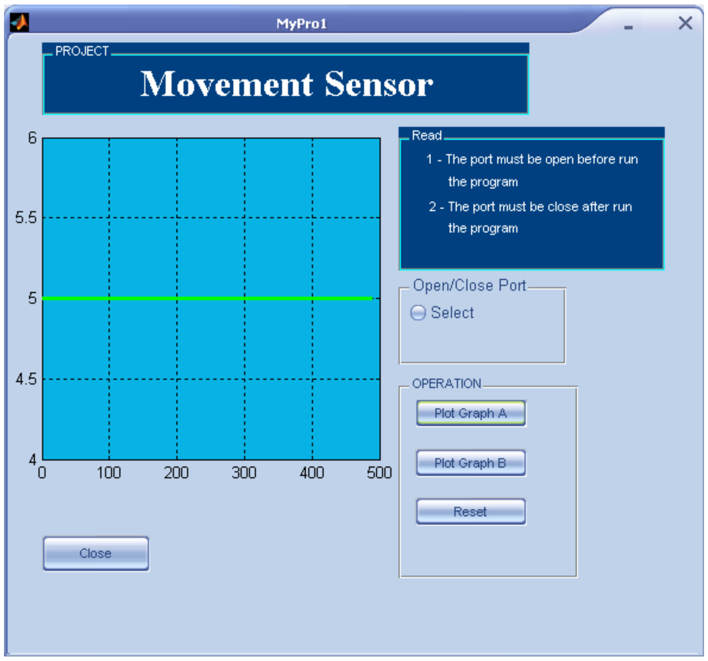


Figure 4.15: Temperature, Movement and Encoder Sensor

For this part, the movement sensor and encoder sensor had been designed. The design that had been creates only on GUI. The movement sensor is work when there is any movement through this sensor. When the movement if detect then it will give 5V but 0V when no movement if detected. For the encoder sensor, the waveform that will display at GUI is like pulse. This sensor is like wheel it will rotate continuously and the graph will display as shown in Figure 4.17.



1 - The port must be open before run the program

2 - The port must be close after run the program

Figure 4.16: Movement Sensor

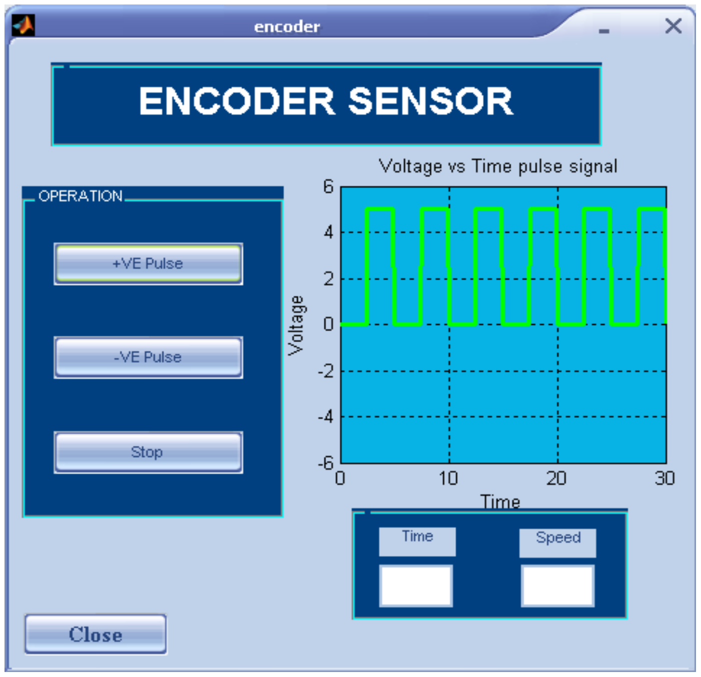


Figure 4.17: Encoder Sensor

4.5 Users Help

For this part, it will help and guide the user to understand this project. When click at help button, Figure 4.18 will display. This help consists of two parts. The first part is tried to explain to the user about the detail of this project. The second is to guide the users on how to select the communication port as shown in Figure 4.20.

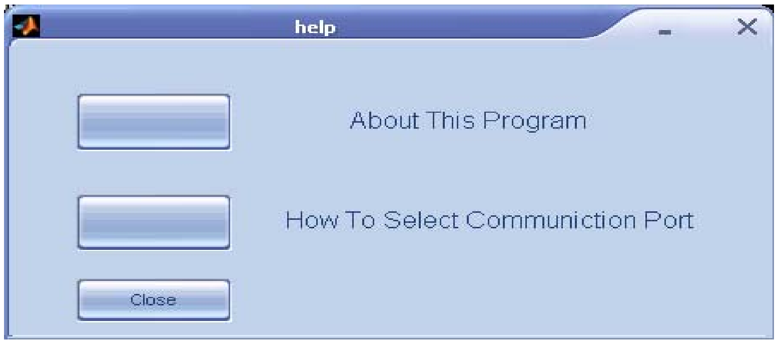
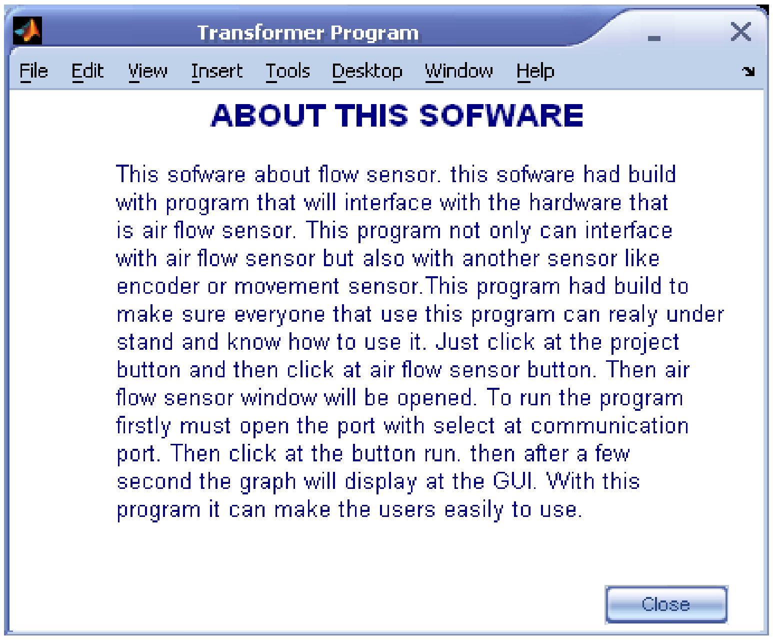


Figure 4.18: Help



ABOUT THIS SOFWARE

This sofware about flow sensor. this sofware had build with program that will interface with the hardware that is air flow sensor. This program not only can interface with air flow sensor but also with another sensor like encoder or movement sensor.This program had build to make sure everyone that use this program can realy under stand and know how to use it. Just click at the project button and then click at air flow sensor button. Then air flow sensor window will be opened. To run the program firstly must open the port with select at communication port. Then click at the button run. then after a few second the graph will display at the GUI. With this program it can make the users easily to use.

Figure 4.19: About This Software

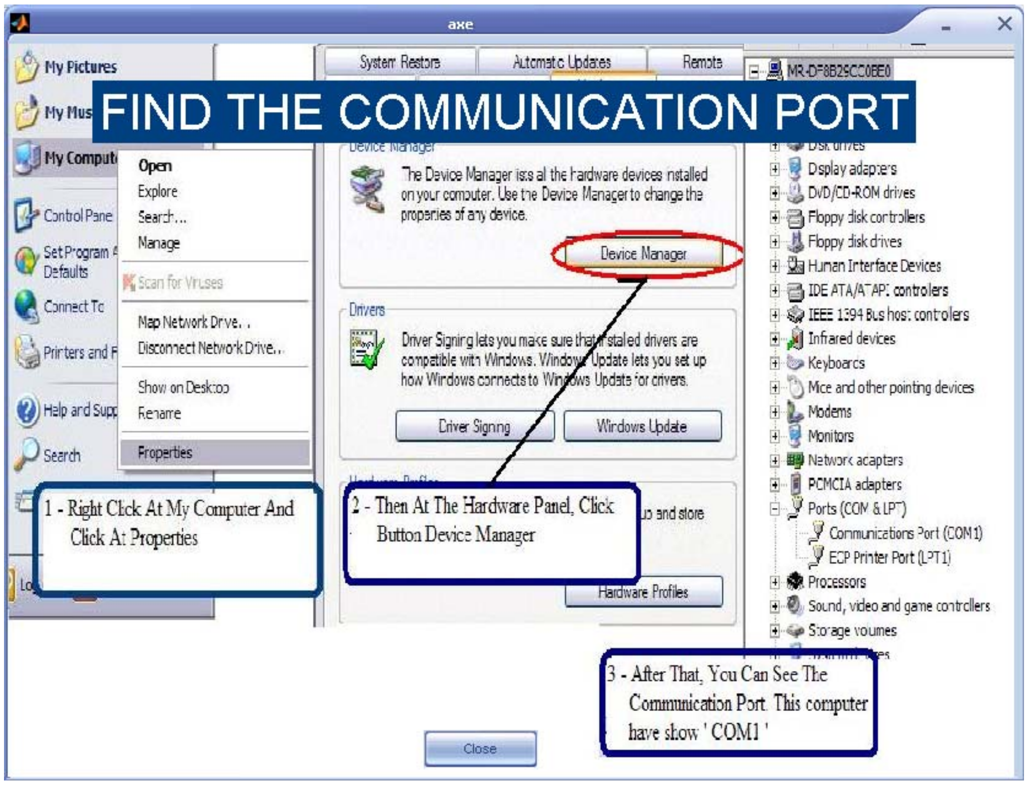

FIND THE COMMUNICATION PORT
1 - Right Click At My Computer And Click At Properties
2 - Then At The Hardware Panel, Click Button Device Manager
3 - After That, You Can See The Communication Port. This computer have show ' COM1 '
Device Manager
Communications Port (COM1)
ECP Printer Port (LPT1)
Close

Figure 4.20: Find the Communication Port

CHAPTER 5:

CONCLUSION AND RECOMMENDATION

5.1 Conclusion

From this project, the implement of PIC and MATLAB GUI has been presented. There are many function in using MATLAB, with the function in MATLAB, it can create GUIDE and design the layout of the GUI. From the GUI, it can show many thing based on its application. Based on this project, the GUI is creating to display the information of the signal from flow sensor. But to display the information of the signal the sensor must be interface to computer using PIC where it's the other part of this project. Through the study of this project, it's difficult to interface the PIC and MATLAB GUI because the PIC and MATLAB GUI need the right coding and program to interface between these two parts. The graph that shown in GUI is the actual signal that came from the sensor.

The objective of this project is to interface the MATLAB GUI and flow sensor that is achieved. The main contribution of this project is to interfacing the GUI with air flow sensor.

5.2 Future Recommendations

For the future development, to improve this project many other sensors can be use besides this flow sensor for this project such as temperature, movement and encoder sensor. These varieties of sensor can detect difference input and provided difference output that can be display at MATLAB GUI. From this development, it can add more features and functions to this project. In future development, maybe air flow sensor can be changing or adding with other flow sensor like solid, liquid or gas flow sensor.

To make this project more advance and interesting is with adding the feedback to this project. The sensor and the speed of fan can be control with GUI and then the sensor will detect the movement and will display the signal data back to GUI. The GUI design can be improve with additional more features and make more interesting like use slider to control the speed and pop-up menu to select difference task. Also can be adding toggle button, check box or list box for this project in future development.

5.3 Costing and Commercialization

For this topic, it will discuss about the future of this project for commercialization. The total cost of this project is dividing for 2 parts that is for hardware and software. For the hardware, it is include voltage regulator, PIC16F877A, MAX232, DB9 connecter and air flow sensor. The total cost for this project is about RM200. The air flow sensor is quite expensive. The second hand part for this air flow sensor is around RM100 and above. For the software, it is very expensive because this project is involving with MATLAB. The license for this MATLAB must be renewing every year and it is very high cost.

For the commercialization, this project is a very big opportunity for commercialization. This is because, the MATLAB GUI is quite difficult and not many user know and using it to interface with their hardware. Many users use visual basic to interface with their hardware because visual basic is easy to learn. The student can use this reference project for their further study. The industry side can implement such this project to their own development. Maybe the Industry can use this project when there are sensor usages. They can set the sensor to be more accurately and also they can check the performance of their sensor with just click at computer (MATLAB GUI). For the further development, this project can make easier to anybody who involve with any sensor and MATLAB GUI.

REFERENCE

[1] bellevuelinux. Retrieved on 21 January 2007 from
http://www.bellevuelinux.org/gui.html

[2] Ghaphical User Interface Retrieved on 21 January 2007 from
http://www.ewh.ieee.org/r8/uae/GUI.pdf

[3] MATLAB - The Language of Technical Computing Retrieved on
15 February 2007 from
http://www.mathworks.com/access/helpdesk/help/pdf_doc/matlab/buildgui.pdf

[4] MATLAB GUI Retrieved on 15 February 2007 from
http://rclsgi.eng.ohio-state.edu/~gateway/docs/pres_and_prog_reports/MatlabGUI.ppt

[5] PIC Microcontroller Retrieved on 15 February 2007 from
http://en.wikipedia.org/wiki/PIC_microcontroller

[6] 20 February 2007, Citing Internet source URL
http://whyfiles.larc.nasa.gov/text/kids/Problem_Board/problems/light/glossary.html

[7] 20 February 2007, Citing Internet source URL
http://www.pcmag.com/encyclopedia_term/0,2542,t=sensor&i=51107,00.asp

[8] 20 February 2007, Citing Internet source URL
www.stjude.org/glossary

[9] 20 February 2007, Citing Internet source URL
www.ueidaq.com/support/glossary/S/

[10] 20 February 2007, Citing Internet source URL
http://www.definethat.com/define/1916.htm

[11] Sensor Retrieved on 21 February 2007 from
http://en.wikipedia.org/wiki/Sensor

[12] 21 February 2007, Citing Internet source URL
www.ticms.com/wizard/glossary.htm

[13] 21 February 2007, Citing Internet source URL
www.gaf.de/presshelp/glossary/p81.htm

[14] Flow Sensor Retrieved on 22 February 2007 from
http://en.wikipedia.org/wiki/Flow_sensor

[15] Flow Meter Retrieved on 22 February 2007 from
http://www.flowmeterdirectory.com/flowmeter_flowsensors.html

[16] Flow Sensor Retrieved on 22 February 2007 from
http://www.flowmeterdirectory.com/flow_sensors.html

[17] Analog to Digital Converter Retrieved on 22 February 2007 from
http://en.wikipedia.org/wiki/Analog-to-digital_converter

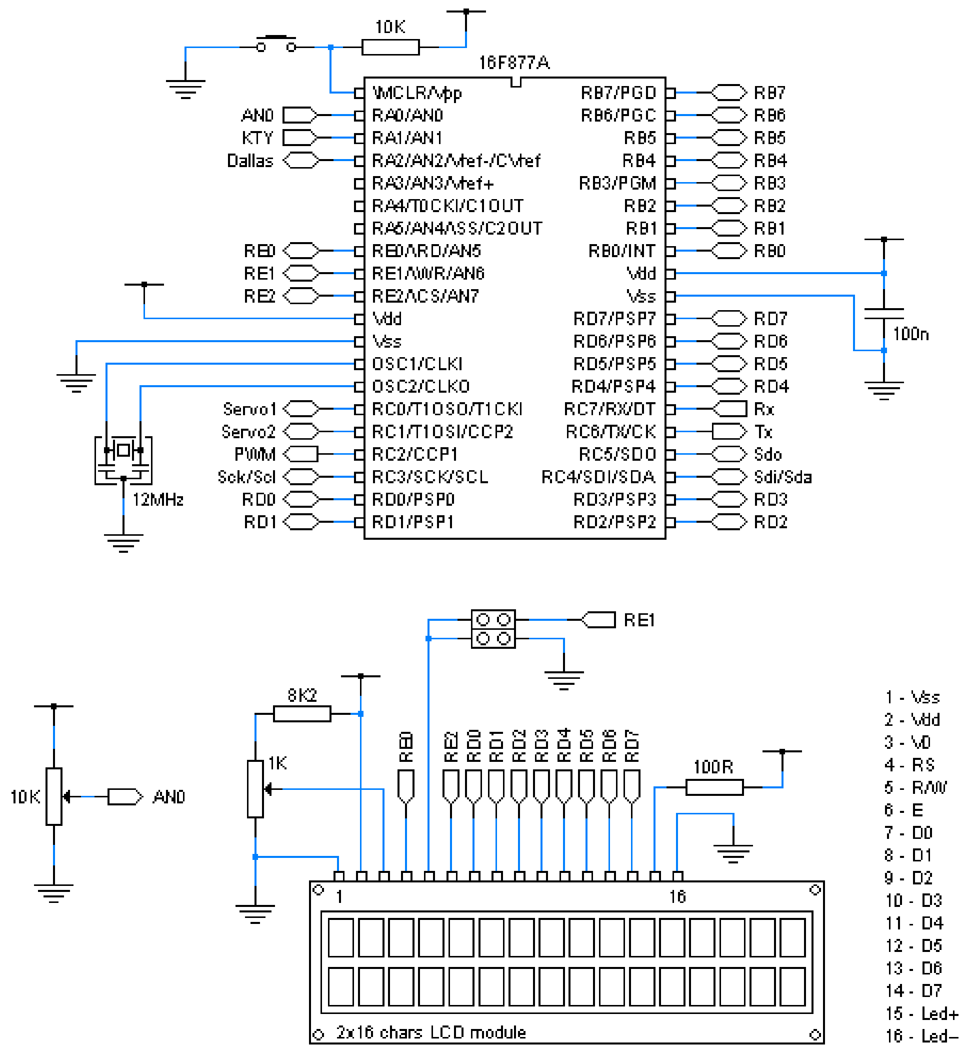
[18] Ladder Logic Diagram Retrieved on 11 July 2007 from
http://Cq.ladder.cl

[19] Duane Hanselman and Bruce Little Field. *Mastering MATLAB 7*,
: Pearson/ Prentice Hall. 2005

[20] Yan-Fang Li, Saul Harari, Hong Wong and Vikram Kapila (2004).
Matlab-Based Graphical User Interface Development for Basic Stamp 2
Microcontroller Project: Polytechnic University, Brooklyn New York

APPENDIX A

EXAMPLE 1: Reading analog input

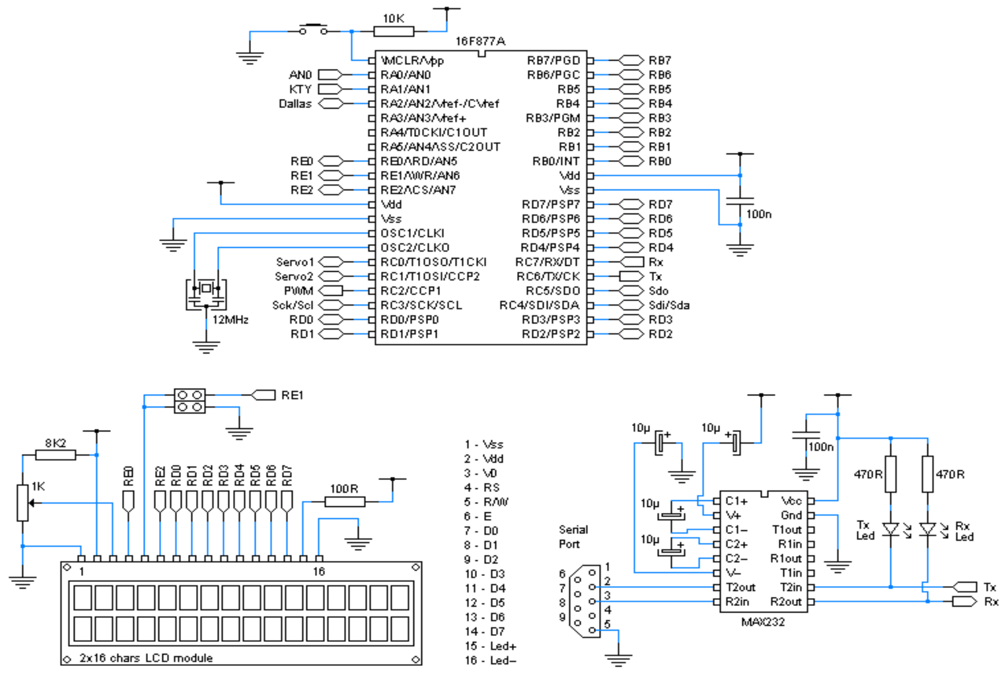


<u>BASIC SOURCE PROGRAM:</u>

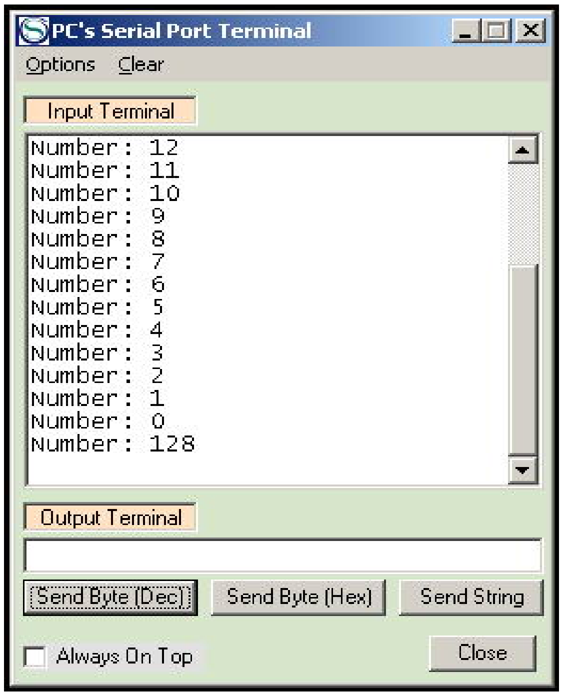
```
Define CONF_WORD = 0x3f72
Define CLOCK_FREQUENCY = 12
AllDigital
ADCON1 = 0x0e

Define LCD_BITS = 8
Define LCD_DREG = PORTD
Define LCD_DBIT = 0
Define LCD_RSREG = PORTE
Define LCD_RSBIT = 0
Define LCD_RWREG = PORTE
Define LCD_RWBIT = 1
Define LCD_EREG = PORTE
Define LCD_EBIT = 2
Define LCD_READ_BUSY_FLAG = 1
Lcdinit

Dim an0 As Word

loop:
	Adcin 0, an0
	Lcdcmdout LcdClear
	Lcdout "Analog input AN0"
	Lcdcmdout LcdLine2Home
	Lcdout "Value: ", #an0
	WaitMs 250
Goto loop
```

APPENDIX B

EXAMPLE 2: RS-232 communication with PC



The screenshot of the PIC Simulator IDE serial port terminal:



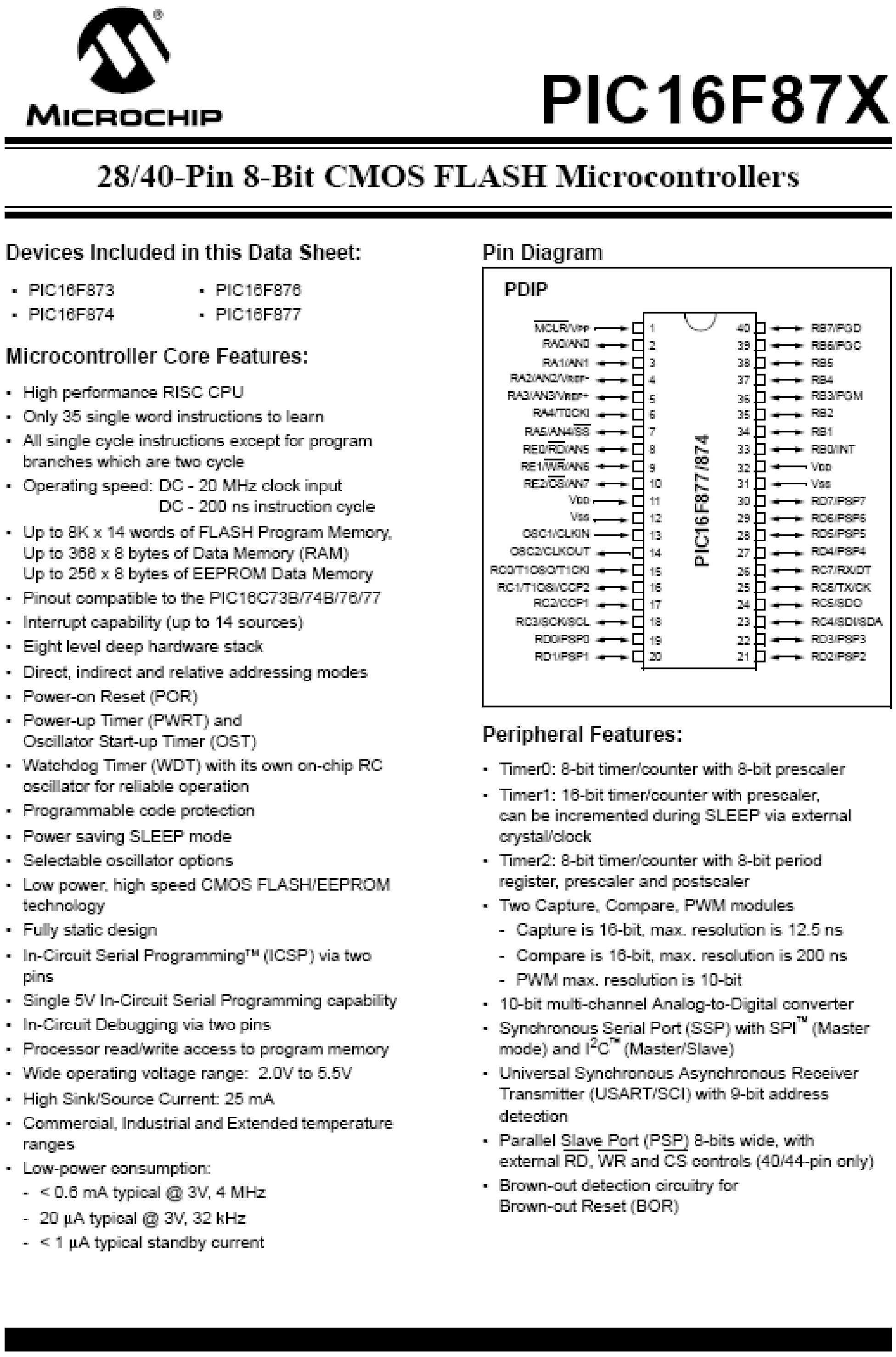
```
Number: 12
Number: 11
Number: 10
Number: 9
Number: 8
Number: 7
Number: 6
Number: 5
Number: 4
Number: 3
Number: 2
Number: 1
Number: 0
Number: 128
```

<u>BASIC SOURCE PROGRAM:</u>

```
Define CONF_WORD = 0x3f72
Define CLOCK_FREQUENCY = 12
AllDigital

Define LCD_BITS = 8
Define LCD_DREG = PORTD
Define LCD_DBIT = 0
Define LCD_RSREG = PORTE
Define LCD_RSBIT = 0
Define LCD_RWREG = PORTE
Define LCD_RWBIT = 1
Define LCD_EREG = PORTE
Define LCD_EBIT = 2
Define LCD_READ_BUSY_FLAG = 1
Lcdinit

Dim i As Byte

Hseropen 19200
WaitMs 1000

For i = 20 To 0 Step -1
        Hserout "Number: ", #i, CrLf
        Lcdcmdout LcdClear
        Lcdout "Number: ", #i
        WaitMs 1000
Next i

loop:
        Hserin i
        Hserout "Number: ", #i, CrLf
        Lcdcmdout LcdClear
        Lcdout "Number: ", #i
Goto loop
```

APPENDIX C

Data Sheet for PIC16F877



MICROCHIP

PIC16F87X

28/40-Pin 8-Bit CMOS FLASH Microcontrollers

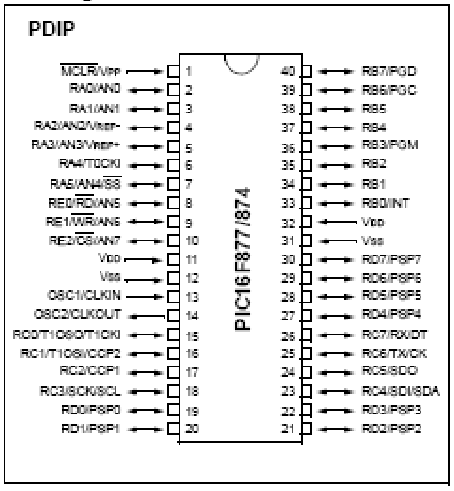
Devices Included in this Data Sheet:

- PIC16F873
- PIC16F874
- PIC16F876
- PIC16F877

Microcontroller Core Features:

- High performance RISC CPU
- Only 35 single word instructions to learn
- All single cycle instructions except for program branches which are two cycle
- Operating speed: DC - 20 MHz clock input
  DC - 200 ns instruction cycle
- Up to 8K x 14 words of FLASH Program Memory,
  Up to 368 x 8 bytes of Data Memory (RAM)
  Up to 256 x 8 bytes of EEPROM Data Memory
- Pinout compatible to the PIC16C73B/74B/76/77
- Interrupt capability (up to 14 sources)
- Eight level deep hardware stack
- Direct, indirect and relative addressing modes
- Power-on Reset (POR)
- Power-up Timer (PWRT) and Oscillator Start-up Timer (OST)
- Watchdog Timer (WDT) with its own on-chip RC oscillator for reliable operation
- Programmable code protection
- Power saving SLEEP mode
- Selectable oscillator options
- Low power, high speed CMOS FLASH/EEPROM technology
- Fully static design
- In-Circuit Serial Programming™ (ICSP) via two pins
- Single 5V In-Circuit Serial Programming capability
- In-Circuit Debugging via two pins
- Processor read/write access to program memory
- Wide operating voltage range: 2.0V to 5.5V
- High Sink/Source Current: 25 mA
- Commercial, Industrial and Extended temperature ranges
- Low-power consumption:
  - < 0.6 mA typical @ 3V, 4 MHz
  - 20 µA typical @ 3V, 32 kHz
  - < 1 µA typical standby current

Pin Diagram

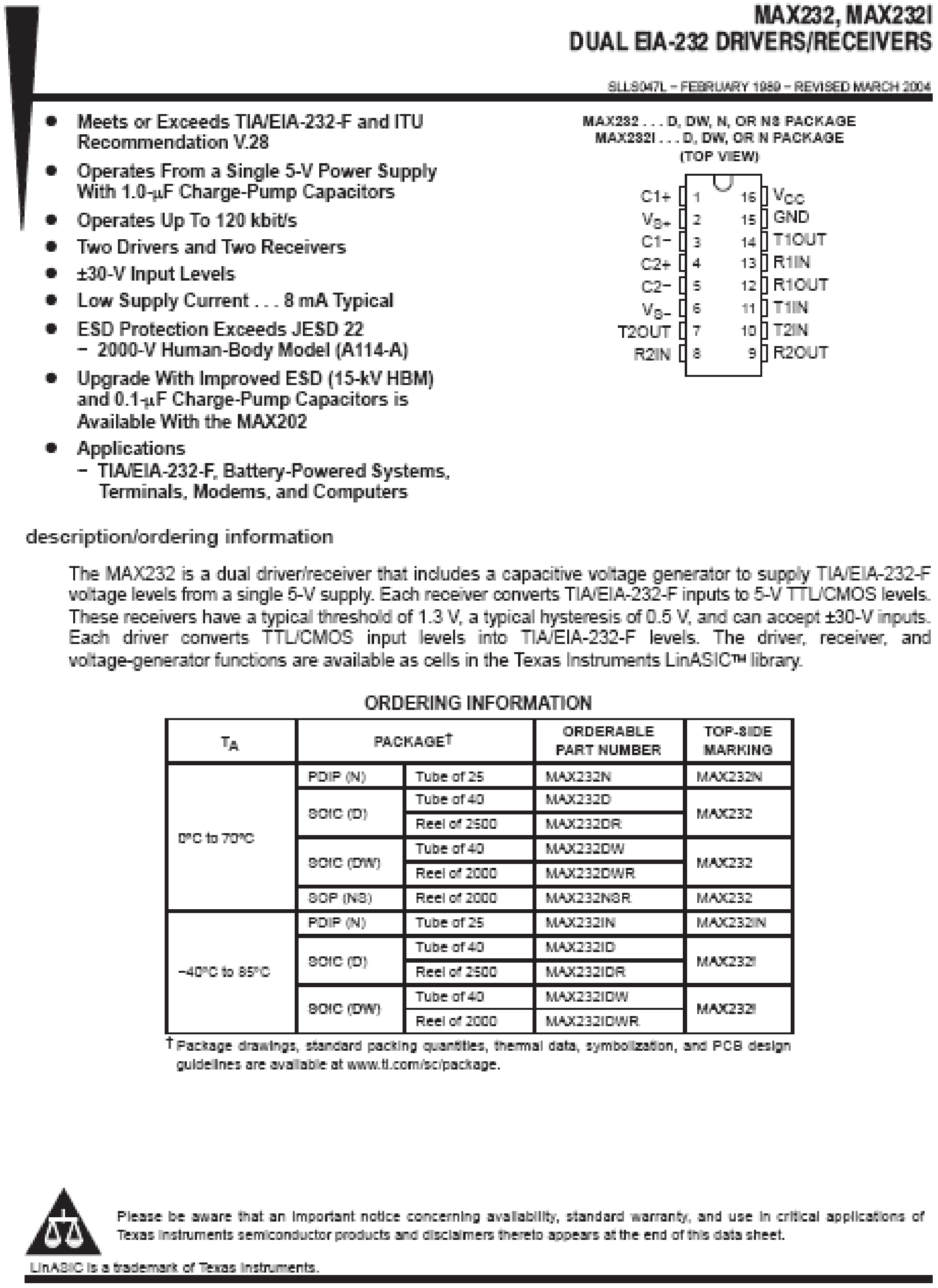


Peripheral Features:

- Timer0: 8-bit timer/counter with 8-bit prescaler
- Timer1: 16-bit timer/counter with prescaler, can be incremented during SLEEP via external crystal/clock
- Timer2: 8-bit timer/counter with 8-bit period register, prescaler and postscaler
- Two Capture, Compare, PWM modules
  - Capture is 16-bit, max. resolution is 12.5 ns
  - Compare is 16-bit, max. resolution is 200 ns
  - PWM max. resolution is 10-bit
- 10-bit multi-channel Analog-to-Digital converter
- Synchronous Serial Port (SSP) with SPI™ (Master mode) and I^{2}C™ (Master/Slave)
- Universal Synchronous Asynchronous Receiver Transmitter (USART/SCI) with 9-bit address detection
- Parallel Slave Port (PSP) 8-bits wide, with external $\overline{\text{RD}}$, $\overline{\text{WR}}$ and $\overline{\text{CS}}$ controls (40/44-pin only)
- Brown-out detection circuitry for Brown-out Reset (BOR)

APPENDIX D

Datasheet for MAX232

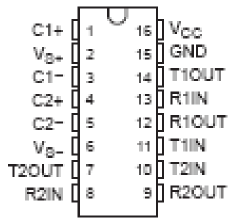


MAX232, MAX232I
DUAL EIA-232 DRIVERS/RECEIVERS

SLLS047L – FEBRUARY 1989 – REVISED MARCH 2004

- Meets or Exceeds TIA/EIA-232-F and ITU Recommendation V.28
- Operates From a Single 5-V Power Supply With 1.0-μF Charge-Pump Capacitors
- Operates Up To 120 kbit/s
- Two Drivers and Two Receivers
- ±30-V Input Levels
- Low Supply Current . . . 8 mA Typical
- ESD Protection Exceeds JESD 22
  - 2000-V Human-Body Model (A114-A)
- Upgrade With Improved ESD (15-kV HBM) and 0.1-μF Charge-Pump Capacitors is Available With the MAX202
- Applications
  - TIA/EIA-232-F, Battery-Powered Systems, Terminals, Modems, and Computers

MAX232 . . . D, DW, N, OR NS PACKAGE
MAX232I . . . D, DW, OR N PACKAGE
(TOP VIEW)



description/ordering information

The MAX232 is a dual driver/receiver that includes a capacitive voltage generator to supply TIA/EIA-232-F voltage levels from a single 5-V supply. Each receiver converts TIA/EIA-232-F inputs to 5-V TTL/CMOS levels. These receivers have a typical threshold of 1.3 V, a typical hysteresis of 0.5 V, and can accept ±30-V inputs. Each driver converts TTL/CMOS input levels into TIA/EIA-232-F levels. The driver, receiver, and voltage-generator functions are available as cells in the Texas Instruments LinASIC™ library.

ORDERING INFORMATION

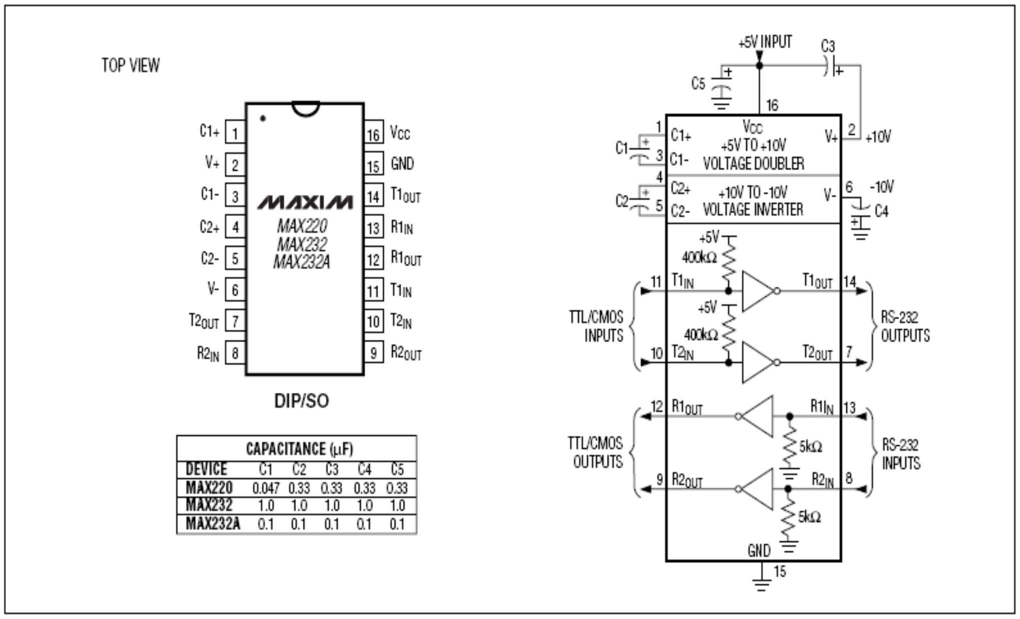
T_A	PACKAGE†		ORDERABLE PART NUMBER	TOP-SIDE MARKING
0°C to 70°C	PDIP (N)	Tube of 25	MAX232N	MAX232N
	SOIC (D)	Tube of 40	MAX232D	MAX232
		Reel of 2500	MAX232DR	
	SOIC (DW)	Tube of 40	MAX232DW	MAX232
		Reel of 2000	MAX232DWR	
	SOP (NS)	Reel of 2000	MAX232NSR	MAX232
−40°C to 85°C	PDIP (N)	Tube of 25	MAX232IN	MAX232IN
	SOIC (D)	Tube of 40	MAX232ID	MAX232I
		Reel of 2500	MAX232IDR	
	SOIC (DW)	Tube of 40	MAX232IDW	MAX232I
		Reel of 2000	MAX232IDWR	

† Package drawings, standard packing quantities, thermal data, symbolization, and PCB design guidelines are available at www.ti.com/sc/package.

Please be aware that an important notice concerning availability, standard warranty, and use in critical applications of Texas Instruments semiconductor products and disclaimers thereto appears at the end of this data sheet.

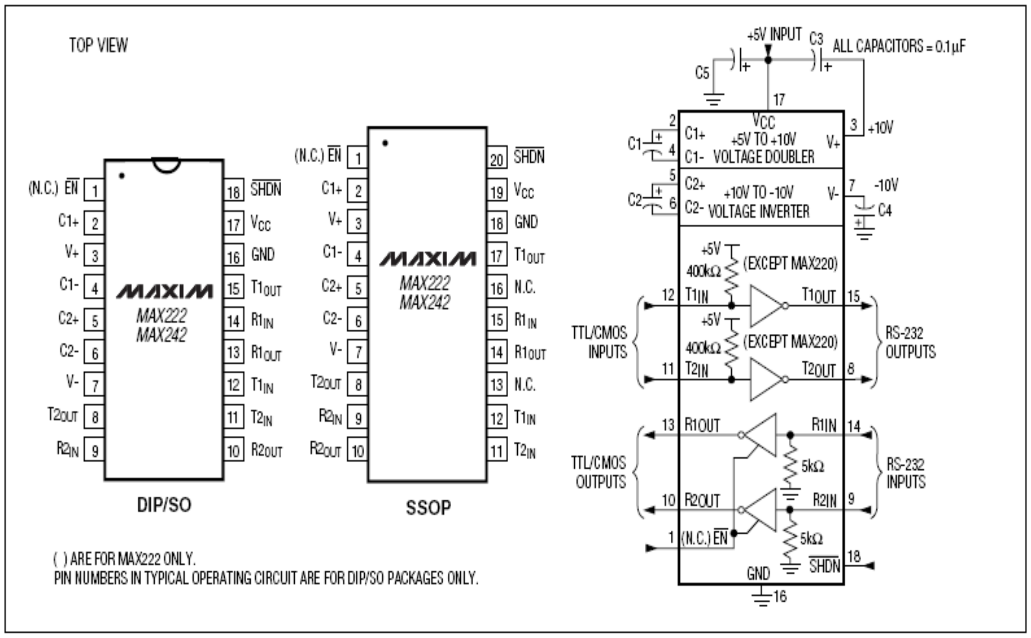
LinASIC is a trademark of Texas Instruments.

+5V-Powered, Multichannel RS-232 Drivers/Receivers



CAPACITANCE (μF)					
DEVICE	C1	C2	C3	C4	C5
MAX220	0.047	0.33	0.33	0.33	0.33
MAX232	1.0	1.0	1.0	1.0	1.0
MAX232A	0.1	0.1	0.1	0.1	0.1

MAX220/MAX232/MAX232A Pin Configuration and Typical Operating Circuit



MAX222/MAX242 Pin Configurations and Typical Operating Circuit

APPENDIX E

Programming M-file for Main Menu

```
function varargout = Project3(varargin)
% PROJECT3 M-file for Project3.fig
%      PROJECT3, by itself, creates a new PROJECT3 or raises the existing
%      singleton*.
%
%      H = PROJECT3 returns the handle to a new PROJECT3 or the handle to
%      the existing singleton*.
%
%      PROJECT3('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in PROJECT3.M with the given input
arguments.
%
%      PROJECT3('Property','Value',...) creates a new PROJECT3 or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before Project3_OpeningFunction gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to Project3_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Copyright 2002-2003 The MathWorks, Inc.

% Edit the above text to modify the response to help Project3

% Last Modified by GUIDE v2.5 29-Oct-2007 23:15:18

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Project3_OpeningFcn, ...
                   'gui_OutputFcn',  @Project3_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
```

```
end
% End initialization code - DO NOT EDIT


% --- Executes just before Project3 is made visible.
function Project3_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Project3 (see VARARGIN)
movegui('center')
% Choose default command line output for Project3
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes Project3 wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = Project3_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in pushbutton1.
function varargout = pushbutton1_Callback(h, eventdata, handles, varargin)
figure(Info)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes on button press in pushbutton2.
function varargout = pushbutton2_Callback(h, eventdata, handles, varargin)
figure(Cre)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes on button press in pushbutton3.
```

```
function varargout = pushbutton3_Callback(h, eventdata, handles, varargin)
figure(sensor2)
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes on button press in pushbutton4.
function pushbutton4_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
user_response = close_all('Title','Project3');
switch lower(user_response)
case 'no'
        % take no action
case 'yes'
        % Prepare to close GUI application window
        %           .
        %           .
        %           .
%       delete(handles.figure1)
end



% --- Executes during object creation, after setting all properties.
function axes1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to axes1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: place code in OpeningFcn to populate axes1
[x,map] = imread('backg','jpg');
      image(x)
     set(gca,'visible','off')



% --- Executes on button press in pushbutton5.
function varargout = pushbutton5_Callback(h, eventdata, handles, varargin)
figure(help)
% hObject    handle to pushbutton5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

APPENDIX F

Programming M-File for Air Flow Sensor

```
function varargout = MyPro1(varargin)
% MYPRO1 M-file for MyPro1.fig
%      MYPRO1, by itself, creates a new MYPRO1 or raises the existing
%      singleton*.
%
%      H = MYPRO1 returns the handle to a new MYPRO1 or the handle to
%      the existing singleton*.
%
%      MYPRO1('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in MYPRO1.M with the given input arguments.
%
%      MYPRO1('Property','Value',...) creates a new MYPRO1 or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before MyPro1_OpeningFunction gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to MyPro1_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Copyright 2002-2003 The MathWorks, Inc.

% Edit the above text to modify the response to help MyPro1

% Last Modified by GUIDE v2.5 30-Oct-2007 12:57:00

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @MyPro1_OpeningFcn, ...
                   'gui_OutputFcn',  @MyPro1_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
```

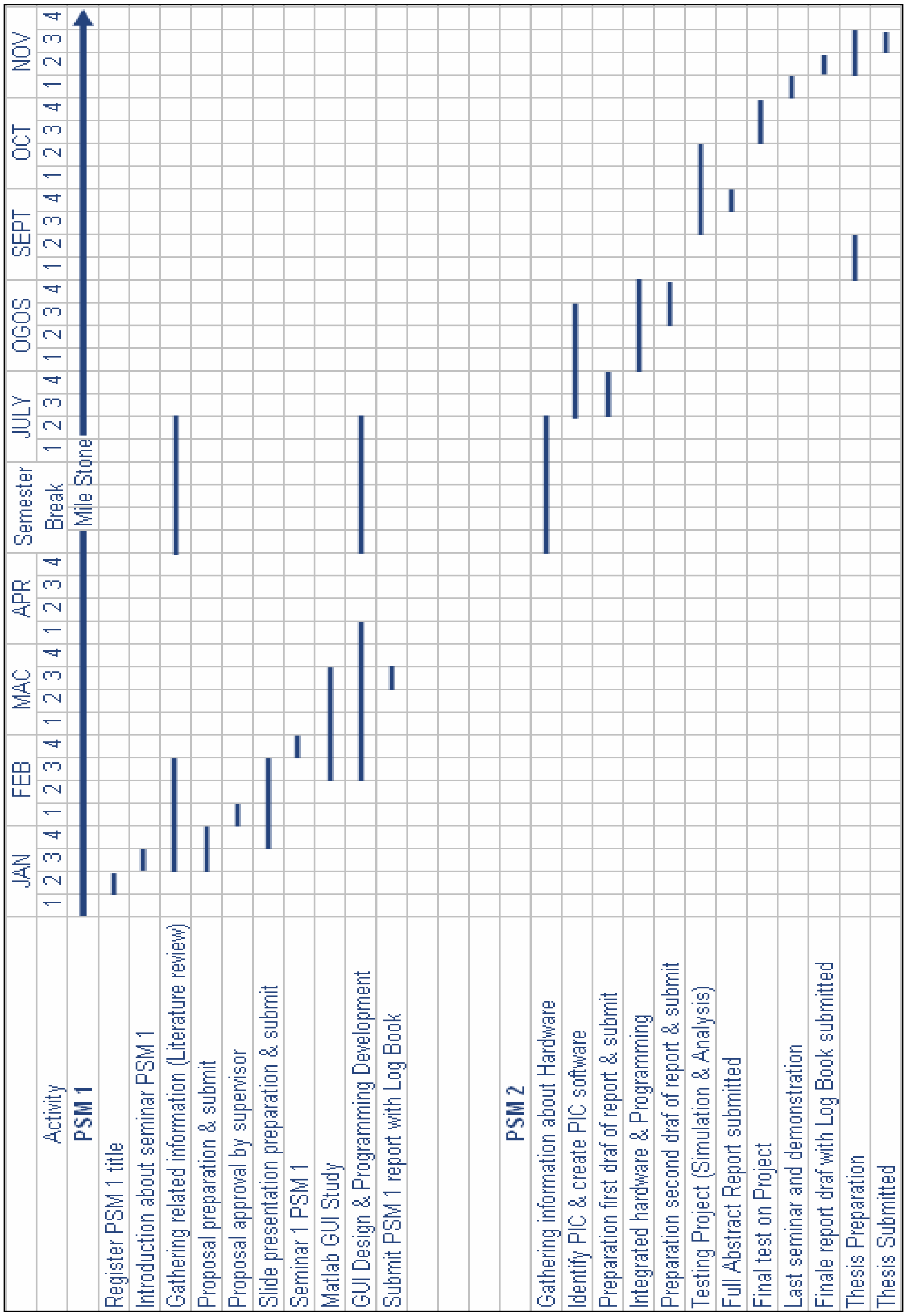
```
% End initialization code - DO NOT EDIT


% --- Executes just before MyPro1 is made visible.
function MyPro1_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to MyPro1 (see VARARGIN)

movegui('center')
s=serial('com1')
c=s.status
handles.status=c
% set(s,'OutputBufferSize',4096)
handles.op=s  % store data
guidata(hObject, handles); %save data

% Choose default command line output for MyPro1
handles.output=hObject;
axes(handles.axes4)
[x,map] = imread('asal','jpg');
        image(x)
       set(gca,'visible','off')


axes(handles.axes5)
[x,map] = imread('mafs','jpg');
        image(x)
       set(gca,'visible','off')

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes MyPro1 wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = MyPro1_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;
```

```
% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
s=handles.op
handles.output = hObject;
axes(handles.axes1)

out=fread(s)
plot(out);
set(findobj(gca,'Type','line','Color',[0 0 1]),'Color',[0 1 0],'LineWidth',1.0)
set(gca,'color',[0.027 0.702 0.894])
grid on




% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
user_response = closed('Title','flow4');
switch lower(user_response)
case 'no'
        % take no action
case 'yes'
        % Prepare to close GUI application window
        %           .
        %           .
        %           .
        delete(handles.figure1)
end




% --- Executes on button press in checkbox1.
function checkbox1_Callback(hObject, eventdata, handles)
% hObject    handle to checkbox1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of checkbox1
% --- Executes on button press in pushbutton4.
function pushbutton4_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

m=1:0.1:50;
n=-m;
c=m+n;
plot(c);
set(findobj(gca,'Type','line','Color',[0 0 1]),'Color',[1 1 1],'LineWidth',2.5)
set(gca,'color',[1 1 1])
%s set(gca,'color',[0.027 0.702 0.894])




% --- Executes on button press in radiobutton1.
function radiobutton1_Callback(hObject, eventdata, handles)
% hObject    handle to radiobutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of radiobutton1
if (get(hObject,'Value')==get(hObject,'Max'));
    s=handles.op  % retrieve data
    fopen(s)
    guidata(hObject,handles);
    axes(handles.axes4)
    [x,map] = imread('open','jpg');
        image(x)
       set(gca,'visible','off')

     %save data    ;
else
    s=handles.op
    fclose(s)
    guidata(hObject,handles)
    axes(handles.axes4)
    [x,map] = imread('clo','jpg');
        image(x)
       set(gca,'visible','off')

end        %bawah end ni mungkin ada handles gak kot
guidata(hObject,handles);
```

```
% --- Executes on button press in pushbutton5.
function pushbutton5_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% s=handles.op

% handles.output = hObject;
% axes(handles.axes3)

% if 125.5<out<255
%     then out=5
%
%
% if 0<out<125
%     then out=0







% --- Executes on button press in pushbutton6.
function pushbutton6_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
s=handles.op
c=handles.status
b=s.status

set(handles.text6,'string',b)





% --- Executes on button press in pushbutton7.
function varargout = pushbutton7_Callback(h, eventdata, handles, varargin)
figure(flow)
% hObject    handle to pushbutton7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

APPENDIX G

Ghant Chart


Activity
JAN
FEB
MAC
APR
Semester Break
JULY
OGOS
SEPT
OCT
NOV
Mile Stone
PSM 1
Register PSM 1 title
Introduction about seminar PSM 1
Gathering related information (Literature review)
Proposal preparation & submit
Proposal approval by supervisor
Slide presentation preparation & submit
Seminar 1 PSM 1
Matlab GUI Study
GUI Design & Programming Development
Submit PSM 1 report with Log Book
PSM 2
Gathering information about Hardware
Identify PIC & create PIC software
Preparation first draf of report & submit
Integrated hardware & Programming
Preparation second draf of report & submit
Testing Project (Simulation & Analysis)
Full Abstract Report submitted
Final test on Project
Last seminar and demonstration
Finale report draf with Log Book submitted
Thesis Preparation
Thesis Submitted