MICROCONTROLLER BASED LIQUID VOLUME FLOW

RATE METERING SYSTEM

By

NORARFAH BINTI SUUT

FINAL PROJECT REPORT

Submitted to the Electrical & Electronics Engineering Programme

in Partial Fulfillment of the Requirements

for the Degree

Bachelor of Engineering (Hons)

(Electrical & Electronics Engineering)

Universiti Teknologi Petronas

Bandar Seri Iskandar

31750 Tronoh

Perak Darul Ridzuan

CERTIFICATION OF APPROVAL

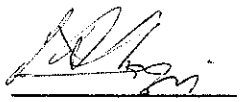
MICROCONTROLLER BASED LIQUID VOLUME FLOW RATE METERING SYSTEM

by

Norarfah binti Suut

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

Approved by,



(Dr. Nazir Ahmed Arian)
Project Supervisor

UNVERSITI TEKNOLOGI PETRONAS
TRONOH, PERAK
December 2006

CERTIFICATION OF ORIGINALITY

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

(Norarfah binti Suut)

ABSTRACT

This document presents the project entitled Microcontroller-Based Liquid Volume-Flow-Rate Metering System. The proposed system is composed of hardware and software structures. The hardware structure consists of a flow sensor, solenoid valve, and microcontroller circuitry. Flow sensor senses the liquid flows in the pipeline. The solenoid valve shall be the final control element to allow appropriate flow to be controlled in the metering system. The microcontroller circuitry is equipped with LCD and keypad. LCD displays the flow rate of the liquid flowing in the pipeline meanwhile keypad allows the user to input data of the volume. The software structure commands the whole process via the microcontroller input/output ports. In general, the principle of sensing, measurement, and control is used in this project.

ACKNOWLEDGEMENT

Firstly I would like to praise Allah the Almighty, whom has helped and guided me in completing my second semester of FYP. My deepest gratitude goes to the University Technology of PETRONAS, Electrical and Electronic Engineering department where students were trained with essential skills to excel in theoretical and practical work.

I have come to rely on several individuals for guidance and support for my Final Year Project .In particular, I would like to thank my supervisor, Dr. Nazir Ahmed Arian. I have vast respect for him and am thankful to have had the opportunity to work with him.

Next, I would like to express my appreciation to the following individuals for their continuous help, support and guidance in technical professionalism during my Final Year Project.

- Kak Siti Hawa, FYP Lab Technician
- Mr Zuki, Microprocessor 1 Lecturer
- Kak Siti Fatimah, Microprocessor Lab Technician

I would also like to thank my family for their unconditionally support of my academic activities over the years. Last but not least, I would like to give credits to my fellow friends especially Adzni, Asnur, Alin, Epul, Supik, Hambali, Ariff, Yani, Sim, who have helped me a lot while doing the troubleshooting in the lab. Thank you so much.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

LCD	Liquid Crystal Display
A/D	Analog to Digital
D/A	Digital to Analog
EEPROM	Electrically Erasable Programmable Read-Only Memory
RAM	Random Access Memory
USART	Universal Synchronous–Asynchronous Receiver/Transmitter
I/O	Input/Output
PC	Personal Computer
EPROM	Erasable Programmable Read-Only Memory
PIC	Peripheral Interface Controller
LP	Low Power
XT	Standard Crystal
HS	High Speed
ASCII	American Standard Code for Information Interchange
CCS	Custom Computer Services
ADC	Analog to Digital Converter
DC	Direct Current
AC	Alternating Current
FSR	File Select Register
PVC	Polyvinyl Chloride

CHAPTER 1
INTRODUCTION

1.1 Background of Study

Flow meters are used to measure the flow of air, fluids and gas. Some of the most popular flow meters are water flow meters, fuel flow meters, and air flow meters. An example of units of measurement for flow meters is volumetric flow rate. Volumetric flow rate is measured in liters per minute, liters per second and cubic centimeters per minute

Flow meters are useful in a variety of fields due to their varied uses. Example of industries that benefit from the use of water flow meters include the petroleum and gas industries, utility services, food processing and raw materials industry.

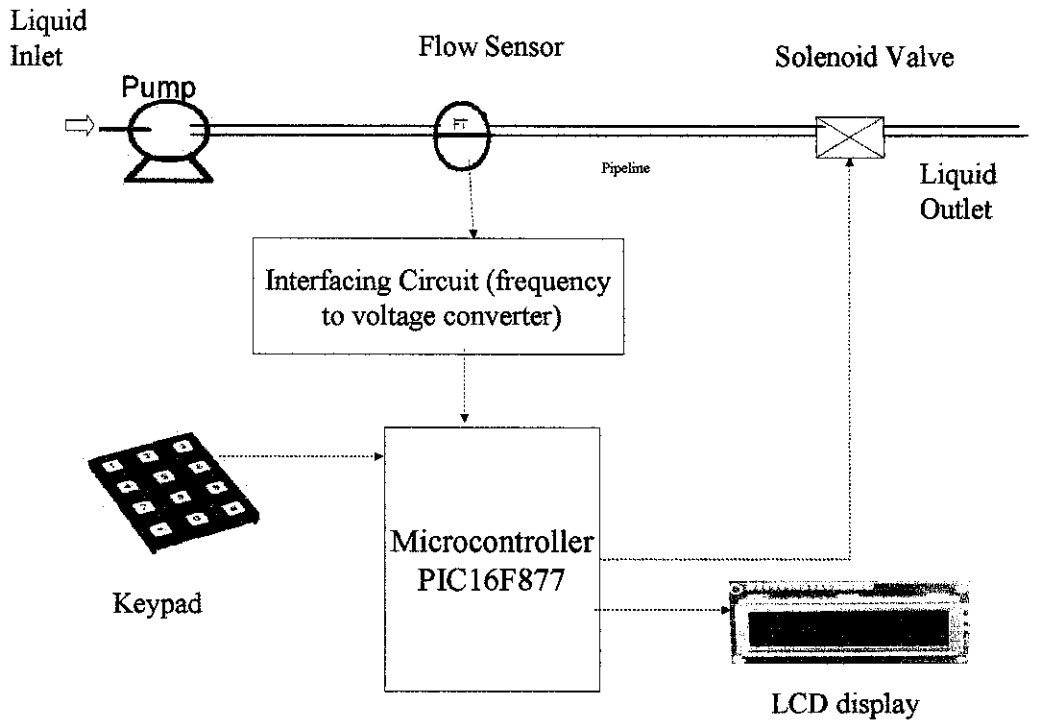


Figure A: The flow metering system of the project

The flow metering system proposed for the project will be basis on interfacing circuit, microcontroller, display, keypad, solenoid valve and flow sensor. The pump will provide certain pressure range for the liquid to flow. Then, the flow sensor will sense the liquid flowing through the pipeline.

Initially, the user will need to enter the input data which is volume of the liquid by using keypad. The microcontroller will process the data entered and send the output for display and to actuate the solenoid valve.

1.2 Problem Statement

Industrial automatic controllers have been available for years but have a price tag in the thousands of dollars. On the other hand, microcontroller chips are very inexpensive, usually costing much less than $100. They are very small, can operate under very severe environmental conditions, and lend themselves to new, simplified programming procedures. These features make the microcontroller the preferred choice for embedded control, such as that used in appliances and industrial automation.

1.3 Objectives

The objectives of this project are to:

1. Design and integrate the hardware of the metering system
2. The system measure the volume-flow-rate (liters/minute) and volume flow (liters) of liquid flowing through a pipe
3. The system should include the usage of microcontroller
4. The system should be easy to learn and use for everybody.

1.4 Scope of Study

There are several topics and issues that must be considered before proceeding any further in the design of the system. The scope of study depends mainly on these few areas:

- controlling the sensor as input and produce the output
- interfacing the controller to the LCD display and solenoid valve

The applications and devices that are considered must be reliable, cost-effective and practical for a feasible implementation. This will ensure that the design produced is economical and marketable.

The PIC16F877 microcontroller is made the microcontroller of choice for the specified task. This choice is obvious as this microcontroller is competitively priced, not to mention its sound applicability in this system.

CHAPTER 2
LITERATURE REVIEW

2.1 Microcontroller

A microcontroller is a single-chip computer, including most of a computer's features, but in limited sizes. Today, there are hundreds of different types of microcontrollers, ranging from 8-pin devices to 40-pin, or even 64- or higher pin devices.

2.1.1 Microcontroller Applications

The evolution of the microprocessor technology is marked by a major fork. One branch of the evolutionary tree is represented by Pentium chips and Power-PC chips which form the heart of personal computers. This mentioned branch is mainly focused upon enhancing raw computers. Meanwhile, the other branch, which is 30 times larger in unit volume, is represented by microcontrollers. Basically, in microcontroller technology, the emphasis is upon the integration of many features needed in a single chip so that the chip has versatility to sense inputs and control outputs in a device or instrument. Mainly, the 8-bit microcontroller market is dominated by Motorola and Microchip Technology. All these devices are optimized to meet stringent performance requirements in cost-effective applications.

After 30 years of constant evolution in chip manufacturing, this embedded technology is no stranger is the daily-used applications. Generally, they can be divided to 5 broad markets:

- Consumer segment – home appliances and home entertainment
- Automotive – a modern car has nearly 50 microcontrollers providing intelligence and control, like keyless entry, antilock braking and air bas.
- Office automation – PCs, keyboards, copiers, printers etc

- Telecommunications – cell phones, pagers and answering machines
- Industrial products – door locks in hotel rooms, automatic faucets and industrial machinery

Among the main features that contribute to the wide popularity of microcontrollers are:

- Speed
- Instruction set simplicity
- Integration of operational features
- Programmable timer options
- Interrupt control
- Powerful output pin control
- I/O port expansion
- Serial programming via two pins
- EPROM support
- Mail-order support
- Free assembler and simulator

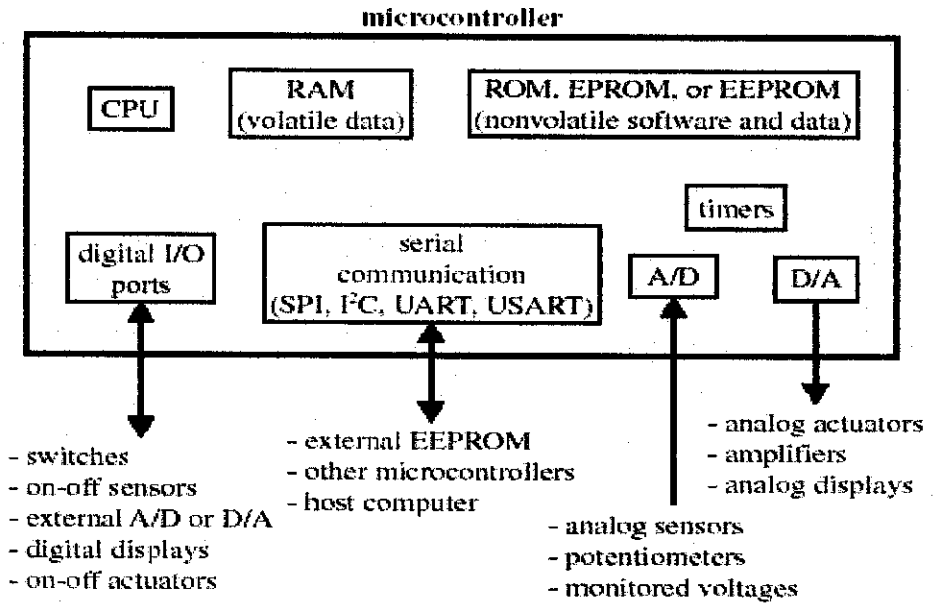


Figure 1: Components of a Microcontroller

2.1.2 PIC16F877 Microcontroller

The PIC16F877 is an 8-bit, 40-pin microcontroller with the following features:

- operation up to 20 MHz;
- 8K flash program memory;
- 368 bytes RAM memory;
- 256 bytes electrically erasable programmable read-only memory (EEPROM) memory;
- 15 types of interrupts;
- 33 bits of parallel I/O capability;
- 2 timers;
- universal synchronous–asynchronous receiver/transmitter (USART) serial communications;
- 10-bit, 8-channel A/D converter;
- 2 analog comparators;
- 33 instructions;
- programming in assembly or high-level languages;
- low cost

Flash memory is nonvolatile and is used to store the user program. This memory can be erased and reprogrammed electrically. EEPROM memory is used to store nonvolatile user data and can be written to or read from under program control. The microcontroller has 8K program memory, which is quite large for control based applications. In addition, the RAM memory is 368 bytes, which again is quite large for control based applications. Most microcontrollers have the built-in circuits necessary for computer control applications. For example, a microcontroller may have A/D converters so that the external signals can be sampled. They also have parallel input–output ports so that digital data can be read or output from the microcontroller. Some devices have built-in D/A converters and the output of the converter can be used to drive the plant through an actuator (e.g. an amplifier). Microcontrollers may also have built-in timer and interrupt

logic. Using the timer or the interrupt facilities, the microcontroller can be programmed to implement the control algorithm accurately [1].

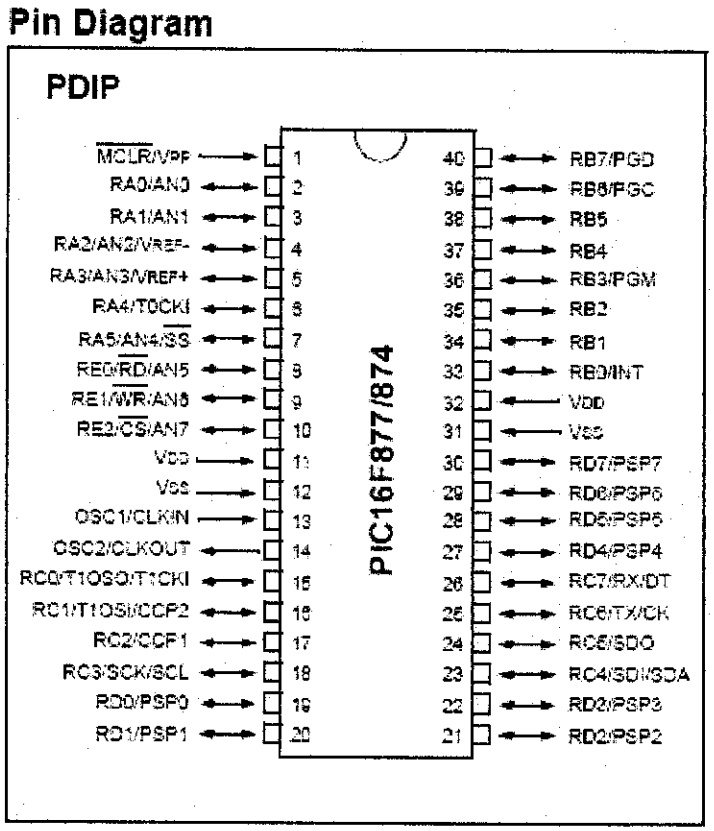


Figure 2: PIC 16F877

Nowadays microcontrollers can be programmed using high level languages such as BASIC, PASCAL or C. High-level languages offer several advantages compared to the assembly language:

- It is easier to develop programs using a high-level language.
- Program maintenance is much easier if the program is developed using a high-level language.
- Testing a program developed in a high-level language is much easier.
- High-level languages are more user-friendly and less prone to making errors.
- It is easier to document a program developed using a high-level language.

In addition to the above advantages, high-level languages have some disadvantages. For example, the length of the code in memory is usually larger when a high-level language is used, and the programs developed using the assembly language usually run faster than those developed using a high-level language [2].

There are four oscillator options for the microcontroller, of which the single pin RC oscillator provides a low-cost solution, the LP oscillator minimizes power consumption, XT is a standard crystal, and the HS is for High Speed crystals.

2.1.3 PIC 16F877 Analog to Digital Conversion

Analog to Digital Conversion is useful for many purposes. Often, a simple ON or OFF (1 or 0 which is the only thing the digital ports can do) isn't enough. Sometimes, the microcontroller needs to carry out different tasks at intermediate levels between on and off. On the PIC16F877, only A & E ports can do A/D conversion.

By default, the A/D conversion outputs an 8 bit number. This means a conversion value with a range between 0-255 (2^8). For example, since max input voltage is 5V, an A/D conversion value of 128 would indicate an input of 2.5V

At the beginning of the source file before the main statements, input the following:

#device PIC16f877 ADC= 8

The conversion from bits to actual voltage (0-5V) is:

output_value * (5v / precision)

where precision is 255 (8 bit) or 1023 (10 bit) [3].

2.2 Instrumentation and devices

2.2.1 Flow Sensor

Flow rate measurements are important in control of chemical processes. Flow rate is the quantity of a substance passed through the given cross-section area in the unit of time. Volumetric flow rate is expressed in m^3/s, m^3/h, cm^3/s, l/s etc. Usually the volumetric or mass flow rates are applied to liquids since liquids are incompressible.

Flowing systems require energy, typically provided by pumps and compressors, to produce a pressure difference as the driving force, and flow sensors should introduce a small flow resistance, increasing the process energy consumption as little as possible.

For this project, the flow sensor chosen is bought from RS manufacturer. Its pulses are proportional to the fluid flow rate. From the datasheet of the flow sensor, the flow sensor will give the pulse output versus flow rates as shown below.

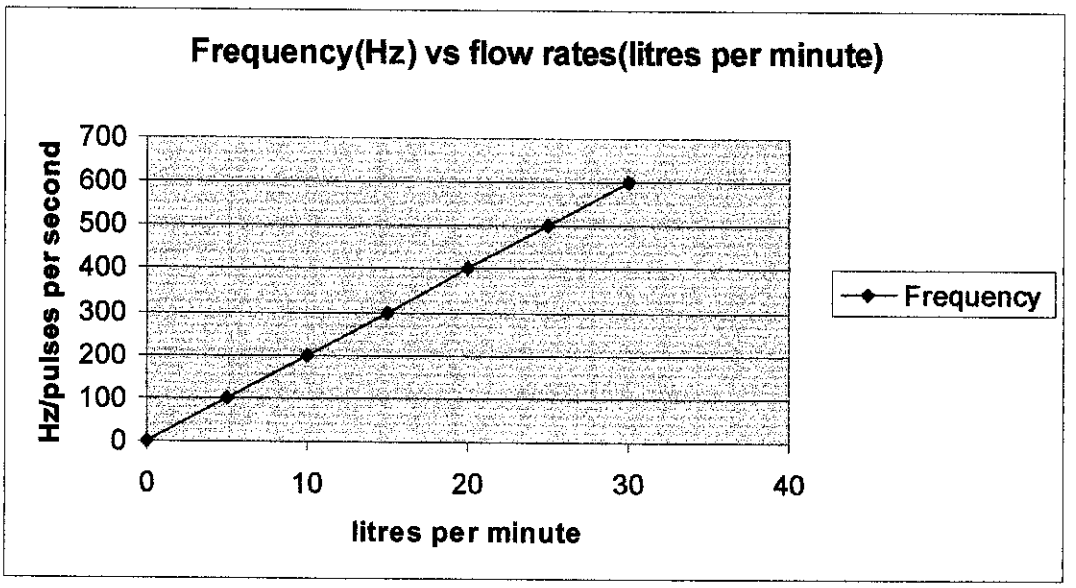


Figure 3 : Frequency versus flow rates for flow sensor

Based on the graph plotted, it can be said that the relation between the frequency (y-axis) and the flow rates (x-axis) are

$$y=20x$$

Flow Sensor Electronic Components

Inside the housing of the flow sensor is an infrared light emitting diode. The LED is directed at the receiver which has a built-in voltage regulator, photo diode, amplifier, Schmitt trigger and output stage. Every turbine blade passage blocks the light beam and so reduces the level of the output signal to typically 200mV (maximum 400mV). The unblocked level is the supply voltage and the output is tied to this through a 10k resistor. The output rise and fall times are typically 60ns and 6ns respectively. At the maximum continuous operating temperature (70°C) the power dissipation of the detector output stage is 2.5mV; care must therefore be taken not to overload the output. At ambient temperatures the unit can dissipate up to 250mW.

The power to the LED is dependent on supply current, and this must be limited with a series resistor and potentiometer. Current required is subject to optical density of medium being measured. Maximum recommended current = 30mA [4]. The figure of the electronic circuit of the flow sensor is attached in the Appendices.

2.2.2 Solenoid Valve

Solenoid valves are used wherever fluid flow has to be controlled automatically. They are being used to an increasing degree in the most varied types of plants and equipment. The variety of different designs which are available enables a valve to be selected to specifically suit the application in question.

Solenoid valves are control units which, when electrically energized or de-energized, either shut off or allow fluid flow. The actuator takes the form of an electromagnet. When energized, a magnetic field builds up which pulls a plunger or pivoted armature against the action of a spring. When de-energized, the plunger or pivoted armature is returned to its original position by the spring action [5].

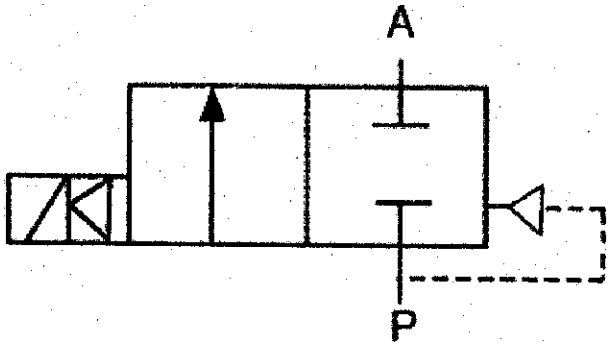


Figure 4 : 2 Way Water Solenoid Valve

The diagram describes the valves operation when the coil is energized and de-energised.

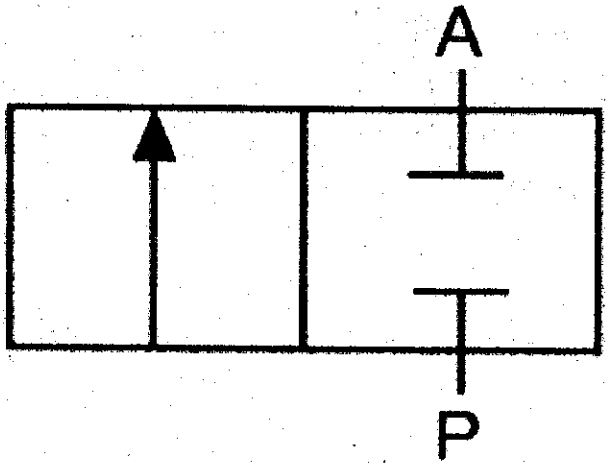


Figure 5: A 2-way normally closed valve. When the coil is energised the valve is open.

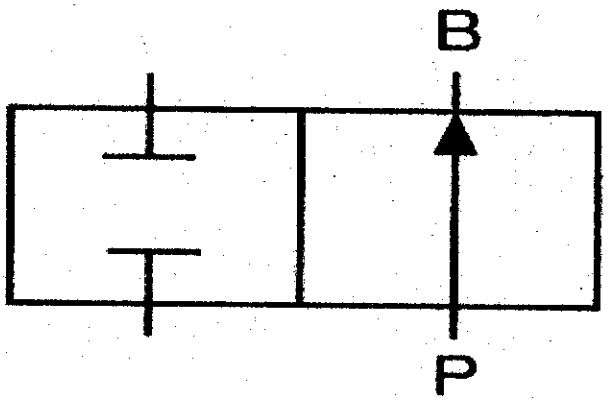


Figure 6: De-energizing closes the valve.

2.3 Interfacing circuit

2.3.1 Frequency to voltage converter circuit

An excellent integrated circuit to use for converting from frequency to voltage is the National Semiconductor LM2907. The LM2907 consists of three major components as shown in Figure 7.

- The first is a comparator with built-in hysteresis. This converts the sinusoidal waveform of the tachometer to a pulse train.
- The next component is a charge pump that injects a fixed amount of charge for each cycle of the pulse train. That is to say a I = dQ/dt which is forced through a resistor R1. Therefore the voltage across R1 is proportional to the frequency.
- The last component is an op-amp transistor combination that can be used to buffer the signal whose voltage is proportional to the tachometer frequency [6].

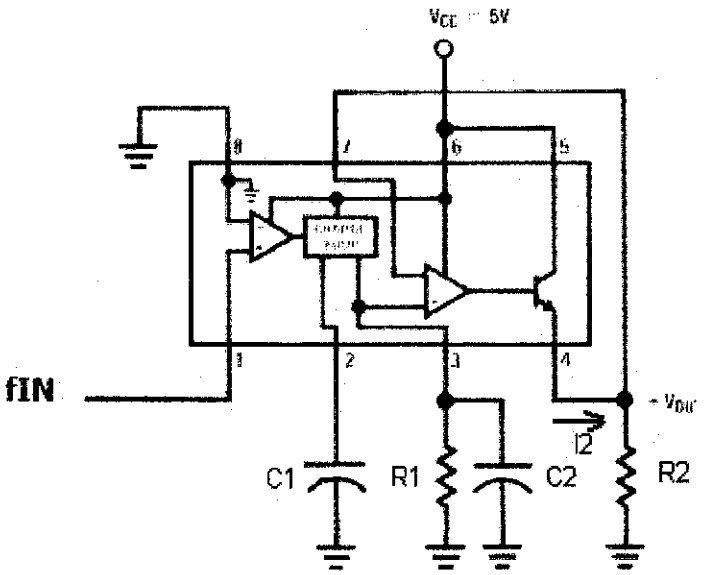


Figure 7: LM2907N-8 Block Diagram.

LM2907-8 has one input internally grounded so that an input signal must swing above and below ground and exceed the input thresholds to produce an output. Following the input stage is the charge pump where the input frequency is converted to a dc voltage. To do this requires one timing capacitor, one output resistor, and an integrating or filter capacitor.

$$Vo=Vcc \times f_{IN} \times R1 \times C1 \times K$$

K is the gain constant, which is typically 1.0 [6].

2.4 Input / Output Device

2.4.1 Liquid Crystal Display Module

LCD can add a lot to application in terms of providing useful interface for the user, debugging an application or just giving it a professional look. The most common type of LCD controller is the Hitachi 44780 which provides a relatively simple interface between a processor and an LCD.

The Character Set available in the 44780 is basically ASCII [8].

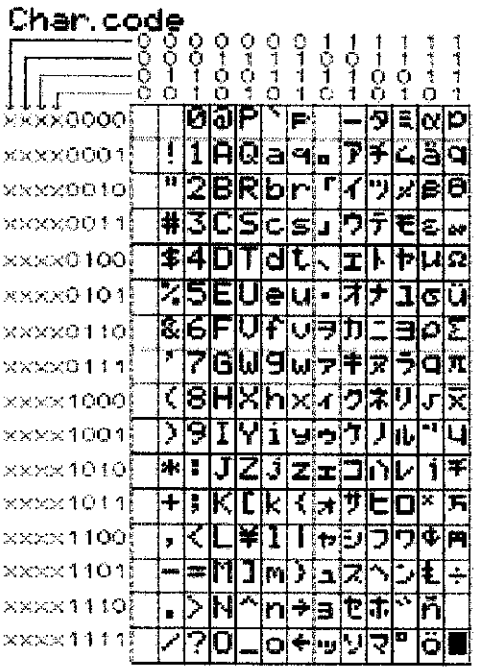


Figure 8: ASCII Code

2.4.2 Keypad

Using a keypad input in application is sometimes required for specific operations. It may be used to input data following a request or simply to trigger to initiate a particular operation. The keypad can be polled continuously for an input, or the keypad interface can be configured in an interrupt driven system, depending on the system requirements. There are two methods of operation: continuous polling, which most or all of its time check the keypad for a key press; and interrupt driven, which checks the keypad only when a key is pressed [9].

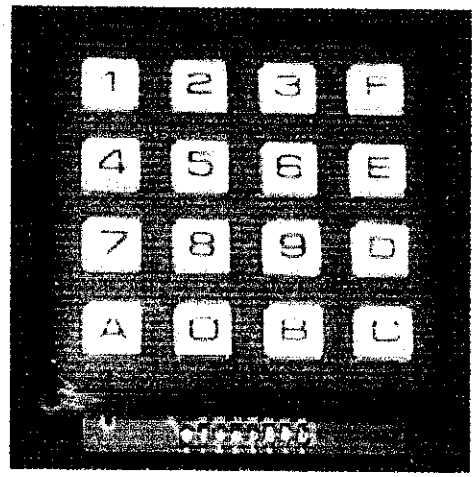


Figure 9 : Keypad (front view)

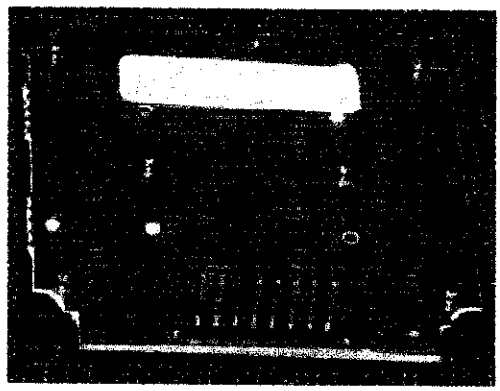


Figure 10: Keypad (rear view)

CHAPTER 3
SYSTEM DESIGN AND PROJECT WORK

3.1 Procedure Identification

As the title suggests, the procedures involved for implementation throughout the project work are divided into several systematic steps. The flow of the procedures for the design of the system is shown in the figure 11 below.

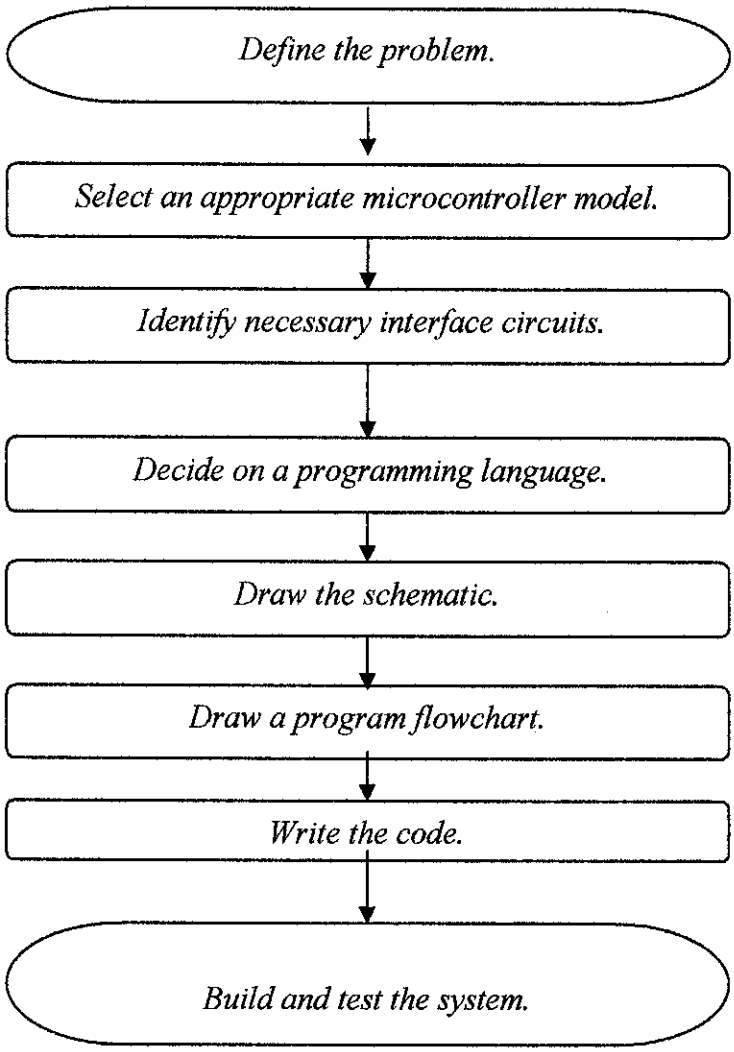


Figure 11 The design flow of the system

3.2 Tools Required

3.2.1 Hardware requirements

The microcontroller based liquid volume flow rate metering system will consist of the following:

1) Instrumentation and devices:

a) Flow sensor
b) Solenoid valve
c) Small pump

2) Interfacing Circuits:

a) Flow sensor to microcontroller

3) Metering controls and display:

a) Microcontroller
b) LCD display
c) Keypad

3.2.2 Software requirements

1. CCS Compiler - Microchip PIC C programming software
2. WARP-13 – Microchip PIC Programmer Burner

3.3 The system

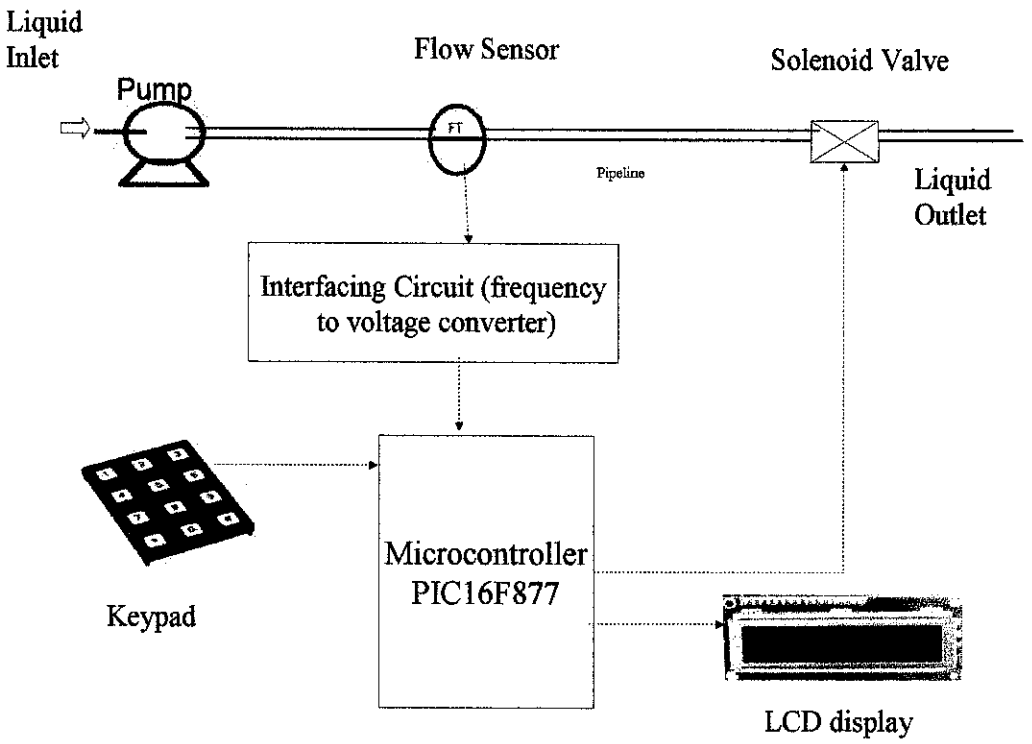


Figure 12: System Block diagram

The figure above shows the block diagram of the system. The pump initially will provide certain pressure range for the liquid to flow. Before the system begins, the user will be asked to enter data by using keypad for initialization of the volume flow of the liquid. The microcontroller will process the data entered.

When the liquid has flown through the pipeline, the flow sensor will sense the flow rate and give the pulse output to the interfacing circuit. The interfacing circuit between the flow sensor and PIC16F877, which is frequency to voltage converter, will convert the pulse per second output to voltage. This voltage signal will be sent to the analog to digital converter (ADC) pin of the microcontroller.

The microcontroller will calculate the flow rate and the volume of the liquid. The flow rate and the volume will be displayed at the LCD. The solenoid valve will be opened during the operation to let the liquid flow. When the volume flow has reached its maximum flow, the microcontroller will send signal to shut off the solenoid valve. The operation will terminate, and thus the pump will be automatically shut down.

3.4 Microcontroller

Microcontroller is divided into two important sections:

- Microcontroller circuitry
- Microcontroller programming

In microcontroller circuitry, it is important to specify in detail the pins involved for the input and output. This is to ensure that the microcontroller PIC 16F877 is utilized to its optimum performance.

- Port A is for analog to digital conversion from the flow sensor output
- Port B is used for LCD display
- Port C for Solenoid Valve
- Port D for keypad
- Port E is not used

Upon completion in defining the pin details for the microcontroller circuitry, the process of programming the PIC16F877 can commence. The C language is chosen to program the microcontroller. The programming must therefore coincide with all the pins that will be utilized. The three main considerations for the programming are:

- Input control
- Output display
- Output control

i) Input Control

The initializations for input control by using keypad will allow the microcontroller to initialize the volume of the liquid. The program must also include input control to allow the microcontroller to respond to the incoming input from the sensor.

ii) **Output Display**

The input data of the volume and flow rate of the system will be gathered by the microcontroller and will be sent for display to the LCD.

iii) Output Control

The program should include the control of the output (solenoid valve). The solenoid valve will be opened or closed based on the feedback of the volume gathered by the microcontroller.

The flow chart of the program is shown in figure 13.

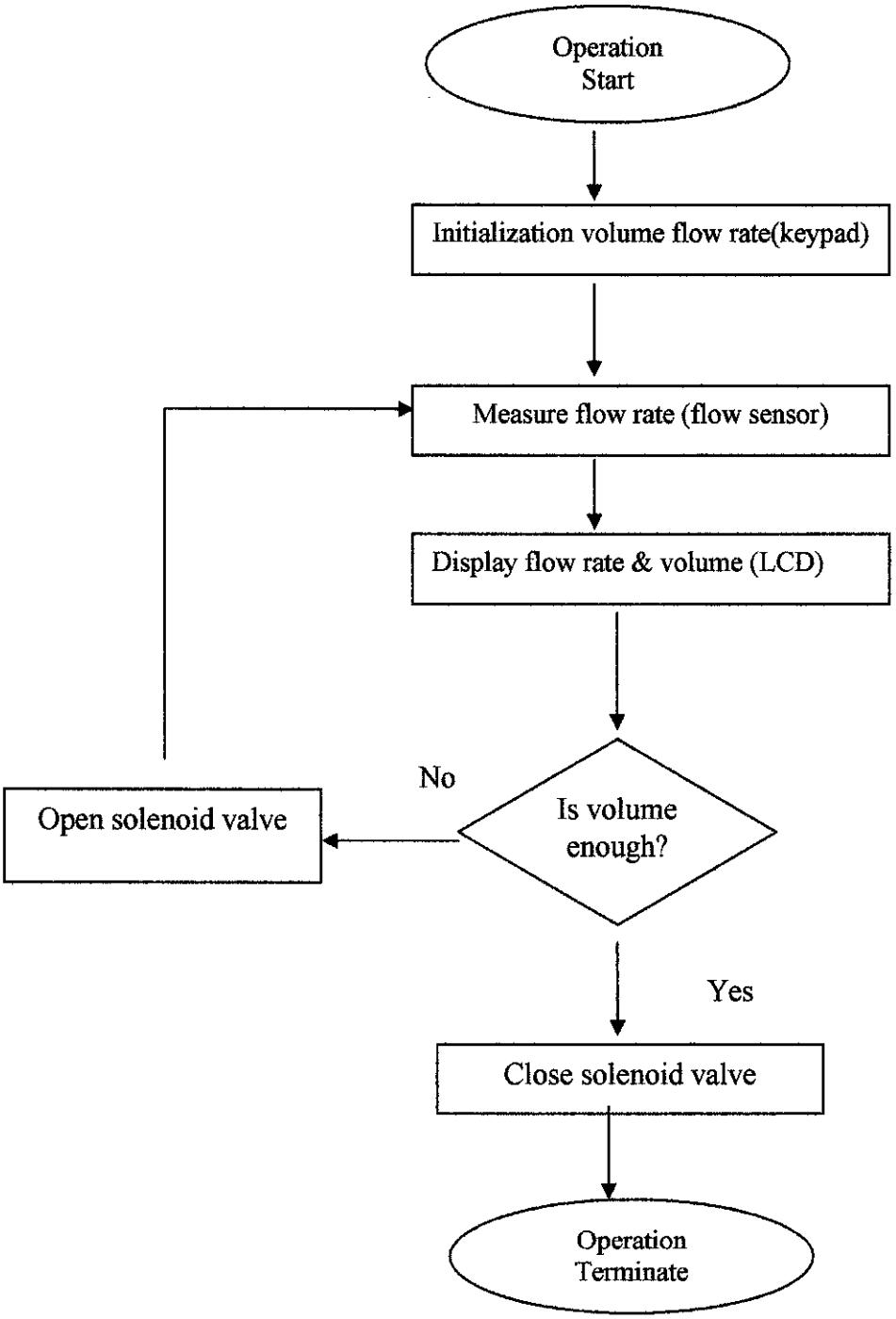


Figure 13: Main program flow chart

3.5 Project Work

3.5.1 PIC16F877 Microcontroller

Basically, the Peripheral Interface Controller or better known simply as PIC is the Microchip Technology's alternative to Motorola's MC68HC05 and MC68HC11 families. The PIC16F877 microcontroller is one of the types offered under the PIC microcontroller family. The features and the pin description of the PIC are described in tables below.

Technical/Catalog Information	PIC16F877
Category	Integrated Circuit (IC)
Family	Microcontrollers / Microprocessors
Vendor	Microchip Technology
Program Memory Size	8K
RAM Size	368
Number of I/O	33 pins
Maximum Frequency	20 MHz
Package / Case	SO-40 (40 pins)

Table 1 PIC16F877 features

Pin Number	Description
1, 11, 32	Positive 5V supply voltage for input/output pins
33 – 40 (port B)	LCD
10 (port A)	Flow sensor input signal A/D conversion
12, 31	Ground reference for input/output pins
13	Oscillator crystal input clock
23	Solenoid valve
19 – 22, 27 – 30 (port D)	Keypad

Table 2 PIC16F877 pin description

3.5.2 CCS Compiler

The CCS Compiler is programming software which is used to compile the program for PIC in C language. The procedure of using the software is shown in the figure at the next page.

1. Go to Project → New...

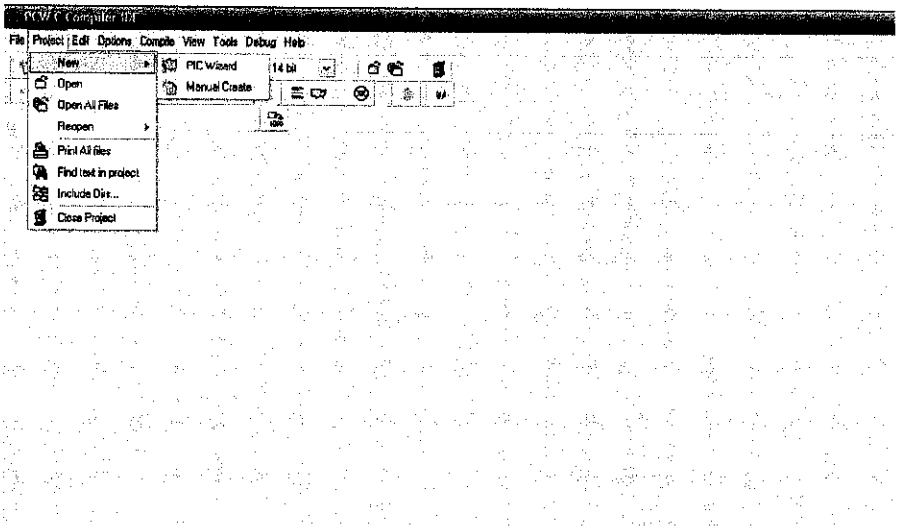


Figure 14 New Project

2. Select Manual create. Enter a name for the project and the target device.

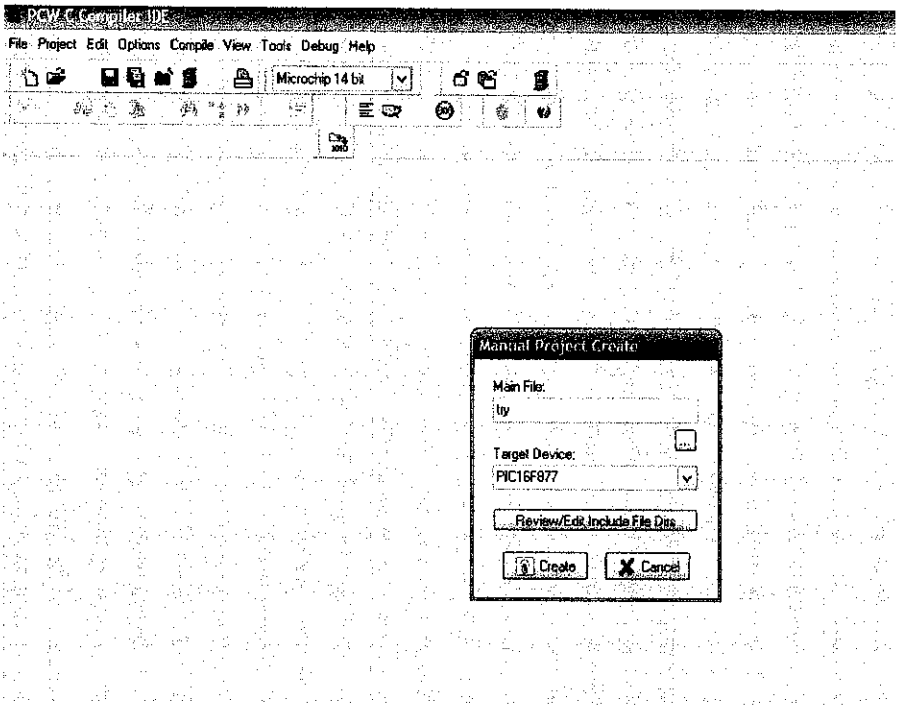


Figure 15 Manual Project Create

3. Include file directory..... and save.

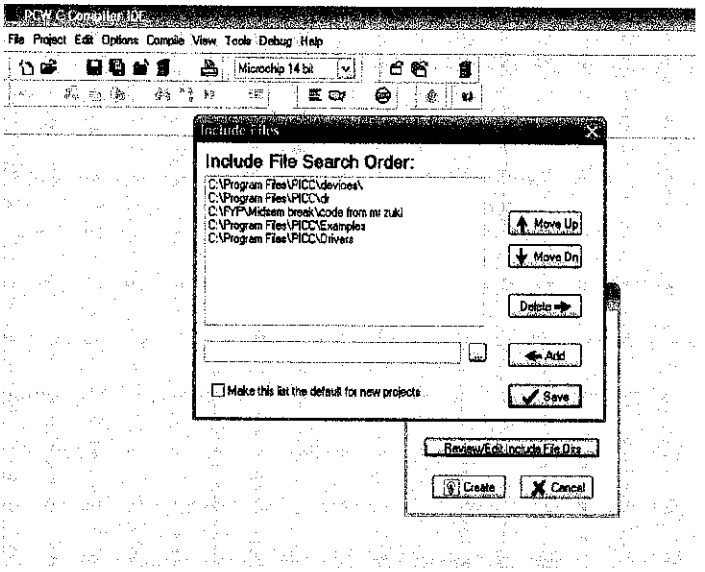


Figure 16 Include file directory

3. Click create and the project is created.

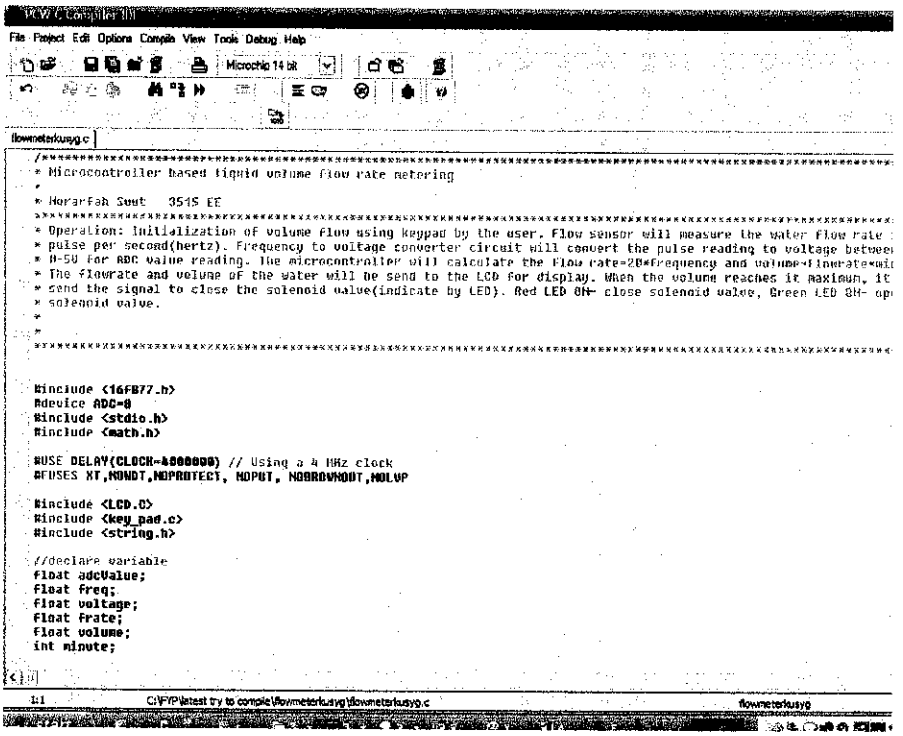


Figure 17 Create project

4. Lastly, compile the project.

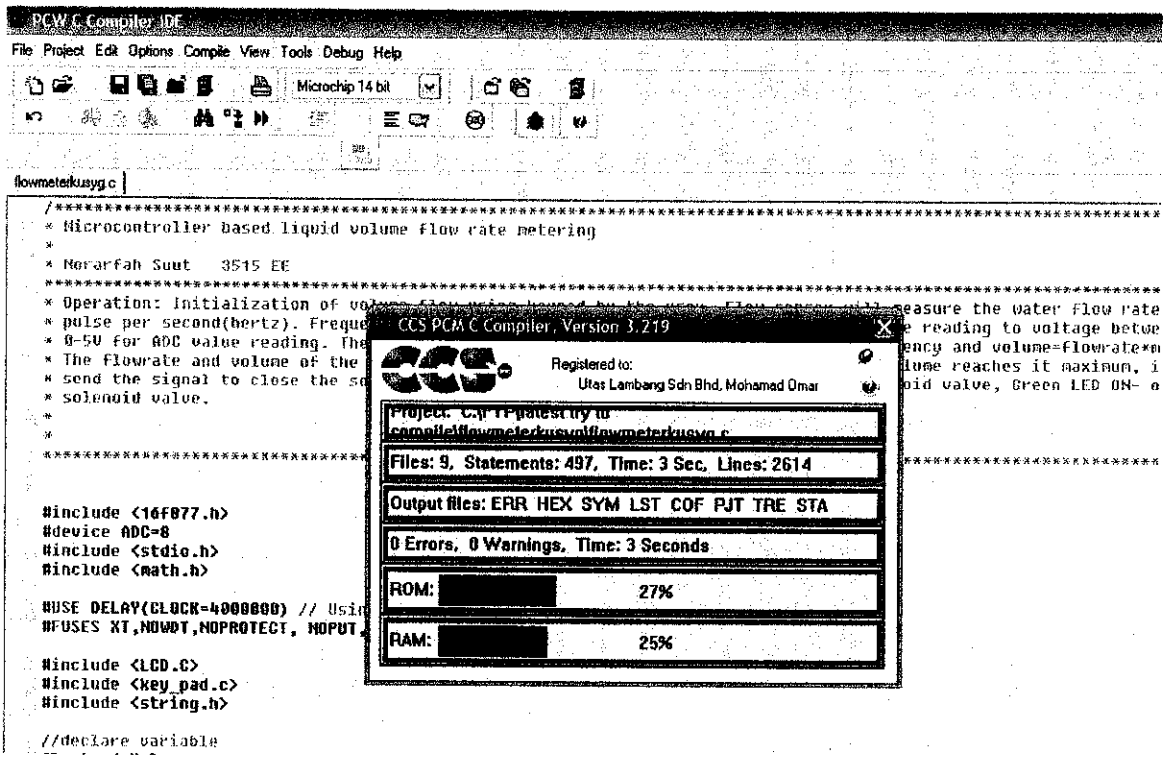


Figure 18 Compile the project

3.5. 3 Flow Sensor

The flow sensor detects the liquid flow in the pipe and converts the detected signals into electrical signal. The electrical signal will be sent to the microcontroller.

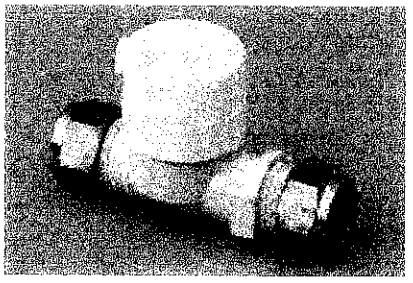


Figure 19: Liquid Flow Sensors-15mm diameter Pipe

The technical specification for the flow sensor chosen for this project is shown in Table 3.

Technical specification	
Recommended current	30mA
Operating free air temperature range	-40 °C to +70°C
Power dissipation	100mW
Operating supply voltage	4.5V - 16V (Vcc)
Output voltage - low	200mV
Output voltage - high	Vcc
Rise/fall time	60/6 ns

Table 3 : Technical specification for Liquid Flow Sensors-15mm diameter Pipe

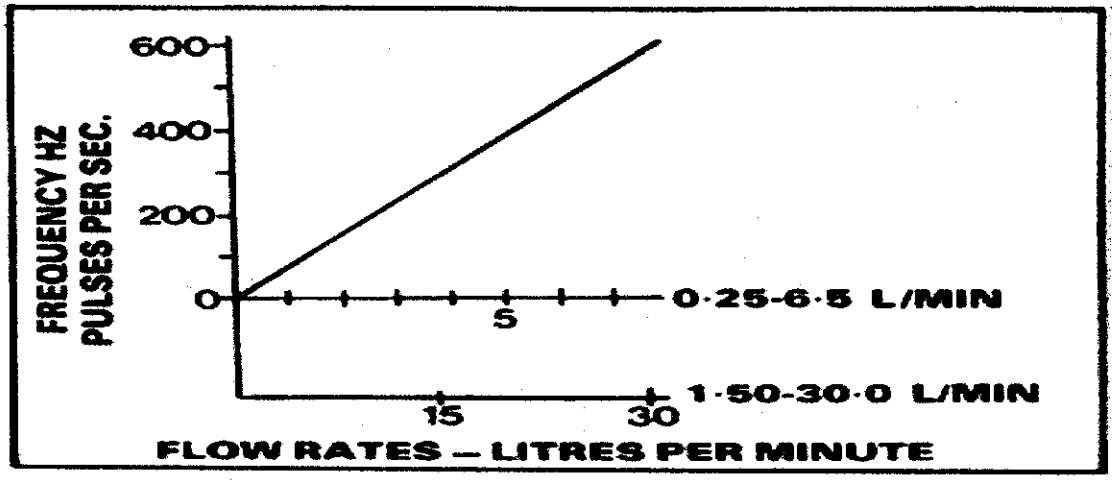


Figure 20 : Graph of frequency (Hz) output vs flow rates (litres per minute)

The chosen flow sensor model for this project is Beige (257-133). The flow rate ranges from 1.5 – 30 Liters/minute. The flow sensor above is chosen based on its measurement range, cost and specification.

Before installation, the free running of the flow sensor is checked by blowing through it. Ideally the flow sensor should be installed with the arrow on the turbine housing pointing vertically upwards. As the detection system is optical it is undesirable to install the flow

sensor near a strong light source or in direct sunlight, as this may 'swamp' the detection system [4].

3.5.4 Solenoid Valve

Solenoid valve is a device that uses a solenoid to control valve activation. Solenoid valve would receive electrical signal to open or close the valve from the microcontroller.

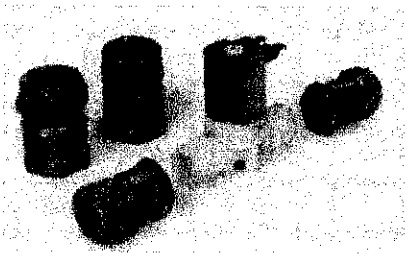


Figure 21: 2 Way Water Solenoid Valve

Technical specification	
Voltage	230V, 120V or 24V AC 24V or 12 V AC/DC
Frequency	50/60 Hz or DC
Power Draw	6VA
Coil Insulation	Class F-140° operating temperature
Pressure range	0.2 to 10 bar
Ambient Temperature	60°C maximum
Medium/Temperature	Water/ at 90°C maximum
Flow Regulation	0.25 to 17 liters per minute

Table 4: Technical specification for 2 Way Water Solenoid Valve

The solenoid valve used for the project is powered by 12V DC power supply. It has been checked and tested to verify that the equipment is working out properly. When powered, the valve should be open to let the water flow.

Figure 22: Testing Solenoid Valve

3.5.5 Pump

The pump provides a certain pressure range for the liquid transfer to flow in the metering system. Based on the size of the project, an aquarium pump 1200B Benz Aquarium Power Liquid Filter is considered to supply water to the tanks. Among its features are:

- Compact size
- Completely submersible motor
- Adjustable water flow

3.5.6 Interfacing circuit: Frequency to Voltage Converter

The interfacing circuit will convert the frequency output received from the sensor to voltage as an input for the microcontroller. The LM2907 specifically has been chosen to do the frequency to voltage conversion.

LM2907 series are monolithic frequency to voltage converters with a high gain op amp/comparator designed to operate a relay, lamp, or other load when the input frequency reaches or exceeds a selected rate.

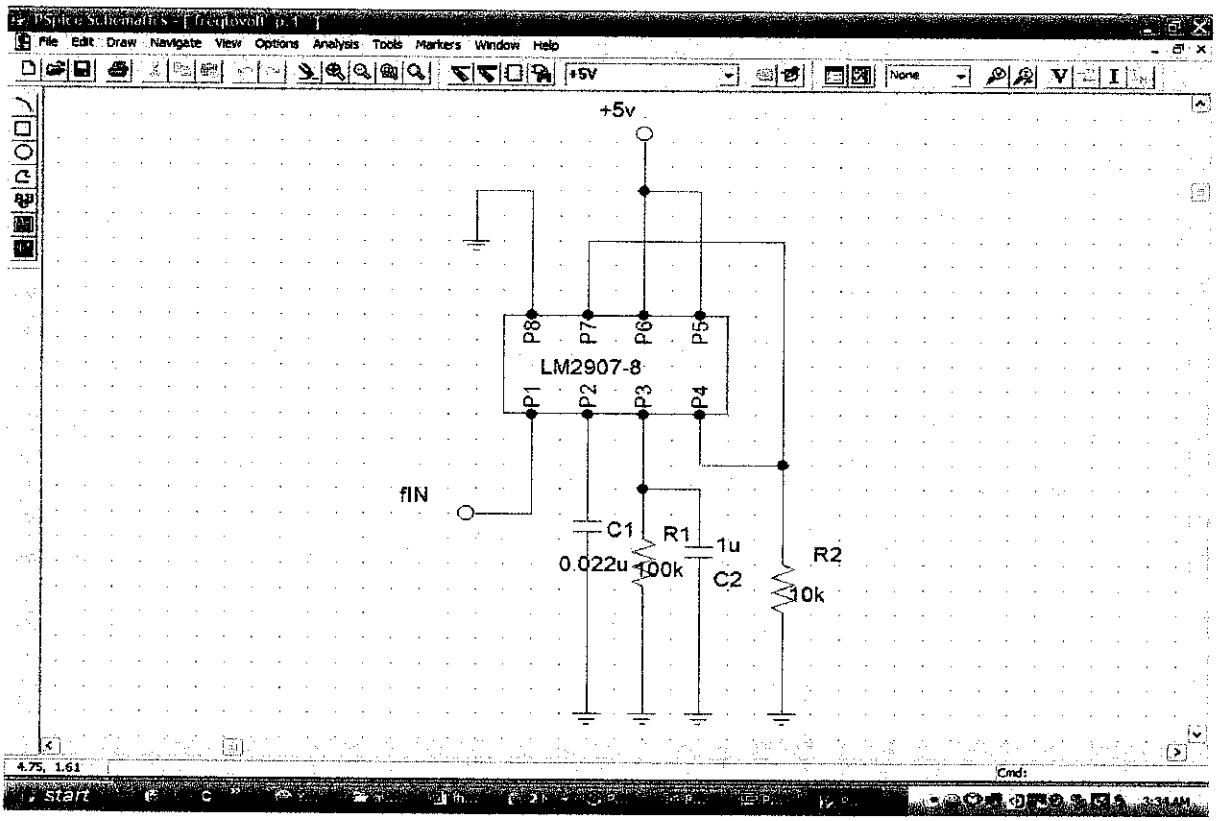


Figure 23 Frequency to Voltage Converter Circuit

3.5.7 LCD Display

The most common type of LCD controller that can be considered for the project is the Hitachi 44780 which provides a relatively simple interface between a processor and an LCD. The most common connector used for the 44780 based LCDs is 14 pins in a row. The pins are wired as describe in Table 5:

Pins	Description
1	Ground
2	Vcc
3	Contrast Voltage
4	"R/S" _Instruction/Register Select
5	"R/W" _Read/Write LCD Registers
6	"E" Clock
7 - 14	Data I/O Pins

Table 5: LCD pin description

3.5.8 Keypad

The keypad is used in this project to enable the users to enter the input data. A numeric keypad, 4x4 matrixes has been selected for this purpose. The layout and connection of keypad for 16 keys matrix are shown in the figure for the connection with female connector HE14.

The keypad used for this application is a 16-way XY-Matrix hexadecimal keypad Farnell. This keypad has eight connections at the rear—four columns (X1–X4) and four rows (Y1–Y4).

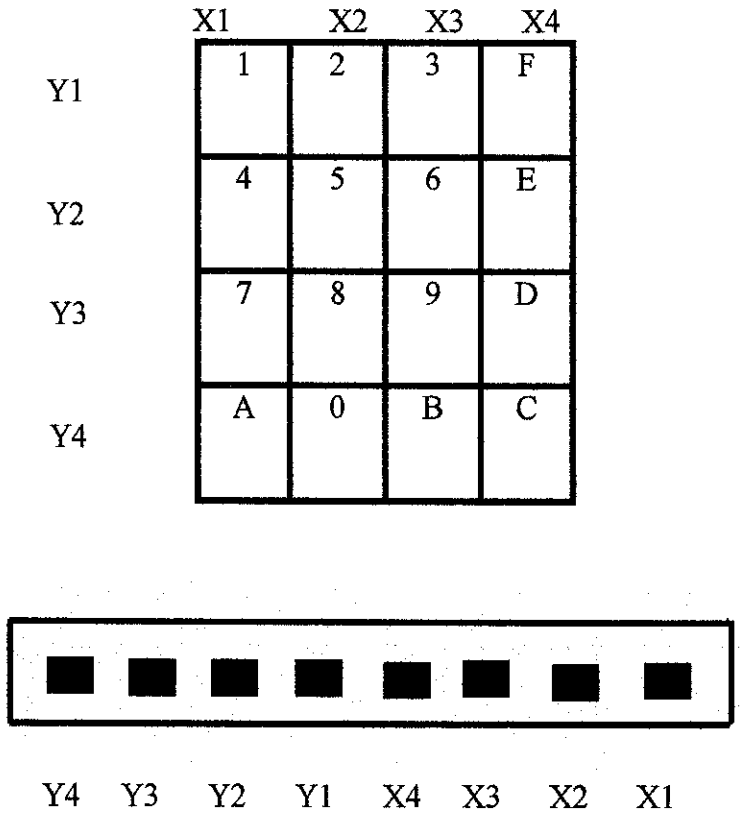


Figure 24 Connection between keypad and the female connector HE14

CHAPTER 4
RESULTS AND DISCUSSION

4.1 PIC16F877

4.1.1 Microcontroller Architecture

Generally, the performance of the PIC16F877 can be attributed to a number of architectural features commonly found in RISC microprocessors. The figure below shows the architecture of the PIC16F877 microcontroller.

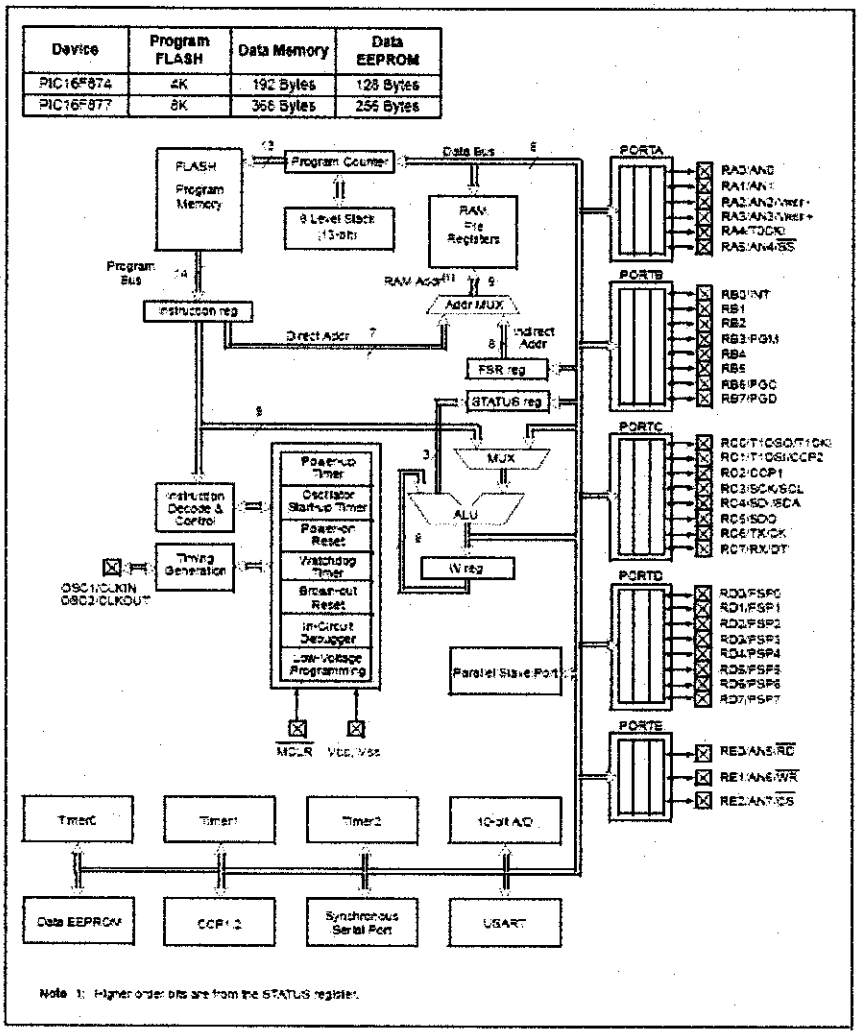


Figure 25: Block Diagram for PIC16F877

Harvard architecture has the program memory and data memory as separate memories are accessed from separate buses. This improves bandwidth over traditional von Neumann architecture in which program and data are fetched from the same memory using the same bus.

The Program Memory and Data Memory have separate buses so that concurrent access can occur. Separating program and data memory further allows instructions to be sized differently than the 8-bit wide data word.

The register file can be accessed either directly or indirectly through the File Select Register (FSR). All special function registers including the program counter are mapped in the data memory. An orthogonal (symmetrical) instruction set makes it possible to carry out any operation on any register using any addressing mode. This makes programming with the PIC16F877 simple yet efficient [1].

4.1.2 PIC16F877 Microcontroller Circuitry

The microcontroller circuitry must be well designed for optimum usage. The pins at PORT B are used for the inputs of LCD. Meanwhile, the pins at PORT D are used for the keypad. Port C for valve, and Port A for flow sensor and ADC. The circuit schematic is attached in the Appendices.

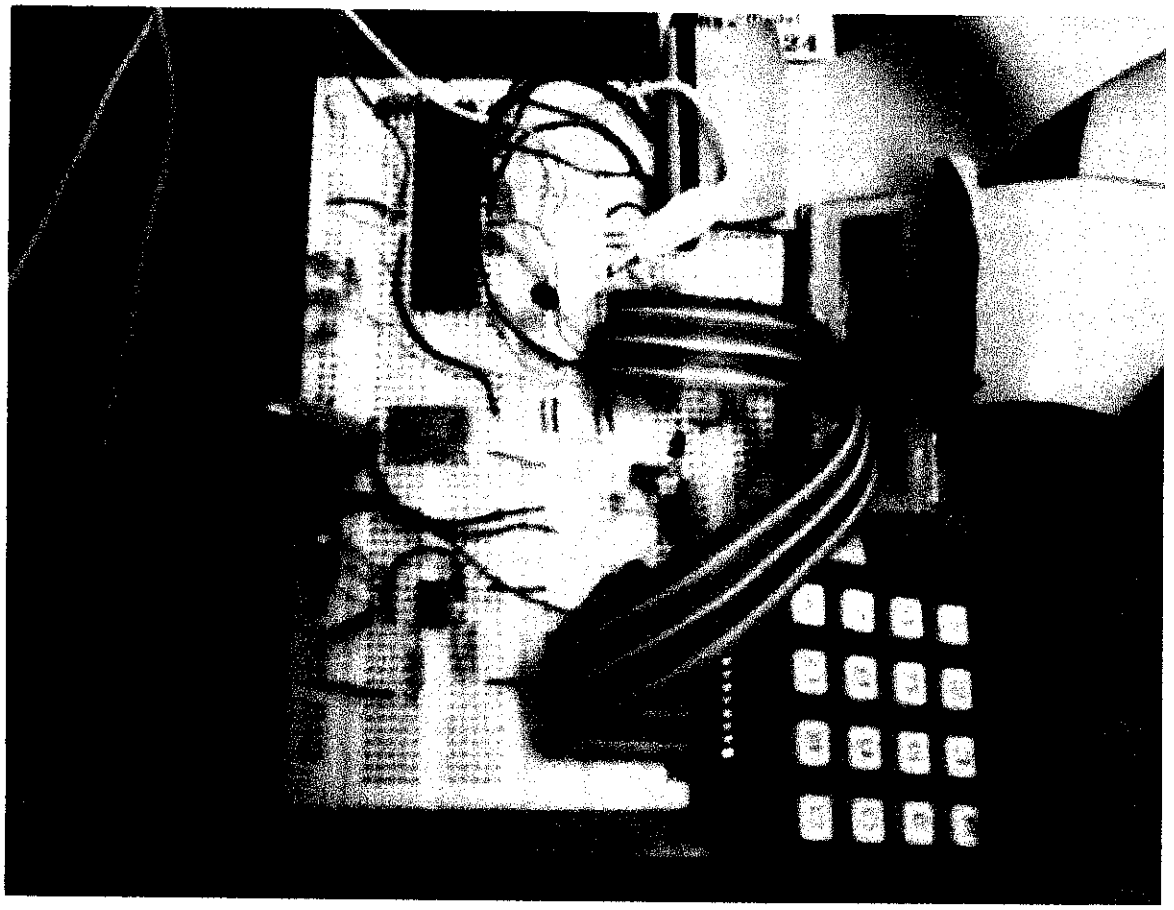


Figure 26 The microcontroller circuitry

4.1.3 PIC16F877 Microcontroller Programming

The C programming language is chosen to program the PIC16F877. The program will flow in such a way that the PIC16F877 will gather all the inputs into the PIC and produce an output display by the LCD. In order to run this program smoothly, several important initializations must be included. They are:

- PIC16F877 header file
- Declare the desired fuses
- Define the ADC bit used
- Define the delay or the clock oscillation speed
- Function math library

- Include other files used: LCD.c and key_pad.c

```
# include <16f877.h>
#device ADC=8
#include <stdio.h>
#include <math.h>

#USE DELAY(CLOCK=4000000) // Using a 4 MHz clock
#FUSES XT,NOWDT,NOPROTECT, NOPUT,
NOBROWNOUT,NOLVP

#include <LCD.C>
#include <key_pad.c>
#include <string.h>
```

Initially, a flowchart of the whole operation is designed to guide the coding and the flow of the program. This flowchart shapes the programming codes in the C programming.

Before downloading a program into a PIC, the program must be verified and compiled with the C Compiler. The program will be converted into a hex file. Then, the EPROM of the PIC must be cleared and blanked. Once this stage is successful, the program can be downloaded into the PIC. This was done using the WARP-13 PIC programmer.

The source code for the project is attached in the Appendices.

4.2 Frequency to voltage converter

An interfacing circuit is needed to convert the pulse per second (hertz) output from flow sensor to voltage as the PIC16F877 can not accept the pulse input. The output of this circuit will goes into one of the Analog to Digital Converter (ADC) pin of the PIC16F877.

Based on the theory application, the LM2907-8 will convert the frequency input to voltage based on the equation:

$$Vo = fIN \times Vcc \times R1 \times C1$$

For the design, the value of R1 = 100k and C1 = 0.022uF has been selected. The calculated and measured voltage of the circuit is shown in the table below. The Vcc applied is 5V. The design is based on the range of ADC voltage can accept which is 0 to 5.5V.

fIN (Hertz)	Vout calculated (voltage) = fINxVccxR1xC1	Vout measured (voltage)
66	0.726	0.667
132	1.452	1.32
198	2.178	1.99
264	2.904	2.64
330	3.63	3.29

Table 6: Voltage measurement and voltage calculated for the input frequency

Based on these data, the graph of measured and calculated output voltage versus frequency can be plotted and the formula for conversion from frequency to voltage can be obtained which is :

$$y = 0.0099x + 0.0116$$

y= voltage

x= frequency / pulse per second

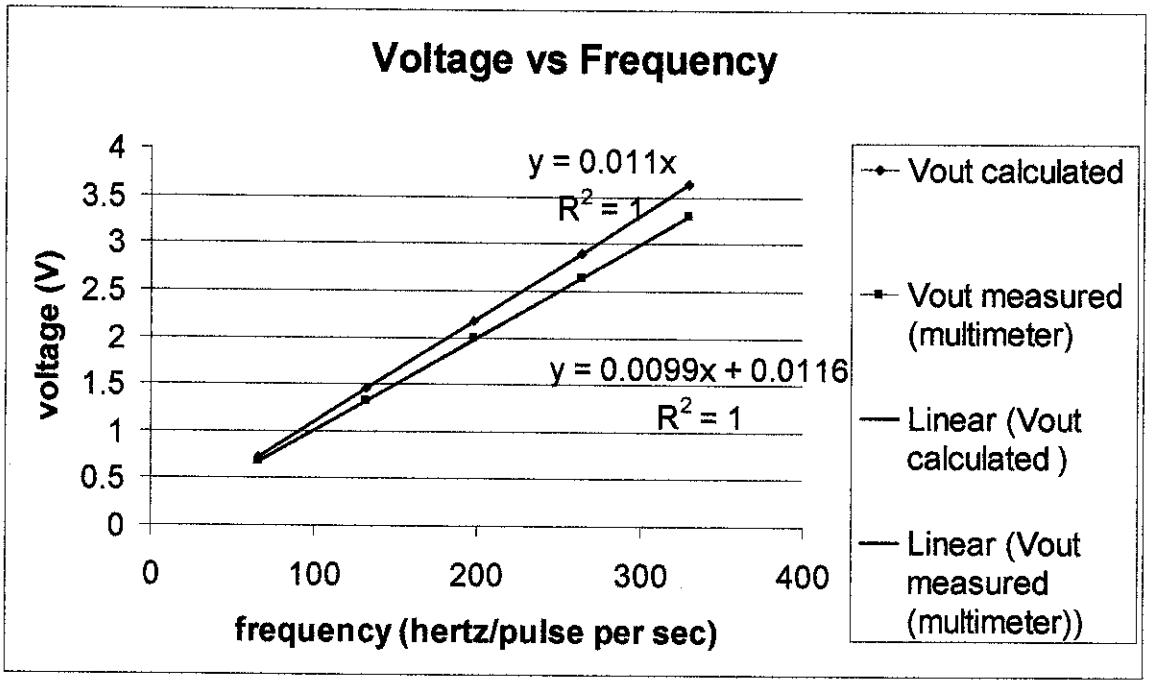


Figure 27 Graph for measured and calculated output voltage versus frequency

Referring to the figure 3, the calibration of the flow sensor will give value

Frequency =20 x flow rate

This conversion formula will be included in the source code to enable the microcontroller to calculate the flow rate data.

4.3 LCD display

Though the CCS compiler doesn't detect any error for the program, the PIC is still prone for errors. The figures show the status for the LCD display which asks the user to enter the volume and the display of the flow rate. Initially, the liquid volume range has been set for 5 to 9 liters only.

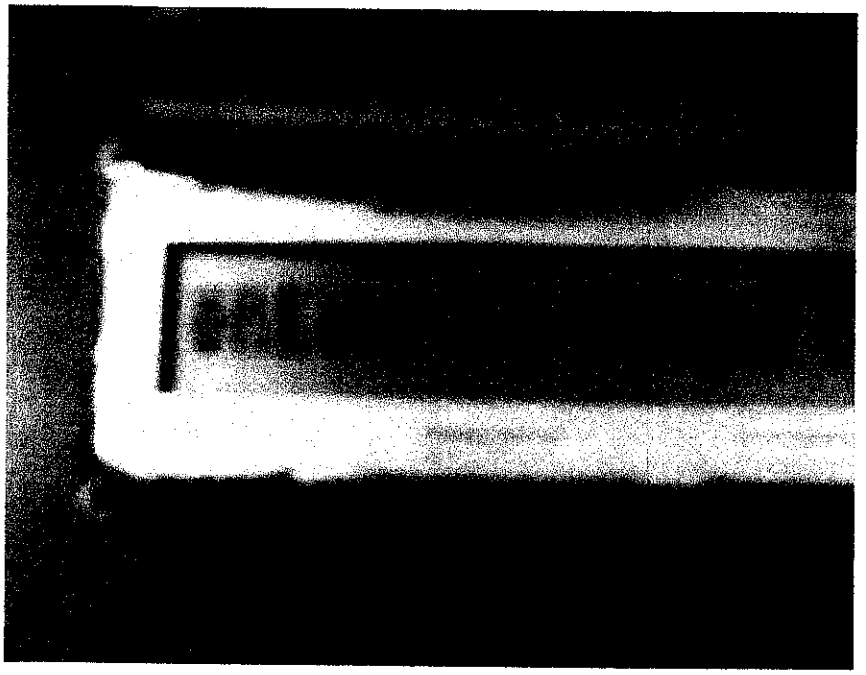


Figure 28 LCD displays "Enter volume"

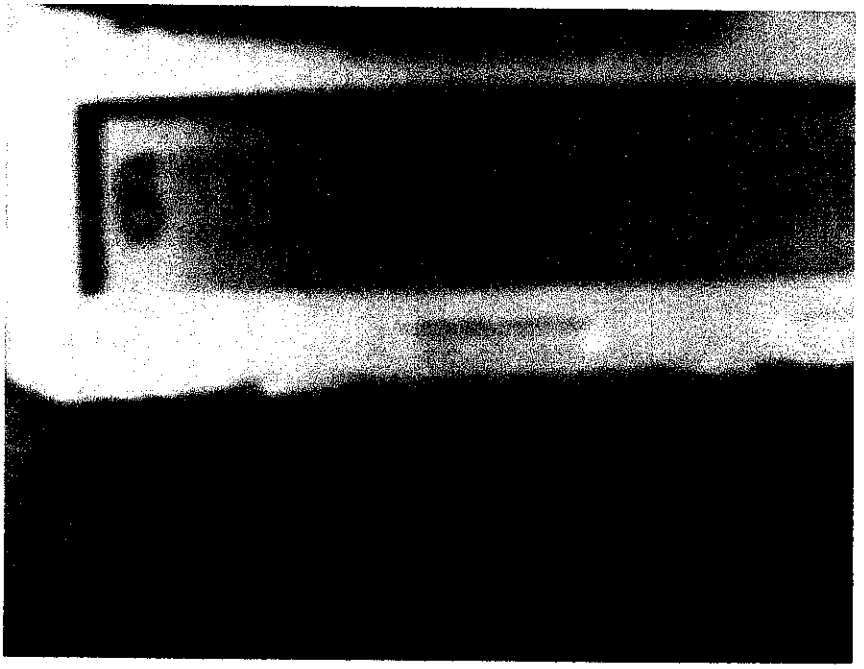


Figure 29 User enter volume: 6 liters

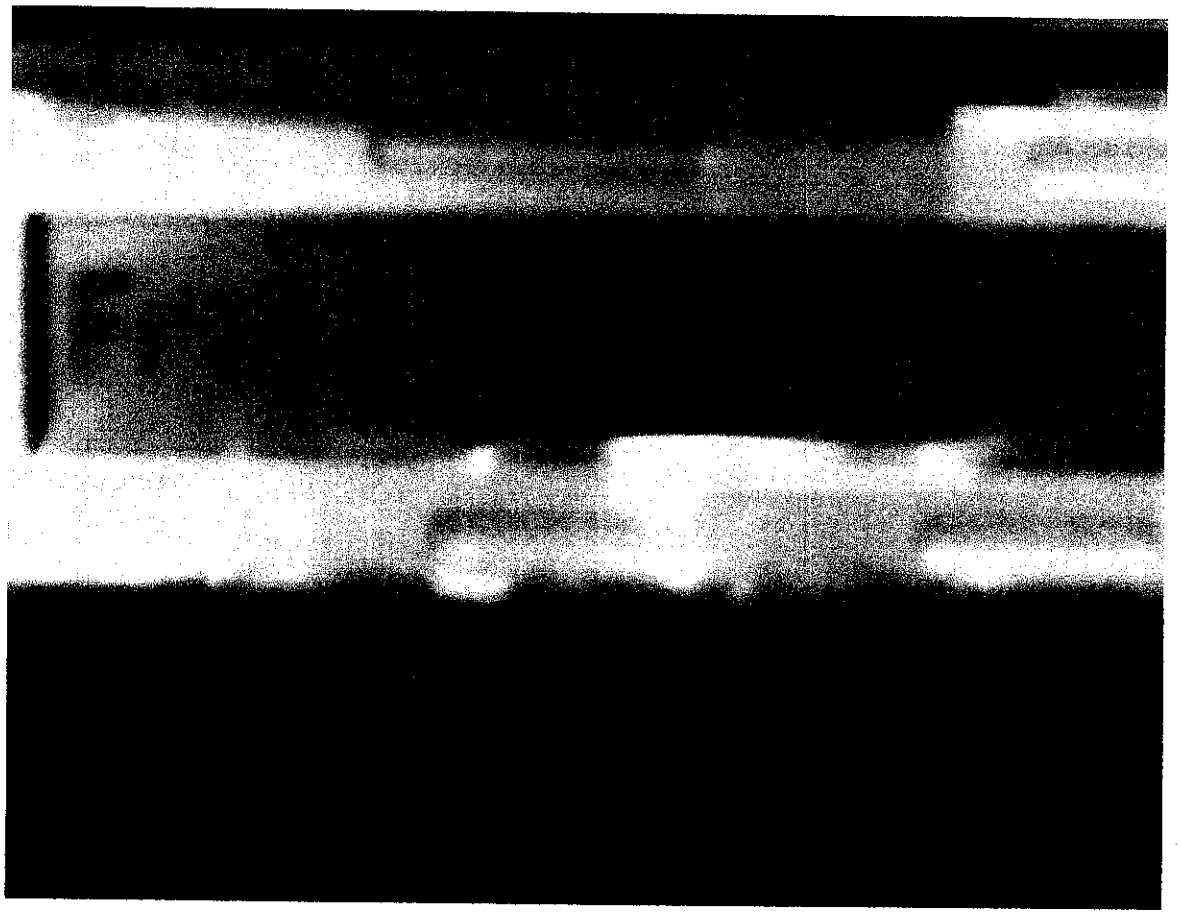


Figure 30 The flow rate display

4.4 Prototype Built

In conjunction with the hardware circuitry and software designed, a prototype is built to better illustrate the capability of the system. In this prototype, the flow sensor and the solenoid valve is connected to water hose and pipe PVC. This prototype enables the demonstration of the flow metering system.

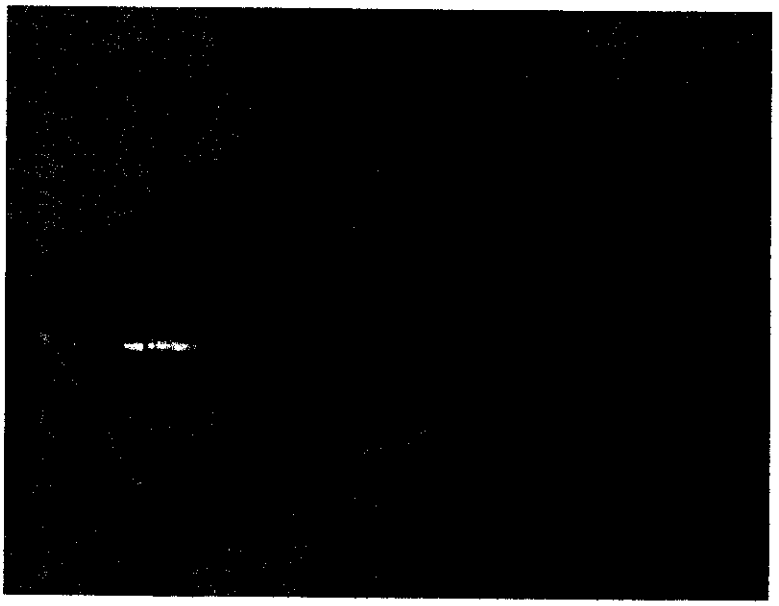


Figure 31 Solenoid valve

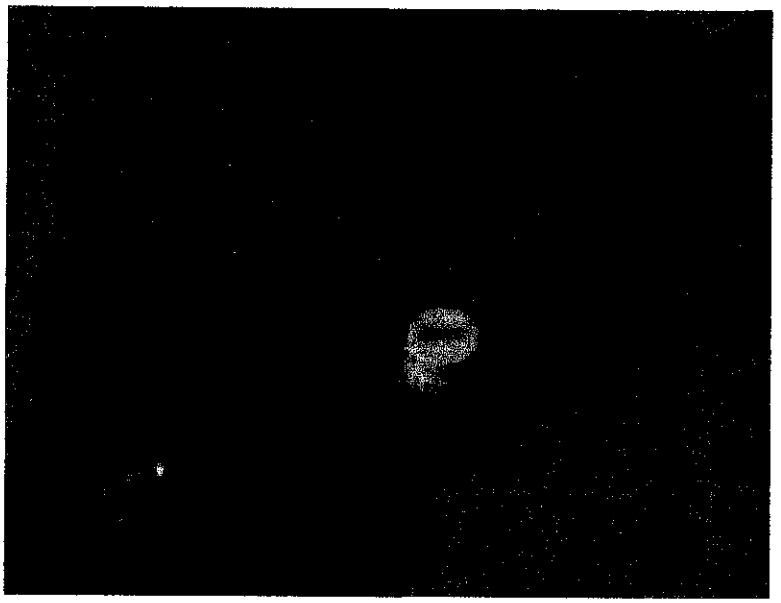


Figure 32 Flow sensor

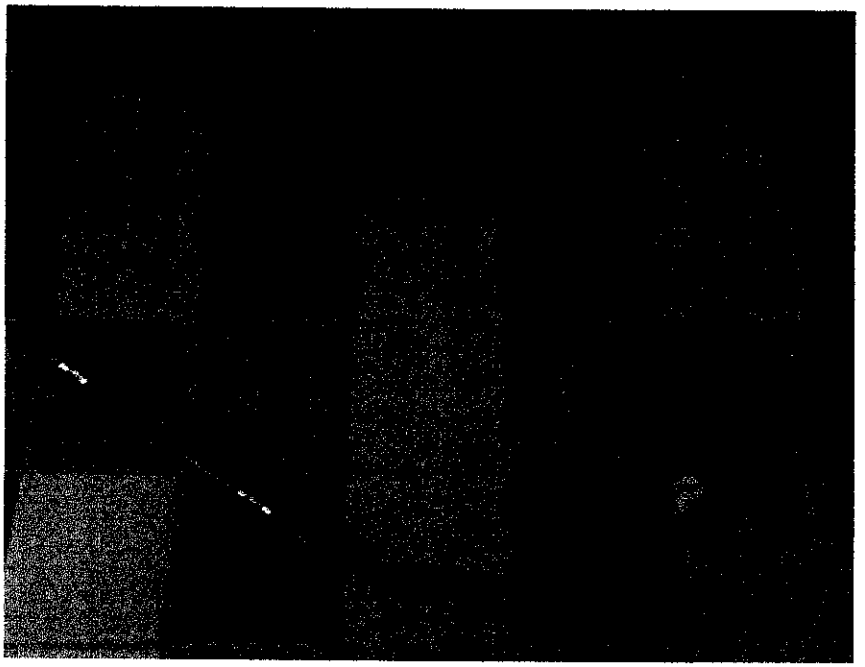


Figure 33 The prototype

4.5 Constraints and Problems Encountered

Naturally, it is almost inevitable to avoid problems and complications when it comes to developing a system from scratch. It can be said that the time spent on designing is almost equal to the time spent in the troubleshooting phases. The main area affected by this consequence was the integration of all the various components to establish the system.

The PIC programming was done using the C programming as it is known to be less complicated than the assembly. Furthermore, the WARP-13 programmer available can be quite tricky. The process of burning the program into the PIC may take many attempts as the programmer is prone to errors.

The circuitries also have to go through a series of troubleshooting. Upon checking every connections and components, some components were found to be faulty. Besides that, the LCD and the PIC needs 5V voltage supply regulated in order to make it function.

CHAPTER 5
CONCLUSION

5.1 Conclusion

Basically, the challenge in this project is to design the microcontroller PIC16F877 to monitor the metering system and give the output data to the display. The equipment chosen had to be reliable, rugged, accurate and easy to install and operate. The challenges for the application included the ability to diagnose any problems immediately plus the ability to deliver flow data to the controller.

Though this system is small scale at this stage, with some improvisations, it can be adapted and implemented in various domestic and industrial operations.

6.0 REFERENCES

[1] http://ww1.microchip.com/downloads/en/DeviceDoc/30292c.pdf

[2] http://ww1.microchip.com/downloads/en/DeviceDoc/39582b.pdf

[3] http://www.jhu.edu/nthakor/teaching/bmi1/lab5_2005.doc

[4] http://www.rsmalaysia.com/cgi-bin/bv/browse/Module.jsp

[5] http://www.rsmalaysia.com/cgi-bin/bv/browse/Module.jsp

[6] www.ecgf.uakron.edu/grover/web/ee470/labs/lab04.pdf

[7] http://www.myke.com/lcd.htm

[8]] http://www.analog.com/UploadedFiles/Application_Notes/
770996085AN_660_0.pdf

7.0 APPENDICES

7.1 Gantt Chart

No	Details	Week																			
	FIRST SEMESTER	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	Selection of Project Topic																				
2	Preliminary Research																				
3	Submission of Preliminary Report				•																
4	Project Work																				
5	Submission of Progress Report									•											
6	Project Work continue																				
7	Submission Draft Report												•								
8	Submission Interim Report															•					
9	Oral Presentation																•				

No	Details	Week																			
	SECOND SEMESTER	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
10	Project Work Continue																				
11	Submission of Progress Report 1			•																	
12	Project Work Continue																				
13	Submission of Progress Report 2								•												
14	Project Work Continue																				
15	Draft Report												•								
16	Final Report/Dissertation														•						
17	Technical Report															•					
18	Oral Presentation																		•		
19	Final Report(Hard Cover)																				•

Legend

Process

Project Milestone •

7.2 Coding

7.2.1 lcd.c

```
////////////////////////////////////////////////////////////////////////
////                        LCDD.C                                  ////
////              Driver for common LCD modules                     ////
////                                                                ////
//// lcd_init()  Must be called before any other function.         ////
////                                                                ////
//// lcd_putc(c)  Will display c on the next position of the LCD.  ////
////                The following have special meaning:            ////
////                 \f  Clear display                             ////
////                 \n  Go to start of second line                ////
////                 \b  Move back one position                    ////
////                                                                ////
//// lcd_gotoxy(x,y) Set write position on LCD (upper left is 1,1)  ////
////                                                                ////
//// lcd_getc(x,y)  Returns character at position x,y on LCD        ////
////                                                                ////
////////////////////////////////////////////////////////////////////////
////      (C) Copyright 1996,2003 Custom Computer Services          ////
//// This source code may only be used by licensed users of the CCS C ////
//// compiler.  This source code may only be distributed to other   ////
//// licensed users of the CCS C compiler.  No other use, reproduction ////
//// or distribution is permitted without written permission.       ////
//// Derivative programs created using this software in object code ////
//// form are not restricted in any way.                            ////
////////////////////////////////////////////////////////////////////////

// As defined in the following structure the pin connection is as follows:
//   RB0   Chip Enable (CE)
//   RB1   Register Select (RS)
//   RB2   Read/Write* (R/W*)
//   RB4   Data Bit 4 (DB4)
//   RB5   Data Bit 5 (DB5)
//   RB6   Data Bit 6 (DB6)
//   RB7   Data Bit 7 (DB7)
//
//  LCD pins DB0-DB3 are not used and PIC's RB3 is not used.

// Un-comment the following define to use port B
#define use_portb_lcd TRUE
```

```
struct lcd_pin_map {              // This structure is overlayed
        BOOLEAN enable;             // on to an I/O port to gain
        BOOLEAN rs;               // access to the LCD pins.
        BOOLEAN rw;               // The bits are allocated from
        BOOLEAN unused;             // low order up.  ENABLE will
        int    data : 4;        // be pin B0.
      } lcd;


#if defined(__PCH__)
#if defined use_portb_lcd
  #byte lcd = 0xF81               // This puts the entire structure
#else
  #byte lcd = 0xF83               // This puts the entire structure
#endif
#else
#if defined use_portb_lcd
  #byte lcd = 6                 // on to port B (at address 6)
#else
  #byte lcd = 8                 // on to port D (at address 8)
#endif
#endif

#if defined use_portb_lcd
  #define set_tris_lcd(x) set_tris_b(x)
#else
  #define set_tris_lcd(x) set_tris_d(x)
#endif


#define lcd_type 2          // 0=5x7, 1=5x10, 2=2 lines
#define lcd_line_two 0x40    // LCD RAM address for the second line


BYTE const LCD_INIT_STRING[4] = {0x20 | (lcd_type << 2), 0xc, 1, 6};
                    // These bytes need to be sent to the LCD
                    // to start it up.


                    // The following are used for setting
                    // the I/O port direction register.

struct lcd_pin_map const LCD_WRITE = {0,0,0,0,0}; // For write mode all pins are out
struct lcd_pin_map const LCD_READ = {0,0,0,0,15}; // For read mode data pins are in
```

```
BYTE lcd_read_byte() {
    BYTE low,high;
    set_tris_lcd(LCD_READ);
    lcd.rw = 1;
    delay_cycles(1);
    lcd.enable = 1;
    delay_cycles(1);
    high = lcd.data;
    lcd.enable = 0;
    delay_cycles(1);
    lcd.enable = 1;
    delay_us(1);
    low = lcd.data;
    lcd.enable = 0;
    set_tris_lcd(LCD_WRITE);
    return( (high<<4) | low);
}


void lcd_send_nibble( BYTE n ) {
    lcd.data = n;
    delay_cycles(1);
    lcd.enable = 1;
    delay_us(2);
    lcd.enable = 0;
}


void lcd_send_byte( BYTE address, BYTE n ) {

    lcd.rs = 0;
    while ( bit_test(lcd_read_byte(),7) ) ;
    lcd.rs = address;
    delay_cycles(1);
    lcd.rw = 0;
    delay_cycles(1);
    lcd.enable = 0;
    lcd_send_nibble(n >> 4);
    lcd_send_nibble(n & 0xf);
}


void lcd_init() {
  BYTE i;
  set_tris_lcd(LCD_WRITE);
```

```
   lcd.rs = 0;
   lcd.rw = 0;
   lcd.enable = 0;
   delay_ms(15);
   for(i=1;i<=3;++i) {
     lcd_send_nibble(3);
     delay_ms(5);
   }
   lcd_send_nibble(2);
   for(i=0;i<=3;++i)
     lcd_send_byte(0,LCD_INIT_STRING[i]);
}


void lcd_gotoxy( BYTE x, BYTE y) {
  BYTE address;

  if(y!=1)
    address=lcd_line_two;
  else
    address=0;
  address+=x-1;
  lcd_send_byte(0,0x80|address);
}

void lcd_putc( char c) {
  switch (c) {
    case '\f'   : lcd_send_byte(0,1);
                  delay_ms(2);
                                  break;
    case '\n'   : lcd_gotoxy(1,2);       break;
    case '\b'   : lcd_send_byte(0,0x10);  break;
    default     : lcd_send_byte(1,c);     break;
  }
}

char lcd_getc( BYTE x, BYTE y) {
  char value;

   lcd_gotoxy(x,y);
   while ( bit_test(lcd_read_byte(),7) ); // wait until busy flag is low
   lcd.rs=1;
   value = lcd_read_byte();
   lcd.rs=0;
   return(value);
}
```

7.2.2 key_pad.c

```
//#byte port_d = 0x08

char get_key(void)
{
  char t;

        while (1) {


    output_d(input_d() | 0xFF);         /* set RD4 to low to scan the first row */

                output_bit(PIN_D4,0);


                if (input(PIN_D0) ==0){
                        delay_us(10);

      while(input(PIN_D0) ==0)
      {
      }

                                return 'A';       /* return the ASCII code of A */
                }


                if (input(PIN_D1) ==0) {
                        delay_ms(10);
      while(input(PIN_D1) ==0)
      {
      }

                                return '7';       /* return the ASCII code of 7 */
                }


                if (input(PIN_D2) ==0) {
                        delay_ms(10);
      while(input(PIN_D2) ==0)
      {
      }

                                return '4';       /* return the ASCII code of 4 */
```

```
        }


        if (input(PIN_D3) ==0) {
                delay_ms(10);
  while(input(PIN_D3) ==0)
  {
  }

                        return '1';     /* return the ASCII code of 1 */
        }


output_d(input_d() | 0xFF);     /* set RD5 to low to scan the second row */
        output_bit(PIN_D5,0);
        if (input(PIN_D0) ==0) {
                delay_ms(10);
  while(input(PIN_D0) ==0)
  {
  }

                        return '0';     /* return the ASCII code of 0 */
        }


        if (input(PIN_D1) ==0) {
                delay_ms(10);
  while(input(PIN_D1) ==0)
  {
  }

                        return '8';     /* return the ASCII code of 8 */
        }


        if (input(PIN_D2) ==0) {
                delay_ms(10);
  while(input(PIN_D2) ==0)
  {
  }

                        return '5';     /* return the ASCII code of 5 */
        }
```

```
            if (input(PIN_D3) ==0) {
                    delay_ms(10);
  while(input(PIN_D3) ==0)
  {
  }

                            return '2';       /* return the ASCII code of 2 */
            }


output_d(input_d() | 0xFF);         /* set RD6 to low to scan the third row */
            output_bit(PIN_D6,0);
            if (input(PIN_D0) ==0) {
                    delay_ms(10);
  while(input(PIN_D0) ==0)
  {
  }

                            return 'B';       /* return the ASCII code of B */
            }


            if (input(PIN_D1) ==0) {
                    delay_ms(10);
  while(input(PIN_D1) ==0)
  {
  }

                            return '9';       /* return the ASCII code of 9 */
            }


            if (input(PIN_D2) ==0) {
                    delay_ms(10);
  while(input(PIN_D2) ==0)
  {
  }

                            return '6';       /* return the ASCII code of 6 */
            }


            if (input(PIN_D3) ==0) {
                    delay_ms(10);
  while(input(PIN_D3) ==0)
```

```
{
}

                    return '3';     /* return the ASCII code of 3 */
        }


output_d(input_d() | 0xFF);     /* set RD7 to low to scan the fourth row */
        output_bit(PIN_D7,0);
        if (input(PIN_D0) ==0) {
                delay_ms(10);
 while(input(PIN_D0) ==0)
 {
 }

                    return 'C';     /* return the ASCII code of C */
        }


        if (input(PIN_D1) ==0) {
                delay_ms(10);
 while(input(PIN_D1) ==0)
 {
 }

                    return 'D';     /* return the ASCII code of D */
        }


        if (input(PIN_D2) ==0) {
                delay_ms(10);
 while(input(PIN_D2) ==0)
 {
 }

                    return 'E';     /* return the ASCII code of E */
        }


        if (input(PIN_D3) ==0) {
                delay_ms(10);
 while(input(PIN_D3) ==0)
 {
 }
```

```
                    return 'F';        /* return the ASCII code of F */
                }
        }
}
```

7.2.3 Display and enter volume

```
#include <16F877.h>
#fuses XT,NOWDT,NOPROTECT,NOLVP,NOPUT,NOBROWNOUT
#use delay(clock = 4000000)
#include <LCD.C>
#include <key_pad.c>
//#include <string.h>


main()
{
    char k;
    lcd_init();



     while(1)
     {

            lcd_gotoxy(1,1);
            lcd_putc('\f');
            lcd_putc("enter ");

            delay_ms(100);
            lcd_putc('\n');
            lcd_putc("volume:");

            delay_ms(1000);
            lcd_putc('\f');


    k = get_key();

    if( k=='5' || k=='6' || k=='7' || k=='8' || k=='9')
    {

      lcd_putc(k);
     }
     else
```

```
    {
    lcd_putc('\f');
    }
delay_ms(1000);
    }
}
```

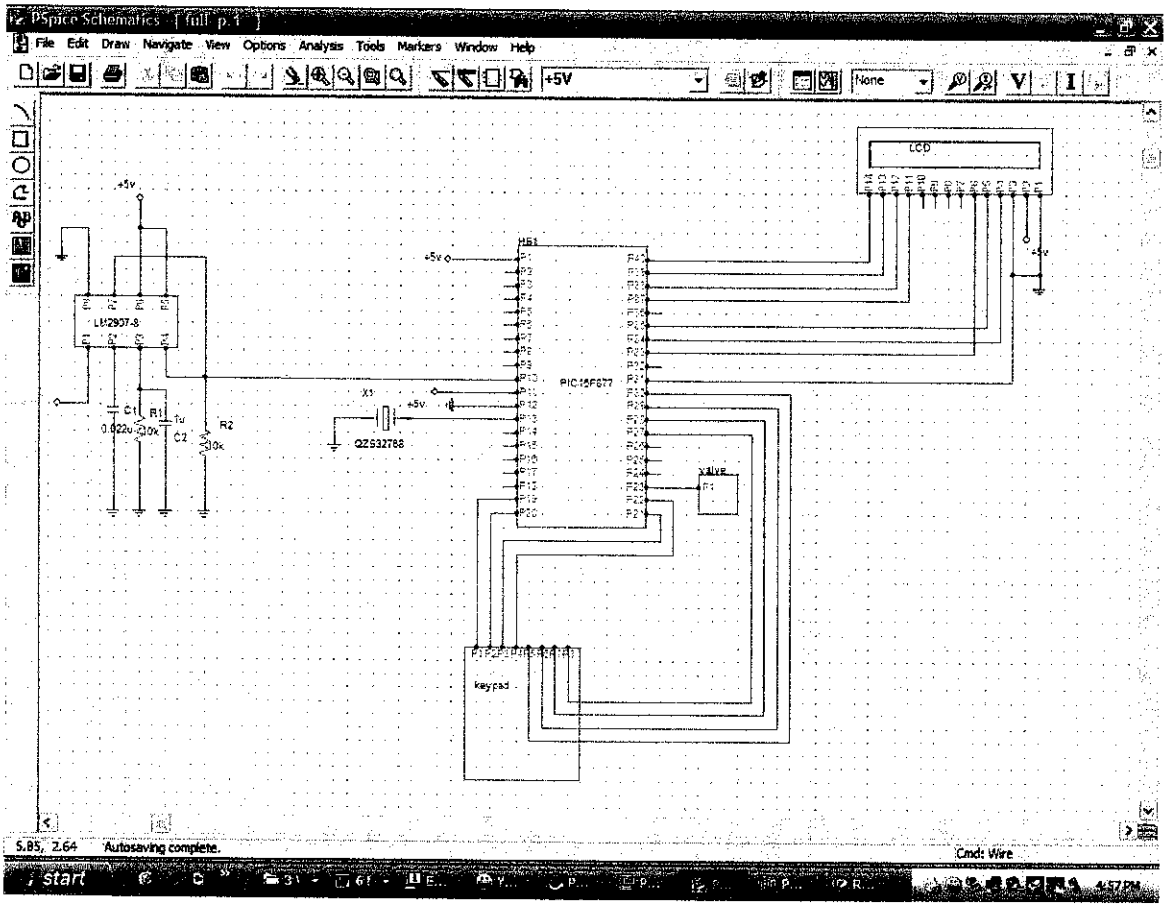
7.2.4 Flow rate display

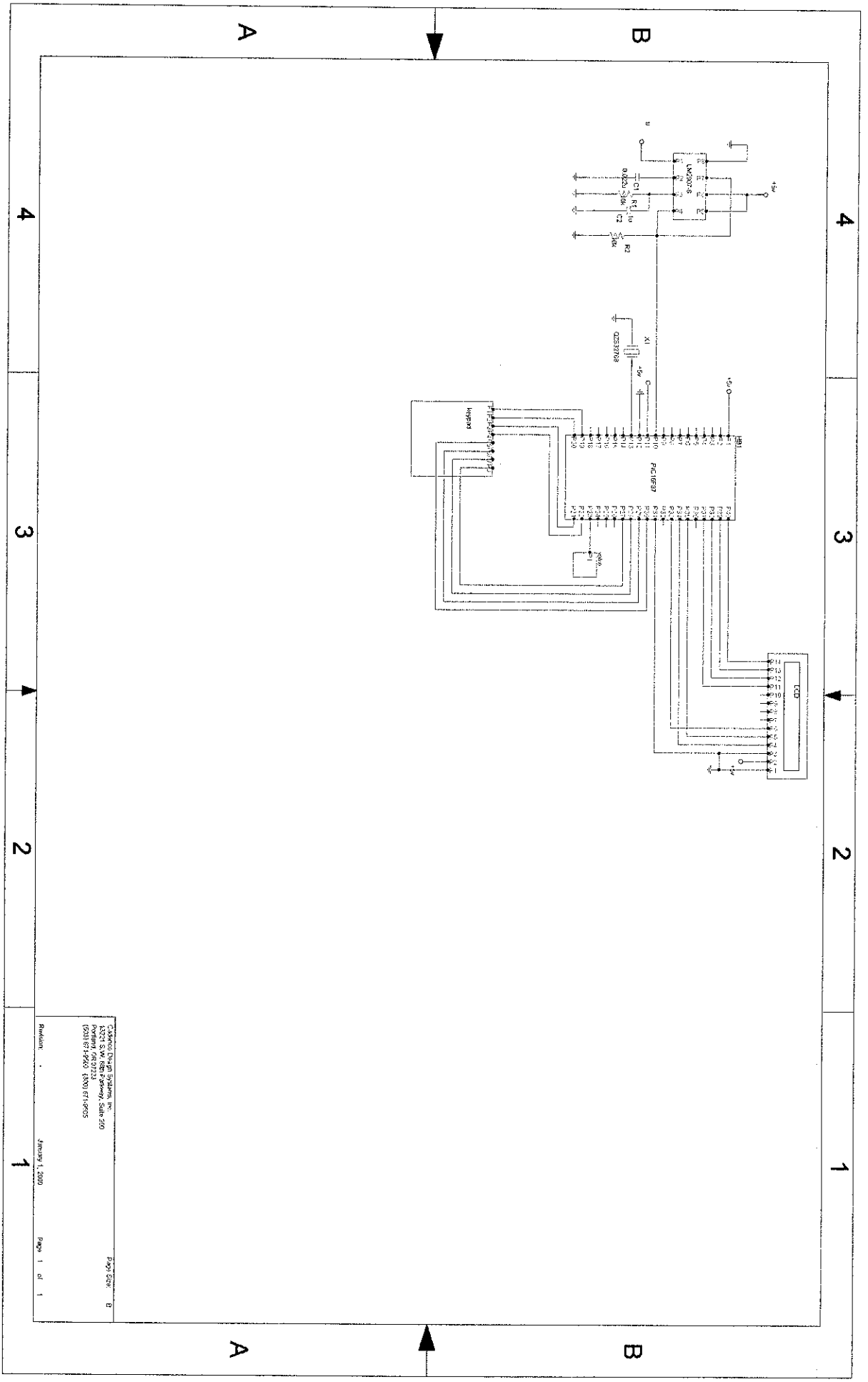
```
#include <16f877.h>
#device ADC=8
#include <stdio.h>
#include <math.h>

#USE DELAY(CLOCK=4000000) // Using a 4 MHz clock
#FUSES XT,NOWDT,NOPROTECT, NOPUT, NOBROWNOUT,NOLVP

#include <LCD.C>
#include <key_pad.c>
#include <string.h>

//declare variable

float value;

float voltage=(92.72 * 5.5) / 255;
float freq =(2- 0.0116) / (0.0099);
float frate = 20* 201.9;

// voltage = 5.5 * value /255;
void adc(int value)
{
int value1,value2,x;
if (value!=0)
{
value1=value;
if(value1!=0)
{
for (x=3;x>=1;x--)
{
lcd_gotoxy(x,2);
value2=value1%10;
```

```
value2=value2+48;
lcd_putc(value2);
value1=value1/10;
}
}
}

else
{
lcd_gotoxy(1,1);
lcd_putc("0");
}
}



main()
{

//declare variable

int adcValue;

float voltage;
int frate;

//led
output_high(pin_A0);

//adc
setup_adc_ports(ALL_ANALOG);
setup_adc(ADC_CLOCK_INTERNAL);                    //use internal clock
set_adc_channel(7);



lcd_init();



while(TRUE)
{

delay_ms(500);
value=read_adc();
```

```
//freq=(value-0.0116)/0.0099 ;
//frate= 20*freq ;       //calculate flowrate

lcd_gotoxy(1,1);
lcd_putc("Frate %f \n\r, frate");
adc(frate);

delay_ms(1000);
}

}
```

7.3 Microcontroller Circuitry




A
B
4
3
2
1
LM2907-8
PIC16F627
LCD
Revision:
January 1, 2000
Page: 1 of 1
Page Size: B

7.4 Flow Sensor Datasheet

Issued / Herausgegeben / Publicado / Pubblicato 05/2000 **V8189**

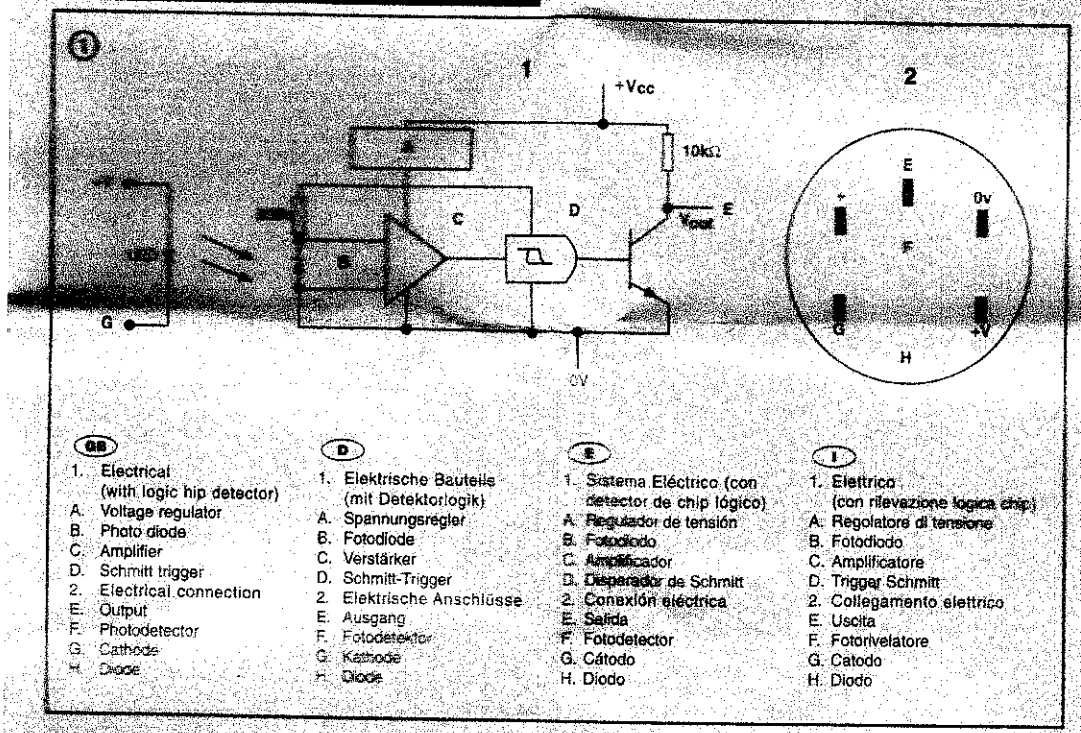
Flow sensors - 15mm Dia. pipe (GB)

Durchflußsensoren - 15mm Rohrleitungsdurchmesser (D)

Sensores de Caudal Tubería de 15mm de diám. (E)

Sensori di flusso- tubo da 15mm dia. (I)

Figures / Abbildung / Figure / Figura



(GB)
1. Electrical (with logic hip detector)
A. Voltage regulator
B. Photo diode
C. Amplifier
D. Schmitt trigger
2. Electrical connection
E. Output
F. Photodetector
G. Cathode
H. Diode

(D)
1. Elektrische Bauteile (mit Detektorlogik)
A. Spannungsregler
B. Fotodiode
C. Verstärker
D. Schmitt-Trigger
2. Elektrische Anschlüsse
E. Ausgang
F. Fotodetektor
G. Kathode
H. Diode

(E)
1. Sistema Eléctrico (con detector de chip lógico)
A. Regulador de tensión
B. Fotodiodo
C. Amplificador
D. Disparador de Schmitt
2. Conexión eléctrica
E. Salida
F. Fotodetector
G. Cátodo
H. Diodo

(I)
1. Elettrico (con rilevazione logica chip)
A. Regolatore di tensione
B. Fotodiodo
C. Amplificatore
D. Trigger Schmitt
2. Collegamento elettrico
E. Uscita
F. Fotorivelatore
G. Catodo
H. Diodo