LPG Leakage Detection

And Monitoring

Using GSM

Submitted in partial fulfillment of the Degree of

Bachelor of Technology in Electronics & Communication

May – 2014

Under the Supervision of

Mr. Vikas Hastir

By

Pratyush Chauhan (101075)

Sarthak Sharma (101083)

Naveen Thakur (101087)

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY

WAKNAGHAT

CERTIFICATE

This is to certify that the project report entitled "**LPG Leakage Detection And Monitoring"** submitted by Pratyush Chauhan, Sarthak Sharma and Naveen Thakur in partial fulfillment for the award of degree of Bachelor of Technology in Electronics and Communication Engineering to Jaypee University of Information Technology, Waknaghat, Solan has been carried out under my supervision.

This work has not been submitted partially or fully to any other University or Institute for the award of this or any other degree or diploma.

Date:

Mr. Vikas Hastir
Asst. Professor
Department Of ECE
JUIT,Waknaghat

ACKNOWLEDGEMENT

At this level of understanding it is often difficult to understand the wide spectrum of knowledge without proper guidance and advice. Hence, we take this opportunity to express our heartfelt gratitude to our project guide **Mr. Vikas Hastir** who had faith in us and allowed us to work on this project. We owe him a great debt of gratitude for without his support, this work wouldn't have been accomplished indeed. We just have no words to express our obligation for this learned and noble scholar.

We also acknowledge our profound sense of gratitude to all the teachers who have been instrumental for providing us the technical knowledge and moral support to complete the project with full understanding.

We thank our **friends and family** for their moral support to carve out this project and above all **GOD** for removing all hurdles in the way.

Date:

Pratyush Chauhan (101075)

Sarthak Sharma (101083)

Naveen Thakur (101087)

TABLE OF CONTENTS

ABSTRACT

The aim of this project is to monitor for liquefied petroleum gas (LPG) leakage to avoid fire accidents providing house safety feature where security has been an important issue. The system detects the leakage of the LPG using gas sensor and alerts the consumer about the gas leakage by sending SMS. The proposed system uses the GSM to alert the person about the gas leakage via SMS. When the system detects the LPG concentration in the air exceeds the certain level then it immediately alert the consumer by sending SMS to specified mobile phone and alert the people at home by activating the alarm which includes the Buzzer and displays the message on LCD display regarding leakage. In the mean time the Stepper Motor rotates and closes the knob of the gas supply system to avoid further leakage.

…….
Pratyush Chauhan

…….
Sarthak Sharma

…….
Naveen Thakur

…….
Mr. Vikas Hastir

Date:

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 Background

Microcontrollers are used in industrial world to control many types of equipment, ranging from consumer to specialized devices. They have replaced the older types of microcontrollers, including microprocessor. The demand will grow as more equipment uses more intelligence. Applications range from controlling engines in modern automobiles to controlling laser printer and other computer peripherals. Technology has evolved to the point where this same washing machine could be connected to internet. We can envision a future with wearable computing where wrist watch-type devices could communicate with and control the washing machine using wireless networking. In this particular project, AT89s8253 will be used as controller. The microcontroller offers various functions that are suitable to design our project. It is very flexible and it makes designer's work become easier.

The AT89s8253 is a low power, high performance CMOS 8-bit microcomputer with 8K bytes of flash programmable and erasable read only memory(PEROM).The device is manufactured using Atmel's high density non volatile memory technology and is compatible with the industry standard 80c51 and 80C52 instruction set and pin out. The on-chip flash allows the program memory to be reprogrammed in system or by a conventional non volatile memory programmer. By combining a versatile 8-bit CPU with flash on a monolithic chip, the Atmel AT89s8253 is a powerful microcomputer which provides a highly flexible and cost effective solution to many embedded control applications. The main advantages of AT89s8253 over 8051 are:

- Software Compatibility
- Program Compatibility

1.2 Objective of the project

LPG leakage detection system is self-operated control system that will sense the LPG gas at an optimum level. This detected gas will change the resistivity levels of the MQ6. MQ6 will act as an electronic nose for the whole circuit. As the change in resistivity levels will be sensed by the circuit, microcontroller will be informed about the detection. Immediately microcontroller will send instruction to the buzzer, stepper motor, LCD display and GSM module to perform their own function. Interfacing is done by their respective driver's circuits of the components.

In the circuit we have used MQ-6 sensor for gas leakage detection. MQ-6 sensor composed by micro aluminium oxide (AL_2O_3) ceramic tube, Tin Dioxide (SnO_2) sensitive layer, measuring electrode and heater are fixed into a crust made by plastic and stainless steel net. The heater provides necessary work conditions for work of sensitive components. The enveloped MQ-6 has 6 pins, 4 of them are used to fetch signals, and other 2 are used for providing heating current. Here MQ-6 sensor works on basics of combustion process, and output is given in variable voltage form, so when LPG gas is leaked voltage at the output pin of MQ-6 is increased and we use IC2 (Op-amp LM324) as a comparator to compare the LPG leakage with respect to normal condition. Output of comparator is fed to IC1 microcontroller (ATMEL 89S8253) and corresponding coding LCD is display gas leakage and give another instruction to stepper motor via ULN2803 to turn 180^o to turn off the regulator of gas tank.

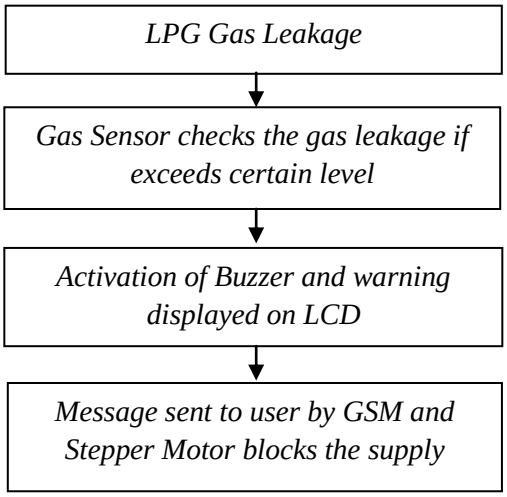


Fig 1.1: Flow Chart

1.3 Realization

The project started off with a brainstorming session. All ideas about how the project should work and what functions it should have were written on a project notebook. We discussed all possible solutions and ideas that we had come up with and removed things that were not possible to implement within this project scope.

All that was left in the notebook in the end is complied into the schematic diagram.

1.4 Design

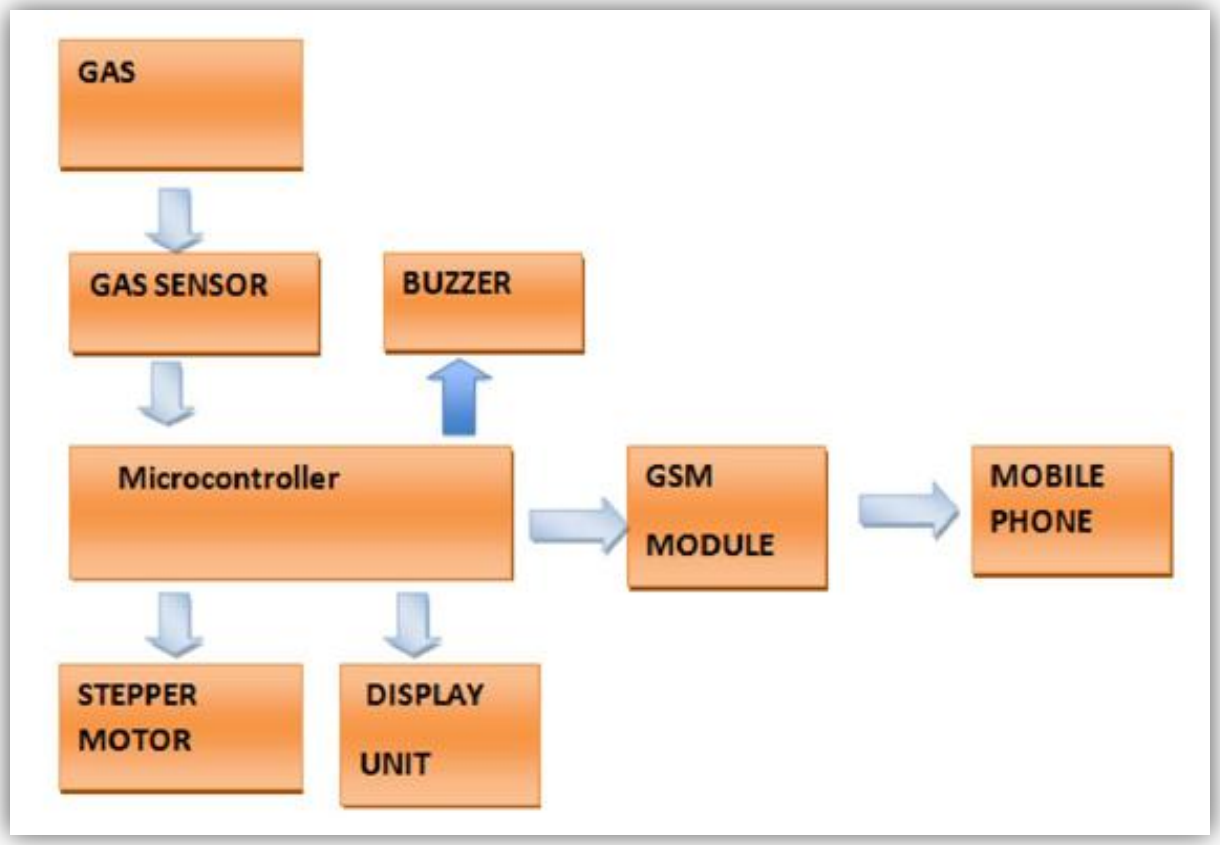


Fig 1.2: Basic Design

CHAPTER 2

CIRCUIT SIMULATION

2.1 Circuit Diagram

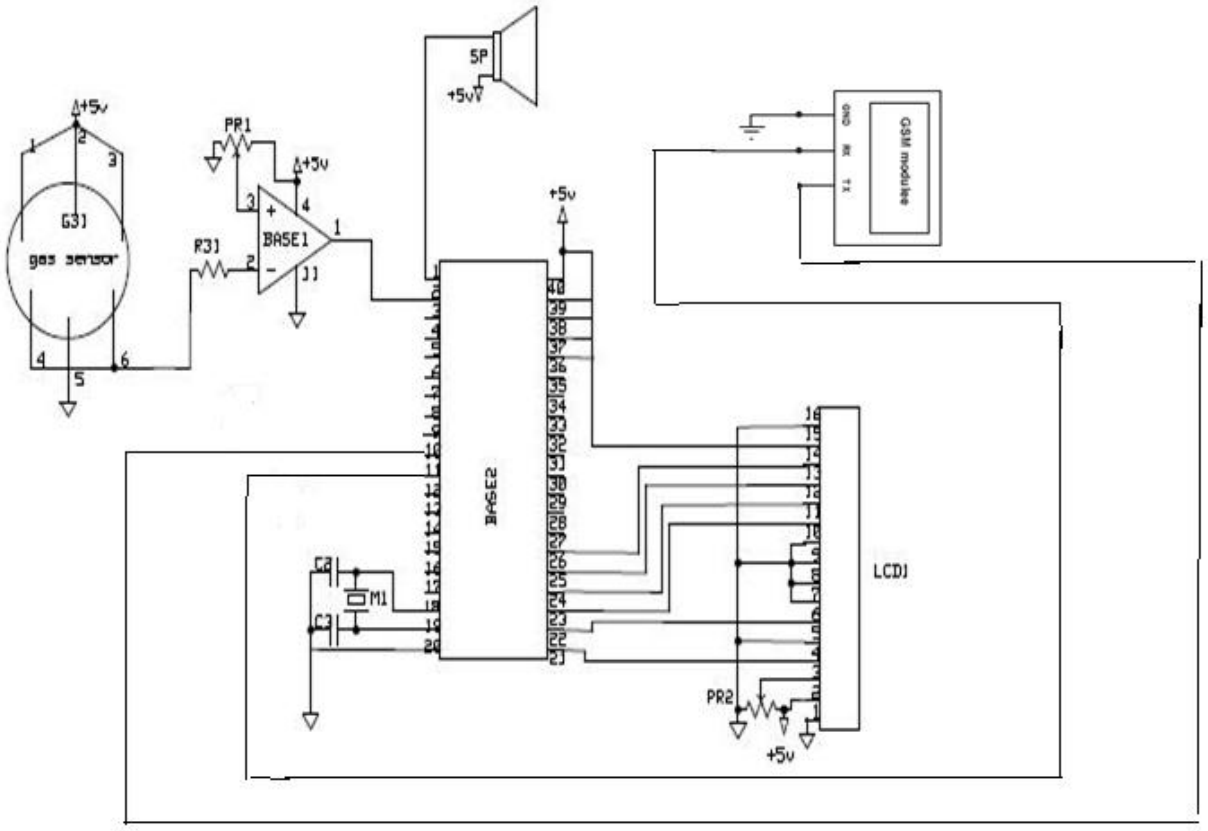


Fig 2.1: The Circuit

2.2 Microcontroller Coding

Programming or burning a microcontroller means to transfer the program from the compiler to the memory of the microcontroller. A compiler is software which provides an environment to write, test and debug a program for the microcontroller. The program for a microcontroller is generally written in C or assembly language. Finally the compiler generates a hex file which contains the machine language instruction understandable by a microcontroller. It is the content of this hex file which is transferred to the memory of the microcontroller. Once a program is transferred or written in the memory of the microcontroller, it then works in accordance with the program. To program a microcontroller, we need a device called a burner/programmer. A programmer is a hardware device with dedicated software which reads the content of the hex file stored on the PC or the laptop and transfers it to the microcontroller to be burned. It reads the data of the hex file by connecting itself to the PC via a serial or USB cable and transfers the data to the memory of the microcontroller to be programmed in accordance with the protocols as described by the manufacturer in the datasheet.

2.2.1 Creating HEX file

This software was used to generate the HEX code that will be transferred to the microcontroller.

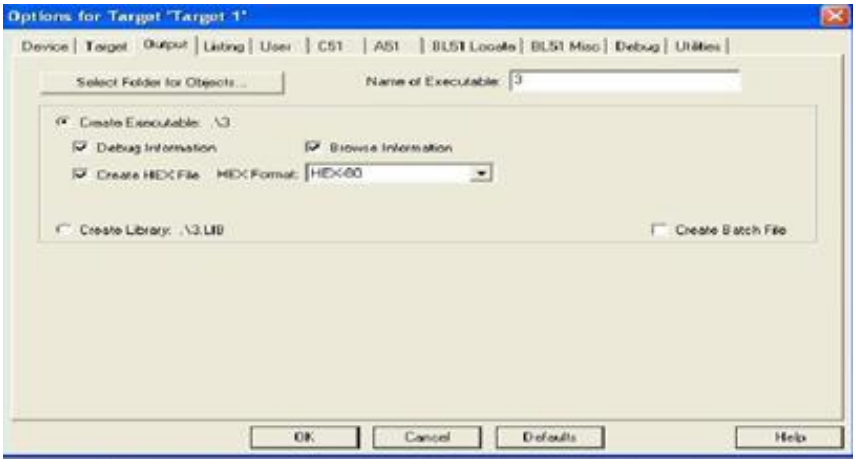


Fig 2.2: Creating HEX file

CHAPTER 3

HARDWARE DESIGN

3.1 Main Components

1. MQ6 Sensor

2. Microcontroller

3. LCD 16*2

4. Buzzer

3.2 MQ6 Sensor

Sensitive material of MQ-6 gas sensor is SnO_2, which has lower conductivity in clean air. When the target combustible gas exist, the sensor's conductivity is higher along with the gas concentration rising. An electronic design is needed to convert this change of conductivity to correspond output signal as per the gas concentration. MQ-6 gas sensor has high sensitivity to Propane, Butane and LPG, also response to Natural gas. The sensor could be used to detect different combustible gas, especially methane; it is with low cost and suitable for different application.

Fig 3.1: MQ6

3.2.1 Characteristics

1. Good sensitivity to combustible gas in wide range.

2. High sensitivity to Propane, Butane and LPG.

3. Long life and low cost.

4. Simple drive circuit.

3.2.2 LM324 as Comparator

In the comaprator mode of operation the op-amp compares the input voltage of the system with a pre-defined threshold voltage and depending on the mode of operation of the op-amp generates the output in digital format. If the mode of operation is inverting in that case if the input is less than the threshold the output is dugital high else low and vice versa is the case for non- inverting mode of operation i.e. if input is higher than the threshold voltage output is high else the output is low. Finally, it is interpreted that the op-amp in comparator mode is used for analog variation in two digital discrete levels of voltages. This is used to convert analog output of sensor to digital because microcontroller works on digital input.

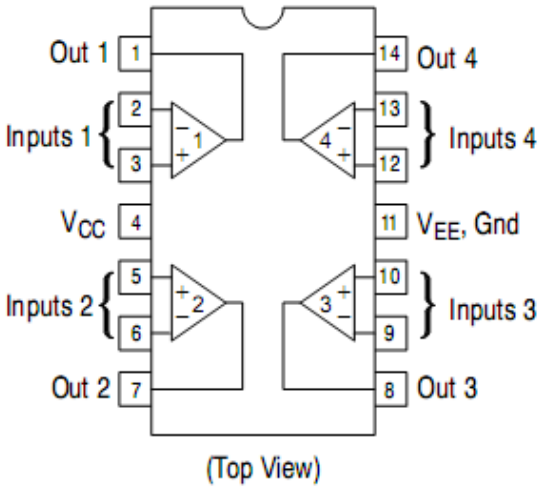


Fig 3.2: Pin Out

3.3 Microcontroller

A microcontroller is a computer with most of the necessary support chips on-board. All computers have several things in common, namely:

1. A central processing unit (CPU) that 'executes' programs.
2. Some read only memory (ROM) where programs to be executed can be stored.
3. Input and output (I/O) devices that enable communication to be established with the outside world i.e. connection to devices such as keyboard, mouse, monitors and other peripherals.

There are a number of other common characteristics that define microcontrollers. If a computer matches a majority of these characteristics, then it can be classified as a microcontroller.
Microcontrollers may be:

1. Embedded inside some other device (often a consumer product) so that they can control the features or actions of the product. Another name for a microcontroller is therefore an 'embedded controller'.
2. Dedicated to one task and run one specific program. The program is stored in ROM and generally does not change.
3. A low-power device, a battery-operated microcontroller might consume as little as 50mW.
4. A microcontroller may take an input from the device it is controlling and controls the device by sending signals to different components in the device.

3.3.1 ATMEL AT89s8253

The Atmel AT89 series is an Intel 8051-compatible family of 8 bit microcontrollers (µCs) manufactured by the Atmel Corporation. Based on the Intel 8051 core, the AT89 series remains very popular as general purpose microcontrollers, due to their industry standard instruction set, and low unit cost. This allows a great amount of legacy code to be reused without modification in new applications.

The AT89s8253 is a low power, high performance CMOS 8-bit micro computer with 8K bytes of flash programmable and erasable read only memory(PEROM).The device is

manufactured using Atmel's high density non-volatile memory technology and is compatible with the industry standard 80c51 and 80C52 instruction set and pin out. The on-chip flash allows the program memory to be reprogrammed in system or by a conventional non-volatile memory programmer. By combining a versatile 8-bit CPU with flash on a monolithic chip, the Atmel AT89s8253 is a powerful microcomputer which provides a highly flexible and cost effective solution to many embedded control applications. The AT89S8253 provides the following standard features: 12K bytes of In-System Programmable Flash, 2K bytes of EEPROM, 256 bytes of RAM, 32 I/O lines, programmable watchdog timer, two data pointers, three 16-bit timer/counters, a six-vector, four-level interrupt architecture, a full duplex serial port, on-chip oscillator, and clock circuitry.

In addition, the AT89S8253 is designed with static logic for operation down to zero frequency and supports two software selectable power saving modes. The Idle Mode stops the CPU while allowing the RAM, timer/counters, serial port, and interrupt system to continue functioning. The Power-down mode saves the RAM contents but freezes the oscillator, disabling all other chip functions until the next external interrupt or hardware reset. The on-board Flash/EEPROM is accessible through the SPI serial interface. Holding RESET active forces the SPI bus into a serial programming interface and allows the program memory to be written to or read from, unless one or more lock bits have been activated.

3.3.2 Microcontroller Architecture

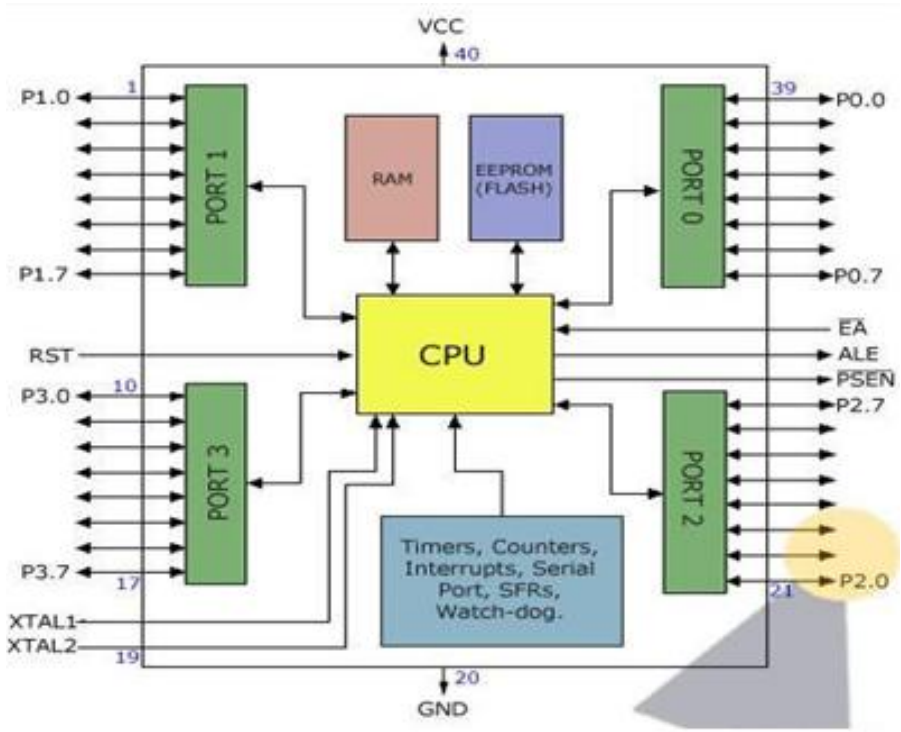


Fig 3.3: Interfacing with CPU

The figure 3.3 shows the main features and components that the designer can interact with. You can notice that the AT 89S8253 has four different ports, each one having eight Input/output lines providing a total of 32 I/O lines. Those ports can be used to output DATA and orders do other devices, or to read the state of a sensor, or a switch. Most of the ports of the 89S52 have 'dual function' meaning that they can be used for two different functions: the first one is to perform input/output operations and the second one is used to implement special features of the microcontroller like counting external pulses, interrupting the execution of the program according to external events, performing serial data transfer or connecting the chip to a computer to update the software. Each port has eight pins, and will be treated from the software point of view as an 8-bit variable called 'register', each bit being connected to a different input/output pin.

3.3.3 Pin Out of Microcontroller

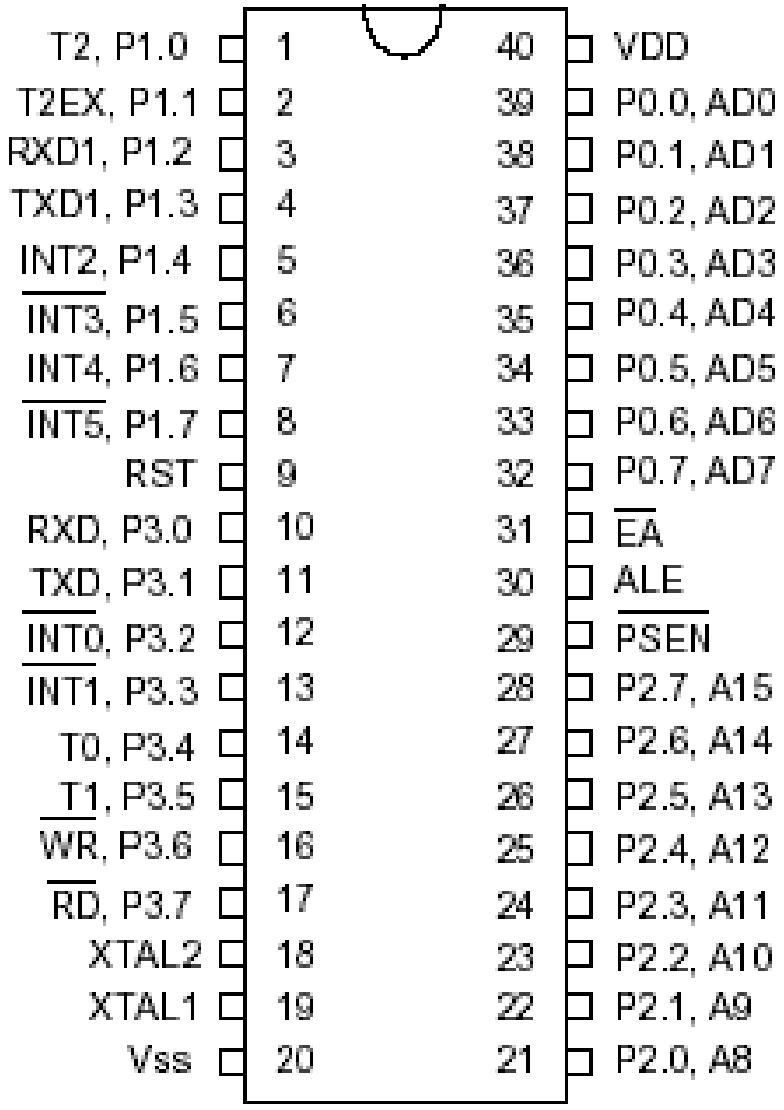


Fig 3.4: Pin Out

3.3.4 Function of Ports:

V_{CC} : Pin no. 40 is used for the supply to the micro-controller.

GND : Pin no. 20 acts as ground.

RST : This is pin no.9, used to reset the device by keeping it high for 2 machine cycles. The microcontroller should be reset at the time of starting.

Oscillator : Pins XTAL1 and XTAL2 are used for connecting a quartz crystal for the internal oscillator. Crystal frequency-10MHz.

External Access (EA): The 8051 family members all come with on chip ROM to store the program. In such case, EA pin is connected to V_{cc}.

PSEN : PSEN stands for Program Store Enable. This is an output pin and is connected to OE pin of ROM.

Port 0 : Port 0 is an 8-bit open drain bi-directional I/O port. As an output port, each pin can sink eight TTL inputs. When 1s are written to port 0 pins, the pins can be used as high impedance inputs. Port 0 also be configured to be the multiplexed lower order address/data bus during accesses to external program and data memory. In this mode it has internal pull ups.

Port 1 and Port 2 : Port 1 is an 8-bit bi-directional I/O port with internal pull-ups. The Port1 output buffers can sink/source four TTL inputs. When 1s are written to Port 1 pins, they are pulled high by the internal pull-ups and can be used as inputs. As inputs, Port 1 pins that are externally being pulled low will source current (IIL) because of the internal pull-ups. In addition, P1.0 and P1.1 can be configured to be the

timer/counter 2 external count input (P1.0/T2) and the timer/counter 2.

Port 3 : It has internal pull-ups and can sink/source 4 TTL inputs. Port 3 occupies a total of 8 pins, pins 10 through 17. It can be used as input or output. Port 3 has additional function of providing some extremely signal as interrupts.

Port Pin	Alternate Functions
P3.0	RXD (serial input port)
P3.1	TXD (serial output port)
P3.2	$\overline{INT0}$ (external interrupt 0)[1]
P3.3	$\overline{INT1}$ (external interrupt 1)[1]
P3.4	T0 (timer 0 external input)
P3.5	T1 (timer 1 external input)
P3.6	$\overline{WR}$ (external data memory write strobe)
P3.7	$\overline{RD}$ (external data memory read strobe)

Table 3.1: Ports and their functions

ALE/PROG : Address Latch Enable is an output pulse for latching the low byte of the address (on its falling edge) during accesses to external memory. This pin is also the program pulse input (PROG) during Flash programming. This pin is also the program pulse input (PROG) during Flash programming. In normal operation, ALE is emitted at a constant rate of 1/6 the oscillator frequency and may be used for external timing or locking purposes. Note, however, that one ALE pulse is skipped during each access to external data memory.

PSEN : Program Store Enable. PSEN is the read strobe to external program memory (active low). When the AT89S8253 is executing code from external program

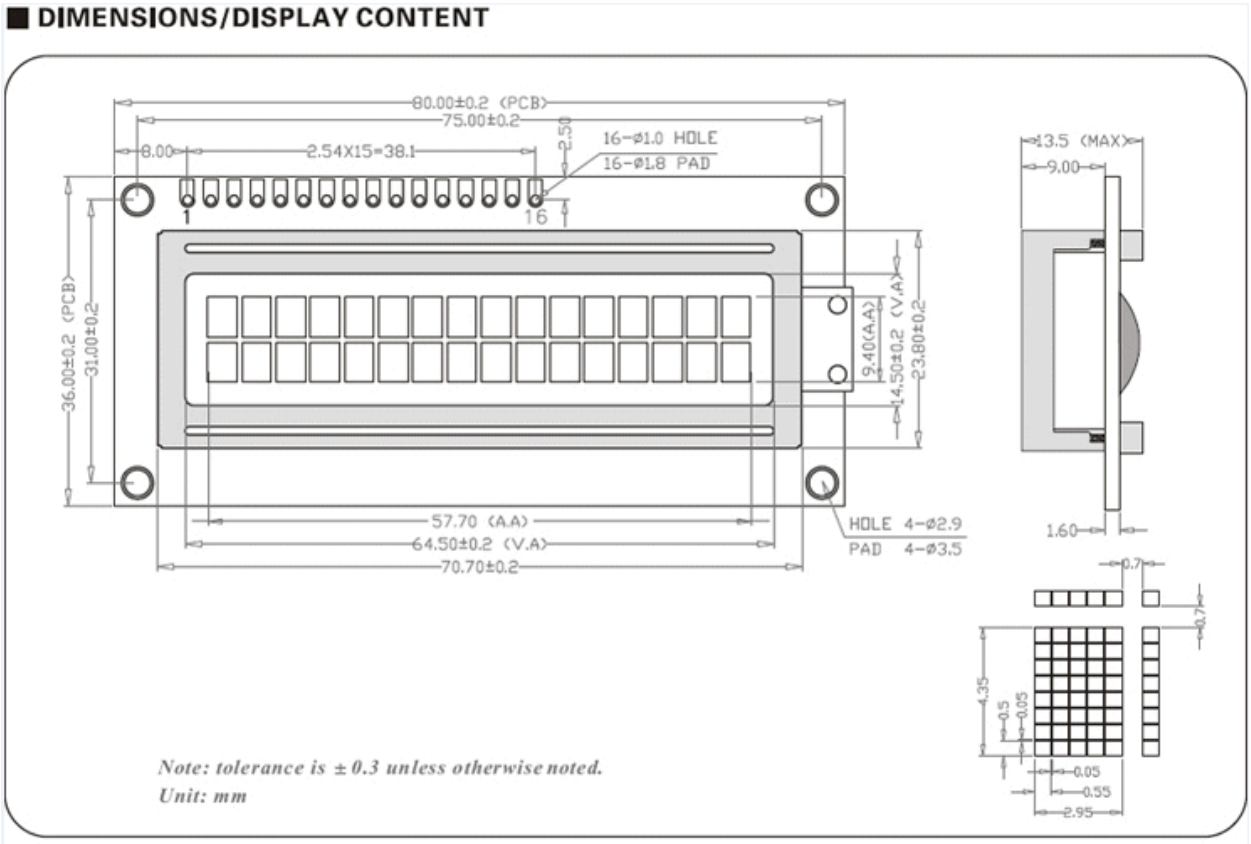
memory, PSEN is activated twice each machine cycle, except that two PSEN activations are skipped during each access to external data memory.

3.4 LCD 16*2

LCD (Liquid Crystal Display) screen is an electronic display module and find a wide range of applications. A 16x2 LCD display is very basic module and is very commonly used in various devices and circuits. These modules are preferred over seven segments and other multi segment LEDs. The reasons being: LCDs are economical, easily programmable, have no limitation of displaying special & even custom characters (unlike in seven segments), animations and so on.

LCDs are available to display arbitrary images (as in a general-purpose computer display) or fixed images which can be displayed or hidden, such as preset words, digits, and 7-segment displays as in digital clock. They use the same basic technology, except that arbitrary images are made up of a large number of small pixels, while other displays have larger elements. LCDs are used in a wide range of applications including computer monitors, televisions, instrument panels, aircraft cockpit displays, and signage. They are common in consumer devices such as video players, gaming devices, clocks, watched, calculators, and telephones, and have replaced cathode ray tube (CRT) displays in most applications. They are available in a wider range of screen sizes than CRT and plasma displays, and since they do not use phosphors, they do not suffer image burn-in. LCDs are, however, susceptible to image persistence.

A 16x2 LCD means it can display 16 characters per line and there are 2 such lines. In this LCD each character is displayed in 5x7 pixel matrix. This LCD has two registers, namely, command and data. The command register stores the command instructions given to the LCD. A command is an instruction given to LCD to do a predefined task like initializing it, clearing its screen, setting the cursor position, controlling display etc. The data register stores the data to be displayed on the LCD. The data is the ASCII value of the character to be displayed on the LCD.



1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Vss	VDD	V0	RS	R/W	E	DB0	DB1	DB2	DB3	DB4	DB5	DB6	DB7	LEDA	LEDK

Fig 3.5: LCD Pin Config. & Dimensions

3.4.1 Features

1. 5 x 8 dots with cursor

2. Built-in controller (KS 066 or Equivalent)

3. + 5V power supply (Also available for + 3V)

4. 1/16 duty cycle

5. B/L to be driven by pin 1, pin 2 or pin 15, pin 16 or A.K (LED)

6. N.V. optional for + 3V power supply

3.4.2 LCD Interface Diagram

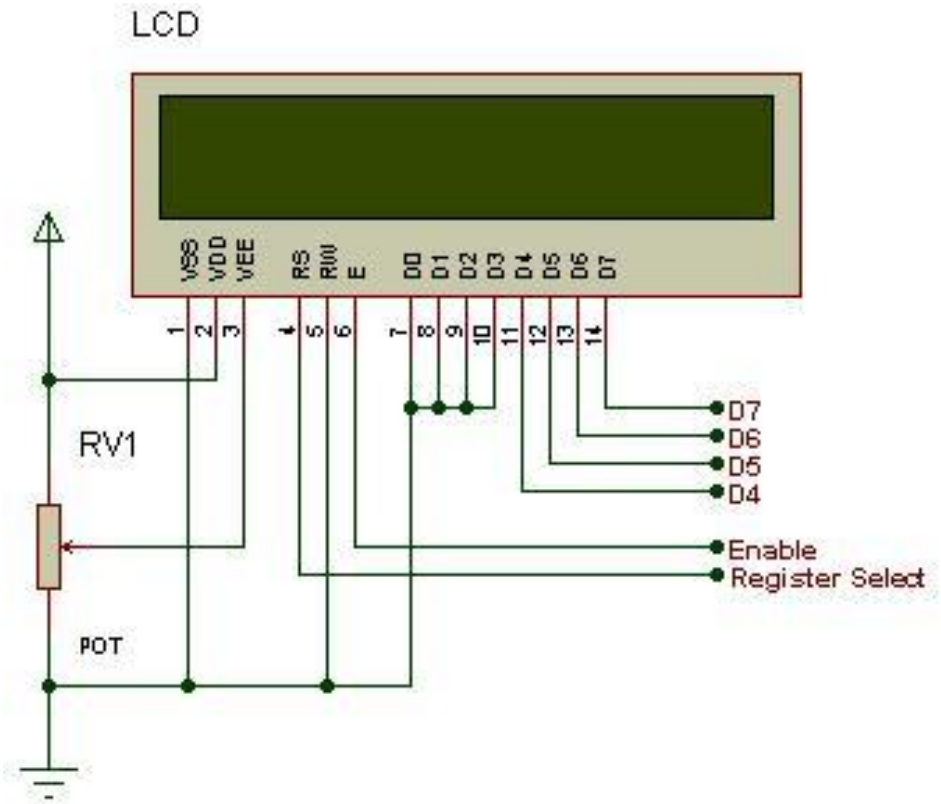


Fig 3.6: LCD Interfacing

Figure 3.6 shows connection diagram of LCD in 4-bit mode, where we only need 6 pins to interface an LCD. D4-D7 is the data pins connection and Enable and Register select are for LCD control pins. We are not using Read/Write (RW) Pin of the LCD, as we are only writing on the LCD so we have made it grounded permanently. If you want to use it, then you may connect it on your controller but that will only increase another pin and does not make any big difference. Potentiometer RV1 is used to control the LCD contrast. The unwanted data pins of LCD i.e. D0-D3 are connected to ground.

3.5 Buzzer

A buzzer is an audio signaling device, which may be mechanical, electromechanical, or piezoelectric. Typical uses of buzzers and beepers include alarm devices, timers and confirmation of user input such as a mouse click or keystroke.

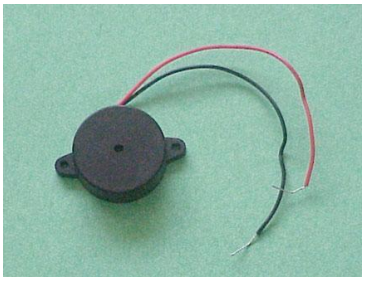


Fig 3.7: Buzzer

3.5.1 Mechanical

A joy buzzer is an example of a purely mechanical buzzer.

3.5.2 Electro-mechanical

Early devices were based on an electromechanical system identical to an electric bell without the metal gong. Similarly, a relay may be connected to interrupt its own actuating

current, causing the contacts to buzz. Often these units were anchored to a wall or ceiling to use it as a sounding board. The word buzzer comes from the rasping noise that electromechanical buzzers made.

Fig 3.8: EM Buzzer

3.5.3 Piezoelectric

A piezoelectric element may be driven by an oscillating electronic circuit or other audio signal source, driven with a piezoelectric audio amplifier. Sounds commonly used to indicate that a button has been pressed are a click, a ring or a beep.

Fig 3.9: Piezo Buzzer

3.5.4 Buzzer Interfacing

If the proper potential is provided across the buzzer it generates the sound else not. It is of two pin configuration that is anode (+) and cathode (-). For interfacing to any device like controller etc. we directly connect the cathode to the GND and anode to the device like controller. Now when signal on anode goes high buzzer activates else does-not. Vice-versa is also true. A piezoelectric element may be driven by an oscillating electronic circuit or other audio signal source, driven with a piezoelectric audio amplifier. Sounds commonly used to indicate that a button has been pressed are a click, a ring or a beep. When the input port pin from microcontroller is changed, the sound wave is changed in Buzzer. Here we have interfaced the buzzer to the port P1 of the microcontroller and in our code a simple ON/OFF program is written. The triggering of the buzzer depends on the signal which it will receive from the sensor. The buzzers negative terminal or cathode is connected to the pin number one and the buzzers positive terminal or the anode is connected to a supply of plus five volts. Now when the output at pin one of the microcontroller is high then there is no potential drop across the terminals of the buzzer and hence the buzzer do not beeps. But when the output at pin one of the microcontroller is low then there is a potential drop across the buzzer and buzzer starts beeping. Therefore this whole process depends on the output of the sensor which is in analog form and is converted into digital form by a comparator circuit.

CHAPTER 4

STEPPER MOTOR

A stepper motor (or step motor) is a brushless DC electric motor that divides a full rotation into a number of equal steps. The motor's position can then be commanded to move and hold at one of these steps without any feedback sensor (an open-loop controller), as long as the motor is carefully sized to the application.

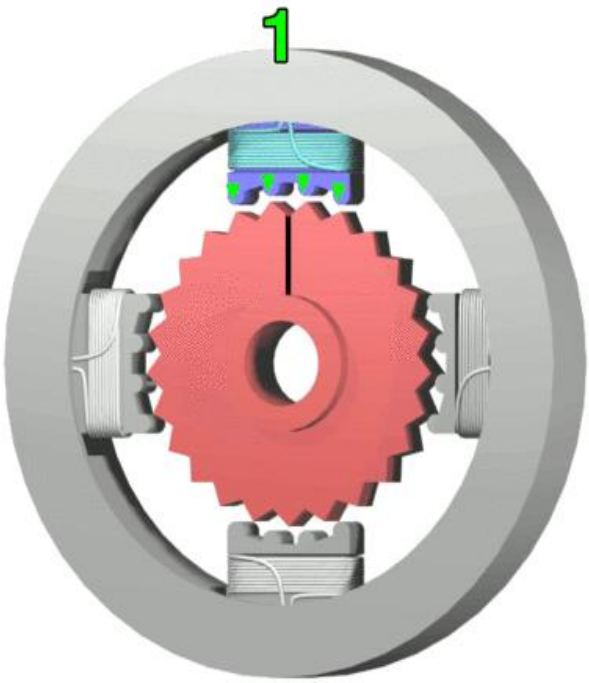


Fig 4.1: Stepper Motor

4.1 Introduction

A stepper motor (or step motor) is a brushless, synchronous electric motor that can divide a full rotation into a large number of steps. The motor's position can be controlled precisely without any feedback mechanism, as long as the motor is carefully sized to the application. Stepper motors are similar to switched reluctance motors which are very large stepping motors with a reduced pole count, and generally are closed-loop commutated. Stepper motors operate differently from DC brush motors, which rotate when voltage is applied to their terminals. Stepper motors, on the other hand, effectively have multiple "toothed" electromagnets arranged around a central gear-shaped piece of iron. The electromagnets are energized by an external control circuit, such as a microcontroller.

To make the motor shaft turn, first one electromagnet is given power, which makes the gear's teeth magnetically attracted to the electromagnet's teeth. When the gear's teeth are thus aligned to the first electromagnet, they are slightly offset from the next electromagnet. So, when the next electromagnet is turned on and the first is turned off, the gear rotates slightly to align with the next one, and from there the process is repeated. Each of those slight rotations is called a "step," with an integer number of steps making a full rotation. In that way, the motor can be turned by a precise angle.

Fig 4.2: Stepper Motor

4.1.2 Stepper Motor Characteristics

- ✓ Stepper motors are constant power devices.
- ✓ As motor speed increases, torque decreases.
- ✓ The torque curve may be extended by using current limiting drivers and increasing the driving voltage.
- ✓ Steppers exhibit more vibration than other motor types, as the discrete step tends to snap the rotor from one position to another.
- ✓ This vibration can become very bad at some speeds and can cause the motor to lose torque.
- ✓ The effect can be mitigated by accelerating quickly through the problem speeds range, physically damping the system, or using a micro-stepping driver.
- ✓ Motors with a greater number of phases also exhibit smoother operation than those with fewer phases.

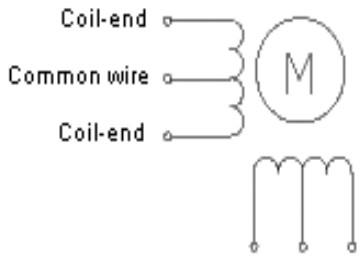


Fig 4.3: Internal Structure

4.2 Types of Stepper Motor

There are three main types of stepper motors:

1. *Permanent Magnet Stepper*
2. *Hybrid Synchronous Stepper*
3. *Variable Reluctance Stepper*

Permanent magnet motors use a permanent magnet (PM) in the rotor and operate on the attraction or repulsion between the rotor PM and the stator electromagnets. Variable reluctance (VR) motors have a plain iron rotor and operate based on the principle of that minimum reluctance occurs with minimum gap, hence the rotor points are attracted toward the stator magnet poles. Hybrid stepper motors are named because they use a combination of PM and VR techniques to achieve maximum power in a small package size.

4.3 Stepper Motor Basics

Control of a stepper motor comes from applying a specific step sequence; rotational speed is controlled by the timing of the applied steps. The simplified diagrams below illustrate the effect of phase sequencing on rotational motion.

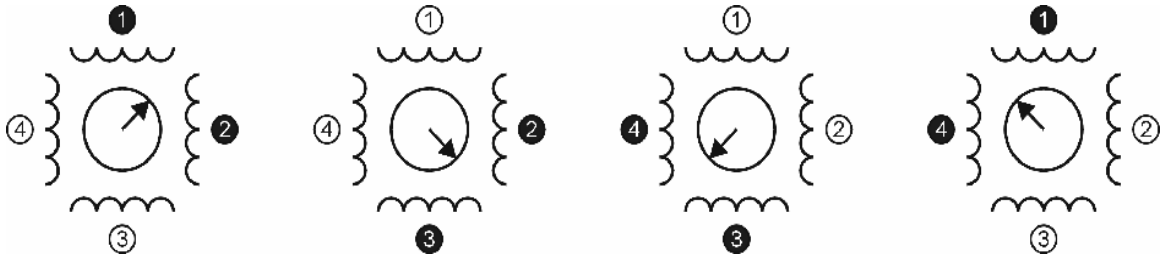


Fig 4.4(a) Full Step, High Torque (standard application)

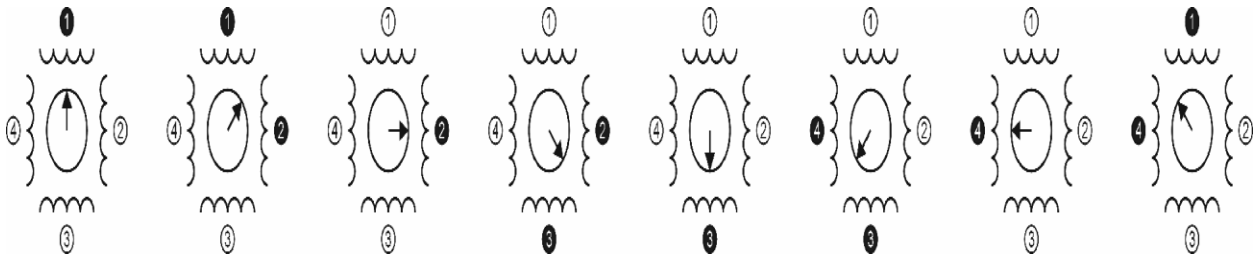


Fig 4.4(b) Half Step (best precision)

4.4 Working of Stepper Motor

Stepper motors consist of a permanent magnet rotating shaft, called the rotor, and electromagnets on the stationary portion that surrounds the motor, called the stator. Figure 1 illustrates one complete rotation of a stepper motor. At position 1, we can see that the rotor is beginning at the upper electromagnet, which is currently active (has voltage applied to it). To move the rotor clockwise (CW), the upper electromagnet is deactivated and the right electromagnet is activated, causing the rotor to move 90 degrees CW, aligning itself with the active magnet. This process is repeated in the same manner at the south and west electromagnets until we once again reach the starting position. In the example, we used a motor with a resolution of 90 degrees or demonstration purposes. In reality, this would not be a very practical motor for most applications. The average stepper motor's resolution -- the amount of degrees rotated per pulse -- is much higher than this. For example, a motor with a resolution of 5 degrees would move its rotor 5 degrees per step, thereby requiring 72 pulses (steps) to complete a full 360 degree rotation.

One may double the resolution of some motors by a process known as half-stepping. Instead of switching the next electromagnet in the rotation on one at a time, with half stepping one needs to turn on both electromagnets, causing an equal attraction between, thereby doubling the resolution. As it can be seen in Figure 4.4 b, in the first position only the upper electromagnet is active, and the rotor is drawn completely to it. In position 2, both the top and right

electromagnets are active, causing the rotor to position itself between the two active poles. Finally, in position 3, the top magnet is deactivated and the rotor is drawn all the way right. This process can then be repeated for the entire rotation.

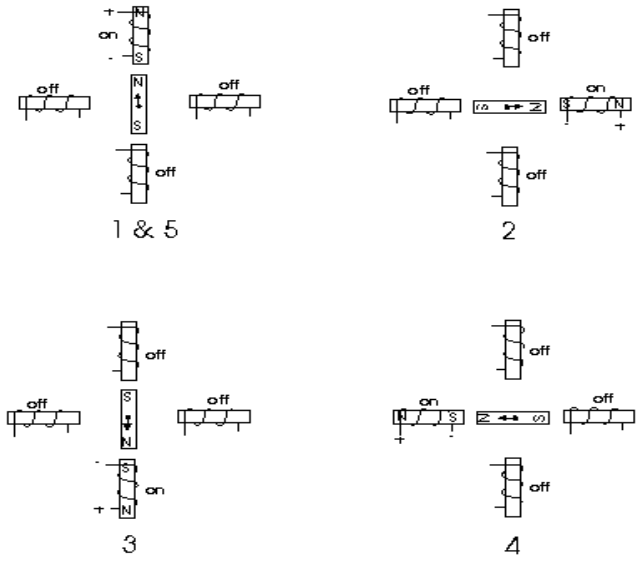


Fig 4.5 a

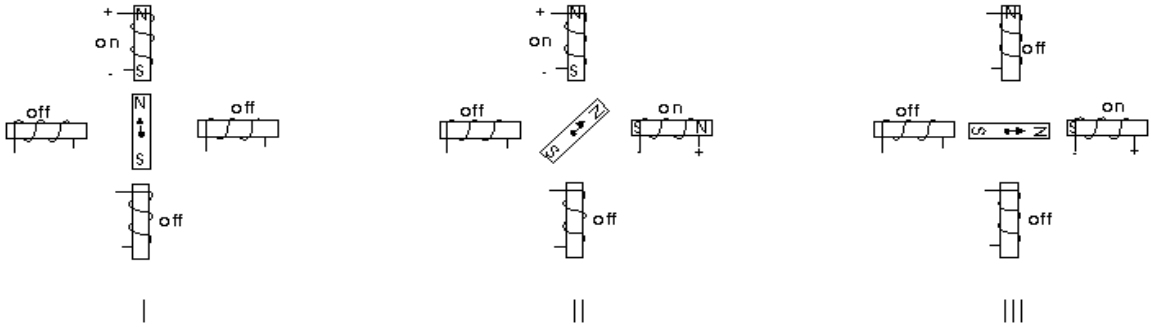


Fig 4.5 b: Polarity Changes

There are several types of stepper motors. 4-wire stepper motors contain only two electro-magnets, however the operation is more complicated than those with three or four magnets, because the driving circuit must be able to reverse the current after each step. For our purposes, we will be using a 6-wire motor. Unlike these example motors which rotated 90 degrees per step, real-world motors employ a series of mini-poles on the stator and rotor to increase resolution. Although this may seem to add more complexity to the process of driving the motors, the operation is identical to the simple 90 degree motor we used in our example.

4.5 Stepper Motor Driving Circuits

Stepper motor performance is strongly dependent on the drive circuit. Torque curves may be extended to greater speeds if the stator poles can be reversed more quickly, the limiting factor being the winding inductance. To overcome the inductance and switch the windings quickly, one must increase the drive voltage. This leads further to the necessity of limiting the current that these high voltages may otherwise induce.

- *L R Drive circuits:* L/R drive circuits are also referred to as constant voltage drives because a constant positive or negative voltage is applied to each winding to set the step positions. However, it is winding current, not voltage that applies torque to the stepper motor shaft. The current I in each winding is related to the applied voltage V by the winding inductance L and the winding resistance R. The resistance R determines the maximum current according to Ohm's law I=U/R. The inductance L determines the maximum rate of change of the current in the winding according to the formula for an Inductor dI/dt = U/L. Thus when controlled by an L/R drive, the maximum speed of a stepper motor is limited by its inductance since at some speed, the voltage U will be changing faster than the current I can keep up. With an L/R drive it is possible to control a low voltage resistive motor with a higher voltage drive simply by adding an external resistor in series with each winding.
- *Chopper Drive circuits:* Chopper drive circuits are also referred to as constant current drives because they generate a somewhat constant current in each winding rather than applying a constant voltage. On each new step, a very high voltage is applied to the winding initially. This causes the current in the winding to rise quickly since dI/dt = V/L

where V is very large. The current in each winding is monitored by the controller, usually by measuring the voltage across a small sense resistor in series with each winding. When the current exceeds a specified current limit, the voltage is turned off or "chopped", typically using power transistors. When the winding current drops below the specified limit, the voltage is turned on again. In this way, the current is held relatively constant for a particular step position. This requires additional electronics to sense winding currents, and control the switching, but it allows stepper motors to be driven with higher torque at higher speeds than L/R drives. Integrated electronics for this purpose are widely available.

4.5.1 Phase Current Waveforms

A stepper motor is a poly-phase AC synchronous motor and it is ideally driven by sinusoidal current. A full step waveform is a gross approximation of a sinusoid, and is the reason why the motor exhibits so much vibration. Various drive techniques have been developed to better approximate a sinusoidal drive waveform: these are half stepping and micro-stepping.

- *Full step drive (two phases on):* This is the usual method for full step driving the motor. Both phases are always on. The motor will have full rated torque.
- *Wave drive:* In this drive method only a single phase is activated at a time. It has the same number of steps as the full step drive, but the motor will have significantly less than rated torque. It is rarely used.
- *Half stepping :* When half stepping, the drive alternates between two phases on and a single phase on. This increases the angular resolution, but the motor also has less torque at the half step position (where only a single phase is on). This may be mitigated by increasing the current in the active winding to compensate. The advantage of half stepping is that the drive electronics need not change to support it.
- *Micro-stepping:* What is commonly referred to as micro-stepping is actual "sine cosine micro-stepping" in which the winding current approximates a sinusoidal AC waveform. Sine cosine micro-stepping is the most common form, but other waveforms

are used. Regardless of the waveform used, as the micro-steps become smaller, motor operation becomes more smooth, thereby greatly reducing resonance in any parts the motor may be connected to, as well as the motor itself. It should be noted that while micro-stepping appears to make running at very high resolution possible, this resolution is rarely achievable in practice regardless of the controller, due to mechanical restriction and other sources of error between the specified and actual positions. In professional equipment gear heads are the preferred way to increase angular resolution.

Step size repeatability is an important step motor feature and a fundamental reason for their use in positioning. Example: many modern hybrid step motors are rated such that the travel of every Full step (example 1.8 degrees per full step or 200 full steps per revolution) will be within 3% or 5% of the travel of every other Full step; as long as the motor is operated within its specified operating ranges. Several manufacturers show that their motors can easily maintain the 3% or 5% equality of step travel size as step size is reduced from full stepping down to 1/10th stepping. Then, as the micro-stepping divisor number grows, step size repeatability degrades. At large step size reductions it is possible to issue many micro-step commands before any motion occurs at all and then the motion can be a "jump" to a new position.

4.6 Stepper Motor Driver: LM293D

The most common method to drive DC motors in two directions under control of a computer is with an H-bridge motor driver. H-bridges can be built from scratch with bi-polar junction transistors (BJT) or with field effect transistors (FET), or can be purchased as an integrated unit in a single integrated circuit package such as the L293. The L293 is simplest and inexpensive for low current motors, for high current motors, it is less expensive to build your own H-bridge from scratch.

The L293D is an integrated circuit motor driver that can be used for simultaneous, bi-directional control of two small motors (small means small). The L293D is limited to 600 mA, but in reality can only handle much small currents unless you have done some serious heat sinking to keep the case temperature down. Unsure about whether the L293D will work with

your motor? Hook up the circuit and run your motor while keeping your finger on the chip. If it gets too hot to touch, you can't use it with your motor.

The L293D is a quadruple high-current half-H driver designed to provide bidirectional drive currents of up to 600-mA at voltages from 4.5 V to36 V. It is designed to drive inductive loads such as relays, solenoids, dc and bipolar stepping motors, as well as other high-current/high-voltage loads in positive-supply applications .All inputs are TTL-compatible. Each output is a complete totem-pole drive circuit with a darlington transistor as sink and a pseudo-darlington as a source. Drivers are enabled in pairs with drivers 1 and 2enabled by 1,2EN and drivers 3 and 4 enabled b3, 4 EN. When enable input is high, the associated drivers are enabled, and their outputs are active and in phase with their inputs. External high-speed output clamp diodes should be used for inductive transient suppression. When the enable input is low, those drivers are disabled, and their outputs are off and in a high-impedance state. With the proper data inputs, each pair of drivers forms a full-H (or bridge) reversible drive suitable for solenoid or motor applications.

L293D is a bipolar motor driver IC. This is a high voltage, high current push pull four channel driver compatible to TTL logic levels and drive inductive loads. It has 600 mA output current capabilities per channel and internal clamp diodes. The L293 is designed to provide bidirectional drive currents of up to 1A at voltages from 4.5 V to 36 V. The L293D is designed to provide bidirectional drive currents of up to 600-mA at voltages from 4.5 V to 36 V. Both devices are designed to drive inductive loads such as relays, solenoids, dc and bipolar stepping motors, as well as other high-current/high-voltage loads in positive supply applications. All inputs are TTL compatible. Each output is a complete totem-pole drive circuit, with a Darlington transistor sink and a pseudo-darlington source.

Drivers are enabled in pairs, with drivers 1 and 2 enabled by 1,2EN and drivers 3 and 4 enabled by 3,4EN. When enable input high is given then the associated drivers are enabled, and their outputs are active and in phase with their inputs. When the enable input is low, those drivers are disabled, and their outputs are off and in the high-impedance state. With the proper data inputs, each pair of drivers forms a full-H (or bridge) reversible drive suitable for solenoid or motor applications. L293D is a dual H-bridge motor driver integrated circuit (IC). Motor

drivers act as current amplifiers since they take a low-current control signal and provide a higher-current signal. This higher current signal is used to drive the motors.

L293D contains two inbuilt H-bridge driver circuits. In its common mode of operation, two DC motors can be driven simultaneously, both in forward and reverse direction. The motor operations of two motors can be controlled by input logic at pins 2 & 7 and 10 & 15. Input logic 00 or 11 will stop the corresponding motor. Logic 01 and 10 will rotate it in clockwise and anticlockwise directions, respectively.

Enable pins 1 and 9 (corresponding to the two motors) must be high for motors to start operating. When an enable input is high, the associated driver gets enabled. As a result, the outputs become active and work in phase with their inputs. Similarly, when the enable input is low, that driver is disabled, and their outputs are off and in the high-impedance state.

4.6.1 Features

1. 600-mA Output Current Capability Per Driver
2. Pulsed Current 1.2-A Per Driver
3. Output Clamp Diodes for Inductive
4. Transient Suppression
5. Wide Supply Voltage Range
6. 4.5 V to 36 V
7. Separate Input-Logic Supply
8. Thermal Shutdown
9. Internal ESD Protection
10. High-Noise-Immunity Inputs

4.6.2 Pin Diagram

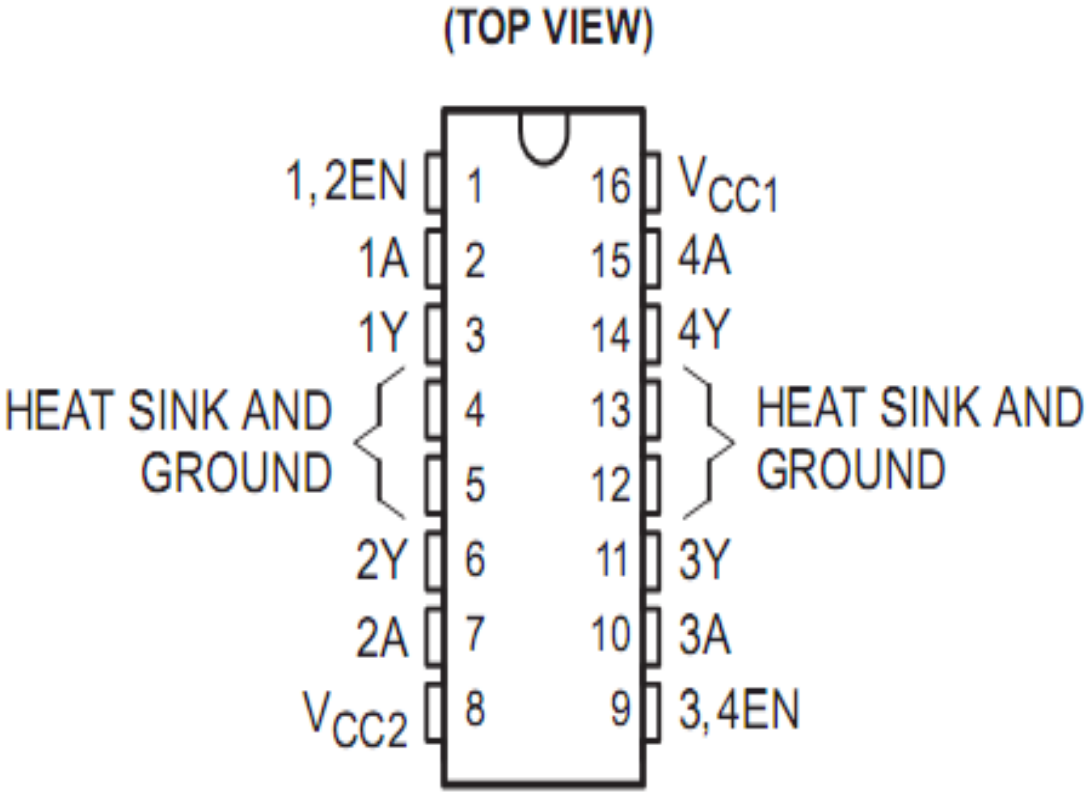


Fig 4.6: Pin Out

Drivers are enabled in pairs with drivers 1 and 2 enabled by 1, 2EN and drivers 3 and 4 enabled b3, 4 EN. When enable input is high, the associated drivers are enabled, and their outputs are active and in phase with their inputs. External high-speed output clamp diodes should be used for inductive transient suppression. When the enable input is low, those drivers are disabled, and their outputs are off and in a high-impedance state. With the proper data inputs, each pair of drivers forms a full-H (or bridge) reversible drive suitable for solenoid or motor applications.

4.6.3 Logic Equivalent

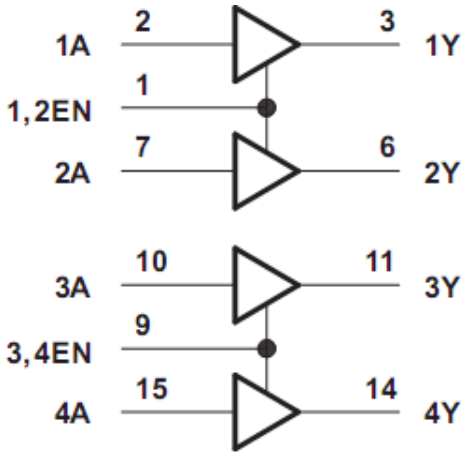


Fig 4.7: Logic Equivalent

FUNCTION TABLE
(each driver)

INPUTS‡		OUTPUT
A	EN	Y
H	H	H
L	H	L
X	L	Z

H = high-level, L = low level,
X = irrelevant, Z = high-impedance (off)
‡ In the thermal shutdown mode, the output is in the high-impedance state regardless of the input levels.

Table 4.1: Logic Equivalent

4.6.4 Motor Driving Circuit

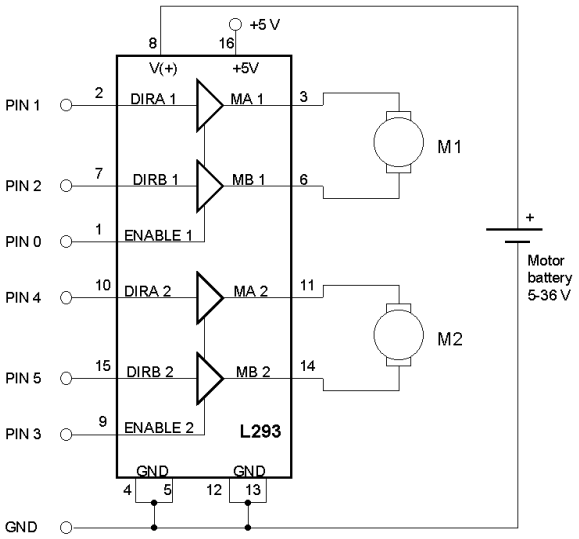


Fig 4.8: Motor Driving Circuit

EN	IN 1	IN 2	Motor Status
0	X	X	Stopped
1	0	0	Stopped
1	1	1	Stopped
1	1	0	CW
1	0	1	CCW

Table 4.2: Motor Status

CHAPTER 5

GSM

5.1 GSM Modem: SIM 900

GSM stands for Global System for Mobile Communications. It is a standard set developed by the European Telecommunications Standards Institute (ETSI) to describe protocols for second generation (2G) digital cellular networks used by mobile phones.

A Modem is a device which modulates and demodulates signals as required to meet the communication requirements. It modulates an analog carrier signal to encode digital information, and also demodulates such a carrier signal to decode the transmitted information. A GSM Modem is a device that modulates and demodulates the GSM signals and in this particular case 2G signals. The modem we are using is SIMCOM SIM 900. It is a Tri-band GSM/GPRS Modem as it can detect and operate at three frequencies (EGSM 900 MHz, DCS 1800 MHz and PCS1900 MHz). Default operating frequencies are EGSM 900MHz and DCS 1800MHz.

SIM 900 is a widely used in many projects and hence many variants of development boards for this have been developed. These development boards are equipped with various features to make it easy to communicate with the SIM 900 module. Some boards provide only TTL interface while some boards include an RS232 interface and some others include an USB interface. If your PC has a serial port (DB9) you can buy a GSM Modem that has both TTL and RS232 interfacings in economy.

SIM 900 GSM module used here, consists of a TTL interface and an RS232 interface. The TTL interface allows us to directly interface with a microcontroller while the RS232 interface includes a MAX232 IC to enable communication with the PC. It also consists of a buzzer, antenna and SIM slot. SIM 900 in this application is used as a DCE (Data Circuit-terminating Equipment) and PC as a DTE (Data Terminal Equipment).

5.2 Why Use a GSM Modem?

GSM Technology has grown so much, that literally there isn't a place on earth where there is no GSM signal. In such a scenario GSM provides us a wide scope in controlling things remotely from any place just with our finger tips. GSM also provides ease to easily communicate in a more robust way.

5.3 AT Commands

SIM 900 GSM Module can be used to send and receive SMS connecting it to a PC when a SIM is inserted. The GSM Modem can be sent commands to send or receive SMS from the PC through a com port (serial port or an usb). These commands are called as AT commands. Through AT commands we can perform several actions like sending and receiving SMS, MMS, etc. SIM 900 has an RS232 interface and this can be used to communicate with the PC. SIM 900 usually operates at a baud rate of 9600, with 1 stop bits, No parity, No Hardware control and 8 data bits. We shall see at some of the AT Commands necessary for sending and receiving SMS.

5.3.1 Some of AT Commands (Taken from: GSM 07.07 AT Command set Ubinetics, www.Zeeman.de)

Command	Description
AT	It is the Prefix of every command sent to the modem. It is also used to test the condition of the modem. The GSM Modem responds with an \r\nOK\r\n or an \r\nERROR\r\n in case of error. Where \r is the carriage return character and \n is a new line feed character).
	Command to check if the Modem has a sim inserted in it. It checks

AT+CSMINS	if the sim is inserted.
AT+CREG	Command to check if the sim is registered with the network. It checks if the sim is registered and returns the status.
ATE1	Command to turn on the ECHO. The GSM Modem continuously echoes back the every byte of data sent to modem until a carriage return character is sensed. It processes the command after a carriage return character is detected. It is usually better to turn off echo to reduce traffic. In this case ECHO is turned on to see how commands are sent and how they are processed.
AT+CMGF=1	Command to set the communication to Text Mode. By default the communication is in the PDU mode.
AT+CMGR=1	Command to read an SMS at the index one. Generally the index depends upon the how many number of SMS that a sim can store. SIM Memory is the only memory available when GSM Modem is used and hence the number of SMS's stored depends on the SIM. It is usually 20. Any message received is arranged in the order of arrival at specific indices.
AT+CMGD=1	Command to delete the SMS at the index 1.
AT+CMGS	Command to send SMS from the GSM Modem.

Table 5.1: AT Commands

5.4 Interfacing with Microcontroller

SIM900 integrates two UART port, one is called serial port, and other is called debug port. Serial port is used to receive AT commands from the MCU while debug port is for firmware updating and bug trace. It is suggested to connect debug port to an external connector for module debug and firmware upgrade consideration. If hardware flow control is not used, DCD, DSR can be left floating. Please refer to the figure given below. DTR can be used to wake up the module from sleep mode and RI can be used to detect a coming call or SMS. These two pins should be connected to GPIO of the MCU.

5.4.1 Schematic Diagram for Interfacing

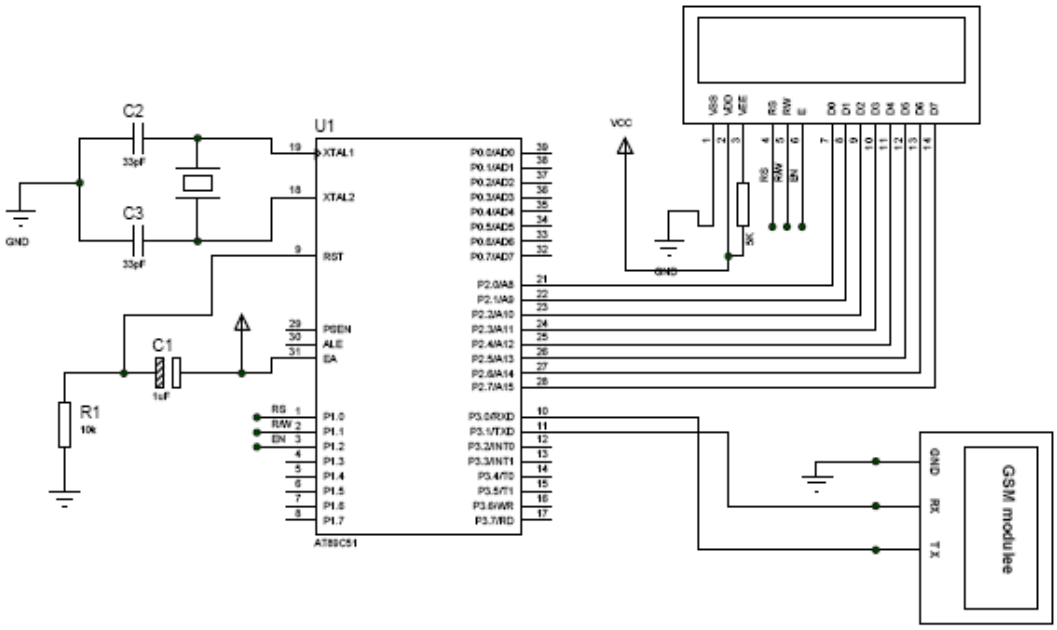


Fig 5.1: Interfacing of GSM

5.4.2 Debugging SIM 900

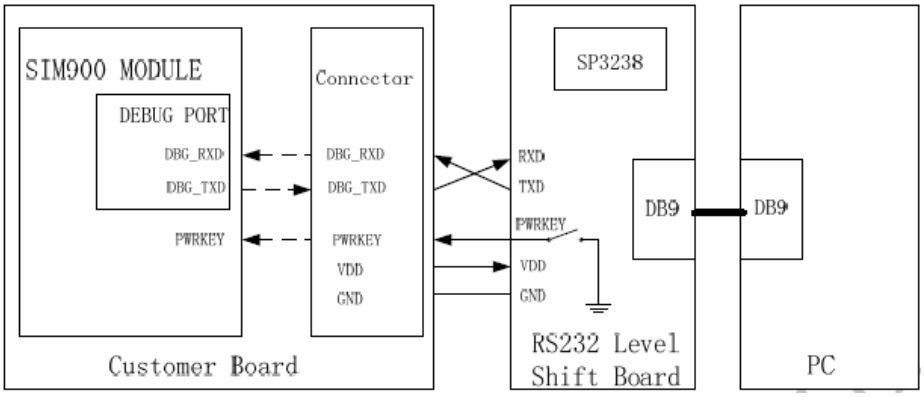


Fig 5.2: Debugging Connections

To debug the SIM 900 device we have to connect it to computer using RS232 port and then 9 pin DB9 port. To programme the SIM 900 we use AT commands by opening the hyper-terminal using DB9 serial bus. The AT commands are mentioned in table 5.1.

RESULTS

We have successfully completed the LPG detection and monitoring system of our project as our assignment for final year project. Overall system is designed and tested by introducing small amount of LPG near gas sensor module. The system detect the level of gas in the air and if it exceeds the safety level then send a SMS to the consumer using GSM modem, activate audio-visual alarm which includes LED, Buzzer to alert the user at home in abnormal condition, display message on LCD and Stepper Motor blocks the supply of LPG.

In this screenshot initially there is no LPG leakage, LCD displays a message: NO leakage

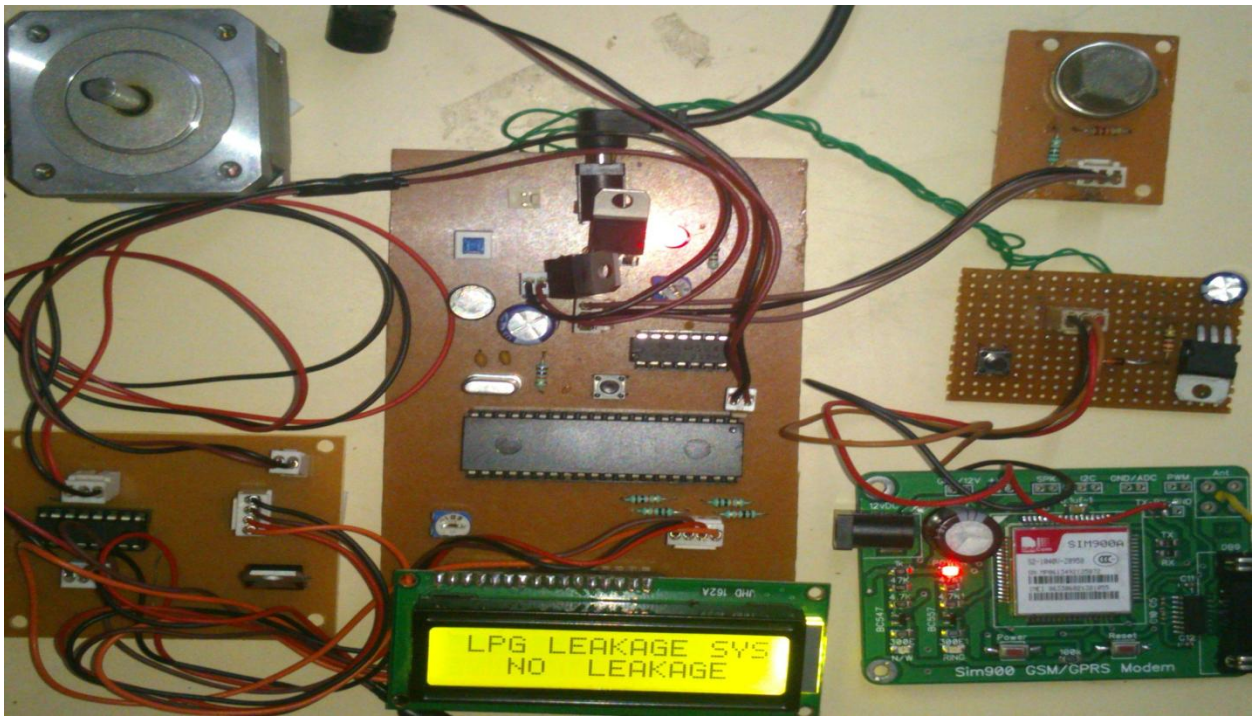


Fig 6.1

In this screenshot there is leakage of LPG as detected by sensor and LCD displays : LPG Leakage

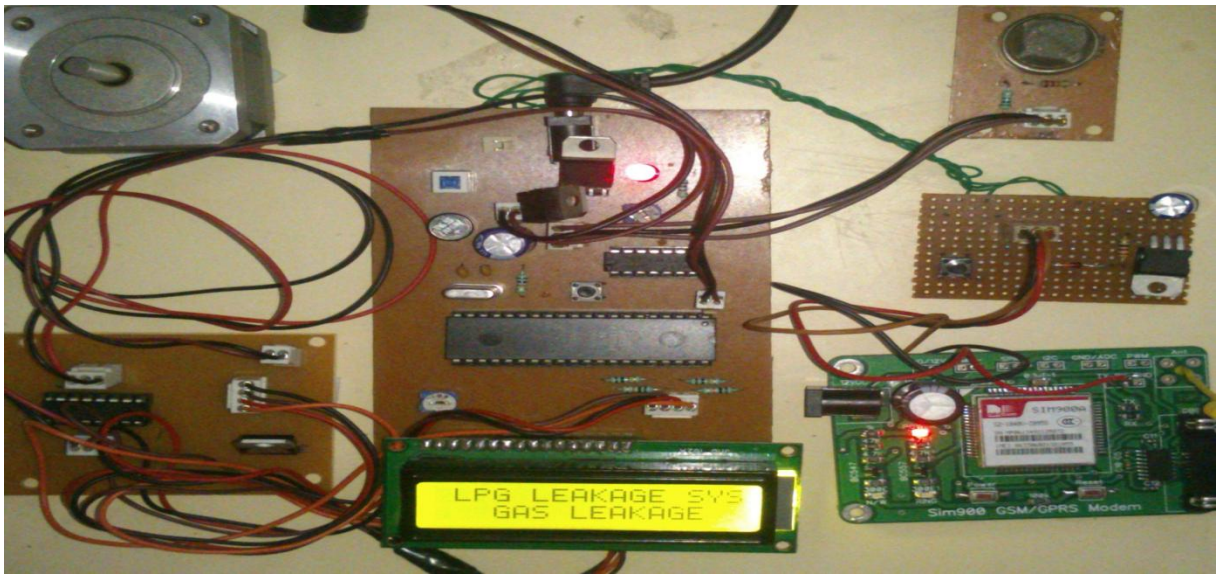


Fig 6.2

After a few seconds the leakage is blocked by stepper motor and message is sent to the user and LCD again displays: No Leakage

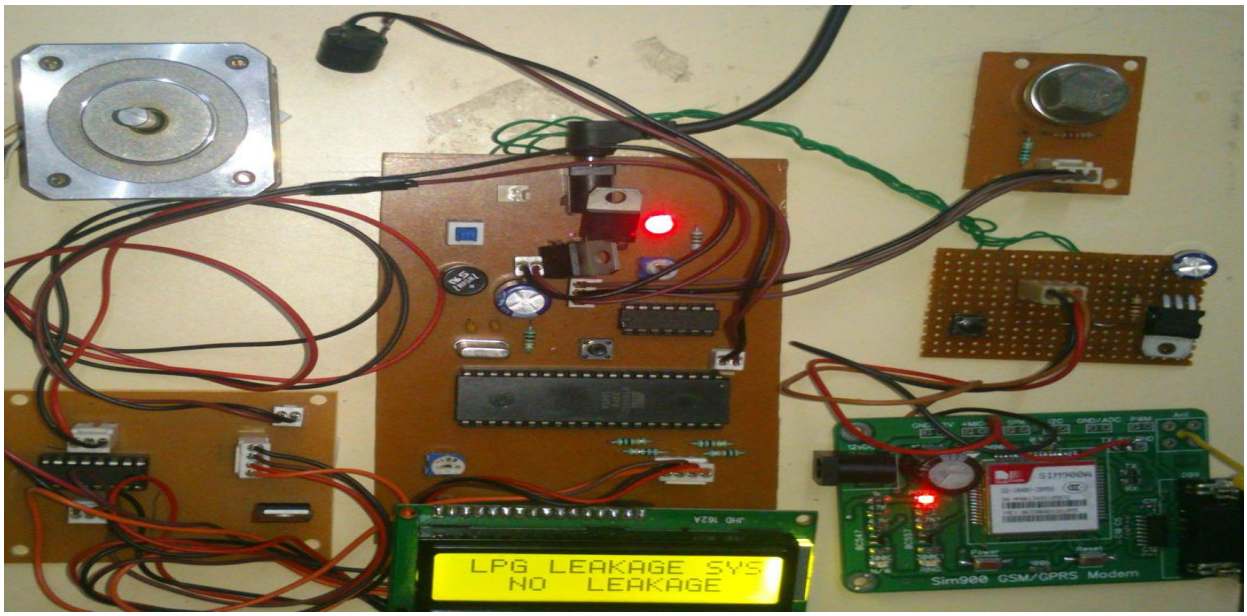


Fig 6.3

CONCLUSION

LPG gas which is used in many applications because of its desirable properties like homes, hotels, industries, vehicles etc. so we can use this device if there is any gas leakage this system detects gas leakage and if gas leakage exceeds certain level this system automatically alert the people by sending the message, alert the people at home by activating Buzzer, displaying warning on LCD display and finally the Stepper Motor closes the knob of the gas supply for preventing the gas leakage. Because of the components used in this system this project forms a cheaper means of creating such system as compared to the other alternatives which are already available in the market.

REFERENCES

[1] Hanwei Electronics Co. Ltd, MQ6 sensor datasheet.

[2] Luay Friwan, Khaldon Lweesy, Aya Bani-Salma, Nour Mani, "A Wireless Home Safety Leakage Detection System", IEEE 2011.

[3] Rajkamal, "Embedded Systems: Architecture, Programming and Design", Tata McGraw Hill Education.

[4] GSM: "Wireless Communications" by T.S Rappaport, PHI, 2009.

[5] Wave share electronics AT89S8253 PDF Datasheet.

[6] Stepper Motor, Basic Electronics and Devices: D.C Kulshreshtha.

[7] W. Chung, and D. Lee, "Real time multi-channel gas leak monitoring System Using CPLD chip, "Sensors and Actuators B", Vol.77,pp.186-189, 2011.

[8] GSM:"Architecture, protocols and services" by Jorg Eberspacher, Christian,Hansjoerg vogel, Christian Hartman, John Wiley Son Ltd, 2009.

APPENDIX A

Code:

```
#include<reg51.h>                    //including sfr registers for ports of the controller
#include<lcd.h>
#define motor P0                     //LCD Module Connections
sbit RS = P2^0;
sbit EN = P2^1;
sbit D4 = P2^2;
sbit D5 = P2^3;
sbit D6 = P2^4;
sbit D7 = P2^5;
unsigned char y,MT_ARR[]={0x04,0x02,0x08,0x01};
sbit sensor = P1^1;
        sbit buzzer = P1^0;
void send(unsigned char *xx);
 void send1(unsigned char abc);
 int i,jj,k;                          //End LCD Module Connections


void Delay(int a)
{
   int j;
   int i;
   for(i=0;i<a;i++)
   {
      for(j=0;j<112;j++)
      {
      }
   }
}
void main()
{
        motor=0x00;
 TMOD=0x20;
 TH1=0xFd;//9600 BAUD
 SCON=0x50;
        TR1=1;
                buzzer=1;
        Lcd4_init();
        send1('A');
                send1('T');
                send1(13);
```

```
                send1(10);


 Lcd4_Set_Cursor(1,1);
          Lcd4_Write_String("......WAIT......");
        Delay(5000);
        send1('A');
                send1('T');
                send1(13);
                send1(10);
                Delay(1000);
                send1('A');
                send1('T');
                send1(13);
                send1(10);
        Delay(1000);
          Lcd4_Set_Cursor(1,1);
          Lcd4_Write_String("LPG LEAKAGE SYS");
y=1;
                buzzer=1;
 while(1)
 {

if(sensor==1)
 {      Lcd4_Set_Cursor(2,1);
          Lcd4_Write_String("  GAS LEAKAGE  ");

send("AT+CMGS=#");
        send1('"');
                        send("+919805007951#");

                        send1('"');
                        send1(13);
                send1(10);
                Delay(500);
                        send("GAS LEAKAGE#");

                send1(26);
                send1(13);
                send1(10);
buzzer=0;
                                Delay(1000);
                                        buzzer=1;
        for(k=0;k<50;k++)
```

```
                        {
                                for(jj=0;jj<4;jj++)
                                {
                                        motor=MT_ARR[jj ];

                                Delay(100);

                                }

                }
        }
                if(sensor==0)
                {
                                        buzzer=1;
                        motor=0X00;
                                 Lcd4_Set_Cursor(2,1);
         Lcd4_Write_String("  NO  LEAKAGE  ");
                }

 }
}

 void send1(unsigned char abc)
{
SBUF=abc;
while(TI==0);
        TI=0;
 }
void send(unsigned char *xx)
                {
                while(*xx!='#')
                {
                 send1(*xx);
                 xx++;
                 }
                }
```

Header File :

```
                                                    //LCD Module Connections
extern bit RS;
extern bit EN;
extern bit D0;
extern bit D1;
extern bit D2;
extern bit D3;
extern bit D4;
extern bit D5;
extern bit D6;
extern bit D7;
                                                    //End LCD Module Connections


void Lcd_Delay(int a)
{
   int j;
   int i;
   for(i=0;i<a;i++)
   {
      for(j=0;j<112;j++)
      {
      }
   }
}

                                                    //LCD 8 Bit Interfacing Functions
void Lcd8_Port(char a)
{
        if(a & 1)
                D0 = 1;
        else
                D0 = 0;

        if(a & 2)
                D1 = 1;
        else
                D1 = 0;
```

```
        if(a & 4)
                D2 = 1;
        else
                D2 = 0;

        if(a & 8)
                D3 = 1;
        else
                D3 = 0;

        if(a & 16)
                D4 = 1;
        else
                D4 = 0;

        if(a & 32)
                D5 = 1;
        else
                D5 = 0;

        if(a & 64)
                D6 = 1;
        else
                D6 = 0;

        if(a & 128)
                D7 = 1;
        else
                D7 = 0;
}
void Lcd8_Cmd(char a)
{
  RS = 0;                                   // => RS = 0
  Lcd8_Port(a);                             //Data transfer
  EN                                        // => E = 1
  Lcd_Delay(5);
  EN  = 0;                                 // => E = 0
}

Lcd8_Clear()
{
          Lcd8_Cmd(1);
}
```

```
void Lcd8_Set_Cursor(char a, char b)
{
        if(a == 1)
          Lcd8_Cmd(0x80 + b);
        else if(a == 2)
                Lcd8_Cmd(0xC0 + b);
}

void Lcd8_Init()
{
        Lcd8_Port(0x00);
        RS = 0;
        Lcd_Delay(200);

        /*////////////// Reset process from datasheet /////////

 Lcd8_Cmd(0x30);
        Lcd_Delay(50);
 Lcd8_Cmd(0x30);
        Lcd_Delay(110);
 Lcd8_Cmd(0x30);

*/
 ////////////////////////////////////////////////

 Lcd8_Cmd(0x38);    //function set
 Lcd8_Cmd(0x0C);    //display on,cursor off,blink off
 Lcd8_Cmd(0x01);    //clear display

}

void Lcd8_Write_Char(char a)
{
   RS = 1;           // => RS = 1
   Lcd8_Port(a);           //Data transfer
   EN  = 1;          // => E = 1
   Lcd_Delay(5);
   EN  = 0;          // => E = 04
}

void Lcd8_Write_String(char *a)
{
        int i;
        for(i=0;a[i]!='\0';i++)
```

```
        Lcd8_Write_Char(a[i]);
}

void Lcd8_Shift_Right()
{
        Lcd8_Cmd(0x1C);
}

void Lcd8_Shift_Left()
{
        Lcd8_Cmd(0x18);
}
//End LCD 8 Bit Interfacing Functions

//LCD 4 Bit Interfacing Functions

void Lcd4_Port(char a)
{
        if(a & 1)
                D4 = 1;
        else
                D4 = 0;

        if(a & 2)
                D5 = 1;
        else
                D5 = 0;

        if(a & 4)
                D6 = 1;
        else
                D6 = 0;

        if(a & 8)
                D7 = 1;
        else
                D7 = 0;
}
void Lcd4_Cmd(char a)
{
        RS = 0;          // => RS = 0
        Lcd4_Port(a);
        EN  = 1;          // => E = 1
  Lcd_Delay(5);
```

```
  EN  = 0;           // => E = 0
}

Lcd4_Clear()
{
        Lcd4_Cmd(0);
        Lcd4_Cmd(1);
}

void Lcd4_Set_Cursor(char a, char b)
{
        char temp,z,y;
        if(a == 1)
        {
          temp = 0x80 + b;
                z = temp>>4;
                y = (0x80+b) & 0x0F;
                Lcd4_Cmd(z);
                Lcd4_Cmd(y);
        }
        else if(a == 2)
        {
                temp = 0xC0 + b;
                z = temp>>4;
                y = (0xC0+b) & 0x0F;
                Lcd4_Cmd(z);
                Lcd4_Cmd(y);
        }
}

void Lcd4_Init()
{
        Lcd4_Port(0x00);
        Lcd_Delay(200);

        /////////////// Reset process from datasheet /////////

  Lcd4_Cmd(0x03);
        Lcd_Delay(50);
  Lcd4_Cmd(0x03);
        Lcd_Delay(110);
  Lcd4_Cmd(0x03);
  //////////////////////////////////////////////////
  Lcd4_Cmd(0x02);
```

```
        Lcd4_Cmd(0x02);
  Lcd4_Cmd(0x08);
        Lcd4_Cmd(0x00);
        Lcd4_Cmd(0x0C);
  Lcd4_Cmd(0x00);
  Lcd4_Cmd(0x06);
}

void Lcd4_Write_Char(char a)
{
   char temp,y;
   temp = a&0x0F;
   y = a&0xF0;
        RS = 1;                 // => RS = 1
   Lcd4_Port(y>>4);           //Data transfer
        EN = 1;
        Lcd_Delay(5);
        EN = 0;
        Lcd4_Port(temp);
        EN = 1;
        Lcd_Delay(5);
        EN = 0;
}

void Lcd4_Write_String(char *a)
{
        int i;
        for(i=0;a[i]!='\0';i++)
         Lcd4_Write_Char(a[i]);
}

void Lcd4_Shift_Right()
{
        Lcd4_Cmd(0x01);
        Lcd4_Cmd(0x0C);
}

void Lcd4_Shift_Left()
{
        Lcd4_Cmd(0x01);
        Lcd4_Cmd(0x08);
}
//End LCD 4 Bit Interfacing Functions
```