REPRESENTATION OF BRAILLE PATTERN USING THERMAL RESISTANCE NETWORK

Project Report submitted in partial fulfilment of the requirement for the degree of

Bachelor of Technology

in

Electronics and Communication Engineering

under the supervision of

Mr. Pardeep Garg

By

Kshitij Sachdeva-101078

Amanjyot Singh Kalra-101080

Sarvleen Singh Lamba-101105

To

May-2014

Jaypee University of Information and Technology Waknaghat,
Solan, 173234, Himachal Pradesh

TABLE OF CONTENTS

Chapter No. **Topics** **Page No.**

CERTIFICATE

This is to certify that project report entitled "**REPRESENTATION OF BRAILLE PATTERN USING THERMAL RESISTANCE NETWORK**", submitted by **Kshitij Sachdeva (101078), Amanjyot Singh Kalra (101080) and Sarvleen Singh Lamba (101105)** in partial fulfillment for the award of degree of Bachelor of Technology in Electronics and Communication Engineering to Jaypee University of Information Technology, Waknaghat, and 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:

Supervisor's Name: Mr. Pardeep Garg

Designation: Assistant Prof. (Grade-II)

Signature:

ACKNOWLEDGEMENT

We owe a great many thanks to a great many people who have helped and supported us during this project. Our deepest thanks to **Mr. Pardeep Garg**, our project guide for his exemplary guidance, monitoring and constant encouragement throughout the course of this project work. He has taken great pains to go through the project and make necessary corrections as and whenever needed. We are also grateful to **Mr. Mohan Sharma (ECE Project lab**) for his practical help and guidance. We would also like to thank all the faculty members of ECE department without whom the progress of this project would have been a distant reality. We also extend our heartfelt thanks to our family and well-wishers.

Date:

Name of the students:

Kshitij Sachdeva (101078)

Amanjyot Singh Kalra (101080)

Sarvleen Singh Lamba (101105)

LIST OF FIGURES

LIST OF SYMBOLS AND ACRONYMS

- Op-amp – operational amplifier
- SCU - signal control unit
- LCD - liquid crystal display
- LED - light emitting diode
- XTAL - crystal oscillator
- Vout - output voltage
- Vin - input voltage
- Vcc - power supply
- mV - millivolts
- V - volts
- V + - voltage at positive terminal
- GND – ground(zero potential)
- R - resistance
- C - capacitance
- W - watts
- AC - alternating current
- DC - direct current
- uA - microamperes

- pF – pico-farad
- I/O - input/output
- Rx - receiver
- Tx - transmitter
- mfd - microfarad
- kΩ - kilo ohms
- sw – switch
- CTM – compact thermal model
- TIM – thermal interface material

ABSTRACT

Braille is a tactical writing system that enables a blind person to read, write, learn, participate and communicate. This system is credited to Louis Braille.

Many efforts are being put to improve, enhance and innovate in this field to help the visually impaired. Our project is neither an innovation nor an enhancement in the field; rather a cost effective teaching aid. In our project we are basically trying to make the Braille pattern palpable; we will also try to represent different patterns which could be sensed by the blind and imagined by them.

For example, suppose we pattern a square; with the help of an 8X8 matrix of resistors, we use their thermal properties to enable the blind person to touch and sense what actually a square might look like. Similarly other patterns can also be represented by using this method; this is a prototype for developing more complex figures and graphs. It is an **embedded-system based project** in which a Microcontroller, active-passive components and various IC's are employed.

The first phase of our project includes a Motor circuit which would represent the Braille pattern and would be tangible, with the specific motors vibrating according to the required pattern. The next phase includes the thermal network of resistors which heat up according to the patterns we burn into the microcontroller.

The LCD would show us the alphabet or word we are demonstrating. And a new feature we have added to our project is the LED panel, which will show shapes (geometric figures like a "square" or a "triangle")!

Signature of students

Names:

Date:

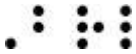
Signature of supervisor

Name:

Date:

CHAPTER 1

INTRODUCTION

1.1 Braille 

Braille is writing system which enables blind and partially sighted people to read and write through touch. It was invented by Louis Braille (1809-1852), a French teacher of the blind. It consists of patterns of raised dots arranged in cells of up to six dots in a 3 x 2 configuration. Each cell represents a letter, numeral or punctuation mark. Some frequently used words and letter combinations also have their own single cell patterns. [1]

There are a number of different versions of Braille:

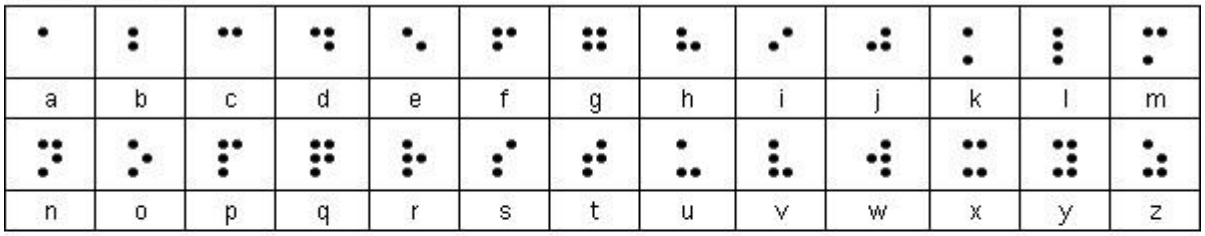
- **Grade 1**, which consists of the 26 standard letters of the alphabet and punctuation. It is only used by people who are first starting to read Braille.

- **Grade 2**, which consists of the 26 standard letters of the alphabet, punctuation and contractions. The contractions are employed to save space because a Braille page cannot fit as much text as a standard printed page. Books, signs in public places, menus, and most other Braille materials are written in Grade 2 Braille.

- **Grade 3**, which is used only in personal letters, diaries, and notes. It is a kind of shorthand, with entire words shortened to a few letters.

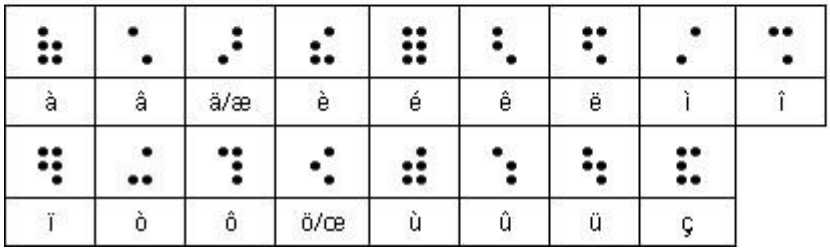
Braille has been adapted to write many different languages, including Chinese, and is also used for musical and mathematical notation.

Basic letters



a	b	c	d	e	f	g	h	i	j	k	l	m
n	o	p	q	r	s	t	u	v	w	x	y	z

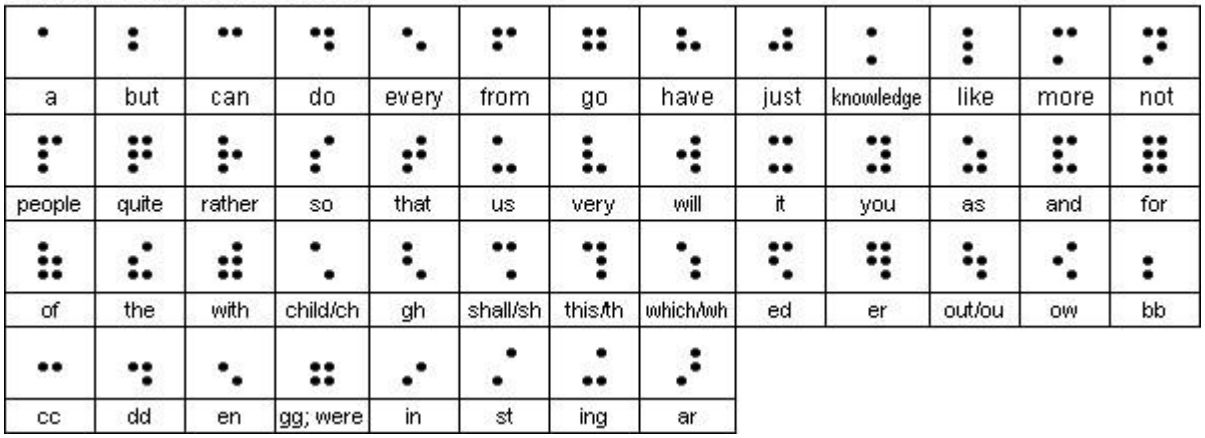
Accented letters



à	â	ä/æ	è	é	ê	ë	ì	î
ï	ò	ô	ö/œ	ù	û	ü	ç	

Figure 1: Braille Letters [1]

Words and abreviations



a	but	can	do	every	from	go	have	just	knowledge	like	more	not
people	quite	rather	so	that	us	very	will	it	you	as	and	for
of	the	with	child/ch	gh	shall/sh	this/th	which/wh	ed	er	out/ou	ow	bb
cc	dd	en	gg; were	in	st	ing	ar					

Figure 2: Braille Words [2]

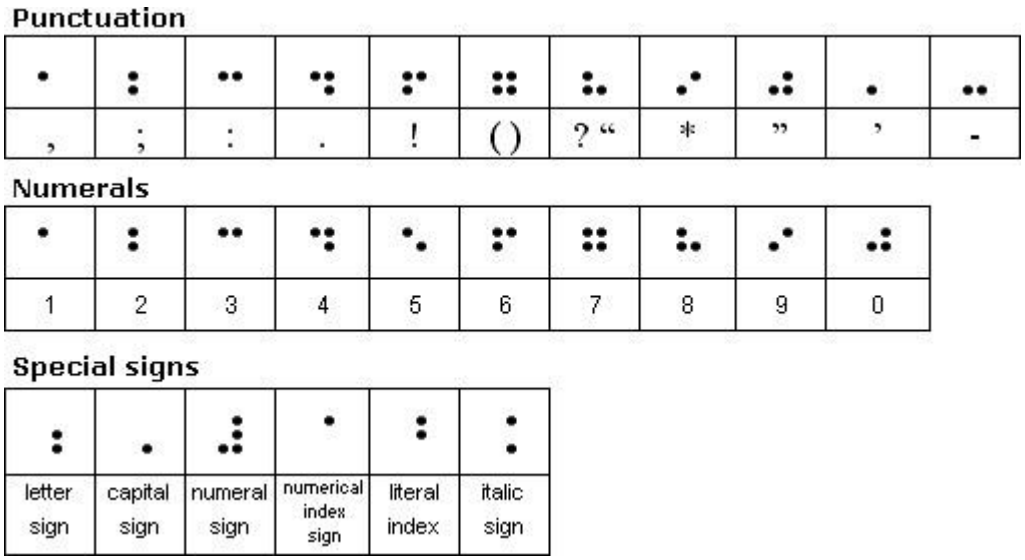


Figure 3: Punctuations and Symbols [3]

The sample texts in Braille are as follows, the second text is the Article #1 of the Universal Declaration of Human Rights:

Figure 4: Transliteration: "Be kind to others" [4]

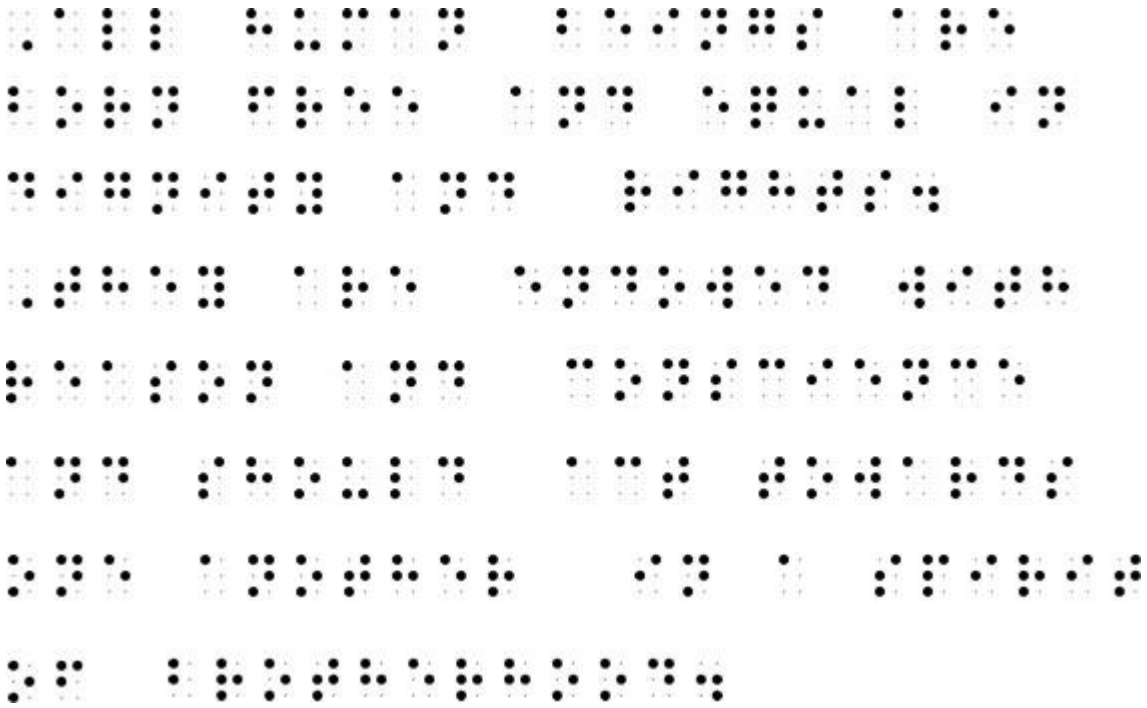


Figure 5: Article in Braille [4]

Transliteration of the text:

“All human beings are born free and equal in dignity and rights. They are endowed with reason and conscience and should act towards one another in a spirit of brotherhood.”

This project requires the following hardware components:

- **AT89C52**
- **ULN2803AG**
- **DC MOTOR RF-130**
- **L7812**
- **CD7805**

The purpose of this project and its benefits can be listed as follows:

- To make an ordinary person without any knowledge of Braille communicate or teach alphabets and words to a Blind Person.
- It can be applied in the medical industry
- It can be used as a demonstrator for a teaching utility.

CHAPTER 2

POWER SUPPLY

2.1 Regulated Power Supply:

Power supplies are designed to convert high voltage AC mains to a suitable low voltage supply for electronic circuits and other devices. A power supply can be broken down into a series of blocks, each of which performs a particular function.

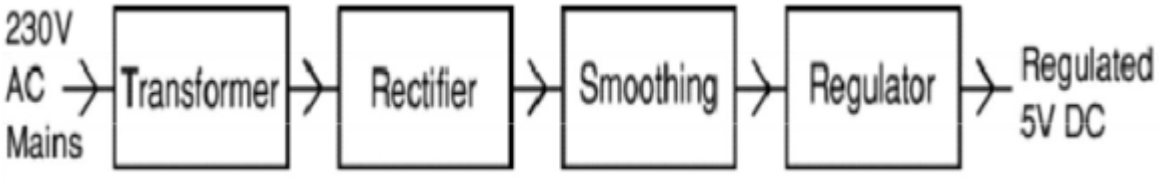


Figure 6: Power supply

Each of the blocks has its own function as described below:

- **Transformer** – steps down high voltage AC mains to low voltage AC.
- **Rectifier** – converts AC to DC, but the DC output is varying.
- **Smoothing** – smoothens the DC from varying greatly to a small ripple.
- **Regulator** – eliminates ripple by setting DC output to a fixed voltage.

2.2 Generation of varying and smooth DC

The varying DC is generated using a transformer and a rectifier and in order to generate a smooth DC a smoothening capacitor is added to the circuit as shown below.

Transformer + Rectifier + Smoothing + Regulator → Varying and smooth DC

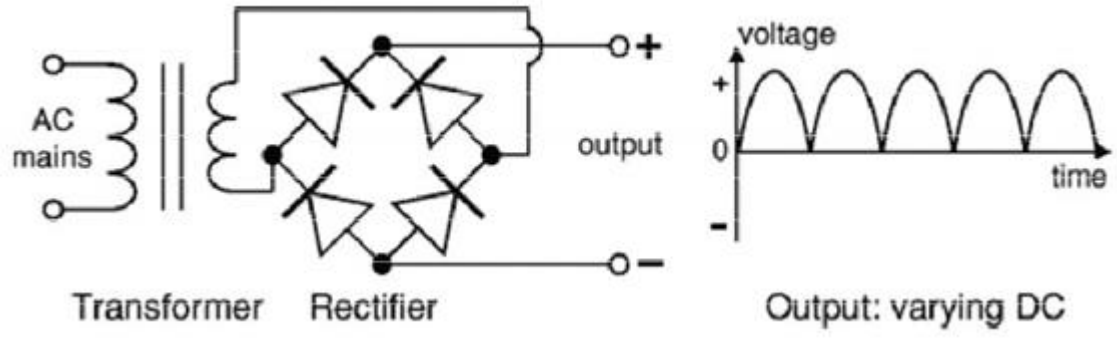


Figure 7: Generation of Varying DC [4]

The low voltage AC output is suitable for lamps, heaters and special AC motors. It is not suitable for electronic circuits unless they include a rectifier and a smoothing capacitor.

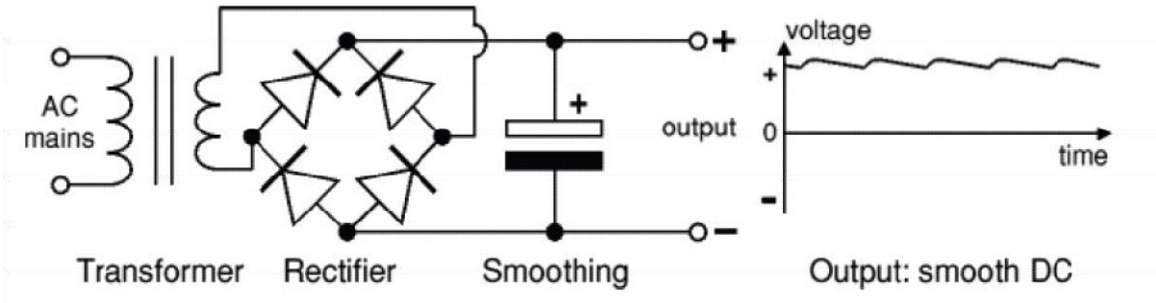


Figure 8: Generation of smooth DC [5]

The smooth DC output has a small ripple. It is suitable for most electronic circuits.

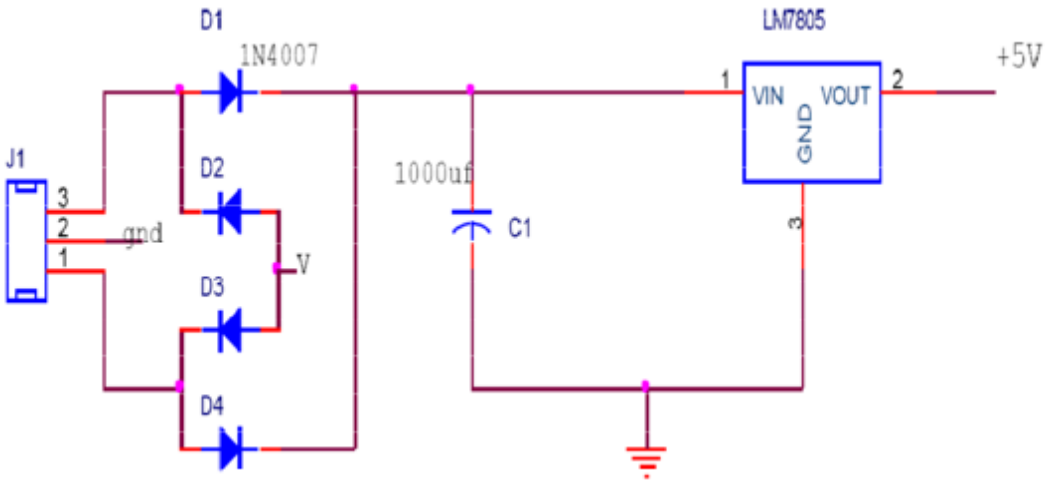


Figure 9: Circuit diagram of regulated power supply [6]

CHAPTER 3

MICROCONTROLLER UNIT

The AT89C52 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 AT89C52 is a powerful microcomputer which provides a highly-flexible and cost-effective solution to many embedded control applications [6].

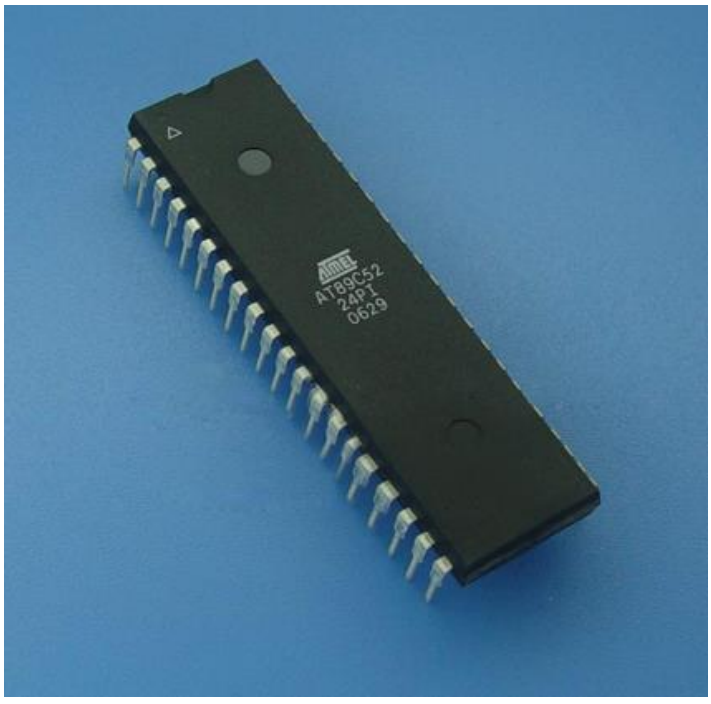


Figure 10: 89C52 [6]

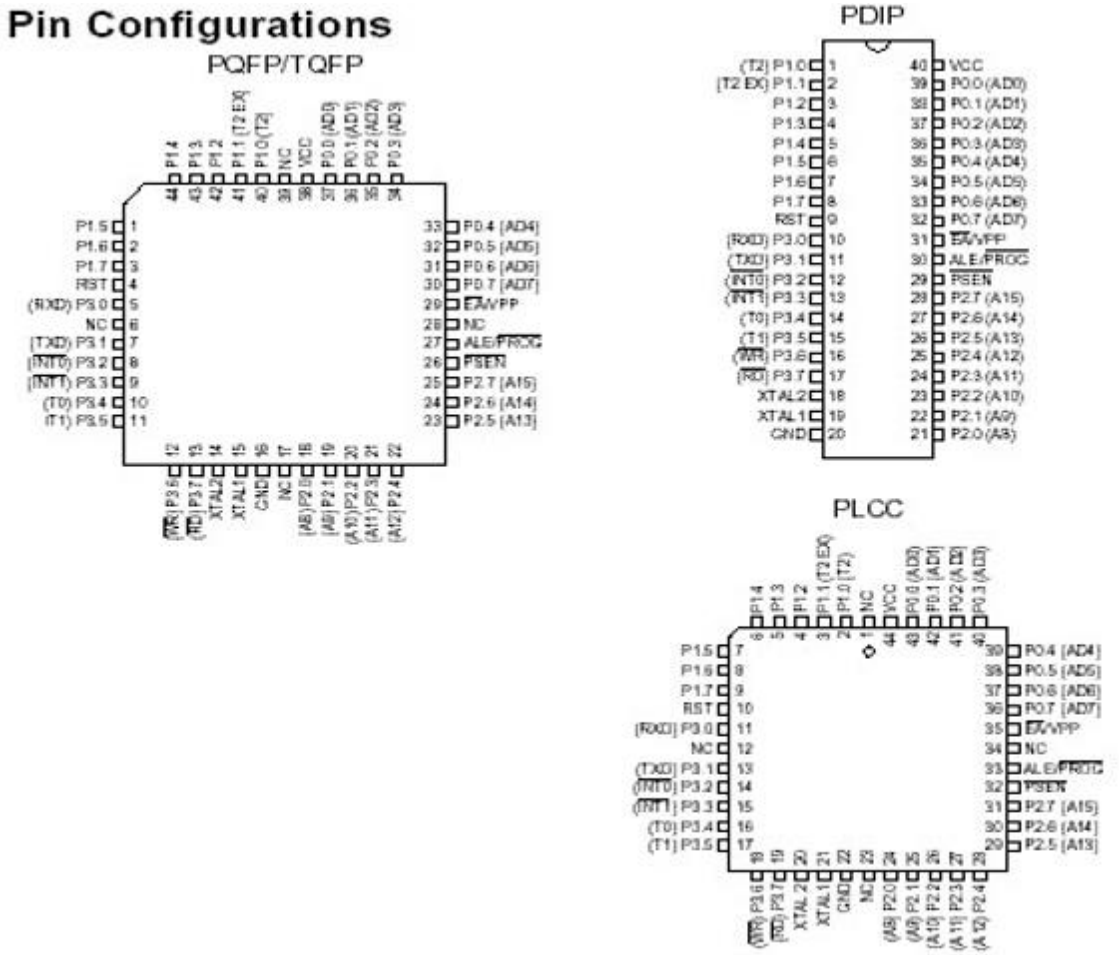


Figure 11: 89C52 Pin Description [6]

3.1 8-bit Microcontroller

Features of the 8-bit microcontroller with 8k Bytes Flash:

- Compatible with MCS-51™ Products
- 8K Bytes of In-System Reprogrammable Flash Memory
- Endurance: 1,000 Write/Erase Cycles
- Fully Static Operation: 0 Hz to 24 MHz
- Three-level Program Memory Lock

- 256 x 8-bit Internal RAM
- 32 Programmable I/O Lines
- Three 16-bit Timer/Counters
- Eight Interrupt Sources
- Programmable Serial Channel
- Low-power Idle and Power-down Modes

3.2 Pin description

The Pins of the 89C52 are described as follows:

VCC: Supply voltage.

GND: Ground.

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 can also be configured to be the multiplexed low order address/data bus during accesses to external program and data memory. In this mode, P0 has internal pull-ups. It also receives the code bytes during Flash programming and outputs the code bytes during program verification. External pull-ups are required during program verification.

Port 1: Port 1 is an 8-bit bi-directional I/O port with internal pull-ups. The Port 1 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 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 trigger input (P1.1/T2EX), respectively, as shown in the following table.
Port 1 also receives the low-order address bytes during Flash programming and verification.

Port Pin	Alternate Functions
P1,0	T2(External count input to timer)clock out
P1,1	T2EX(Timer reload trigger & direction control)

Table 1: Pin 1 description

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

Port 2 emits the high-order address byte during fetches from external program memory and during accesses to external data memory that uses 16-bit addresses. In this application, Port 2 uses strong internal pull-ups when emitting 1s. During accesses to external data memory that use 8-bit addresses Port 2 emits the contents of the P2 Special Function Register. Port 2 also receives the high-order address bits and some control signals during Flash programming and verification.

Port 3: Port 3 is an 8-bit bi-directional I/O port with internal pull-ups. The Port 3 output buffers can sink/source four TTL inputs. When 1s are written to Port 3 pins, they are pulled high by the internal pull-ups and can be used as inputs. As inputs, Port 3 pins that are externally being pulled low will source current because of the pull-ups.

Port 3 also serves the functions of various special features of the AT89C51, as shown in the following table. Port 3 also receives some control signals for Flash programming and verification.

Port Pin	Alternate Functions
P3,0	RXD(Serial Input Port)
P3,1	TXD(Serial Output Port)
P3,2	INT 0(External Interrupt 0)
P3,3	INT 1(External Interrupt 1)
P3,4	T0(Timer 0 External Input)
P3,5	T1(Timer 1External Input)
P3,6	WR(Write Strobe)
P3,7	RD(Read Strobe)

Table 2: Pin 3 description

RST: Reset input. A high on this pin for two machine cycles while the oscillator is running resets the device.

ALE/PROG: Address Latch Enable is an output pulse for latching the low byte of the address during accesses to external memory.

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 clocking purposes. Note, however, that one ALE pulse is skipped during each access to external data memory.

If desired, ALE operation can be disabled by setting bit 0 of SFR location 8EH. With the bit set, ALE is active only during a MOVX or MOVC instruction. Otherwise, the pin is weakly pulled high. Setting the ALE-disable bit has no effect if the microcontroller is in external execution mode.

PSEN: Program Store Enable is the read strobe to external program memory. When the AT89C52 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.

EA/VPP: External Access Enable. EA must be strapped to GND in order to enable the device to fetch code from external program memory locations starting at 0000H up to FFFFH.

Note, however, that if lock bit 1 is programmed, EA will be internally latched on reset. EA should be strapped to VCC for internal program executions. This pin also receives the 12-volt programming enable voltage (VPP) during Flash programming when 12-volt programming is selected.

XTAL1: Input to the inverting oscillator amplifier and input to the internal clock operating circuit.
XTAL2: Output from the inverting oscillator amplifier.

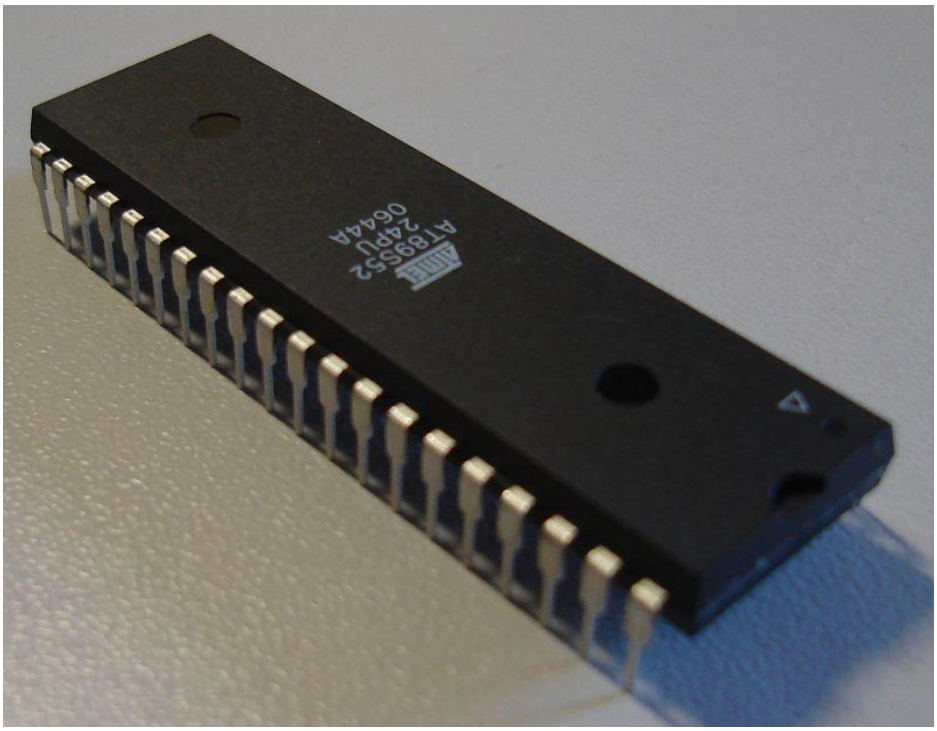


Figure 12: Atmel 89C52/89S52 [7]

3.3 Oscillator Characteristics:

XTAL1 and XTAL2 are the input and output, respectively, of an inverting amplifier which can be configured for use as an on-chip oscillator, as shown in Figure. Either a quartz crystal or ceramic resonator may be used. To drive the device from an external clock source, XTAL2 should be left unconnected while XTAL1 is driven as shown in Figure.

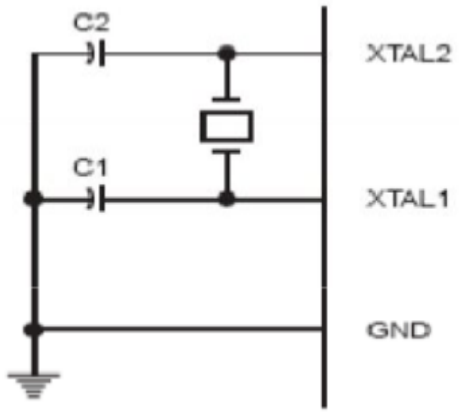


Figure 13: Oscillator Connections [6]

It must be noted that there are various speeds of the 8051 family. Speed refers to the maximum oscillator frequency connected to the XTAL. For example, a 12 MHz chip must be connected to a crystal with 12 MHz frequency or less. Likewise, a 20 MHz microcontroller requires a crystal frequency of no more than 20 MHZ. When the 8051 is connected to a crystal oscillator and is powered up, we can observe the frequency on the XTAL2 pin using oscilloscope.

CHAPTER 4

Darlington Driver (ULN2803A)

4.1 Introduction

IC ULN2803 consists of octal high voltage, high current transistor arrays. The eight NPN Darlington connected transistors in this family of arrays are ideally suited for interfacing between low logic level digital circuitry (such as TTL, CMOS or PMOS/NMOS) and the higher current/voltage requirements of lamps, relays, printer hammers or other similar loads for a broad range of computer, industrial, and consumer applications [8].

4.2 Features

- Eight darlingtons with common emitters.
- Output current to 500 mA.
- Output voltage to 50V.
- Integral suppression diodes.
- Versions for all popular logic families.
- Output can be paralleled.
- Inputs pinned opposite outputs to simplify board layout.

4.3 Description

The ULN2803A contain eight darlington transistors with common emitters and integral suppression diodes for inductive loads. Each darlington features a peak load current rating of 600mA (500mA 21capability.

Figure 14: ULN2803A [9]

Five versions are available to simplify interfacing standard logic families: the ULN2803A has a 2.7kΩ input resistor for 5V TTL and CMOS and designed to sink a minimum of 350mA for standard and Schottky TTL where higher output current is required.This is supplied in a 18-lead plastic DIP with a copper lead from and feature the convenient input-opposite-output pin-out to simplify board layout.

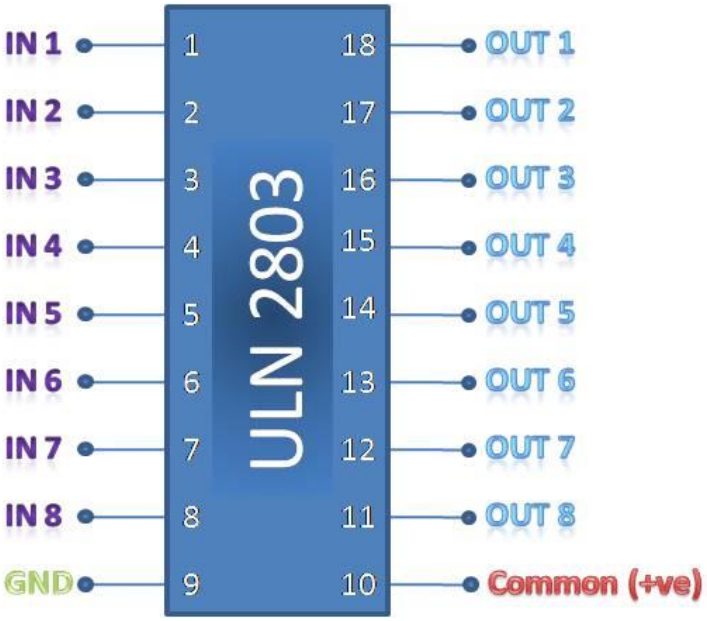


Figure 15: Pin description [10]

CHAPTER 5

DISPLAY DEVICE (LCD)

5.1 LCD pin description

Liquid crystal displays (LCD) are widely used in recent years as compares to LEDs. It has the ability to display numbers, characters and graphics. The LCD, which is used as a display in the system, is LMB162A. [11]

The main features of this LCD are:

- 16 X 2 display intelligent LCD, used for alphanumeric characters & based on ASCII codes.
- This LCD contains 16 pins, in which 8 pins are used as 8-bit data I/O.
- Three pins are used as control lines these are Read/Write pin, Enable pin and Register select pin.
- Two pins are used for Backlight and LCD voltage, another two pins are for Backlight & LCD ground and one pin is used for contrast change.

Figure 16: Liquid Crystal Display [11]

5.2 LCD pin description

The LCD discuss in this section has the most common connector used for the Hitachi 44780 based LCD is 14 pins in a row and modes of operation and how to program and interface with microcontroller is describes in this section.

The voltage VCC and VSS provided by +5V and ground respectively while VEE is used for controlling LCD contrast. Variable voltage between Ground and Vcc is used to specify the contrast (or "darkness") of the characters on the LCD screen.

The pins of the LCD are as follows:

RS (register select)

There are two important registers inside the LCD. The RS pin is used for their selection as follows. If RS=0, the instruction command code register is selected, then allowing to user to send a command such as clear display, cursor at home etc. If RS=1, the data register is selected, allowing the user to send data to be displayed on the LCD.

R/W (read/write)

The R/W (read/write) input allowing the user to write information from it. R/W=1, when it read and R/W=0, when it writing.

EN (enable)

The enable pin is used by the LCD to latch information presented to its data pins. When data is supplied to data pins, a high power, a high-to-low pulse must be applied to this pin in order to for the LCD to latch in the data presented at the data pins.

D0-D7 (data lines)

The 8-bit data pins, D0-D7, are used to send information to the LCD or read the contents of the LCD‟s internal registers. To displays the letters and numbers, we send ASCII codes for the letters A-Z, a-z,

and numbers 0-9 to these pins while making RS =1. There are also command codes that can be sent to clear the display or force the cursor to the home position or blink the cursor. We also use RS =0 to check the busy flag bit to see if the LCD is ready to receive the information. The busy flag is D7 and can be read when R/W =1 and RS =0, as follows: if R/W =1 and RS =0, when D7 =1(busy flag =1), the LCD is busy taking care of internal operations and will not accept any information. When D7 =0, the LCD is ready to receive new information.

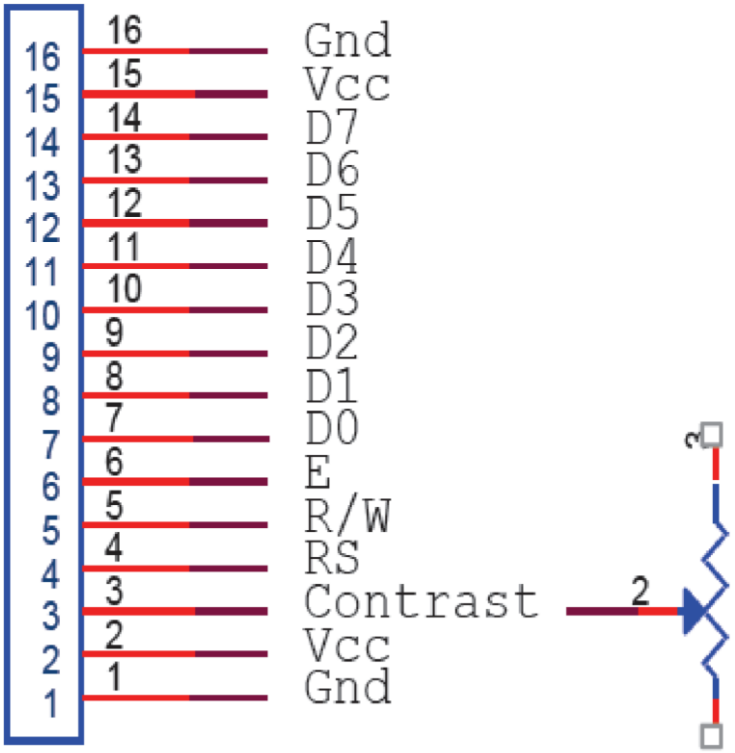


Figure 17: Pin Diagram of LCD

5.3 Interfacing of micro controller with LCD display

In most applications, the "R/W" line is grounded. This simplifies the application because when data is read back, the microcontroller I/O pins have to be alternated between input and output modes. In this case, "R/W" to ground and just wait the maximum amount of time for each instruction (4.1ms for clearing the display or moving the cursor/display to the "home position", 160µs for all other commands) and also the application software is simpler, it also frees up a microcontroller pin for other uses. Different LCD execute instructions at different rates and to avoid problems later on (such as if the LCD is changed to a slower unit). Before sending commands or data to the LCD module, the Module must be initialized. Once the initialization is complete, the LCD can be written to with data or instructions as required. Each character to display is written like the control bytes, except that the "RS" line is set. During initialization, by setting the "S/C" bit during the "Move Cursor/Shift Display" command, after each character is sent to the LCD, the cursor built into the LCD will increment to the next position (either right or left). Normally, the "S/C" bit is set (equal to "1")

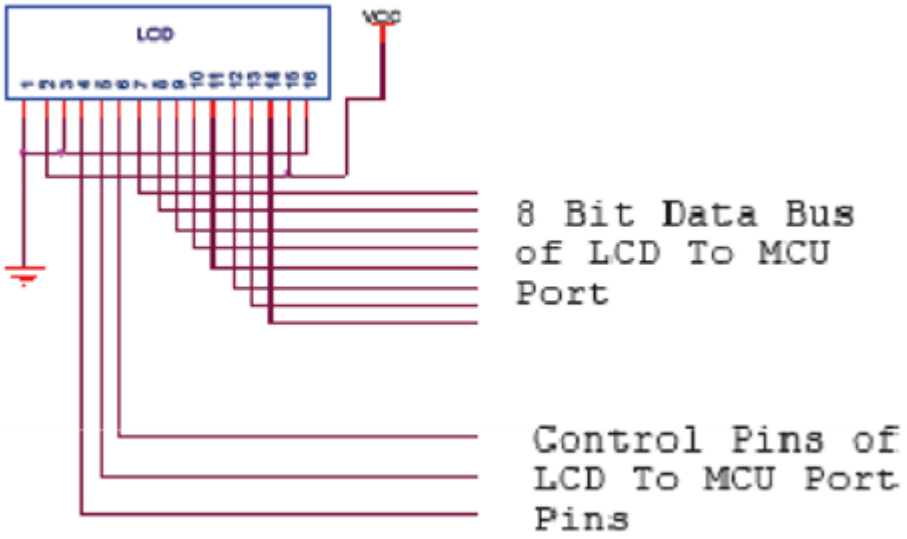


Figure 18: Interfacing of Microcontroller with LCD [11]

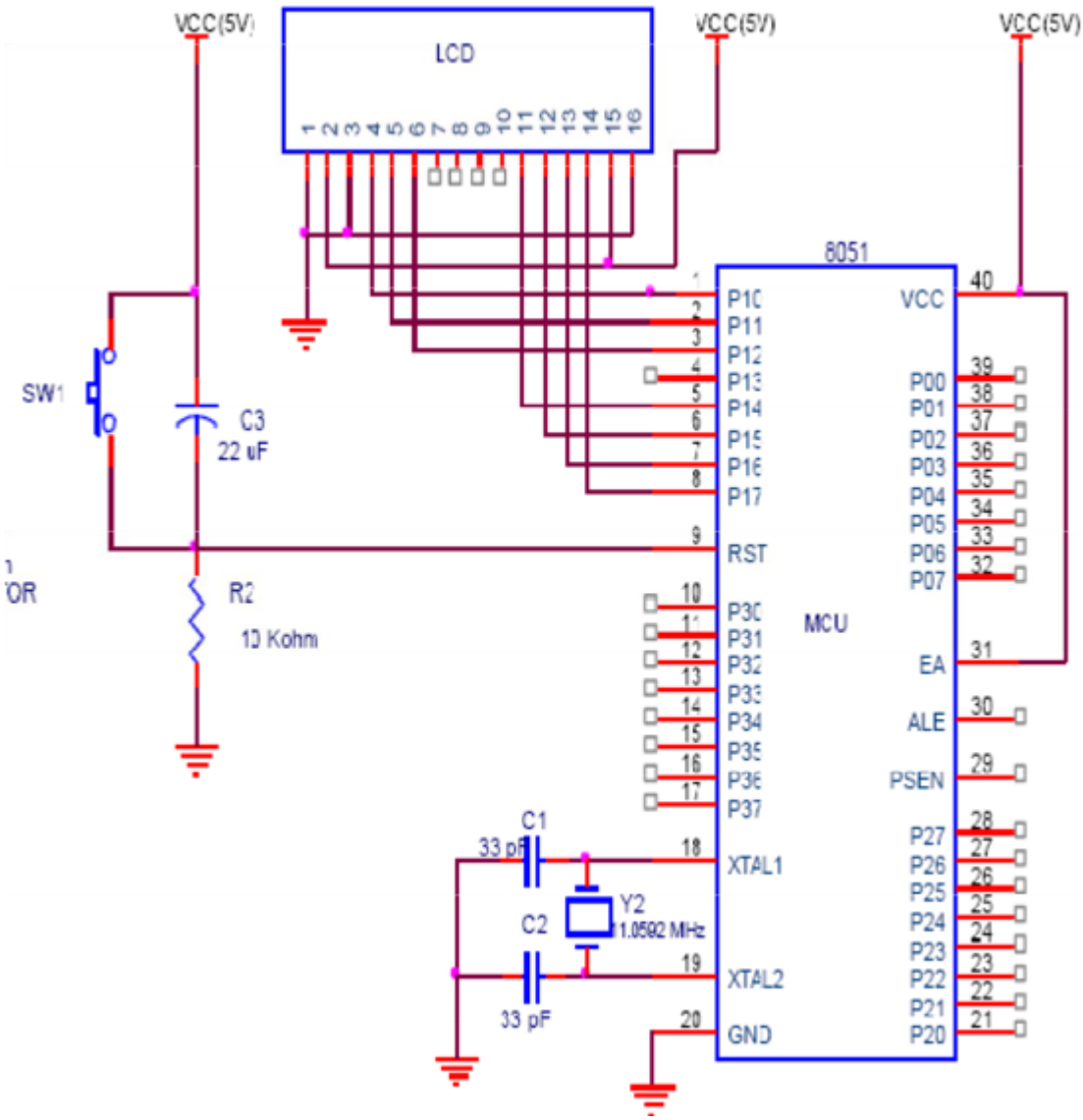


Figure 19: LCD attached to the Microcontroller unit [11]

CHAPTER 6

DISPLAY DEVICE (LED)

6.1 Introduction

A **light-emitting diode** (**LED**) is a two-lead semiconductor light source that resembles a basic pn-junction diode, except that an LED also emits light When an LED's anode lead has a voltage that is more positive than its cathode lead by at least the LED's forward voltage drop, current flows. Electrons are able to recombine with holes within the device, releasing energy in the form of photons. This effect is called electroluminescence, and the color of the light (corresponding to the energy of the photon) is determined by the energy band gap of the semiconductor.

An LED is often small in area (less than 1 mm^2), and integrated optical components may be used to shape its radiation pattern

Appearing as practical electronic components in 1962 the earliest LEDs emitted low-intensity infrared light. Infrared LEDs are still frequently used as transmitting elements in remote-control circuits, such as those in remote controls for a wide variety of consumer electronics. The first visible-light LEDs were also of low intensity, and limited to red. Modern LEDs are available across the visible, ultraviolet, and infrared wavelengths, with very high brightness.

Early LEDs were often used as indicator lamps for electronic devices, replacing small incandescent bulbs. They were soon packaged into numeric readouts in the form of seven-segment displays, and were commonly seen in digital clocks.

Recent developments in LEDs permit them to be used in environmental and task lighting. LEDs have many advantages over incandescent light sources including lower energy consumption, longer lifetime, improved physical robustness, smaller size, and faster switching. Light-emitting diodes are now used 38in applications as diverse as aviation lighting, automotive headlamps, advertising, general lighting, traffic signals, and camera flashes. However, LEDs powerful enough for room lighting are still relatively expensive, and require more precise current and heat management than compact fluorescent lamp sources of comparable output.

LEDs have allowed new text, video displays, and sensors to be developed, while their high switching rates are also useful in advanced communications technology.

6.2 LED pin description

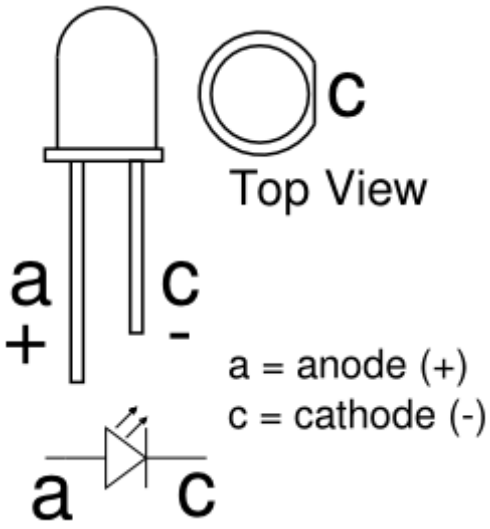


Figure 20: LED [12]

1. Looking down from the top of the LED there is usually a flattened edge, this identifies the negative pin of the LED.
2. If the LED is new, there will be one lead longer than the other. The long lead is the positive pin and the short lead will be the negative pin.
3. If you hold the LED sideways on and view through the plastic, there will be two parts, a small straight pin and a fatter 'L' shaped pin. The small pin is the positive side and the fatter 'L' shaped pin is the negative side.

6.3 LED 8X8 multiplexing

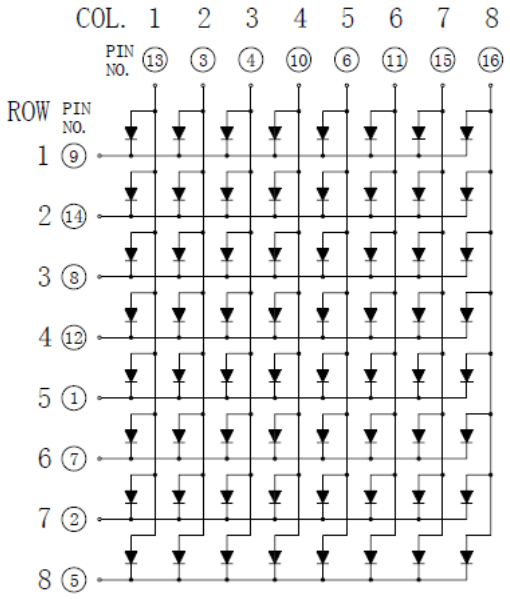


Figure 21: 8X8 Multiplexing [12]

Dot matrix units typically come in either a 5x7 or 8x8 matrix of LEDs. The LEDs are wired in the matrix such that either the anode or cathode of each LED is common in each row. In other words, in a common anode LED dot matrix unit, each row of LEDs would have all of their anodes in that row wired together. The cathodes of the LEDs would all be wired together in each column. A typical single color 8x8 dot matrix unit will have 16 pins, 8 for each row and 8 for each column. The reason the rows and columns are all wired together is to minimize the number of pins required. If this were not the case, a single color 8x8 dot matrix unit would need 65 pins, one for each LED and a common anode or cathode connector. By wiring the rows and columns together, only 16 pins are required.

However, this now poses a problem if you want a particular LED to light in a certain position. If, for example, you had a common anode unit and wanted to light the LED at X, Y position 5, 3 (5th column, 3rd row), then you would apply a current to the 3rd Row and ground the 5th column pin. The LED in the 5th column and 3rd row would now light.

Now let's imagine that you want to also light the LED at column 3, row 6. So you apply a current to the 6th row and ground the 3rd column pin. The LED at column 3, row 6 now illuminates. But, the LEDs at column 3, row 6 and column 5, row 6 have also lit up.

This is because you are applying power to row 3 and 6 and grounding columns 3 and 5. You can't turn off the unwanted LEDs without turning off the ones you want on. It would appear that there is no way you can light just the two required LEDs with the rows and columns wired together as they are. The only way this would work would be to have a separate pin-out for each LED, meaning the number of pins would jump from 16 to 65. A 65-pin dot matrix unit would be very hard to wire up and control because you'd need a microcontroller with at least 64 digital outputs.

Multiplexing is the technique of switching one row of the display on at a time. By selecting the column that contains the row that contains the LED that you want to be lit, and then turning the power to that row on (or the other way round for common cathode displays), the chosen LEDs in that row will illuminate. That row is then turned off and the next row is turned on, again with the appropriate columns chosen and the LEDs in the second row will now illuminate. Repeat with each row till you get to the bottom and then start again at the top. If this is done fast enough (at more than 100Hz, or 100 times per second) then the phenomenon of persistence of vision (where an afterimage remains on the retina for 1/25th of a second) will mean that the display will appear to be steady, even though each row is turned on and off in sequence.

By using this technique, we get around the problem of displaying individual LEDs without the other LEDs in the same column or row also being lit. By scanning down the rows and illuminating the respective LEDs in each column of that row and doing this very fast (more than 100Hz) the human eye will perceive the image as steady [13].

CHAPTER 7

METHODOLOGY

7.1 Introduction

Our project, Representation of Braille Pattern using Thermal Resistance Network was accomplished by bringing different modules together, the various modules are shown in the polygon diagram!

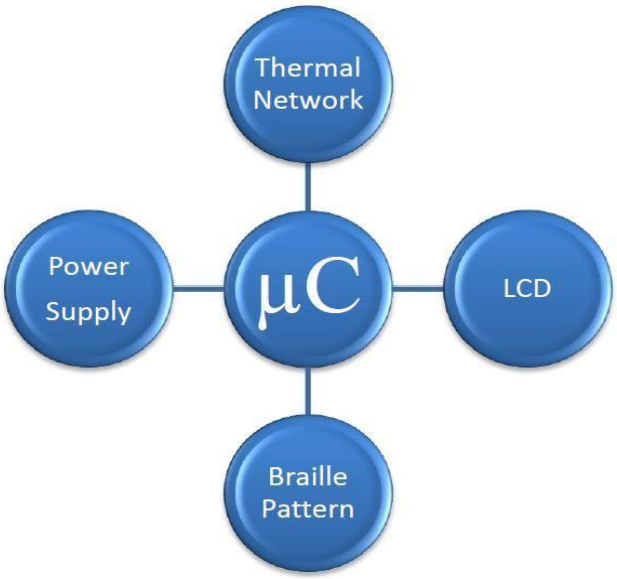


Figure 22: Block Diagram for the Project

We divided our project into the following modules and finally assembled all the hardware:

a) **Power Supply**:

The Power Circuit is made up of a bridge rectifier, Voltage regulators and other active passive components. The components we are going to use in the circuit include voltage of 5Vand 12V so voltage regulator 7805 and 7812 are used. One can get a constant high voltage power supply using inexpensive 3-terminal voltage regulators through some simple techniques described below.

Depending upon the current requirement, a reasonable load regulation can be achieved. Line regulation in all cases is equal to that of the voltage regulator used.

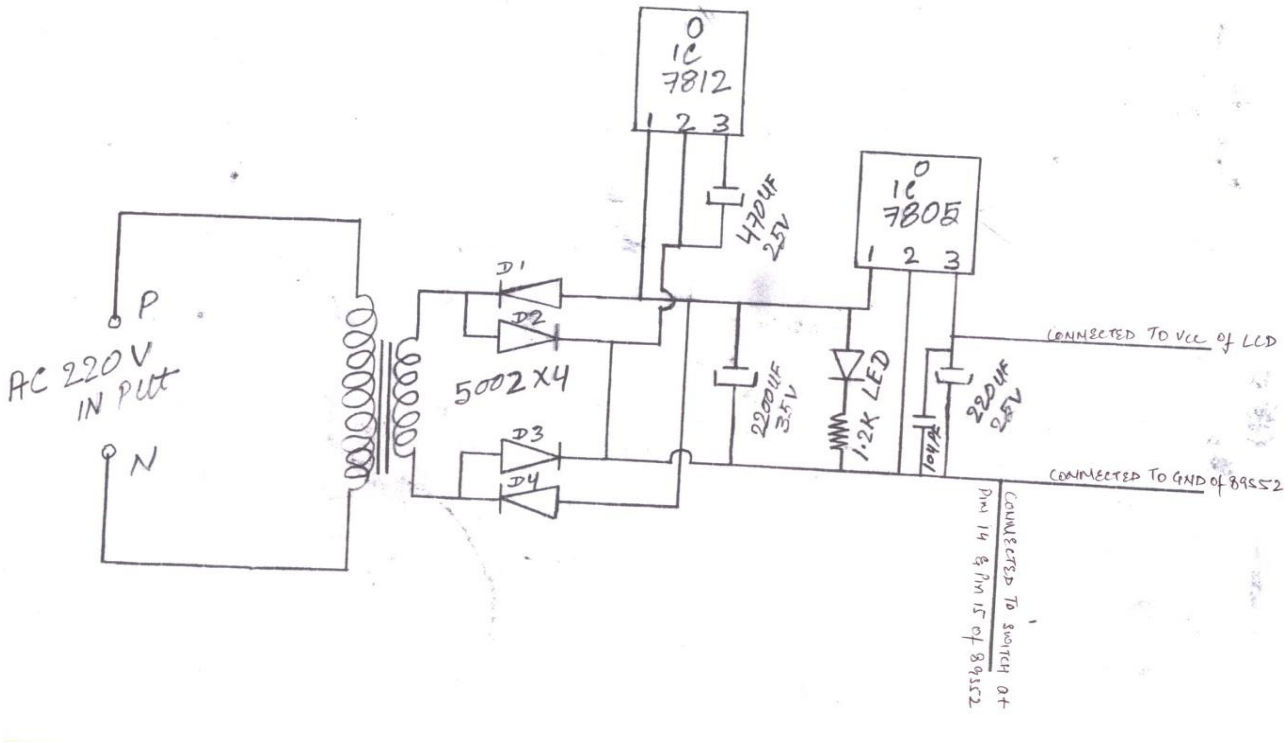


Figure 23: Power Supply

Though high voltage can be obtained with suitable voltage boost circuitry using ICs like LM 723, some advantages of the circuits presented below are: simplicity, low cost, and practically reasonable regulation characteristics. For currents of the order of 1A or less, only one Zener and some resistors and capacitors are needed. [5] For higher currents, one pass transistor such as ECP055 is needed. Before developing the final circuits let us see the schematic in the figure where 78XX is a 3-terminal voltage regulator.

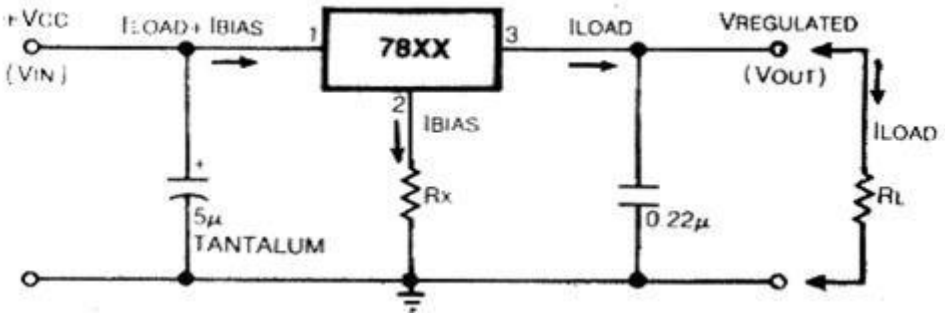


Figure 24: Power Supply using 78XX [5]

b) **LCD interfacing with microcontroller**: LCD interfacing is mainly done to represent data which is used by the developers to know what output or input has been given. In our scenario, we have displayed the alphabet to be represented with the motor circuit. The LCD with our **89C52** interfacing has been thoroughly explained in the Section 5.2. The Source Code for the interfacing can be referred to in the end.

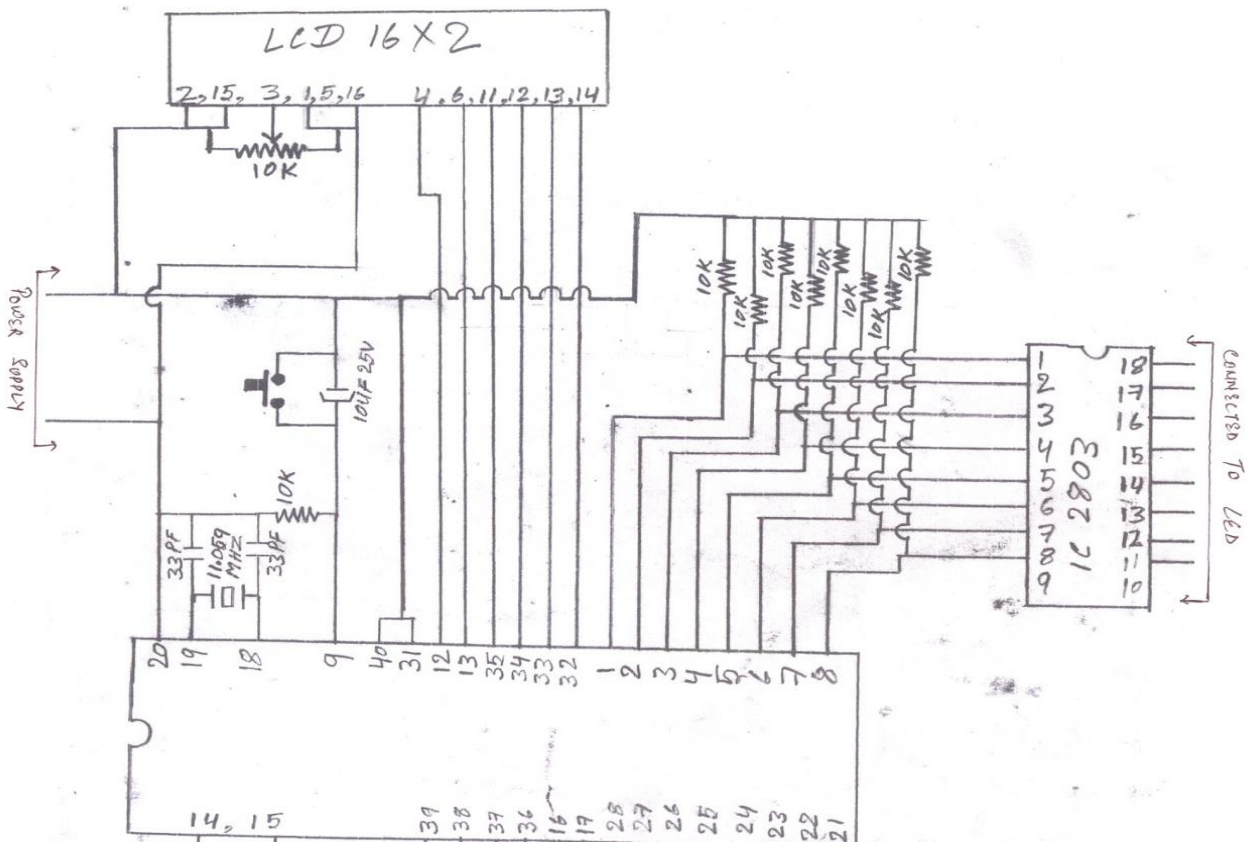


Figure 25: Interfacing of the LCD with 89C52

c) **LED panel**: With the help of a driver circuit, we have also installed a LED panel which would represent visual geometric shapes and figures for demonstration purpose. It can also represent the alphabets and patterns. It will also have an 8X8 LED matrix.[14]

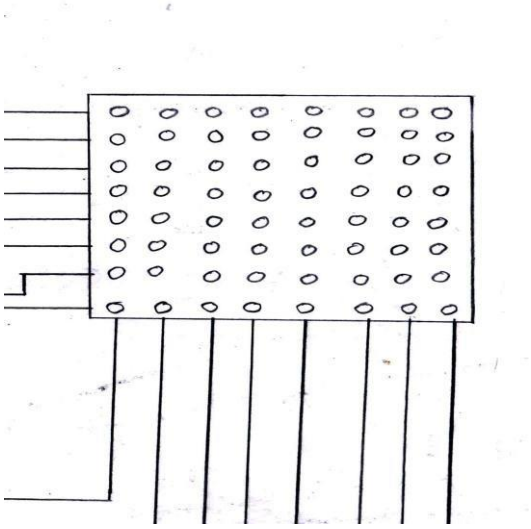


Figure 26: 8X8 matrix (LED Panel)

The **Port 1: Pins 32** to **39** of the 89C52 is connected to the **IC2803** for the LED panel. This according to the programming which was written in to the microcontroller will display the geometric shape we have assumed.

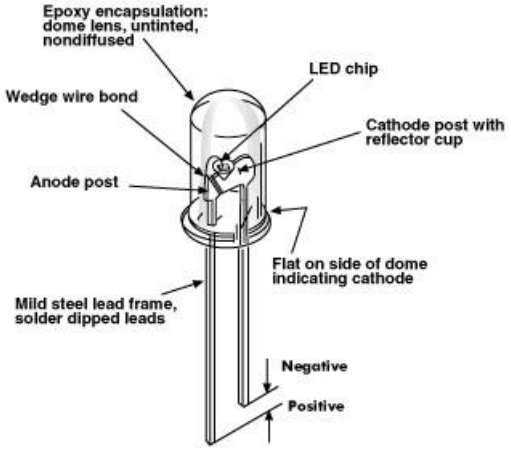


Figure 27: LED Detailed Diagram [14]

d) **Driver Circuit**: For the DC Motors moving clockwise and anticlockwise are controlled by a microcontroller so to drive the motors we need a driver circuit to control the movements of the motor. **Port 2: Pins 21** to **28** is used to interface the driver circuit, which will also apply to the LED panel.

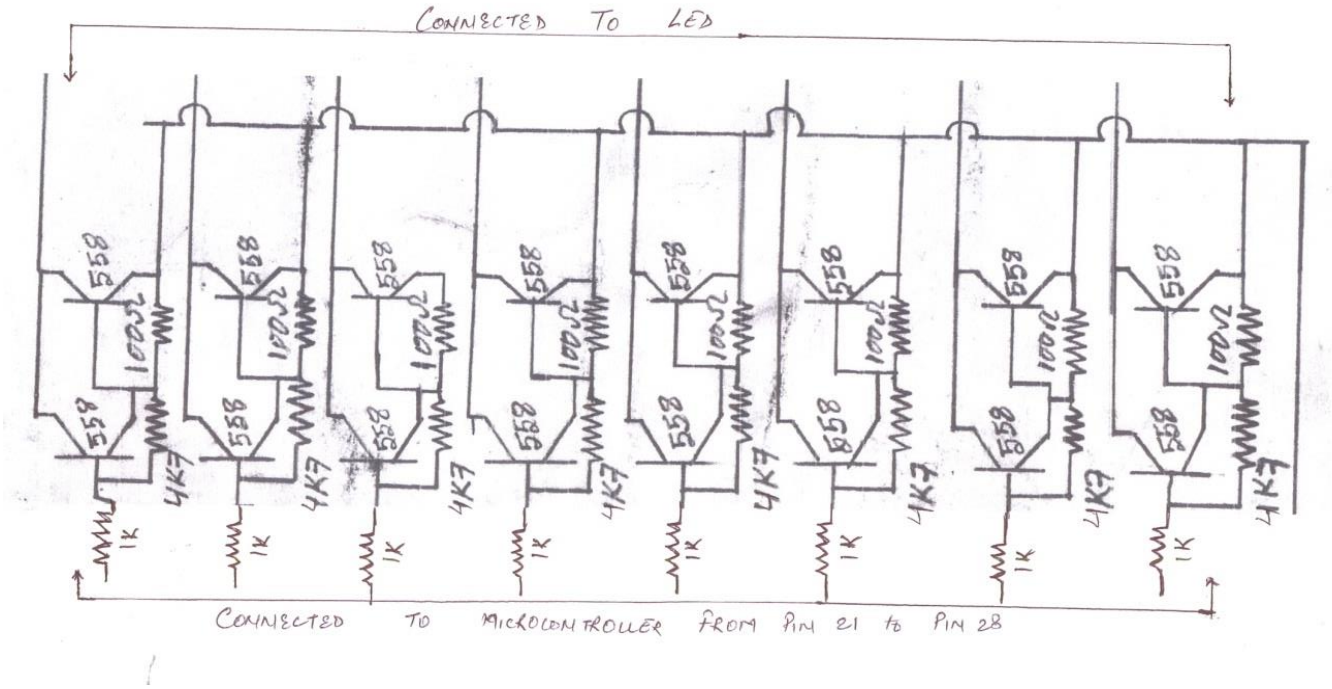


Figure 28: Driver Circuit

e) **DC Motor (Braille Pattern)**: We have used DC motors, **RS130**. These are continuous rotation motors, when you give supply to the DC motor, it will start until that power is withdrawn. The speed of DC motor is controlled using PWM Modulation, a technique for rapidly fluctuating the power on and off. The **Port 3**, pins **20-17** are used for the connection to the microcontroller.

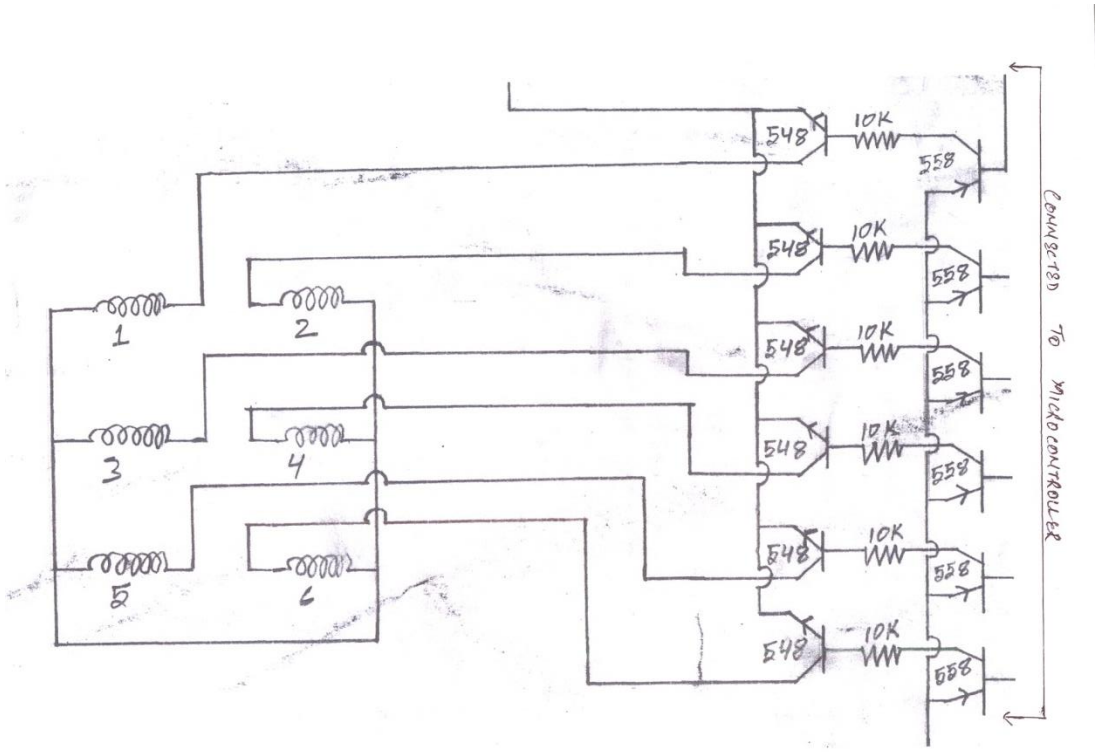


Figure 29: Braille pattern using DC Motors

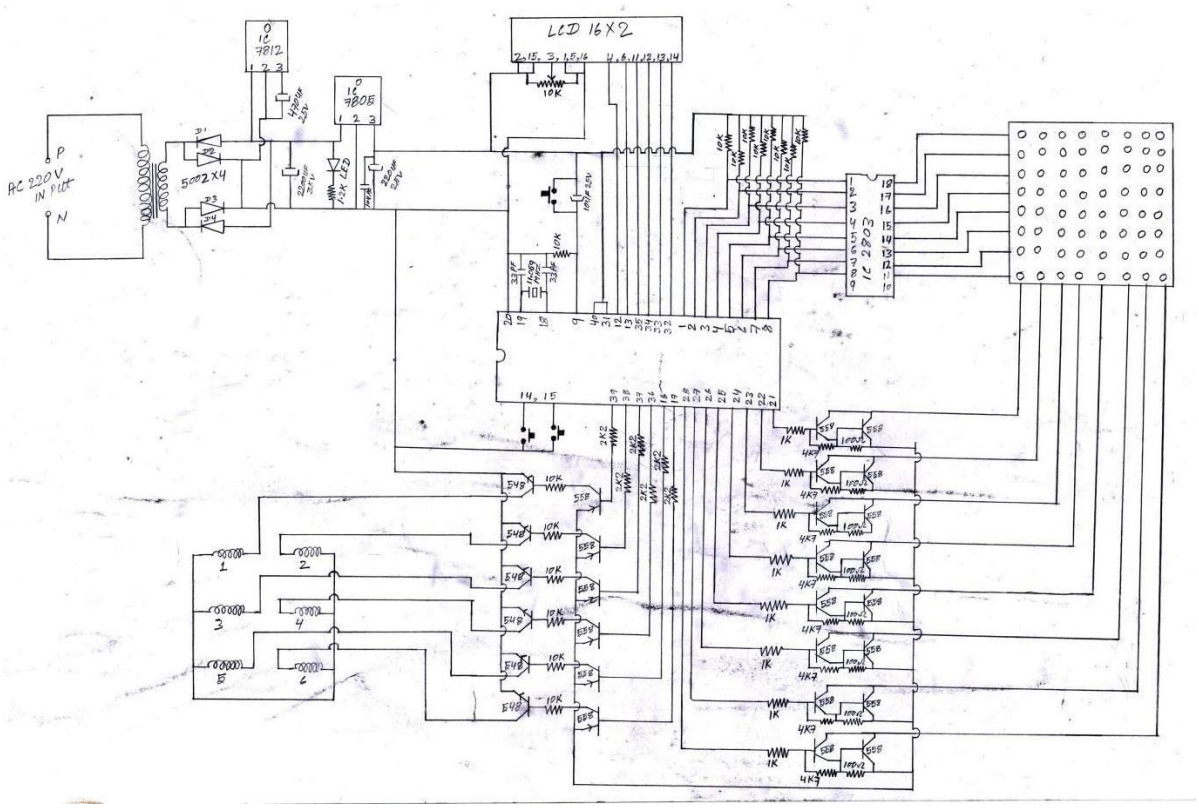


Figure 30: Circuit Diagram for our Project

7.2 Working and Interfacing of Darlington Driver (ULN2803A)

ULN2803 is high-voltage, high-current Darlington drivers comprised of eight NPN Darlington pairs. The eight NPN Darlington connected transistors in this family of arrays are ideally suited for interfacing between low logic level digital circuitry (such as TTL, CMOS or PMOS/NMOS) and the higher current/voltage requirements of lamps, relays, printer hammers or other similar loads for a broad range of computer, industrial, and consumer applications. All devices feature open–collector outputs and freewheeling clamp diodes for transient suppression. The ULN2803 is designed to be compatible with standard TTL families while the ULN2804 is optimized for 6 to 15 volt high level CMOS or PMOS.

Working of ULN2803A

The ULN 2803 IC consists of eight NPN Darlington connected transistors (often called a Darlington pair). Darlington pair consists of two bipolar transistors such that the current amplified by the first is amplified further by the second to get a high current gain β or h_{FE}. The figure shown below is one of the eight Darlington pairs of ULN 2803 IC.

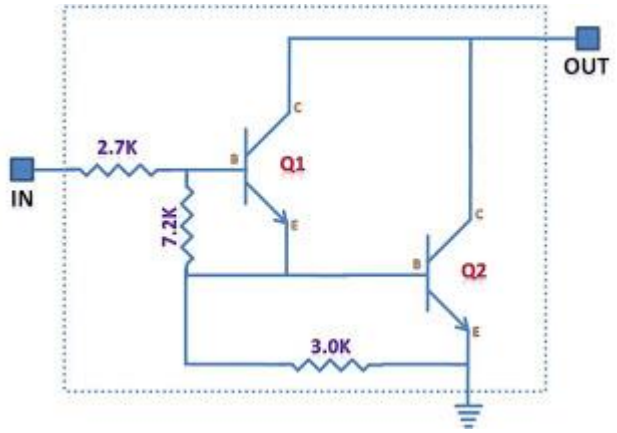


Figure 31: Working of ULN2803A [8]

Now 2 cases arise:

Case 1: When IN is 0 volts.

Q1 and Q2 both will not conduct as there is no base current provided to them. Thus, nothing will appear at the output (OUT).

Case 2: When IN is 5 volts.

Input current will increase and both transistors Q1 and Q2 will begin to conduct. Now, input current of Q2 is combination of input current and emitter current of Q1, so Q2 will conduct more than Q1 resulting in higher current gain which is very much required to meet the higher current requirements of devices like motors, relays etc. Output current flows through Q2 providing a path (sink) to ground for the external circuit that the output is applied to. Thus, when a 5V input is applied to any of the input pins (1 to 8), output voltage at corresponding output pin (11 to 18) drops down to zero providing GND for

the external circuit. Thus, the external circuit gets grounded at one end while it is provided $+V_{cc}$ at its other end. So, the circuit gets completed and starts operating.

Interfacing 8X8 LED, ULN2803A and Microcontroller

An LED dot matrix display consists of a matrix of LED's arranged in a rectangular configuration. The desired character or graphics can be displayed by switching ON /OFF a desired configuration of LED's. Common display configurations available are 7×5, 8×8, 7×15, etc. LED dot matrix can be used in simple display applications where the resolution is not a big concern. The figure below shows the arrangement of LEDs in a typical 8X8 dot matrix display [15].

Any individual LED or a group of LEDs in the matrix can be activated by switching the required number of rows and columns. For example, in the above figure, if Row1 is made high and Column1 is made low, the top left LED (address R1C1) will glow.

In the above diagram you can see that only one LED in a row will be ON at a time but any number of LEDs in a column can be ON at a time. That means the microcontroller's port pin can directly drive a row but it requires additional circuit for driving the column lines. The circuit diagram for interfacing dot matrix display and 8051 microcontroller is shown.

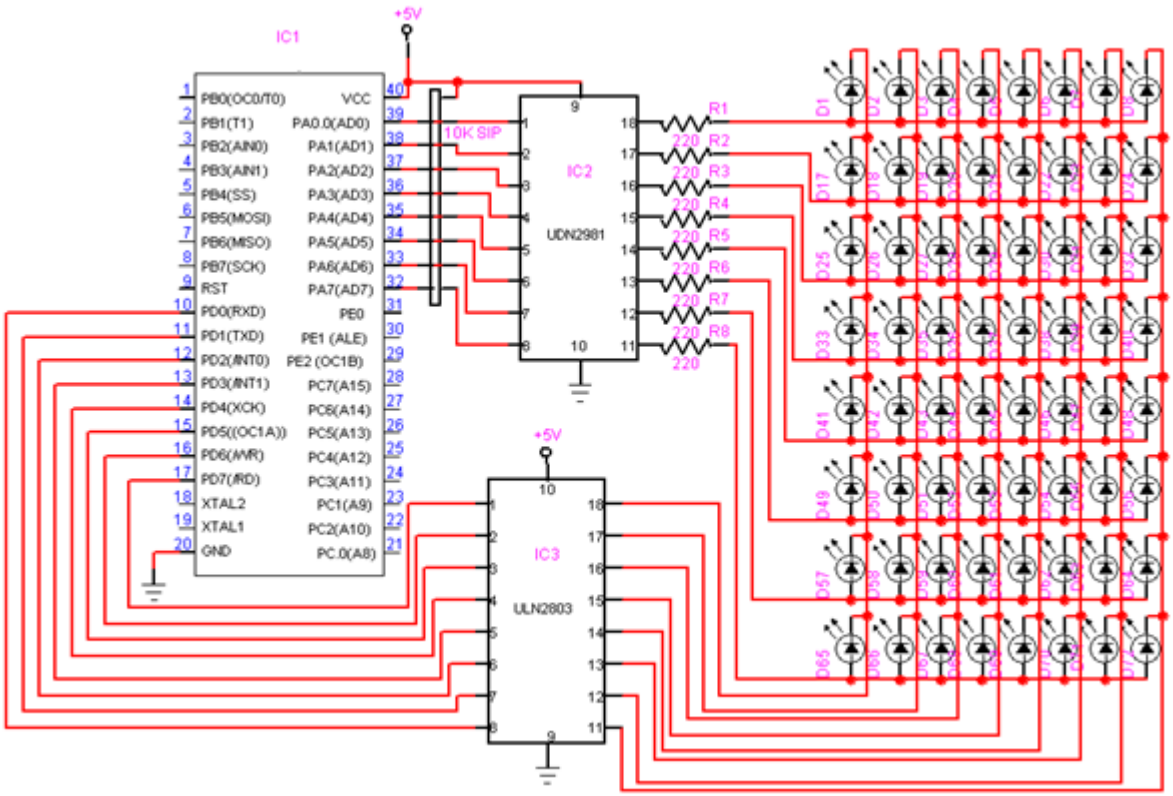


Figure 32: Interfacing of LED, ULN and 89C52 [8][11][6]

7.3 Thermal Resistance Network

The 8X8 **Thermal Heating Network (THN)** has been added. It has been made with the help of an 8X8 resistor network and employs their thermal properties to make the polygon pattern (or graphical diagram like a square or triangle).

The primary challenge was the difficulty of implementing accurate operating temperature predictions directly into system designs using near-exact physical models (known as detailed thermal models) due to the wide disparity in length scales that result in large computational inefficiencies, a **compact thermal model** (CTM) attempts to solve this problem by reducing the detailed model to a far less grid intensive representation, while preserving accuracy in predicting the temperatures at key points with a short calculation time.

A CTM is condensed using thermal network modelling with thermal resistance and thermal capacitance. System model is based on a **thermal interface material** (TIM) measurement apparatus taking into consideration the lateral heat flow in the TIM. The thermal impedance and the temperature difference between the junction and ambient nodes are used to estimate the heat flow division on each heat transfer path.

The thermal network is designed using 8X8 resistor matrix used to generate 2D patterns, and the heat dissipating from it is bearable so that a blind can touch it and feel what actually 2D patterns are, the heat dissipated is controlled through voltage regulator.

CONCLUSION

The hardware for the whole project is successfully working. The coding (refer to the end of the report) has been written and implemented once using the **KEIL C** software. The tested hardware has now been mounted on the PCB. The complete project is fully functional with the addition of a LED panel, previously not planned. From this project we believe to have achieved a cost efficient solution to a daily tool, which can be used for helping the young blind children in learning activities. There is still a large scope to improve this idea further, make it more user friendly, but we have achieved our target goal as per planned upon taking up this project for our final year.

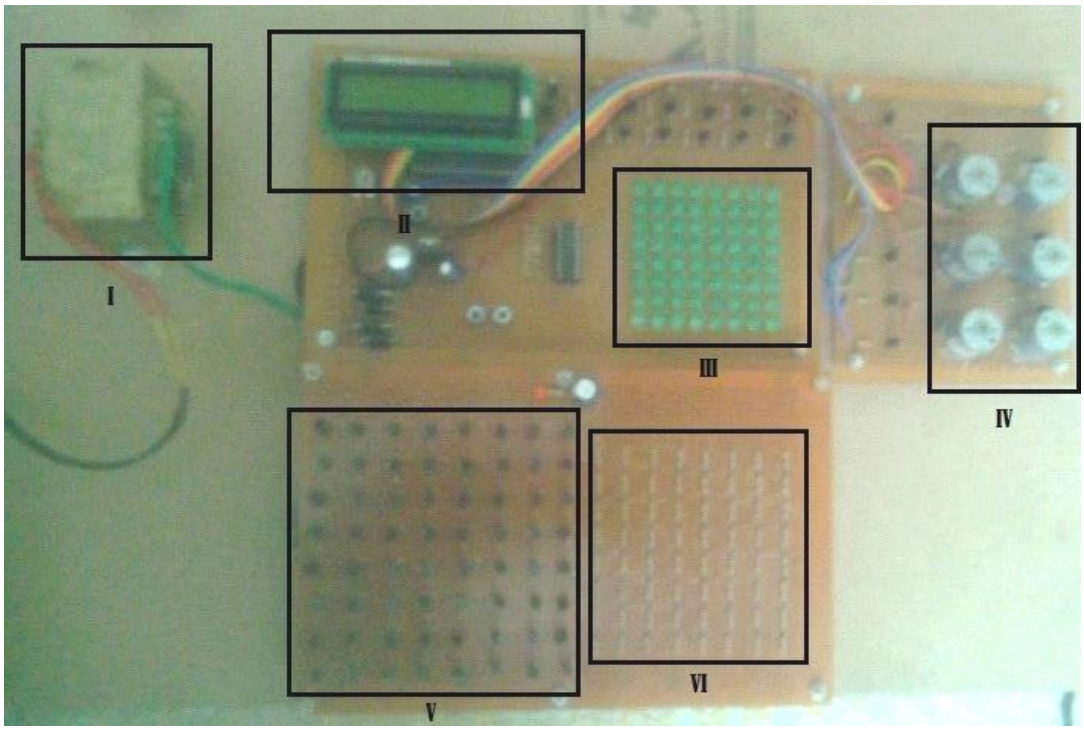


Figure 33: Actual Photograph of the project

Legends:

I.	Transformer being used for Power Supply
II.	Liquid Crystal Display
III.	LED 8X8 panel
IV.	Braille Pattern Representation using 6 motors
V.	8X8 Transistor Logic
VI.	8X8 Thermal Resistance Pattern

SOURCE CODE

This code is the one implemented in our project. This has been written into the microcontroller using the KEIL C, 8051-Based Interfacing software.

```
#include "reg52.h"
//#include "rtc.c"
 sbit rs=P2^4;
 sbit en=P2^5;
#define  lcd_port P0

void delay(unsigned int);
void initialize_lcd();
void lcd_clr();
void lcd_gotoxy(char,char);
void lcd_write_aray(unsigned char  *);
void lcd_putc(unsigned char);
void   send_command(unsigned   char);
void send_data(unsigned char );
          / *void lcd_initialize();
void send_command(unsigned char );
 void send_data(unsigned char );
void  lcd_gotoxy(unsigned  char,unsignedchar);
 void lcd_clr(unsigned char);
 void lcd_write_string(unsigned  char*);
 void delay(unsigned int);
 void lcd_putc(unsigned char);
 */ unsigned char countchar=0;
 // ***************************************************************************
          void initialize_lcd()
```

```
            {
            send_command(0x03); delay(10);
            send_command(0x03); delay(10);
            send_command(0x03); delay(10);
            send_command(0x02);
            send_command(0x02);
            delay(10);
            send_command(0x28); delay(10);
            send_command(0x28); delay(10);

            send_command(0x28); delay(10);
            send_command(0x01); delay(10);
            send_command(0x0c); delay(10);
      //    send_command(0x10,selected_lcd);
send_command(0x80);              delay(10);
      //    send_command(0x06,selected_lcd);
//    send_command(0x0f,selected_lcd);
send_command(0x0e);
            }
    //    ********************************************
void send_command(unsigned char temp)
         { unsigned char
   data1,data2;
   data2=lcd_port;
   data2=data2 & 0x0f;
   data1=temp;   data1=data1
   & 0xf0;   lcd_port= data1 |
   data2;
   rs=0;             en=1;
            delay(500);
   en=0;
   delay(40);
```

```
 data2=lcd_port;    data2=data2
          & 0x0f;
 data1=temp<<4;
data1=data1 & 0xf0;
lcd_port= data1 | data2;
    rs=0;              en=1;
              delay(500);
    en=0;
    delay(40);
          }
// ************************************************************
void send_data(unsigned char temp)
          { unsigned char
          data1,data2;
          data2=lcd_port;
          data2=data2 & 0x0f;
 data1=temp;   data1=data1
& 0xf0;   lcd_port= data1 |
data2; rs=1;
  en=1;
              delay(500);
  en=0;
              delay(40);
  data2=lcd_port;    data2=data2
          & 0x0f;
  data1=temp<<4;
data1=sssdata1 & 0xf0;
lcd_port= data1 | data2;
rs=1;   en=1;
              delay(500);
  en=0;
```

```
            delay(40);
        }
//      ***************************************************************************
void lcd_gotoxy(unsigned char row,unsigned char col)
        { unsigned char j;
delay(10);
    if(row==1)
    j=128;
    if(row==2)
            j=192;
    j=j+col;
send_command(j); delay(10);
        }
// ************************************************************
 void lcd_clr()
        {
        delay(400); send_command(0x01);
        }
// ************************************************************
 void lcd_write_aray(unsigned char  *ptr)
        { while(*ptr!=0)
        {
                send_data(*ptr);
            ptr++;
            }
        }
 // ************************************************************
void delay(unsigned int i )
        { while(i--);
        }
// ************************************************************
void lcd_putc(unsigned char char_data)
```

```
{ send_data(char_data);
}
```

MAIN FILE:

```
#include"fourbitlcdcom.c"
#define output_port P1
//f1 05
//f2 06
//f3 04
//esc 76
//enter 5a
//u arrow 75
//l arrow 6b
//r arrow 74
#define f1    1
#define f2    2
#define f3    3
#define esc   4
#define enter 5
#define f4    6
unsigned  char code sh[]="Keyboard scanner";
unsigned char const
co_ordi_add[]={0x10,0x01,0x00,0x13,0xa2,0x00,0x40,0x0a,0x4c,0x0b,0xff,0xfe,0x00,0x00};
unsigned long orignal_card_id=0;
unsigned long  tempcardno=0;
```

```
unsigned char idata count=0,cursor_pos=0;
unsigned char idata  validkeyno=0;3333333
unsigned char idata key_press=0;
unsigned char  user_name[21]={"SKA"};
unsigned char rf_data_rec[30];
unsigned char bitcount1=11,temp;
unsigned char data_from_kbd1=0;
unsigned char newdata=0;
unsigned int reload_time=0;


bit f_zero_flag=0;
bit e_zero_flag=0;
bit check_scan_bit=0;
bit run_mode=0;
sbit kbdata=P3^3;

//sbit relay1=P2^1;
void convertbcd_time(unsigned char);
void checkkey();
void send_data_rf();
void init();
void ldelay(unsigned int);
void keyserve();
//void convertbcd_time(unsigned char var);
```

```
//void convert_bcd( unsigned long sensor_value,unsigned int sensor_value1);
sbit led=P2^0;
bit flag=0;
unsigned char command_recieved=0;
typedef unsigned int ui;
typedef unsigned char uc;
void main()
{
unsigned char sec1,i=0;
    init();
     P2=0;
    led=1;
//   relay1=0;
   // P1=0;
     delay(50000);
     sec1=1;
     initialize_lcd();
     lcd_write_aray(sh);
//lcd_gotoxy(2,0);
    delay(50000);
     lcd_clr();
     send_command(0x0f);
//   read_minutes(&sec1);
//   store_data(&sec1,(i*10)+1);
 //   run_mode=1;
```

```
while(1)
  {
                if(check_scan_bit==1 )
   {
      check_scan_bit=0;
       //lcd_gotoxy(2,1);
       //lcd_putc('1');
       checkkey();
      keyserve();
      f_zero_flag=0;
       e_zero_flag=0;
   }
 }
}
//*********************************************
void checkkey()
{
              if(cursor_pos<32)
               {
                        if(newdata==0x1c)
                         {
                          user_name[cursor_pos]='A';
                                    output_port=1;
                          lcd_putc('A');cursor_pos++;
                        }
```

```
else if(newdata==0x32)
                    {
                     user_name[cursor_pos]='B';
                                output_port=2;
                     lcd_putc('B');cursor_pos++;
                    }
 else if(newdata==0x21)
                    {
                     user_name[cursor_pos]='C';output_port=3;
                     lcd_putc('C');cursor_pos++;
                    }
 else if(newdata==0x23)
                    {
                     user_name[cursor_pos]='D';output_port=4;
                     lcd_putc('D');cursor_pos++;
                    }

  else if(newdata==0x24)
                    {
                     user_name[cursor_pos]='E';output_port=5;
                     lcd_putc('E');cursor_pos++;
                    }
                     else if(newdata==0x2B)
                    {
                     user_name[cursor_pos]='F';output_port=6;
```

```
                    lcd_putc('F');cursor_pos++;
                   }
  else if(newdata==0x34)
                    {
                    user_name[cursor_pos]='G';output_port=7;
                    lcd_putc('G');cursor_pos++;
                   }
 else if(newdata==0x33)
                    {
                    user_name[cursor_pos]='H';output_port=8;
                    lcd_putc('H');cursor_pos++;
                   }
 else if(newdata==0x43)
                    {
                    user_name[cursor_pos]='I';output_port=9;
                    lcd_putc('I');cursor_pos++;
                   }
 else if(newdata==0x3B)
                    {
                    user_name[cursor_pos]='J';output_port=10;
                    lcd_putc('J');cursor_pos++;
                   }
else if(newdata==0x42)
                    {
                    user_name[cursor_pos]='K';output_port=11;
```

```
                    lcd_putc('K');cursor_pos++;
                   }
else if(newdata==0x4B)
                    {
                     user_name[cursor_pos]='L';output_port=12;
                     lcd_putc('L');cursor_pos++;
                   }
else if(newdata==0x3A)
                    {
                     user_name[cursor_pos]='M';output_port=13;
                     lcd_putc('M');cursor_pos++;
                   }
else if(newdata==0x31)
                    {
                     user_name[cursor_pos]='N';output_port=14;
                     lcd_putc('N');cursor_pos++;
                   }
else if(newdata==0x44)
                    {
                     user_name[cursor_pos]='O';output_port=15;
                     lcd_putc('O');cursor_pos++;
                   }
else if(newdata==0x4D)
                    {
                     user_name[cursor_pos]='P';output_port=16;
```

```
                    lcd_putc('P');cursor_pos++;
                   }
else if(newdata==0x15)
                    {
                    user_name[cursor_pos]='Q';output_port=17;
                    lcd_putc('Q');cursor_pos++;
                   }
else if(newdata==0x2D)
                    {
                    user_name[cursor_pos]='R';output_port=18;
                    lcd_putc('R');cursor_pos++;
                   }
else if(newdata==0x1B)
                    {
                    user_name[cursor_pos]='S';output_port=19;
                    lcd_putc('S');cursor_pos++;
                   }
else if(newdata==0x2C)
                    {
                    user_name[cursor_pos]='T';output_port=20;
                    lcd_putc('T');cursor_pos++;
                   }
else if(newdata==0x3C)
                    {
                    user_name[cursor_pos]='U';output_port=21;
```

```
                    lcd_putc('U');cursor_pos++;
                   }
else if(newdata==0x2A)
                    {
                     user_name[cursor_pos]='V';output_port=22;
                     lcd_putc('V');cursor_pos++;
                   }
else if(newdata==0x1D)
                    {
                     user_name[cursor_pos]='W';output_port=23;
                     lcd_putc('W');cursor_pos++;
                   }
else if(newdata==0x22)
                    {
                     user_name[cursor_pos]='X';output_port=24;
                     lcd_putc('X');cursor_pos++;
                   }
else if(newdata==0x35)
                    {
                     user_name[cursor_pos]='Y';output_port=25;
                     lcd_putc('Y');cursor_pos++;
                   }
else if(newdata==0x1A)
                    {
                     user_name[cursor_pos]='Z';output_port=26;
```

```
                    lcd_putc('Z');cursor_pos++;
                   }
else if(newdata==0x45)
                    {
                     user_name[cursor_pos]='0';output_port=27;
                     lcd_putc('0');cursor_pos++;
                   }
else if(newdata==0x16)
                    {
                    user_name[cursor_pos]='1';output_port=28;
                     lcd_putc('1');cursor_pos++;
                   }
else if(newdata==0x1e)
                    {
                    user_name[cursor_pos]='2';output_port=29;
                     lcd_putc('2');cursor_pos++;
                   }
else if(newdata==0x26)
                    {
                    user_name[cursor_pos]='3';output_port=30;
                     lcd_putc('3');cursor_pos++;
                   }
else if(newdata==0x25)
                    {
                     user_name[cursor_pos]='4';output_port=31;
```

```
                    lcd_putc('4');cursor_pos++;
                   }
else if(newdata==0x2e)
                    {
                     user_name[cursor_pos]='5';output_port=32;
                     lcd_putc('5');cursor_pos++;
                   }
else if(newdata==0x36)
                    {
                    user_name[cursor_pos]='6';output_port=33;
                     lcd_putc('6');cursor_pos++;
                   }
else if(newdata==0x3d)
                    {
                     user_name[cursor_pos]='7';output_port=34;
                     lcd_putc('7');cursor_pos++;
                   }
else if(newdata==0x3e)
                    {
                     user_name[cursor_pos]='8';output_port=35;
                     lcd_putc('8');cursor_pos++;
                   }
else if(newdata==0x46)
                    {
                    user_name[cursor_pos]='9';output_port=36;
```

```
                        lcd_putc('9');cursor_pos++;
                      }
    }
      if(newdata==0x05)
     {
      key_press=1;
     }
      else if(newdata==0x06)
     {
              key_press=2;
     }
    else if(newdata==0x76)
     {
            key_press=3;
     }
}
//*********************************************
void keyserve()
{
unsigned int idata i;
switch(key_press)
 {
   /*case f1: lcd_clr();
         lcd_gotoxy(1,0);
                  lcd_write_aray(sh);
```

```
                lcd_gotoxy(2,0);
        cursor_pos=0;
                for(i=0;i<10;i++)
                user_name[i]=' ';
           run_mode=0;
                //rec_mode=0;
        break;
case f2 :
         lcd_clr();
        lcd_gotoxy(1,0);
                lcd_write_aray(sh);
                lcd_gotoxy(2,0);
        //cursor_pos=0;
                //for(i=0;i<10;i++)
                //user_name[i]=' ';
        run_mode=0;
         //rec_mode=0;
            break;*/
   case f1: lcd_clr();
          cursor_pos=0;
         key_press=0;
                  break;
case f2:key_press=0;
          if(cursor_pos>=16)
            lcd_gotoxy(2,0);
```

```
            else if(cursor_pos==32)
             cursor_pos=0;
            break;
        }
}
void serout0(unsigned char var)
{
SBUF=var;
delay(1000);
}
//*****************************************
void timer0() interrupt 1 using 2
{
 TH0= 0x3C;                  //load the reload values in the timer
    TL0 = 0xb0;
if(++reload_time==10)
 {
    bitcount1 = 11;
   reload_time=0;
 }
 }
//***********************************************
void ext0_isr() interrupt 0 using  1
{
reload_time=0
```

```
if(bitcount1 < 11 && bitcount1 > 2)
    {
     //flag=1;
              data_from_kbd1 = data_from_kbd1 >> 1;
    //        delay(3);
              if(kbdata)
                       data_from_kbd1 = data_from_kbd1 | 0x80;
    }
    if(--bitcount1 == 0)
    {
      // flag=1;
              bitcount1 = 11;
     // P0= data_from_kbd1;
     if(f_zero_flag==1 || (e_zero_flag==1 && f_zero_flag==1))
       {
                 check_scan_bit=1;

          newdata=data_from_kbd1;
          //  ET0=1;
       }
           if(data_from_kbd1==0xf0)
                f_zero_flag=1;
       if(data_from_kbd1==0xe0)
              e_zero_flag=1;
        data_from_kbd1=0;
```

```
    }
}
//*********************************************
void ldelay(unsigned int k)
{
while(k--);
}
void init()
{
    P0 = 0XFF;     //initializing ports
    P1 = 0XFF;
    P2 = 0XFF;
    P3 = 0XFF;
   TMOD=0x21;  //21 FOR timer 1 autoreload mode for serial comn and timer0 in mode 1
    TH1=0xFd;     //setting baud rate 9600
    TH0     = 0x3C;                    //load the reload values in the timer
    TL0 = 0xB0;
   IT0=1;
    SCON=0x50;
   //TR1=1;
    IE=0X83;        //enable timer0 and serial interrupt
  TR0=1;
}
void convertbcd_time(unsigned char var)
{
```

```
unsigned char var1;
var1=(var>>4);
if(var1<10)
lcd_putc(var1+48);
else
lcd_putc(' ');
var1=var & 0x0f;
if(var1<10)
lcd_putc(var1+48);
else
lcd_putc(' ');
}
{
signed char loop=9;
unsigned int temp;
unsigned char display_value[10];
//unsigned char display_value1[10];
while(loop>=0)
  {
   temp=sensor_value%10;
   display_value[loop]=temp;
   sensor_value=sensor_value/10;
   loop--;
   //count_digit++;
   }
```

```
   //loop=4;
for(loop=0;loop<10;loop++)
   lcd_putc(display_value[loop]+48);
}*/
//************************************************
void serialcom(void) interrupt 4 using 1
{
 unsigned char i,flag_on=0,recv_buf0;
 static unsigned int rf_data_length;
/*  if(TI)
    {
            TI=0;
    }
    if(RI)
    {
    flag=1;
     if(SBUF==0x7e)
        count=0;
     else if(SBUF==13)
     {
         flag=1;
              orignal_card_id=tempcardno;
              tempcardno=0;
      }
     else
```

```
   {
          // r[count]=SBUF;
           tempcardno=tempcardno<<8 ;
    tempcardno=tempcardno | SBUF;
           count++;
   }
RI = 0;
}*/
if(TI)
{
          TI=0;
}
if(RI)
  {
     RI=0;
          recv_buf0 = SBUF;
                    if(recv_buf0==0x7e)
                    {
                     count=0;
                     rf_data_length=4;
                    }
                    rf_data_rec[count]=recv_buf0;
                    count++;
if( count==3)
                     {
```

```
                    rf_data_length=rf_data_rec[1];
                        rf_data_length=((rf_data_length<<8) |  rf_data_rec[2])+3;
                }
               else
                 {
                        if(count>rf_data_length)
                         {
                         if(rf_data_rec[3]==0x8b &&
rf_data_rec[rf_data_length2]==0x00) //successfull transmission
                                        {
                                          led=0;
                                        // if(++line_add==8)
                                         //   line_add=0;
                                        }
                    else if(rf_data_rec[3]==0x90)  //one frame is recieved
                                        {
                                            led=0;
                                          // s0_ric=0;
                                       command_recieved=1;
                                            //relay1= 1;
                                        }
}
}}}
```

REFERENCES

[1] Daniels & Bright, 1996, *The World's Writing Systems*, p 817–818

[2] Peter Daniels, 1996, "Analog and Digital Writing", in *The World's Writing Systems*,p 886

[3] Roy, Noëlle, "Louis Braille 1809–1852, a French genius", *Valentin Haüy Association website*, retrieved 2011-02-05

[4] http://electroniccomponents.globalspec.com/LearnMore/Electrical_Electronic_Components/Power_Supplies_Conditioners

[5] Miller, Rex. Electronics the Easy Way, 4th ed. Barron's Educational Series, 2002 p. 88-89.

[6] Atmel Corporation 89C52 – Microcontroller (Datasheet)

[7] www.atmel.in/Images/doc0313

[8] Texas Instruments - ULN2803A (Datasheet)

[9] www.semicon.toshiba.co.jp/info/docget.jsp?pid=ULN2803APG

[10] kwlhk.en.seekic.com/product/integrated_circuits_ics/ULN2803AG.html

[11] T.J. Scheffer and J. Nehring,"A new highly multiplexable LCD," Appl. Phys. Lett., vol. 48, no. 10, pp. 1021–1023, Nov. 1984

[12] Stan D'Souza. "Microchip AN529: Multiplexing LED Drive and a 8x8 Keypad Sampling"

[13] Claus Kühnel (2001). *BASCOM Programming of Microcontrollers with Ease: An Introduction by Program Examples*. Universal Publishers. pp. 114–119.

[14] www.circuitstoday.com/simple-led-projects-using-arduino

[15] Mohd, Wajid, Mohmammad Waris Abdullah, Dr. Omar Farooq. "*Imprinted Braille-Character Pattern Recongnition using Image Processing*" pp. 11-12, ICIIP 2011

DATA SHEETS:

- Atmel Corporation 89C52 – Microcontroller

- Xiamen Amotec Display Co. Ltd. – LCD Module: ADM1602K

RESEARCH PAPERS

- Avinash Chaudhary, Pardeep Garg, Arjun Agarwal "Using Rotaion Method for removal of Misalignment of Scanned Braille Patern". International Journal Of Advancements In Electronics And Electrical Engineering Year: 2012, Page(s):145 – 149.

- Jan Mennens "Optical recognition of Braille writing", IEEE 1993.

- Vidyashankar, G. and Shivakumara, P. (2004) "Rotation Invariant Histogram Feature for Recognition of Braille Symbols". In: International Conference on Cognitive Systems, December 14-15, 2004, New Delhi.

- Mennens, J., et al, "Optical Recognition of Braille Writing Using Standard Equipment", IEEE transactions of rehabilitation engineering, Vol. 2, No. 4, December 1994 Tavel, P. 2007 Modeling and Simulation Design. AK Peters Ltd.

- Hentzschel, T. W., and P. Blenkhorn, "An Optical Reading Systems for Embossed Braille Characters using a Twin Shadows Approach", Journal of Microcomputer Applications, 1995.

- Blenkhorn, P., "A System for Converting Braille into Print", IEEE transactions on rehabilitation engineering, Vol. 3, No. 2, June 1995.