MINOR PROJECT REPORT

PROPELLER LED DISPLAY

SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE AWARD OF DEGREE OF BACHELOR OF TECHNOLOGY IN ELECTRONICS AND COMMUNICATION ENGINEERING

Guide: Mr. B.K.Singh

Submitted By:

Shorya Gupta (06714802810)
DeepanshuTrehan(04896402810)
NirajAgarwal(07114802810)

MAHARAJA AGRASEN INSTITUTE OF TECHNOLOGY
SECTOR-22, ROHINI, DELHI-110086

AFFILIATED TO

GURUGOBINDSINGHINDRAPRASTHAUNIVERSITY, NEW DELHI-110078
(2010-2014)

Certificate

This is to certify that the Minor Project Report (ETEC 459) entitled "**Propeller LED Display**" done by team comprising of Shorya Gupta (06714802810), Deepanshu Trehan (04896402810) and NirajAgarwal (07114802810) is an authentic work carried out by them under my guidance. The matter embodied in this minor project work has not been submitted earlier for the award of any degree or diploma to the best of my knowledge and belief.

Date:

Name of the Guide: Mr. B. K. Singh

Designation

Address

ACKNOWLEDGEMENT

We would like to articulate our profound gratitude and indebtedness to our project guide Mr. B. K. Singh, who has always been a constant motivation and guiding factor throughout the project time in and out as well. It has been a great pleasure for us to get an opportunity to work under him and complete the project successfully. We wish to extend our sincere thanks to Prof. R S Gupta, Head of Department, for approving our project work with great interest. An undertaking of this nature could never have been attempted with our reference to and inspiration from the works of others whose details are mentioned in references section. We acknowledge our indebtedness to all of them.

Shorya Gupta (06714802810)
shoryagupta1493@gmail.com

DeepanshuTrehan (04896402810)
deepanshu.trehan@yahoo.com

NirajAgarwal (07114802810)
niraj.agarwal@hotmail.co.in

ABSTRACT

The basic idea of this project comes from the concept of persistence of vision. This phenomenon is related to vision capability of the human eye, where an afterimage is thought to persist for approximately 1/25th of a second. So if someone is observing at a rate of 25 images per second, they appear to form a continuous picture.

In this display, the LED strip moves fast that one is able to see a matrix of LEDs. The time of a single rotation is divided into several shorter time periods during which each individual LED is kept on/off to display different characters.

Motivation:

The work is motivated by two types of applications:

1. Better alternative to conventional LED screens having hundreds of LEDSs.
2. Visible from all sides (360 degrees), compared to conventional display boards at public places.

Goal:

The goal of this project is to build a rotating propeller display using 8 LEDs in a cost effective way. Also the project aims at developing energy efficient and an attractive display.

CONTENTS

LIST OF FIGURES

Chapter 1.INTRODUCTION

The moving message display is not something that we are not aware of; we observe such things almost every day in our day to day life. These displays are used in shops, railway stations, airports etc. Now-a-days the demand of moving message displays is continuously increasing. In case of normal LED display, around 450 - 550 or even more LED's are used. So, evidently the space requirement and cost for obvious reasons is huge. To overcome this problem, the concept of PROPELLER LED DISPLAY is introduced. This concept is based on the principle of "**Persistence of Vision**". This phenomenon is related to vision capability of the human eye, where an afterimage is thought to persist for approximately 1/25th of a second. So if someone is observing images at a rate of 25 per second, they appear to form a continuous picture.

Nowadays electronic manufacturers of TVs and Monitors are trying to employ this principle. The goal of this project is to make a single color **PROPELLER LED DISPLAY** that uses persistence of vision to create an optical illusion to display the required message.

1.1 Basic Details:

The propeller message display system consists of a microcontroller, a LED strip consisting of 8 LEDs and an AC Motor over which the microcontroller board and the LED strip are mounted. The Basic setup of the system is shown in Fig 1.1.

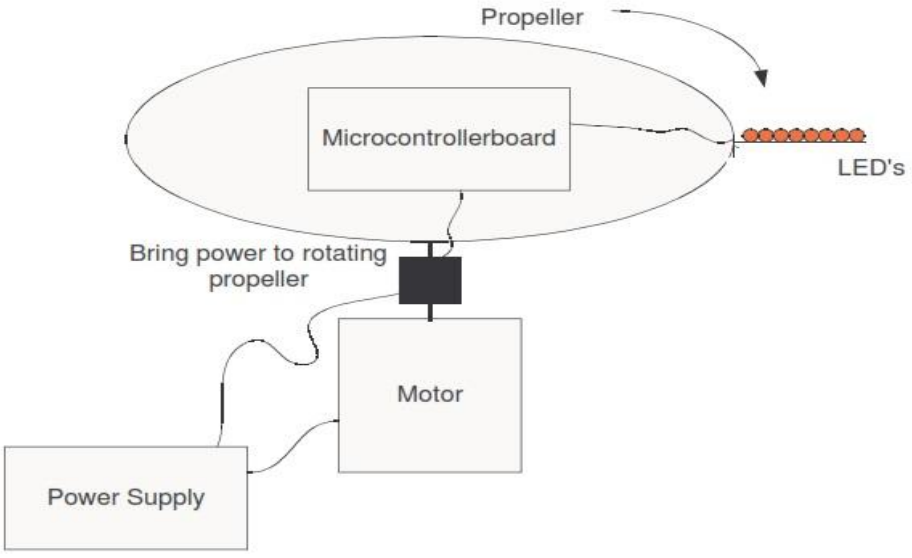

Fig 1.1: Basic Setup of Propeller Display

1.2 Advantages:

- Number of LEDs used is much less, reducing the complexity of the system.
- Power consumption reduces to an optimum level.
- Visibility of the message displayed is 360 degrees which is helpful at public places like railway stations and airports.
- Power Consumption is independent of the length of text displayed, as only 8 LEDs are used.

1.3 Applications:

- Useful in places like railway stations where the display boards are visible from only one side. It can make any display visible from all sides (360 degrees). To provide a 360 degree view using the conventional LED boards, it will increase the complexity of the system. And moreover the number of LEDs used will increase by a large amount, thereby increasing the cost. Hence propeller display is of great use in public places.

- Better alternative to the conventional LED screens that have hundreds of LEDs. Propeller LED Display use only 8 LEDs to display any length of text whereas the conventional boards consists of a large number of LEDs.

Fig 1.2: Railway Station Display

Fig 1.3: Conventional LED Board

Chapter 2. COMPONENTS USED

2.1 MICROCONTROLLER

2.1.1 AVR Microcontroller:

AVR was developed in the year 1996 by Atmel Corporation. The architecture of AVR was developed by Alf-EgilBogen and VegardWollan. AVR derives its name from its developers and stands for Alf-EgilBogenVegardWollanRISC microcontroller. The AT90S8515 was the first microcontroller which was based on AVR architecture however the first microcontroller to hit the commercial market was AT90S1200 in the year 1997.
AVR microcontrollers are available in four different categories:

- TinyAVR – Less memory, small size, suitable only for simpler applications.
- MegaAVR – These are the most popular ones having good amount of memory (upto 256 KB), higher number of inbuilt peripherals and suitable for moderate to complex applications.
- XmegaAVR – Used commercially for complex applications, which require large program memory and high speed.
- Application-specific AVR - MegaAVRs with special features not found on the other members of the AVR family, such as LCD controller, USB controller, advanced PWM, CAN etc.

2.1.2 AT Mega 8535:

The ATmega8535 is a low-power CMOS 8-bit microcontroller based on the AVRenhanced RISC architecture. By executing instructions in a single clock cycle, theATmega8535 achieves throughputs approaching 1 MIPS per MHz allowing the systemdesigned to optimize power consumption versus processing speed.

Pin Description

VCC- Digital supply voltage.

GNDG- Ground.

Port-A (PA7-PA0) - Port A serves as the analog inputs to the A/D Converter.Port A also serves as an 8-bit bi-directional I/O port, if the A/D Converter is not used.Port pins can provide internal pull-up resistors (selected for each bit). The Port A outputbuffers have symmetrical drive characteristics with both high sink and source capability.When pins PA0 to PA7 are used as inputs and are externally pulled low, they will sourcecurrent if the internal pull-up resistors are activated. The Port A pins are tri-stated whena reset condition becomes active, even if the clock is not running.

Port-B (PB7-PB0) - Port B is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for eachbit). The Port B output buffers have symmetrical drive characteristics with both high sinkand source capability. As inputs, Port B pins that are

externally pulled low will source current if the pull-up resistors are activated. The Port B pins are tri-stated when a reset condition becomes active, even if the clock is not running.

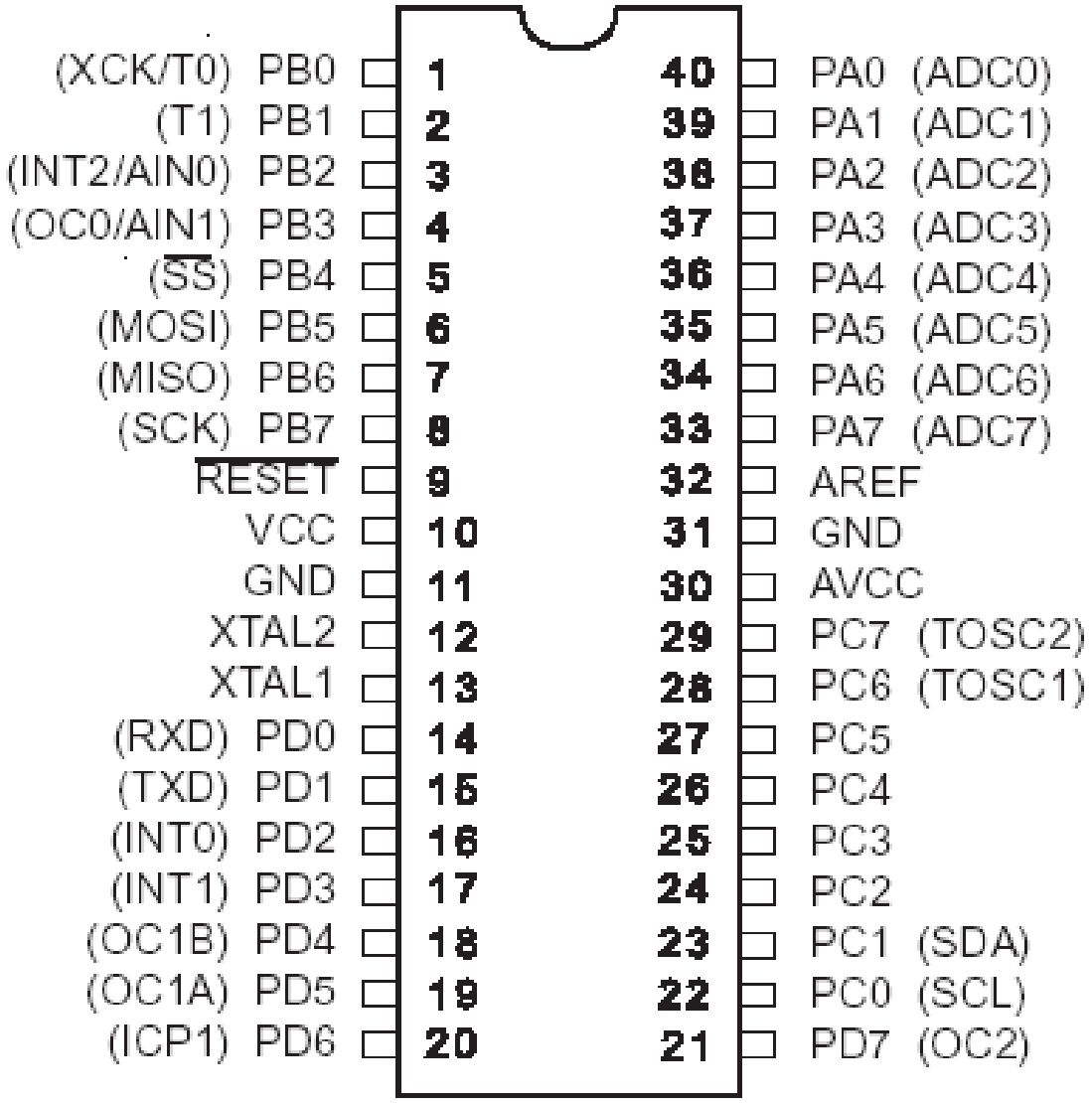

Fig 2.1: PIN Diagram

Port-C (PC7-PC0) - Port C is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port C output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port C pins that are externally pulled low will source current if the pull-up resistors are activated. The Port C pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port-D (PD7-PD0) - Port D is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port D output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port D pins that are externally pulled low will source current if the pull-up resistors are activated. The Port D

pins are tri-stated when a reset condition becomes active, even if the clock is not running.

RESET- A low level on this pin for longer than the minimum pulse length will generate a reset, even if the clock is not running. Shorter pulses are not guaranteed to generate a reset.

XTAL1- Input to the inverting Oscillator amplifier and input to the internal clock operating circuit.

XTAL2- Output from the inverting Oscillator amplifier.

AVCC- AVCC is the supply voltage pin for Port A and the A/D Converter. It should be externally connected to VCC, even if the ADC is not used. If the ADC is used, it should be connected to VCC through a low-pass filter.

AREF- AREF is the analog reference pin for the A/D Converter.

Architecture

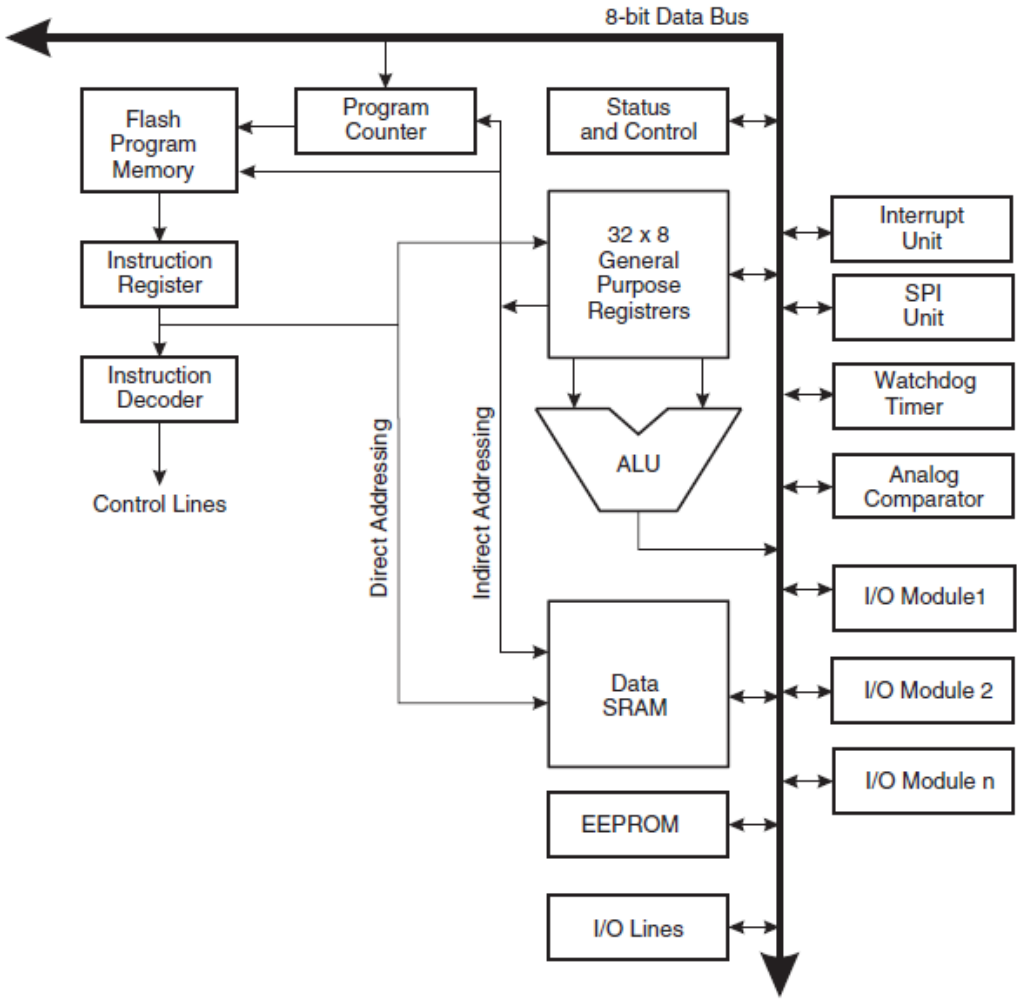

Fig 2.2: Architecture of AT Mega 8535

In order to maximize performance and parallelism, the AVR uses Harvard architecture– with separate memories and buses for program and data. Instructions in the program memory are executed with a single level pipelining. While one instruction is being executed, the next instruction is pre-fetched from the program memory. This concept enables instructions to be executed in every clock cycle. The program memory is In-System Re-Programmable Flash memory.

The fast-access Register File contains 32 x 8-bit general purpose working registers witha single clock cycle access time. This allows single-cycle Arithmetic Logic Unit (ALU)operation. In a typical ALU operation, two operands are output from the Register File, the operation is executed, and the result is stored back in the Register File – in one clock cycle.

- **ADC Interface:** The ATmega8535 features a 10-bit successive approximation ADC. The ADC is connected to an 8-channel Analog Multiplexer which allows eight single-ended voltage inputs constructed from the pins of Port A. The single-ended voltage inputs refer to 0V(GND).

- **Timers/Counters:**ATmega8535 consists of two 8-bit and one 16-bit timer/counter. These are useful for generating precision actions for e.g., creating time delays between two operations.

- **Interrupts:** ATmega8535 consists of 21 interrupt sources out of which four are external. The remaining are internal interrupts support the peripherals like USART, ADC, timers etc.

- **USART:** It is available for interfacing with external device capable of communicating serially (data transmission bit by bit).

- **Memory:** ATmega8535 consist of three different memory sections:
  1. 8K Bytes of In-System Self-Programmable Flash.
  2. 512 Bytes EEPROM.
  3. 512 Bytes Internal SRAM.

- **ISP:** AVR family of controllers have In System Programmable Flash Memory which can be programmed without removing the IC from the circuit, ISP allows to reprogram the controller while it is in the application circuit.

- **SPI (Serial Peripheral Interface**): The Serial Peripheral Interface (SPI) allows high-speed synchronous data transferbetween the ATmega8535 and peripheral devices or between several AVR devices.

2.2AC-MOTOR

An AC motor is an electric motor driven by an alternating current (AC).

It commonly consists of two basic parts:

1. An outside stationary stator having coils supplied with alternating current to produce a rotating magnetic field
2. An inside rotor attached to the output shaft that is given a torque by the rotating field.

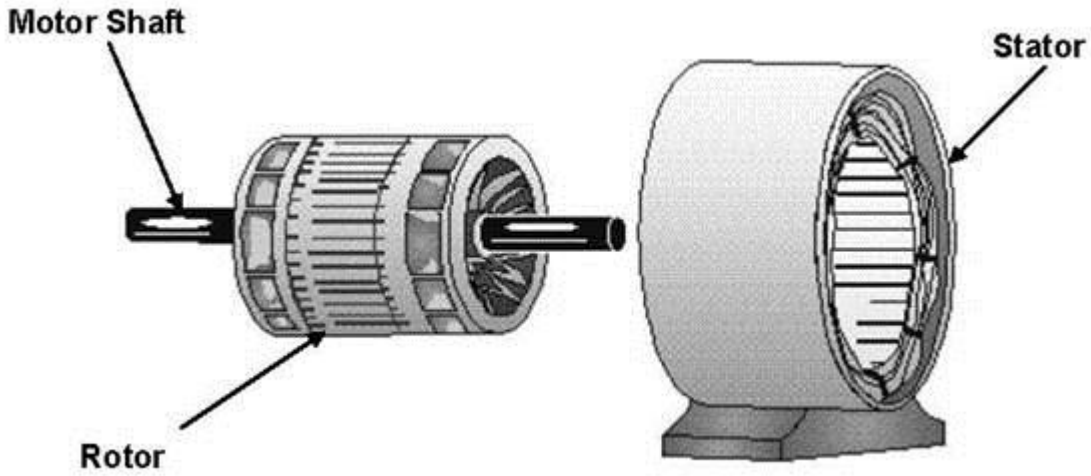

Fig 2.3: Rotor & Stator

There are two main types of AC motors, depending on the type of rotor used.

1. Induction motor or asynchronous motor: this type relies on a small difference in speed between the rotating magnetic field and the rotor to induce rotor current.
2. Synchronous motor: This type does not rely on induction and as a result, can rotate exactly at the supply frequency or a sub-multiple of the supply frequency.

The magnetic field on the rotor is either generated by current delivered through slip rings or by a permanent magnet. Other types of motors include eddy current motors, and also AC/DC mechanically commutated machines in which speed is dependent on voltage and winding connection.

AC Motor required for the Propeller Display project is the Induction Motor having a speed of around 825 rounds per minute. (This will be explained later in Chapter).

2.3 LED (Light Emitting Diode):

A **light-emitting diode** (**LED**) is a semiconductor light source.

When a light-emitting diode is switched on, electrons are able to recombine with holes within the device, releasing energy in the form of photons. This effect is called electroluminescence, and the colour 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 cm^2).

LEDs have many advantages over incandescent light sources:

1. Lower energy consumption
2. Longer lifetime
3. Improved physical robustness
4. Smaller size and faster switching.

Fig 2.4: LED

2.3.1 Working Principle of LED

Working of the Light Emitted Diode is based on the Principle that when diode is forward biased the electron and hole recombination takes place in the depletion region of P-N junction diode. When the electron hole recombination takes place the radiations of light are emitted in the form of photons of light and the colour of the light which is emitted by the LED will depend on the Energy band gap between the semiconductors.

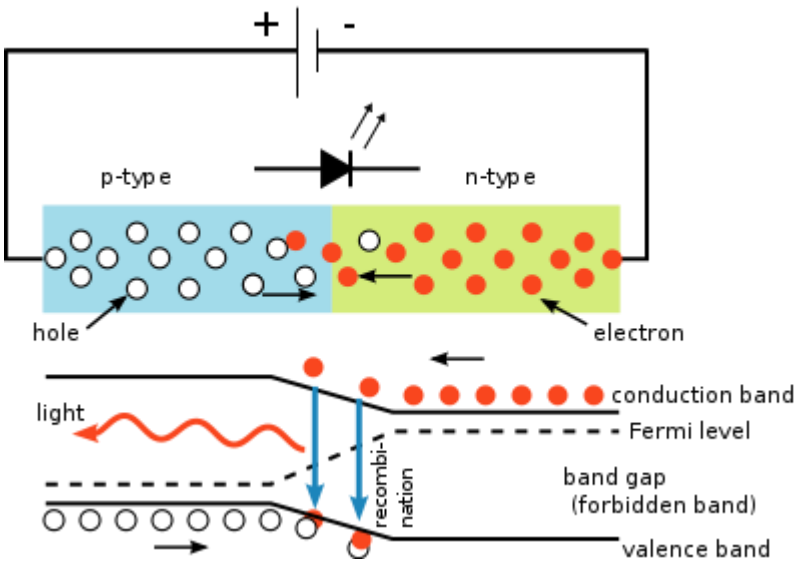

Fig 2.5 Working of LED

Chapter 3 HARDWARE INTERFACING

3.1 BLOCK DIAGRAM:

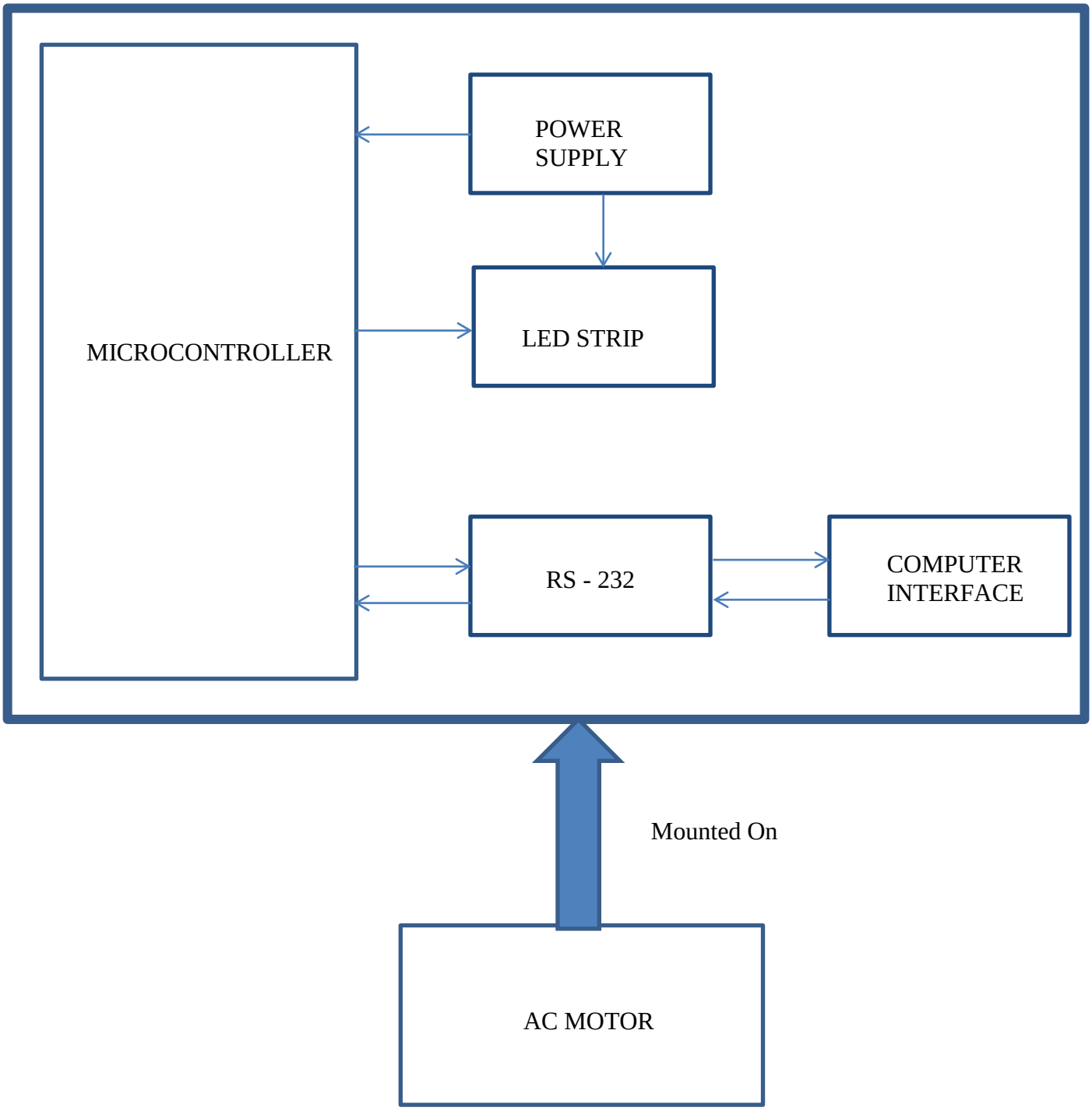

Fig 3.1: Block Diagram

3.2 CIRCUIT DIAGRAM:

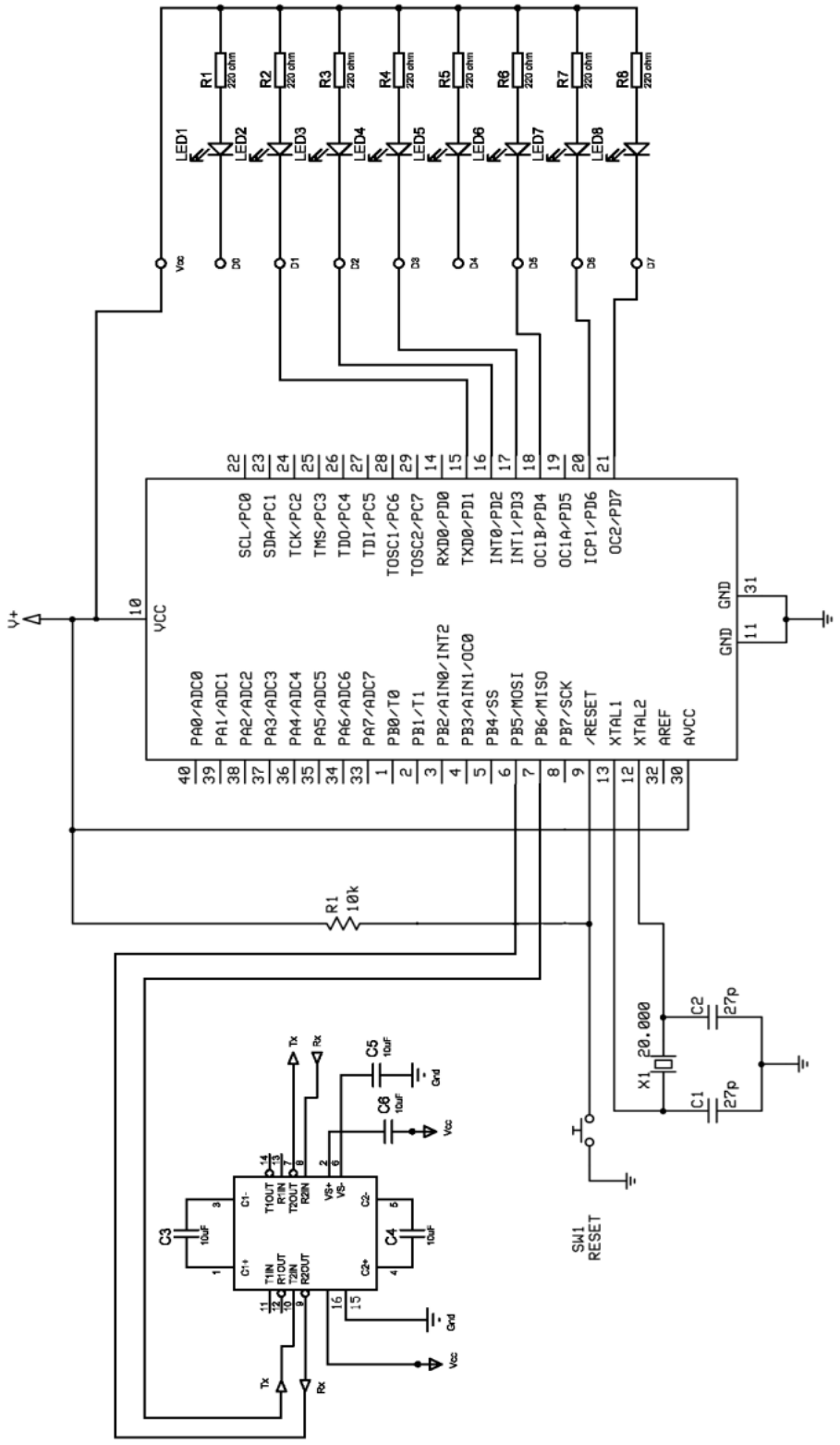

Fig 3.2: Circuit Diagram

3.3 Development Board Used

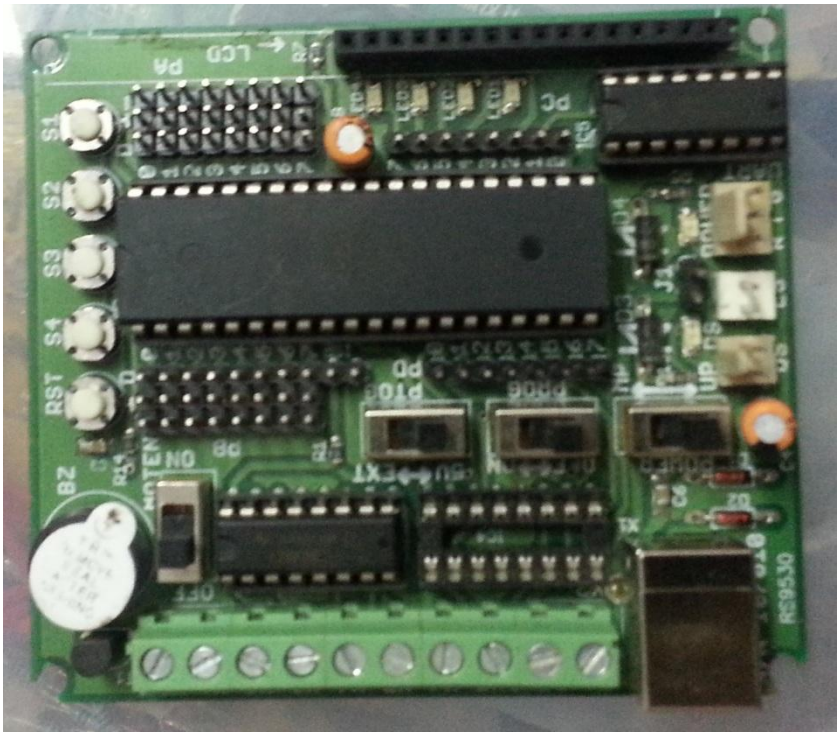

Fig 3.3: Development Board

3.4 PCB for LED Strip

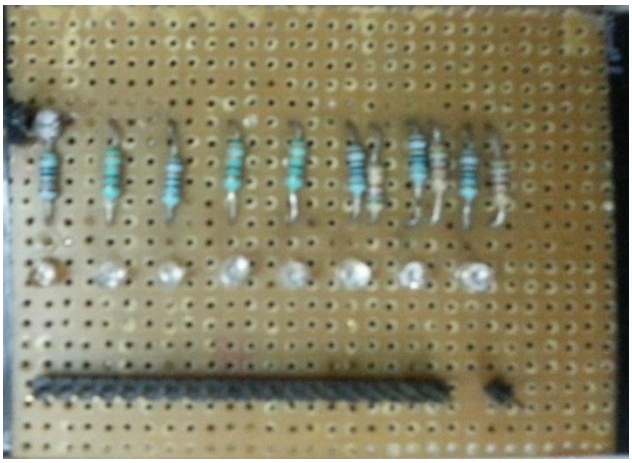

Fig 3.4: PCB for LED Strip

Chapter 4.PROGRAMMING

4.1 CALCULATIONS

Speed of AC motor	= 800 rpm
time for one rotation	= 1/800 min
	= 60/800 sec
	= (60*1000/800) msec
	= 75 msec

So, **one rotation takes 75 milli seconds.**

length from axle to start of first LED	= 11.5 cm
distance between the start of first LED and end of last LED	= 5.5 cm
radius	= (11.5 + 5.5) cm
	= 17 cm
perimeter	=(2 * 22 * 17 / 7) cm
	= 106.86 cm
width of led	= 0.3 cm
total no of columns (led)	= 106.86 / 0.3
	= 354

So, **1 rotation includes 354 times LED display.**

354 LED column display time	= 75 msec
1 LED column time	= 200.86 usec (200 usec)

So, **delay time to be used between 2 successive delays = 200 micro seconds**

LED columns used for 1 letter	= 11
So, length of 1 letter	=(11 * 0.3) cm
	= 3.3 cm
total letters possible	= 106.86 / 3.3
	= 32.38 (32)

So, **one time display can have maximum 32 characters.**

4.2 SOURCE CODE

```
#include <avr/io.h>             // header file for AVR Microcontroller
#include "delay.h"              // Header file to use delay functions

void display (char);            // Function to create a character display

int main (void)
{
        PORTD=0x00;                     // PORTD initialisation all bits low
        DDRD=0xFF;          // Data Direction of PORTD as output

        char text[20]="SHORYA DEEPU NIRAJ";  //String to be displayed
        int LED_COL = 354;
        inti,sum,count;

        while(1)                // infinte loop
        {
                i=0;
                sum=0;

                while(text[i] != '\0')
                {
                        if (text[i] == ' ')
                                sum += 44;
                        else
                                sum += 11;
                        display (text[i]);
                        i++;
                }

                count = 0;\

                while (count< (LED_COL-sum))
                {
                        PORTD=0xFF;
                        delayus(220);
                        count++;
                }

        }
}
```

```
void display (char ch)
{
switch (ch)
                {

                        case 'A':       PORTD=0x81;         delayus(220);
                                        PORTD=0x6F;         delayus(220);
                                        PORTD=0x6F;         delayus(220);
                                        PORTD=0x6F;         delayus(220);
                                        PORTD=0x6F;         delayus(220);
                                        PORTD=0x6F;         delayus(220);
                                        PORTD=0x91;         delayus(220);

                                        PORTD=0xFF;         delayus(220);
                                        PORTD=0xFF;         delayus(220);
                                        PORTD=0xFF;         delayus(220);
                                        PORTD=0xFF;         delayus(220);
                                        break;


                        case 'B':       PORTD=0x01;         delayus(220);
                                        PORTD=0x6D;         delayus(220);
                                        PORTD=0x6D;         delayus(220);
                                        PORTD=0x6D;         delayus(220);
                                        PORTD=0x6D;         delayus(220);
                                        PORTD=0x6D;         delayus(220);
                                        PORTD=0x93;         delayus(220);

                                        PORTD=0xFF;         delayus(220);
                                        PORTD=0xFF;         delayus(220);
                                        PORTD=0xFF;         delayus(220);
                                        PORTD=0xFF;         delayus(220);
                                        break;


                        case 'C':       PORTD=0x01;         delayus(220);
                                        PORTD=0x7D;         delayus(220);
                                        PORTD=0x7D;         delayus(220);
                                        PORTD=0x7D;         delayus(220);
                                        PORTD=0x7D;         delayus(220);
                                        PORTD=0x7D;         delayus(220);
                                        PORTD=0x7D;         delayus(220);

                                        PORTD=0xFF;         delayus(220);
```

```
                PORTD=0xFF;     delayus(220);
                PORTD=0xFF;     delayus(220);
                PORTD=0xFF;     delayus(220);
                break;

case 'D':       PORTD=0x01;     delayus(220);
                PORTD=0x7D;     delayus(220);
                PORTD=0x7D;     delayus(220);
                PORTD=0x7D;     delayus(220);
                PORTD=0x7D;     delayus(220);
                PORTD=0xB7;     delayus(220);
                PORTD=0xCF;     delayus(220);

                PORTD=0xFF;     delayus(220);
                PORTD=0xFF;     delayus(220);
                PORTD=0xFF;     delayus(220);
                PORTD=0xFF;     delayus(220);
                break;

case 'E':       PORTD=0x01;     delayus(220);
                PORTD=0x6D;     delayus(220);
                PORTD=0x6D;     delayus(220);
                PORTD=0x6D;     delayus(220);
                PORTD=0x6D;     delayus(220);
                PORTD=0x6D;     delayus(220);
                PORTD=0x6D;     delayus(220);

                PORTD=0xFF;     delayus(220);
                PORTD=0xFF;     delayus(220);
                PORTD=0xFF;     delayus(220);
                PORTD=0xFF;     delayus(220);
                break;

case 'F':       PORTD=0x01;     delayus(220);
                PORTD=0x6F;     delayus(220);
                PORTD=0x6F;     delayus(220);
                PORTD=0x6F;     delayus(220);
                PORTD=0x6F;     delayus(220);
                PORTD=0x6F;     delayus(220);
                PORTD=0x6F;     delayus(220);
```

```
                PORTD=0xFF;     delayus(220);
                PORTD=0xFF;     delayus(220);
                PORTD=0xFF;     delayus(220);
                PORTD=0xFF;     delayus(220);
                break;

case 'G':       PORTD=0x01;     delayus(220);
                PORTD=0x7D;     delayus(220);
                PORTD=0x7D;     delayus(220);
                PORTD=0x6D;     delayus(220);
                PORTD=0x6D;     delayus(220);
                PORTD=0x6D;     delayus(220);
                PORTD=0x61;     delayus(220);

                PORTD=0xFF;     delayus(220);
                PORTD=0xFF;     delayus(220);
                PORTD=0xFF;     delayus(220);
                PORTD=0xFF;     delayus(220);
                break;

case 'H':       PORTD=0x01;     delayus(220);
                PORTD=0xEF;     delayus(220);
                PORTD=0xEF;     delayus(220);
                PORTD=0xEF;     delayus(220);
                PORTD=0xEF;     delayus(220);
                PORTD=0xEF;     delayus(220);
                PORTD=0x01;     delayus(220);

                PORTD=0xFF;     delayus(220);
                PORTD=0xFF;     delayus(220);
                PORTD=0xFF;     delayus(220);
                PORTD=0xFF;     delayus(220);
                break;

case 'I':PORTD=0x7D;    delayus(220);
                PORTD=0x7D;     delayus(220);
                PORTD=0x7D;     delayus(220);
                PORTD=0x01;     delayus(220);
                PORTD=0x7D;     delayus(220);
                PORTD=0x7D;     delayus(220);
                PORTD=0x7D;     delayus(220);
```

```
                PORTD=0xFF;     delayus(220);
                PORTD=0xFF;     delayus(220);
                PORTD=0xFF;     delayus(220);
                PORTD=0xFF;     delayus(220);
                break;

case 'J':       PORTD=0x7F;     delayus(220);
                PORTD=0x77;     delayus(220);
                PORTD=0x7B;     delayus(220);
                PORTD=0x7D;     delayus(220);
                PORTD=0x7D;     delayus(220);
                PORTD=0x7D;     delayus(220);
                PORTD=0x01;     delayus(220);

                PORTD=0xFF;     delayus(220);
                PORTD=0xFF;     delayus(220);
                PORTD=0xFF;     delayus(220);
                PORTD=0xFF;     delayus(220);
                break;

case 'K':       PORTD=0x01;     delayus(220);
                PORTD=0xCF;     delayus(220);
                PORTD=0xB7;     delayus(220);
                PORTD=0x7B;     delayus(220);
                PORTD=0xFD;     delayus(220);
                PORTD=0xFF;     delayus(220);
                PORTD=0xFF;     delayus(220);

                PORTD=0xFF;     delayus(220);
                PORTD=0xFF;     delayus(220);
                PORTD=0xFF;     delayus(220);
                PORTD=0xFF;     delayus(220);
                break;

case 'L':       PORTD=0x01;     delayus(220);
                PORTD=0xFD;     delayus(220);
                PORTD=0xFD;     delayus(220);
                PORTD=0xFD;     delayus(220);
                PORTD=0xFD;     delayus(220);
                PORTD=0xFD;     delayus(220);
```

```
                PORTD=0xFD;     delayus(220);

                PORTD=0xFF;     delayus(220);
                PORTD=0xFF;     delayus(220);
                PORTD=0xFF;     delayus(220);
                PORTD=0xFF;     delayus(220);
                break;

case 'M':       PORTD=0x01;     delayus(220);
                PORTD=0xBF;     delayus(220);
                PORTD=0xDF;     delayus(220);
                PORTD=0xEF;     delayus(220);
                PORTD=0xDF;     delayus(220);
                PORTD=0xBF;     delayus(220);
                PORTD=0x01;     delayus(220);

                PORTD=0xFF;     delayus(220);
                PORTD=0xFF;     delayus(220);
                PORTD=0xFF;     delayus(220);
                PORTD=0xFF;     delayus(220);
                break;

case 'N':       PORTD=0x01;     delayus(220);
                PORTD=0xBF;     delayus(220);
                PORTD=0xDF;     delayus(220);
                PORTD=0xEF;     delayus(220);
                PORTD=0xF7;     delayus(220);
                PORTD=0xFB;     delayus(220);
                PORTD=0x01;     delayus(220);

                PORTD=0xFF;     delayus(220);
                PORTD=0xFF;     delayus(220);
                PORTD=0xFF;     delayus(220);
                PORTD=0xFF;     delayus(220);
                break;

case 'O':       PORTD=0x01;     delayus(220);
                PORTD=0x7D;     delayus(220);
                PORTD=0x7D;     delayus(220);
                PORTD=0x7D;     delayus(220);
                PORTD=0x7D;     delayus(220);
```

```
                PORTD=0x7D;     delayus(220);
                PORTD=0x01;     delayus(220);

                PORTD=0xFF;     delayus(220);
                PORTD=0xFF;     delayus(220);
                PORTD=0xFF;     delayus(220);
                PORTD=0xFF;     delayus(220);
                break;

case 'P':       PORTD=0x01;     delayus(220);
                PORTD=0x6F;     delayus(220);
                PORTD=0x6F;     delayus(220);
                PORTD=0x6F;     delayus(220);
                PORTD=0x6F;     delayus(220);
                PORTD=0x6F;     delayus(220);
                PORTD=0x0F;     delayus(220);

                PORTD=0xFF;     delayus(220);
                PORTD=0xFF;     delayus(220);
                PORTD=0xFF;     delayus(220);
                PORTD=0xFF;     delayus(220);
                break;

case 'Q':       PORTD=0x01;     delayus(220);
                PORTD=0x7D;     delayus(220);
                PORTD=0x7D;     delayus(220);
                PORTD=0x75;     delayus(220);
                PORTD=0x79;     delayus(220);
                PORTD=0x01;     delayus(220);
                PORTD=0xFE;     delayus(220);

                PORTD=0xFF;     delayus(220);
                PORTD=0xFF;     delayus(220);
                PORTD=0xFF;     delayus(220);
                PORTD=0xFF;     delayus(220);
                break;

case 'R':       PORTD=0x01;     delayus(220);
                PORTD=0x5F;     delayus(220);
                PORTD=0x4F;     delayus(220);
                PORTD=0x57;     delayus(220);
```

```
                PORTD=0x5B;     delayus(220);
                PORTD=0x5D;     delayus(220);
                PORTD=0x1F;     delayus(220);

                PORTD=0xFF;     delayus(220);
                PORTD=0xFF;     delayus(220);
                PORTD=0xFF;     delayus(220);
                PORTD=0xFF;     delayus(220);
                break;

case 'S':       PORTD=0x0D;     delayus(220);
                PORTD=0x6D;     delayus(220);
                PORTD=0x6D;     delayus(220);
                PORTD=0x6D;     delayus(220);
                PORTD=0x6D;     delayus(220);
                PORTD=0x6D;     delayus(220);
                PORTD=0x61;     delayus(220);

                PORTD=0xFF;     delayus(220);
                PORTD=0xFF;     delayus(220);
                PORTD=0xFF;     delayus(220);
                PORTD=0xFF;     delayus(220);
                break;

case 'T':       PORTD=0x7F;     delayus(220);
                PORTD=0x7F;     delayus(220);
                PORTD=0x7F;     delayus(220);
                PORTD=0x01;     delayus(220);
                PORTD=0x7F;     delayus(220);
                PORTD=0x7F;     delayus(220);
                PORTD=0x7F;     delayus(220);

                PORTD=0xFF;     delayus(220);
                PORTD=0xFF;     delayus(220);
                PORTD=0xFF;     delayus(220);
                PORTD=0xFF;     delayus(220);
                break;

case 'U':       PORTD=0x01;     delayus(220);
                PORTD=0xFD;     delayus(220);
                PORTD=0xFD;     delayus(220);
```

```
                PORTD=0xFD;     delayus(220);
                PORTD=0xFD;     delayus(220);
                PORTD=0xFD;     delayus(220);
                PORTD=0x01;     delayus(220);

                PORTD=0xFF;     delayus(220);
                PORTD=0xFF;     delayus(220);
                PORTD=0xFF;     delayus(220);
                PORTD=0xFF;     delayus(220);
                break;

case 'V':       PORTD=0x3F;     delayus(220);
                PORTD=0xCF;     delayus(220);
                PORTD=0xF3;     delayus(220);
                PORTD=0xFC;     delayus(220);
                PORTD=0xF3;     delayus(220);
                PORTD=0xCF;     delayus(220);
                PORTD=0x3F;     delayus(220);

                PORTD=0xFF;     delayus(220);
                PORTD=0xFF;     delayus(220);
                PORTD=0xFF;     delayus(220);
                PORTD=0xFF;     delayus(220);
                break;

case 'W':       PORTD=0x01;     delayus(220);
                PORTD=0xFB;     delayus(220);
                PORTD=0xF7;     delayus(220);
                PORTD=0xEF;     delayus(220);
                PORTD=0xF7;     delayus(220);
                PORTD=0xFB;     delayus(220);
                PORTD=0x01;     delayus(220);

                PORTD=0xFF;     delayus(220);
                PORTD=0xFF;     delayus(220);
                PORTD=0xFF;     delayus(220);
                PORTD=0xFF;     delayus(220);
                break;

case 'X':       PORTD=0x7D;     delayus(220);
                PORTD=0xBB;     delayus(220);
```

```
                PORTD=0xD7;     delayus(220);
                PORTD=0xEF;     delayus(220);
                PORTD=0xD7;     delayus(220);
                PORTD=0xBB;     delayus(220);
                PORTD=0x7D;     delayus(220);

                PORTD=0xFF;     delayus(220);
                PORTD=0xFF;     delayus(220);
                PORTD=0xFF;     delayus(220);
                PORTD=0xFF;     delayus(220);
                break;


case 'Y':       PORTD=0x7F;     delayus(220);
                PORTD=0xBF;     delayus(220);
                PORTD=0xDF;     delayus(220);
                PORTD=0xE1;     delayus(220);
                PORTD=0xDF;     delayus(220);
                PORTD=0xBF;     delayus(220);
                PORTD=0x7F;     delayus(220);

                PORTD=0xFF;     delayus(220);
                PORTD=0xFF;     delayus(220);
                PORTD=0xFF;     delayus(220);
                PORTD=0xFF;     delayus(220);
                break;


case 'Z':       PORTD=0x7D;     delayus(220);
                PORTD=0x79;     delayus(220);
                PORTD=0x75;     delayus(220);
                PORTD=0x6D;     delayus(220);
                PORTD=0x5D;     delayus(220);
                PORTD=0x3D;     delayus(220);
                PORTD=0x7D;     delayus(220);

                PORTD=0xFF;     delayus(220);
                PORTD=0xFF;     delayus(220);
                PORTD=0xFF;     delayus(220);
                PORTD=0xFF;     delayus(220);
                break;

default:        PORTD=0xFF;     delayus(220);
                PORTD=0xFF;     delayus(220);
```

```
PORTD=0xFF;     delayus(220);
PORTD=0xFF;     delayus(220);
PORTD=0xFF;     delayus(220);
PORTD=0xFF;     delayus(220);
PORTD=0xFF;     delayus(220);

PORTD=0xFF;     delayus(220);
PORTD=0xFF;     delayus(220);
PORTD=0xFF;     delayus(220);
PORTD=0xFF;     delayus(220);


PORTD=0xFF;     delayus(220);
PORTD=0xFF;     delayus(220);
PORTD=0xFF;     delayus(220);
PORTD=0xFF;     delayus(220);
PORTD=0xFF;     delayus(220);
PORTD=0xFF;     delayus(220);
PORTD=0xFF;     delayus(220);

PORTD=0xFF;     delayus(220);
PORTD=0xFF;     delayus(220);
PORTD=0xFF;     delayus(220);
PORTD=0xFF;     delayus(220);


PORTD=0xFF;     delayus(220);
PORTD=0xFF;     delayus(220);
PORTD=0xFF;     delayus(220);
PORTD=0xFF;     delayus(220);
PORTD=0xFF;     delayus(220);
PORTD=0xFF;     delayus(220);
PORTD=0xFF;     delayus(220);

PORTD=0xFF;     delayus(220);
PORTD=0xFF;     delayus(220);
PORTD=0xFF;     delayus(220);
PORTD=0xFF;     delayus(220);

PORTD=0xFF;     delayus(220);
PORTD=0xFF;     delayus(220);
PORTD=0xFF;     delayus(220);
PORTD=0xFF;     delayus(220);
PORTD=0xFF;     delayus(220);
```

```
                    PORTD=0xFF;     delayus(220);
                    PORTD=0xFF;     delayus(220);

                    PORTD=0xFF;     delayus(220);
                    PORTD=0xFF;     delayus(220);
                    PORTD=0xFF;     delayus(220);
                    PORTD=0xFF;     delayus(220);
                    break;
            }

}
```

Chapter 5. OUTPUTS CATALOGUE

5.1 Final Setup

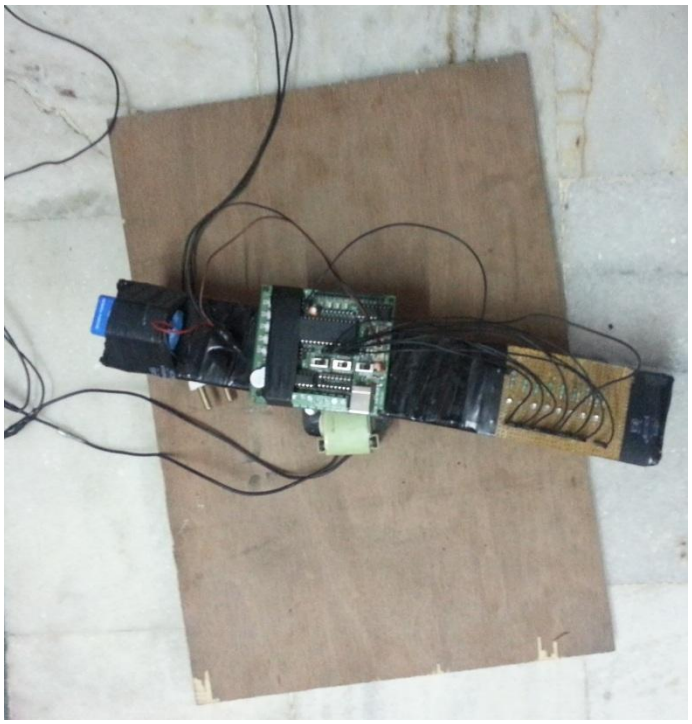

Fig 5.1: Final Setup of Project

5.2 Outputs Displaying Different Texts

1. “PROPELLER DISPLAY”

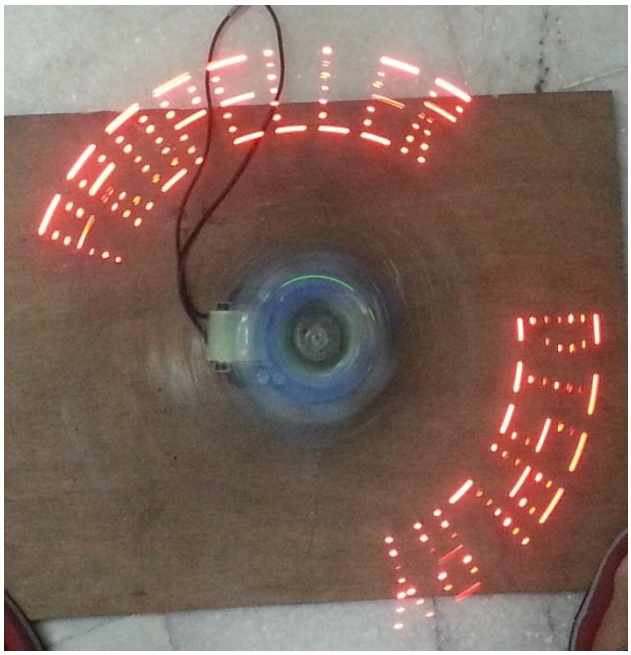

Fig 5.2: Output showing “PROPELLER DISPLAY”

2. “MENTOR BK SINGH”

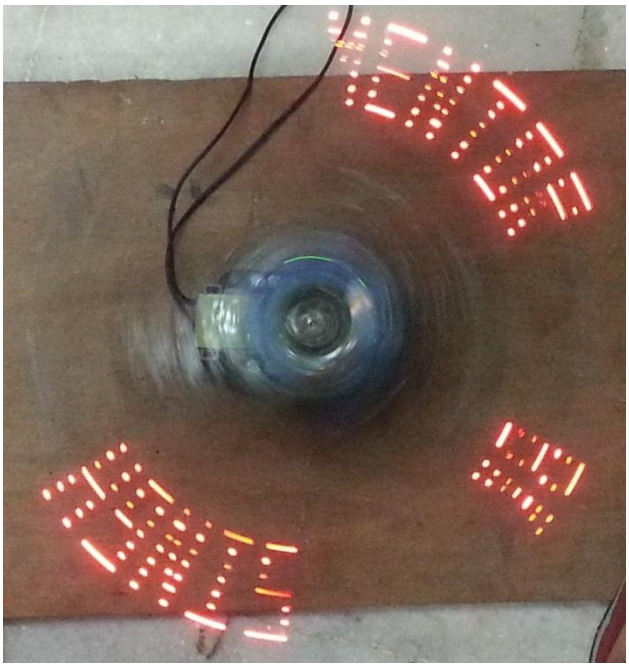

Fig 5.3 Output showing “MENTOR BK SINGH”

3. “SHORYA DEEPU NIRAJ”

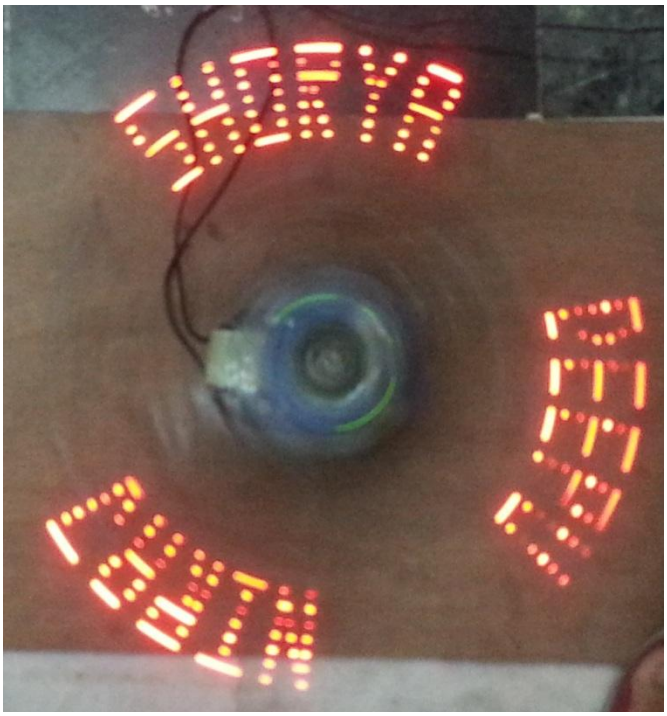

Fig 5.4: Output Showing “SHORYA DEEPU NIRAJ”

CONCLUSION

Our basic idea was based on this stereotyped concept of persistence of vision, where human does not bear the capability to distinguish an object if it is moving at high speed and an image just seen persists on his retina for 25^{th} or 30^{th} part of a second. This limitation of the human eye was something we could exploit.

An illustrative model was built to show this concept practically, where we thought of displaying a small set of LEDs for a time period much less than the time period for which an image persists on our retina, and continue this to build up whole text by that small LEDS set! We worked out on displaying it on the circumference of a circle created by mounting a wooden piece on the axle of ac motor. Various forms of design were worked upon to get the required angular velocity and to stabilize the propeller. The PCB, LED strip and the battery had to be perfectly balanced on either side in order to have maximum rotating speed and also avoid their detachment from the system when they experience a large centripetal force because of high torque of ac motor. Timings for the glowing LEDs also had to be right while rotating. We had to make sure that the output is revolving around with a suitable speed such that text can be easily read by human. We have been able to justify all our thoughts practically by completing this project successfully.

An interesting future study and further implementation might involve the usage of this Propeller Display in areas like Railway station, Airports etc making it easy to be read from every possible angle, eliminating the needs of installing multiple LED displays at different angles. Also proper research on design may help to bring down the cost of this project so as to make it much more feasible.

REFERENCES

1. Microcontroller based control system for rotating display with multiplexed LED array: Lita, I. (Comp. & Elec. Eng. Dept., Univ. of Pitesti, Pitesti, Romania)-IEEE; Visan, D.A. – IEEE
   Published in: Electronics, Computers and Artificial Intelligence (ECAI), 2013 International Conference (27-29 June 2013).

2. Embedded C Programming and the Atmel AVR, Richard H Barnett

3. http://en.wikipedia.org/wiki/LED

4. http://en.wikipedia.org/wiki/AC_motor

5. http://www.technophilia.co.in/

6. http://www.alldatasheet.com/datasheet-pdf/pdf/164169/ATMEL/ATMEGA8535.html

7. www.worldwideelectric.net/Data/DataSheet/HT600-36-5812.pdf

8. http://www.labcenter.com/index.cfm