Vidarbha Youth Welfare Society's
Prof. Ram Meghe Institute of Technology & Research Badnera, Amravati (M.S.) 444701

Laboratory Manual
Semester VIII

Subject: - Embedded System Lab-8KS06

Sant Gadge Baba Amravati University Amravati

Department of Computer Science & Engineering

Phone: (0721) - 2580402/2681246
Fax No. 0721 – 2681337
Website: www.mitra.ac.in

Vidarbha Youth Welfare Society's

Prof Ram Meghe Institute of Technology & Research

Badnera, Amravati (M.S.) 444701

CERTIFICATE

This is to certify that Mr/Miss ____
_________________Enrollment No._____________Roll No. _______________
Section _______ of B. E. Fourth Year Semester VIII Department of Computer Science & Engineering has satisfactory completed the term work of the subject **Embedded System Practical Lab** prescribed by Sant Gadge Baba Amravati University, Amravati during the academic year 2017-18.

Date:

Signature of the faculty

Department of Computer Science & Engineering

INDEX

1. Mission & Vision statement of the Institute:

VISION

To become a pace-setting
centre of excellence believing in three
universal values namely
Synergy, Trust and Passion,
with zeal to serve the Nation
in the global scenario

MISSION

To dedicate ourselves
to the highest standard of technical education
& research in core & emerging engineering
disciplines and strive for the overall personality
development of students so as to nurture
not only quintessential technocrats
but also responsible citizens

Mission and Vision of the Department

Vision

To ensure that the world saves time and other depletable resources and free it from complexity by providing efficient computing services.

Mission

To dedicate ourselves to the highest standard by providing knowledge, skills and wisdom to the incumbent by imparting value based education to enable them to solve complex system by simple algorithms and to make them innovative, research oriented to serve the global society, while imbibing highest ethical values.

2.Programme Educational Objectives (PEOs)

1. **Preparation:** To prepare students for successful careers in software industry that meet the needs of Indian and multinational companies or to excel in Higher studies.
2. **Core competence:** To develop the ability among students to synthesize data and technical concepts for software design and development.
3. Breadth: To inculcate in students professional and ethical attitude, effective communication skills, teamwork skills, multidisciplinary approach and an ability to relate engineering issues to broader social context.
4. **Professionalism:** To provide students with a sound foundation in the mathematical, scientific and computer engineering fundamentals required to solve engineering problems and also peruse higher studies.
5. **Learning Environment:** To promote student with an academic environment aware of excellence, leadership, written ethical codes and guidelines and the life-long learning needed for a successful to professional career.

PROGRAM OUTCOMES (POs)

Engineering Graduates will be able to:

1. **Engineering knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis**: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design/development of solutions**: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems**: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of

the information to provide valid conclusions.

5. **Modern tool usage**: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
6. **The engineer and society**: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and sustainability**: Understand the impact of the professional engineering solutions in social and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics**: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work**: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication**: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **Project management and finance**: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. **Life-long learning**: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

PROGRAM SPECIFIC OUTCOMES (PSO's)

A graduate of the Computer Science and Engineering Program will demonstrate:

1. **Foundation of Computer System**: Ability to understand the principles and working, assembly of computer systems. Students can understand the hardware and software installation as well as up-gradation of computer systems.

2. **Foundations of Software development:** Ability to understand the problem structure and development methodologies of software systems. Possess professional problem solving skill and knowledge of software design process. By imparting ethical values and practical competence with a broad range of programming language and open source platforms

3. **Entrepreneurship and Research Ability**: Ability to use knowledge in various domains to identify problems and research gaps and hence to provide solution to new ideas and innovations by imbibing highest ethical values.

3. Subject Name and Code:	**Embedded Systems (8KS02)**
Faculty Name:	P.K.Agrawal (pkagrawal@mitra.ac.in)
Course Type:	Theory + Lab
Compulsory / Elective:	Compulsory
Teaching Methods:	**Lecture:** 04 Hrs/ week. **Laboratory:** 02 Hrs/ week (03 Batches) **Credits:** 04 Th + 01 Lab (as per given in syllabus). **Discussion:** Office hour discussion, Quiz.
Course Assessment:	**Homework:** Consisting of assignments. **Exams:** 2 Class Tests + 1 Improvement Test Semester end examination by SGBAU
Grading Policy:	**20%** Assignments, class test and viva-voce **80%** Semester end examinations
Prerequisite	Assembly Language Programming Computer Organization Computer Architecture
Unit wise Course Content:	**Unit 1:** Introduction to Embedded System: Embedded Systems Vs General Computing Systems. History, classification, major application areas and purpose of Embedded Systems. Components of Embedded system: General Purpose and Domain Specific Processors, Memories for embedded systems. **Unit 2:** Components of Embedded system: Sensors & Actuators, Communication Interface, Embedded Firmware and other components. Characteristics of Embedded System, Quality Attributes of Embedded System. Embedded Systems Examples: Washing machine. Automotive application. **Unit 3:** Introduction to 8051 Microcontroller: 8051 Architecture, 8051 Memory Organization, Registers, Oscillator Unit, Ports, 8051 Interrupt System, Timer units, the Serial Port, 8051 Power Saving Modes. **Unit 4:** Programming the 8051 Microcontroller: Addressing modes. 8051 Instruction Set: Data transfer instructions, Arithmetic instructions, Logical instructions, Boolean instructions, and Program Control Transfer

instructions. Assembly Language based Embedded Firmware development.

Unit 5: Programming in Embedded C: Review of various constructs in C. Constant declarations, ‘volatile’ type qualifier, Delay generation and Infinite loops in Embedded C. Coding Interrupt Service Routines, Recursive and Re-entrant Functions, Dynamic memory allocation.

Unit 6: Vx Works Real Time Operating System (RTOS): , Real Time Kernel, Hard/Soft Real time. VxWorks Task Creation, Management and Task Scheduling, Kernel Services, Inter Task Communication, VxWorks Task Synchronization and Mutual Exclusion, Interrupt Handling, Watchdog for task Execution monitoring, Timing and Reference in Vx Works.

Text Book: Shibu K V “Introduction to Embedded Systems” McGraw-Hill.

Reference Books:

1. Rajkamal, “Embedded Systems, Architecture, Programming & Design” TMH.
2. Tammy Noergaard “Embedded Systems Architecture” Elsevier Newness Publication.
3. Vahid and Givargis “Embedded System Design” John Wiley & Sons P Ltd.
4. Peter Marwedel “Embedded Systems Design” Springer, Netherland.

Units covered in the course and contact hours per unit:	**Unit 1:** 8 hrs	**Unit 2:** 8 hrs	**Unit 3:** 8 Hrs
	Unit 4: 8 hrs	**Unit 5:** 8 hrs	**Unit 6:** 8 Hrs

Course Learning Objective:

K802.1. In this course, Student will learn basics of embedded system hardware with required programming language to make the electronic device to work according to the order.

K802.2. Learn about various hardware which is essential for any general purpose system and its components with having various attributes such as single 7-seg LED and multi-seg LED and development of embedded code for various system with different architecture.

K802.3. Understanding the architecture of embedded system, with its attributes. Assigning the required hardware for the existing system. Understanding the working of Timer, PORTs and other hardware with embedded system.

K802.4. Enabling the hardware to communicate with each other using inter process communication and various methods. Understanding the data transfer between two electronic devices.

K802.5. Introduce the student with programming in embedded 'C', with its advanced features. Enabling the embedded system with its connected devices using programming languages. Student will learn to develop the software for the upliftment of the standalone electronic device.

K802.6. Student will learn the Real Time system which works with the continuous connectivity and the continuous data generation. By monitoring such system such will understand the scheduling of the process and allotment of memory space available.

K806.1 Subject specific skills with ability to interface various components of Embedded kit with 8051 microcontroller.

Upon completion of this course, students will have had an opportunity to learn about the following:

	Course Specific Practical Outcomes	**Course Outcomes**
1	Understand the basics of Embedded System and its different architecture like Von -Nueman, Harvard architecture.	**K802.1, K802.2, K806.1**
2	Learn basics of domain specific controller and General Purpose Processor.	**K802.1, K802.2, K802.3, K806.1**
3	Design & implement the embedded program for output devices such as Single 7 segment display using embedded C programming	**K802.1, K802.2, K802.4, K806.1**
4	Able to understand the differentiable attribute of General Purpose System and Embedded System.	**K802.1, K802.2, K802.4, K806.1**
5	Design & implement the embedded programs on the buzzer and the usage of buzzer using embedded C Programming	**K802.1, K802.2, K802.4, K806.1**
6	Design & implement the program for 16x2 display using embedded C programming.	**K802.1, K802.2, K802.5, K806.1**
7	Design & implement the program for stepper motor which will rotates in clock wise or anticlockwise direction or in a certain formation	**K802.1, K802.2, K802.5, K806.1**
8	Able to understand the working of switch for what purpose it is used and where	**K802.1, K802.2, K802.5, K806.1**

9	Able to understand the requirement of PSW and its functionality in embedded system	**K802.1, K802.2, K802.5, K806.1**
10	Understand the matrix key terminology using embedded C Programming.	**K802.1, K802.2, K802.5, K806.1**
11	Understand to demonstrate the working of IR Remote using embedded C programing.	**K802.1, K802.2, K802.5, K806.1**
12	Able to understand the Delay generation in embedded system using C Programming	**K802.1, K802.2, K802.5, K806.1**
13	Understand the Hard real time system for the betterment of understanding the real time operating system.	**K802.1, K802.2, K802.6, K806.1**

Mapping of Course Specific Practical Outcomes with Course Outcomes

Embedded Systems	**K802.1, K802.2, K802.3, K802.4, K802.5, K802.6, K806.1**
Strong CO'S (≥70%)	**K802.1, K802.2, K802.5, K806.1**
Average CO's(<70)	**K802.4**
Somewhat CO's	**K802.3**

Mapping of Course Outcomes with Program Outcomes

Enter correlation levels 1, 2 or 3 as defined below:

1: Slight (Low) 2: Moderate (Medium) 3: Substantial (High)

Course	**PO 1**	**PO 2**	**PO 3**	**PO 4**	**PO 5**	**PO 6**	**PO 7**	**PO 8**	**PO 9**	**PO 10**	**PO 11**	**PO 12**
K802.1	2	2	2	2	3	1	-	2	-	2	1	3
K802.2	2	2	2	2	3	1	-	2	-	2	1	3
K802.3	2	2	3	3	2	3	-	3	-	-	-	3
K802.4	2	2	3	3	2	3	-	3	-	-	-	3
K802.5	2	2	3	3	2	3	-	3	-	-	-	3
K802.6	2	2	3	3	2	3	-	3	-	-	-	3

4. Assessment Format i.e. ACIPV (Guidelines)

Regular Assessment of Experiment during the practical session

(Sample Sheet)
Regular Assessment

Year:-
Branch:-

Assessment Sheet

Roll no.	**A (5marks)**	**C (5marks)**	**I (5marks)**	**P (5marks)**	**V (5marks)**	**Total (25Marks)**

(A-Attendance, C- Competency, I-Innovation, P-Presentation, V-Viva)

Guidelines for Awarding Internal Marks For Practical

Each experiment/ practical carries 25 marks. The student shall be evaluated for 25 marks as per the following guidelines. At the end of the semester, internal assessment marks for practical shall be the average of marks scored in all the experiments.

a. Attendance (5 marks): These 5 marks will be given to the regularity of a student. If the student is present on the scheduled time, he/ she will be awarded 5 marks otherwise will be given 2.5 marks for his attendance.

b. Competency (5 marks): Here the basic aim is to check whether the student has developed the skill to write down the code on his own and debug. If a student executes the practical on the scheduled day within the allocated time, he will be awarded full marks. Otherwise, marks will be given according to the level of completion/ execution of practical

c. Innovation (5 marks): Here the basic objective is to explore the innovative ideas from the students with respect to the corresponding practical or how innovatively he interprets the aim of the practical. It is expected that the students must be aware about the scope of practical precisely such that in future, they could implement the task in various applications.

d. Performance/Participation in-group activity (5 marks): These marks will be given on how a student is actively participating the group. If the student is performing the practical as the part of a group he must participate actively.

e. Viva-Voce (5 marks): These 5 marks will be totally based on the performance in viva-voce. There shall be viva-voce on the completion of each practical in the same practical session. The student shall get the record checked before next practical.

Mapping of Guidelines with PO's of the department:

Guidelines for Awarding Internal Marks For Practical	Program Outcomes
Attendance	**PO1, PO2, PO3, PO4, PO5, PO6, PO7, PO8, PO12**
Competency	**PO1, PO2, PO3, PO4, PO5, PO10, PO12**
Innovation	**PO1, PO2, PO3, PO4, PO5, PO11, PO12**
Performance/Participation in-group activity	**PO1, PO2, PO3, PO4, PO5,PO9, PO11, PO12**
Viva-Voce	**PO1, PO2, PO3, PO4, PO5, PO10, PO12**

5. About the Lab manual:

This manual is designed for the Fourth year (VIII Semester) students of Computer Science & Engineering. The aim of the Embedded System Lab is to develop a student's programming skills in embedded system their applications.

This lab manual can be used as instructional book for students, staff and instructors to assist in performing and understanding the practical. In the first part of the manual, practical as per syllabus are described and in the second part of the manual, practical that are beyond the syllabus but expected for university laboratory examination are displayed.

6. List of Practical's

Department of Computer Science & Engineering

Subject:-Embedded System Lab
(8KS06) Semester:-VIII

Sr. No.	LIST OF PRACTICALS	Page
1.	To study 8051 microcontroller and its peripherals.	
2.	Write and Execute a program for flashing LED's	
3.	Write and execute a program to display numbers with the help of single 7-segment display	
4.	Write and execute a program to display numbers with the help of multiple 7-segment display	
5.	Write and execute a program to turn the buzzer ON and OFF	
6.	Write and execute a program for display the character using the 16 X 2 LCD display	
7.	Write and execute a program for rotating of stepper motor in clockwise and anticlockwise using Full Drive	
8.	Write and execute a program for rotating of stepper motor in clockwise and anticlockwise using Wave Drive	
9	Write and execute a program to demonstrate the Relay function	
10.	Write and execute a program to demonstrate the key technology on matrix	
11.	Write a Program to Demonstrate Switch.	
12.	Write a Program to Demonstrate PWM.	
13.	Write a Program to Demonstrate IR Remote.	
14.	To Study Vx Works Operating System	

Sr. No.	***LIST OF EXPERIMENTS*** ***(Actual Conducted For the Session)***	**Date**	**Page No.**	**Remark**
1.	To study 8051 microcontroller and its peripherals.			
2.	Write and Execute a program for flashing LED's using			
3.	Write and execute a program to display numbers with the help of single 7-segment display			
4.	Write and execute a program to display numbers with the help of multiple 7-segment display			
5.	Write and execute a program to turn the buzzer ON and OFF using			
6.	Write and execute a program for display the character using the 16 X 2 LCD display			
7.	Write and execute a program for rotating of stepper motor in clockwise and anticlockwise using Wave Drive			
8.	Write and execute a program for rotating of stepper motor in clockwise and anticlockwise in full Drive			
9.	Write and execute a program to demonstrate the Relay function.			
10.	Write and execute a program to demonstrate the key technology on matrix			

Practical No. 01

Aim: To study 8051 microcontroller and its peripherals.

Theory:

1. Introduction

- A highly integrated silicon chip containing a CPU, scratch pad RAM, Special and General purpose Register Arrays, On Chip ROM/FLASH memory for program storage, Timer and Interrupt control units and dedicated I/O ports
- Microcontrollers can be considered as a super set of Microprocessors
- Microcontroller can be general purpose (like Intel 8051, designed for generic applications and domains) or application specific (Like Automotive AVR from Atmel Corporation. Designed specifically for automotive applications)
- Since a microcontroller contains all the necessary functional blocks for independent working, they found greater place in the embedded domain in place of microprocessors
- Microcontrollers are cheap, cost effective and are readily available in the market

Factors to be considered in Microcontroller Selection

- Feature Set
    - Speed of Operation
    - Code Memory space
    - Data Memory Space
    - Development Support
    - Availability
    - Power Consumption
    - Cost

2. Block diagram 8051 MC

The following illustration shows the block diagram of an 8051 microcontroller −

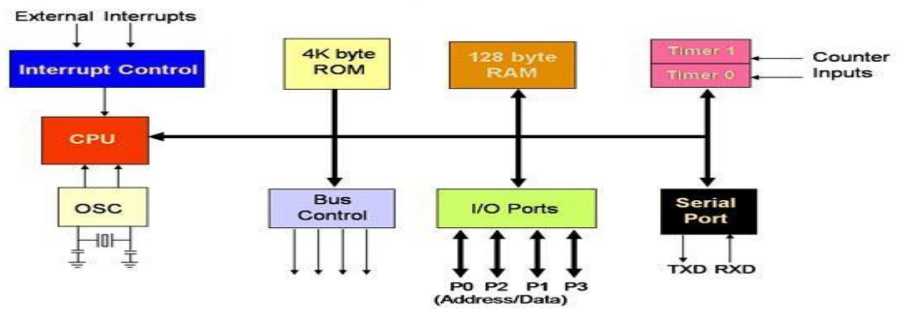


Figure1.1: Block Diagram of 8051 Microcontroller

3. Architecture Diagram of 8051MC

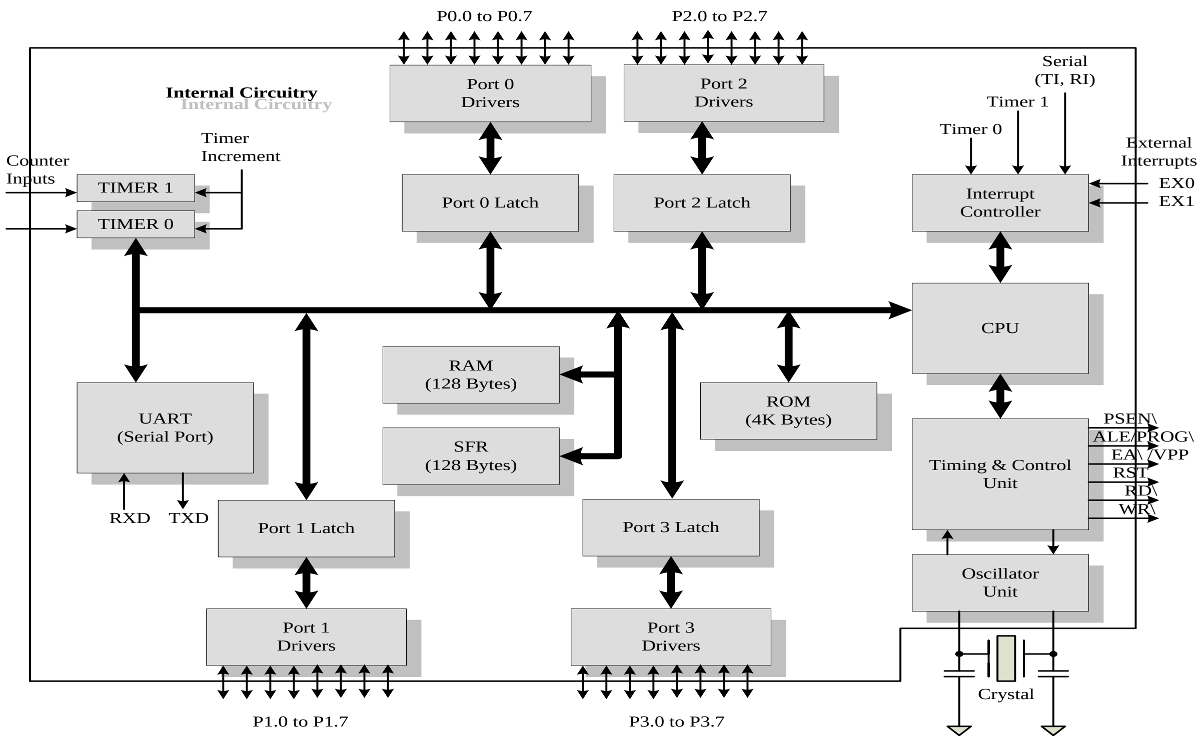


Figure 1.2: Architecture of 8051 microcontroller

4. Feature of 8051 MC

- Built around an 8bit CPU with Boolean processing capability
- Built-in oscillator driver unit
- 4K bytes of On-chip Program memory
- 128 bytes of internal Data memory
- 128 bytes of Special Function Register memory area
- 32 general purpose I/O lines organized into four 8bit bi-directional ports
- Two 16 bit timer units
- A full duplex programmable UART for serial data transmission with configurable baudrates
- Built around the ‘*Harvard*’ processor architecture
- The program and data memory of 8051 is logically separated and they physically reside separately
- Separate address spaces are assigned for data memory and program memory
- 16 bit wide address bus which can address code memory up to 64KB (2^{16})

Conclusion:

____

____

____

____

Viva Question:

1) What is Embedded System?
2) What is task oriented system?
3) How are the ports utilized for the transfer of data?
4) Is it required that, embedded system must enabled with operating system.

ACHIEVED COURSE OUTCOMES:

At the end of this experiment the student will be able to:-

Sr. No	Course Outcomes	Mapping with Pos
1	Understanding the concepts of 8051 microcontroller and the its peripheral.	**PO1, PO2, PO3, PO4, PO5, PO10, PO12**
2	Understand the requirement of the task oriented devices	**PO1, PO2, PO3, PO4, PO5, PO10, PO12**

1	Program Course outcomes	**PO1, PO2, PO3, PO4, PO5, PO10,**
2	Strong POs(≥50%)	**PO1, PO2, PO3, PO4, PO5, PO10,**
3	Somewhat Pos	

Signature

Department of Computer Science & Engineering

Practical No. 02

Aim: Write and execute a program for flashing LED's.

Software Required: Kiel Version: 2 or 3, STC ISP

Theory:

A **light-emitting diode** (**LED**) is a two-lead semiconductor light source. It is a p–n junction diode that emits light when activated. When a suitable voltage is applied to the leads, electrons are able to recombine with electron 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. LEDs are typically small (less than 1 mm2) and integrated optical components may be used to shape the radiation pattern.

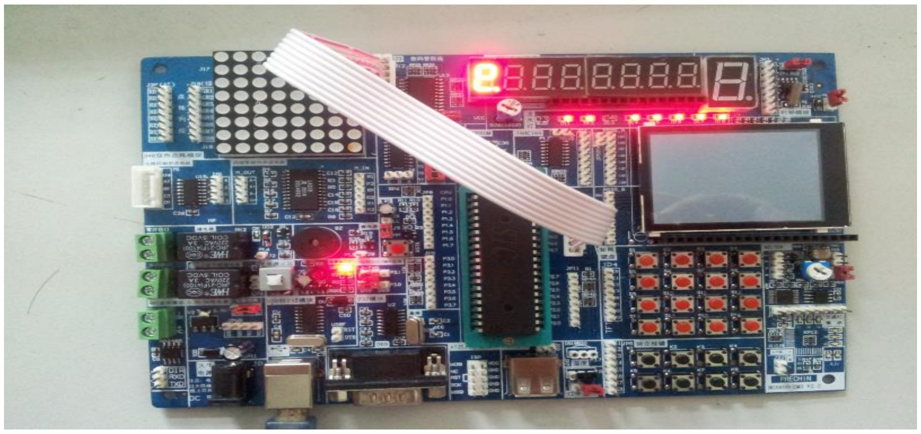
LEDs have many advantages over incandescent light sources, including lower energy consumption, longer lifetime, improved physical robustness, smaller size, and faster switching. Light-emitting diodes are used in applications as diverse as aviation lighting, automotive headlamps, advertising, general lighting, traffic signals, camera flashes, and lighted wallpaper. They are also significantly more energy efficient and, arguably, have fewer environmental concerns linked to their disposal.

LED Conn Seq (cross)		**A**	**B**	**C**	**D**	**E**	**F**	**G**	**H**
Sr.No	**HEX DEC**	**P7**	**P6**	**P5**	**P4**	**P3**	**P2**	**P1**	**P0**
1	0X80	1	0	0	0	0	0	0	0
2	0X40	0	1	0	0	0	0	0	0
3	0X20	0	0	1	0	0	0	0	0
4	0X10	0	0	0	1	0	0	0	0
5	0X08	0	0	0	0	1	0	0	0
6	0X04	0	0	0	0	0	1	0	0
7	0X02	0	0	0	0	0	0	1	0
8	0X01	0	0	0	0	0	0	0	1

PROGRAM:

```
#include <reg51.h>
void Delay10ms (unsigned int c);
void main ()
{
while (1)
        {
                        {
                                P1 = 0x80;
                                Delay10ms(50);
                                P1 = 0x40;
                                Delay10ms(50);
                                P1 = 0x20;
                                Delay10ms(50);
                                P1 = 0x10;
                                Delay10ms(50);
                                P1 = 0x08;
                                Delay10ms(50);
                                P1 = 0x04;
                                Delay10ms(50);
                                P1 = 0x02;
                                Delay10ms(50);
                                P1 = 0x01;
                                Delay10ms(50);
                        }
        }

}

void Delay10ms (unsigned int c)
{
   unsigned char a, b;
   for (; c> 0; c--)
        {
                {
                            for (b = 38; b> 0; b--)
                        {
                            for (a = 130; a> 0; a--);
                        }
                }

        }

}
```

OUTPUT:



Conclusion:

____

____

Viva Question:

1. Explain the Hexadecimal conversion of the binary no format.
2. Explain the sequence of generation of series in Led.

ACHIEVED COURSE OUTCOMES:

At the end of this experiment the student will be able to:-

Sr. No	Course Outcomes	Mapping with Pos
1	Students will be able to interface the programming with the hardware	**PO1, PO2, PO3, PO4, PO5, PO12**
2	Students will learn the hexadecimal code conversion.	**PO1, PO2, PO3, PO4, PO5, PO12**
3	Students will be able to generate the series of led sequence.	**PO1, PO2, PO3, PO4, PO5, PO12**
1	Program Course outcomes	**PO1, PO2, PO3, PO4, PO5, PO12**
2	Strong POs (≥50%)	**PO1, PO2, PO3, PO4, PO5, PO12**
3	Somewhat Pos	

Signature

Department of Computer Science & Engineering

Practical No. 03

Aim: Write a Program to Demonstrate Single Seven Segment Display.

Software Required: Kiel Version: 2 or 3, STC ISP

Theory:

Digital control is how to show it 1,2,3,4? Actually 7 LED component consisting of 8 characters, plus decimal point is 8. We were named as his A, B, C, D, E, F, G, H. if we want to show a figure 2, A, B, G, E, D That paragraph 5 of the LED light on it. That is, the B, E, H (decimal point) does not shine the rest light. According to the law then we have the hardware for the following procedure. Prior to this, of course, it is also necessary to specify which LED light; we are here for the appointment of the last P2.7.

From high to low rank, (p0.7_p0.0) is written in binary 01111110, 16, into his band for A2H. We can hardware connection to the digital display figures into a form after the call directly on the line

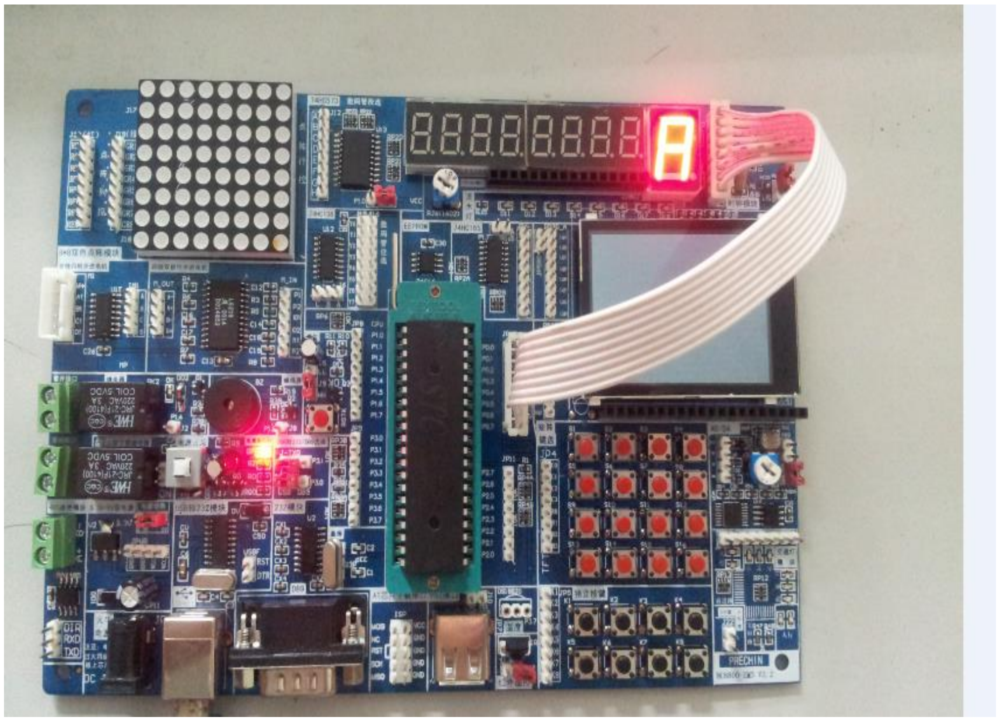
<u>Logic:</u>

LED Conn Seq			**H**	**G**	**F**	**E**	**D**	**C**	**B**	**A**
<u>Sr. No</u>	**<u>Number</u>**	**HEX DEC**	**P7**	**P6**	**P5**	**P4**	**P3**	**P2**	**P1**	**P0**
1	0	**0X40**	0	1	0	0	0	0	0	0
2	1	**0XF9**	1	1	1	1	1	0	0	1
3	2	**0X24**	0	0	1	0	0	1	0	0
4	3	**0X30**	0	0	1	1	0	0	0	0
5	4	**0X19**	1	0	0	1	0	0	0	1
6	5	**0X12**	0	0	0	1	0	0	1	0
7	6	**0X02**	0	0	0	0	0	0	1	0
8	7	**0XF8**	1	1	1	1	1	0	0	0
9	8	**0X00**	0	0	0	0	0	0	0	0
10	9	**0XF10**	0	0	0	1	0	0	0	0

Program:

```
#include <reg51.h>
void Delay10ms (unsigned int c);
void main ()
{
while (1)
          {
                    {

                              P1 = 0xF9;
                              Delay10ms(50);
                              P1 = 0x24;
                              Delay10ms(50);
                              P1 = 0x30;
                              Delay10ms(50);
                              P1 = 0x19;
                              Delay10ms(50);
                    }
          }

}
void Delay10ms (unsigned int c)
{
   unsigned char a, b;
   for (; c> 0; c--)
          {
               {
                         for (b = 38; b> 0; b--)
                    {
                         for (a = 130; a> 0; a--);
                    }
               }

          }

}
```

Output:



Conclusion:

Viva Questions:

1) How many cathode and anodes are available in single 7-segment display?
2) How encoding are performed to display ‘3‘ in a single 7 segment display?

ACHIEVED COURSE OUTCOMES:

At the end of this experiment the student will be able to:-

Sr.No	Course Outcomes	Mapping with POs
1	Understand the concept of 7-Segment display	**PO1, PO2, PO3, PO4, PO5, PO12**
2	Design and implement 7-Segment	**PO1, PO2, PO3, PO4, PO5, PO12**

1	Program Course outcomes	**PO1, PO2, PO3, PO4, PO5, PO12**
2	Strong POs(≥50%)	**PO1, PO2, PO3, PO4, PO5, PO12**
3	Somewhat Pos	

Signature

Department of Computer Science & Engineering

Practical No. 04

Aim: Write and execute a program to display numbers with the help of multiple seven segment display

Theory:

Digital control is how to show it 1,2,3,4? Actually 7 LED component consisting of 8 characters, plus decimal point is 8. We were named as his A, B, C, D, E, F, G, H. if we want to show a figure 2, A, B, G, E, D That paragraph 5 of the LED light on it. That is, the B, E, H (decimal point) does not shine the rest light. According to the law then we have the hardware for the following procedure. Prior to this, of course, it is also necessary to specify which LED light; we are here for the appointment of the last P2.7.

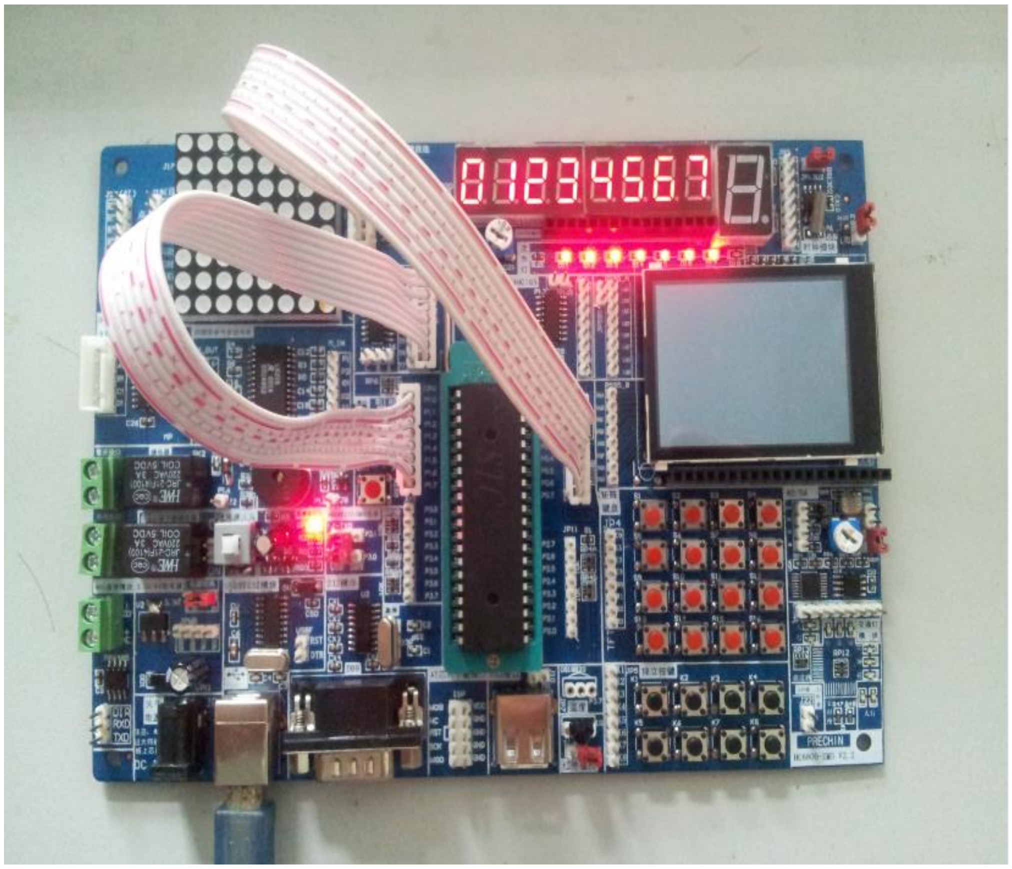
From high to low rank, (p0.7_p0.0) is written in binary 01111110, 16, into his band for A2H. We can hardware connection to the digital display figures into a form after the call directly on the line

PROGRAM:

```
#include<reg51.h>

#define GPIO_DIG   P0
#define GPIO_PLACE P1

unsigned char code DIG_PLACE[8] = {
0xfe,0xfd,0xfb,0xf7,0xef,0xdf,0xbf,0x7f};

unsigned char code DIG_CODE[17] = {
0x3f,0x06,0x5b,0x4f,0x66,0x6d,0x7d,0x07,
0x7f,0x6f,0x77,0x7c,0x39,0x5e,0x79,0x71};
unsigned char DisplayData[8];

void DigDisplay();
void main(void)
{
        unsigned char i;

        for(i=0; i<8; i++)
        {
                DisplayData[i] = DIG_CODE[i];
        }
        while(1)
        {
                DigDisplay();
        }
}
```

```
void DigDisplay()
{
        unsigned char i;
        unsigned int j;

        for(i=0; i<8; i++)
        {
                GPIO_PLACE = DIG_PLACE[i];
                GPIO_DIG = DisplayData[i];
                j = 10;
                while(j--);
                GPIO_DIG = 0x00;

}
```

Output:



Conclusion:

____

____

____

____

Viva Questions:

1) How to control the current for many cathode?
2) How many ports are required for enable the multiple 7 segment display?
3) Can we show the alphanumeric characters in multiple 7 segment display?

ACHIEVED COURSE OUTCOMES:

At the end of this experiment the student will be able to:-

Sr.No	Course Outcomes	Mapping with Pos
1	Students will be able to generate sequence of no on the screen	**PO1, PO2, PO3, PO4, PO5, PO12**
2	Students will interface the seven segment display with the other devices	**PO1, PO2, PO3, PO4, PO5, PO12**

1	Program Course outcomes	**PO1, PO2, PO3, PO4, PO5, PO12**
2	Strong POs(≥50%)	**PO1, PO2, PO3, PO4, PO5, PO12**
3	Somewhat Pos	

Signature

Department of Computer Science & Engineering

Practical No. 05

Aim: Write a Program to Demonstrate Buzzer ON and OFF

Software Required: Kiel Version: 2 or 3, STC ISP

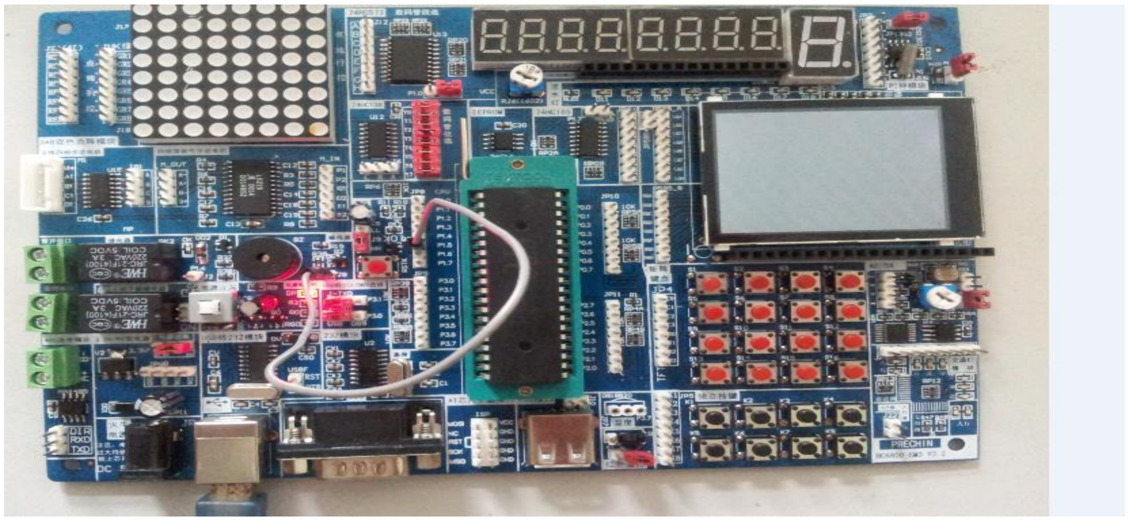
Theory:

Sound is generated by the shock and vibration of a certain frequency on a certain frequency of sound. The experiment was trumpet, issued A long one tick short of the alarm sounds, out of the port is p3.3 output 1khz, 2khz frequency alarm signal, each of the first exchange of seconds. Connection: 1PIN with a data line to insert one end of the CPU part of the JP53 (P3.I) Insert the other end of the P3.3 small part of the speaker's input JP16.

Program:

```
#include<REGX52.h>
#define buzzer P0_1
void HmsDelay(unsigned char) ;
void main()
{
while(1)
{
buzzer = 1 ; // Make LED ON
HmsDelay(500) ; // 500 ms delay
buzzer = 0 ;
HmsDelay(500) ;
}
}
void HmsDelay(unsigned char hmsDelay) { // 100 mSec @ 11.0592 MHz
unsigned int i ;
unsigned char j ;
if (hmsDelay == 0) hmsDelay = 1 ; // Non zero value
for (j=0; j<=hmsDelay; j++)
for(i=0;i<=500;i++);

}
```

Output:



Conclusion:

____

____

____

Viva Questions:

1) How many pins are required to activate Buzzer.
2) State the Logic for the Buzzer activation.

ACHIEVED COURSE OUTCOMES:

At the end of this experiment the student will be able to:-

Sr.No	Course Outcomes	Mapping with POs
1	Understand the concept of buzzer sequencing	**PO1, PO2, PO3, PO4, PO5, PO12**
2	Design and implement delay functionality in buzzer	**PO1, PO2, PO3, PO4, PO5, PO12**
1	Program Course outcomes	**PO1, PO2, PO3, PO4, PO5, PO12**
2	Strong POs(≥50%)	**PO1, PO2, PO3, PO4, PO5, PO12**
3	Somewhat Pos	

Signature

Department of Computer Science & Engineering

Practical No. 6

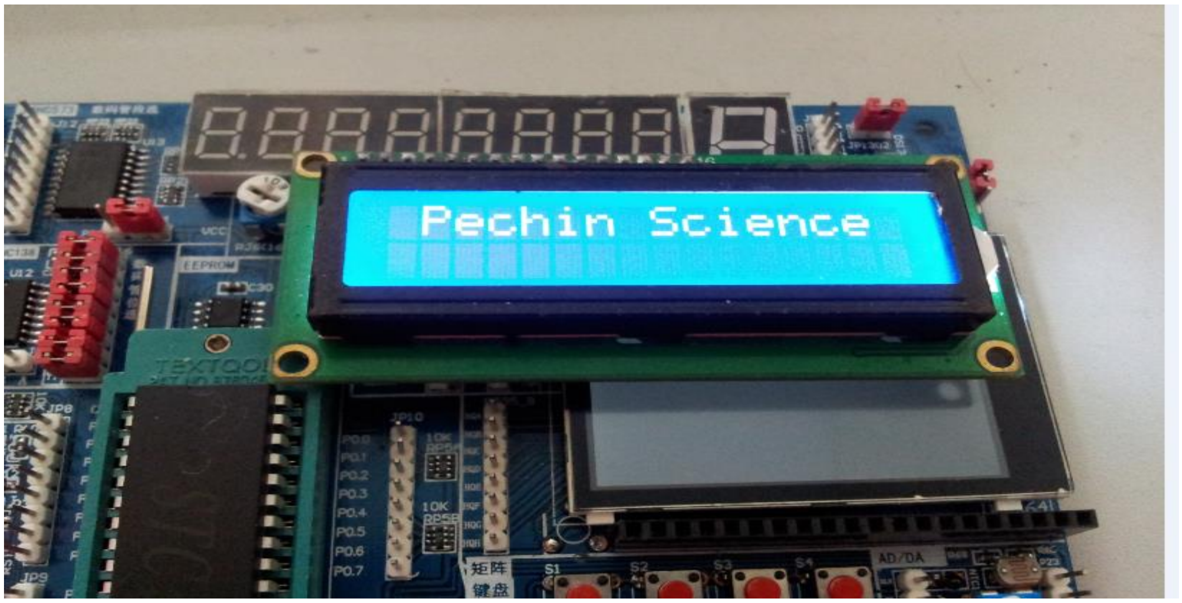
Aim: Write and execute a program for display the character using the 16 X 2 LCD display using

Software Required: Kiel Version: 2 or 3, STC ISP

Theory: XL600 part of the LCD screen can be mounted screen 12,864 Chinese characters, such as the 1602 display letters, random letters 1602 is attached to the screen. 2 show that he can visit each line of 16 English characters. 12,864 Chinese LCD can display 128 * 64 lattice, including with non-character font and two more are currently on the market with the font.
In this package, all of the pins have been part of the CPU and the relevant port to connect well. Directly in accordance with the plans shown in the direction of liquid crystal can be inserted. JP41 is one jumper to make liquid crystal. If the liquid crystal for pilot must close
the jumper, or should be disconnected in order to avoid interference. R55 which is the 1602/0802 public resistance to adjust the contrast of the LCD.

Program:
```
#include<reg51.h>
#include"lcd.h"

unsigned char PuZh[16] = " Pechin Science ";
unsigned char CnCh[27] = "Welcome to the world of MCU";


void Delay10ms(unsigned int c);

void main(void)
{
        unsigned char i;

        LcdInit();


        for(i=0; i<16; i++)
        {
                LcdWriteData(PuZh[i]);
        }

        LcdWriteCom(0xC0);
        for(i=0; i<27; i++)
        {
                LcdWriteData(CnCh[i]);
        }

        LcdWriteCom(0x07);
        while(1)
        {
                LcdWriteCom(0xC0);
                for(i=0; i<27; i++)
                {
                        LcdWriteData(CnCh[i]);
                        Delay10ms(50);
                }

        }
}



void Delay10ms(unsigned int c)
{
   unsigned char a, b;

   for (;c>0;c--)
        {
                for (b=38;b>0;b--)
                {
                        for (a=130;a>0;a--);
                }
        }
}
```

Output



Conclusion:

____

____

____

Viva Questions:

1) **How many Rows & Columns are present in 16*2 LCD?**
2) **How many Data lines are present in 16*2 Alphanumeric LCD?**
3) **Which pins of LCD is used for adjusting the contrast?**

ACHIEVED COURSE OUTCOMES:

At the end of this experiment the student will be able to:-

Sr.No	**Course Outcomes**	**Mapping with POs**
1	Understand the concept of LCD Screen Display	**PO1, PO2, PO3, PO4, PO5, PO12**
2	Design and implement 16x2 LCD Screen	**PO1, PO2, PO3, PO4, PO5, PO12**

1	Program Course outcomes	**PO1, PO2, PO3, PO4, PO5, PO12**
2	Strong POs(≥50%)	**PO1, PO2, PO3, PO4, PO5, PO12**
3	Somewhat Pos	

x

Signature

Department of Computer Science & Engineering

Practical No. 07

Aim: Write and execute a program for rotating of stepper motor in clockwise and anticlockwise Using Full Drive.

Software Required: Kiel Version: 2 or 3, STC ISP

Theory:

Stepper motor is electric pulses into a line of angular displacement or displacement of the open-loop control components. In the non-overloading, the motor speed, and stop only depend on the location of the signal pulse frequency and pulse a few, and not subject to changes in the load.

Stepper motor pulse must be double-ring signal, power-driven control system components such as circuit can be used. Therefore, make good use of the stepper motor is not an easy task, which involves mechanical, electrical, electronic and computer expertise.

The main characteristics of stepper motor:

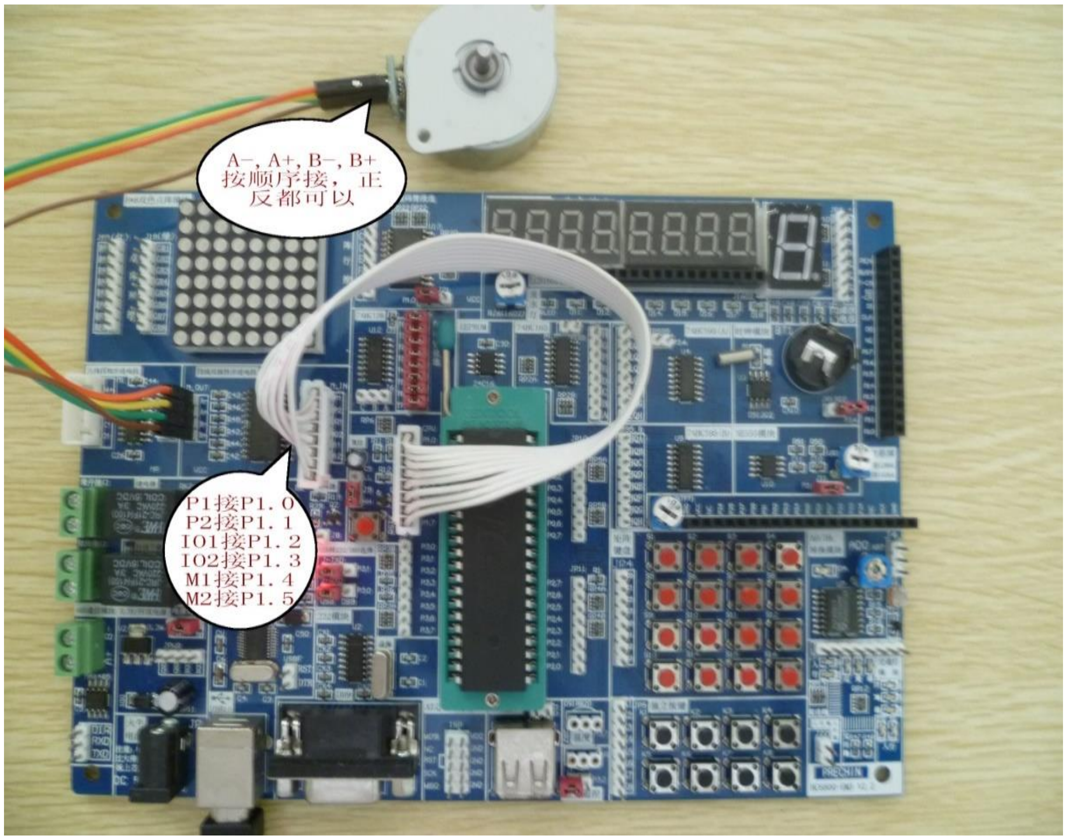
1 stepper motor can be driven to increase operation, must be driven model for the pulse, no pulse, still stepping motor, adding that if the appropriate pulse signals, it will be based on a specific point of view (known as the step angle) rotation. The rotation speed and frequency of the pulse is directly proportional. 3 step motor with an instant start and stop the rapid superior characteristics. Pulse 4 to change the order, you can easily change the direction of rotation. Embedded Development Kit package adopted by the 12v is a stepping motor, in order to demonstrate the convenience, we provide him with a 5v power supply, this time turning small moment, the reader can also be applied to him for the 12v. A stepper motor the power flow around 200ma using uln2003 drive, the drive for the port p1.0, p1.1, p1.2, p1.3. As the uln2003 it is a reverse, in practice we have In front of him, we designed a reverse- 74ls14. He was with the final results of the phase.

Program

Full Step

```
#include<reg52.h>

#include<stdio.h>

void delay(int);

void main()

{

  do
  {
    P2 = 0x03; //0011
    delay(1000);
    P2 = 0x06; //0110
    delay(1000);
    P2 = 0x0C; //1100
    delay(1000);
    P2 = 0x09; //1001
    delay(1000);
  }
while(1);
}

void delay(int k)
{
  int i,j;
  for(i=0;i<k;i++)
  {
    for(j=0;j<100;j++)
    {;}
  }
}
```

Output:



Conclusion:

____

____

____

Viva Questions:

1) How can the No of Steps per revolution of Stepper motor can be increased.

2) What happen if the direction of the current at the terminals of a series motor is reversed?

ACHIEVED COURSE OUTCOMES:

At the end of this experiment the student will be able to:-

Sr.No	**Course Outcomes**	**Mapping with POs**
1	Understand the concept of Stepper Motor	**PO1, PO2, PO3, PO4, PO5, PO12**
2	Design and implement Stepper Motor	**PO1, PO2, PO3, PO4, PO5, PO12**

1	Program Course outcomes	**PO1, PO2, PO3, PO4, PO5, PO12**
2	Strong POs(≥50%)	**PO1, PO2, PO3, PO4, PO5, PO12**
3	Somewhat Pos	

Signature

Department of Computer Science & Engineering

Practical No. 08

Aim: Write and execute a program for rotating of stepper motor in clockwise and anticlockwise Using Half Drive

Software Required: Kiel Version: 2 or 3, STC ISP

Theory:

A stepper motor or step motor or stepping motor is a brushless DC electric motor that divides a full rotation into a number of equal steps. The motor's position can then be commanded to move and hold at one of these steps without any position sensor for feedback (an open-loop controller), as long as the motor is carefully sized to the application in respect to torque and speed. Stepper drives control how a stepper motor operates, there are three commonly used excitation modes for stepper motors, full step, half step and microstepping. These excitation modes have an effect on both the running properties and torque the motor delivers.

A stepper motor converts electronic signals into mechanical movement each time an incoming pulse is applied to the motor. Each pulse moves the shaft in fixed increments. If the stepper motor has a 1.8° step resolution, then in order for shaft to rotate one complete revolution, in full step operation, the stepper motor would need to receive 200 pulses, 360° ÷ 1.8 = 200.

Half step excitation mode is a combination of one phase on and two phase on full step modes. This results in half the basic step angle. This smaller step angle provides smoother operation due the increased resolution of the angle.

Half step produces about 15% less torque than two phase on - full step, however modified half stepping eliminates the torque decrease by increasing the current applied to the motor when a single phase is energized.

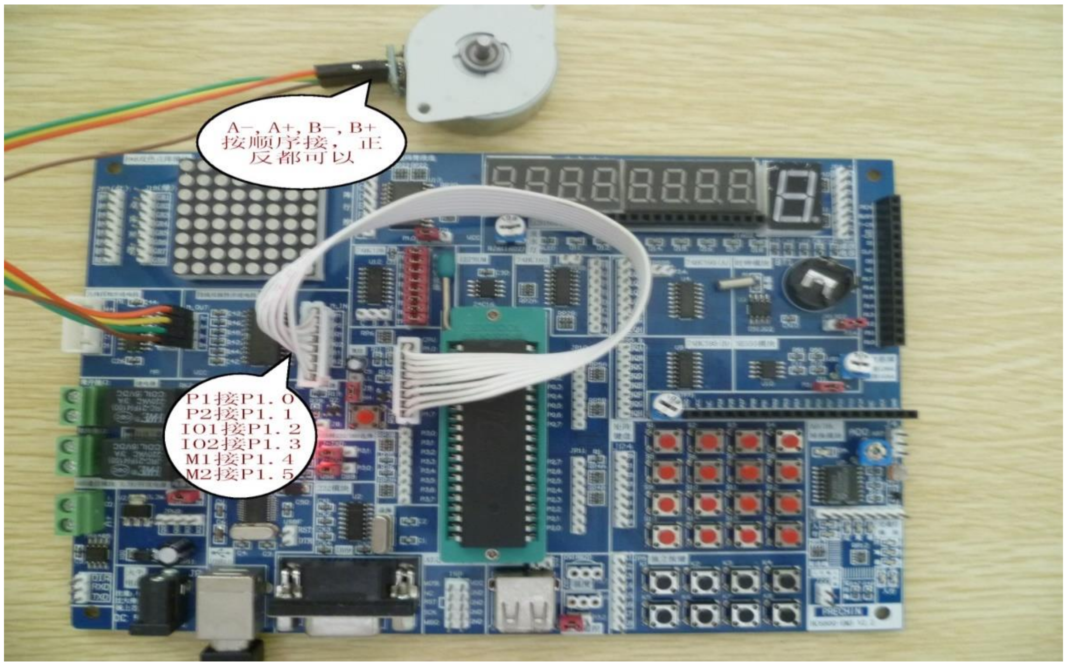
In this mode alternatively one and two electromagnets are energized, so it is a combination of Wave and Full drives. This mode is commonly used to increase the angular resolution of the motor but the torque will be less, about 70% at its half step position. We can see that the angular resolution doubles when using Half Drive.

Half Drive Stepping Sequence				
Step	A	B	C	D
1	1	0	0	0
2	1	1	0	0
3	0	1	0	0
4	0	1	1	0
5	0	0	1	0
6	0	0	1	1
7	0	0	0	1
8	1	0	0	1

Program:

```
#include<reg52.h>
#include<stdio.h>

void delay(int);

void main()

{

  do

  {

    P2=0x01; //0001

    delay(1000);

    P2=0x02; //0010

    delay(1000);

    P2=0x04; //0100

    delay(1000);

    P2=0x08; //1000
    delay(1000);
  }
while(1);
}

void delay(int k)
{
  int i,j;
  for(i=0;i<k;i++)
  {
    for(j=0;j<100;j++)
    {}
  }
}
```

Output:



Conclusion:

____

____

____

____

Viva Questions:

1) What do you mean by step angle?
2) Explain the pattern of Half Drive Stepper Motor?
3) Where the Stepper motor applications can be found?

ACHIEVED COURSE OUTCOMES:

At the end of this experiment the student will be able to:-

Sr.No	Course Outcomes	Mapping with POs
1	Understand the concept of Stepper Motor in Half drive format	**PO1, PO2, PO3, PO4, PO5, PO12**
2	Design and implement Stepper Motor for Half Drive Format	**PO1, PO2, PO3, PO4, PO5, PO12**

1	Program Course outcomes	**PO1, PO2, PO3, PO4, PO5, PO12**
2	Strong POs(≥50%)	**PO1, PO2, PO3, PO4, PO5, PO12**
3	Somewhat Pos	

Signature

Department of Computer Science & Engineering

Practical No. 9

Aim: Write and execute a program to demonstrate the Relay

Software Required: Kiel Version: 2 or 3, STC ISP

Theory:

A relay is an electromagnetic switch operated by a relatively small electric current that can turn on or off a much larger electric current. The heart of a relay is an electromagnet (a coil of wire that becomes a temporary magnet when electricity flows through it). You can think of a relay as a kind of electric lever: switch it on with a tiny current and it switches on ("leverages") another appliance using a much bigger current. As the name suggests, many sensors are incredibly sensitive pieces of electronic equipment and produce only small electric currents. But often we need them to drive bigger pieces of apparatus that use bigger currents. Relays bridge the gap, making it possible for small currents to activate larger ones. That means relays can work either as switches (turning things on and off) or as amplifiers (converting small currents into larger ones).

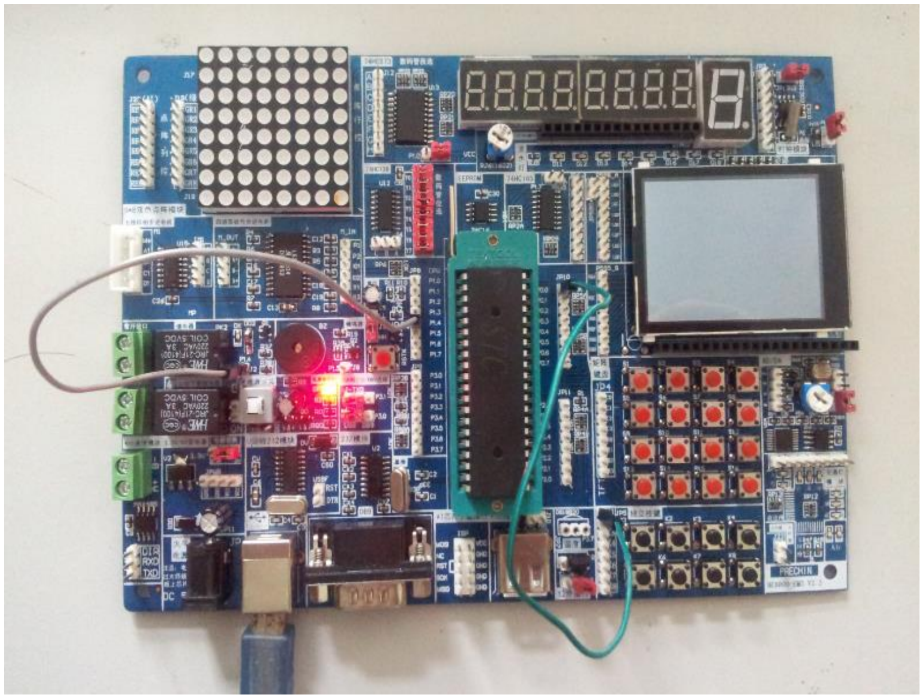
Relays are generally used to switch smaller currents in a control circuit and do not usually control power consuming devices except for small motors and Solenoids that draw low amps. Nonetheless, relays can "control" larger voltages and amperes by having an amplifying effect because a small voltage applied to a relays coil can result in a large voltage being switched by the contacts.

Protective relays can prevent equipment damage by detecting electrical abnormalities, including overcurrent, undercurrent, overloads and reverse currents. In addition, relays are also widely used to switch starting coils, heating elements, pilot lights and audible alarms.

Program:

```
#include<reg51.h>
sbit RELAY = P1^4;
sbit K1    = P0^0;


void main()
{
        while(1)
        {
                if(K1 == 0)
                {
                        RELAY = 0;
                }
                else
                {
                        RELAY = 1;
                }
        }
}
```

Output:



Conclusion:

____

____

____

____

Viva Questions:

1) **What is the operational Principal of differential relay?**
2) **Where Impedance relay, reactance relay and mho relay are used?**
3) **What is Percentage of Differential Relay?**

ACHIEVED COURSE OUTCOMES:

At the end of this experiment the student will be able to:-

Sr.No	**Course Outcomes**	**Mapping with POs**
1	Understand the concept of Relay	**PO1, PO2, PO3, PO4, PO5, PO12**
2	Design and implement Relay	**PO1, PO2, PO3, PO4, PO5, PO12**

1	Program Course outcomes	**PO1, PO2, PO3, PO4, PO5, PO12**
2	Strong POs(≥50%)	**PO1, PO2, PO3, PO4, PO5, PO12**
3	Somewhat Pos	

Signature

Department of Computer Science & Engineering

Practical No. 10

Aim: Write and execute a program to demonstrate the key technology on matrix.

Software Required: Kiel Version: 2 or 3, STC ISP

Theory: Matrix Keypads are commonly used in calculators, telephones etc where a number of input switches are required. We know that matrix keypad is made by arranging push button switches in row and columns. In the straight forward way to connect a 4×4 keypad (16 switches) to a microcontroller we need 16 inputs pins. But by connecting switches in the following way we can read the status of each switch using 8 pins of the microcontroller

The status of each keys can be determined by a process called Scanning. For the sake of explanation lets assume that all the column pins (Col1 – Col4) are connected to the inputs pins and all the row pins are connected to the output pins of the microcontroller. In the normal case all the column pins are pulled up (HIGH state) by internal or external pull up resistors. Now we can read the status of each switch through scanning.

1. A logic LOW is given to Row1 and others (Row2 – Row-4) HIGH
2. Now each Column is scanned. If any switch belongs to 1st row is pressed corresponding column will pulled down (logic LOW) and we can detect the pressed key.
3. This process is repeated for all rows.

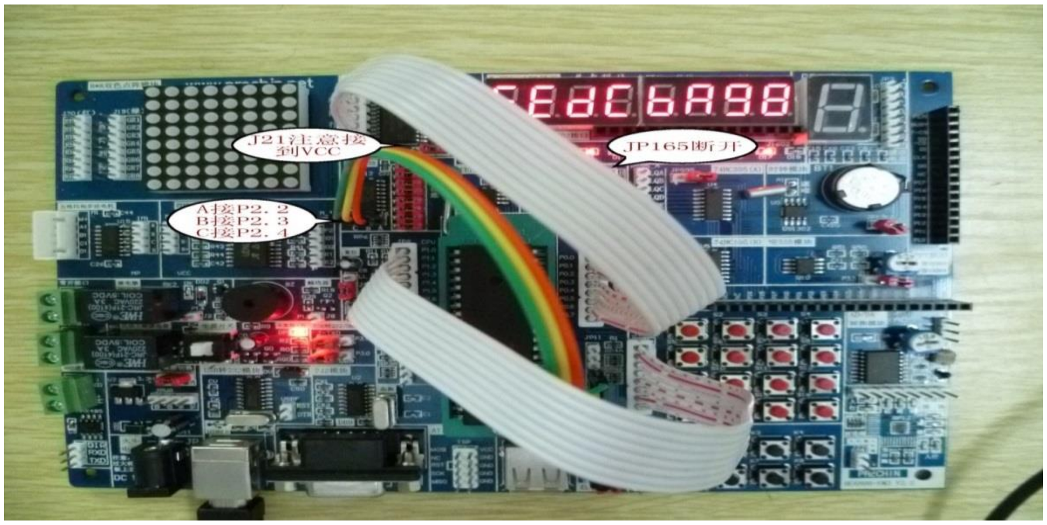
Program:

```
#include<reg51.h>
#define GPIO_DIG P0
#define GPIO_KEY P1

sbit LSA=P2^2;
sbit LSB=P2^3;
sbit LSC=P2^4;

unsigned char code DIG_CODE[17]={
0x3f,0x06,0x5b,0x4f,0x66,0x6d,0x7d,0x07,
0x7f,0x6f,0x77,0x7c,0x39,0x5e,0x79,0x71};

unsigned char KeyValue;

unsigned char KeyState;
unsigned char DisplayData[8];
```

```
void Delay10ms(unsigned int c);
void KeyDown();
void DigDisplay();
void main(void)
{
	KeyState=0;
	while(1)
	{
		KeyDown();
		if(KeyState==1)
		{
			DisplayData[7]=DisplayData[6];
			DisplayData[6]=DisplayData[5];
			DisplayData[5]=DisplayData[4];
			DisplayData[4]=DisplayData[3];
			DisplayData[3]=DisplayData[2];
			DisplayData[2]=DisplayData[1];
			DisplayData[1]=DisplayData[0];
			DisplayData[0]=DIG_CODE[KeyValue];

			KeyState=0;
		}
		DigDisplay();
	}
}



void DigDisplay()
{
	unsigned char i;
	unsigned int j;

	for(i=0;i<8;i++)
	{
		switch(i)
		{
			case(0):
				LSA=0;LSB=0;LSC=0; break;
			case(1):
				LSA=1;LSB=0;LSC=0; break;
			case(2):
				LSA=0;LSB=1;LSC=0; break;
			case(3):
				LSA=1;LSB=1;LSC=0; break;
```

```
                        case(4):
                                LSA=0;LSB=0;LSC=1; break;
                        case(5):
                                LSA=1;LSB=0;LSC=1; break;
                        case(6):
                                LSA=0;LSB=1;LSC=1; break;
                        case(7):
                                LSA=1;LSB=1;LSC=1; break;
                }
                GPIO_DIG=DisplayData[i];
                j=10;
                while(j--);
                GPIO_DIG=0x00;
        }
}

void KeyDown(void)
{
        unsigned int a=0;

        GPIO_KEY=0x0f;
        if(GPIO_KEY!=0x0f)
        {
                Delay10ms(1);
                a++;
                a=0;
                if(GPIO_KEY!=0x0f)
                {
                        KeyState=1;

                        GPIO_KEY=0X0F;

                        switch(GPIO_KEY)
                        {
                                case(0X07):     KeyValue=0;break;
                                case(0X0b):     KeyValue=4;break;
                                case(0X0d): KeyValue=8;break;
                                case(0X0e):     KeyValue=12;break;
                        }

                        GPIO_KEY=0XF0;

                        switch(GPIO_KEY)
                        {
                                case(0X70):     KeyValue=KeyValue+3;break;
                                case(0Xb0):     KeyValue=KeyValue+2;break;
                                case(0Xd0): KeyValue=KeyValue+1;break;
                                case(0Xe0):     KeyValue=KeyValue;break;
//                              default:        KeyValue=17;
                        }
                        while((a<50)&&(GPIO_KEY!=0xf0))
                        {
                                Delay10ms(1);
                                a++;
                        }
                        a=0;
                }
        }
}


void Delay10ms(unsigned int c)
{
   unsigned char a, b;


   for (;c>0;c--)
        {
                for (b=38;b>0;b--)
                {
                        for (a=130;a>0;a--);
                }
        }
}
```

Output:



Conclusion:

____

____

____

Viva Questions:

1) Explain the Key technology concept used in matrix keypad?

2) Explain the Circuit Diagram of Matrix Keypad?

3) Explain the logic LOW & HIGH in the Matrix Keypad?

ACHIEVED COURSE OUTCOMES:

At the end of this experiment the student will be able to:-

Sr.No	**Course Outcomes**	**Mapping with POs**
1	Understand the concept of Key	**PO1, PO2, PO3, PO4, PO5, PO12**
2	Design and implement Key Technology	**PO1, PO2, PO3, PO4, PO5, PO12**
1	Program Course outcomes	**PO1, PO2, PO3, PO4, PO5, PO12**
2	Strong POs(≥50%)	**PO1, PO2, PO3, PO4, PO5, PO12**
3	Somewhat Pos	

Signature

Department of Computer Science & Engineering

Practical No. 11

Aim: The aim of the practical is to familiarize with the use of push button switch with the micro-controller.

Software Requirements: Keil 2.0 , ISP

Theory:

In this case the micro-controller is AT89C51, a re-programmable derivative of 8051.It will be useful whenever a decision is to be made according to the press of a switch.

Circuit Diagram

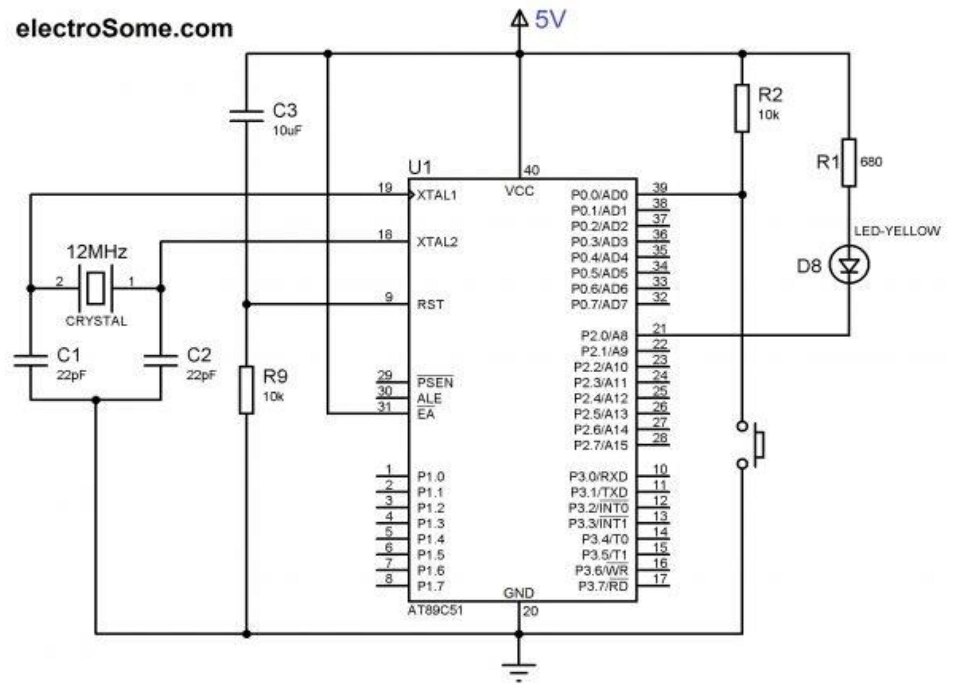


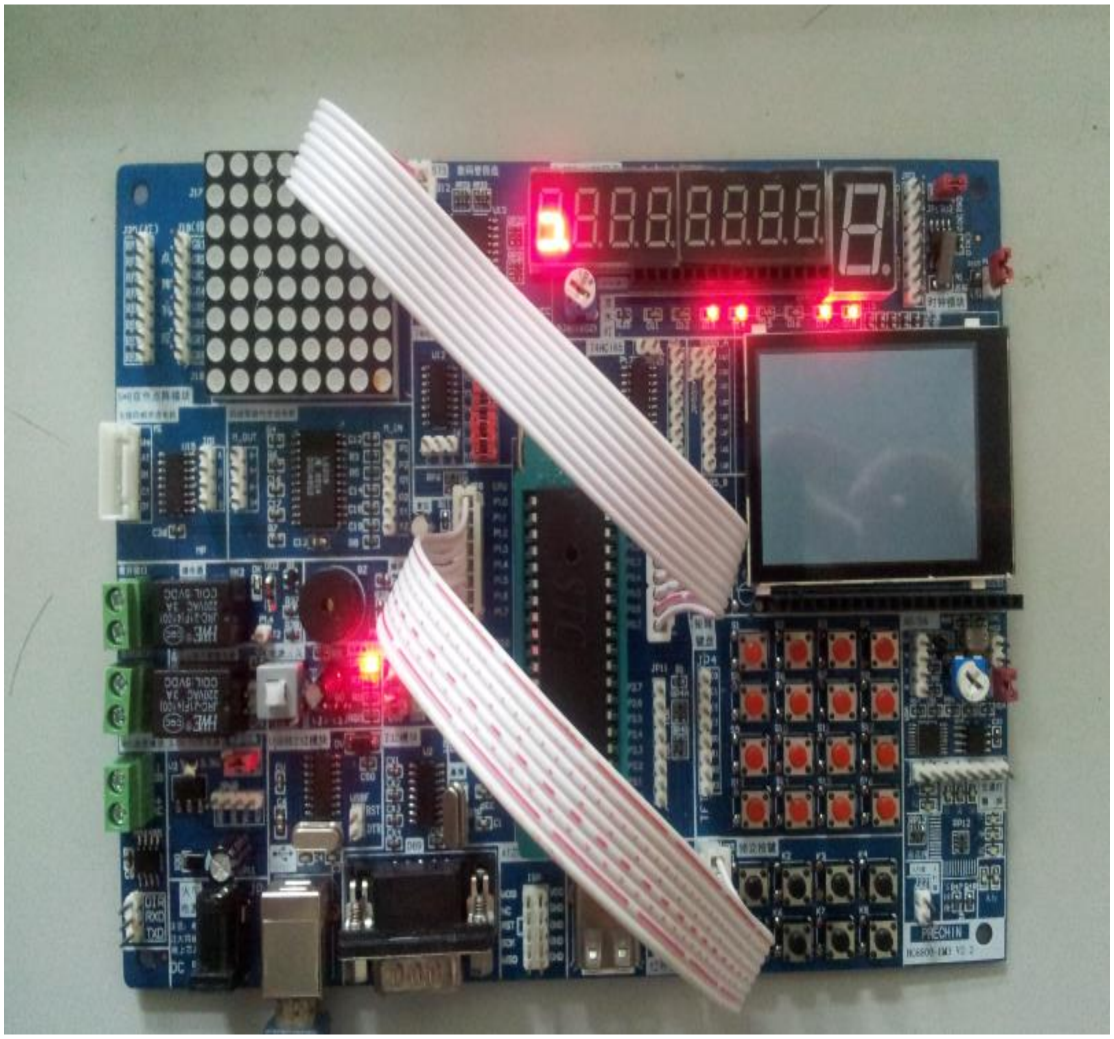
Figure: LED and Switch Interfacing with AT89C51 8051 Microcontroller

Push button switch is connected to the first bit of PORT 0 (P0.0) which is configured as an input pin. Which is connected to a pull up resistor as there is **NO INTERNAL PULL UP RESISTORS FOR PORT P0**. Thus P0.0 pin is at Vcc potential when the switch is not pressed. When the switch is pressed this pin P0.0 will be grounded. The LED is connected to the first bit of PORT 2 (P2.2) and a resistor is connected in series with it to limit the current.

Program:

```
#include <reg51.h>
#include <intrins.h>

#define  GPIO_KEY P1
#define  GPIO_LED P0

void Delay10ms(unsigned int c);
unsigned char Key_Scan();

void main(void)
{
        unsigned char ledValue, keyNum;
        ledValue = 0x01;
        while (1)
        {
                keyNum = Key_Scan();
                switch (keyNum)
                {
                        case(0xFE) :
                                ledValue = 0x01;
                                break;
                        case(0xFD) :
                                ledValue = 0x02;
                                break;
                        case(0xFB) :
                                ledValue = 0x04;
                                break;
                        case(0xF7) :
                                ledValue = 0x08;
                                break;
                        case(0xEF) :
                                ledValue = 0x10;
                                break;
                        case(0xDF) :
                                ledValue = 0x20;
                                break;
                        case(0xBF) :
                                ledValue = 0x40;
                                break;
                        case(0x7F) :
                                ledValue = 0x80;
                                break;
                        default:
                                break;
                }

                GPIO_LED = ledValue;
        }
}


unsigned char Key_Scan()
{
        unsigned char keyValue = 0 , i;
        if (GPIO_KEY != 0xFF)
        {
                Delay10ms(1);

                if (GPIO_KEY != 0xFF)
                {
                        keyValue = GPIO_KEY;
                        i = 0;
                        while ((i<50) && (GPIO_KEY != 0xFF))
                        {
                                Delay10ms(1);
                                i++;
                        }
                }
        }

        return keyValue;
}



void Delay10ms(unsigned int c)
{
   unsigned char a, b;


   for (;c>0;c--)
        {
                for (b=38;b>0;b--)
                {
                        for (a=130;a>0;a--);
                }

        }
}
```

Output:



Conclusion:

____

____

____

____

Viva Questions:

1) What is the main function of Switch?
2) Explain the circuit Diagram of the Switch in applications area?

ACHIEVED COURSE OUTCOMES:

At the end of this experiment the student will be able to:-

Sr.No	**Course Outcomes**	**Mapping with POs**
1	Understand the concept of Swtich Technology	**PO1, PO2, PO3, PO4, PO5, PO12**
2	Design and implement Switch Technology	**PO1, PO2, PO3, PO4, PO5, PO12**
1	Program Course outcomes	**PO1, PO2, PO3, PO4, PO5, PO12**
2	Strong POs(≥50%)	**PO1, PO2, PO3, PO4, PO5, PO12**
3	Somewhat Pos	

Signature

Department of Computer Science & Engineering

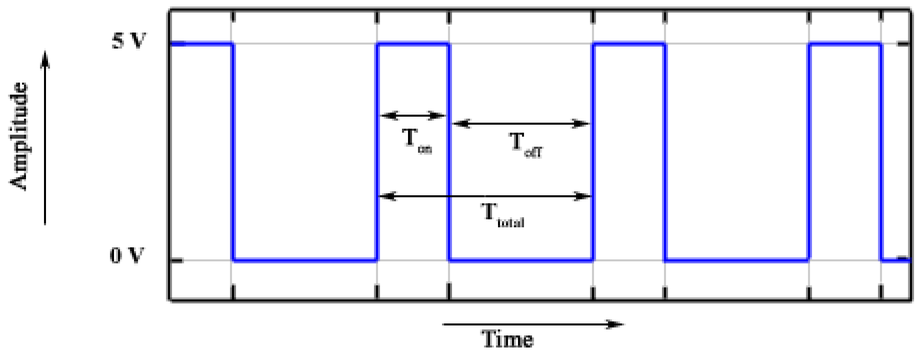
Practical No. 12

Name of Experiment: Write a Program to Demonstrate PWM.

Software Requirements: Keil 2.0, ISP

Theory: Basic Principal of PWM

Pulse width modulation is basically a square wave with a varying high and low time. A basic PWM signal is shown in the figure below.



Pulse width modulation wave

There are various terms associated with PWM:

1. On-Time: Duration of time signal is high
2. Off-Time: Duration of time signal is low
3. Period: It is represented as the sum of on-time and off-time of PWM signal
4. Duty cycle: It is represented as percentage of time signal remains on during the period of the PWM signal

Period

As shown in the the figure, T_{on} denotes the on-time and T_{off} denotes the off time of signal. Period is the sum of both on and off times and is calculated as shown in the equation below:

$$T_{total} = T_{on} + T_{off}$$

Duty Cycle

Duty cycle is calculated as on-time to the period of time. Using the period calculated above, duty cycle is calculated as:

$$D = \frac{T_{on}}{(T_{on} + T_{off})} = \frac{T_{on}}{T_{total}}$$

PWM: Voltage Regulation

PWM signal when used at a different duty cycles gives a varying voltage at the output. This method is used in various areas of application like:

- Switching regulators
- LED dimmers
- Audio
- Analog signal generation
- and many more...

Voltage regulation is done by averaging the PWM signal. Output voltage is represented by the following equation:

$$V_{out} = D \times V_{in} \quad \boxed{V_{out} = \frac{T_{on}}{T_{total}} \times V_{in}}$$

As you can see from the equation the output voltage can be directly varied by varying the T_{on} value.

If T_{on} is 0, V_{out} is also 0. if T_{on} is T_{total} then V_{out} is V_{in} or say maximum.

Implementing PWM on 8051

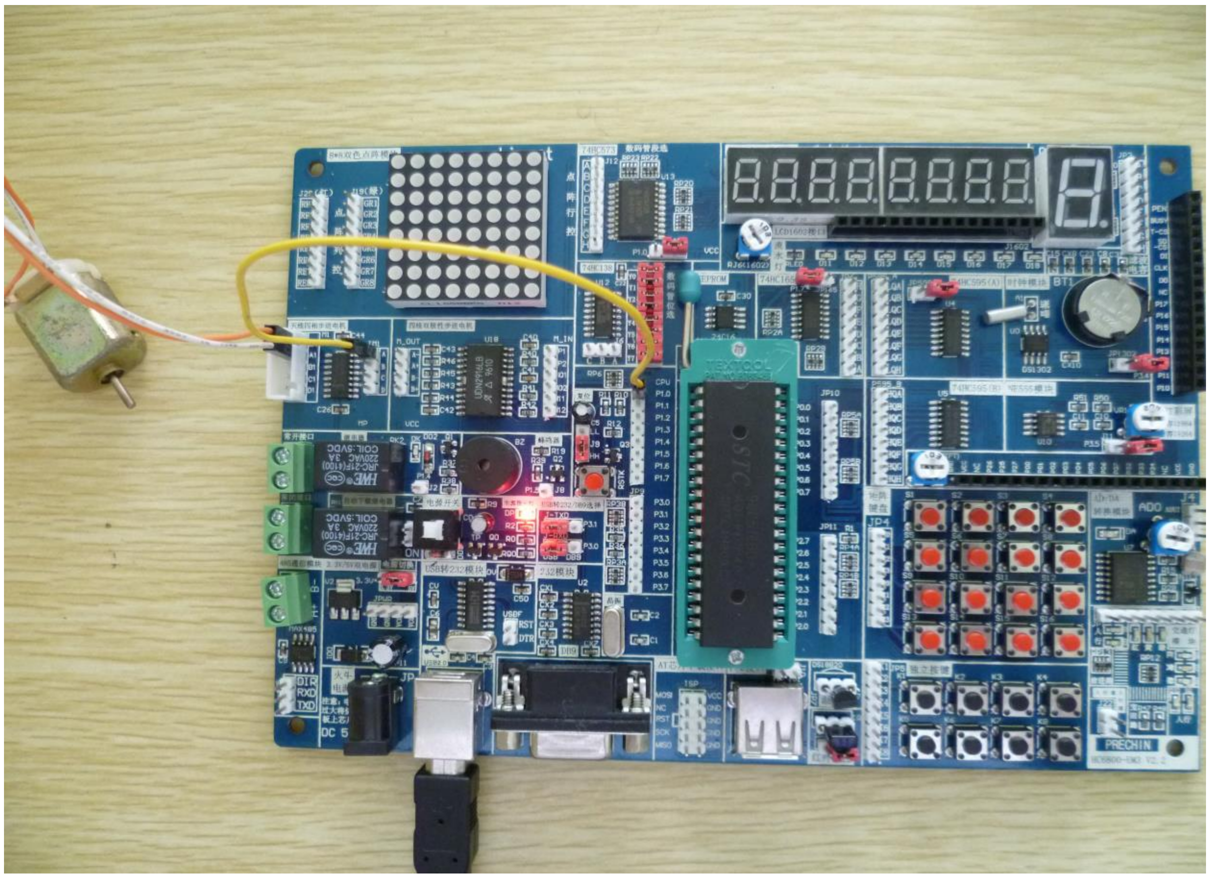
The basic idea behind PWM implementation on 8051 is using timers and switching port pin high/low at defined intervals. As we have discussed in the introduction of PWM that by changing the Ton time, we can vary the width of square wave keeping same time period of the square wave.

We will be using 8051 Timer0 in Mode 0. Values for high and low level will be loaded in such a way that total delay remains same. If for high level we load a value X in TH0 then for low level TH0 will be loaded with 255-X so that total remains as 255.

Program:

```
#include <reg52.h>
void Time1Config();

void main(void)
{
        Time1Config();
        while(1)
        {
                if(timer1>100)
                {
                        timer1=0;
                }
                if(timer1 < 30)
                {
                        PWM=1;
                }
                else
                {
                        PWM=0;
                }
        }

 }

void Time1Config()
{
        TMOD|= 0x10;
        TH1 = 0xFE;
        TL1 = 0x0C;

        ET1 = 1;
        EA = 1;
        TR1 = 1;
}


void Time1(void) interrupt 3
{
        TH1 = 0xFE;
        TL1 = 0x0C;
        timer1++;
}
```

Output:



Conclusion:

Viva Questions:

1) What is Amplitude Modulation?

2) What is PAM?

3) What are the advantages of PAM & PWM?

ACHIEVED COURSE OUTCOMES:

At the end of this experiment the student will be able to:-

1	Program Course outcomes	**PO1, PO2, PO3, PO4, PO5, PO12**
2	Strong POs(≥50%)	**PO1, PO2, PO3, PO4, PO5, PO12**
3	Somewhat Pos	

Signature

Department of Computer Science & Engineering

Practical No. 13

Name of Experiment: Write a Program to Demonstrate IR Remote.

Software Requirements:

Theory:

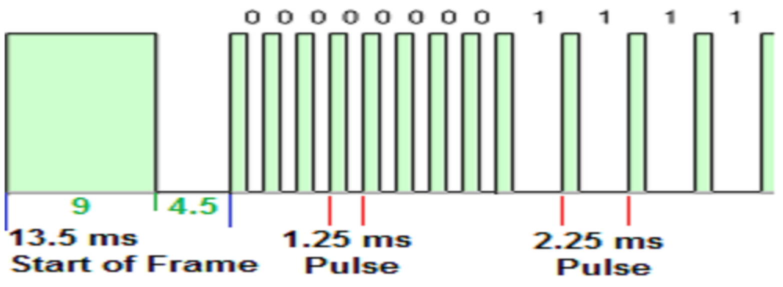
IR Remote Controllers and receivers follow standard protocols for sending and receiving the data. Some of the standard protocols are NEC , JVC , SIRC (Sony Infrared Remote Control) etc. We will be discussing only the NEC protocol. After understanding the frame format of IR Remote, We will be interfacing a IR receiver with 8051 and decode the key pressed.

NEC Protocol

NEC IR protocol encodes the keys using a 32bit frame format as shown below.

NEC Frame Format			
Address	Complement of Address	Command	Complement of Command
LSB-MSB(0-7)	LSB-MSB(8-15)	LSB-MSB(16-23)	LSB-MSB(24-31)

Each bit is transmitted using the pulse distance as shown in the image.
Logical '0': A 562.5µs pulse burst followed by a 562.5µs space, with a total transmit time of 1.125ms
Logical '0': A 562.5µs pulse burst followed by a 1.6875ms space, with a total transmit time of 2.25ms

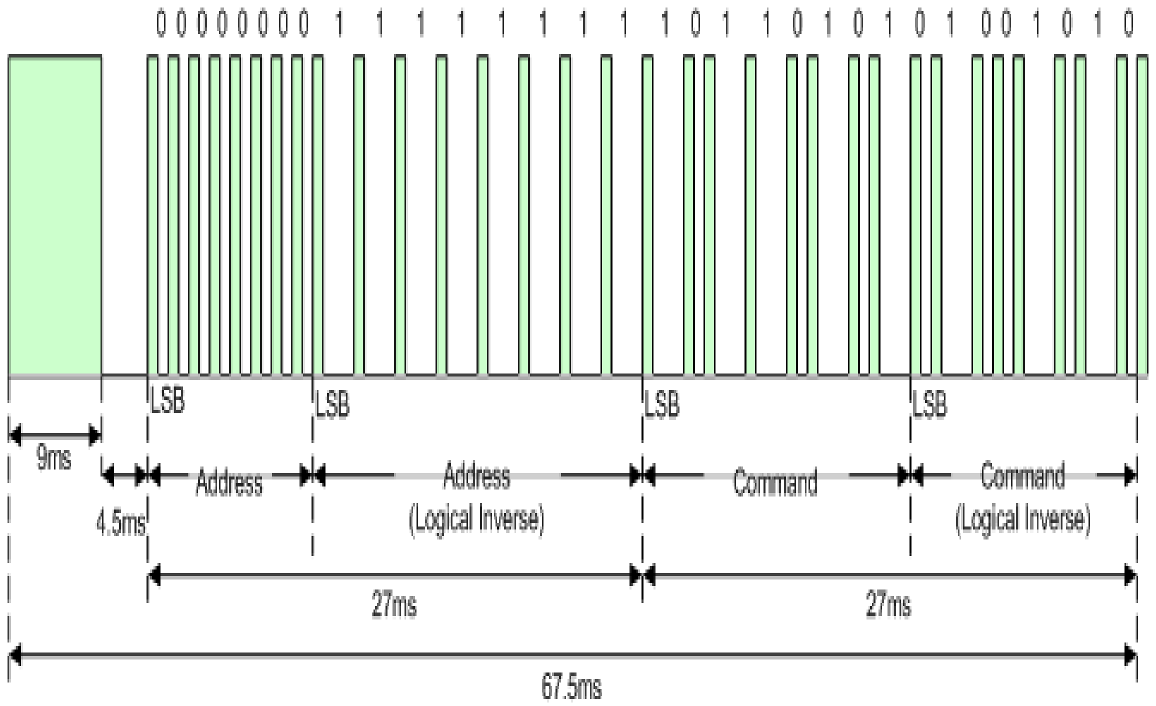


When a key is pressed on the remote controller, the message transmitted consists of the following, in order:

1. A 9ms leading pulse burst (16 times the pulse burst length used for a logical data bit)
2. A 4.5ms space
3. The 8-bit address for the receiving device
4. The 8-bit logical inverse of the address
5. The 8-bit command
6. The 8-bit logical inverse of the command
7. A final 562.5µs pulse burst to signify the end of message transmission.

The four bytes of data bits are each sent least significant bit first. Below image illustrates the format of an NEC IR transmission frame, for an address of 00h (00000000b) and a command of ADh (10101101b).

A total of 67.5ms is required to transmit a message frame. It needs 27ms to transmit the 16 bits of address (address + inverse) and the 16 bits of command (command + inverse).



IR Key Codes

Below table gives the complete list of Codes for NEC IR Remote.

Key	Encoded Value	Key	Encoded Value
CH-	0xFFA25D	1	0xFF30CF
CH	0xFF629D	2	0xFF18E7
CH+	0xFFE21D	3	0xFF7A85
PREV	0xFF22DD	4	0xFF10EF
NEXT	0xFF02FD	5	0xFF38C7
PLAY/PAUSE	0xFFC23D	6	0xFF5AA5
VOL-	0xFFE01F	7	0xFF42BD
VOL+	0xFFA857	8	0xFF4AB5
EQ	0xFF906F	9	0xFF52AD
0	0xFF6897		
100+	0xFF9867		
200+	0xFFB04F		

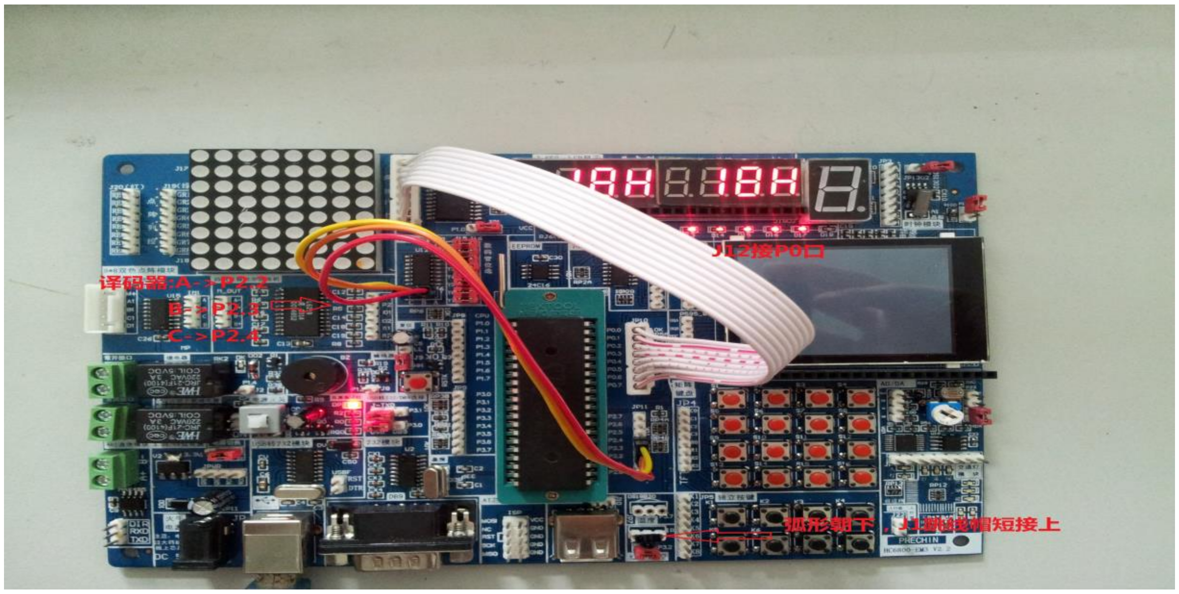
Program:

```
#include<reg51.h>
#define GPIO_DIG P0


sbit LSA  = P2^2;
sbit LSB  = P2^3;
sbit LSC  = P2^4;
sbit IRIN = P3^2;

unsigned char code DIG_CODE[17]={
0x3f,0x06,0x5b,0x4f,0x66,0x6d,0x7d,0x07,
0x7f,0x6f,0x77,0x7c,0x39,0x5e,0x79,0x71};

unsigned char DisplayData[8];


unsigned char IrValue[6];
unsigned char Time;


void DigDisplay();
void IrInit();
void DelayMs(unsigned int );

void main()
{
        unsigned char i;
        IrInit();
        while(1)
        {
                IrValue[4]=IrValue[2]>>4;
                IrValue[5]=IrValue[2]&0x0f;
                DisplayData[0] = 0x00;
                DisplayData[1] = DIG_CODE[IrValue[4]];
                DisplayData[2] = DIG_CODE[IrValue[5]];
                DisplayData[3] = 0x76;          //01110110
                DisplayData[4] = 0x00;
                DisplayData[5] = DIG_CODE[IrValue[4]];
                DisplayData[6] = DIG_CODE[IrValue[5]];
                DisplayData[7] = 0x76;

                DigDisplay();

        }
```

```
}


void DelayMs(unsigned int x)
{
        unsigned char i;
        while(x--)
        {
                for (i = 0; i<13; i++)
                {}
        }
}


void IrInit()
{
        IT0=1;
        EX0=1;
        EA=1;

        IRIN=1;
}


void ReadIr() interrupt 0
{
        unsigned char j,k;
        unsigned int err;
        Time=0;
        DelayMs(70);

        if(IRIN==0)
        {

                err=1000;
                while((IRIN==0)&&(err>0))
                {
                        DelayMs(1);
                        err--;
                }
                if(IRIN==1)
                {
                        err=500;
                        while((IRIN==1)&&(err>0))
                        {
                                DelayMs(1);
                                err--;
                        }
                        for(k=0;k<4;k++)
                        {
                                for(j=0;j<8;j++)
                                {

                                        err=60;
                                        while((IRIN==0)&&(err>0))
                                        {

                                                DelayMs(1);
                                                err--;
                                        }
                                        err=500;
                                        while((IRIN==1)&&(err>0))
                                        {
                                                DelayMs(1);//0.14ms
                                                Time++;
                                                err--;
                                                if(Time>30)
                                                {
                                                        EX0=1;
                                                        return;
                                                }
                                        }
                                        IrValue[k]>>=1;
                                        if(Time>=8)
                                        {
                                                IrValue[k]|=0x80;
                                        }
                                        Time=0;

                                }
                        }
                }
                if(IrValue[2]!=~IrValue[3])
                {
                        return;
                }
        }
}


void DigDisplay()
{
        unsigned char i;
        unsigned int j;
```

```
          for(i=0;i<8;i++)
          {
                    switch(i)
                    {
                              case(0):
                                        LSA=0;LSB=0;LSC=0; break;
                              case(1):
                                        LSA=1;LSB=0;LSC=0; break;
                              case(2):
                                        LSA=0;LSB=1;LSC=0; break;
                              case(3):
                                        LSA=1;LSB=1;LSC=0; break;
                              case(4):
                                        LSA=0;LSB=0;LSC=1; break;
                              case(5):
                                        LSA=1;LSB=0;LSC=1; break;
                              case(6):
                                        LSA=0;LSB=1;LSC=1; break;
                              case(7):
                                        LSA=1;LSB=1;LSC=1; break;

}

                    GPIO_DIG=DisplayData[i];
                    j=10;
                    while(j--);
                    GPIO_DIG=0x00;
          }
}
```

Output:



Conclusion:

____

____

____

____

Viva Questions:

1) Explain the working of IR remote Sensor?

2) Explain the Circuit Interfacing of IR remote?

3) Explain the Applications area of IR Remote?

ACHIEVED COURSE OUTCOMES:

At the end of this experiment the student will be able to:-

Sr. No	**Course Outcomes**	**Mapping with Pos**
1	Understand the concept of IR Remote	**PO1, PO2, PO3, PO4, PO5, PO12**
2	Understand the concepts of the operating principal of the IR System	**PO1, PO2, PO3, PO4, PO5, PO12**

1	Program Course outcomes	**PO1, PO2, PO3, PO4, PO5,**
2	Strong POs(≥50%)	**PO1, PO2, PO3, PO4, PO5,**
3	Somewhat Pos	

Signature

Department of Computer Science & Engineering

Practical No. 14

Aim: To Study Vx Works Operating System

Theory:

VxWorks is a real-time operating system (RTOS) developed as proprietary software by Wind River Systems of Alameda, California, US. First released in 1987, VxWorks is designed for use in embedded systems requiring real-time, deterministic performance and, in many cases, safety and security certification, for industries, such as aerospace and defense, medical devices, industrial equipment, robotics, energy, transportation, network infrastructure, automotive, and consumer electronics.

VxWorks supports Intel (x86, including the new Intel Quark SoC, and x86-64), MIPS, PowerPC, SH-4, and ARM architectures. The RTOS can be used in multicore asymmetric multiprocessing (AMP), symmetric multiprocessing (SMP), and mixed modes and multi-OS (via Type 1 hypervisor)[5] designs on 32- and 64-bit processors.

VxWorks comes with the kernel, middle ware, board support packages, Wind River Workbench development suite and complementary third-party software and hardware technologies. In its latest release, VxWorks 7, the RTOS has been re-engineered for modularity and upgrade-ability so the OS kernel is separate from middle ware, applications and other packages. Scalability, security, safety, connectivity, and graphics have been improved to address Internet of Things (IoT) needs.

Platform Overview:

VxWorks supports Intel (x86, including the new Intel Quark SoC, and x86-64), MIPS, PowerPC, SH-4, and ARM architectures. The RTOS can be used in multicore asymmetric multiprocessing (AMP), symmetric multiprocessing (SMP), and mixed modes and multi-OS (via Type 1 hypervisor) designs on 32- and 64-bit processors.

The VxWorks Core Platform consists of a set of runtime components and development tools. The run time components are an operating system (UP and SMP; 32- and 64-bit), software for applications support (file system, core network stack, USB stack and inter-process communications) and hardware support (architecture adaptor, processor support library, device driver library and board support packages). VxWorks core development tools are compilers such as Diab, GNU, and Intel C++ Compiler (ICC)) and its build and config tools. The system also includes productivity tools such as its Workbench development suite and Intel tools and development support tools for asset tracking and host support.

The platform is a modular, vendor-neutral, open system that supports a range of third-party software and hardware. The OS kernel is separate from middleware, applications and other packages, which enables easier bug fixes and testing of new features. An implementation of a layered source build system allows multiple versions of any stack to be installed at the same time so developers can select which version of any feature set should go into the VxWorks kernel libraries.

Optional Profiles for VxWorks add incremental functionality required for specific industries (such as medical, industrial, networking and consumer) or technology-related capabilities, such as a small footprint RTOS (Microkernel Profile) and a Type 1 real-time embedded hypervisor (Virtualization Profile).

Features:

Vx Works is designed for use in embedded systems.

A list of some of the features of the OS is:

- Multitasking kernel with preemptive and round-robin scheduling and fast interrupt response
- Native 64-bit operating system (only one 64-bit architecture supported: x86-64). Data model: LP64.
- User-mode applications ("Real-Time Processes", or RTP) isolated from other user-mode applications as well as the kernel via memory protection mechanisms.
- SMP, AMP and mixed mode multiprocessing support
- Error handling framework
- Bluetooth, USB, CAN protocols, Firewire IEEE 1394, BLE, L2CAP, Continua stack, health device profile
- Binary, counting, and mutual exclusion semaphores with priority inheritance
- Local and distributed message queues
- POSIX PSE52 certified conformity in user-mode execution environment

- File systems: High Reliability File System (HRFS), FAT-based file system (DOSFS), Network File System (NFS), and TFFS
- Dual-mode IPv6 networking stack with IPv6 Ready Logo certification
- Memory protection including real-time processes (RTPs), error detection and reporting, and IPC
- Multi-OS messaging using TIPC and Wind River multi-OS IPC
- Symbolic debugging

Conclusion:

Viva Questions:

1) What is Soft Real System?

2) Which scheduling algorithms are preferable used for hard real time system?

3) Is Ostrich Methodology used for Deadlock Prevention in Hard Real Time System?

ACHIEVED COURSE OUTCOMES:

At the end of this experiment the student will be able to:-

Sr. No	Course Outcomes	Mapping with Pos
1	Understand the concept of Hard Real Time System	**PO1, PO2, PO3, PO4, PO5, PO12**
2	Understand the concepts of the operating system like threads, cores, semaphores, etc	**PO1, PO2, PO3, PO4, PO5, PO12**
1	Program Course outcomes	**PO1, PO2, PO3, PO4, PO5,**
2	Strong POs(≥50%)	**PO1, PO2, PO3, PO4, PO5,**
3	Somewhat Pos	

Signature

Department of Computer Science & Engineering