

LECTURE NOTES (E- CONTENTS) for

B.Sc. III Electronics (2021-22)

Semester: V Paper- V DSE 1005E1

**Linear Integrated Circuits, 8051 Microcontroller Interfacing and
Embedded C**

Section II: 8051 Microcontroller Interfacing and Embedded C

Unit- 4: Applications of 8051

Prepared and Circulated for

B .Sc. III Electronics Students

BY

Dr. C. B. Patil

Associate Professor, Department of Electronics

Vivekanand College (Autonomous), Kolhapur

(For Private Circulation only)

Unit 4

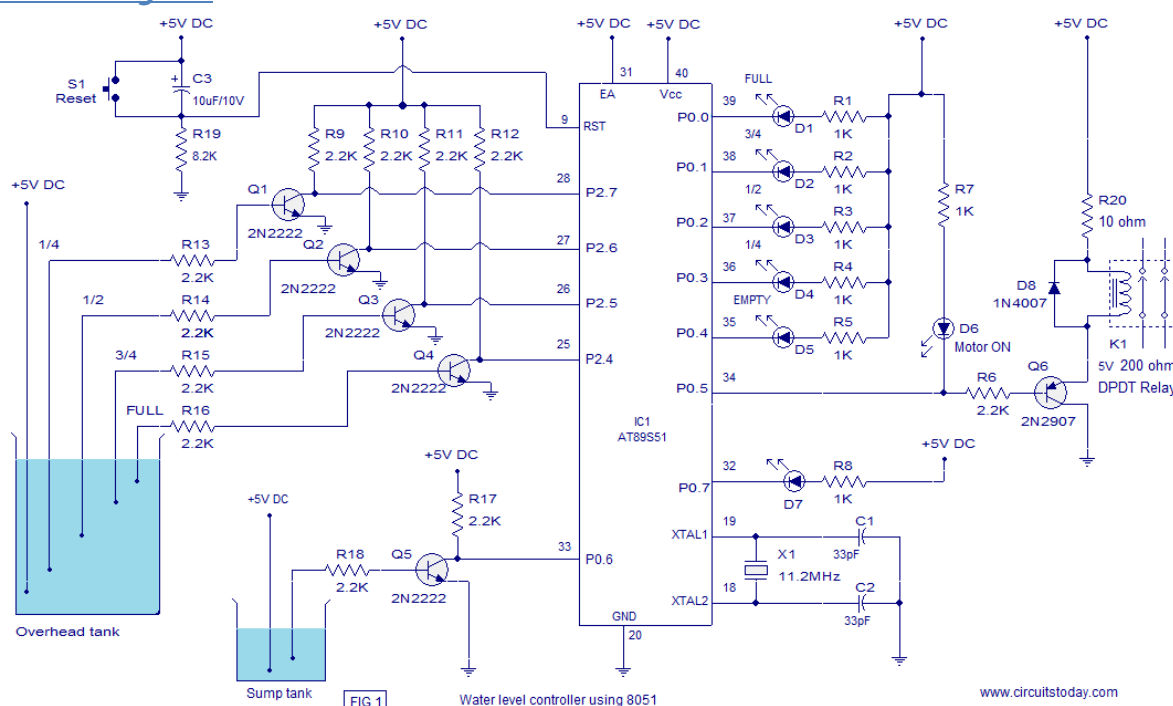
Applications of 8051

Syllabus: Applications of 8051 Case study's: i) Gate Emulator (Logic Gate study using microcontroller) ii) Water level controller iii) speed control of DC motor iv) Temperature measurement using LM35, ADC0804, LCD v) Bluetooth module interfacing vi) Speed control of Stepper Motor

1) Water level controller using 8051:

A water level controller using 8051 is shown in fig. This water level controller monitors the level of the over head tank and automatically switches on the water pump whenever the level goes below a preset limit. The level of the over head tank is indicated using 5 LED's and the pump is switched OFF when the over head tank is full. The pump is not allowed to start if the water level in the sump tank is low and also the pump is switched OFF when the level inside the sump tank goes low during a pumping cycle.

Circuit diagram:



The level sensor probes for the overhead tank are interfaced to the port 2 of the microcontroller through transistors. The sensor probe arrangement for the overhead tank is shown in Fig. A positive voltage supply probe goes to the down bottom of the tank. The probes for sensing 1/4, 1/2, 3/4 and FULL levels are placed with equal spacing one by one above the bottom positive probe. Consider the topmost (full level) probe, its other end is connected to the base of transistor Q4 through resistor R16. Whenever water rises to the full level current flows into the base of transistor Q4 which makes it ON and so its collector voltage goes low. The collector of Q4 is connected to P2.4 and a low voltage at P2.4 means the over head tank is FULL. When water level goes below the full level probe, the base of Q2

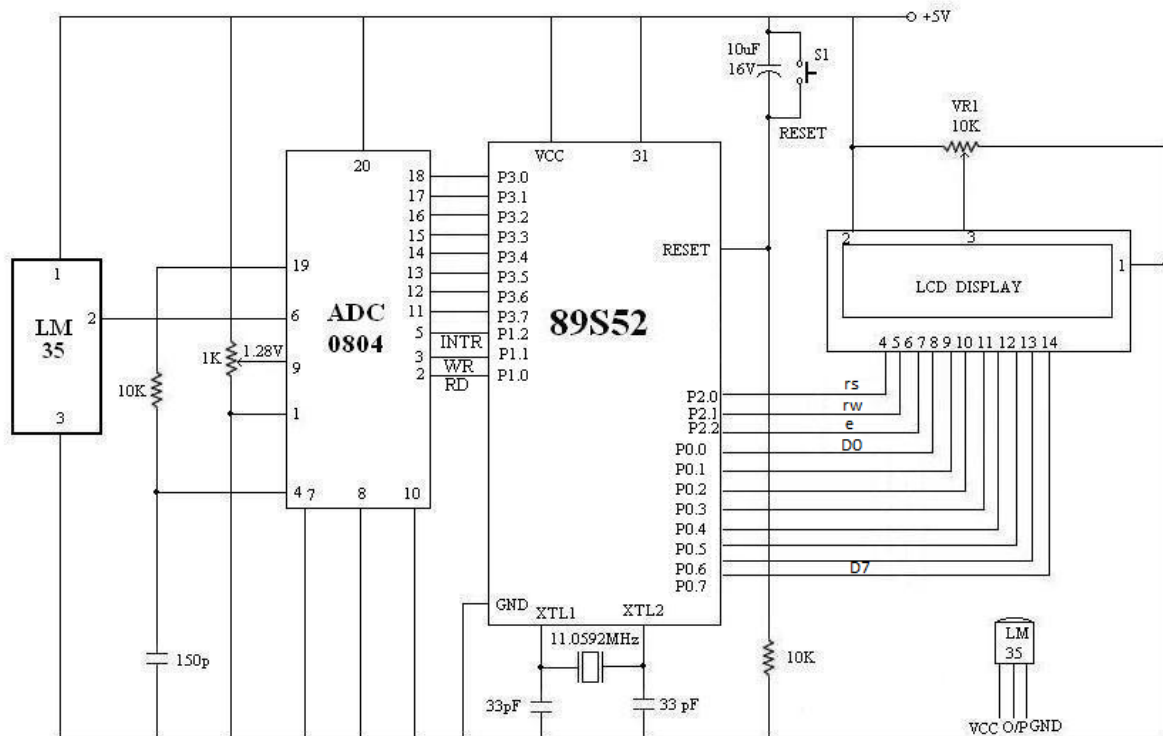
becomes open making it OFF. Now its collector voltage goes high and high at P2.4 means the tank is not full. The same applies to other sensor probes (3/4, 1/2, 1/4) and the microprocessor understands the current level by scanning the port pins P2.4, P2.5, P2.6 and P2.7. All these port pin are high (all sensor probes are open) means the tank is empty.

Port pin P0.5 is used to control the pump. Whenever it is required start pumping, the controller makes P0.5 low which makes transistor Q6 ON which in turn activates the relay K1 that switches the pump. Also the LED d6 glows indicating the motor is ON.

LED D7 is the low sump indicator. When the water level in the sump tank goes low, the controller makes P0.7 low which makes LED D7 to glow. The circuit diagram of the water level controller is shown in the figure below.

```
#include <REGX51.H>
#define led_full P0_0
#define led_threefourth P0_1
#define led_half P0_2
#define led_onefourth P0_3
#define led_empty P0_4
#define pump P0_5
#define sump P0_6
#define led_sump P0_7
#define tank_full P2_4
#define tank_threefourth P2_5
#define tank_half P2_6
#define tank_onefourth P2_7
void main()
{
while(1){
int i = 0;
if(tank_onefourth == 1 && sump == 0)    //if upper tank empty AND sump tank is not
empty
pump = 0;          //then turn ON motor
if(tank_full == 0 || sump == 1) // if upper tank full OR sump tank is empty
pump = 1;          //then turn OFF motor
if(sump == 0)      // if sump tank is not empty
led_sump = 0;      //then turn ON sump LED
else
led_sump = 1;      //else turn OFF sump LED
if(tank_onefourth == 0)    //if upper tank is filled 1/4
i = 4;          //then i=4 to turn ON/OFF LEDs
if(tank_half == 0)        //if upper tank is filled 1/2
i = 3;          //then i=3 to turn ON/OFF LEDs
if(tank_tbf == 0)        //if upper tank is filled 3/4
i = 2;          //then i=2 to turn ON/OFF LEDs
if(tank_full == 0)        //if upper tank is filled full
```

```
i = 1;                //then i=1 to turn ON/OFF LEDs
switch(i)
{
case 1:
led_full = 0;         //turn ON LED full
led_threefourth = 0; //turn ON LED 3/4 full
led_half = 0;         //turn ON LED 1/2 full
led_onefourth = 0;    //turn ON LED 1/4 full
led_empty = 1;        //turn OFF LED empty
break;
case 2:
led_full = 1;         //turn OFF LED full
led_threefourth = 0; //turn ON LED 3/4 full
led_half = 0;         //turn ON LED 1/2 full
led_onefourth = 0;    //turn ON LED 1/4 full
led_empty = 1;        //turn OFF LED empty
break;
case 3:
led_full = 1;         //turn OFF LED full
led_threefourth = 1; //turn OFF LED 3/4 full
led_half = 0;         //turn ON LED 1/2 full
led_onefourth = 0;    //turn ON LED 1/4 full
led_empty = 1;        //turn OFF LED empty
break;
case 4:
led_full = 1;         //turn OFF LED full
led_threefourth = 1; //turn OFF LED 3/4 full
led_half = 1;         //turn OFF LED 1/2 full
led_onefourth = 0;    //turn ON LED 1/4 full
led_empty = 1;        //turn OFF LED empty
break;
default:
led_full = 1;         //turn OFF LED full
led_threefourth = 1; //turn OFF LED 3/4 full
led_half = 1;         //turn OFF LED 1/2 full
led_onefourth = 1;    //turn OFF LED 1/4 full
led_empty = 0;        //turn ON LED empty
}
}
}
```

2) Temperature measurement using LM35, ADC0804, LCD:

- A precision IC sensor LM35 is used here for measurement of temperature. Its resolution is 10 mV/°C. It does not required external calibration. The output of LM35 is in voltage (analog) form. It is converted into digital form by using ADC 0804.
- The microcontroller port P3 is used here for read data from ADC.
- The port pin P1.0, P1.1 and P1.2 are connected to the RD, WR and INTR pins of the ADC respectively.
- P0 is used for send command and data to the LCD.
- The port pin P2.0, P2.1 and P2.2 are connected to the RS, RW and E pins of the LCD respectively.
- The ADC resolution is adjusted to 10mV/step by using potentiometer connected at Vref/2 pin(1.28 V)

```
//----- Temperature Measurement System -----//
```

```
#include <REGX51.H>
```

```
#define port P0 // LCD data lines are connected to Port P0
```

```
sbit rs=P2^0; // LCD rs pin connected at P2.0
```

```
sbit rw=P2^1; // LCD rw pin connected at P2.1
```

```
sbit en=P2^2; // LCD en pin connected at P2.2
```

```
sbit rd=P1^0; // ADC rd pin connected at P1.0
```

```
sbit wr=P1^1; // ADC wr pin connected at P1.1
```

```
sbit intr=P1^2; // ADC intr pin connected at P1.2
```

```
void ini();
```

```
void lcdcmd(unsigned char); //LCD Command function
```

```
void MSDelay(unsigned int); // delay function
```

```
void LCDData(unsigned char);      //LCD data function
void DISPLAY(unsigned char); // Display function
void CONVERT(unsigned int); // data conversion function
void main()
{
    unsigned int value,i,j;
    unsigned char msg1[]="WELCOME";
    unsigned char msg2[]="TEMPERATURE=";
    ini();
    MSDelay(1);
    lcdmd(0x84);
    MSDelay(1);
    for(i=0;i<=6;i++)
    {
        j=msg1[i];
        LCDData(j);
    }
    lcdmd(0xC0);
    MSDelay(1);
    for(i=0;i<=11;i++)
    {
        j=msg2[i];
        LCDData(j);
    }
    intr=1;
    rd=1;
    wr=1;
    while(1)
    {
        wr=0;
        wr=1;
        while(intr==1);
        rd=0;
        value=P3;
        CONVERT(value);
        rd=1;
        MSDelay(10);
        MSDelay(10);
        MSDelay(10);
    }
}
void CONVERT(unsigned int e)
{
    unsigned char x,d1,d2,d3, units, tens, hundreds;
    x=e/10;
    d1=e%10;
    d2=x%10;
    d3=x/10;
    units=(d1+0x30);    //ASCII conversion
    tens=(d2+0x30);     //ASCII conversion
```

```
    hundreds=(d3+0x30);          //ASCII conversion
    lcdmd(0xCC);                  // Cursor at line 2, position 13
    LCDData (hundreds);
    LCDData (tens);
    LCDData (units);
    LCDData('C');
    MSDelay(10);
}
void ini()
{
    lcdmd(0x38);
    lcdmd(0x0C);
    lcdmd(0x01);
    lcdmd(0x06);
    lcdmd(0x80);
    LCDData(' ');
    lcdmd(0x90);
    LCDData(' ');
    lcdmd(0xC0);
    LCDData(' ');
    lcdmd(0xD0);
    LCDData(' ');
}
void lcdmd(unsigned char value)
{
    port=value;
    rs=0;
    rw=0;
    en=1;
    en=0;
    MSDelay(1);
    return;
}
void MSDelay(unsigned int itime)
{
    unsigned int i,j;
    for(i=0;i<=itime;i++)
        for(j=0;j<1275;j++);
}
void LCDData(unsigned char value)
{
    port=value;
    rs=1;
    rw=0;
    en=1;
    en=0;
    MSDelay(1);
    return;
}
```

3) Bluetooth (HC-05) interfacing with 8051 :

Working of Bluetooth module (HC-05)

The HC-05 Bluetooth Module is an easy-to-use Bluetooth serial module. It is designed for transparent wireless serial connection setup. Its communication is via the serial mode, which makes it easy to interface it with any microcontroller.

The HC-05 Bluetooth module provides a switching mode between master and slave mode. A single master device can be connected to up to several different slave devices. But, a slave device in the Bluetooth network can only be connected to a single master. The master device can send data to any of its slaves and request data from them as well, but, the slaves are only allowed to transmit to and receive from their master. They can't talk to other slaves in the network.

The HC-05 Bluetooth module can be configured as a master device or a slave device with the help of AT commands.

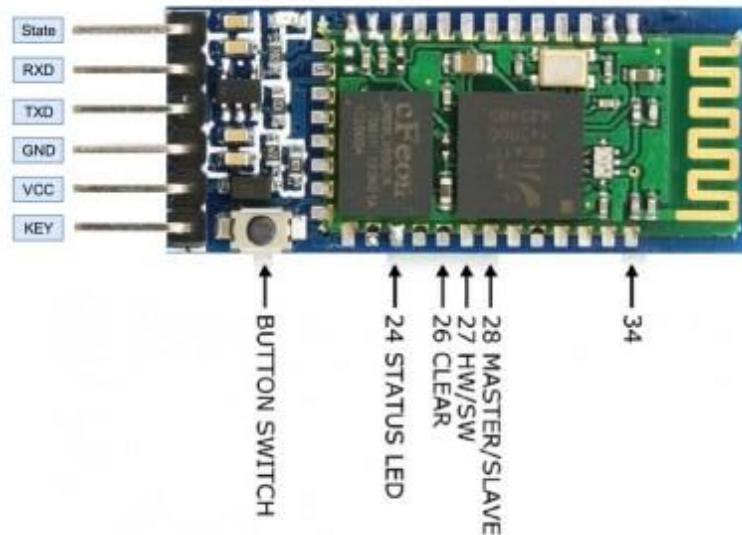
To enable Bluetooth communication, the HC-05 module is interfaced with the 8051 microcontroller via the built-in UART circuitry present in the microcontroller. And the data is transmitted in the form of packets. The 8051 microcontrollers have inbuilt pins P3.0 (RXD) and P3.1 (TXD) for receiving and transmitting the data from/to the HC-05 Bluetooth module respectively.

Features of Bluetooth module

Here are some specifications of the HC-05 Bluetooth module:

- Bluetooth protocol: Bluetooth Specification v2.0 + EDR.
- The frequency is 2.4GHz ISM band.
- The range of operation is less than 100m.
- It uses frequency-hopping spread spectrum (FHSS) radio technology to send data over the air.
- The emission power is less than 4dBm, and it is a Class 2 Bluetooth Module.
- The operating current of the HC-05 Bluetooth module is 30mA.
- The operating voltage of the HC-05 Bluetooth module is 3.3V- 5V.
- Can operate in Master, Slave or Master/Slave mode
- The dimensions are 29.9mm x 13mm x 2.2mm.
- It can be easily interfaced with laptops or smartphones.
- It has a security overlay in the form of authentication and encryption.
- Supported baud rates are 9.6, 19.2, 38.4, 57.6, 115.2, 230.4, 460.8 Kbps.

Pin diagram



Pin Number	Pin Name	Description
1	Enable / Key	This pin is used to toggle between Connection Mode and AT command mode. By default, it is in Connection mode. <i>LOW</i> – Connection Mode, <i>HIGH</i> – AT command mode.
2	VCC	Powers the module. Connect to +5V Supply voltage
3	GND	Ground pin of the module, connect to system ground.
4	TXD – Transmitter	Transmits Serial Data. Everything received via Bluetooth will be given out by this pin as serial data.
5	RXD – Receiver	Receive Serial Data. Every serial data given to this pin will be broadcasted/ transmitted via Bluetooth.
6	State	The state pin is connected to an on-board LED, and it can be used as feedback to check if Bluetooth is working properly.
7	LED	Indicates the status of Module
		Repeated Blinking: Waiting for connection in Connection Mode
		Blink twice in 1 sec: Connection successful in Connection Mode
		Blink once in 2 sec: Module has entered AT Command Mode

8	Button	Used to control the Key/Enable pin to toggle between Connection and command Mode
---	--------	--

Modes of operation – AT and Connection

AT Command Mode

- Whenever the Enable (EN)/key command is set HIGH, the AT command mode is activated.
- The AT commands are set of commands which are used to set up and configure the Bluetooth module.
- In this mode, the module is not discoverable by the other Bluetooth devices.
- The commands are sent to the module serially like string.
- The string should be in the *capital letters*, and every command should be appended with 'rn' at the end.
- If the command is known by the module, then the module will reply with the string: 'OK'.
- If the command is unknown by the module, then it will reply with the string: ERROR ().
- The error determines the type of error and a specific code.

```
Enter AT commands:
OK
+ADDR:98d3:33:811471
OK
+PSWD:1234
OK
+VERSION:2.0-20100601
OK
```

- The AT commands are used to set the password for Bluetooth pairing or change the name of the connection.
- Also, to check or change the baud rate, AT commands are written in the Bluetooth Terminal.

Connection Mode

- When the Enable (EN)/key is LOW, the connection mode is set.
- In the connection mode, this device can connect directly to any other device.
- We can discover/ identify this device on other devices and can connect with it using a default passkey which is usually '1234' or '0000'.
- The configuration cannot be changed, but we can communicate with it serially.

Steps to set up the mobile app:

- **Step 1:** Download any Bluetooth terminal application on your Smartphone. Just launch the app store and search for "BT Terminal" or "Bluetooth Terminal" and install any application with a good rating.
- **Step 2:** Go to the Bluetooth settings on your phone and click on "Add new device". Select the device named "HC-05", the default passkey is "1234" or "0000".
- **Step 3:** Now that your smartphone is paired with the Bluetooth module, which is connected to the 8051 microcontroller, you can send data serially into the microcontroller via Bluetooth.
- **Step 4:** For that, open the application in your phone and make sure you select the required baud rate. The default baud rate of the HC-05 module is 9600 bps.
- **Step 5:** The setup is complete now. All you need to do is program the microcontroller to respond to whatever serial data you'll be sending via the Bluetooth module.

Working principle of the project

Our main goal is to receive information wirelessly via the Bluetooth module connected to the microcontroller to control the pins P0.0 and P0.1, which are connected to the LEDs via a relay.

Here, we're using a current amplifying IC – the ULN2003a (also called the *relay driver*), because the output current of pins of 8051 is a maximum of 10mA which is not sufficient. The output of this relay driver is given to a relay to which the appliance is connected, in our case, the LEDs.

We'll execute the following logic to receive characters from the smartphone via Bluetooth and control the LEDs.

- If the character 'A' is received, P0.0 is set to LOW, which implies LED 1 is turned ON.
- If the character 'a' is received, P0.0 is set to HIGH, which implies LED 1 is turned OFF.
- If the character 'B' is received, P0.1 is set to LOW, which implies LED 2 is turned ON.
- If the character 'b' is received, P0.1 is set to HIGH, which implies LED 2 is turned OFF.

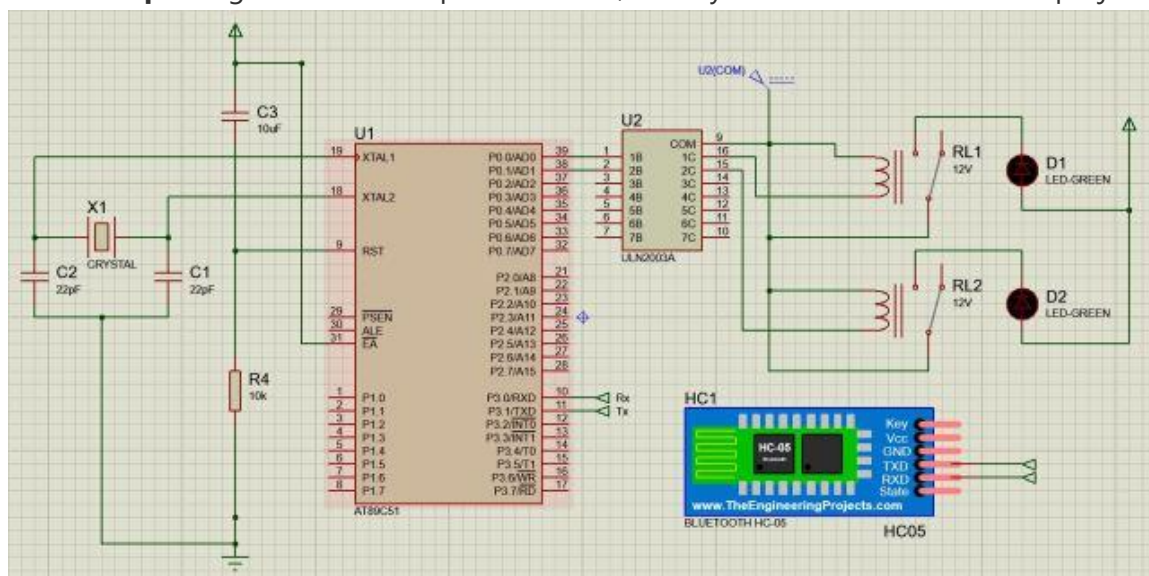
All this is possible due to serial communication with UART present in the 8051 microcontrollers. To summarize the configuration in which we execute the serial communication,

1. Timer 1 is selected as a timer in Mode 2 (8-bit auto-reload mode) by using TMOD = 0x20; command.
2. The UART frame consists of 10-bit of data to be transmitted or received, including *one start bit* and *one stop bit*.
3. A baud rate of 9600 bps is set by using TH1 = 0xFD; command.

4. The SCON register (Serial Control) is set in such a way that it is selected in Mode 2 and the REN (reception) bit is set by using SCON = 0x50; command.

Circuit diagram to interface Bluetooth module and relay with 8051

- **Step 1:** If you're using Proteus or any other simulation software or even hardware, select the AT89C51 or AT89S51 microcontroller or any other compatible variant.
- **Step 2:** Connect a 12 MHz oscillator between pin 18 and 19.
- **Step 3:** Connect two capacitors of 22pF, with one terminal on either side of the oscillator and the other terminal to ground, as shown below.
- **Step 4:** Set Pin 31, i.e., EA pin to HIGH by connecting it to the +5V DC source.
- **Step 5:** Now, to make the RESET circuit, connect Pin 9 (RST) to +5V through a capacitor of 10 μ F and connect the same pin to +0V (GND) through a 10k Ω resistor or a potentiometer.
- **Step 6:** Connect two LEDs to the microcontroller such that they turn ON when a low pulse is given through Port 0's output. For this, ULN2003a driver is used, pin P0.0 and P0.1 are connected to 1B and 2B of ULN2003A respectively.
- **Step 7:** The output pins 1C and 2C of ULN2003a are given to one terminal of the coil of the relay and the other terminal of the coil of the relay is connected to the COM pin of ULN2003A which is connected to a DC supply.
- **Step 8:** One side of the contactor of the relay is connected to the cathode of the LED, and the anode is connected to the power supply. Connect the other side of the contactor to the COM pin of ULN2003A.
- **Step 9:** Connect the RXD (P3.0) and TXD (P3.1) of the 8051 microcontrollers to TXD and RXD pins of HC-05 Bluetooth module respectively.
- **Step 10:** VCC and GND pins of the HC-05 Bluetooth module are to be given to +5V/3.3V and GND, respectively, for the module to turn ON.
- **Step 11:** Ignore the other pins of HC-05, as they do not contribute to our project.



C Program to interface the HC-05 Bluetooth module with 8051

```
#include<reg51.h>

#include<stdio.h>

sbit LED1 = P0^0; //Assign LED1 to pin P0.0

sbit LED2 = P0^1; //Assign LED2 to pin P0.1

void main()

{

    char mychar; // Assign a variable for the character to be received

    TMOD = 0x20; // Set time Timer 1 in 8-bit auto-reload mode (mode 2)

    TH1 = 0xFD; // FD = -3, to send/ receive the data at 9.6Kbps

    SCON = 0x50; // Serial Control is set in Mode 1, with REN = 1 (reception ON)

    TI = 0; // Initialize the Timer Flag

    RI = 0; // Initialize the Timer

    TR1 = 1; // Start the timer (Timer 1)

    while(1) // Runs forever until power supply is cutoff

    {

        while (RI == 0); // Wait till the data is recieved, as soon as the data is received
        completely, the RI flag is set

        RI = 0; // Clear the RI flag to declare the availability for reception of another data


        if (mychar == 'A') // If the data received through the Bluetooth module is the character
        'A'

        {

            LED1 = 0; // Turn ON LED 1

        }

    }

}
```

```
    else if (mychar == 'a') // If the data received through the Bluetooth module is the
character 'a'

    {

        LED1 = 1; // Turn OFF LED 1

    }

    if (mychar == 'B') // If the data received through the Bluetooth module is the character
'B'

    {

        LED2 = 0; // Turn ON LED 2

    }

    else if (mychar == 'b') // If the data received through the Bluetooth module is the
character 'b'

    {

        LED2 = 1; // Turn OFF LED 2

    }

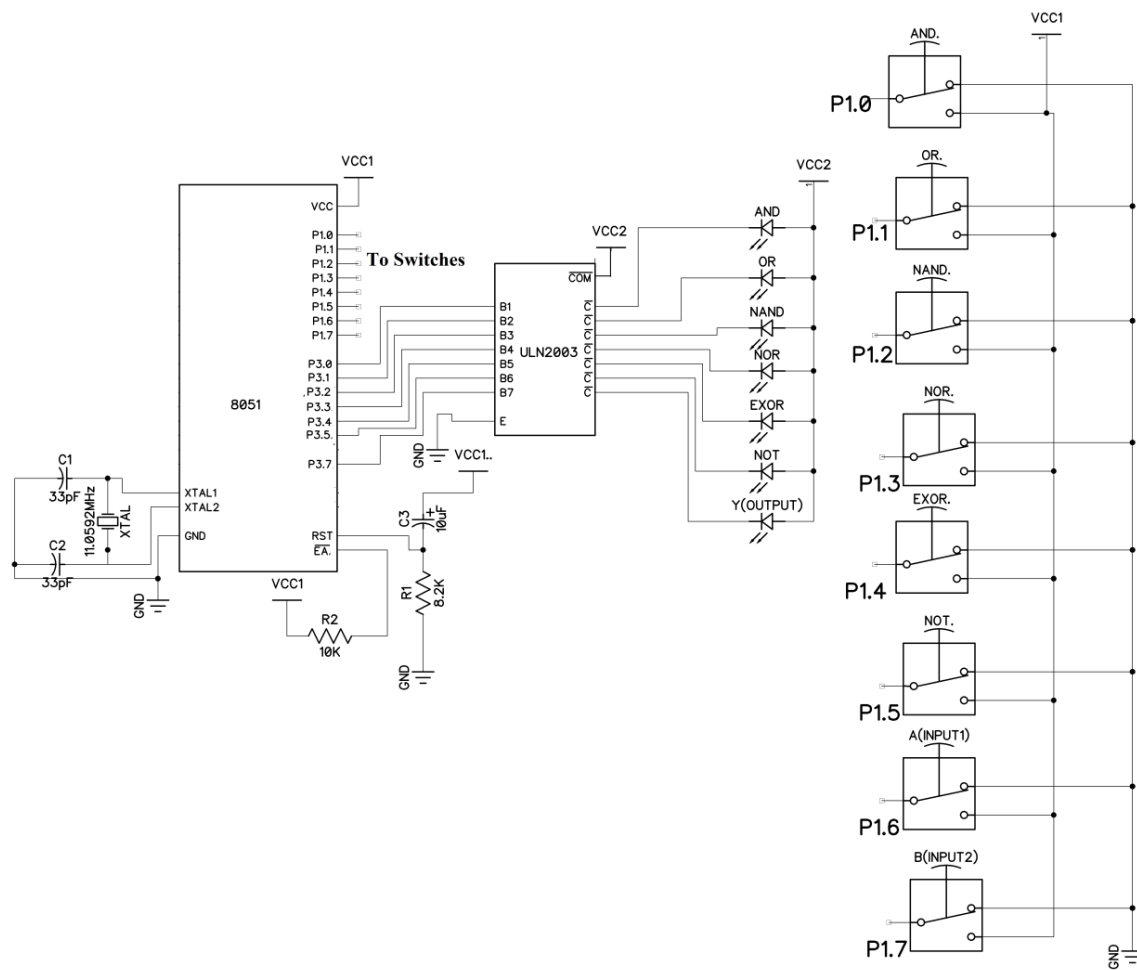
}

}
```

4) Gate Emulator (Logic Gate study using microcontroller)

To study logic gates a numbers of ICs are required for AND gate 7408, OR gate 7432, NAND gate 7400, NOR gate 7402, EX-OR gate 7486 and NOT gate 7404 i.e for each gate required separate IC. Therefore, studying various gates become costly and bulky. Therefore, a system is needed using that we can study the all the logic gates. Keeping this in mind using a single microcontroller 8051 IC a gate emulator system is developed.

The complete circuit schematic of the Gate Emulator system using 8051 is as shown in Fig 4.1. Here five SPDT switches are used as selector for particular gate operation. They are connected at port P1 pins P1.0-P1.5. The two inputs (A and B) are given by using switches connected at P1.6 and P1.7. The 8051 receives input from P1.6 and P1.7 produce the output at pin P3.7 according to the selected gate. The ULN2003 is used as current buffer for drive LED. The ULN 2003 consists seven inverted logic buffer. The LEDs are connected in sinking mode therefore it will be ON for logic '1' and OFF for logic '0' due to the inverting buffer. Gate is selected when port pin at logic '0'.



```
#include <REGX51.H>
```

```
sbit AND= P1^0;
```

```
sbit OR= P1^1;
```

```
sbit NAND= P1^2;
```

```
sbit NOR= P1^3;
```

```
sbit EXOR= P1^4;
```

```
sbit NOT= P1^5;
```

```
sbit Ainput= P1^6;
```

```
sbit Binput= P1^7;
```

```
sbit Youtput= P3^0;
```

```
void main(void)

{while(1)

{

if(AND==1)

Youtput=(Ainput & Binput);

if(OR==1)

Youtput=(Ainput | Binput);

if(NAND==1)

Youtput=~(Ainput & Binput);

if(NOR==1)

Youtput=~(Ainput | Binput);

if(EXOR==1)

Youtput=(Ainput ^ Binput);

if(NOT==1)

Youtput=~Ainput;

else

Youtput=1;

}

}
```