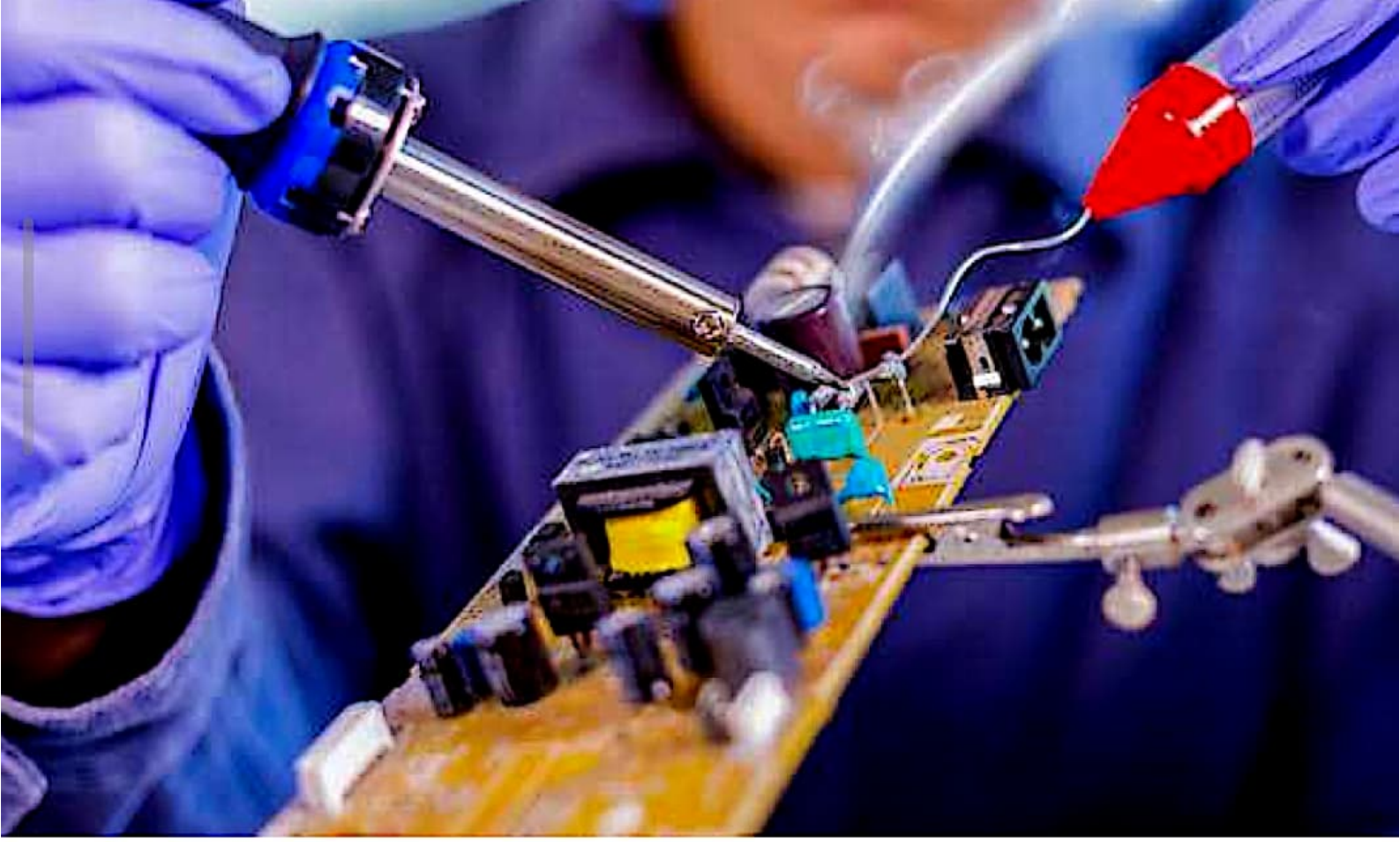
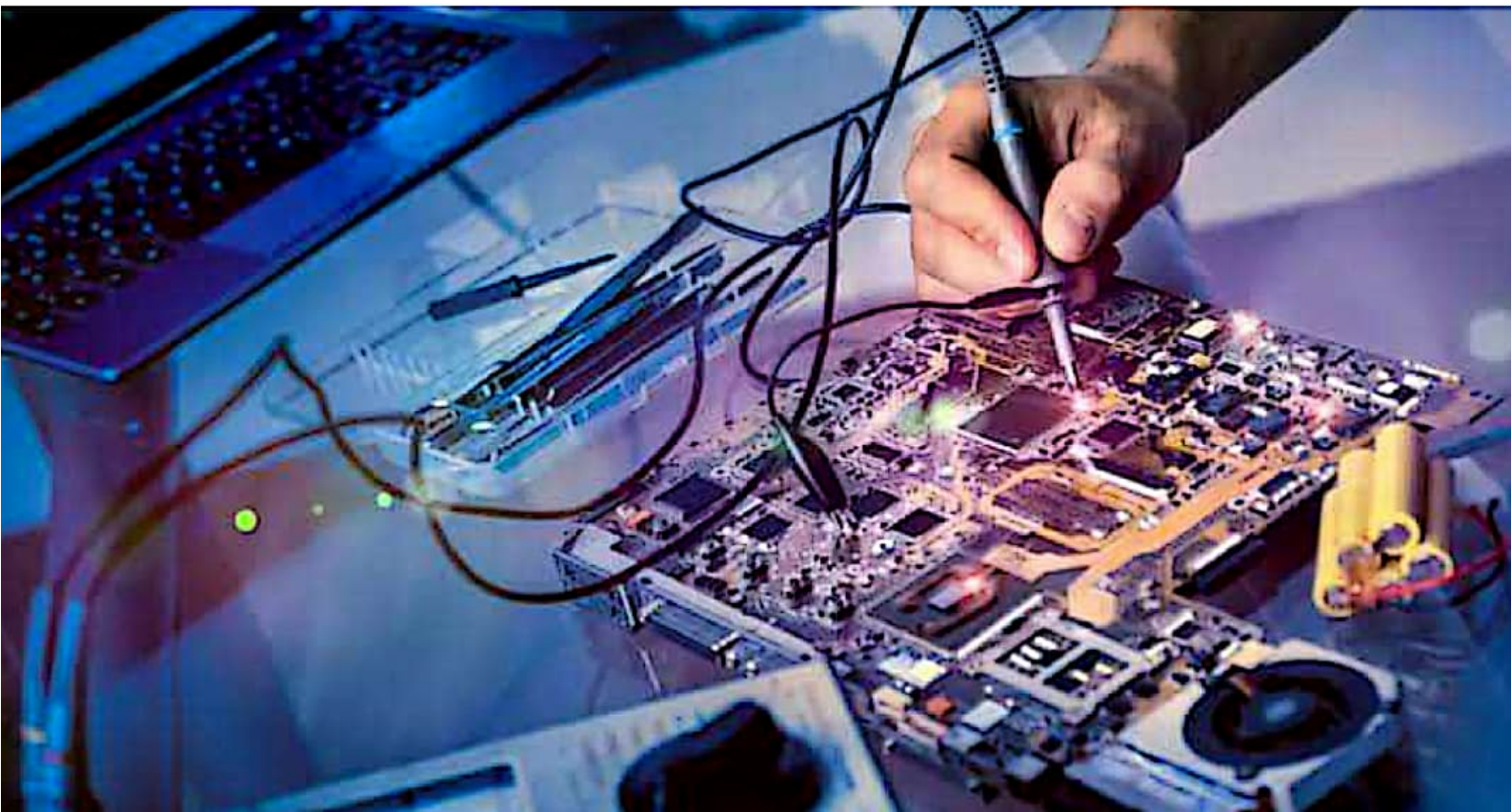


DIGITAL CLOCK USING : 8051 MICROCONTROLLERS



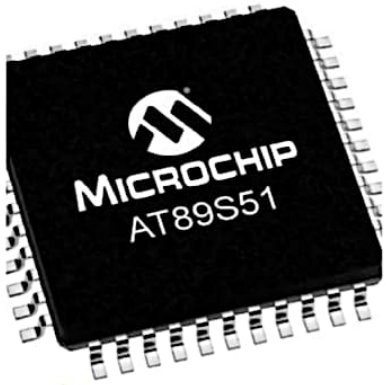
PROJECT

Title : Digital Clock using 8051 Microcontroller and LCD Display

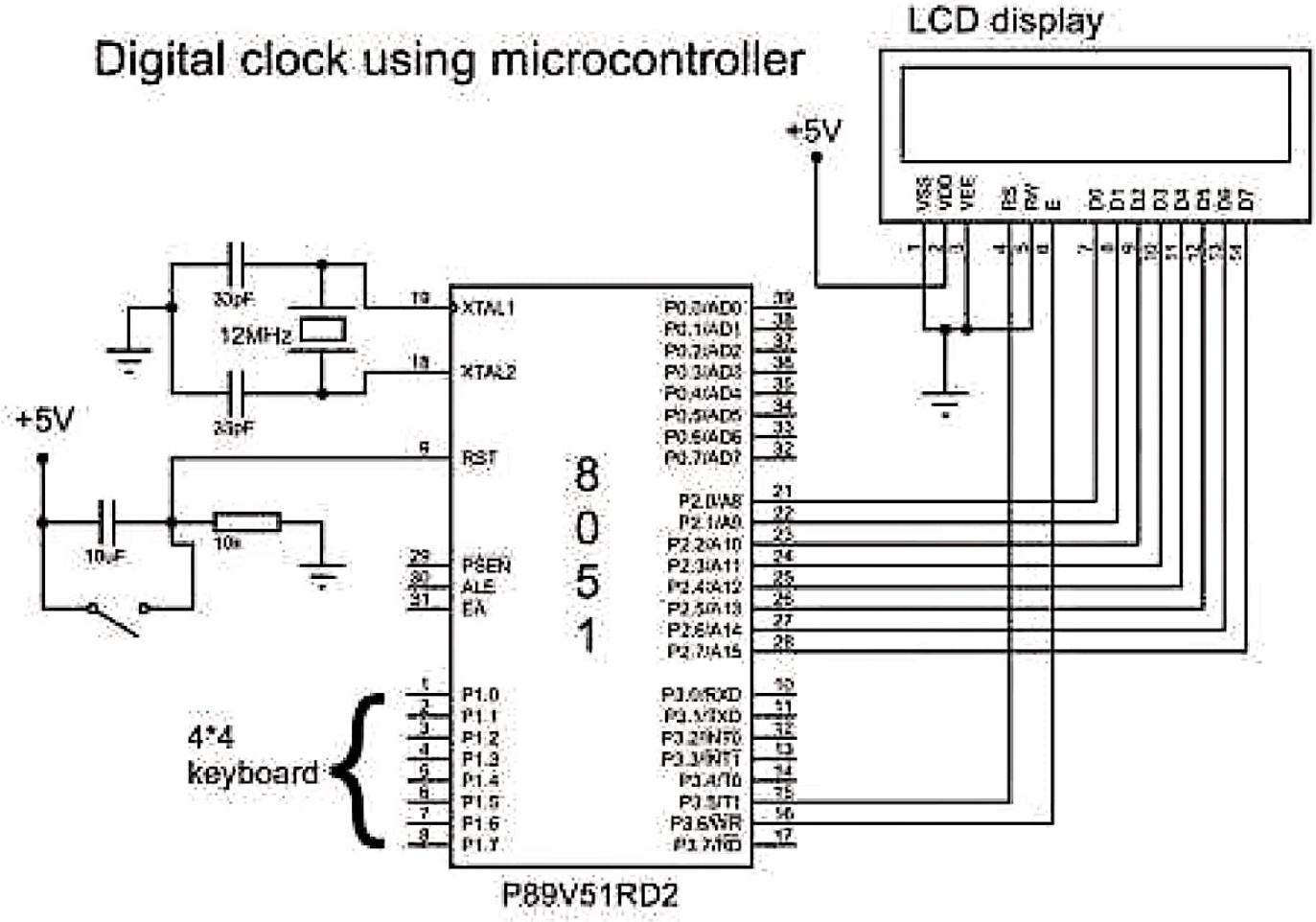
Objective : The objective of this project is to design and implement a digital clock using the AT89C51 (8051) microcontroller. The system displays real-time hours, minutes, and seconds in the format hh:mm:ss on a 16x2 LCD display. This project aims to demonstrate the use of embedded C programming for time management and LCD interfacing in a microcontroller-based system.

Components used:

Component	Quantity
AT89C51 MCU	1
16x2 LCD	1
Crystal Oscillator (11.0592 MHz)	1
Capacitors (33pF)	2
Resistor (10k)	1
Push Buttons (Optional for reset)	1-2
Power supply	5V DC



Circuit Diagram :



Working Explanation :

The project is designed to function as a basic digital clock using the 8051 microcontroller and a 16x2 LCD. Here's how it works step by step:

1) Initialisation Phase :

The LCD is initialized using the lcd_init() function.

LCD is set to 2-line mode, cursor off, and display is cleared.

2) Time Variables :

Three variables: hr, min, and sec are used to store hours, minutes, and seconds respectively.

An 8-character array time[9] stores the formatted time as a string (e.g., "12:34:56")

3) Displaying Time :

In the display_time() function:

The numeric values of hours, minutes, and seconds are converted into ASCII characters using (value + 48).

The formatted string is then sent to the LCD character-by-character using the lcd_data() function.

4) Time Update Logic :

The update_time() function increases the seconds by 1 after every delay.

When seconds reach 60, they reset to 0 and minutes are incremented.

Similarly, when minutes reach 60, they reset and hours are incremented.

When hours reach 24, they reset to 0 (24-hour format).

5) Delay :

A simple software delay (delay(1000)) is used to simulate a one-second pause.

This is done using nested loops (though it's not highly accurate, it's sufficient for basic demonstration).

6) Continuous Operation :

Inside the while(1) loop:

The time is displayed.

Then it's updated.

This loop continues indefinitely, making the digital clock run continuously.

NOTE :

Time accuracy may slightly vary because the software delay is not perfectly precise.

For real-time accuracy, a hardware-based Real Time Clock (RTC) like DS1307 can be used in future upgrades.

Microcontroller Used: AT89C51
Programming Language: Embedded C
Compiler: Keil µVision.

Code :

```
#include<reg51.h>

// Define ports and variables
sbit rs = P2^0;      // Register select pin
sbit rw = P2^1;      // Read/write pin
sbit en = P2^2;      // Enable pin

unsigned char hr = 0, min = 0, sec = 0;
char time[9] = "00:00:00";

// Function prototypes
void delay(unsigned int);
void lcd_cmd(unsigned char);
void lcd_data(unsigned char);
void lcd_init();
void display_time();
void update_time();

void main() {
lcd_init();
lcd_cmd(0x80); // Move cursor to first line
while(1) {
display_time();
update_time();
}
}

void delay(unsigned int t) {
unsigned int i, j;
for(i=0; i<t; i++)
for(j=0; j<1275; j++);
}

void lcd_cmd(unsigned char cmd) {
P1 = cmd;
rs = 0;
rw = 0;
en = 1;
delay(1);
en = 0;
```

```
}
void lcd_data(unsigned char dat) {
P1 = dat;
rs = 1;
rw = 0;
en = 1;
delay(1);
en = 0;
}
void lcd_init() {
lcd_cmd(0x38); // 2 line, 5x7 matrix
lcd_cmd(0x0C); // Display ON, Cursor OFF
lcd_cmd(0x06); // Auto increment cursor
lcd_cmd(0x01); // Clear display
delay(2);
}
void display_time() {
time[0] = (hr / 10) + 48;
time[1] = (hr % 10) + 48;
time[3] = (min / 10) + 48;
time[4] = (min % 10) + 48;
time[6] = (sec / 10) + 48;
time[7] = (sec % 10) + 48;

lcd_cmd(0x80); // Start from first line
for(int i = 0; i < 8; i++) {
lcd_data(time[i]);
}
delay(1000);
}
void update_time() {
sec++;
if(sec == 60) {
sec = 0;
min++;
if(min == 60) {
min = 0;
hr++;
if(hr == 24) {
hr = 0;
}
}
}
}
```

Output :

The output of this project is a real-time digital clock displayed on a 16x2 LCD screen, which continuously shows the current time in the format:

hh:mm:ss

For example: 09:42:17, 14:55:03, 23:59:59

The time starts from 00:00:00 and keeps incrementing every second.

When seconds reach 60, they reset to 00 and minutes are incremented.

Similarly, hours update after 60 minutes and reset after 24 hours (i.e., a 24-hour format).

The LCD display updates in real-time without flickering, giving a clean and readable output.

This demonstrates successful time management and LCD interfacing using the 8051 microcontroller.

Applications:

This digital clock project using the 8051 microcontroller has several practical and educational applications:

1) Basic Timekeeping Devices :

Can be used in DIY digital clocks for homes, schools, or small offices.

2) Embedded systems Learning :

A great beginner-level project for understanding microcontroller programming, LCD interfacing, and time management.

3) Industrial Timers :

Can be adapted as part of timer-based automation in factories or production lines.

4) Appliances Integration :

Can be used in microwave ovens, washing machines, and other appliances where timing functionality is required.

5) Alarm Clocks (With Upgrade) :

With the addition of push buttons and buzzer, it can be converted into a working digital alarm clock.

6)Real-time System(RTC Upgrade)

By interfacing with a Real-Time Clock (RTC) module like DS1307, it can be made highly accurate for real-time applications.

Future scope :

This project can be further improved and expanded in various ways to increase its accuracy, functionality, and usability:

1) Integration with RTC Module :

Replacing the software-based delay with a Real-Time Clock (RTC) like DS1307 or DS3231 will enhance time accuracy and allow battery backup.

2) Alarm Feature :

Adding push buttons and a buzzer circuit can convert this into a full-featured digital alarm clock.

3) Time setting Interface :

Including buttons or a keypad to manually set the current time will make the system more practical.

4) Power Backup Using Battery :

Including a battery backup system will allow the clock to run even during power cuts.

5) Display Enhancement :

Using an OLED or 7-segment display instead of LCD for better visibility and compactness.

6) Wireless Clock Synchronisation

Integration with modules like Bluetooth, Wi-Fi (ESP8266), or RF can enable syncing time wirelessly or through a mobile app.

7) Real-time Data Logging :

Time data can be logged to an SD card or transmitted to a computer for time-stamped monitoring applications.

Conclusion :

In this project, a digital clock was successfully designed and implemented using the 8051 microcontroller and a 16x2 LCD display. The system accurately keeps track of time in the hh:mm:ss format and updates the display in real time using embedded C programming.

This project demonstrates the core concepts of microcontroller programming, LCD interfacing, and basic time logic implementation, making it an excellent learning experience for beginners in embedded systems. Although the current version uses a software-based delay, it can be further enhanced with advanced features such as RTC integration, alarm functions, and wireless control.

Overall, this project serves as a strong foundation for developing more complex embedded applications involving real-time data processing and human-machine interfacing..