

# Maruthi M

## BU21EECE0100554

### 8051(MICRO-CONTROLLER) (Trainer: Dr. Jeevan K M)

Learning Objective:

- Understanding how to create patterns of LED blinking using the 8051

microcontroller. Inputs and Outputs:

- Inputs: None
- Outputs: LED states (ON/OFF)

Logic:

- LEDs are connected to the microcontroller. By assigning a hexadecimal value to the port, the microcontroller converts this value to binary. Each bit in the binary representation controls an individual LED: a bit value of '1' turns the LED on, and '0' turns it off. For example, if  $P1 = 0x01$ , only the first LED (connected to the least significant bit) will be on, and the rest will be off.
- The other terminals of the LEDs are connected to the ground, creating a simple circuit for controlling the LEDs based on the port value.
- Various LED blinking patterns can be implemented using this basic principle. Two main approaches can be utilized:
  1. Direct assignment of hexadecimal values to the port.
  2. Using bitwise left shift operations within a loop to create sequential blinking patterns.

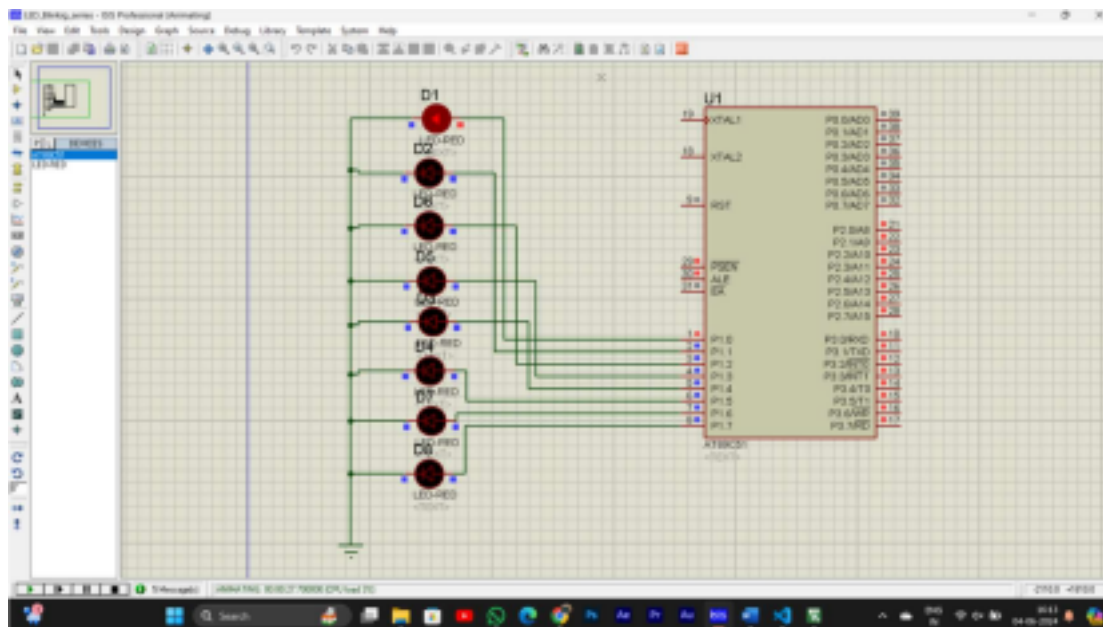
Common Mistakes:

- Syntax Errors: Common when writing embedded C code.
- Indentation Errors: Proper code formatting is crucial to avoid logical errors.
- Port Mismatch: Ensuring the correct port is being used for LED control is essential to achieve the desired output.

```

1 #include<reg51.h>
2 //sbit sw1 = P2^1;
3 //sbit sw2 = P2^2;
4 //void dealy(void);
5 void delay(unsigned int);
6 void main(void)
7 {
8     unsigned char k = 0x01;
9     unsigned int n;
10
11     while(1)
12     {
13         for( n=0;n<=7;n++){
14             P1 = k;
15             delay(500);
16             k=k<<1;
17         }
18         if(k==0){
19             k=0x01;
20         }
21     }
22 }
23
24
25
26
27
28
29
30 // sw1 =0;
31 // sw2 = 0;
32 /*while(1)
33 {
34     /* if(sw1==0 && sw2 ==0 )
35     {
36         P1 = 0x00;

```



## ADC Using 8051

### Learning Objective:

- Understanding the process of converting an analog signal to a digital signal using the ADC 0808.
- Learning how to interface the ADC 0808 with the 8051

microcontroller. Inputs and Outputs:

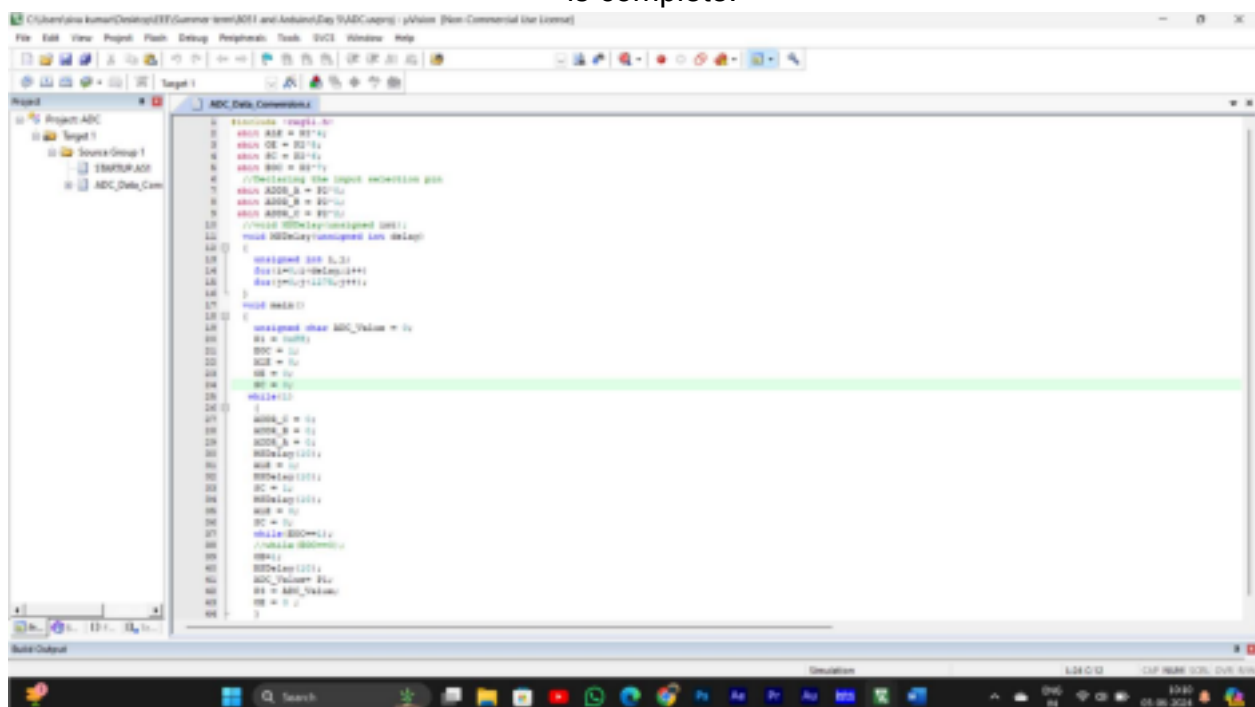
- Input: A potentiometer (1k or 10k ohms).
- Outputs: LEDs.

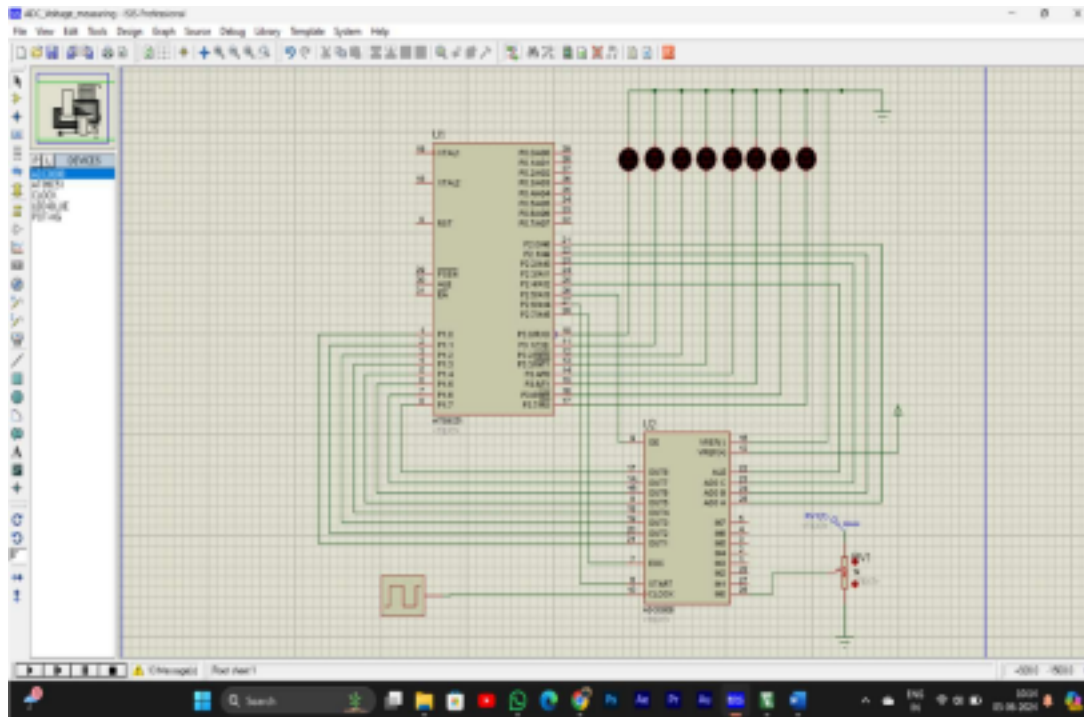
Logic:

- The analog output from the potentiometer is fed into one of the input channels of the ADC 0808.
- The digital output pins (D0-D7) of the ADC 0808 are connected to Port 1 of the 8051 microcontroller.
- The selection lines (A, B, C) of the ADC are connected to Port 2 of the 8051 to select the input channel.
- Additional control lines such as End of Conversion (EOC), Start, Address Latch Enable (ALE), and Clock Pulse are also connected to Port 2.
- The LEDs, which display the digital value of the potentiometer reading, are connected to Port 3 of the 8051 microcontroller.
  - Based on the variation in resistance of the potentiometer, the digital value output from the ADC will change, which in turn will change the state of the LEDs connected to Port 3.

Common Mistakes:

- Selection Line Configuration: Double-check the configuration and connections of the selection lines (A, B, C) to ensure the correct input channel is selected on the ADC 0808.
- EOC Pin Connection: Ensure the End of Conversion (EOC) pin of the ADC 0808 is properly connected to the 8051 to correctly detect when a conversion is complete.

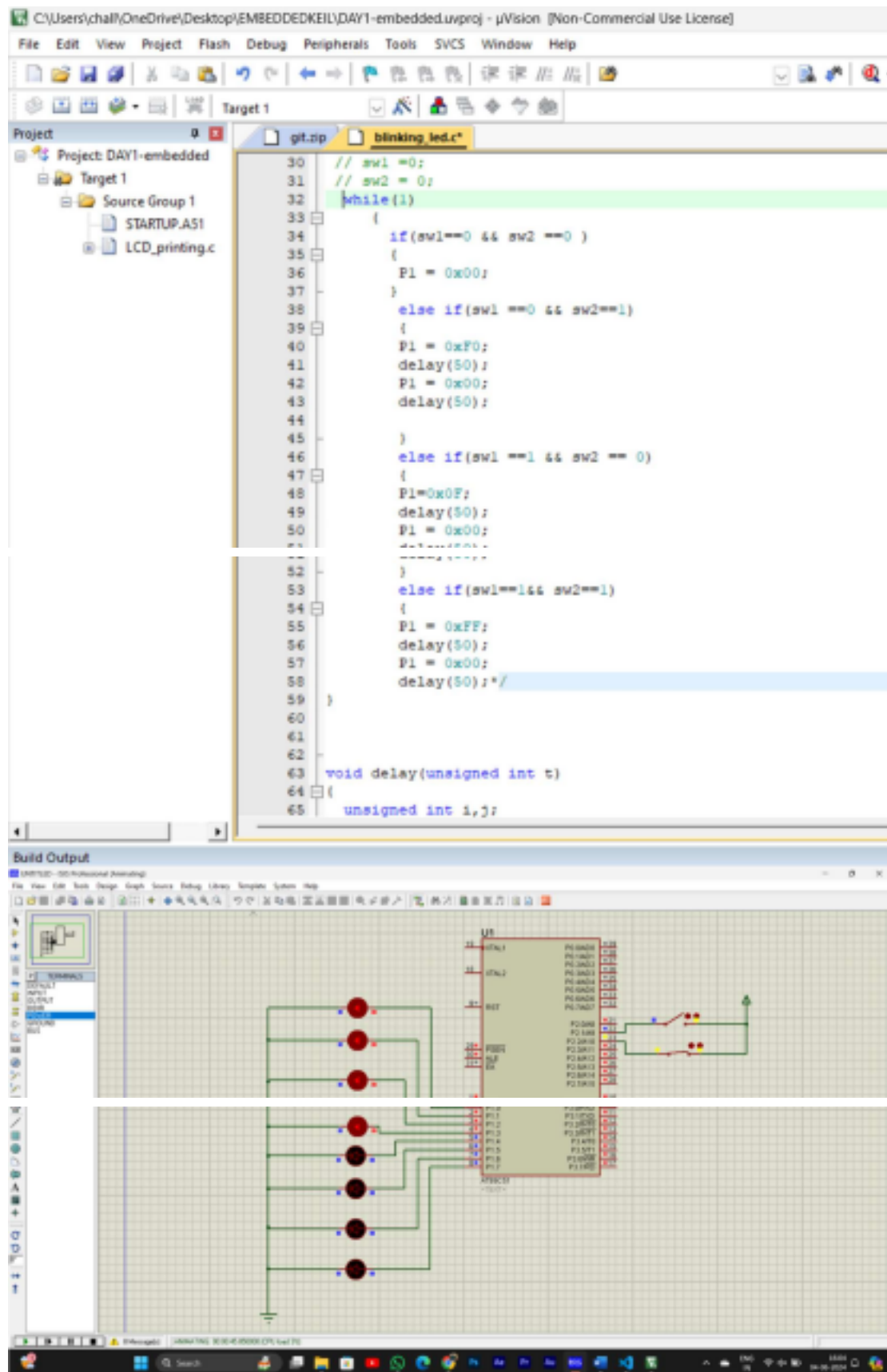




## 8051

### Learning Objective:

- ## 🎬 Learning how to control LED blinking using the 8051



Voltage Measurement and LCD displaying(8051)

Learning Objective:

- Understanding how to measure voltage variations using an ADC.
- Displaying the voltage variations on LEDs.
- Displaying the voltage measurement status on an LCD display.

## Inputs and Outputs:

- Input: Potentiometer.
- Outputs: LEDs and LCD display.

## Logic:

1. Connections:
  - Connect the potentiometer output to an input channel of the ADC 0808.
  - Provide a 5V power supply to the ADC 0808.
  - Connect the digital output pins (D0-D7) of the ADC 0808 to a port on the 8051 microcontroller.
  - Connect the LCD display and LEDs to the 8051 microcontroller.
2. Operation:
  - The ADC converts the analog signal from the potentiometer to a digital signal.
  - The 8051 microcontroller reads the digital signal from the ADC.
  - Based on the digital value, control the LEDs to indicate voltage levels.
  - Convert the digital value to a corresponding voltage value and display it on the LCD.

## Common Mistakes:

- Port Mismatch: Ensure that the port names used in the code match the actual circuit connections.
- Proper Connections: Verify that all connections are secure. A single loose connection can lead to data loss and no output.
- Input Levels: Ensure that input levels to the microcontroller and ADC are correct (either high or low as required).









```

65 while(1) {
66     adc_value = adc_read(); // Read ADC value
67     voltage = adc_to_voltage(adc_value); // Get voltage from lookup table
68
69     lcd_command(0x01); // Move cursor to the second line of LCD
70     lcd_display_voltage(voltage); // Display the voltage on LCD
71
72     if (voltage > 3.0) {
73         LED0 = 1; // Turn on LED0
74         LED1 = 1; // Turn on LED1
75         lcd_command(0x01); // Move cursor to display status
76         lcd_display_status("LED0 ON");
77     } else if (voltage > 2.0) {
78         LED0 = 1; // Turn on LED0
79         LED1 = 0; // Turn off LED1
80         lcd_command(0x01); // Move cursor to display status
81         lcd_display_status("LED0 ON");
82     } else {
83         LED0 = 0; // Turn off LED0
84         LED1 = 0; // Turn off LED1
85         lcd_command(0x01); // Move cursor to display status
86         lcd_display_status("LED0 OFF");
87     }
88
89     delay(100); // Delay for some time (100 ms)
90 }
91
92 // Function to initialize the LCD
93 void lcd_init(void) {
94     lcd_command(0x01); // Initialize LCD in 8-bit mode
95     delay(10);
96     lcd_command(0x02); // Display ON, Cursor OFF
97     delay(10);
98     lcd_command(0x04); // Auto increment cursor
99     delay(10);
100    lcd_command(0x05); // Clear display
101    delay(10);
102    lcd_command(0x06); // Move cursor to the first line
103    delay(10);
104 }
105
106 // Function to send command to LCD
107 void lcd_command(unsigned char command) {
108     P0 = command;
109     RS = 0; // Select command register
110     RW = 0; // Write operation
111     EN = 1;
112     delay(10);
113     EN = 0;
114     delay(10);
115 }
116
117 // Function to send data to LCD
118 void lcd_data(unsigned char data) {
119     P0 = data;
120     RS = 1; // Select data register
121     RW = 0; // Write operation
122     EN = 1;
123     delay(10);
124     EN = 0;
125 }

```

```

51 // Function to initialize the LCD
52 void lcd_init(void) {
53     lcd_command(0x01); // Initialize LCD in 8-bit mode
54     delay(10);
55     lcd_command(0x02); // Display ON, Cursor OFF
56     delay(10);
57     lcd_command(0x04); // Auto increment cursor
58     delay(10);
59     lcd_command(0x05); // Clear display
60     delay(10);
61     lcd_command(0x06); // Move cursor to the first line
62     delay(10);
63 }
64
65 // Function to send command to LCD
66 void lcd_command(unsigned char command) {
67     P0 = command;
68     RS = 0; // Select command register
69     RW = 0; // Write operation
70     EN = 1;
71     delay(10);
72     EN = 0;
73     delay(10);
74 }
75
76 // Function to send data to LCD
77 void lcd_data(unsigned char data) {
78     P0 = data;
79     RS = 1; // Select data register
80     RW = 0; // Write operation
81     EN = 1;
82     delay(10);
83     EN = 0;
84 }

```

## 8051(micro-controller) using LCD Display

### Learning Objective:

- Learning how to use the 8051 microcontroller to display text on an LCD.

### Inputs and Outputs:

- Inputs: Commands for the LCD display.
- Output: Displayed text on the LCD.

## Logic:

- Implement functions to send commands and data to the LCD.
- Use various LCD commands such as clear display, cursor positioning, and line shifting.
- Convert numerical values to ASCII before displaying them on the LCD.
- Make basic connections to power supply, ground, and enable pins.

## Common Mistakes:

- Command Mismatch: Ensure the correct commands are sent to the LCD.
- Command Positioning: Correct positioning of commands is crucial for proper display.
- Clear Display and New Line Commands: Use appropriate commands to clear the display or move to a new line when needed.

