Mirpur University of Science and Technology

(MUST)

Name: Amna wafa, Haleema Sajid

Tayyiba Afraiz, Zara Sajjad

Roll No: FA22-BSE-05, FA22-BSE-011

FA22-BSE-027, FA22-BSE-029

Section: A

Session: 2022-26

Subject: Embedded Systems

Submitted to: Engr. Umair

Open Ended Lab Task

Real Time Video Streaming with Object Detection

Required Apparatus:

1. Microcontroller (AT89C51)

2. LCD display

3.Crystal Capacitor

4. Ceramic Capacitor

5.Virtual terminal

Working of the Circuit:

1. Microcontroller (AT89C51)

- The **AT89C51** microcontroller is an **8051-based** microcontroller with **4K Flash memory**.
- It controls the LCD and processes input/output signals.
- However, it cannot handle video streaming directly. Instead, it could be used to **interface** with an external system, such as a camera module or a PC that processes video.
- It may trigger object detection alerts when connected to a processing unit (e.g., Raspberry Pi).

2. LCD Display

- The **LCD display (16x2 or similar)** is used to display messages or detected object information.
- The **AT89C51** sends commands to display text, such as "Object Detected" or "Streaming Started."
- The LCD is typically interfaced using data pins (D0-D7) and control pins (RS**,** RW, E)**.**

3. Crystal Oscillator & Capacitors

- The **crystal oscillator (usually 11.0592 MHz)** provides the **clock signal** for the microcontroller.
- The **ceramic capacitors (22pF)** stabilize the oscillator, ensuring reliable operation.
- The **crystal** helps in generating baud rate for serial communication if a UART is used.

4. Virtual Terminal (Simulation Purpose)

- If you are using Proteus simulation, the virtual terminal acts as a serial communication monitor**.**
- It can display messages sent via the UART (serial port) of AT89C51.
- It helps debug communication between the microcontroller and a PC or an external processing unit**.**

Basic Operation (If Used for Object Detection)

1. **Microcontroller Initialization**
    - AT89C51 initializes its LCD and serial communication (UART**)**.
    - It waits for input from an external system (e.g., object detection module).
2. **Receiving Object Detection Data**
    - If an external AI-based processor (e.g., Raspberry Pi or ESP32-CAM) detects an object, it sends a signal to AT89C51.
    - This signal can be a simple digital HIGH/LOW **or a** UART message
3. **Displaying on LCD**
    - When the AT89C51 receives detection input, it updates the LCD with messages like:
4. **Sending Data to Virtual Terminal**
    - If a virtual terminal is used, the AT89C51 can send logs through serial communication.
    - Example output on virtual terminal:

Limitations of AT89C51 for Real-Time Video Streaming

- The AT89C51 microcontroller does not support video processing or streaming.
- It lacks a camera interface, video buffer, and AI processing capability.
- For real-time video streaming with object detection, you need a Raspberry Pi, ESP32-CAM, or Jetson Nano.
- The AT89C51 can only serve as a basic alert system for displaying messages based on object detection done by another processor.

Code:

videoTransmission.c

```c
#include <reg51.h>
#include <string.h>

// LCD Pins
sbit RS = P2^0;
sbit RW = P2^1;
sbit EN = P2^2;
sbit D4 = P2^4;
sbit D5 = P2^5;
sbit D6 = P2^6;
sbit D7 = P2^7;

    char data1;


// Function Prototypes
void delay(unsigned int);
void lcd_command(unsigned char);
void lcd_data(unsigned char);
void lcd_init();
void lcd_string(char*);
char bluetooth_receive(void);

// UART Initialization for HC-05
void UART_Init() {
    TMOD = 0x20;  // Timer1 Mode2 (Auto-Reload)
    TH1 = 0xFD;   // Baud Rate 9600 (11.0592 MHz Crystal)
    SCON = 0x50;  // 8-bit UART Mode, Enable Receiver
    TR1 = 1;      // Start Timer1
}

```

videoTransmission.c

```
31
32 void UART_Tx(char ch) {
33     SBUF = ch;     // Load data into UART buffer
34     while (!TI);   // Wait for transmission to complete
35     TI = 0;        // Clear transmit flag
36 }
37
38 char UART_Rx() {
39     while (!RI);   // Wait until data is received
40     RI = 0;        // Clear receive flag
41     return SBUF;   // Return received character
42 }
43
44 // Function to receive character from Bluetooth
45 char bluetooth_receive(void) {
46     while (!RI);   // Wait for reception
47     RI = 0;        // Clear flag
48     return SBUF;   // Return received data
49 }
50
51 // LCD Initialization
52 void lcd_init() {
53     lcd_command(0x02); // 4-bit mode
54     lcd_command(0x28); // 2-line, 5x7 matrix
55     lcd_command(0x0C); // Display ON, Cursor OFF
56     lcd_command(0x06); // Auto-increment cursor
57     lcd_command(0x01); // Clear display
58 }
59
60 // Send Command to LCD
61 void lcd_command(unsigned char cmd) {
62     RS = 0;
```

videoTransmission.c

```
61 void lcd_command(unsigned char cmd) {
62     RS = 0;
63     RW = 0;
64     D4 = (cmd & 0x10) >> 4;
65     D5 = (cmd & 0x20) >> 5;
66     D6 = (cmd & 0x40) >> 6;
67     D7 = (cmd & 0x80) >> 7;
68     EN = 1;
69     delay(2);
70     EN = 0;
71
72     D4 = (cmd & 0x01);
73     D5 = (cmd & 0x02) >> 1;
74     D6 = (cmd & 0x04) >> 2;
75     D7 = (cmd & 0x08) >> 3;
76     EN = 1;
77     delay(2);
78     EN = 0;
79 }
80
81 // Send Data to LCD
82 void lcd_data(unsigned char datal) {
83     RS = 1;
84     RW = 0;
85     D4 = (datal & 0x10) >> 4;
86     D5 = (datal & 0x20) >> 5;
87     D6 = (datal & 0x40) >> 6;
88     D7 = (datal & 0x80) >> 7;
89     EN = 1;
90     delay(2);
91     EN = 0;
92
```

videoTransmission.c

```
93          D4 = (data1 & 0x01);
94          D5 = (data1 & 0x02) >> 1;
95          D6 = (data1 & 0x04) >> 2;
96          D7 = (data1 & 0x08) >> 3;
97          EN = 1;
98          delay(2);
99          EN = 0;
100     }
101
102     // Send String to LCD
103     void lcd_string(char *str) {
104         while (*str) {
105             lcd_data(*str++);
106         }
107     }
108
109     // Delay Function
110     void delay(unsigned int ms) {
111         unsigned int i, j;
112         for (i = 0; i < ms; i++)
113             for (j = 0; j < 1275; j++);
114     }
115
```

```
116     // Main Function
117     void main() {
118         char received_char;
119         lcd_init();
120         UART_Init();
121         lcd_string("Bluetooth Ready");
122         while (1) {
123             data1 = UART_Rx();  // Receive character
124             lcd_command(0x01);  // Clear display
125             UART_Tx(data1);     // Echo back
126         }
127     }
128
```

Output of Circuit Diagram: