EMBEDDED SYSTEMS PROJECT REPORT

(PROJECT No.: 11)

SUBMITTED BY:-

- KESHAB PATRO (122EE0533)
- OMDAS MEENA (122EE0829)
- DEEPAK KUMAR MURMU (122EE0605)

Project no. 11:

Customer's (Client's) problem statement:

We need to acquire three analog signals for monitoring a system output. Therefore, develop an embedded system to acquire them with minimal resources.

Our objective:

To design and develop an embedded system for acquiring and displaying three analog signals—resistance, temperature, and light intensity—with minimal resource utilization. The system aims to accurately capture, process, and display the analog inputs to facilitate real-time monitoring of system outputs.

COMPONENTS USED:-

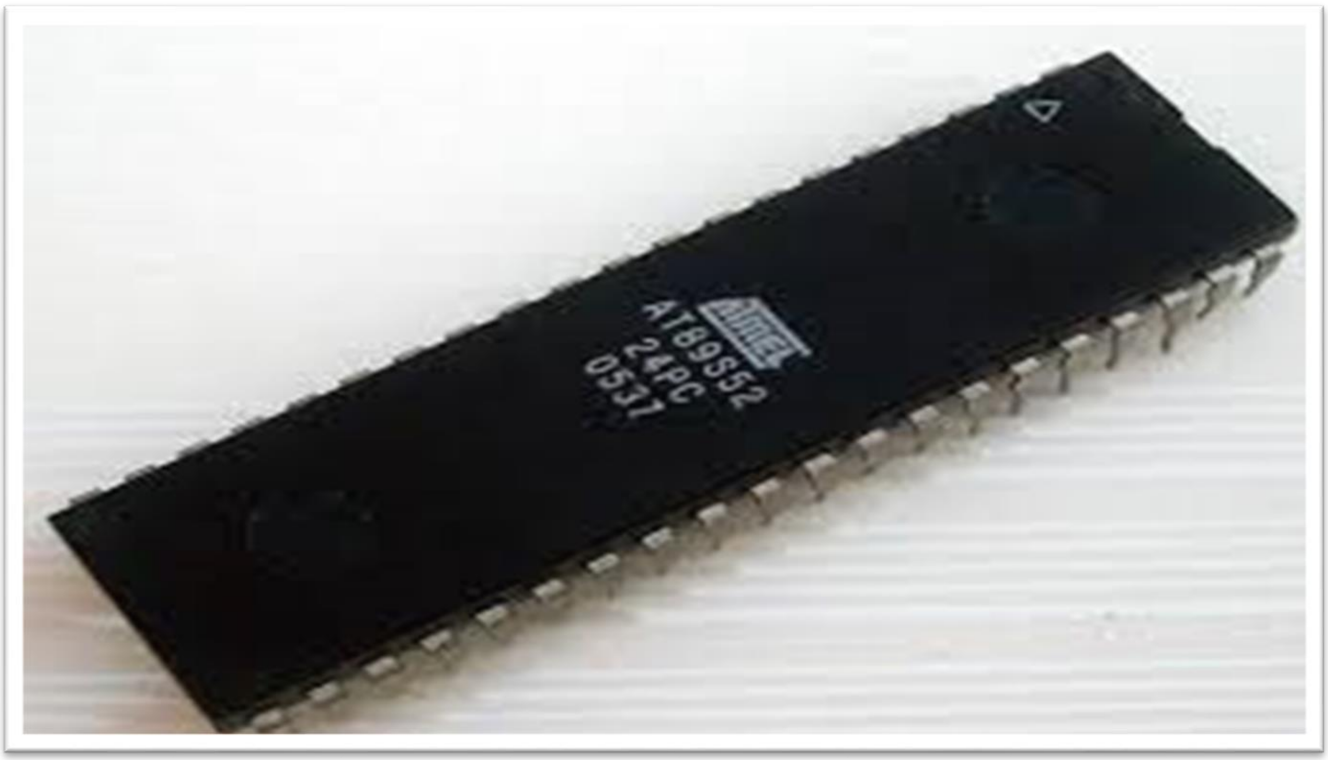
COMPONENTS	SPECIFICATIONS	QUANTITY	COST
Microcontroller	AT89S52	1	Rs 80.00
ADC0809	-	1	Rs 198.00
16 x 2 LCD Display	-	1	Rs 100.00
Potentiometer	10kohm	2	Rs 20.00
Electrolytic Capacitor	10uF, 50V	1	Rs 3.00
Crystal Oscillator	12MHz	1	Rs 8.00
LDR Sensor	-	1	Rs 10.00
LM35 sensor	-	1	Rs 85.00
Breadboard	-	2	Rs 116.00
NOT Gate	IC7404	1	Rs 10.00

Total: Rs 630.00

HARDWARE DETAILS:-

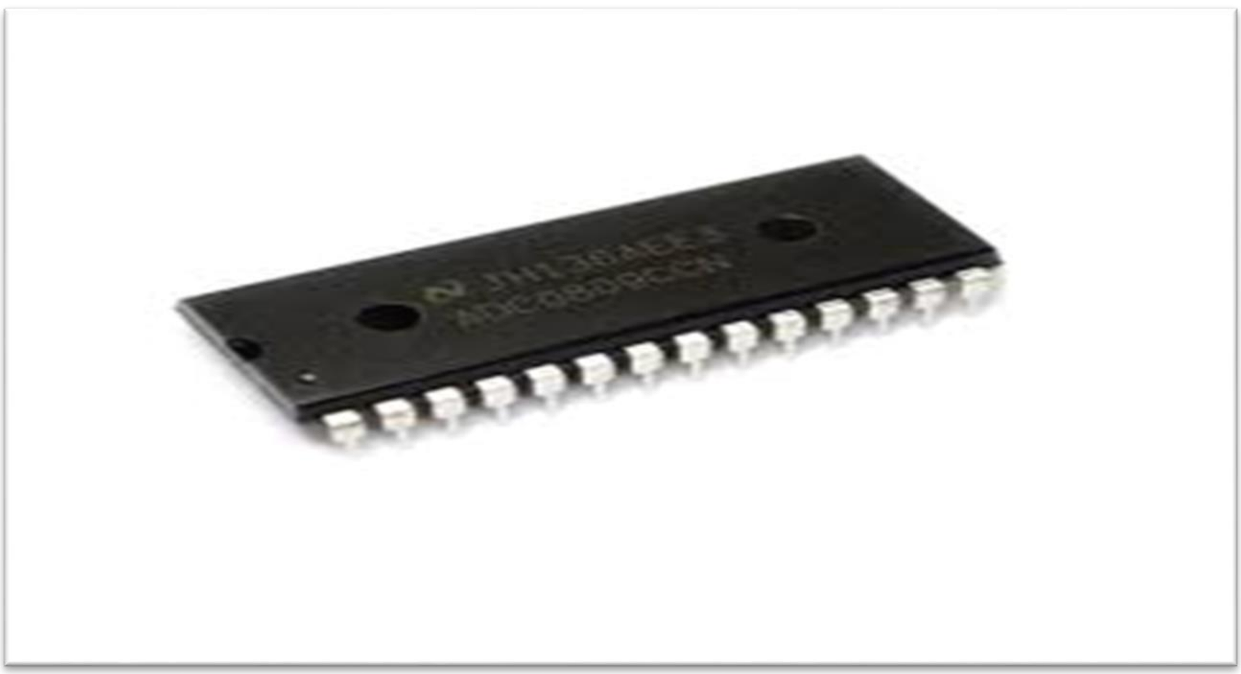
1) 8051 (AT89S52) Microcontroller:-

The AT89S52 is an 8-bit microcontroller based on the 8051 architecture, designed by Atmel. It features a high-performance, low-power CMOS design that is suitable for a wide range of embedded applications. The microcontroller includes 8 KB of in-system programmable Flash memory, 256 bytes of RAM, 2 timers and 32 general-purpose I/O lines organized into four 8-bit ports.



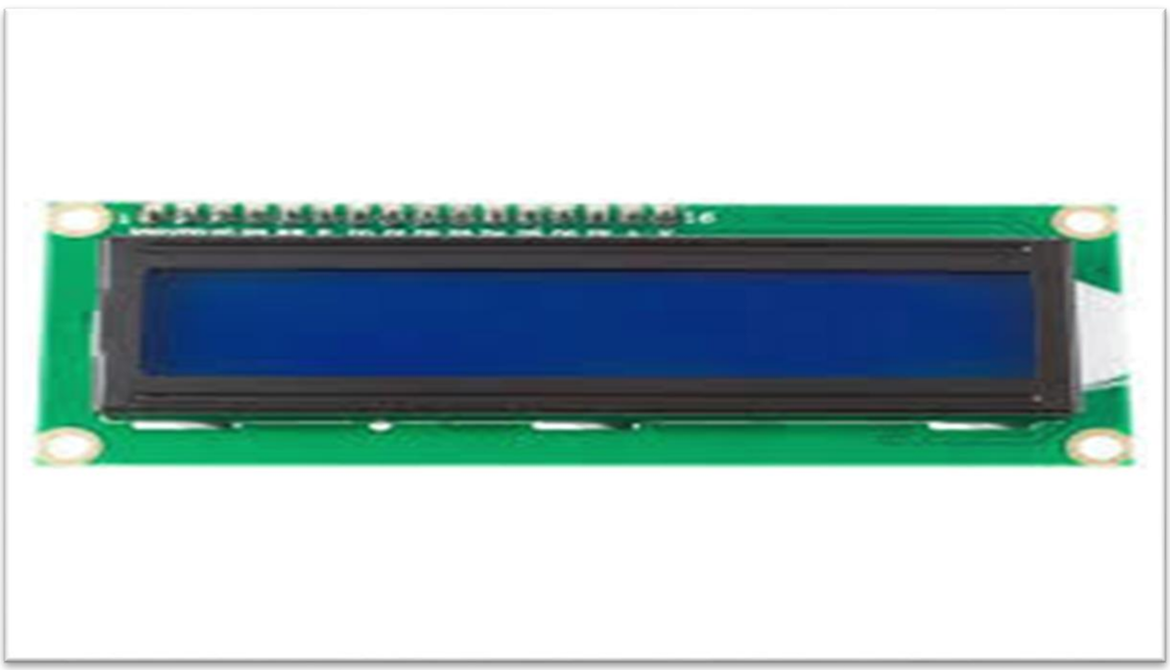
2) ADC 0809:-

The ADC0809 is an 8-bit analog-to-digital converter (ADC) with an 8-channel multiplexer, ideal for multi-source data acquisition. It uses successive approximation for fast, accurate conversion and outputs digital values from 0 to 255. The device operates with an external or built-in clock, requires a reference voltage for input range definition, and interfaces easily with microcontrollers via a tri-state output. Its versatility and efficiency make it suitable for diverse sensor interfacing applications.



3) 16 x 2 LCD DISPLAY:-

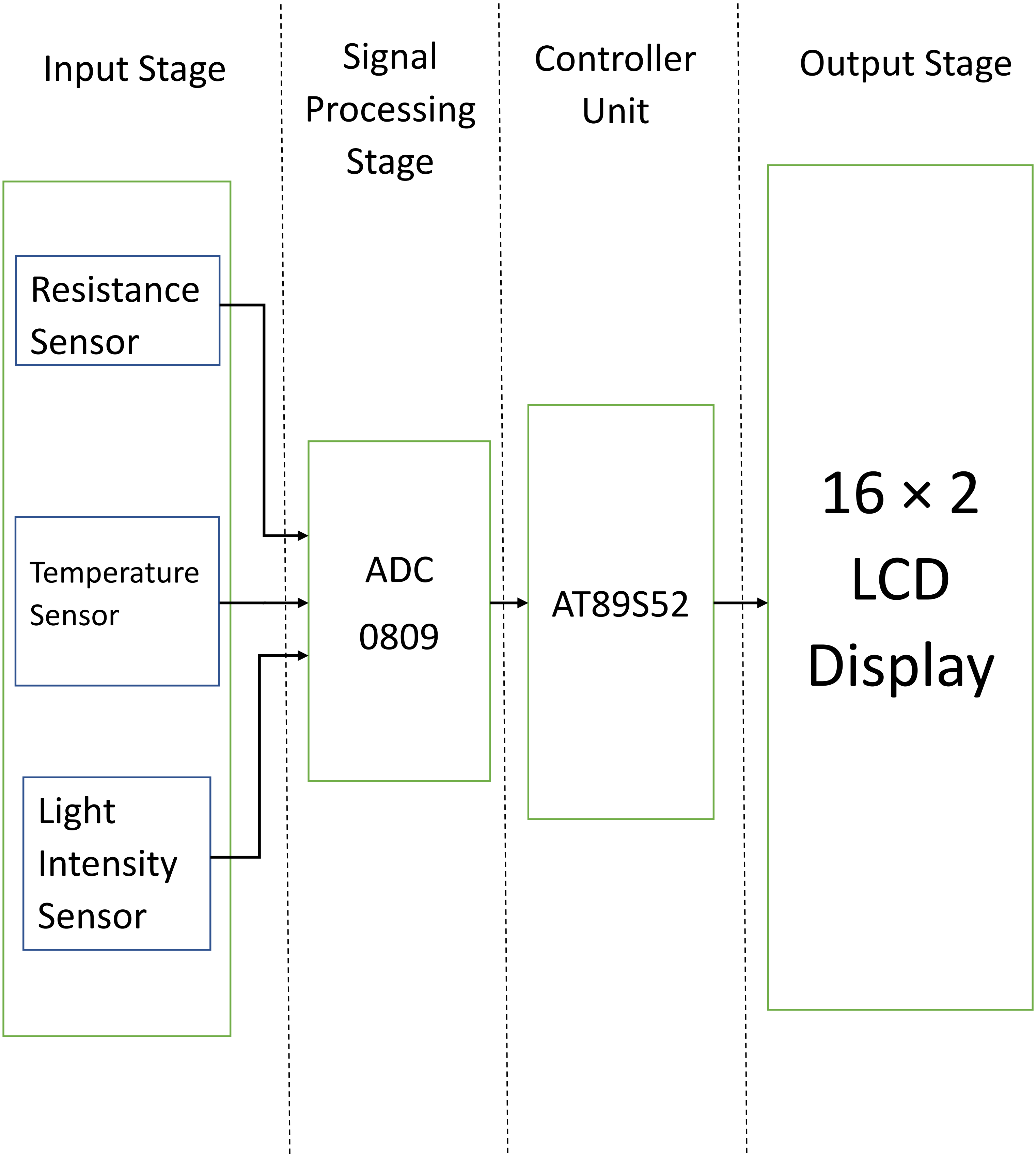
A 16x2 LCD display is a compact module capable of displaying 16 characters per line across 2 lines. It uses an HD44780-compatible controller and supports alphanumeric characters, numbers, and custom symbols. The display operates in 4-bit or 8-bit mode for data communication and includes a backlight for improved visibility. Its ease of use, low power consumption, and compatibility with microcontrollers make it ideal for embedded systems and user interfaces.



4) NOT GATE:-

The IC 7404 is a hex inverter, consisting of six independent NOT gates, each with a single input and output. It operates by inverting the input logic state—outputting a logical HIGH (1) for a LOW (0) input, and vice versa. The IC is built using TTL (Transistor-Transistor Logic) technology, ensuring fast switching and reliable performance. It typically operates at a 5V DC power supply and features a wide range of operating temperatures, making it suitable for various digital logic circuits. The 7404 is widely used in signal inversion, logic building, and digital systems.

BLOCK DIAGRAM:-



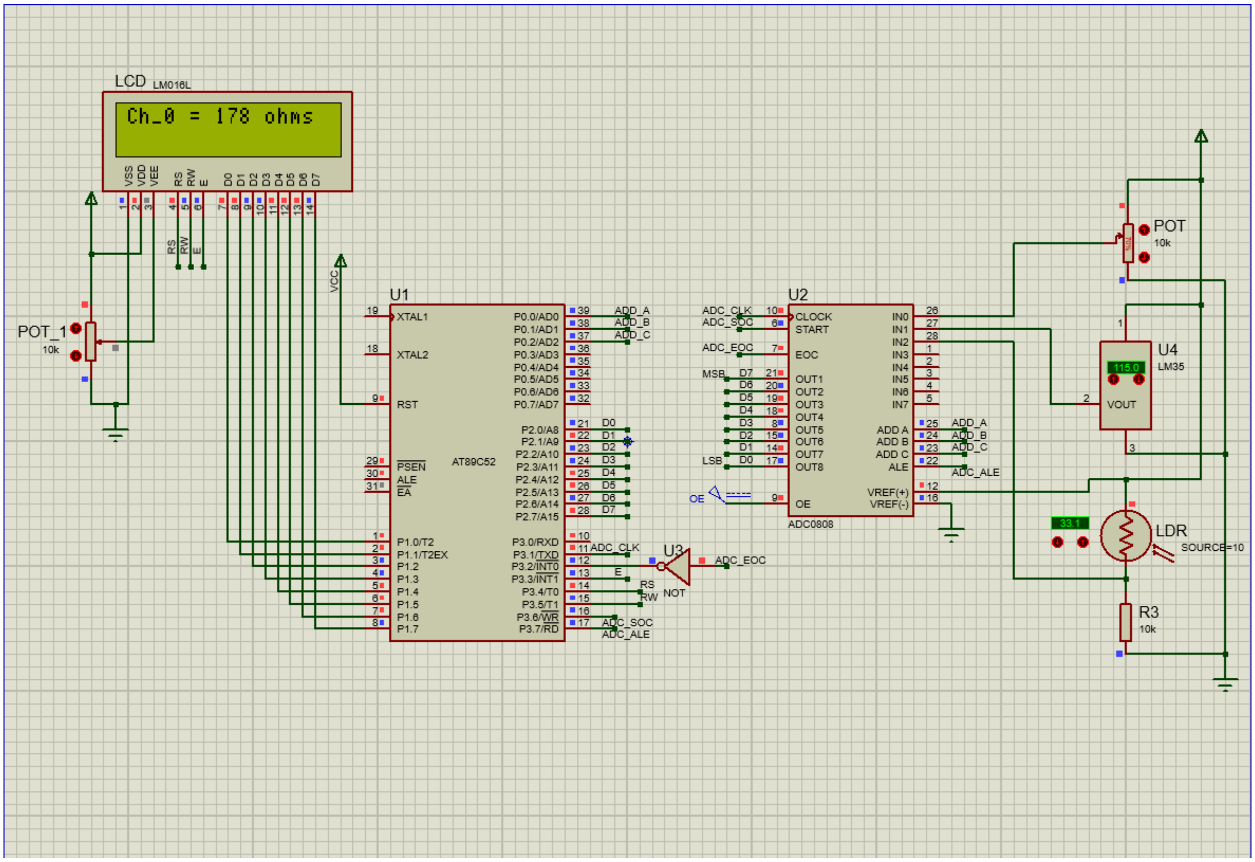
Working Principle

The developed embedded system works as follows:

1. **Analog Signal Acquisition**:
    - Three sensors are used to measure resistance, temperature, and light intensity. These sensors generate analog voltage signals corresponding to the respective physical quantities.
2. **Signal Conversion (ADC0808)**:
    - The analog signals from the sensors are fed into the ADC0808 (Analog-to-Digital Converter).
    - The ADC0808 converts the analog inputs into 8-bit digital data that the microcontroller can process.
3. **Microcontroller Processing (80C51)**:
    - The microcontroller (80C51) receives the digital signals from the ADC0808 through its data lines.
    - It processes the signals to compute the corresponding resistance, temperature, and light intensity values.
    - The microcontroller uses a pre-programmed algorithm to convert the raw ADC data into meaningful readings (e.g., Ohms, Celsius, Lux).
4. **Display Output (16×2 LCD)**:
    - The processed data is sent to the 16×2 LCD.
    - The LCD displays the sensor readings in real-time, showing the resistance, temperature, and light intensity values on separate rows or columns.

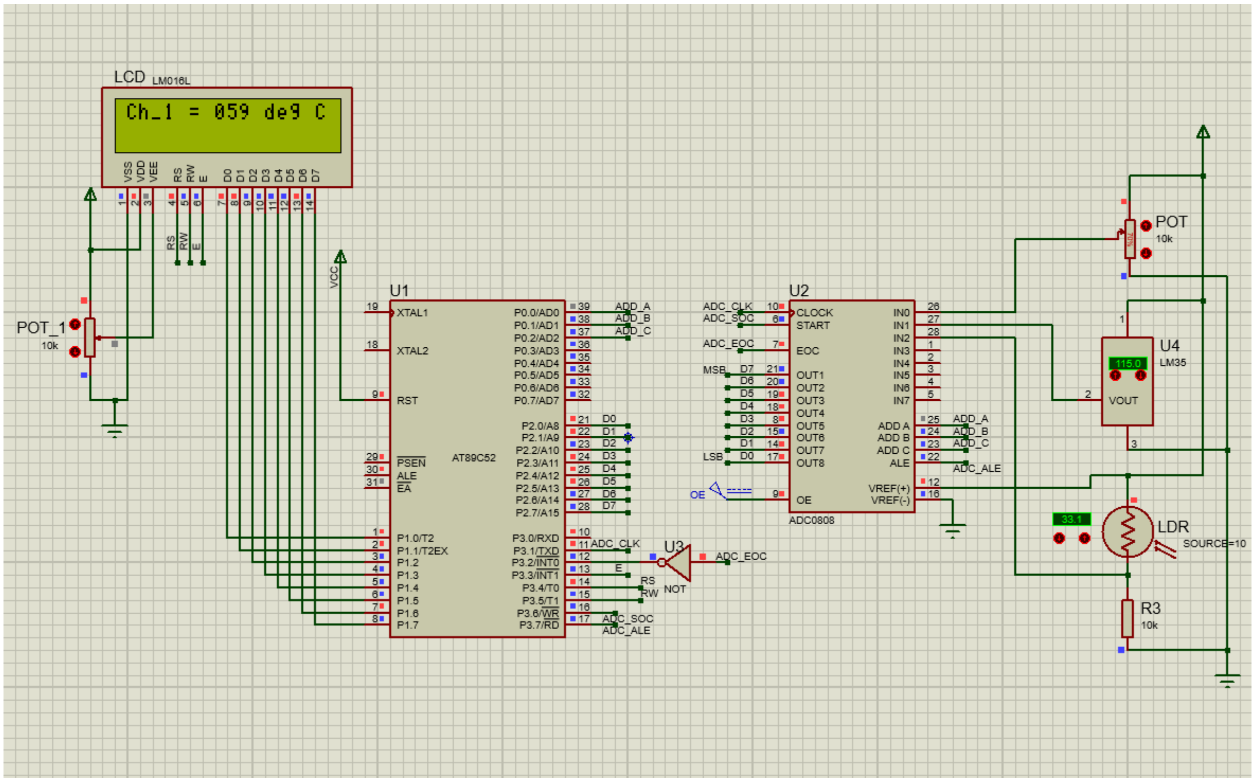
CIRCUIT DIAGRAM:-

1)



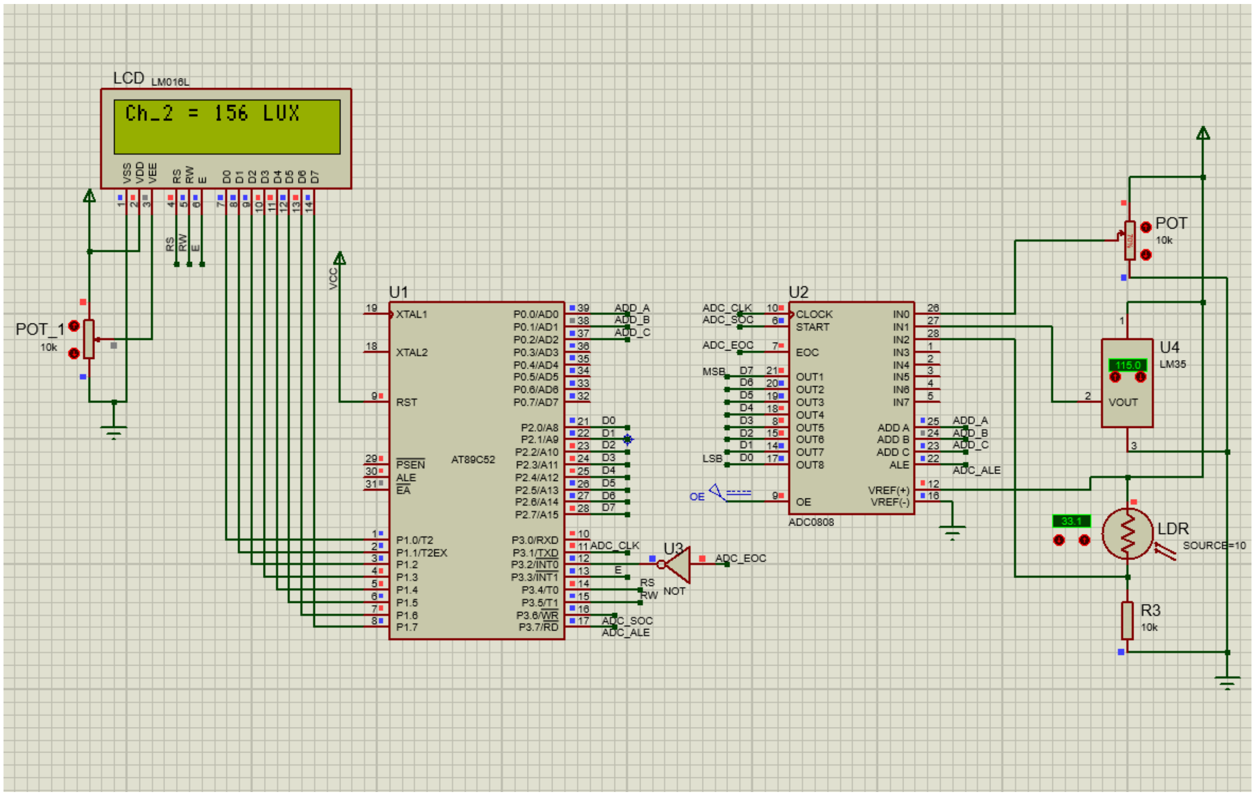
(Channel 0 for displaying value of the resistance)

2)



(Channel 1 for displaying the temperature value)

3)



(Channel 2 for displaying the value of light intensity)

CODE:-

```
$NOMOD51
$INCLUDE (80C52.MCU)

;====================================================================
; DEFINITIONS
;====================================================================
ALE  equ P3.7
SC   equ P3.6
RS   equ P3.4
RW   equ P3.5
E    equ P3.3
;====================================================================
; VARIABLES
;====================================================================
Prev_ADC equ 30h      ; Memory location for storing the previous ADC value
Channel_Index equ 31h  ; Memory location to track the current channel

;====================================================================
; RESET VECTOR
;====================================================================
org 0000h
jmp Start

org 001bh
jmp T1ISR
```

```
;====================================================================
; CODE SEGMENT
;====================================================================

Start:mov ie,#88h
      mov TMOD,#20H
      MOV TL1,#0FAH
      MOV TH1,#0FAH
      SETB TR1

      mov P2,#0FFh;
      mov P0, #00h        ; Initialize ADD_A, ADD_B, ADD_C to Channel 0
      mov Channel_Index, #00h ; Start with Channel 0
      call LCD_Init       ; Initialize LCD
      mov Prev_ADC, #00h ; Initialize previous ADC value to 0

Main_Loop:
      ; Select Channel
      mov A, Channel_Index
      anl A, #07h         ; Ensure only lower 3 bits are used
      mov P0, A           ; Set ADD_A, ADD_B, ADD_C
      setb ALE            ; Latch the address
      setb SC
      clr ALE
      clr SC

      ; Wait for the ADC to stabilize (dummy delay)
      call Delay
```

```

        ; Read ADC value
        mov A, P2          ; Read ADC value
        mov B,P2
        call Clear_LCD     ;
        call Display_Channel ; Display the channel number and value

        ; Delay for 10 seconds
        mov R2, #10        ; 10 seconds delay
Delay_Seconds:
        call Long_Delay
        djnz R2, Delay_Seconds
        mov R7, Channel_Index
        ; Increment Channel Index
        inc Channel_Index
        inc R7
        cjne R7, #03h, Main_Loop ; Loop for 3 channels
        mov Channel_Index, #00h ; Reset to Channel 0
        sjmp Main_Loop

```

```
;======================================================================
; LCD ROUTINES
;======================================================================

LCD_Init:
        mov P1, #38h        ; 8-bit mode, 2-line display
        call LCD_Command
        mov P1, #0Ch        ; Display ON, Cursor OFF
        call LCD_Command
        mov P1, #06h        ; Entry mode, auto increment
        call LCD_Command
        ret

LCD_Command:
        clr RS              ; Command mode
        clr RW              ; Write mode
        setb E              ; High to Low pulse
        clr E
        acall Delay
        ret

LCD_Data:
        setb RS             ; Data mode
        clr RW              ; Write mode
        setb E              ; High to Low pulse
        clr E
        acall Delay
        ret

Clear_LCD:
        mov P1, #01h        ; LCD clear command
        call LCD_Command
        ret
```

```
Display_Channel:
        ; Select message based on Channel_Index
        mov A, Channel_Index
        mov DPTR, #Voltage1
        cjne A, #01h, Check_Channel2
        mov DPTR, #Voltage2
        sjmp Display_Message
Check_Channel2:
        cjne A, #02h, Display_Message
        mov DPTR, #Voltage3

Display_Message:
        call Display_String ; Display the channel message
        call Display_Value  ; Display the ADC value
        mov A, Channel_Index
        mov DPTR, #m1; ohms
        cjne A, #01h, Check_Channel2_2
        mov DPTR, #m2; deg C
        sjmp Display_String
Check_Channel2_2:
        cjne A, #02h, Display_String
        mov DPTR, #m3; LUX
        sjmp Display_String
        ret

Display_String:
        clr A
        movc A, @A+DPTR     ; Get character from code memory
        jz End_String       ; End of string (null terminator)
        mov P1, A           ; Send character to LCD
        call LCD_Data
        inc DPTR            ; Move to next character
        sjmp Display_String
```

```
End_String:
        ret

Display_Value:
        mov A, B            ; Convert to ASCII for higher nibble
        mov B, #100         ; Load divisor for hundreds place
        div AB              ; A = quotient (hundreds), B = remainder
        add A, #30h         ; Convert hundreds to ASCII
        mov P1, A
        call LCD_Data       ; Display hundreds place
        mov A, B            ; Load remainder into A

        mov B, #10          ; Load divisor for tens place
        div AB              ; A = quotient (tens), B = remainder
        add A, #30h         ; Convert tens to ASCII
        mov P1, A
        call LCD_Data       ; Display tens place

        mov A, B            ; Units place (remainder)
        add A, #30h         ; Convert units to ASCII
        mov P1, A
        call LCD_Data       ; Display units place
        ret


T1ISR:
        CPL P3.1
        RETI
```

```
;===================================================================
; DELAY ROUTINES
;===================================================================
Delay:
      mov R0, #100
Delay_Loop1:
      mov R1, #100
Delay_Loop2:
      djnz R1, Delay_Loop2
      djnz R0, Delay_Loop1
      ret

Long_Delay:
      mov R0, #255       ; Adjust this value for 1-second delay
Delay_1s_Loop1:
      mov R1, #255
Delay_1s_Loop2:
      djnz R1, Delay_1s_Loop2
      djnz R0, Delay_1s_Loop1
      ret

;===================================================================
; CONSTANTS
;===================================================================
Voltage1: db 'Ch_0 = ', 0;
Voltage2: db 'Ch_1 = ', 0;
Voltage3: db 'Ch_2 = ', 0;
m1: db ' ohms',0;
m2: db ' deg C',0;
m3: db ' LUX',0;
;===================================================================
END
```

Applications

The developed embedded system for acquiring and displaying resistance, temperature, and light intensity can be used in various real-world applications, including:

1. **Industrial Monitoring**
    - Monitoring system parameters like light intensity in workspaces, resistance in electrical components, and temperature in machinery to ensure safety and efficiency.
2. **Environmental Sensing**
    - Measuring ambient temperature and light intensity for weather stations or greenhouses.
    - Monitoring soil resistance (indirectly measuring soil moisture) in agriculture.
3. **Home Automation Systems**
    - Controlling lighting systems based on ambient light levels.
    - Monitoring room temperature for smart thermostats.
4. **Laboratory Experiments**
    - Providing students and researchers with a real-time monitoring system for educational purposes and testing.
5. **IoT Integration**
    - Acting as a sensor module for IoT-based monitoring systems, sending data remotely for smart applications.
6. **Consumer Electronics Testing**
    - Testing light intensity (e.g., in displays), electrical resistance, or temperature in appliances during quality control.
7. **Renewable Energy Systems**
    - Measuring solar irradiance (light intensity) for optimizing solar panel placement.
    - Monitoring temperature of panels or batteries to ensure safety and efficiency.

Conclusion:

The project successfully fulfills the client's requirement by developing an embedded system capable of acquiring and displaying three analog signals: resistance, temperature, and light intensity. Using minimal resources, the system demonstrates efficient data acquisition, processing, and real-time monitoring, showcasing its practicality and effectiveness. This solution is scalable and can be further enhanced to include additional parameters or functionalities as needed.