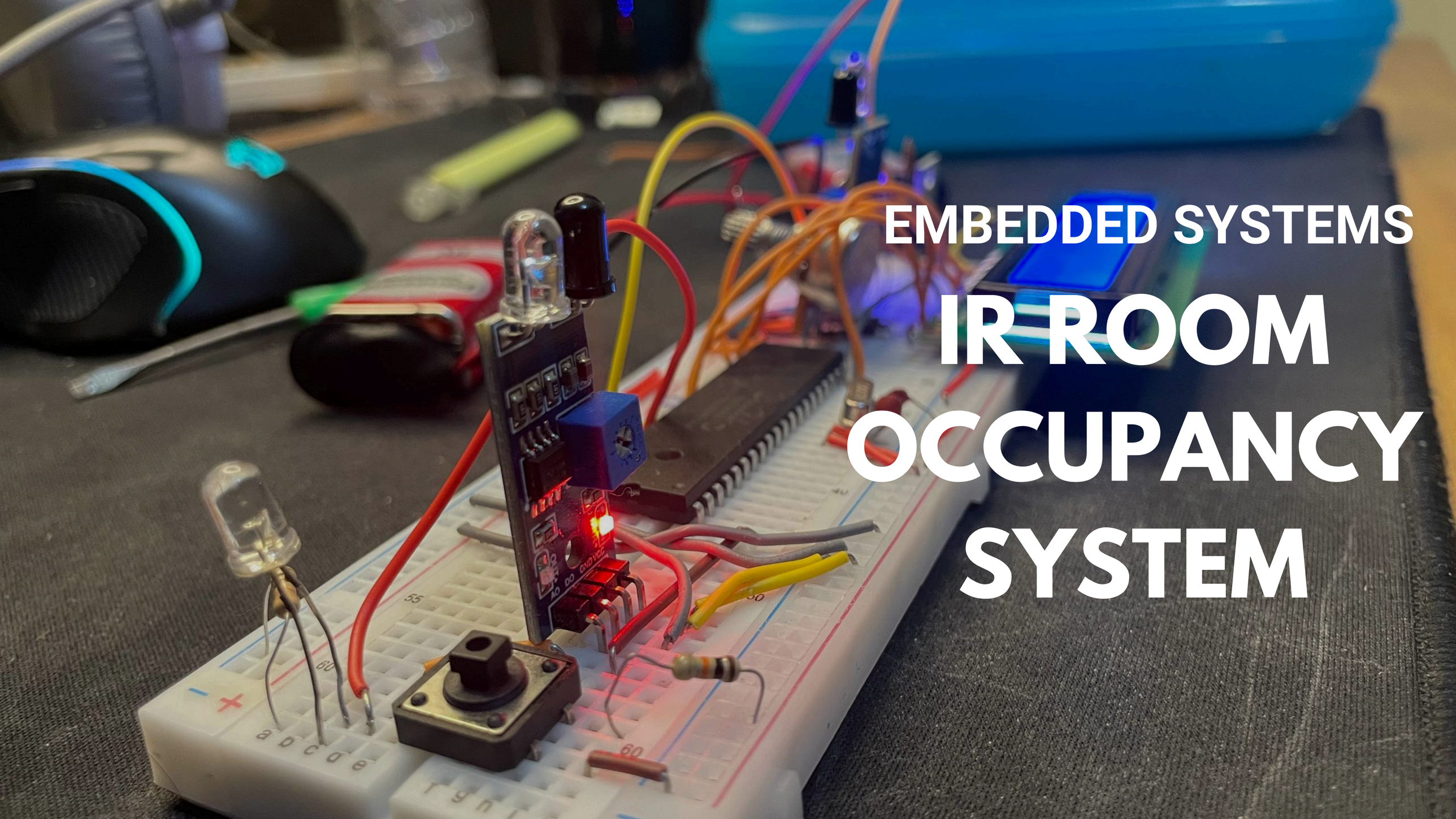

EMBEDDED SYSTEMS
IR ROOM OCCUPANCY SYSTEM

# TEAM 5 MEMBERS

1. احمد محمد توفيق رجب
2. محمود فرج حسين المكبر
3. نور الدين محمد يحيي
4. عمر جمال البرماوي
5. ليديا نبيل رزق عطاالله
6. لبني مصطفي طه جاد الحق
7. يمني عبدالصمد جمال زيدان
8. يارا احمد محمد عبدالعال

# PROJECT EXPLANATION

## Smart Room People Counter

This project is an IR Room Occupancy Counter Controller built with Mikro C

It uses infrared (IR) sensors to detect room entry and exit events

Counts the number of people present

Displays real-time occupancy status on an LCD

it controls a light (LED) based on occupancy

# PROJECT EXPLANATION

## FEATURES

- Entry/Exit Detection: Uses two IR sensors to monitor both entry and exit points.
- Real-Time Occupancy Counter: Counts number of people present in the room.
- LCD Display: Shows current count, room status (empty/full/occupied)
- Maximum Capacity Control: Notifies when the room reaches full capacity.
- Timeout Management: Resets detection state if sensors remain active for too long.
- Automatic Lighting: Turns on a LED when room is occupied.

# PROJECT EXPLANATION

## HOW IT WORKS

- The system uses a simple state machine to interpret sensor changes:
  - Idle: Waits for sensor input.
  - Entry Detected: Marks the beginning of an entry sequence.
  - Exit Detected: Marks the beginning of an exit sequence.
  - Increments or decrements the counter based on completed sequences.
- The count is displayed on the LCD, which updates whenever occupancy changes.
- The LED is controlled by the occupancy count (Counter > 0 => light ON).
- Maximum room capacity can be set (default is 10).

# PROJECT EXPLANATION

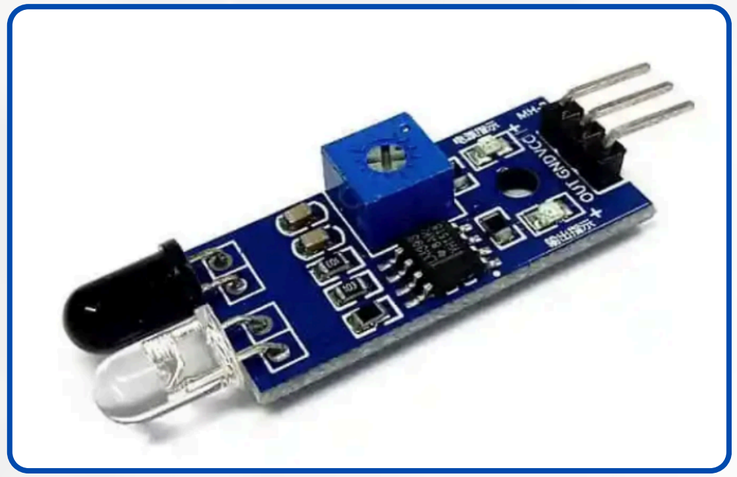
## Core System Structure

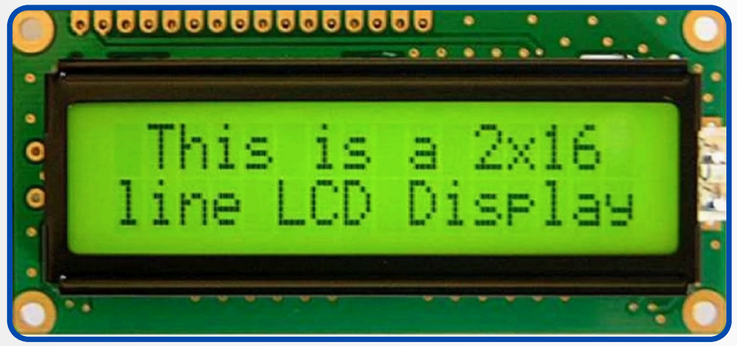
- Sensors: Connected to PORTB (input pins).
- LCD: Connected to PORTD.
- LED: Output pin on PORTB.
- Timer0: Used for timeout handling to improve sensor accuracy.

## Code Highlights

- Entry and exit events are debounced and managed in interrupts.
- LCD updates and lighting decisions are made based on global counter value.
- User can adjust maximum capacity in code.

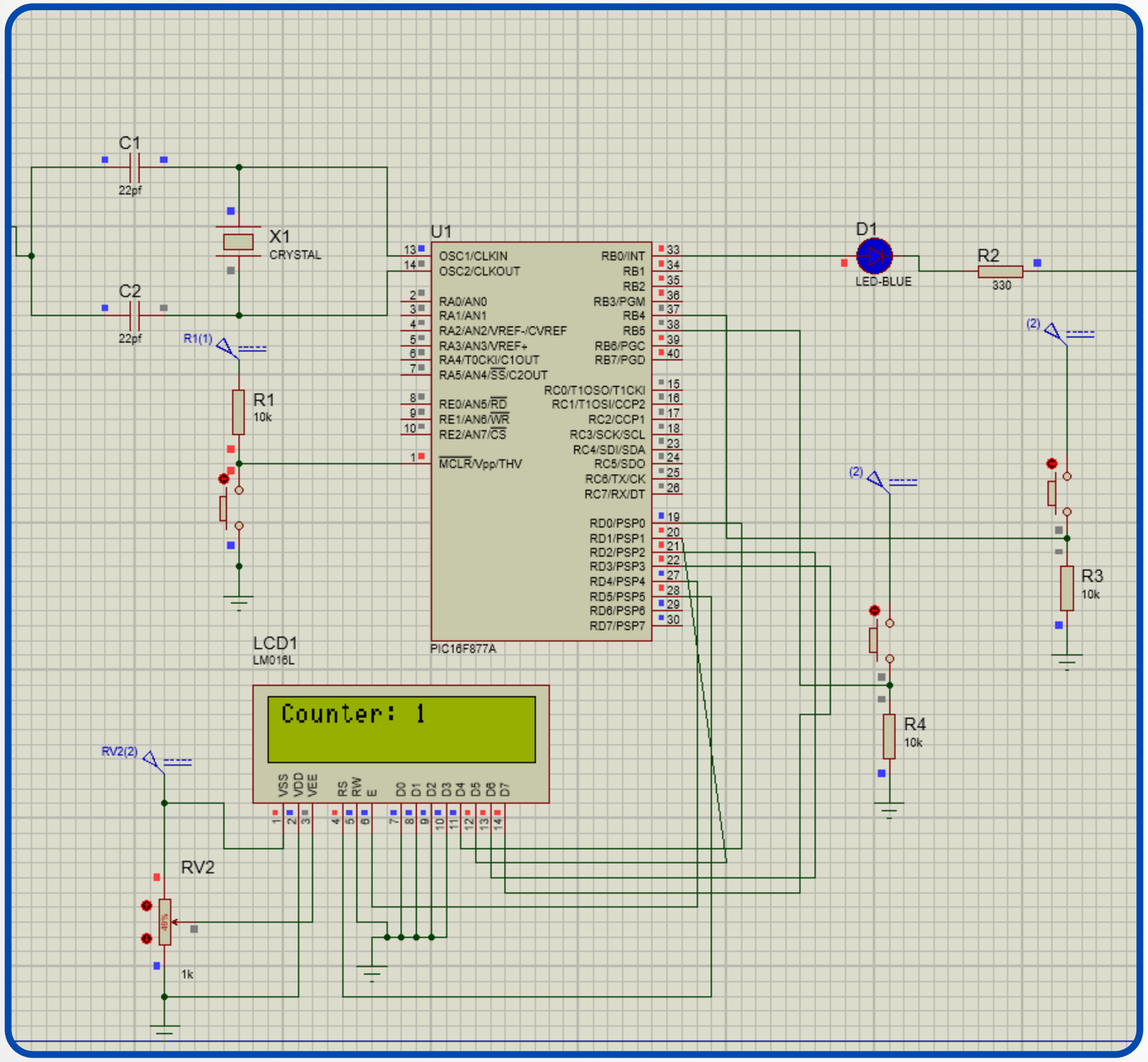
# COMPONENTS






1 - 1x PIC16F877A
2 - 2x Infrared Sensors
3 - 1x 2x16 bit LCD
4 - 2x 9V Battery & 2x 9V Battery Snap
5 - 1x LM7805 Voltage Regulator
6 - 1x 8MHz Crystal
7 - 2x 22pF Capacitors
8- 2x  100nF Capacitors
9 - 1x Reset Button
10 - 2x 330Ω & 1x 10KΩ Resistors
11 - 2x Blue LEDS
12 - 1x Bread Boards & Wire Kit

# CIRCUIT DESIGN




- **Sensors: Connected to PORTB (RB4 & RB5).**
- **LCD: Connected to PORTD.**
- **LED: Output pin on RB0.**

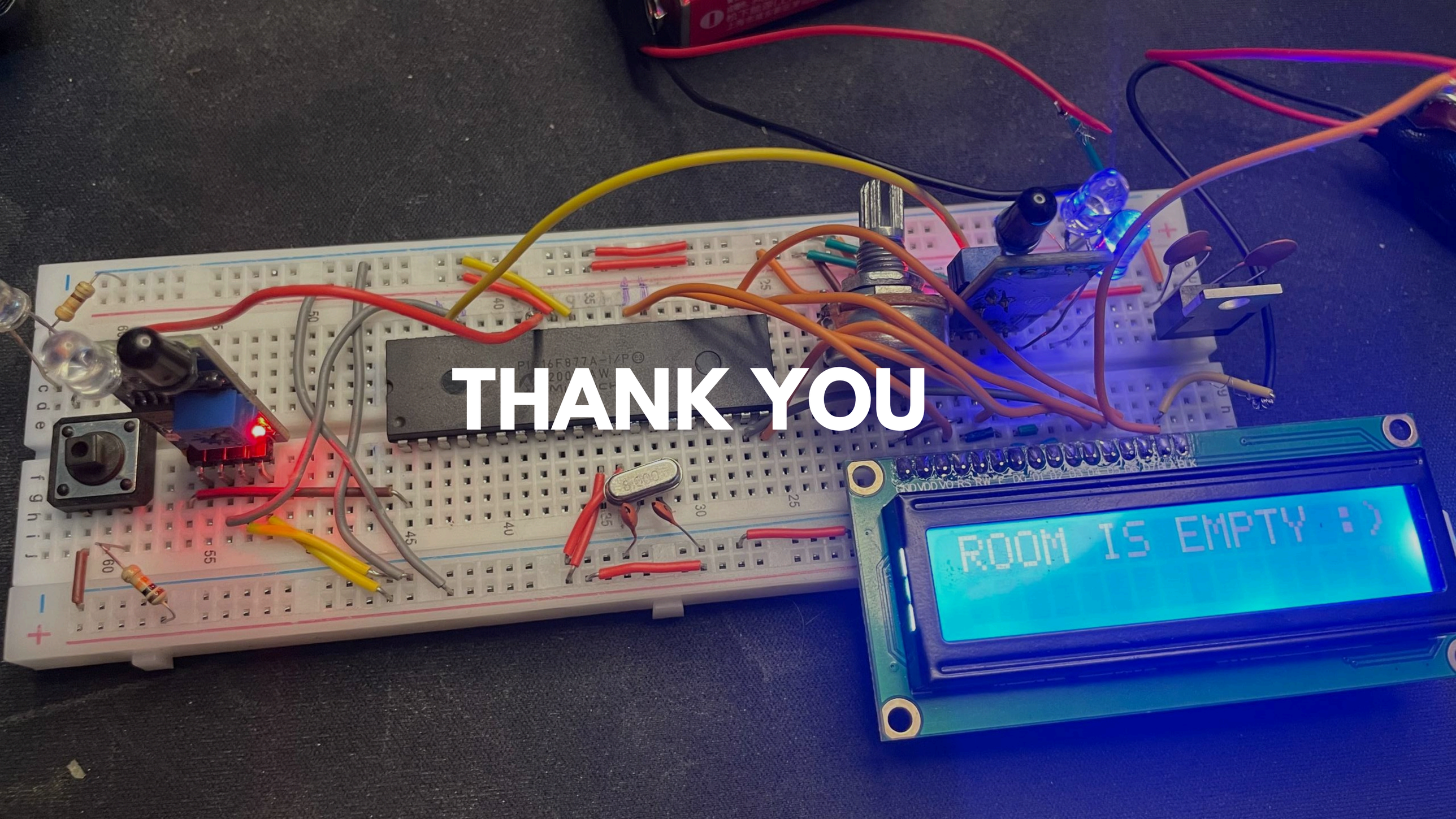
```
// Sensor pins definition
sbit SensorIN at PORTB.B4;
sbit SensorOUT at PORTB.B5;

// LCD data pins definition
sbit LCD_D4 at RD0_BIT;
sbit LCD_D5 at RD1_BIT;
sbit LCD_D6 at RD2_BIT;
sbit LCD_D7 at RD3_BIT;
sbit LCD_EN at RD4_BIT;
sbit LCD_RS at RD5_BIT;

// LCD data direction pins definition
sbit LCD_D4_Direction at TRISD0_BIT;
sbit LCD_D5_Direction at TRISD1_BIT;
sbit LCD_D6_Direction at TRISD2_BIT;
sbit LCD_D7_Direction at TRISD3_BIT;
sbit LCD_EN_Direction at TRISD4_BIT;
sbit LCD_RS_Direction at TRISD5_BIT;

// Global variables
volatile int Counter = 0;                    // Current count of people in room
int LastCounterState = -1;                   // Previous counter value for LCD update detection
int MaxCapacity = 10;                        // Maximum room capacity
volatile int State = 0;                     // State machine variable (0: idle, 1: entry detected, 2: exit detected)
int OverFlows = 0;                          // Maximum number of timer overflows for timeout
volatile int OverFlows_counter = 0;          // Current overflow counter
char Str[7];                                 // String buffer for integer to string conversion
```

```
// Function prototypes
void TimeOutHandler();
void DisplayState();
void ClearLcdBasedOnCounter();
void PICConfig();
void LcdConfig();
void LightControl();
double Time0OverFlows(int time);


void main() {

    PICConfig();

    LcdConfig();

    // Calculate timer overflows for 1 second timeout
    OverFlows = Time0OverFlows(1);

    // Main loop
    while(1) {

        // Convert counter to string
        inttostr(Counter,Str);
        ltrim(Str);

        // Update LCD when counter changes
        ClearLcdBasedOnCounter();

        // Display current state on LCD
        DisplayState();

        // (ON if room occupied)
        LightControl();

        // Reset State if stuck
        TimeOutHandler();
   }
}
```

```
void interrupt() {
    // Sensor triggered
    if (INTCON.RBIF == 1) {
        delay_ms(80);  // Debounce delay

        switch(State) {
            case 0:  // Idle state
                if (SensorIN == 1) {
                    State = 1;  // Entry sequence started
                    OverFlows_counter = 0;
                }
                if (SensorOUT == 1) {
                    State = 2;  // Exit sequence started
                    OverFlows_counter = 0;
                }
                break;

            case 1:  // Entry sequence in progress
                if (SensorOUT == 1) {
                    Counter++;  // Person entered
                    State = 0;
                }
                break;

            case 2:  // Exit sequence in progress
                if (SensorIN == 1) {
                    if(Counter != 0)
                        Counter--;  // Person exited
                    State = 0;
                }
                break;
        }

        INTCON.RBIF = 0;
    }

    // Timeout handling
    if(INTCON.TMR0IF == 1) {
        OverFlows_counter++;
        INTCON.TMR0IF = 0;
        TMR0 = 0;
    }
}
```

```
void PICConfig()
{
     // Configure PORTB: RB0 as output (LED), RB4-RB5 as inputs (sensors)
    TRISB = 0b11111110;

    // Configure PORTD as output for LCD
    TRISD = 0;

    // Enable interrupts
    INTCON.RBIE = 1;        // Enable PORTB change interrupt
    INTCON.GIE = 1;         // Enable global interrupts
    INTCON.TMR0IE = 1;      // Enable Timer0 overflow interrupt

    // Configure Timer0: Prescaler 1:256
    OPTION_REG = 0b00000111;

    PORTB = 0;
    PORTD = 0;

    TMR0 = 0;
}

void LcdConfig()
{
     // Initialize LCD module
     LCD_INIT();
     LCD_CMD(_LCD_CLEAR);
     LCD_CMD(_LCD_CURSOR_OFF);
}

// Calculate number of Timer0 overflows for given time in seconds
// Timer0 with 256 prescaler overflows every 0.032768 seconds (assuming 8MHz clock)
double Time0OverFlows(int time) {
       double overflowtime = 0.032768;  // Time per overflow in seconds
       double result = ceil(time / overflowtime);
       return result;
}
```

```
// Reset state machine if timeout occurs
void TimeOutHandler() {
    if(State != 0) {
        if(OverFlows <= OverFlows_counter) {
            OverFlows_counter = 0;
            State = 0;
        }
    }
}

// Display current room status on LCD
void DisplayState() {
    if(Counter == 0) {
        LCD_OUT(1,1,"ROOM IS EMPTY :)");
    }
    else if(Counter >= MaxCapacity) {
        LCD_OUT(1,1,"ROOM IS FULL :( ");
    }
    else {
        LCD_OUT(1,1,"Counter: ");
        LCD_OUT_CP(Str);
    }
}

//Control LED based on counter
void LightControl()
{
     if(Counter > 0)
         PORTB.B0 = 1;
     else
         PORTB.B0 = 0;
}

// Clear LCD screen when counter value changes
void ClearLcdBasedOnCounter() {
    if(Counter != LastCounterState) {
        LCD_CMD(_LCD_CLEAR);
        LastCounterState = Counter;
    }
}
```


THANK YOU
ROOM IS EMPTY :)