

**YUSHEEL HARINARAIN**

**221005090**

**COMPUTER ENGINEERING DESIGN 2**

**ENEL3CB**

**PROJECT 5 – AIR QUALITY SENSOR**

**2024**

# Abstract

This report details the design, development, and implementation of a low-cost air quality monitoring system that uses the PIC16F690 microcontroller and the MQ-135 air quality sensor. The system is capable of detecting harmful gases such as carbon dioxide (CO2), ammonia, benzene, and other volatile organic compounds (VOCs), which are prevalent in indoor environments. By monitoring pollution levels in real time, the system provides immediate feedback to the user via a 16x2 LCD display, showing air quality in parts per million (PPM). The microcontroller processes the sensor’s output through its analog-to-digital converter (ADC) and compares the air quality readings to predefined thresholds. If pollution levels exceed 250 PPM, a buzzer is triggered to alert the user, indicating poor air quality that could lead to health risks such as headaches, dizziness, or respiratory issues.

The system is powered by a 5V supply and employs a simple yet effective algorithm to monitor and respond to air quality changes. The integration of a buzzer and a user-friendly display ensures that the system can be easily used in various indoor environments such as homes, offices, and classrooms. The project demonstrates how low-cost components can be combined to create a reliable air quality monitoring solution, offering significant potential for further development. Future enhancements may include adding wireless connectivity for remote monitoring or integrating mobile app support for real-time alerts and data tracking. This system provides a foundation for improving indoor air quality monitoring while keeping the overall cost low and accessible to a wide range of users.

Contents

[Abstract 2](#_Toc180779490)

[Introduction 5](#_Toc180779491)

[Problem Statement & Specifications 5](#_Toc180779492)

[Research 6](#_Toc180779493)

[Air Quality Monitoring 6](#_Toc180779494)

[MQ-135 Sensor 6](#_Toc180779495)

[PIC16F690 Microcontroller 6](#_Toc180779496)

[Hardware Design 7](#_Toc180779497)

[4.1 Component Overview 7](#_Toc180779498)

[4.2 Circuit Design & Implementation 7](#_Toc180779499)

[5.1 Software Design 7](#_Toc180779500)

[5.1.1 Main Program 7](#_Toc180779501)

[5.1.2 Key Subroutines 8](#_Toc180779502)

[InitializeLCD: 8](#_Toc180779503)

[ADC Conversion: 8](#_Toc180779504)

[ConvertADCtoPPM: 8](#_Toc180779505)

[BuzzerOn and BuzzerOff: 8](#_Toc180779506)

[LCDWriteMain: 8](#_Toc180779507)

[LCDWriteBad: 8](#_Toc180779508)

[**LCDWriteGood**: 9](#_Toc180779509)

[Flow Diagram and Pseudo Code 9](#_Toc180779510)

[5.2.1 Pseudo Code 9](#_Toc180779511)

[5.2.2 Flow Diagram 10](#_Toc180779512)

[6. Testing & Results 10](#_Toc180779514)

[Sample Test Cases: 10](#_Toc180779515)

[**Final Testing** 12](#_Toc180779516)

[**Code Performance Figures** 12](#_Toc180779518)

[**Time Taken** 12](#_Toc180779526)

[7. Conclusion 13](#_Toc180779533)

[References 14](#_Toc180779534)

[Appendix 15](#_Toc180779535)

[Code used 15](#_Toc180779536)

# Introduction

This project focuses on creating a cost-effective system for measuring indoor air quality using the PIC16F690 microcontroller and the MQ-135 sensor. The goal was to monitor pollution levels and provide real-time feedback via an LCD display. The system triggers an alert when air quality deteriorates, allowing users to take immediate action to improve the environment. The device can detect gases such as CO2 and VOCs, contributing to a better understanding of indoor pollution.

# Problem Statement & Specifications

Indoor air pollution can have significant health impacts, leading to respiratory issues, headaches, and other ailments. The objective of this project was to develop a system that can measure the concentration of gases and display the results in parts per million (PPM). The system also needed to alert the user when pollution levels exceeded a safe threshold of 250 PPM.

Key specifications:

* Real-time display of air quality in PPM on a 16x2 LCD.
* Trigger a buzzer when PPM exceeds 250.
* Use of a PIC16F690 microcontroller to manage sensor data and output.
* The MQ-135 sensor detects a range of gases including CO2, ammonia, and benzene.

# Research

## Air Quality Monitoring

Air quality monitoring is essential for maintaining safe indoor environments, as pollutants like carbon dioxide, ammonia, and volatile organic compounds (VOCs) can build up over time and affect health. Prolonged exposure to high levels of these gases can lead to symptoms such as headaches, drowsiness, and more severe respiratory issues.

## MQ-135 Sensor

The MQ-135 is a popular gas sensor used for detecting air pollutants such as CO2, ammonia, and benzene. It outputs an analog voltage that is proportional to the concentration of these gases. This voltage can be converted to PPM values to represent the air quality levels accurately. The sensor is typically powered by a 5V supply and is suitable for indoor air quality monitoring due to its sensitivity to a wide range of harmful gases. (Anand Kanti)

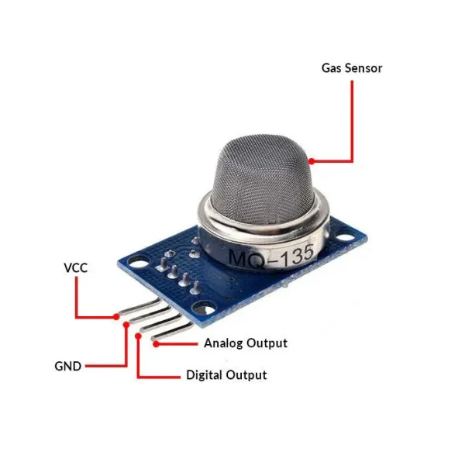


Figure 1 Diagram of the MQ-135 and its pinouts

## PIC16F690 Microcontroller

The PIC16F690 is an 8-bit microcontroller commonly used in embedded systems. It features analog-to-digital conversion (ADC), which is essential for processing the analog output of sensors like the MQ-135. In this project, the PIC16F690 processes the sensor data and drives the LCD display and buzzer. Its low cost and simplicity make it ideal for this application. (Kabilan M)

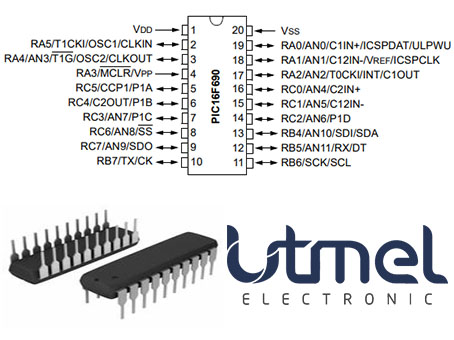


Figure 2 Pinout of the PIC16F690

# Hardware Design

### 4.1 Component Overview

* **PIC16F690 microcontroller**
* **MQ-135 air quality sensor**
* **16x2 LCD display**
* **Buzzer**
* **Passive components** (resistors, capacitors, etc.)

### 4.2 Circuit Design & Implementation

* **MQ-135 Sensor Stage**: The sensor provides an analog output proportional to the concentration of gases. This output is fed into the PIC16F690's ADC.
* **Buzzer and LCD Display**: The microcontroller triggers a buzzer and updates the LCD display when air quality exceeds the threshold.

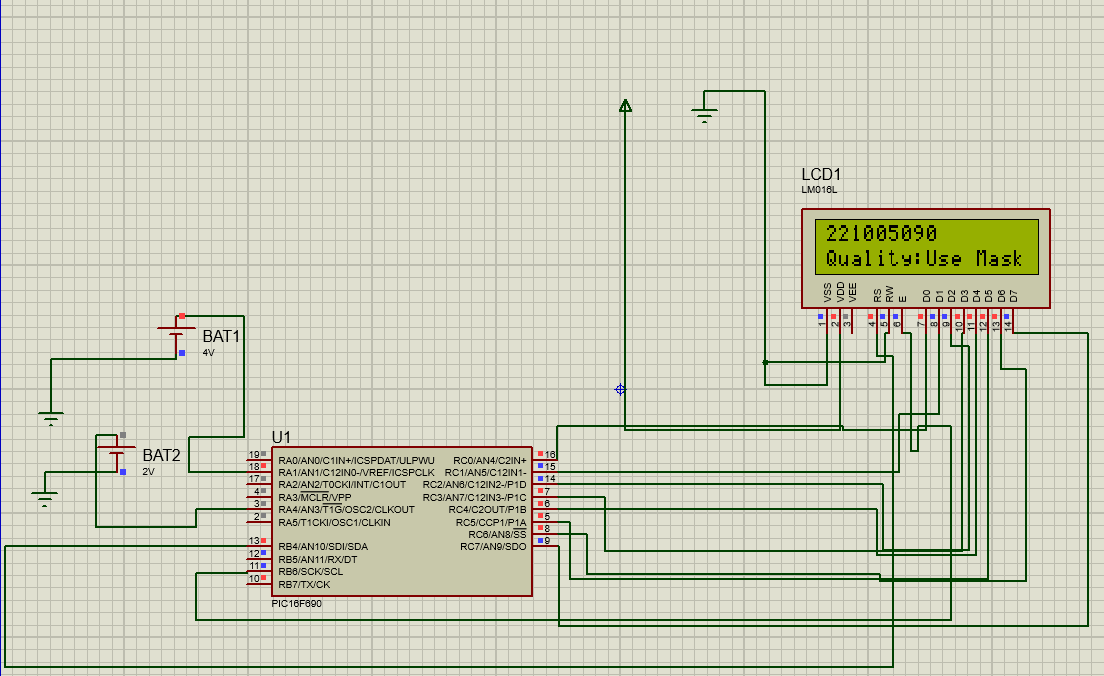


Figure 3 Proteus diagram of the proposed system

## 5.1 Software Design

The code for this project consists of several subroutines responsible for initializing the system, reading air quality levels, converting them to PPM, displaying them on the LCD, and triggering a buzzer when air quality is poor. Below is a breakdown of the key subroutines and their functionality.

### 5.1.1 Main Program

The main program initializes the system and continuously monitors air quality levels by calling the necessary subroutines. The infinite loop (MAINLOOP) ensures the system is always reading the air quality and responding based on the conditions.

### 5.1.2 Key Subroutines

### InitializeLCD:

* This subroutine initializes the 16x2 LCD for display by configuring it for 8-bit mode. It sets the cursor position and prepares the LCD to receive and display data. This step is crucial for showing air quality readings and status messages on the screen.

### ADC Conversion:

Before reading any data from the sensor, the ADC module must be properly configured. This includes:

* Selecting the ADC channel: You must specify which pin of the microcontroller is connected to the sensor.
* Reference Voltage: The PIC16F690 allows the reference voltage (Vref) to be configured. For instance, it can be set to use the system voltage (e.g., 5V) as the reference.

A math equation with white text

Description automatically generated

Figure 4 ADC conversion formula

### ConvertADCtoPPM:

* This subroutine converts the ADC value read from the MQ-135 sensor into a parts per million (PPM) value. This is important for interpreting the sensor’s analog output and converting it into meaningful data that represents air quality levels.

A black background with white text

Description automatically generated

Figure 5 PPM conversion formula

### BuzzerOn and BuzzerOff:

* These subroutines control the buzzer. When BuzzerOn is called, the buzzer sounds, indicating poor air quality. When BuzzerOff is called, the buzzer is silenced.

### LCDWriteMain:

* This subroutine displays the air quality in PPM on the LCD screen. It ensures that the user has real-time visibility of the air quality levels.

### LCDWriteBad:

* This subroutine is responsible for updating the LCD when the air quality exceeds the safe threshold (e.g., 250 PPM). It displays a warning message such as "Polluted Air" and triggers the buzzer.

### **LCDWriteGood**:

* This subroutine updates the LCD to show that the air quality is within safe limits ("Fresh Air"). It also ensures that the buzzer is turned off when the air quality improves.

## Flow Diagram and Pseudo Code

### 5.2.1 Pseudo Code

**Pseudo Code:**

1. **Initialize system**
   * Set up ADC, LCD, and buzzer.
2. **Main Loop**
   * Read ADC value from the sensor.
   * Convert ADC value to PPM.
   * Display PPM value on LCD.
   * If PPM exceeds threshold:
     + Trigger buzzer and display "Polluted Air."
   * Else:
     + Display "Fresh Air."

### 5.2.2 Flow Diagram

### 

# 6. Testing & Results

The system was tested by simulating different air quality levels using the MQ-135 sensor. Various pollutant concentrations were generated to observe how the system responded to changes in air quality. The PPM values were displayed on the LCD, and the buzzer triggered correctly when air quality exceeded the 250 PPM threshold.

## Sample Test Cases:

1. **Fresh Air (PPM < 250)**:

* The system displayed "Fresh Air" on the LCD, and the buzzer remained off.A circuit board with wires and a display

  Description automatically generated

Figure 6 The circuit detecting and displaying Fresh Air

1. **Polluted Air (PPM > 250)**:
   * The system displayed "Polluted Air" on the LCD, and the buzzer was triggered.

A circuit board with wires and a display

Description automatically generated

Figure 7 The circuit detecting and displaying Polluted air

# **Final Testing**

# After modular testing, the entire system was tested to evaluate end-to-end functionality. Different air quality levels were simulated to observe if the system accurately detected threshold levels and updated the LCD display and buzzer accordingly. The final tests confirmed that the system responded quickly and accurately to changes in air quality.

# **Code Performance Figures**

# **ADC Conversion Time**: Approximately 10–20 microseconds, ensuring timely sampling of air quality.

# **LCD Update Frequency**: Updated every 1 second, providing real-time display without overloading the system.

# **Buzzer Response Time**: Immediate, activating within milliseconds when pollution exceeds threshold.

# **Lessons Learnt**

# **Hardware-Software Synchronization**: Aligning hardware responses, such as triggering the buzzer and updating the LCD, with software operations required careful coordination to avoid delays.

# **Sensitivity to Environment**: The MQ-135 sensor’s output can be affected by temperature and humidity, highlighting the need for calibrated conditions or compensation for reliable readings.

# **Debugging ADC Values**: ADC calibration was essential for accurate PPM calculations, especially to scale the sensor’s analog output accurately.

# **Time Taken**

# The project required an estimated:

# **2 days** for circuit design and component selection.

# **3 days** for coding and software testing.

# **1 day** for modular testing.

# **1 day** for final integration and end-to-end testing.

# In total, the project took approximately **1 week** to complete, covering design, development, testing, and final adjustments.

# 7. Conclusion

The air quality monitoring system designed in this project successfully detects indoor air pollutants and provides real-time feedback via an LCD display and buzzer. The system reliably converts the analog sensor output into parts per million (PPM) values, allowing users to monitor their environment's air quality.

This project demonstrates how a low-cost microcontroller-based system can effectively monitor air quality and trigger alerts when pollution exceeds safe levels. While the system performed well during testing, several improvements could be made:

* **Enhanced Range**: Future iterations could incorporate a more sensitive sensor or multiple sensors to cover larger areas.
* **Wireless Connectivity**: Adding a Wi-Fi or Bluetooth module could enable remote monitoring and data logging.
* **Mobile Integration**: Developing a mobile app to receive real-time alerts when air quality drops could further enhance the system’s usability.

Overall, the project provides a solid foundation for further development into a fully functional, cost-effective air quality monitoring system for homes, offices, or schools.

# References

Anand Kanti, Jagadish P M, Prof Soumya. *International Research Journal of Modernization in Engineering Technology and Science* . August 2020. <https://www.irjmets.com/uploadedfiles/paper/volume2/issue\_8\_august\_2020/2844/1628083107.pdf>.

Kabilan M, Kishore R, Magesh R. *AJAST*. June 2024. <https://ajast.net/data/uploads/78261.pdf>.

# Appendix

## Code used

; PIC16F690

#include <p16f690.inc> ; Include the device-specific header

\_\_CONFIG \_CP\_OFF & \_CPD\_OFF & \_BOR\_OFF & \_MCLRE\_ON & \_WDT\_OFF & \_PWRTE\_ON & \_INTRC\_OSC\_NOCLKOUT & \_FCMEN\_OFF & \_IESO\_OFF

GPR\_VAR UDATA

; Start of the program

ORG 0x0000 ; Reset vector

GOTO START ; Jump to the start of the program

; Main program

START:

BCF STATUS, RP0 ; Select Bank 0

BCF STATUS, RP1 ; Ensure Bank 0 is selected

CLRF PORTA ; Clear PORTA

CLRF PORTB ; Clear PORTB

CLRF PORTC ; Clear PORTC

MOVLW b'01001101'

MOVWF ADCON0 ;Left justified, external Vref, enable AN3, enable AD

; Configure I/O ports

BSF STATUS, RP0 ; Select Bank 1

BCF OSCCON,6

BSF OSCCON,5 ;FOSC = 500kHz

BSF OSCCON,4

BSF TRISA,1 ;Set PORTA<1> as input[Vref]

BSF TRISA,4 ;Set PORTA<4> as input for the LM35

BCF TRISB, 7 ; set RB7 as output

CLRF ADCON1

; Set entire PORTB and PORTC as outputs

CLRF TRISB ; Set all pins of PORTB as outputs

CLRF TRISC ; Set all pins of PORTC as outputs

; Set analog inputs to digital

BSF STATUS, RP1 ; Select Bank 2

BCF STATUS, RP0 ; Ensure Bank 2 is selected

CLRF ANSEL ; Set all to digital I/O

CLRF ANSELH ; Set remaining analog pins to digital

BSF ANSEL,3 ;Allow PORTA,4/AN3 to be analog input

BSF ANSEL,1 ;Allow Vref to be analog input (PORTA,5[AN1])/EXT Int

;Bank 0

BCF STATUS,RP0 ;Set back to Bank 0

BCF STATUS,RP1

W EQU 00h

COUNT EQU 26h

W\_TEMP EQU 28h

TMP2 EQU 0x32

TMP3 EQU 0x33

TEMP1\_CONVERT equ 40h ; Temp1 for BCD conversion

TEMP2\_CONVERT equ 41h ; Temp2 for BCD conversion

COUNTER\_CONVERT equ 42h ; Counter for division

VALUE\_TO\_CONVERT equ 0x24 ; Address of the value to convert

TEMP1 EQU 0x50 ; Temporary register for storing 100

TEMP2 EQU 0x51 ; Temporary register for storing 10

C EQU 0

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

MAIN:

MOVLW 0x26

Call InitialiseLCD

Call LCDWriteMain

Call LCDWriteBottomRow

Call BuzzerOff

MAINLOOP

Call ADC

goto MAINLOOP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

InitialiseLCD

call Delay2

bcf PORTB, 4 ; Command

movlw b'00000001'

movwf PORTC

bsf PORTB, 6

call Del1ms

bcf PORTB, 6

call Del1ms

movlw b'00000010' ; Cursor return to home

movwf PORTC

bsf PORTB, 6

call Del1ms

bcf PORTB, 6

call Del1ms

movlw b'00000110'

movwf PORTC

bsf PORTB, 6

call Del1ms

bcf PORTB, 6

call Del1ms

movlw b'00001111'

movwf PORTC

bsf PORTB, 6

call Del1ms

bcf PORTB, 6

call Del1ms

movlw b'00010100'

movwf PORTC

bsf PORTB, 6

call Del1ms

bcf PORTB, 6

call Del1ms

movlw b'00111100'

movwf PORTC

bsf PORTB, 6

call Del1ms

bcf PORTB, 6

call Del1ms

movlw 0x80

movwf PORTC

bsf PORTB, 6

call Del1ms

bcf PORTB, 6

call Del1ms

return

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

; Buzzer Subroutine

BuzzerOn

bsf PORTB, 7 ; Set RB7 high to turn on the buzzer

return

BuzzerOff

bcf PORTB, 7 ; Clear RB7 to turn off the buzzer

return

;\*\*\*\*\*\*\*\*\*\*DELAY 1MS\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Del1ms movlw d'250' ; Initial value

loop1ms addlw d'255' ; Dec WREG

btfss STATUS,Z ; Zero flag set?

goto loop1ms ; No, keep looping

return ; Yes, 1ms done

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;\*\*\*\*\*\*\*\*\*\*100 MS DELAY\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Delay2

movlw 0x64

movwf TMP2

loopDel2

call Del1ms

decfsz TMP2

goto loopDel2

return

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;\*\*\*\*\*\*\*\*\*\*1 S DELAY\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Delay1s

movlw 0x0A

movwf TMP3

Wait

call Delay2

decfsz TMP3

goto Wait

return

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;\*\*\*\*\*\*\*\*\*\*340ms DELAY\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Delay movlw d'255'

movwf COUNT

Begin movlw d'250' ; Initial value

loopDel addlw d'255' ; Dec WREG

btfss STATUS,Z ; Zero flag set?

goto loopDel ; No, keep looping

decfsz COUNT

goto Begin

return ; Yes, 340ms done

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

LCDWriteMain ; Write home message when there is no detection

bcf PORTB,7 ; Disable alarm

bcf PORTB,4 ; Select LCD command register

movlw b'0000001' ; Clear LCD screen

movwf PORTC

bsf PORTB, 6 ; Toggle Data Enable

bcf PORTB, 6

bsf PORTB,4

call Del1ms

; Write the message "Value"

movlw '2'

movwf PORTC

bsf PORTB,6

bcf PORTB,6

call Del1ms

movlw '2'

movwf PORTC

bsf PORTB, 6

bcf PORTB, 6

Call Del1ms

movlw '1'

movwf PORTC

bsf PORTB, 6

bcf PORTB, 6

Call Del1ms

movlw '0'

movwf PORTC

bsf PORTB,6

bcf PORTB,6

call Del1ms

movlw '0'

movwf PORTC

bsf PORTB,6

bcf PORTB,6

call Del1ms

movlw '5'

movwf PORTC

bsf PORTB,6

bcf PORTB,6

call Del1ms

movlw '0'

movwf PORTC

bsf PORTB,6

bcf PORTB,6

call Del1ms

movlw '9'

movwf PORTC

bsf PORTB,6

bcf PORTB,6

call Del1ms

movlw '0'

movwf PORTC

bsf PORTB,6

bcf PORTB,6

call Del1ms

return

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

LCDWriteBottomRow

bcf PORTB, 4 ; Select LCD command register

movlw 0xC0 ; Move cursor to the beginning of the bottom row

movwf PORTC

bsf PORTB, 6 ; Enable Data

bcf PORTB, 6 ; Disable Data

call Del1ms

; Write 'Q'

bsf PORTB, 4 ; Select LCD data register

movlw 'Q'

movwf PORTC

bsf PORTB, 6

bcf PORTB, 6

call Del1ms

; Write 'U'

movlw 'u'

movwf PORTC

bsf PORTB, 6

bcf PORTB, 6

call Del1ms

; Write 'A'

movlw 'a'

movwf PORTC

bsf PORTB, 6

bcf PORTB, 6

call Del1ms

; Write 'L'

movlw 'l'

movwf PORTC

bsf PORTB, 6

bcf PORTB, 6

call Del1ms

; Write 'I'

movlw 'i'

movwf PORTC

bsf PORTB, 6

bcf PORTB, 6

call Del1ms

; Write 'T'

movlw 't'

movwf PORTC

bsf PORTB, 6

bcf PORTB, 6

call Del1ms

; Write 'Y'

movlw 'y'

movwf PORTC

bsf PORTB, 6

bcf PORTB, 6

call Del1ms

; Write ':'

movlw ':'

movwf PORTC

bsf PORTB, 6

bcf PORTB, 6

call Del1ms

return

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ADC

BSF ADCON0,1 ;Start ADC conversion

Wait\_loop

BTFSC ADCON0,1 ;Checks if conversion done, if so, exit loop

GOTO Wait\_loop

MOVFW ADRESH ;Moves the higher 8 bits into W

MOVWF 0x20

MOVWF 0x24

Call CompareThresholdGood

RETURN

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

LCDWriteBad

Call BuzzerOn

bcf PORTB, 4 ; Select LCD command register

movlw 0x0C ; Command to turn off the cursor and blinking

movwf PORTC

bsf PORTB, 6 ; Enable Data

bcf PORTB, 6 ; Disable Data

call Del1ms ; Delay to allow the command to process

; Position cursor on the bottom row, 8 columns to the right (0xC8)

bcf PORTB, 4 ; Select LCD command register

movlw 0xC0 ; Move cursor to the 8th position of the bottom row

movwf PORTC

bsf PORTB, 6 ; Enable Data

bcf PORTB, 6 ; Disable Data

call Del1ms

; Write 'Q'

bsf PORTB, 4 ; Select LCD data register

movlw 'Q'

movwf PORTC

bsf PORTB, 6

bcf PORTB, 6

call Del1ms

; Write 'U'

movlw 'u'

movwf PORTC

bsf PORTB, 6

bcf PORTB, 6

call Del1ms

; Write 'A'

movlw 'a'

movwf PORTC

bsf PORTB, 6

bcf PORTB, 6

call Del1ms

; Write 'L'

movlw 'l'

movwf PORTC

bsf PORTB, 6

bcf PORTB, 6

call Del1ms

; Write 'I'

movlw 'i'

movwf PORTC

bsf PORTB, 6

bcf PORTB, 6

call Del1ms

; Write 'T'

movlw 't'

movwf PORTC

bsf PORTB, 6

bcf PORTB, 6

call Del1ms

; Write 'Y'

movlw 'y'

movwf PORTC

bsf PORTB, 6

bcf PORTB, 6

call Del1ms

; Write ':'

movlw ':'

movwf PORTC

bsf PORTB, 6

bcf PORTB, 6

call Del1ms

; Write 'B'

bsf PORTB, 4 ; Select LCD data register

movlw 'U'

movwf PORTC

bsf PORTB, 6 ; Enable Data

bcf PORTB, 6 ; Disable Data

call Del1ms

; Write 'a'

movlw 's'

movwf PORTC

bsf PORTB, 6

bcf PORTB, 6

call Del1ms

; Write 'd'

movlw 'e'

movwf PORTC

bsf PORTB, 6

bcf PORTB, 6

call Del1ms

; Write 'd'

movlw ' '

movwf PORTC

bsf PORTB, 6

bcf PORTB, 6

call Del1ms

; Write 'd'

movlw 'M'

movwf PORTC

bsf PORTB, 6

bcf PORTB, 6

call Del1ms

movlw 'a'

movwf PORTC

bsf PORTB, 6

bcf PORTB, 6

call Del1ms

movlw 's'

movwf PORTC

bsf PORTB, 6

bcf PORTB, 6

call Del1ms

movlw 'k'

movwf PORTC

bsf PORTB, 6

bcf PORTB, 6

call Del1ms

movlw ' '

movwf PORTC

bsf PORTB, 6

bcf PORTB, 6

call Del1ms

return

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

LCDWriteGood

Call BuzzerOff

bcf PORTB, 4 ; Select LCD command register

movlw 0x0C ; Command to turn off the cursor and blinking

movwf PORTC

bsf PORTB, 6 ; Enable Data

bcf PORTB, 6 ; Disable Data

call Del1ms ; Delay to allow the command to process

; Position cursor on the bottom row, 8 columns to the right (0xC8)

bcf PORTB, 4 ; Select LCD command register

movlw 0xC0 ; Move cursor to the 8th position of the bottom row

movwf PORTC

bsf PORTB, 6 ; Enable Data

bcf PORTB, 6 ; Disable Data

call Del1ms

; Write 'Q'

bsf PORTB, 4 ; Select LCD data register

movlw 'Q'

movwf PORTC

bsf PORTB, 6

bcf PORTB, 6

call Del1ms

; Write 'U'

movlw 'u'

movwf PORTC

bsf PORTB, 6

bcf PORTB, 6

call Del1ms

; Write 'A'

movlw 'a'

movwf PORTC

bsf PORTB, 6

bcf PORTB, 6

call Del1ms

; Write 'L'

movlw 'l'

movwf PORTC

bsf PORTB, 6

bcf PORTB, 6

call Del1ms

; Write 'I'

movlw 'i'

movwf PORTC

bsf PORTB, 6

bcf PORTB, 6

call Del1ms

; Write 'T'

movlw 't'

movwf PORTC

bsf PORTB, 6

bcf PORTB, 6

call Del1ms

; Write 'Y'

movlw 'y'

movwf PORTC

bsf PORTB, 6

bcf PORTB, 6

call Del1ms

; Write ':'

movlw ':'

movwf PORTC

bsf PORTB, 6

bcf PORTB, 6

call Del1ms

; Write 'g'

bsf PORTB, 4 ; Select LCD data register

movlw 'F'

movwf PORTC

bsf PORTB, 6 ; Enable Data

bcf PORTB, 6 ; Disable Data

call Del1ms

; Write 'o'

movlw 'r'

movwf PORTC

bsf PORTB, 6

bcf PORTB, 6

call Del1ms

; Write 'o'

movlw 'e'

movwf PORTC

bsf PORTB, 6

bcf PORTB, 6

call Del1ms

; Write 'd'

movlw 's'

movwf PORTC

bsf PORTB, 6

bcf PORTB, 6

call Del1ms

movlw 'h'

movwf PORTC

bsf PORTB, 6

bcf PORTB, 6

call Del1ms

movlw 'A'

movwf PORTC

bsf PORTB, 6

bcf PORTB, 6

call Del1ms

movlw 'i'

movwf PORTC

bsf PORTB, 6

bcf PORTB, 6

call Del1ms

movlw 'r'

movwf PORTC

bsf PORTB, 6

bcf PORTB, 6

call Del1ms

return

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

; Routine to check if value in register 0x20 is less than 150

CompareThresholdGood

MOVF 0x20, W ; Move the value from 0x20 into WREG

SUBLW 0x26 ; Subtract 100 from WREG

; If the result is negative, the original value was less than 100

; The Zero flag will be clear, and the Carry flag will be set for negative result

BTFSC STATUS, C ; Check if Carry flag is set (indicating WREG < 100)

Call LCDWriteGood ; If not less than 100, skip calling LCDWriteGood

BTFSC STATUS, C ; Check if Carry flag is set (indicating WREG < 100)

RETURN ; Return from the routine

CALL CompareThresholdBad ; Call LCDWriteGood if the value is less than 100

Return

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

; Routine to check if value in register 0x20 is less than 100

CompareThresholdBad

MOVF 0x20, W ; Move the value from 0x20 into WREG

SUBLW 0x15 ; Subtract 100 from WREG

; If the result is negative, the original value was less than 100

; The Zero flag will be clear, and the Carry flag will be set for negative result

BTFSC STATUS, C ; Check if Carry flag is set (indicating WREG < 100)

RETURN ; If not less than 100, skip calling LCDWriteGood

CALL LCDWriteBad ; Call LCDWriteGood if the value is less than 100

RETURN ; Return from the routine

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

END