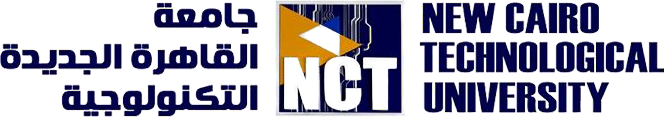
**Faculty of Industry and Energy Technology**

**Information and Communications Technology (ICT) Program**

**Information and Communications Technology (ICT)**

Microcontroller Experiments with PIC16F877A (Practical Guide with Proteus Schematics and MikroC Code)

Presented By:

Eslam Mohamed Dahy

Ahmed Waheed Ismail

Eslam Mohamed Atia

Mazen Mohamed Salah

Farouk Mohsen Adly

Khaled Ibrahim khodary

Supervised By:

Dr. Osama Tharwat

Eng. Moamen Ibrahim

# ACKNOWLEDGMENT

First and foremost, I would like to express my sincere gratitude and appreciation to **Dr. Osama**, the course instructor, for his continuous support, exceptional guidance, and deep commitment to delivering high-quality education throughout the Embedded Systems course. His ability to present complex theoretical concepts in a clear and engaging manner helped us build a strong foundation in this vital field of engineering. The inspiration and motivation he provided encouraged us to think critically, explore beyond the syllabus, and approach every challenge with confidence.

I am also deeply thankful to **Eng. Moamen**, our lab instructor and teaching assistant, for his remarkable dedication and patience during the practical sessions. His efforts in helping us understand the real-world applications of microcontrollers, especially the PIC16F877A, were instrumental in translating theory into practice. His clear explanations, hands-on demonstrations, and constant availability for questions played a crucial role in our learning process.

Special thanks go to our teammates and fellow students whose collaboration, cooperation, and shared enthusiasm made this journey smoother and more enjoyable. Working together on experiments, debugging circuits, and sharing knowledge created a truly supportive and productive environment.

Lastly, I would like to thank our university and the technical staff for providing the necessary equipment, software tools, and lab facilities that enabled us to perform all experiments efficiently and effectively. Without these resources, this report would not have been possible.

This report is the result of the combined effort of everyone involved, and I am truly grateful for all the support and encouragement I received throughout this project.

# ABSTRACT

This report provides an in-depth overview of a series of practical experiments conducted using the PIC16F877A microcontroller, one of the most widely used microcontrollers in embedded systems education and industry. The goal of this report is to introduce students to the fundamental concepts of embedded systems through a structured and hands-on approach. By performing real-world experiments, students not only enhance their technical skills but also gain a deeper understanding of how microcontrollers interact with various electronic components and sensors.

The PIC16F877A was chosen for its rich feature set, including multiple I/O ports, ADC modules, timers, and communication interfaces, making it an ideal choice for a wide range of embedded applications. The experiments covered in this report gradually progress from basic to more advanced levels, starting with simple tasks such as blinking an LED and advancing to more complex operations like reading analog sensor values and displaying real-time data on an LCD.

The report includes 15 experiments, each designed to focus on specific aspects of embedded system design. These include: LED control (basic and sequential), input handling with push buttons, analog signal processing using potentiometers, temperature sensors (LM35), and light sensors (LDR), as well as distance measurement with ultrasonic sensors. Additionally, output interfacing is explored through the use of character LCDs, 7-segment displays, and servo motors.

This compilation of experiments not only reinforces theoretical knowledge but also encourages problem-solving, debugging, and hardware-software integration skills. It is designed to serve as a foundational learning tool for students and enthusiasts aiming to pursue careers in embedded systems, electronics, or IoT development.

# LIST OF FIGURES

[*Figure 1-1: PIC16F877A Datasheet* 4](file://localhost/C:/Users/dahy/Desktop/GProject/doc/NCTU%20Doc%20-%20HRVM.docx%23_Toc170682625)

[*Figure 1-2: Blinking LED Proteus* 6](file://localhost/C:/Users/dahy/Desktop/GProject/doc/NCTU%20Doc%20-%20HRVM.docx%23_Toc170682626)

[*Figure 1-3: Blinking LED Code* 7](file://localhost/C:/Users/dahy/Desktop/GProject/doc/NCTU%20Doc%20-%20HRVM.docx%23_Toc170682627)

[*Figure 1-4: Sequential Three LEDs Proteus* 8](file://localhost/C:/Users/dahy/Desktop/GProject/doc/NCTU%20Doc%20-%20HRVM.docx%23_Toc170682626)

[*Figure 1-5: Sequential Three LEDs Code* 9](file://localhost/C:/Users/dahy/Desktop/GProject/doc/NCTU%20Doc%20-%20HRVM.docx%23_Toc170682627)

[*Figure 1-6: Push Button Input Proteus* 10](file://localhost/C:/Users/dahy/Desktop/GProject/doc/NCTU%20Doc%20-%20HRVM.docx%23_Toc170682626)

[*Figure 1-7: Push Button Input Code* 100](file://localhost/C:/Users/dahy/Desktop/GProject/doc/NCTU%20Doc%20-%20HRVM.docx%23_Toc170682627)

[*Figure 1-8: Controlling an LED Using a Push Button Proteus* 12](file://localhost/C:/Users/dahy/Desktop/GProject/doc/NCTU%20Doc%20-%20HRVM.docx%23_Toc170682626)

[*Figure 1-9: Controlling an LED Using a Push Button Code* 13](file://localhost/C:/Users/dahy/Desktop/GProject/doc/NCTU%20Doc%20-%20HRVM.docx%23_Toc170682627)

[*Figure 1-10: Controlling an LED Using a Potentiometer Proteus* 14](file://localhost/C:/Users/dahy/Desktop/GProject/doc/NCTU%20Doc%20-%20HRVM.docx%23_Toc170682626)

[*Figure 1-11: Controlling an LED Using a Potentiometer Code* 15](file://localhost/C:/Users/dahy/Desktop/GProject/doc/NCTU%20Doc%20-%20HRVM.docx%23_Toc170682627)

[*Figure 1-12: LM35 Interfacing Proteus* 17](file://localhost/C:/Users/dahy/Desktop/GProject/doc/NCTU%20Doc%20-%20HRVM.docx%23_Toc170682626)

[*Figure 1-13: LM35 Interfacing Code* 18](file://localhost/C:/Users/dahy/Desktop/GProject/doc/NCTU%20Doc%20-%20HRVM.docx%23_Toc170682627)

[*Figure 1-14: LDR Interfacing Proteus* 20](file://localhost/C:/Users/dahy/Desktop/GProject/doc/NCTU%20Doc%20-%20HRVM.docx%23_Toc170682626)

[*Figure 1-15:* *LDR Interfacing Code* 21](file://localhost/C:/Users/dahy/Desktop/GProject/doc/NCTU%20Doc%20-%20HRVM.docx%23_Toc170682627)

[*Figure 1-16: Ultrasonic Interfacing Proteus* 23](file://localhost/C:/Users/dahy/Desktop/GProject/doc/NCTU%20Doc%20-%20HRVM.docx%23_Toc170682626)

[*Figure 1-17: Ultrasonic Interfacing Code* 24](file://localhost/C:/Users/dahy/Desktop/GProject/doc/NCTU%20Doc%20-%20HRVM.docx%23_Toc170682627)

[*Figure 1-18: LCD Interfacing (Static Text) Proteus* 26](file://localhost/C:/Users/dahy/Desktop/GProject/doc/NCTU%20Doc%20-%20HRVM.docx%23_Toc170682626)

[*Figure 1-19: LCD Interfacing (Static Text) Code* 27](file://localhost/C:/Users/dahy/Desktop/GProject/doc/NCTU%20Doc%20-%20HRVM.docx%23_Toc170682627)

[*Figure 1-20: LCD Interfacing (Scrolling Text) Proteus* 29](file://localhost/C:/Users/dahy/Desktop/GProject/doc/NCTU%20Doc%20-%20HRVM.docx%23_Toc170682626)

[*Figure 1-21: LCD Interfacing (Scrolling Text) Code* 30](file://localhost/C:/Users/dahy/Desktop/GProject/doc/NCTU%20Doc%20-%20HRVM.docx%23_Toc170682627)

[*Figure 1-22: Display a Temperature Value Proteus* 32](file://localhost/C:/Users/dahy/Desktop/GProject/doc/NCTU%20Doc%20-%20HRVM.docx%23_Toc170682626)

[*Figure 1-23: Display a Temperature Value Code* 33](file://localhost/C:/Users/dahy/Desktop/GProject/doc/NCTU%20Doc%20-%20HRVM.docx%23_Toc170682627)

[*Figure 1-24: Display a Distance Value Proteus* 35](file://localhost/C:/Users/dahy/Desktop/GProject/doc/NCTU%20Doc%20-%20HRVM.docx%23_Toc170682626)

[*Figure 1-25: Display a Distance Value Code* 36](file://localhost/C:/Users/dahy/Desktop/GProject/doc/NCTU%20Doc%20-%20HRVM.docx%23_Toc170682627)

[*Figure 1-26: 7-Segment Display Interfacing Proteus* 37](file://localhost/C:/Users/dahy/Desktop/GProject/doc/NCTU%20Doc%20-%20HRVM.docx%23_Toc170682626)

[*Figure 1-27: 7-Segment Display Interfacing Code* 40](file://localhost/C:/Users/dahy/Desktop/GProject/doc/NCTU%20Doc%20-%20HRVM.docx%23_Toc170682627)

[*Figure 1-28: Controlling a 7-Segment Using Potentiometer Proteus* 49](file://localhost/C:/Users/dahy/Desktop/GProject/doc/NCTU%20Doc%20-%20HRVM.docx%23_Toc170682626)

[*Figure 1-29: Controlling a 7-Segment Using Potentiometer Code* 53](file://localhost/C:/Users/dahy/Desktop/GProject/doc/NCTU%20Doc%20-%20HRVM.docx%23_Toc170682627)

[*Figure 1-30: Servo Motor Interfacing Proteus* 45](file://localhost/C:/Users/dahy/Desktop/GProject/doc/NCTU%20Doc%20-%20HRVM.docx%23_Toc170682626)

[*Figure 1-31: Servo Motor Interfacing Code* 46](file://localhost/C:/Users/dahy/Desktop/GProject/doc/NCTU%20Doc%20-%20HRVM.docx%23_Toc170682627)

TABLE OF CONTENTS

ACKNOWLEDGMENT i

ABSTRACT ii

LIST OF FIGURES iii

[CHAPTER ONE INTRODUCTION 2](#_bookmark1)

* 1. [Overview 2](#_bookmark2)
  2. [Why PIC Microcontroller? 2](#_bookmark3)
  3. [Pic DataSheet 3](#_bookmark3)
  4. [Objective of the Report 3](#_bookmark4)
  5. [Structure of the Report 4](#_bookmark5)

[CHAPTER TWO PRACTICAL EXPERIMENTS WITH PIC16F877A 5](#_bookmark6)

* 1. [Overview 5](#_bookmark7)
  2. [Experiment 1: Blinking LED 5](#_bookmark8)
  3. [Experiment 2: Sequential Blinking of Three LEDs 7](#_bookmark8)
  4. [Experiment 3: Push Button Input 9](#_bookmark8)
  5. [Experiment 4: Controlling an LED Using a Push Button 12](#_bookmark8)
  6. [Experiment 5: Controlling an LED Using a Potentiometer 14](#_bookmark8)
  7. [Experiment 6: LM35 Interfacing 17](#_bookmark8)
  8. [Experiment 7: LDR Interfacing 9](#_bookmark8)
  9. [Experiment 8: Ultrasonic Interfacing 9](#_bookmark8)
  10. [Experiment 9: LCD Interfacing (Static Text) 22](#_bookmark8)
  11. [Experiment 10: LCD Interfacing (Scrolling Text) 28](#_bookmark8)
  12. [Experiment 11: Display a Temperature Value 31](#_bookmark8)
  13. [Experiment 12: Display a Distance Value 34](#_bookmark8)
  14. [Experiment 13: 7-Segment Display Interfacing 38](#_bookmark8)
  15. [Experiment 14: Controlling a 7-Segment Using Potentiometer 41](#_bookmark8)
  16. [Experiment 15: Servo Motor Interfacing 45](#_bookmark8)

[CHAPTER THREE CONCLUSION 48](#_bookmark6)

[REFERENCE 49](#_bookmark67)

C H A P T E R O N E

# INTRODUCTION

## Overview:

Embedded systems have become an essential part of modern technology, integrating software and hardware to perform dedicated functions within larger systems. From household appliances and industrial machines to medical devices and automotive systems, embedded systems play a critical role in automation and control. Understanding how these systems work is therefore a crucial step for any aspiring electronics or computer engineer.

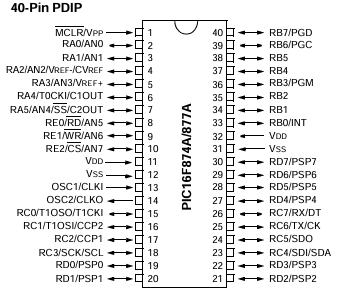
This report focuses on exploring embedded systems through hands-on experimentation using the **PIC16F877A microcontroller**, a widely used and well-documented microcontroller in both academic and industrial applications. PIC microcontrollers, developed by Microchip Technology, are known for their simplicity, versatility, and reliability, making them an ideal choice for educational purposes and small- to medium-scale embedded projects.

## Why PIC Microcontroller:

The PIC16F877A was chosen for several reasons:

* It offers a wide range of I/O ports, analog-to-digital converters (ADC), timers, and communication protocols (such as USART and I2C), all within a single chip.
* It is supported by user-friendly development tools such as MikroC PRO for PIC and simulation platforms like Proteus, which facilitate learning and debugging.
* It allows learners to quickly prototype and test their ideas in real-time, developing both programming and hardware integration skills.

## PIC Datasheet:



**Figure 1: PIC16F877A Datasheet**

ss

## Objective of the Report:

The main objective of this report is to provide students with practical experience in working with embedded systems, focusing on:

* Understanding how microcontrollers interact with input and output devices.
* Writing structured and functional embedded C code using MikroC.
* Designing and simulating embedded circuits using Proteus software.
* Applying problem-solving techniques to real-world scenarios.

## Structure of the Report:

This report consists of 15 practical experiments, each designed to highlight a specific concept in embedded systems. Starting from basic LED blinking tasks, the experiments gradually increase in complexity, introducing analog sensors, user inputs, and output displays. Each experiment includes:

* A clear description and goal.
* List of components.
* Schematic design using Proteus.
* Well-commented MikroC code.
* Explanation of expected behavior and results.

C H A P T E R T W O

# PRACTICAL EXPERIMENTS WITH PIC16F877A

## Overview:

In most of the experiments performed in this project, certain components will be used consistently across different setups. These components are essential for various types of experiments and are included in each practical experiment. Below is a list of the common components used throughout the project:

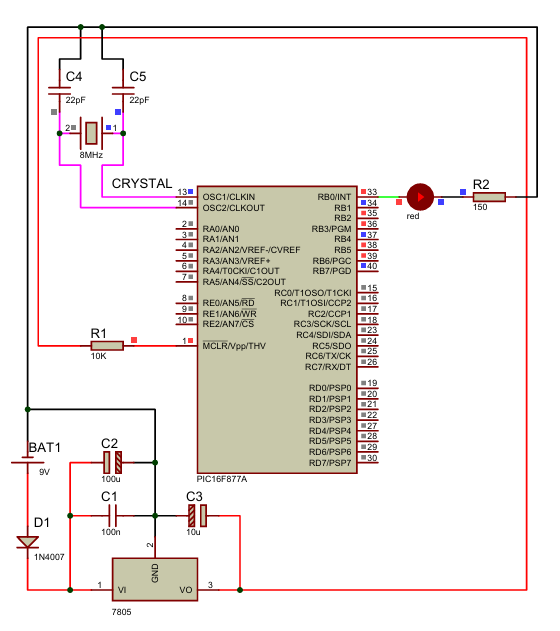
* PIC16F877A Microcontroller.
* Crystal Oscillator (8 MHz).
* 5V Regulator (L7805CV)
* Capacitors:
  + Crystal (22pF)
  + VN Regulator (100nF, 100uF)
* Diodes (1N4007)
* Battery Source (9V)

## Experiment 1: Blinking LED:

* **Objective:**

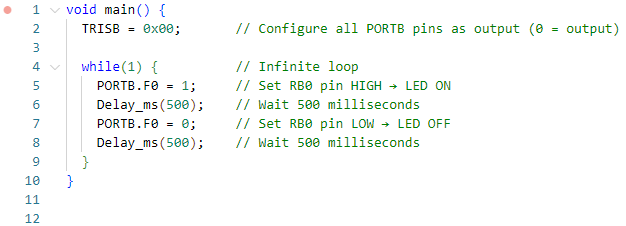
The first experiment focuses on controlling a basic LED using a PIC16F877A microcontroller. The goal is to blink the LED on and off in a predefined interval, allowing students to understand how to control simple output devices using microcontrollers.

* **Components Used:**
  + **LED**.
  + **Resistor** (150Ω).
  + Breadboard and jumper wires.
* **Proteus Schematic:**



**Figure 1.2: Blinking LED Proteus**

* **MikroC Code:**



**Figure 1.3 Blinking LED Code**

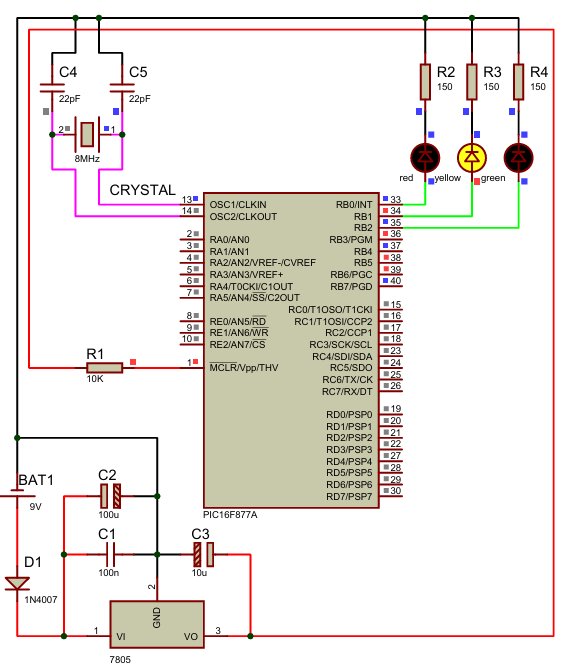
* **Explanation:**
* TRISB = 0x00: This sets all PORTB pins (RB0–RB7) as output. Since we are using RB0 for the LED, this is necessary so that the pin can drive current to the LED.
* PORTB.F0 = 1: This sets the RB0 pin to HIGH (5V). In your circuit, this causes current to flow from RB0 through the LED and resistor to ground → LED turns ON.
* Delay\_ms(500): Keeps the LED ON for half a second.
* PORTB.F0 = 0: Sets RB0 LOW (0V) → no voltage, current stops flowing → LED turns OFF.
* The while(1) loop ensures this ON/OFF cycle repeats forever → the LED keeps blinking every 1 second (0.5s ON, 0.5s OFF).

## Experiment 2: Sequential Three LEDs:

* **Objective:**

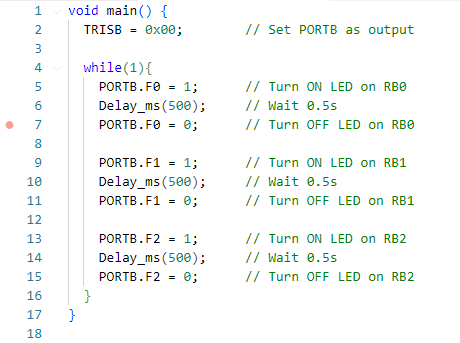
This experiment demonstrates how to control multiple LEDs. The goal is to light up three LEDs one after the other in a continuous loop.

* **Components Used:**
  + **3 LEDs**.
  + **Resistor** (150Ω for each LED).
  + Breadboard and jumper wires.
* **Proteus Schematic:**



**Figure 1.4: Sequential Three LEDs Proteus**

* **MikroC Code:**



**Figure 1.5: Sequential Three LEDs Code**

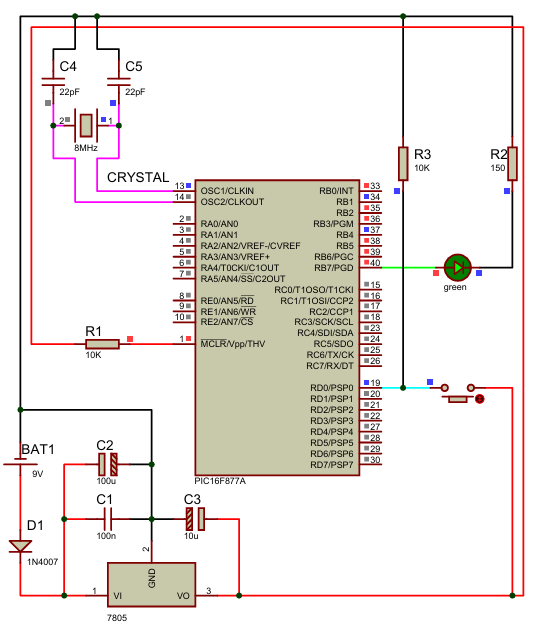
* **Explanation:**
* TRISB = 0x00: Sets all PORTB pins (RB0–RB7) as output so the microcontroller can control them.
* Each LED turns ON and OFF in order:
* PORTB.F0 = 1: Sets RB0 to HIGH → LED1 turns ON.
* Delay\_ms(500): Keeps LED1 ON for 0.5 seconds.
* PORTB.F0 = 0: Turns LED1 OFF.
* The same pattern follows for RB1 (LED2) and RB2 (LED3).
* The while(1) loop repeats the sequence forever, creating a chasing/blinking LED effect.

## Experiment 3: Push Button Input:

* **Objective:**

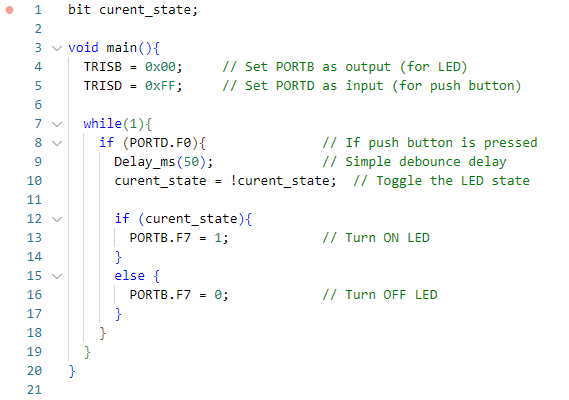
This experiment introduces the concept of using digital input with the PIC16F877A. A push button is used to demonstrate how the microcontroller can detect a button press.

* **Components Used:**
  + **Push Button.**
  + **Resistor** (10kΩ for pull-down).
  + Breadboard and jumper wires.
* **Proteus Schematic:**



**Figure 1.6: Push Button Input Proteus**

* **MikroC Code:**



**Figure 1.7: Push Button Input Code**

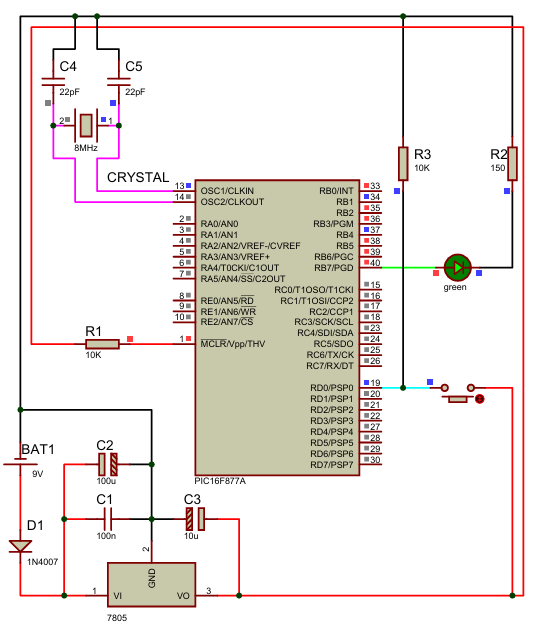
* **Explanation:**
* TRISB = 0x00: Sets PORTB as output to control the LED.
* TRISD = 0xFF: Sets PORTD as input to read the push button state.
* PORTD.F0: Checks the digital input from RD0 (button pin). When the button is pressed, the pin reads 1.
* Delay\_ms(50): Adds a small delay for debouncing, so one press is registered as a single toggle.
* curent\_state = !curent\_state: This line toggles the LED state:
* If LED was OFF → turns ON
* If LED was ON → turns OFF
* PORTB.F7: Controls the LED:
* 1 → ON
* 0 → OFF.

## Experiment 4: Controlling an LED Using a Push Button:

* **Objective:**

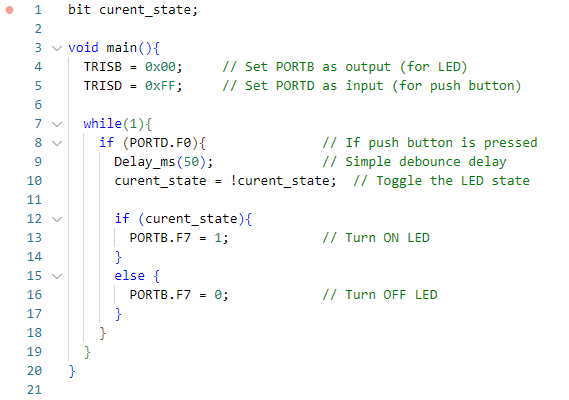
This experiment demonstrates how to use a push button to control an LED using the PIC16F877A microcontroller. It helps students understand how input and output work together in real-time.

* **Components Used:**
  + **LED**.
  + **Push Button**
  + **Resistor** (220Ω for LED, 10kΩ for pull-down).
  + Breadboard and jumper wires.
* **Proteus Schematic:**



**Figure 1.8: Controlling an LED Using a Push Button Proteus**

* **MikroC Code:**



**Figure 1.9: Controlling an LED Using a Push Button Code**

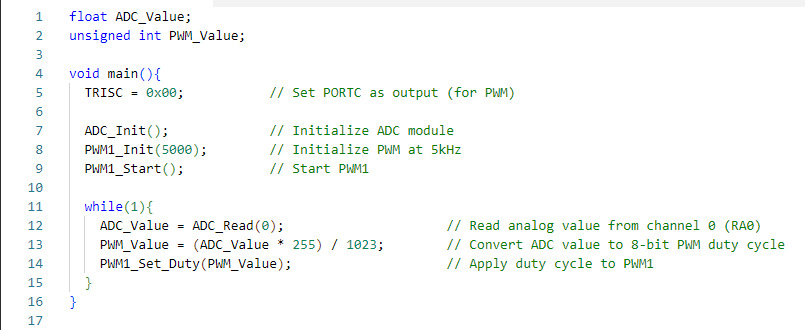
* **Explanation:**
* TRISB = 0x00: Sets PORTB as output to control the LED.
* TRISD = 0xFF: Sets PORTD as input to read the push button state.
* PORTD.F0: Checks the digital input from RD0 (button pin). When the button is pressed, the pin reads 1.
* Delay\_ms(50): Adds a small delay for debouncing, so one press is registered as a single toggle.
* curent\_state = !curent\_state: This line toggles the LED state:
* If LED was OFF → turns ON
* If LED was ON → turns OFF
* PORTB.F7: Controls the LED:
* 1 → ON
* 0 → OFF.

## Experiment 5: Controlling an LED Using a Potentiometer:

* **Objective:**

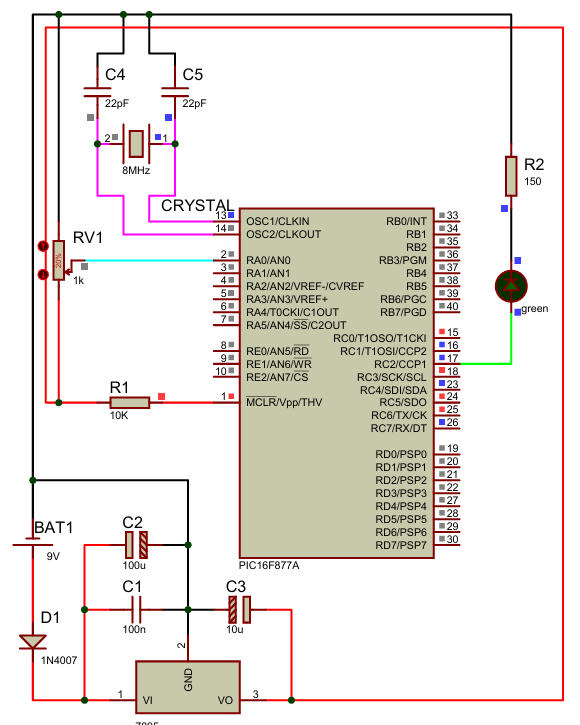
This experiment aims to demonstrate how to use an analog input (potentiometer) to control the brightness of an LED using PWM (Pulse Width Modulation) on the PIC16F877A. This helps students understand analog-to-digital conversion and PWM signal generation.

* **Components Used:**
  + **LED**
  + **Potentiometer**
  + **Resistor** (150Ω)
  + Breadboard and jumper wires.
* **MikroC Code:**



**Figure 1.11: Controlling an LED Using a Potentiometer Code**

* **Proteus Schematic:**



**Figure 1.10: Controlling an LED Using a Potentiometer Proteus**

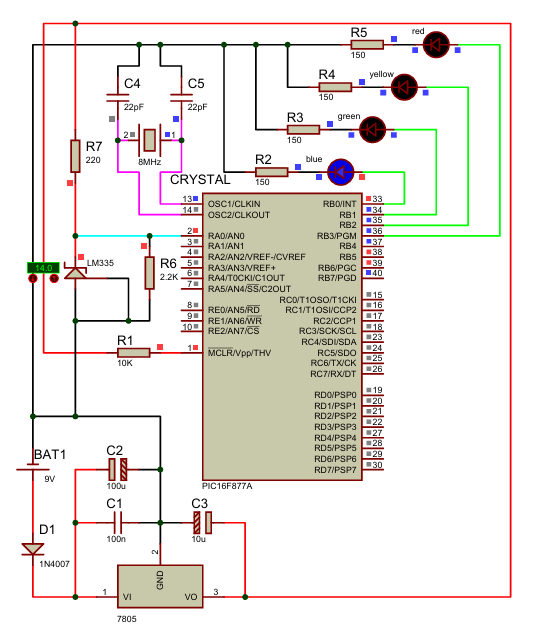
* **Explanation:**
* TRISC = 0x00;: Makes PORTC output to allow PWM signals.
* ADC\_Init();: Initializes the internal ADC to read analog values (like from a potentiometer).
* PWM1\_Init(5000);: Sets up PWM on RC2 at a frequency of 5kHz (good for LED dimming).
* PWM1\_Start();: Starts PWM1 output.
* ADC\_Read(0);: Reads analog value from channel 0 (RA0), returns value between 0 and 1023.
* PWM\_Value = (ADC\_Value \* 255) / 1023;: Maps 10-bit ADC (0–1023) to 8-bit PWM (0–255).
* PWM1\_Set\_Duty(PWM\_Value);: Sets the PWM duty cycle, changing LED brightness.

## Experiment 6: LM35 Interfacing:

* **Objective:**

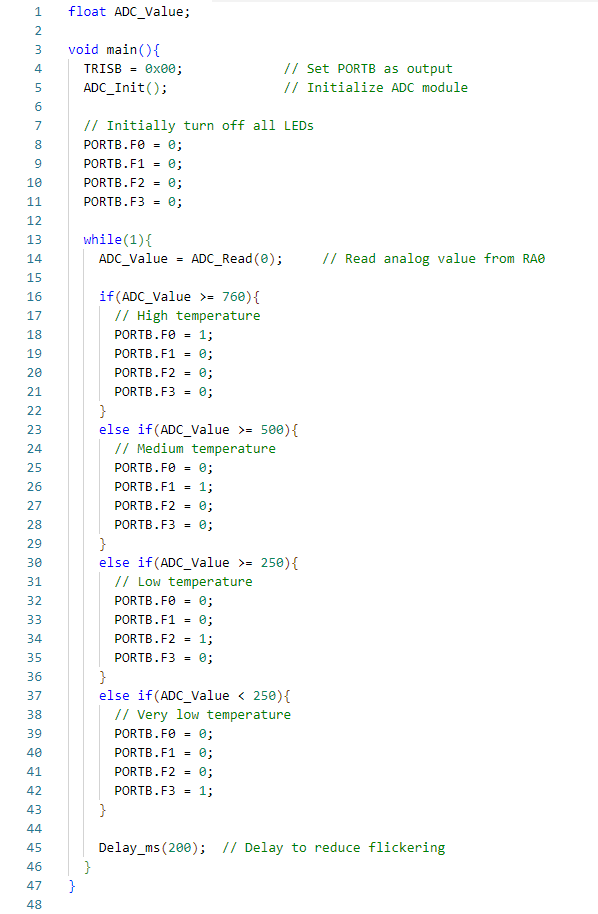
This experiment aims to read analog temperature data from the LM35 sensor using the PIC16F877A’s ADC module and display the result as a temperature in Celsius. It introduces students to sensor interfacing and ADC-to-temperature conversion.

* **Components Used:**
  + **LM35** Temperature Sensor
  + **LCD 16x2** (optional for display)
  + Breadboard and jumper wires.
* **Proteus Schematic:**



**Figure 1.12: LM35 Interfacing Proteus**

* **MikroC Code:**



**Figure 1.13: LM35 Interfacing Code**

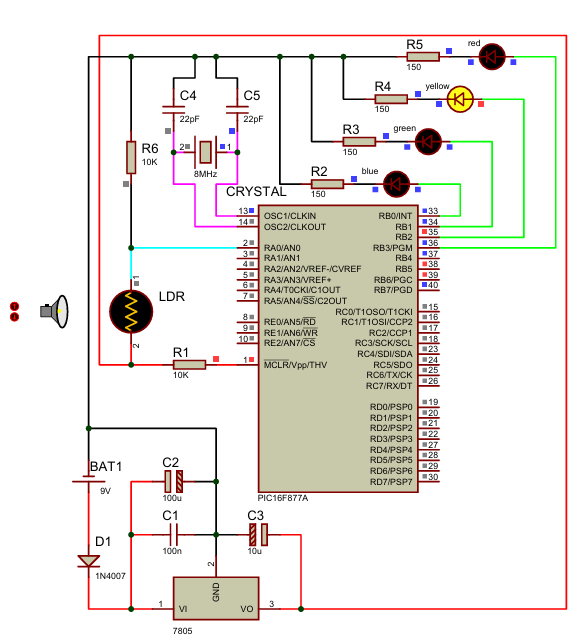
* **Explanation:**
* ADC\_Init(); initializes the Analog-to-Digital Converter in the PIC.
* ADC\_Read(0); reads the analog voltage from channel 0 (RA0), where the LM335 is connected.
* The LM335 outputs voltage proportional to temperature:
* 10mV per Kelvin (e.g. 2.73V at 0°C, 2.98V at 25°C → ~3.03V at 30°C)
* The ADC\_Value (0–1023) represents 0–5V, so:
* ADC\_Value = 1023 ≈ 5V
* ADC\_Value = 600 ≈ 2.93V
* Based on these ranges:
* If ADC value ≥ 760 → High temp → LED1 ON
* If between 500–759 → Medium temp → LED2 ON
* If between 250–499 → Low temp → LED3 ON
* If < 250 → Very low temp → LED4 ON.

## Experiment 7: LDR Interfacing:

* **Objective:**

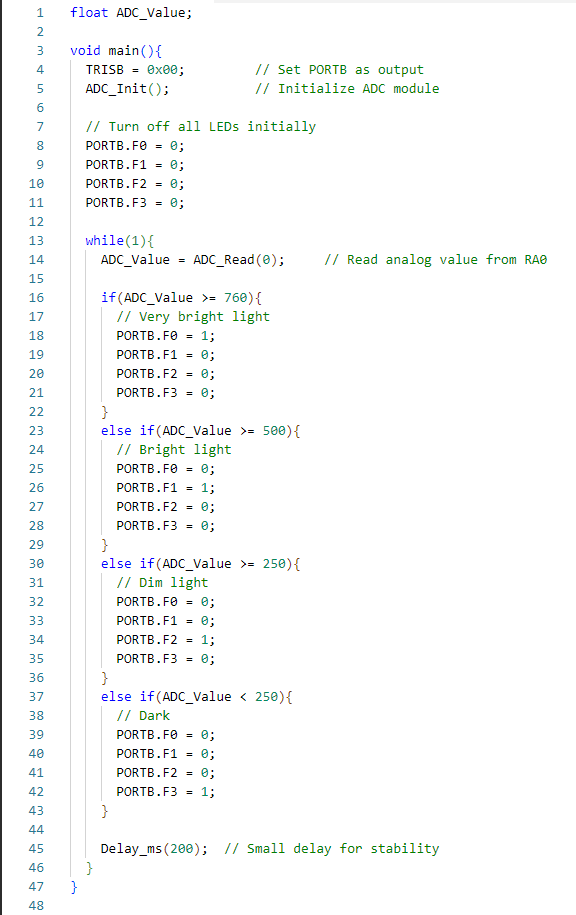
This experiment demonstrates how to interface a Light Dependent Resistor (LDR) with the PIC16F877A microcontroller to detect changes in light intensity. It helps students understand analog sensor integration using ADC.

* **Components Used:**
  + **LDR** (Light Dependent Resistor).
  + **Resistor** (10kΩ).
  + **LCD** (optional for display)
  + Breadboard and wires.
* **Proteus Schematic:**



**Figure 1.14: LDR Interfacing Proteus**

* **MikroC Code:**



**Figure 1.15: LDR Interfacing Code**

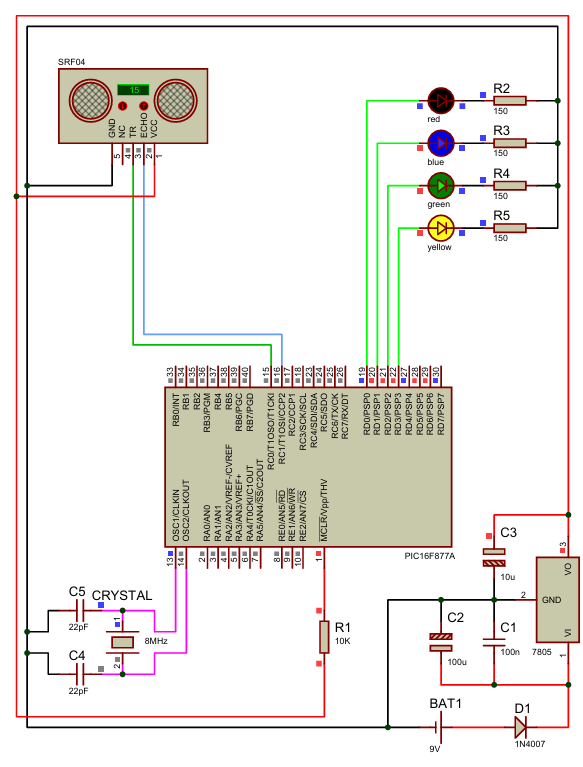
* **Explanation:**
* ADC\_Init(); initializes the ADC module.
* ADC\_Read(0); reads analog voltage from the voltage divider formed by the LDR + 10kΩ resistor.
* The ADC\_Value (range 0–1023) changes depending on light intensity:
* In bright light, voltage ↑ → ADC\_Value ↑
* In darkness, voltage ↓ → ADC\_Value ↓
* The code then:
* Turns ON LED1 if the light is very strong.
* Turns ON LED2 if it's moderately bright.
* Turns ON LED3 if the light is weak.
* Turns ON LED4 if it's dark.

## Experiment 8: Ultrasonic Interfacing:

* **Objective:**

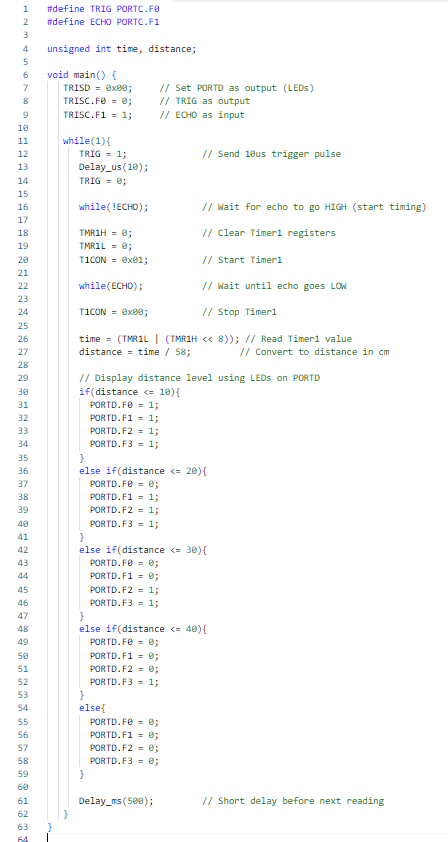
This experiment demonstrates how to interface an ultrasonic sensor with the PIC16F877A microcontroller to measure distances. The ultrasonic sensor emits sound waves, and based on the time taken for the waves to return, the microcontroller calculates the distance.

* **Components Used:**
  + **Ultrasonic Sensor** (HC-SR04)
  + **LCD** (optional for display)
  + Breadboard and jumper wires.
* **Proteus Schematic:**



**Figure 1.16: Ultrasonic Interfacing Proteus**

* **MikroC Code:**



**Figure 1.17: Ultrasonic Interfacing Code**

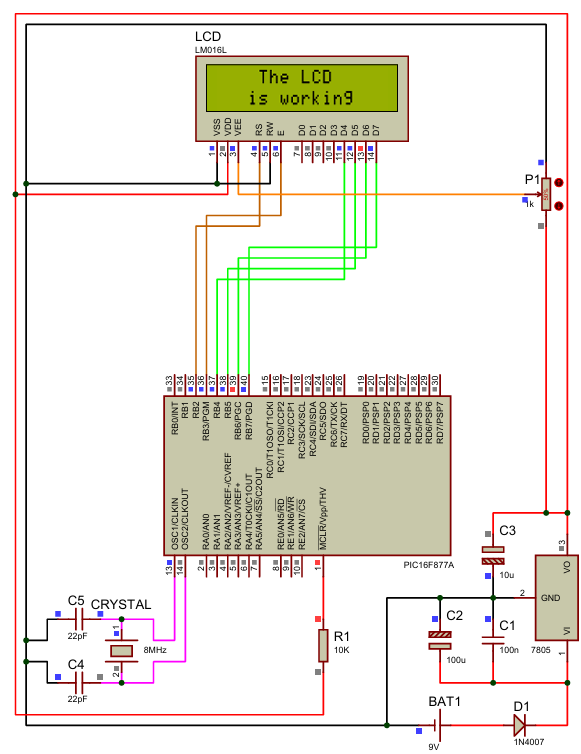
* **Explanation:**
* TRIG Pin (RC0): Sends a 10µs HIGH pulse to start distance measurement.
* ECHO Pin (RC1): Waits for the echo signal to return. The time ECHO remains HIGH is proportional to distance.
* Timer1 is used to measure how long ECHO stays HIGH.
* The distance is calculated by dividing the pulse time by 58 (standard conversion for cm).
* Based on the result:
* If object is very close (≤10cm): All 4 LEDs light up.
* As object moves away: Fewer LEDs are lit.
* If no object within range: All LEDs off.

## Experiment 9: LCD Interfacing (Static Text):

* **Objective:**

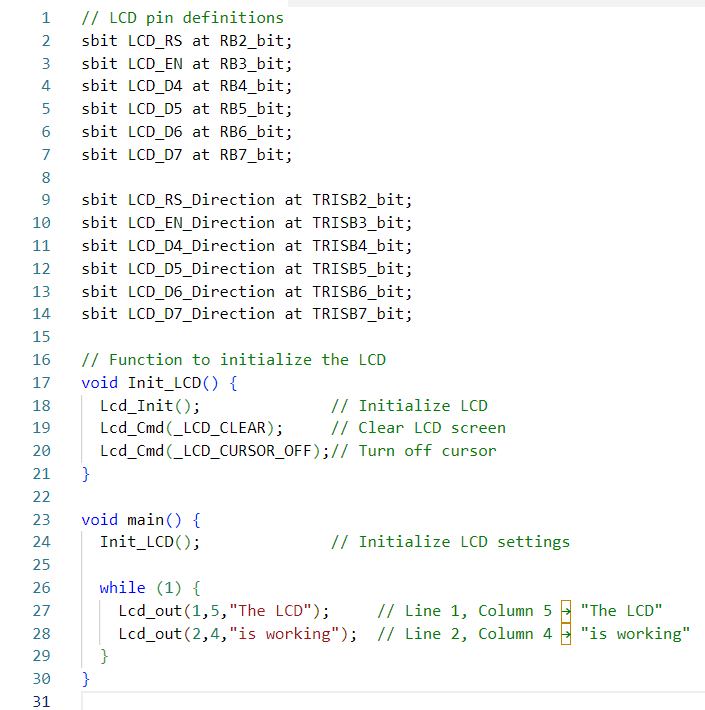
This experiment focuses on displaying static text on an LCD using the PIC16F877A microcontroller. It allows students to learn how to control an LCD and send data to it.

* **Components Used:**
  + **LCD (16x2)** (optional for display)
  + **Resistors** (330Ω).
  + Breadboard and jumper wires.
* **Proteus Schematic:**



**Figure 1.18: LCD Interfacing (Static Text) Proteus**

* **MikroC Code:**



**Figure 1.19: LCD Interfacing (Static Text) Code**

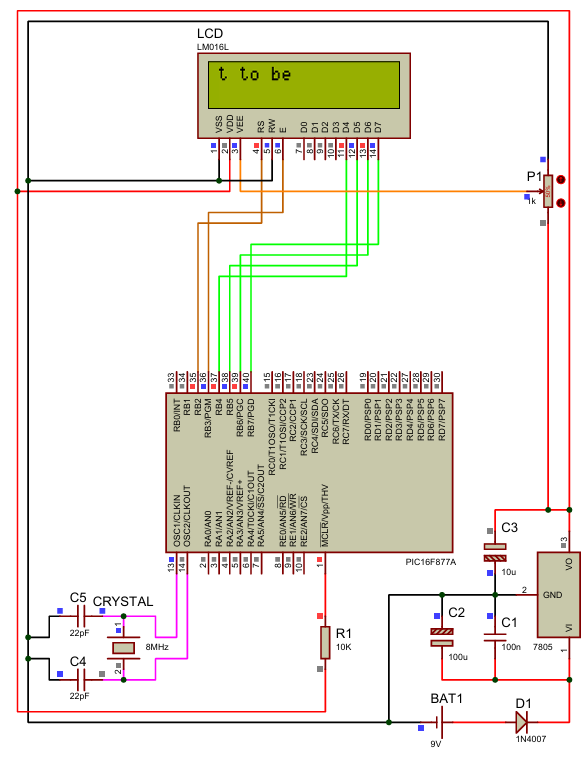
* **Explanation:**
* LCD Pin Configuration: sbit defines the connection between PIC ports (RB2–RB7) and the LCD pins (RS, EN, D4–D7).
* Init\_LCD(): Initializes LCD, clears the screen, hides cursor.
* Lcd\_out(row, column, "text"):
* Displays a string at a specific position.
* Line 1, Column 5 → "The LCD"
* Line 2, Column 4 → "is working".

## Experiment 10: LCD Interfacing (Scrolling Text):

* **Objective:**

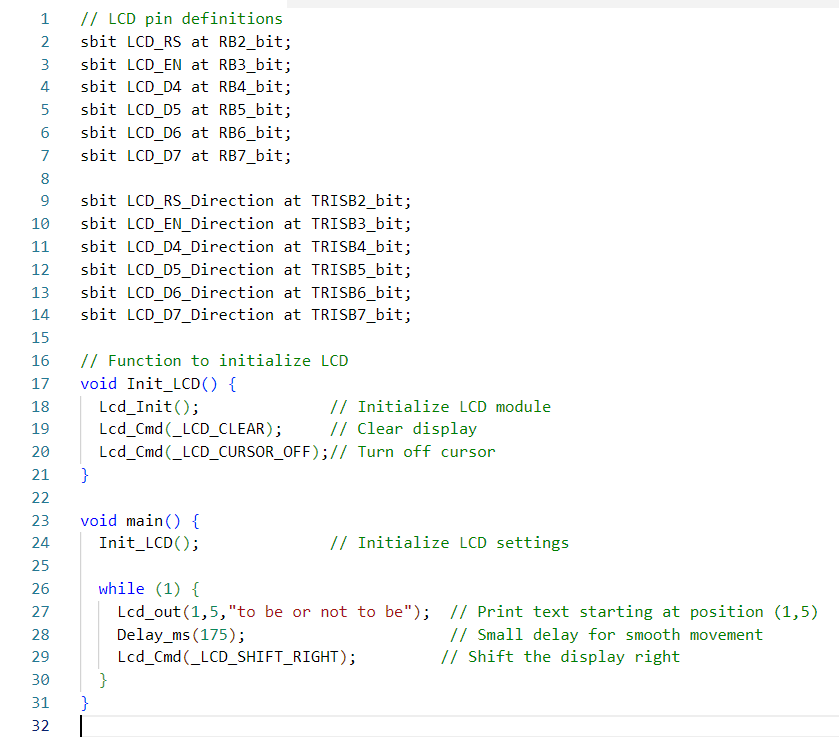
This experiment demonstrates how to display scrolling text on an LCD using the PIC16F877A microcontroller. It allows students to learn how to create dynamic content on the LCD by shifting text from right to left.

* **Components Used:**
  + **LCD (16x2)** (optional for display)
  + **Resistors** (330Ω).
  + Breadboard and jumper wires.
* **Proteus Schematic:**



**Figure 1.20: LCD Interfacing (Scrolling Text) Proteus**

* **MikroC Code:**



**Figure 1.21: LCD Interfacing (Scrolling Text) Code**

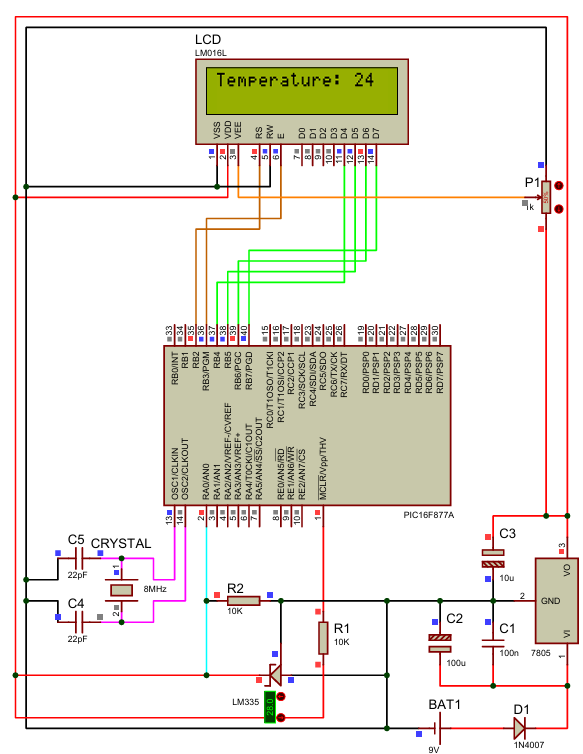
* **Explanation:**
* LCD Pin Definitions: Defines which PIC pins are connected to LCD control/data pins.
* Init\_LCD():
* Initializes the LCD screen
* Clears any existing text
* Turns off the blinking cursor
* Lcd\_out(1,5,"to be or not to be"): Displays the message on row 1, starting from column 5.
* Lcd\_Cmd(\_LCD\_SHIFT\_RIGHT): Shifts the text one position to the right on each loop.
* Delay\_ms(175): Adds a slight delay between each shift so the movement is visible and smooth.

## Experiment 11: Display a Temperature Value:

* **Objective:**

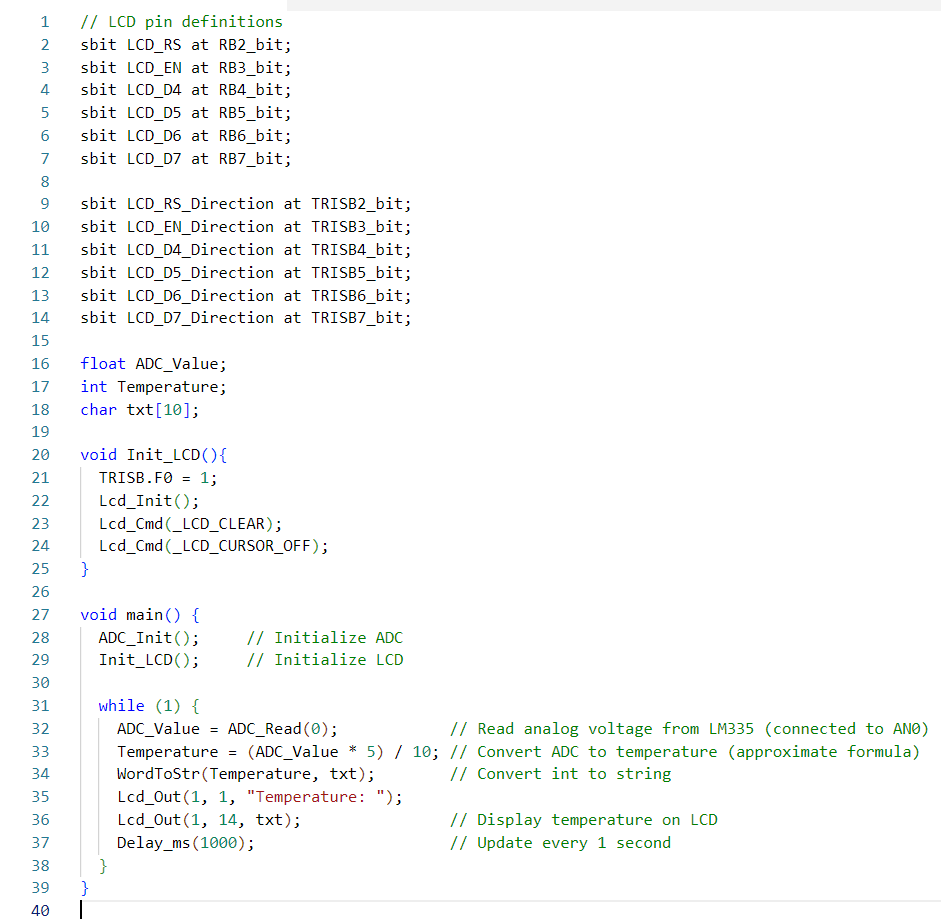
This experiment demonstrates how to read the temperature value from an LM35 temperature sensor and display it on an LCD using the PIC16F877A microcontroller.

* **Components Used:**
  + **LM35** Temperature Sensor
  + **LCD 16x2**
  + **Resistors** (330Ω)
  + Breadboard and jumper wires.
* **Proteus Schematic:**



**Figure 1.22: Display a Temperature Value Proteus**

* **MikroC Code:**



**Figure 1.23: Display a Temperature Value Code**

* **Explanation:**
* ADC\_Read(0);:

Reads the analog voltage from the LM335 connected to AN0 (RA0).

* Temperature = (ADC\_Value \* 5) / 10;:

Converts the ADC value to temperature in Celsius.

LM335 gives 10 mV/°C, and this approximation assumes 5V reference.

* WordToStr(Temperature, txt);:

Converts the temperature integer value into a string to be displayed on the LCD.

* Lcd\_Out(1, 1, "Temperature: ");:

Writes label text on the first line of the LCD.

* Lcd\_Out(1, 14, txt);:

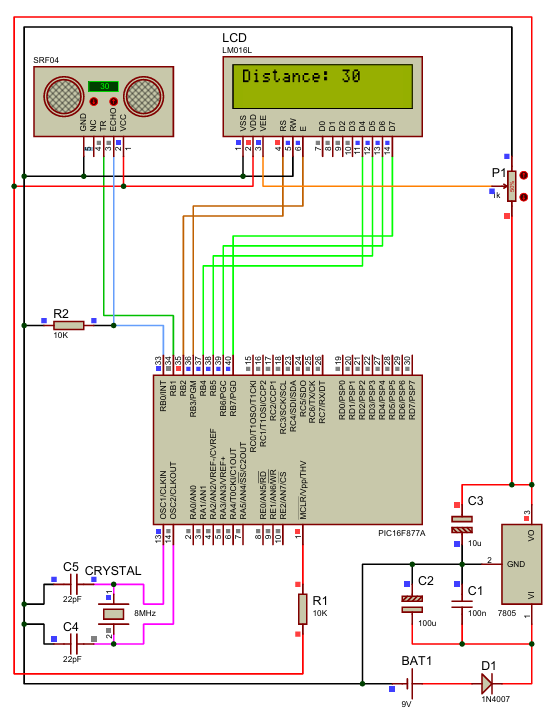
Displays the numeric temperature value next to the label.

## Experiment 12: Display a Distance Value:

* **Objective:**

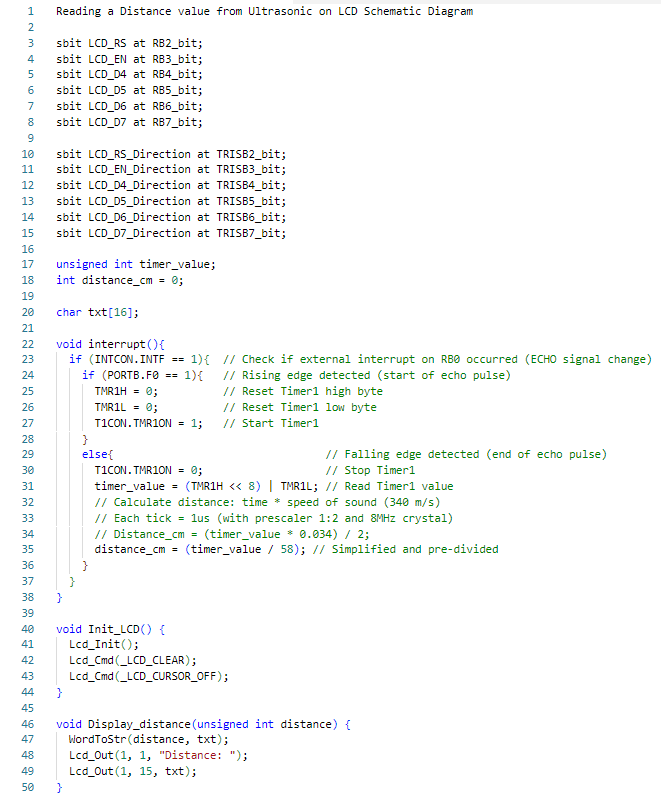
This experiment shows how to measure the distance using an ultrasonic sensor and display the result on an LCD.

* **Components Used:**
  + **Ultrasonic Sensor** (HC-SR04)
  + **LCD 16x2**
  + **Resistors** (330Ω).
  + Breadboard and jumper wires.
* **Proteus Schematic:**



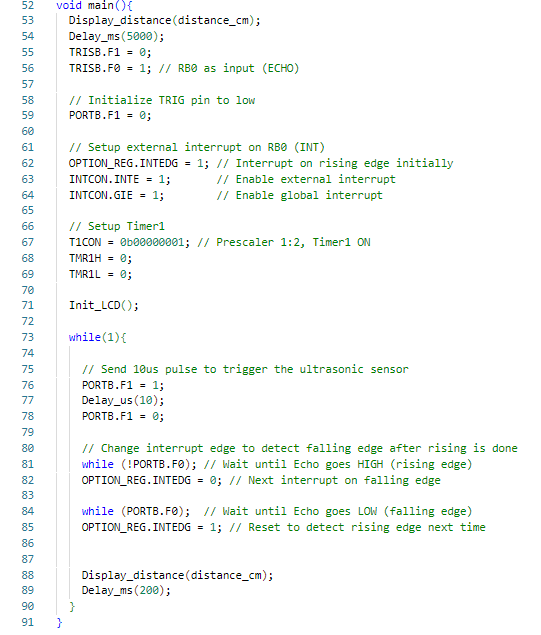
**Figure 1.24: Display a Distance Value Proteus**

* **MikroC Code:**



**Figure 1.25: Display a Distance Value Code**

* **MikroC Code:**



**Figure 1.25: Display a Distance Value Code**

* **Explanation:**
* Interrupt on RB0 (ECHO):

Handles start and stop of echo pulse to measure time.

* Timer1:

Measures the duration of the echo pulse in microseconds.

* Distance Formula:

distance = time / 58; is a simplified conversion for centimeters, assuming sound travels at 340 m/s.

* LCD Output:

Displays "Distance: XX" where XX is the value in centimeters.

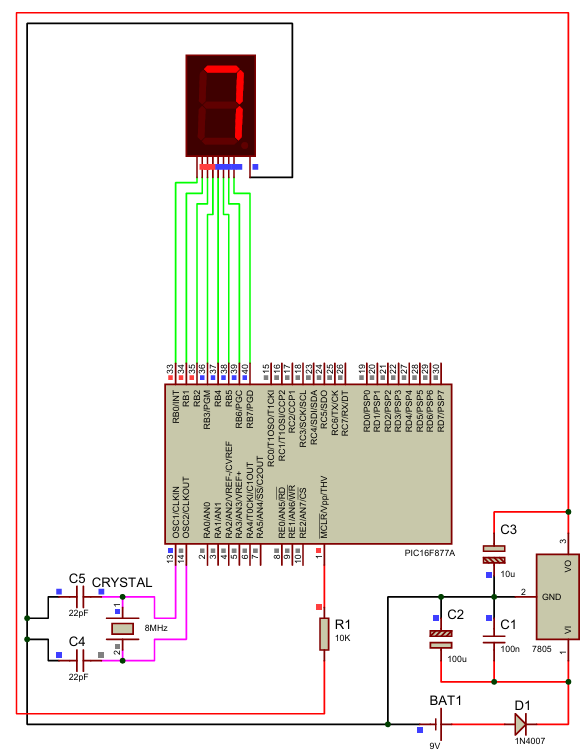
## Experiment 13: 7-Segment Display Interfacing:

* **Objective:**

This experiment demonstrates how to interface a 7-segment display with the PIC16F877A microcontroller. The goal is to display numbers from 0 to 9 on a 7-segment display by controlling each segment using the microcontroller.

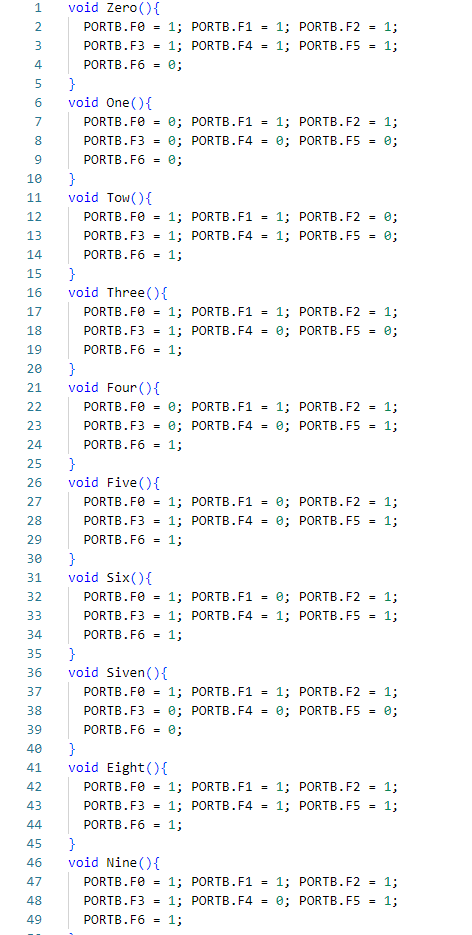
* **Components Used:**
  + **7-Segment Display**
  + **Resistors** (330Ω).
  + Breadboard and jumper wires.

* **Proteus Schematic:**



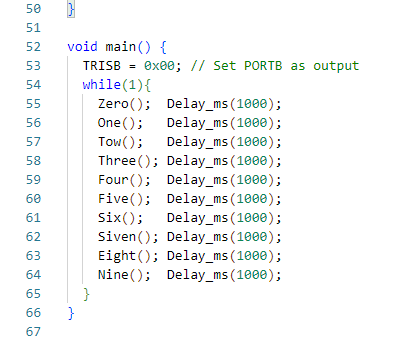
**Figure 1.26: 7-Segment Display Interfacing Proteus**

* **MikroC Code:**



**Figure 1.27: 7-Segment Display Interfacing Code**

* **MikroC Code:**



**Figure 1.27: 7-Segment Display Interfacing Code**

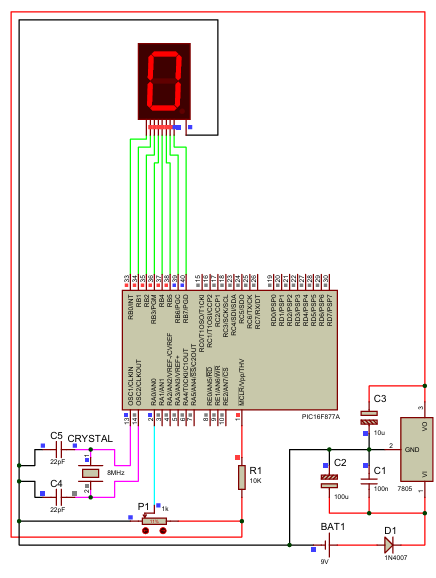
* **Explanation:**
* In this experiment, a single 7-Segment Display is used to show digits 0 through 9. Each segment (labeled A to G) is connected to one pin from PORTB (RB0 to RB6). The display is updated every second by turning ON/OFF the necessary segments to represent each digit.
* The logic HIGH (1) turns ON a segment and logic LOW (0) turns it OFF. Since we use a Common Cathode Display, applying 1 to a pin lights up the corresponding segment.
* The code uses separate functions for each digit to make the logic modular and easy to debug or extend.

## Experiment 14: Controlling a 7-Segment Using Potentiometer:

* **Objective:**

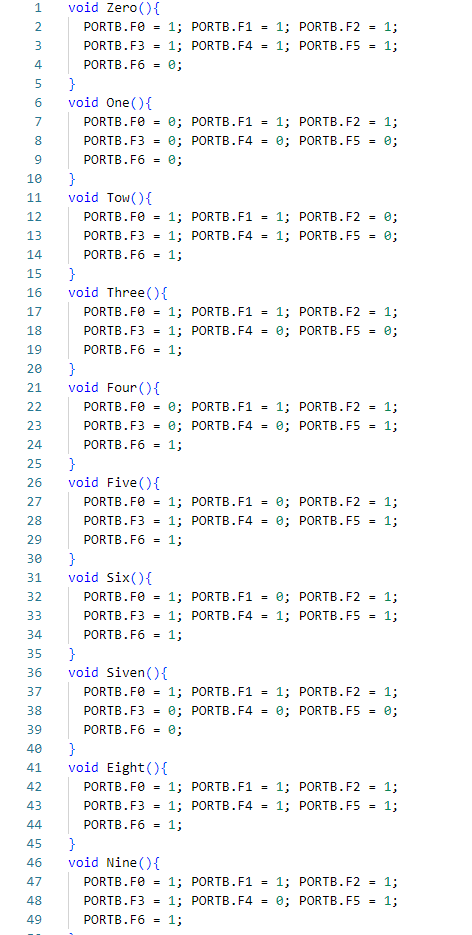
This experiment demonstrates how to control the display of digits on a 7-segment display using a potentiometer as an input device. The potentiometer will determine the displayed number from 0 to 9 based on its position.

* **Components Used:**
  + **7-Segment Display**
  + **Resistors** (330Ω).
  + **Potentiometer**
  + Breadboard and jumper wires.
* **Proteus Schematic:**



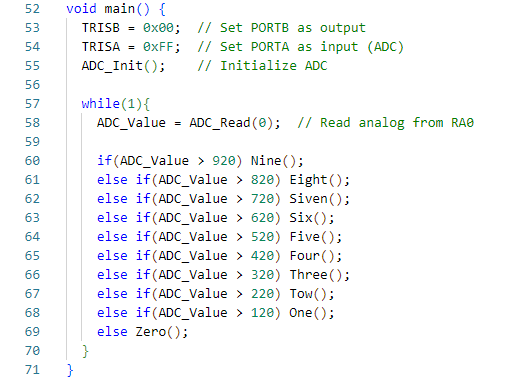
**Figure 1.28: Controlling a 7-Segment Using Potentiometer Proteus**

* **MikroC Code:**



**Figure 1.29: Controlling a 7-Segment Using Potentiometer Code**

* **MikroC Code:**



**Figure 1.27: 7-Segment Display Interfacing Code**

* **Explanation:**
* In this experiment, a potentiometer is connected to the analog pin AN0 (RA0). The ADC module of the PIC16F877A reads the analog voltage and converts it to a digital value (range: 0 to 1023). This value is then mapped to a digit from 0 to 9 and displayed on a 7-segment display.
* The program checks the value of ADC\_Read(0) and shows the corresponding number depending on the range:

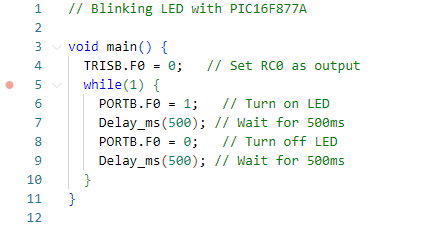
|  |  |
| --- | --- |
| **ADC Value Range** | **Digit Displayed** |
| 0–120 | 0 |
| 121–220 | 1 |
| 221–320 | 2 |
| 321–420 | 3 |
| 421–520 | 4 |
| 521–620 | 5 |
| 621–720 | 6 |
| 721–820 | 7 |
| 821–920 | 8 |
| >920 | 9 |

## Experiment 15: Servo Motor Interfacing:

* **Objective:**

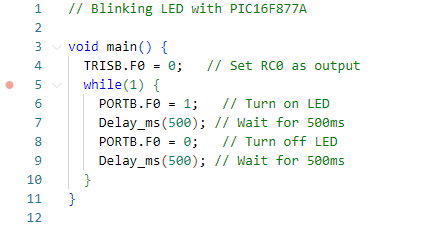
This experiment demonstrates how to interface and control a servo motor using the PIC16F877A microcontroller. The servo motor's position will be controlled by varying the pulse width.

* **Components Used:**
  + **Servo Motor**
  + **Resistors** (330Ω).
  + Breadboard and jumper wires.
* **Proteus Schematic:**



**Figure 1.30: Servo Motor Interfacing Proteus**

* **MikroC Code:**



**Figure 1.31: Servo Motor Interfacing Code**

* **Explanation:**

Explan.

C H A P T E R THREE

# CONCLUSION

we successfully implemented and tested a series of experiments involving the PIC16F877A microcontroller. Each experiment demonstrated a specific function or feature of the microcontroller, such as GPIO control, ADC reading, PWM generation, and communication with external components like LCDs, sensors, motors, and 7-segment displays.

Through these implementations, we gained hands-on experience in embedded system design, circuit interfacing, and MikroC programming. The results of the experiments confirmed the correct behavior of each system and validated the theoretical concepts discussed in earlier chapters.

These implementations also highlighted the flexibility and capabilities of the PIC16F877A microcontroller in real-world applications. The knowledge and skills acquired during this phase will serve as a foundation for developing more complex embedded systems in future chapters or projects.

# REFERENCE

|  |  |
| --- | --- |
| [1] | MikroElektronika, “MikroC PRO for PIC - User Manual,” [Online]. Available: https://download.mikroe.com/documents/compilers/mikroc/pro/pic/mikroc-pro-for-pic-user-manual.pdf. [Accessed: 29 Jun 2024]. |
| [2] | W3Schools, “C Programming Tutorial,” W3schools.com, [Online]. Available: https://www.w3schools.com/c/. [Accessed: 29 Jun 2024]. |
| [3] | Arduino, “Ultrasonic Sensor HC-SR04,” Arduino Project Hub, [Online]. Available: https://create.arduino.cc/projecthub/. [Accessed: 29 Jun 2024]. |
| [4] | Proteus Design Suite, “Proteus VSM for PIC16F877A,” Labcenter Electronics, [Online]. Available: https://www.labcenter.com/. [Accessed: 29 Jun 2024]. |
| [5] | Electronics Hub, “How to Interface Ultrasonic Sensor with PIC Microcontroller,” [Online]. Available: https://www.electronicshub.org/. [Accessed: 29 Jun 2024]. |
| [6] | ElectroSome, “Interfacing LCD with PIC Microcontroller,” ElectroSome.com, [Online]. Available: https://electrosome.com/lcd-pic-mplab-xc8/. [Accessed: 29 Jun 2024]. |
| [7] | Microchip Technology Inc., “PIC16F877A Datasheet,” [Online]. Available: https://ww1.microchip.com/downloads/en/DeviceDoc/39582b.pdf. [Accessed: 29 Jun 2024]. |
|  |  |
|  |  |
|  |  |
|  |  |