1. Introduction

This project involves designing an automated passenger counting system for public transport vehicles, utilizing infrared (IR) sensors and a PIC16F877A microcontroller. The system aims to enhance operational efficiency by automatically counting passengers entering and exiting a vehicle, displaying the count on a seven-segment display, and activating an alarm if passenger limits are exceeded. This document provides an in-depth analysis of the software functionality and system behavior.

2. Functional Description

The passenger counting system operates based on signals from two IR sensors positioned at the entry/exit point of a vehicle. The PIC16F877A microcontroller processes these signals to determine the direction of movement (entry or exit), adjusts the passenger count accordingly, and displays the count on a seven-segment display. An alarm is triggered when the count exceeds 10 or falls below 0.

3. Key Code Sections and Functionality

3.1 Initialization

The system initializes all ports, variables, and interrupts to prepare for real-time operation. Below is a snippet of the initialization code:

```
MAIN:
   CLRF    PORTA
   CLRF    PORTB
   CLRF    PORTD
   CLRF    COUNT
   CLRF    STATE
   CLRF    OLD_PORTB
   ; Enable interrupts
   BSF     INTCON, RBIE
   BSF     INTCON, GIE
```

This code clears all registers and enables interrupts for handling sensor inputs.

3.2 Interrupt Service Routine (ISR)

The ISR handles changes in sensor inputs (PORTB), updates the system state, and triggers appropriate actions based on the sequence of sensor activations. Below is the ISR code:

```
ISR:
   BCF     INTCON, GIE
   BTFSC   INTCON, RBIF
   CALL    HANDLE_PORTB_CHANGE
```

```
  BCF    INTCON, RBIF
  BSF    INTCON, GIE
  RETFIE
```

The ISR ensures real-time processing by responding to sensor input changes and calling appropriate subroutines for handling the logic.

3.3 Passenger Counting Logic

The system uses a state machine to determine whether the passenger is entering or exiting, based on the sequence of sensor triggers. Below is the code for handling the T1 (entry) sensor:

```
CHECK_T1:
  BTFSS   PORTB, 4
  RETURN
  MOVF    STATE, W
  XORLW   STATE_IDLE
  BTFSC   STATUS, Z
    GOTO  SET_RWAIT_T2

  MOVF    STATE, W
  XORLW   STATE_FWAIT_T1
  BTFSC   STATUS, Z
    GOTO  DO_INCREMENT
  RETURN
```

This subroutine checks the T1 sensor and determines whether to increment the count or transition to a waiting state for T2. A similar logic is used for the T2 sensor to handle decrementing.

3.4 Alarm and Buzzer Control

The system activates a buzzer for 5 seconds if the passenger count exceeds 10 or falls below 0. Below is the code for controlling the buzzer:

```
DO_BEEP:
  CALL    BEEP_5SEC
  CLRF    STATE
  ; <<< CLEAR OLD_PORTB so next attempt is recognized >>>
  CLRF    OLD_PORTB
  RETURN
```

```
BEEP_5SEC:
  BSF    PORTA, 2
  MOVLW  d'5'
  MOVWF  loop3
BEEP_LOOP:
  CALL   DELAY_1S
  DECFSZ loop3, F
  GOTO   BEEP_LOOP
  BCF    PORTA, 2
  RETURN
```

This subroutine activates the buzzer and ensures it remains on for 5 seconds before resetting.

4. Testing and Results

The system was tested using simulated scenarios in Proteus to validate the following:

- Passenger count increments correctly for forward motion (entry).

- Passenger count decrements correctly for reverse motion (exit).

- Alarm activates when the count exceeds 10 or falls below 0.

- The seven-segment display accurately represents the current count.

5. Conclusion

The passenger counting system is a reliable and efficient solution for public transport vehicles. It integrates IR sensors, a PIC16F877A microcontroller, and a seven-segment display to automate passenger tracking. The system has been validated through simulation and meets the design objectives. Future enhancements could include support for higher counts, integration with wireless modules for remote monitoring, and real-time clock functionality for timestamping.

6. Proteus Circuit Design

The Proteus circuit design demonstrates the hardware connections and overall system layout used for simulation. Below is the image of the Proteus design:

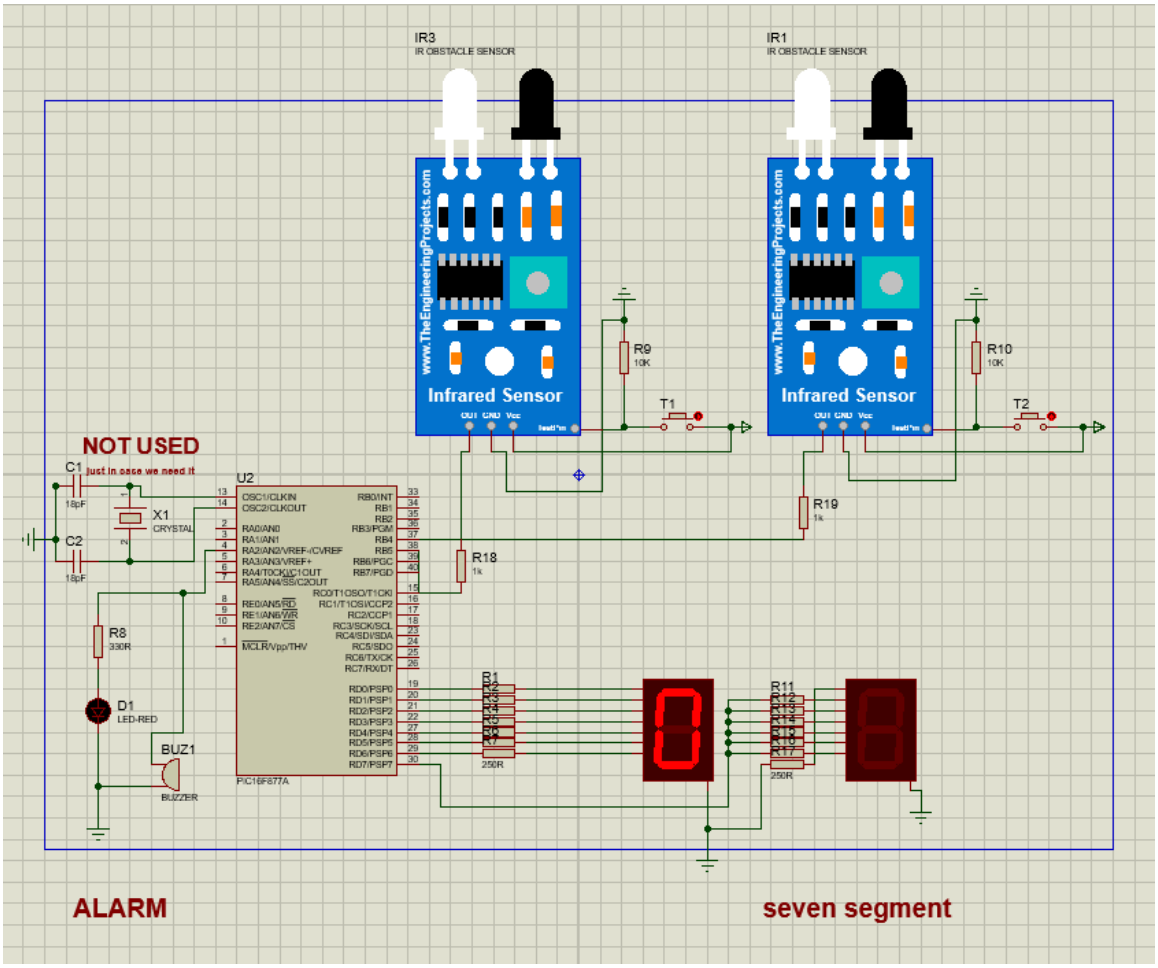

Figure 2: Proteus Circuit Design. This shows the connections between the PIC16F877A microcontroller, IR sensors, seven-segment display, and the buzzer. The design was instrumental in validating the system before hardware implementation.

7. Code Explanation

The complete assembly code provided in the previous section has been written to implement the passenger counting system. Below is a detailed explanation of the purpose and functionality of each key section of the code:

7.1 Initialization

In this section, the microcontroller's ports and variables are initialized. This ensures that all components start in a known state, with PORTA, PORTB, and PORTD cleared to zero. Interrupts are enabled to handle sensor input changes in real time.

7.2 Interrupt Service Routine (ISR)

The ISR is triggered whenever there is a change in the PORTB input caused by the IR sensors. This allows the microcontroller to immediately respond to sensor signals, ensuring accurate detection of passenger movement. The ISR calls subroutines that handle the specific logic for each sensor (T1 or T2).

7.3 State Machine Logic

The state machine manages the sequence of events triggered by the IR sensors. For example, when the T2 sensor is triggered first, the system transitions to a state waiting for T1, indicating a passenger entering. Similarly, when T1 is triggered first, the system transitions to a state waiting for T2, indicating a passenger exiting. This ensures that the count is only incremented or decremented when the complete sequence is detected.

7.4 Alarm System

The alarm system uses a buzzer connected to PORTA. When the passenger count exceeds 10 or falls below 0, the microcontroller activates the buzzer for 5 seconds. This feature ensures that operators are alerted whenever the system operates outside its intended range.

7.5 Display Logic

The seven-segment display is controlled by PORTD. The code includes a subroutine to convert the current passenger count into the appropriate seven-segment pattern. Special handling is included for displaying '10' when the count exceeds a single digit.

8. Detailed Code Breakdown and Explanation

The assembly code for the PIC16F877A microcontroller is designed to manage the counting of passengers in public transport. Each section of the code has a specific purpose and is optimized for real-time performance. Below is a detailed explanation of each section.

8.1 Initialization

This section prepares the microcontroller by clearing all ports, setting initial values for variables, and enabling interrupts for real-time processing. The initialization ensures a predictable starting state for the system.

```
MAIN:
   CLRF    PORTA       ; Clears all pins on PORTA
   CLRF    PORTB       ; Clears all pins on PORTB
   CLRF    PORTD       ; Clears all pins on PORTD
   CLRF    COUNT       ; Initializes the passenger count to zero
   CLRF    STATE       ; Resets the state machine to idle
   CLRF    OLD_PORTB    ; Clears previous readings from PORTB
   ; Enable interrupts
   BSF     INTCON, RBIE  ; Enables interrupt on PORTB change
   BSF     INTCON, GIE   ; Enables global interrupts
```

8.2 Interrupt Service Routine (ISR)

The ISR handles changes in the PORTB input caused by the IR sensors. It is triggered automatically when a sensor signal changes, ensuring real-time detection and handling. The ISR identifies which sensor was triggered and invokes the appropriate handling subroutine.

```
ISR:
  BCF    INTCON, GIE    ; Disables global interrupts during ISR execution
  BTFSC  INTCON, RBIF   ; Checks if PORTB change caused the interrupt
  CALL   HANDLE_PORTB_CHANGE ; Handles the specific PORTB change
  BCF    INTCON, RBIF   ; Clears the interrupt flag for PORTB
  BSF    INTCON, GIE    ; Re-enables global interrupts
  RETFIE               ; Returns from interrupt
```

8.3 State Machine and Counting Logic

The state machine determines the sequence of sensor activations (T1 or T2) to identify whether a passenger is entering or exiting. Based on the sequence, the count is either incremented or decremented.

For example, when T2 is triggered first, the state machine transitions to a state where it waits for T1 to confirm an entry. Similarly, T1 followed by T2 indicates an exit. Below is a snippet of the logic for handling T1:

```
CHECK_T1:
  BTFSS  PORTB, 4      ; Skips if T1 (RB4) is not triggered
  RETURN               ; Returns if T1 is inactive
  MOVF   STATE, W      ; Loads current state
  XORLW  STATE_IDLE    ; Checks if the state is idle
  BTFSC  STATUS, Z     ; Skips if not idle
    GOTO  SET_RWAIT_T2 ; Sets state to wait for T2 after T1
```

8.4 Alarm System

The alarm system activates a buzzer for 5 seconds when the passenger count exceeds the defined limits (0–10). The buzzer is controlled via PORTA (RA2). Below is the code for activating the buzzer:

```
DO_BEEP:
  CALL   BEEP_5SEC     ; Calls subroutine to beep for 5 seconds
  CLRF   STATE         ; Resets the state machine
  CLRF   OLD_PORTB     ; Clears old sensor readings
  RETURN
```

```
BEEP_5SEC:
  BSF    PORTA, 2      ; Turns on the buzzer
  MOVLW  d'5'          ; Sets a 5-second delay
  MOVWF  loop3
BEEP_LOOP:
  CALL   DELAY_1S      ; Calls a 1-second delay subroutine
  DECFSZ loop3, F      ; Decrements the loop counter
  GOTO   BEEP_LOOP     ; Loops until 5 seconds elapse
  BCF    PORTA, 2      ; Turns off the buzzer
  RETURN
```

8.5 Display Logic

The seven-segment display is controlled by PORTD, and the count is converted to a pattern suitable for display. This ensures that the passenger count is always correctly shown. Special handling is included for displaying '10'.

```
UPDATE_DISPLAY:
  MOVF   COUNT, W      ; Loads the current count
  SUBLW  d'10'         ; Checks if the count is 10
  BTFSC  STATUS, Z     ; If count is 10, sets the MSB on PORTD
    BSF    PORTD, 7
  BTFSS  STATUS, Z     ; If not 10, clears the MSB
    BCF    PORTD, 7
  CALL   CONVERT_TO_7SEG ; Converts count to seven-segment pattern
  MOVWF  TEMP          ; Stores the pattern temporarily
```

9. Proteus Circuit Design Explanation

The Proteus simulation plays a crucial role in validating the design and functionality of the passenger counting system before transitioning to physical hardware. Below, the components and their connections in the Proteus design are explained in detail:

9.1 Microcontroller (PIC16F877A)

The PIC16F877A microcontroller is the central unit controlling the system. It processes signals from the sensors, manages the counting logic, and drives the outputs such as the seven-segment display and the buzzer. In the Proteus simulation:

- PORTB (RB4 and RB5) is configured to read signals from the IR sensors.

- PORTD (RD0 to RD6) is connected to the seven-segment display to show the passenger count.

- PORTA (RA2) is connected to a virtual buzzer to simulate the alarm functionality.

9.2 IR Sensors

Two IR sensors are used to detect passenger movement and determine the direction (entry or exit). The sensors are configured in the Proteus simulation as follows:

- The first sensor (T1) is connected to RB4 (PORTB).

- The second sensor (T2) is connected to RB5 (PORTB).

When a passenger interrupts the IR beam, the sensor sends a signal change to the microcontroller, triggering the interrupt service routine to update the count. The placement of the sensors ensures accurate detection of the sequence of movement.

9.3 Seven-Segment Display

The seven-segment display shows the current passenger count in real-time. In the Proteus design:

- PORTD pins (RD0 to RD6) drive the individual segments of the display.

- A specific binary pattern is sent to PORTD to represent each digit (0–9).

- For the count of 10, an additional bit (RD7) is set to indicate the special case, while the segments display 0.

9.4 Buzzer

The buzzer provides an audible alert when the count exceeds 10 or falls below 0. In the Proteus design:

- The buzzer is simulated as an active component connected to RA2 (PORTA).

- The microcontroller activates the buzzer for 5 seconds whenever the count breaches the defined limits.

This helps simulate how the alarm would work in a physical implementation.

9.5 Power Supply

A 5V DC power supply is simulated in Proteus to power the microcontroller and all connected components. In a real-world implementation, a regulated 5V power source is required. This ensures stable operation of all electronic components in the system.