UNIVERSITY OF JORDAN

Computer Engineering Department

Embedded Systems Laboratory Project

Project: Object-Counting System

Spring 2023

Name	ID
Fatima Hesham	**0194433**
Mays AlSeid	**0197655**
Hadeel Adas	**0197144**

Contents

Introduction

In this program we design and implement an embedded system that can accurately count objects in an industrial facility. By utilizing infrared sensors and a microcontroller, the system tracks the movement of objects on a conveyor belt and updates the count accordingly. The program also includes features such as displaying the count on seven-segment displays, activating visual and auditory alerts when count limits are exceeded, and providing a pause switch for temporary counting suspension. The main goal is to eliminate human counting errors and enhance efficiency and monitoring in industrial settings.

System Component

Following are the requirements:

- LED
- Resistances
- Switch
- Oscillator
- 2-PIR motion sensor
- 2-7-segment displays..
- Buzzer
- Microcontroller PIC16F877A
- Capacitors
- Wires

System Description

The embedded system is designed to count objects in industrial facilities using PIR sensors and a PIC16F877A microcontroller. Objects moving on a conveyor belt are detected by the PIR sensors, and the count is incremented or decremented based on the sequence of sensor activations. The count is displayed on two seven-segment displays, and a buzzer is activated for 0.5 seconds whenever the count is updated. Upper and lower count limits (Cmax and Cmin) are set25 and 5, and if the count exceeds Cmax or falls below Cmin, an LED flashes and the buzzer activates for a specified time . The system incorporates a pause switch to temporarily stop counting. Hardware timers ensure accurate timing for count updates and other system operations. Overall,

the system provides automated and reliable object counting in industrial environments, reducing human errors and enabling real-time monitoring.

Subsystems

The **Main** routine is the entry point of the program. It first calls the **INITIAL** subroutine to initialize the microcontroller's configuration and ports. Then, it sets the values of the limit variables (cmin_h, cmin_l, cmax_h, cmax_l). Inside the MAIN_LOOP, timers (TIMER_LED, TIMER_COUNTER, TIMER_7seg) are initialized. Then, the program branches to the Check_Sensors routine.

The **INITIAL** subroutine initializes the microcontroller's configuration. It sets TRISA, TRISD, and TRISB registers to configure the I/O pins. It also clears the PORTA, PORTD, and PORTB registers. Additionally, it initializes the Counter_LOWD and Counter_HIGHD variables to 0. Finally, it sets the necessary interrupt enable flags and configures ADCON1 as Digital.
The program enters the **right_to_left_case** section,The program checks the status of sensor2 by using the BTFSC instruction on PORTA, bit 2. If sensor2 is active (signal detected), the program jumps to the **check_sensor_1** section. Otherwise, it proceeds to the **left_to_right_case.**

in the **check_sensor_1** section The program checks the status of sensor1 by using the BTFSC instruction on PORTA, bit 3. If sensor1 is active, indicating both sensor1 and sensor2 are active, the program jumps to the increment section to increment the count. Otherwise, it returns to the MAIN_LOOP section.
In the **left_to_right_case**,The program checks the status of sensor1 by using the BTFSC instruction on PORTA, bit 3. If sensor1 is active, indicating a signal is detected, the program jumps to the **check_sensor_2 section**. Otherwise, it returns to the MAIN_LOOP section.

In the **check_sensor_2**,The program checks the status of sensor2 by using the BTFSC instruction on PORTA, bit 2. If sensor2 is active, indicating both sensor1 and sensor2 are active, the program jumps to the decrement section to decrement the count. Otherwise, it returns to the MAIN_LOOP section.

In the **increment** section, the program checks if the low digit (Counter_LOWD) is equal to 9 using the SUBLW,If the low digit is 9, the zero flag (Z) will be set,

and the program jumps to the **UPDATE_HIGH_DIGIT** section to increment the high digit (Counter_HIGHD) and reset the low digit. Otherwise, it jumps to the **UPDATE_LOW_DIGIT** section to increment the low digit.
In the **UPDATE_LOW_DIGIT** section, the low digit (Counter_LOWD) is incremented by 1, and the program jumps to the check_bounds section to check if the count exceeds the maximum bounds.
In the **UPDATE_HIGH_DIGIT** section, the low digit (Counter_LOWD) is cleared, and the high digit (Counter_HIGHD) is incremented by 1. Then, the program jumps to the check_bounds section.

In the **decrement** section, the program checks if the low digit (Counter_LOWD) is zero. If the low digit is not zero, the program jumps to the **decrement_low** section to decrement the low digit. Otherwise, it jumps to the **decrement_high** section to decrement the high digit and set the low digit to 9.
In the **decrement_low** section, the low digit (Counter_LOWD) is decremented by 1, and the program jumps to the **check_bounds** section.
in the **decrement_high** section, the high digit (Counter_HIGHD) is decremented by 1, and the low digit (Counter_LOWD) is set to 9. Then,the program jumps to the **check_bounds** section.

The program enters the **check_bounds** section.It moves the value of Counter_HIGHD to the W register and subtracts it from the value of cmax_h then checks the zero flag to determine if the subtraction result is zero, indicating that Counter_HIGHD is equal to cmax_h. If it is zero, the program jumps to the **check_cmax_l** section. Otherwise, it checks the carry flag (C) to determine if the subtraction result is negative, indicating that Counter_HIGHD is greater than cmax_h. If the carry flag is set, the program jumps to the **check_cmin** section. Otherwise, it proceeds to the **DisplayLED** section, indicating that the counter is out of bounds.

In the **check_cmax_l** section, the program moves the value of Counter_LOWD to the W register and subtracts it from the value of cmax_l then checks the carry flag (C) to determine if the subtraction result is negative, indicating that Counter_LOWD is greater than cmax_l. If the carry flag is set, the program jumps to the **check_cmin** section. Otherwise, it proceeds to the DisplayLED section, indicating that the counter is out of bounds.

In the **check_cmin** section, the program checks the value of Counter_HIGHD to see if it is zero by testing the zero flag (Z). If the zero flag is not set, indicating that Counter_HIGHD is not zero, the program jumps to the DisplayCounter section to display the counter value. Otherwise, The program moves the value of Counter_LOWD to the W register and subtracts it from the value of cmin_l then checks the carry flag (C) to determine if the subtraction result is negative, indicating that Counter_LOWD is greater than cmin_l. If the carry flag is set, the program jumps to the DisplayLED section, indicating that the counter is out of bounds. Otherwise, it jumps to the DisplayCounter section to display the counter value.

In the **DisplayCounter** section ,have a REPEAT label which responsible for display the high & low digits on 7-segment display multiplexing-more explaining about it next paragraph- .In REPEAT ,the program moves the value of Counter_LOWD to the W register then call a Look_TABLE subroutine which contain a binary values for represent digits (0-9) on anode 7-segment display .To display the value the next instruction enable LSD 7_segment display ,Call DELAY_ subroutine (50 ms hardware delay by TMR0) twice time , disable LSD 7_segment display .
Also in REPEAT label apply same explaining above for Counter_HIGHD to display the value on 7-segment MSD but here set and clear PORTA bit1 which is responsible about disable and enable the 7-segment MSD. To display the value enough to be seen by the eye the REPEAT label have a loop to repeat the execution of it by DECFSZ instruction that decrement the TIMER_7seg register which initially have 10 decimal and check if is not zero go to REPEAT label else that save 10 decimal in TIMER_7seg register so, after 10 rounds the counter values display for 2 seconds .To turn on the buzzer ,set the PORTB bit1 follows it repeat_buzzer label which have a loop to turn on the buzzer 0.5 seconds that call DELAY_ subroutine (50 ms hardware delay by TMR0) ,then by DECFSZ instruction that decrement the TIMER_7seg register which initially have 10 decimal and check if is not zero go to repeat_buzzer label else that turn off the buzzer that make the buzzer turn on 0.5 seconds after 9 rounds finally goto MAIN_LOOP to check again the objects.

Multiplexing of two 7-segment displays is a technique used to control and display information on both displays using a single set of microcontroller pins.

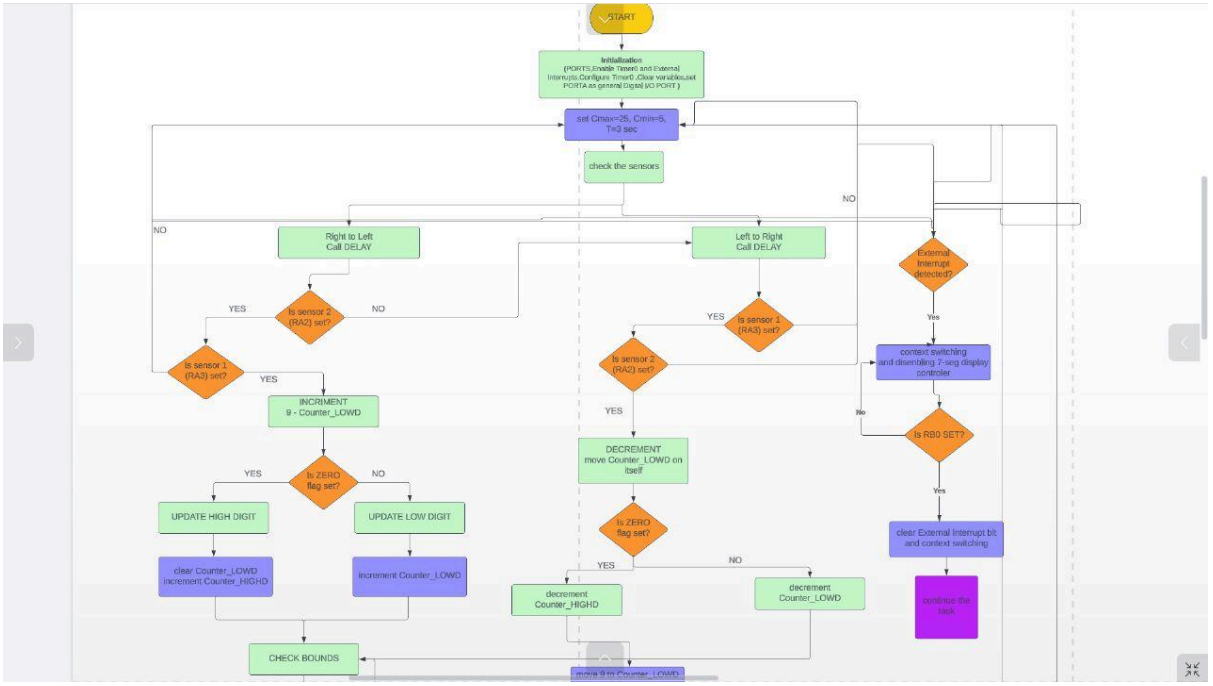
It involves rapidly switching between the two displays in a synchronized manner.
By selectively activating one display at a time while quickly cycling through them, the illusion of both displays being active simultaneously is created. This technique reduces the number of required pins and allows for efficient control of multiple displays, typically through a combination of transistors and time-division multiplexing.

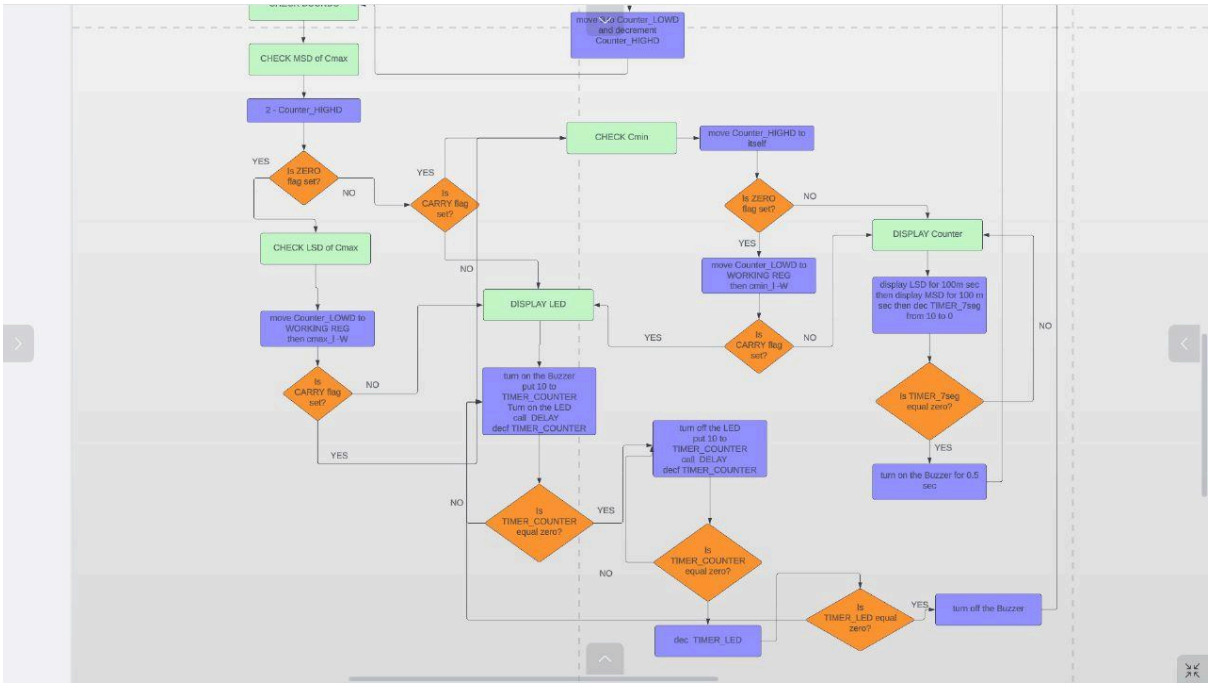
In the **DisplayLED** section turn on the buzzer and flash the LED 3 seconds ,to turn on the buzzer ,set the RB1 follows it LED_FLASH which has two labels LED_ON label which turn on the LED for 0.5 seconds by set the RB5 ,call delay then have loop by DECFSZ instruction that decrement the TIMER_COUNTER register which initially have 10 decimal and check if is not zero go to LED_ON label that make the LED turn on 0.5 seconds after 10 rounds else save 10 decimal in TIMER_COUNTER register which follows LED_OFF label which turn off the LED for 0.5 seconds by clear the PORTB bit5 ,call delay then have loop by DECFSZ instruction that decrement the TIMER_COUNTER register which and save the result in same register and check if is not zero go to LED_OFF label that make the LED turn off 0.5 seconds after 9 rounds else by DECFSZ instruction decrement the TIMER_LED register which initially have 3 decimal and check if is not zero go to LED_FLASH repeat the LED_FLASH 3 times to flash LED 3 seconds else that turn off the buzzer by clear PORTB bit 1 ,finally goto MAIN_LOOP to check again the objects.

IN the **DELAY** subroutine it is 50 ms hardware delay by TMR0,select TMR0 by BANKSEL instruction ,after make some calculations with 4 MHZ FOSC we find that N=195,and prescaler =256 ,So move (256-N → 195) to TMR0 , select OPTION_REG to set up TMR0 for internal clock(prescale by 256) move B'00000111' to OPTION_REG.Testing Timer Overflow flag ,test bit T0IF if it clear (no timer overflow) goto L1 loop else clear Timer Overflow flag bit T0IF then return .

The **ISR** make context switching for Counter_LOWD and Counter_HIGHD ,then disable MSD and LSD 7_segment display ,check RB0 (External Interrupt) if it clear goto PASS **Label** else clear flag INIF, context switching again and return from interrupt

Flow Chart :


START
Initialization
(PORTS,Enable Timer0 and External Interrupts,Configure Timer0 ,Clear variables,set PORTA as general Digital I/O PORT)
set Cmax=25, Cmin=5, T=3 sec
check the sensors
NO
Right to Left
Call DELAY
Left to Right
Call DELAY
External Interrupt detected?
YES
Is sensor 2 (RA2) set?
NO
YES
Is sensor 1 (RA3) set?
YES
Is sensor 1 (RA3) set?
YES
Is sensor 2 (RA2) set?
YES
context switching and disenbling 7-seg display controller
INCREMENT 9 - Counter_LOWD
DECREMENT move Counter_LOWD on itself
No
Is RB0 SET?
Yes
Is ZERO flag set?
YES
NO
Is ZERO flag set?
YES
NO
clear External Interrupt bit and context switching
UPDATE HIGH DIGIT
UPDATE LOW DIGIT
clear Counter_LOWD increment Counter_HIGHD
increment Counter_LOWD
decrement Counter_HIGHD
decrement Counter_LOWD
continue the task
CHECK BOUNDS


move Counter_LOWD and decrement Counter_HIGHD
CHECK MSD of Cmax
2 - Counter_HIGHD
CHECK Cmin
move Counter_HIGHD to itself
YES
Is ZERO flag set?
NO
YES
Is CARRY flag set?
Is ZERO flag set?
NO
DISPLAY Counter
CHECK LSD of Cmax
NO
YES
move Counter_LOWD to WORKING REG then cmin_I -W
display LSD for 100m sec then display MSD for 100 m sec then dec TIMER_7seg from 10 to 0
DISPLAY LED
move Counter_LOWD to WORKING REG then cmax_I -W
YES
Is CARRY flag set?
NO
NO
Is CARRY flag set?
turn on the Buzzer put 10 to TIMER_COUNTER Turn on the LED call DELAY decf TIMER_COUNTER
Is TIMER_7seg equal zero?
NO
YES
YES
turn off the LED put 10 to TIMER_COUNTER call DELAY decf TIMER_COUNTER
turn on the Buzzer for 0.5 sec
NO
Is TIMER_COUNTER equal zero?
YES
Is TIMER_COUNTER equal zero?
NO
YES
Is TIMER_LED equal zero?
turn off the Buzzer
dec TIMER_LED

Hardware System:

The system aims to accurately count objects moving on a conveyor belt in both directions and display the count on the seven-segment displays. Additionally, it provides visual and auditory alerts when the count exceeds certain limits.Also,that have a pause switch when the switch is closed, the system stops counting regardless of the number of objects that pass by. Otherwise, the system operates normally when the switch is open.

Inputs:

1-Digital Infrared Sensors (t1 and t2): These sensors consist of digital output IR sensors. When an object blocks the infrared beam, the digital sensor outputs a logic level change, indicating the presence of an object.Configured on PORTA bit 2 & 3 after configure PORTA as a digital which is connected to a pull-up resistor.
2-Pause Switch: The pause switch is an input that allows users to control the counting operation.

Outputs:

1-Seven-Segment Displays:Configured on PORTD (0-6) bits as output with 2 enable bits on PORTA bit 0&1 .They are common Anode Seven-Segment Displays.Initially they are clear, they set when meet conditions.
2-LED: The LED is activated when the count value exceeds Cmax or falls below Cmin. It flashes for 3 seconds to provide a visual alert.Configured on PORTB bit5 that connected to a pull-up resistor.initially is off.

3-Buzzer: The buzzer is activated for a duration of 3 seconds when the count value exceeds Cmax or falls below Cmin.Also,is activated for a duration of 0.5 whenever the count value is updated.It provides an audible alert.Configured on PORTB bit4 ,it is connected to a pull-up resistor.

The chosen hardware design and its configuration were determined after considering the specific requirements of the industrial facility counting system.

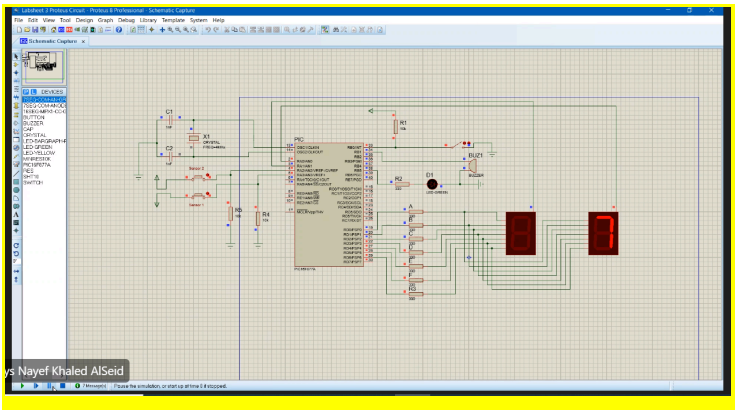
The selected components and their interconnections enable accurate object detection, reliable counting, clear display, and timely alerts. The use of digital infrared sensors simplifies the sensing process, while the incorporation of hardware timers and multiplexing optimizes performance and efficiency. The pause switch allows users to control the counting operation conveniently. Overall, the design aims to provide a robust, user-friendly, and effective solution for counting objects in an industrial environment.

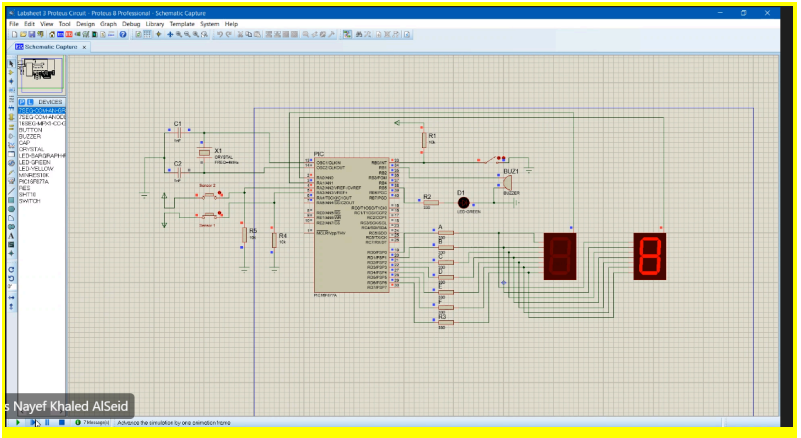
System Testing and Results:

To test our project we test many cases to ensure that cover all possible regions of operation of the system.

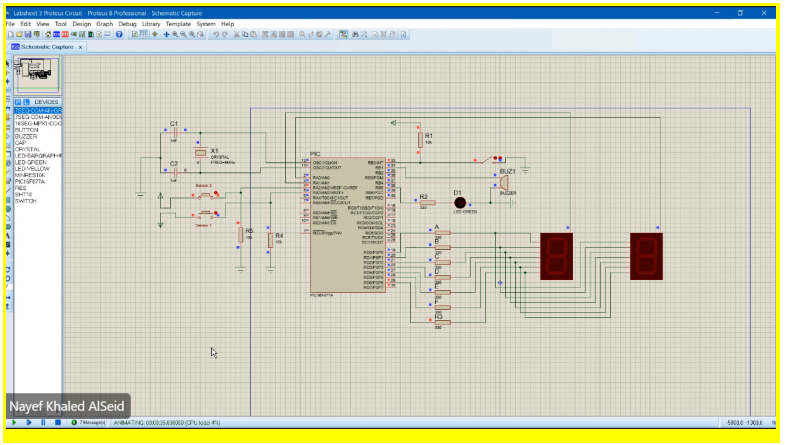
A)Puase switch is open:

1) Object entering from right to left (considered as forward motion) ,counter value in between Cmax & Cmin . In this case, the count value is incremented by 1 when the object blocks the t2 sensor then the t1 sensor. Display the value on 7 Segment and turn on the Buzzer foe 0.5s.

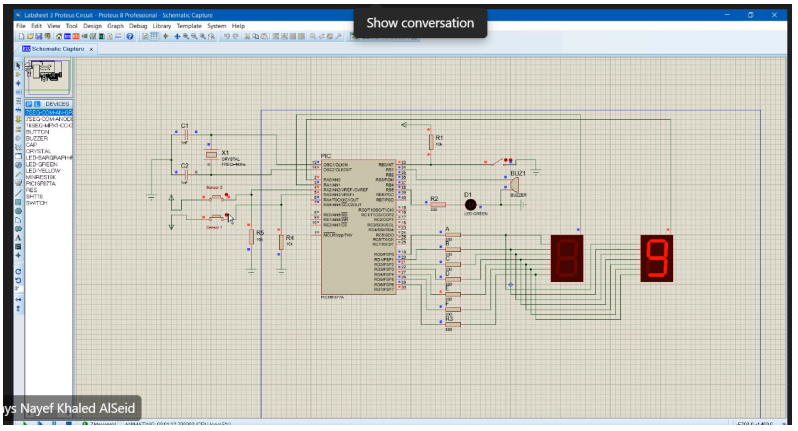


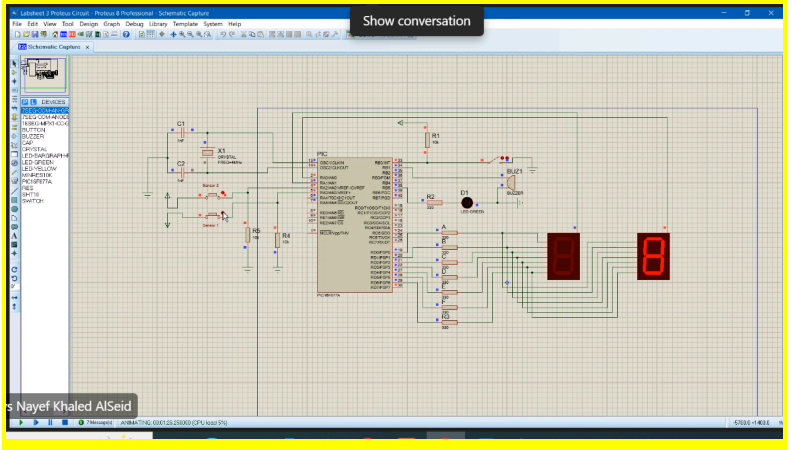


2)Object entering from right to left (considered as forward motion) ,counter value in between Cmax & Cmin . In this case, The object is not counted If blocks the t2 sensor only.

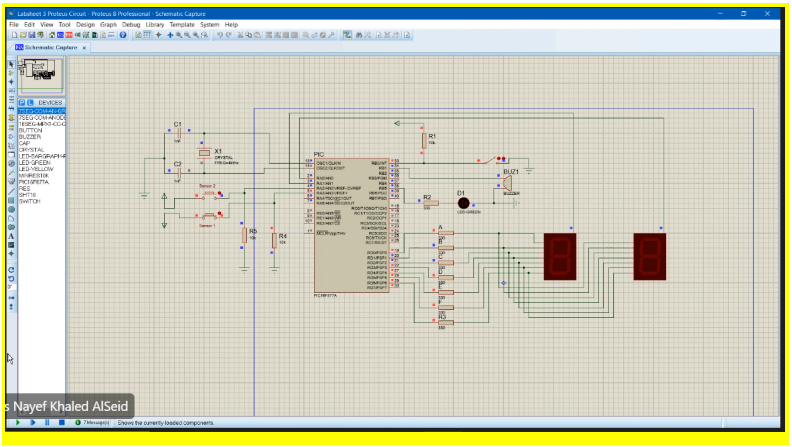


3) Object entering from left to right (considered as reverse motion),counter value in between Cmax & Cmin . In this case, the count value is decremented by 1 when the object blocks the t1 sensor then the t2 sensor.Display the value on 7 Segment and turn on the Buzzer foe 0.5s.





4) Object entering from left to right (considered as reverse motion),counter value in between Cmax & Cmin . In this case, The object is not counted If blocks the t1 sensor only.



5) Object entering from right to left (considered as forward motion) ,counter value greater than Cmax . In this case, the object blocks the t2 sensor then the t1 sensor.The object is not counted .Flash the Led and turn on the Buzzer for 3s.

Showen in attached video.

6) Object entering from right to left (considered as forward motion) ,counter value smaleer than Cmin . In this case, the object blocks the t2 sensor then the t1 sensor.The object is not counted .Flash the Led and turn on the Buzzer for 3s.

Showen in attached video.

7) Object entering from left to right (considered as reverse motion),counter value greater than Cmax . In this case, the the object blocks the t1 sensor then the t2 sensor.The object is not counted .Flash the Led and turn on the Buzzer for 3s.

Showen in attached video.

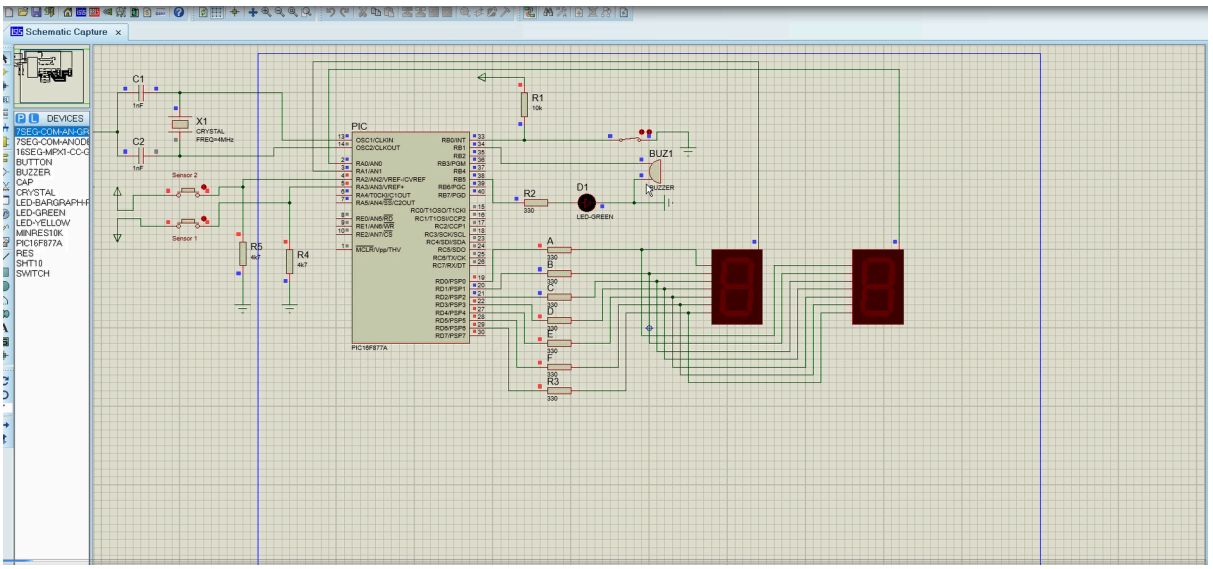
9) Object entering from left to right (considered as reverse motion),counter value smaller than Cmin . In this case, the the object blocks the t1 sensor then

the t2 sensor.The object is not counted .Flash the Led and turn on the Buzzer for 3s.

Showen in attached video.

B)Puase switch is close:

The system stops counting regardless of the number of objects that pass by in all cases .



more showen in attached video.

Video :
https://drive.google.com/file/d/1HQmXKAdNUlayHcdvtp-shrZYeUyldKCl/view?usp=sharing

Conclusion :

In this project, an Object-Counting System was made for the Embedded Systems Lab.We wite a program that can accurately count objects in an industrial facility. As long as switch is open and the counter in between bounds.The systems simulation works as expected in Proteus, 7-segment displays,Led and Buzzer are working fine.

Project is aim to start working within a team of 3 students and dividing the work between us. And make sure that we have the basic skills to write code in the Assembly language using a PIC16F877A® microcontroller.In addition to our ability to write the code using the MBLAB software,design the hardware using proteus, and check the work of each part.

Contributions of Each Student:

*Fatima Hesham

DisplayLED,DisplayCounter

*Mays AlSeid

Check_bounds,Proteus

*Hadeel Adas

Check_Sensors

Work sharing:

DELAY,ISR,Report,Hardware

Obstacles Faced:

• Software obstacles: One of the problems we faced is about ISR code ,firstly when we write the program it does not continue counting correctly after return

from interrupt ,after thinking we made a context switching for counter values and disable the 7 segment and it solved.

• Hardware obstacles: The problem we are facing with our hardware is selecting enough value of timer0 to show counter values on 7-segment display multiplexing.

References

- **PIC16F877A datasheet.**
- **Embedded lab experiments**