Embedded Project

Name: Ziad Aymen Abdallah Ahmed .

Level : 2.

Major : Communication and computer department

SEC : 1.

Table of Contents

Task1:

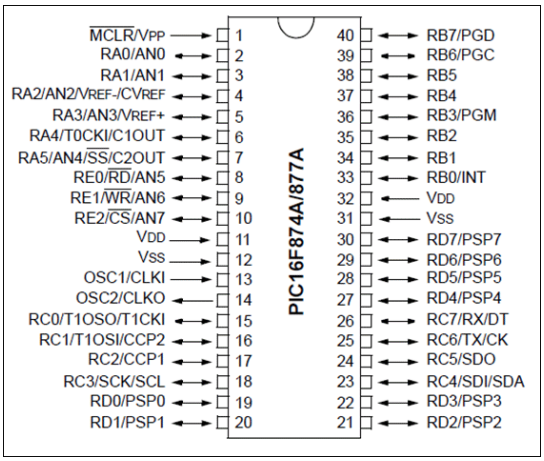
A) Description of microcontroller (16F877A) pins:

The PIC16F877A microcontroller comes in a 40-pin DIP package. Here is a description of each pin:

Power Supply Pins

1. VDD (Pin 11 and Pin 32): Positive supply voltage. Typically, +5V.

2. VSS (Pin 12 and Pin 31): Ground reference voltage.



Oscillator Pins

3. OSC1/CLKIN (Pin 13): Oscillator crystal or external clock input.

4. OSC2/CLKOUT (Pin 14): Oscillator crystal or clock output. When used with a

crystal, this pin connects to the other terminal of the crystal.

Reset Pin

5. MCLR (Pin 1): Master Clear (Reset) input. An active-low pin used to reset the microcontroller.

Port A Pins (RA0-RA7)

6. RA0/AN0 (Pin 2): Analog input channel 0.

7. RA1/AN1 (Pin 3): Analog input channel 1.

8. RA2/AN2/VREF- (Pin 4): Analog input channel 2 or negative voltage reference input.

9. RA3/AN3/VREF+ (Pin 5): Analog input channel 3 or positive voltage reference input.

10. RA4/T0CKI (Pin 6): Digital I/O or Timer0 external clock input.

11. RA5/AN4/SS (Pin 7): Analog input channel 4 or Slave Select for SPI communication.

Port B Pins (RB0-RB7)

12. RB0/INT (Pin 33): Digital I/O or external interrupt input.

13. RB1,2,4,5 (Pin 34,35,37,38): Digital I/O.

15. RB3/PGM (Pin 36): Digital I/O or low voltage programming input.

18. RB6/PGC (Pin 39): Digital I/O or In-Circuit Debugging clock.

19. RB7/PGD (Pin 40): Digital I/O or In-Circuit Debugging data.

Port C Pins (RC0-RC7)

20. RC0/T1OSO/T1CKI (Pin 15): Digital I/O, Timer1 oscillator output, or Timer1 clock

input.

21. RC1/T1OSI/CCP2 (Pin 16): Digital I/O, Timer1 oscillator input, or

Capture/Compare/PWM module 2.

22. RC2/CCP1 (Pin 17): Digital I/O or Capture/Compare/PWM module 1.

23. RC3/SCK/SCL (Pin 18): Digital I/O, SPI clock, or I2C clock.

24. RC4/SDI/SDA (Pin 23): Digital I/O, SPI data input, or I2C data.

25. RC5/SDO (Pin 24): Digital I/O or SPI data output.

26. RC6/TX/CK (Pin 25): Digital I/O or USART transmit.

27. RC7/RX/DT (Pin 26): Digital I/O or USART receive.

Port D Pins (RD0-RD7)

28. RD0/PSP0 (Pin 19,20,21,22,27,28,29,30): Digital I/O or Parallel Slave Port data bit 0.

Port E Pins (RE0-RE2)

36. RE0/RD/AN5 (Pin 8): Digital I/O, Read control for Parallel Slave Port, or Analog input

channel 5.

37. RE1/WR/AN6 (Pin 9): Digital I/O, Write control for Parallel Slave Port, or Analog input

channel 6.

38. RE2/CS/AN7 (Pin 10): Digital I/O, Chip Select for Parallel Slave Port, or Analog input channel 7.

B) Explaining the functions of the main blocks in PIC16f877A:

1- Arithmetic Logic Unit (ALU): performs arithmetic (addition, subtraction, multiplication, ..) and logical (AND, OR, XOR, NOT, ..) operations on data, acting as the core computational engine.

2- Status and Control: These registers hold the status of the ALU operations and control the operation of the microcontroller.

Status Register: Contains flags that indicate the result of the last ALU operation (e.g., Zero, Carry, Digit Carry, and Overflow flags).
Control Registers: Control various aspects of the microcontroller's operation, such as enabling/disabling interrupts, configuring I/O ports, and setting operating modes.

3- Program counter (PC): The Program Counter holds the address of the next instruction to be executed.

The PC is incremented automatically after fetching an instruction, ensuring the sequential execution of instructions.

4- Flash Program Memory: This is non-volatile memory that stores the program code, it retains the program even when the power is turned off, allowing the microcontroller to retain its functionality upon power-up.

The PIC16F877A has 14-bit wide Flash memory for storing instructions

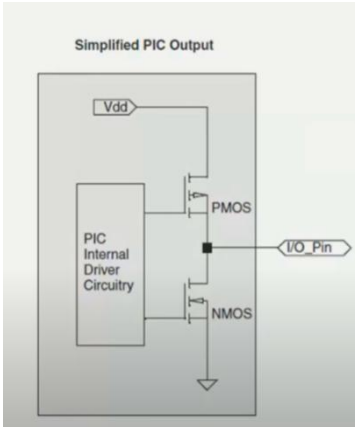
And size 8KB.

5- Instruction Register: Temporarily holds the current instruction fetched from the program memory, it receives the 14-bit instruction code from the program memory.

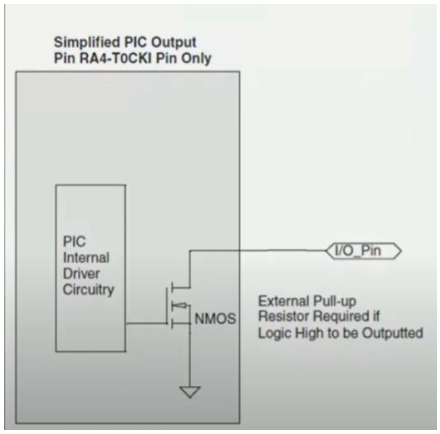
6- Instruction Decoder: Decodes the instruction fetched into the instruction register and generates the necessary control signals to execute the instruction.

C) Why led connected in RA4 for flashing purpose not working:

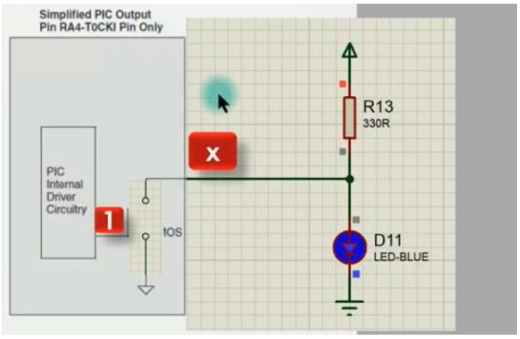
- Configuration of pins of port A and port E except RA4 PMOS will work when the out of pic is one because PMOS connect with Vdd(5v) and NMOS does not work, NMOS will work when the out of pic is zero because NMOS connect with ground and PMOS does not work When we connect led as a source or sink will work



- Configuration of RA4: pin PMOS is not here NMOS will work when the out of pic is zero and the out of pin become zero because NMOS connect with ground When we connect led as a source led does not work (no flashing) because when the out of pic is one NMOS does not work and when the out of pic is zero the NMOS will work and the voltage of cathode=voltage of anode = zero When we connect led as a sink led will work (flashing) because when the out of pic is one NMOS does not work and the (open circuit) and when the out of pic is zero the NMOS will work and the voltage of cathode.



- We can make led work when it is connected as a source by connecting the led with ground and connecting the resistor with power as the following:



D) ATmega328P vs PIC16F877A Comparison Table:

Feature	**ATmega328P**	**PIC16F877A**
Architecture	AVR (RISC)	PIC (RISC)
Word Size	8-bit	8-bit
Flash Memory (Program)	32 KB	14 KB
SRAM (Data Memory)	2 KB	368 Bytes
EEPROM	1 KB	256 Bytes
CPU Speed	Up to 20 MHz	Up to 20 MHz
I/O Pins	23	33
ADC	10-bit, 6/8 channels	10-bit, 14 channels
PWM Channels	6	2
Timers	3 (2×8-bit, 1×16-bit)	3 (2×8-bit, 1×16-bit)
USART	1	1
Operating Voltage	1.8V – 5.5V	2V – 5.5V
Power Consumption	~0.2 mA ,1 MHz, 1.8V	~1.5 mA ,1 MHz, 2V
Package Options	DIP-28, TQFP-32, MLF-32	DIP-40, PLCC-44, QFP-44
Development Tools	Arduino IDE, Atmel Studio	MPLAB IDE

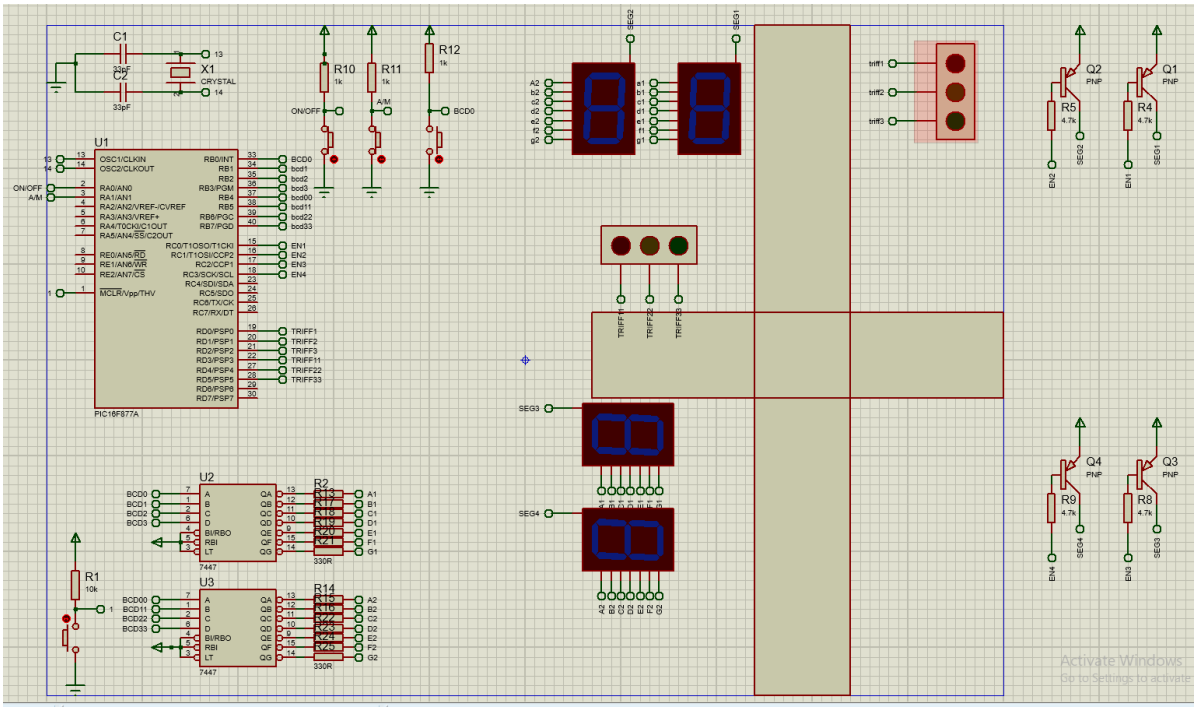
Where ATmega328P is a better choice:

1. Arduino-based Prototyping and DIY Projects
    - Since ATmega328P is the main microcontroller in Arduino Uno, it has better support for rapid prototyping, a large user community, and is easier to program using high-level libraries.
2. Low-Power Battery-Operated Devices
    - With its lower power consumption, sleep modes, and better power efficiency, ATmega328P is a better choice for devices like:
        - Remote environmental sensors
        - Wearables or fitness trackers

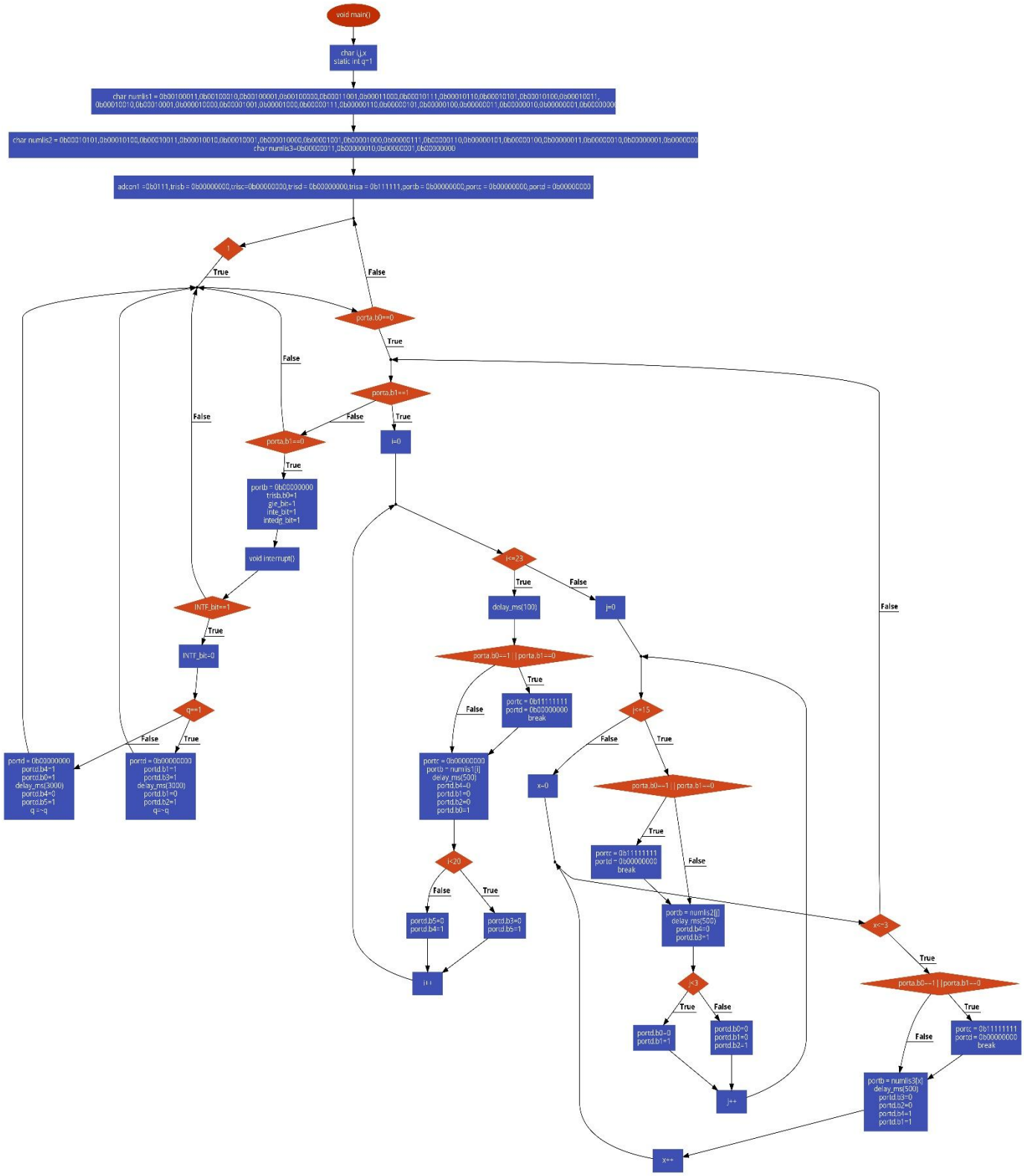
Task2:

A) Circuit:

1. Objective: Develop a traffic light control system using the PIC16F877A microcontroller, designed to 2 modes, The system will manage traffic lights at an intersection with two streets: West and South.
2. Modes of Operation:
• Automatic: The traffic lights switch according to predefined timing:
• West: (15s R, 3s Y, 20s G) / South: (23s R, 3s Y, 12s G)
• Manual: Allows switching between 2 streets.
3. Control Mechanism:
• Three switches are used: 1st switch for start, 2nd for toggling between Manual and Automatic modes, 3rd for selecting between the two streets in Manual mode.
4. Timing in Manual Mode: a (3 seconds) yellow light is applied.
5. Display: a 7-segment displays at each road show the remaining time. BJTs and a 7447 IC are recommended to manage these displays and reduce the number of microcontroller pins used.



B) Flowchart:


void main()
char i,j,x
static int q=1
adcon1 =0b0111,trisb = 0b00000000,crisc=0b00000000,trisd = 0b00000000,trisa = 0b111111,portb = 0b00000000,portc = 0b00000000,portd = 0b00000000
1
True
False
porta.b0==0
porta.b1==1
porta.b1==0
i=0
portb = 0b00000000
trisb.b0=1
gie_bit=1
inte_bit=1
intedg_bit=1
void interrupt()
INTF_bit==1
INTF_bit=0
q==1
portd = 0b00000000
portd.b4=1
portd.b0=1
delay_ms(3000)
portd.b4=0
portd.b5=1
q=~q
portd = 0b00000000
portd.b1=1
portd.b2=1
delay_ms(3000)
portd.b1=0
portd.b2=1
q=~q
i<=23
delay_ms(100)
porta.b0==1 || porta.b1==0
portc = 0b11111111
portd = 0b00000000
break
portc = 0b00000000
portb = numlis1[i]
delay_ms(500)
portd.b4=0
portd.b1=0
portd.b2=0
portd.b0=1
i<20
portd.b5=0
portd.b4=1
portd.b3=0
portd.b5=1
i++
j=0
j<=15
x=0
porta.b0==1 || porta.b1==0
portc = 0b11111111
portd = 0b00000000
break
portb = numlis2[j]
delay_ms(500)
portd.b4=0
portd.b3=1
j<3
portd.b0=0
portd.b1=1
portd.b0=0
portd.b1=0
portd.b2=1
j++
x<=3
porta.b0==1 || porta.b1==0
portc = 0b11111111
portd = 0b00000000
break
portb = numlis3[x]
delay_ms(500)
portd.b3=0
portd.b2=0
portd.b4=1
portd.b1=1
x++

c) Code:

```
static int q=1;
void interrupt(){
  if (INTF_bit==1){
    INTF_bit=0;
    if(q==1){
     portd = 0b00000000;
    portd.b1=1;
    portd.b3=1;
    delay_ms(3000);
    portd.b1=0;
    portd.b2=1;
    q=~q;
    }
    else{
    portd = 0b00000000;
    portd.b4=1;
    portd.b0=1;
    delay_ms(3000);
    portd.b4=0;
    portd.b5=1;
    q =~q;
    }}}
void main(){
char i,j,x;
char numlis1 [] =
{0b00100011,0b00100010,0b00100001,0b00100000,0b00011001,0b00011000,0b00010111,0b00010110,0b00010101,0b00010100,0b00010011,
0b00010010,0b00010001,0b00010000,0b00001001,0b00001000,0b00000111,0b00000110,0b00000101,0b00000100,0b00000011,0b00000010,0b00000001,0b00000000};
char numlis2 []
={0b00010101,0b00010100,0b00010011,0b00010010,0b00010001,0b00001000,0b00001001,0b00001000,0b00000111,0b00000110,0b00000101,0b00000100,0b00000011,0b00000010,0b00000001,0b00000000};
char numlis3 [] ={0b00000011,0b00000010,0b00000001,0b00000000};
adcon1 =0b0111;
```

```
trisb = 0b00000000;
trisc = 0b00000000;
trisd = 0b00000000;
trisa = 0b111111;
portb = 0b00000000;
portc = 0b00000000;
portd = 0b00000000;
for(;;){
    while(porta.b0==0){
          while(porta.b1==1){
                      for (i=0;i<=23;i++){
                              delay_ms(100);
                              if(porta.b0==1||porta.b1==0){
                              portc = 0b11111111;
                              portd = 0b00000000;
                              break;}
                              portc = 0b00000000;
                              portb = numlis1[i];
                              delay_ms(500);
                              portd.b4=0;
                              portd.b1=0;
                              portd.b2=0;
                              portd.b0=1;
                              if(i<20){
                              portd.b3=0;
                              portd.b5=1;
                              }
                              else{
                              portd.b5=0;
                              portd.b4=1;
                              }}
                      for (j=0;j<=15;j++){
                              delay_ms(100);
                              if(porta.b0==1||porta.b1==0){
                              portc = 0b11111111;
```

```
                portd = 0b00000000;
                break;}
                portb = numlis2[j];
                delay_ms(500);
                portd.b4=0;
                portd.b3=1;
                if(j<3){
                portd.b0=0;
                portd.b1=1;
                }
                else{
                portd.b0=0;
                portd.b1=0;
                portd.b2=1;
                }}
        for (x=0;x<=3;x++){
                delay_ms(100);
                if(porta.b0==1||porta.b1==0){
                portc = 0b11111111;
                portd = 0b00000000;
                break;}
                portb = numlis3[x];
                delay_ms(500);
                portd.b3=0;
                portd.b2=0;
                portd.b4=1;
                portd.b1=1;
                }}
        if (porta.b1==0){
           portb = 0b00000000;
           trisb.b0=1;
           gie_bit=1;
           inte_bit=1;
           intedg_bit=1;
}}}}
```

Code overview:

- The automatic mode works in 2 while loops Overlapse**:**
    1- While loop works if start switch is on.
    2- While loop works if the automatic switch in on.
- When the manual switch is on the program exits the second loop and goes to if condition checks the manual switch if true it make the portb.b0 a input pin and works as interrupt pin connected to convert switch that toggles between streets by going to interrupt function and considering the 3s yellow light.
- After interrupt happened the program return to the main function.

Links:

Github link: https://github.com/ziadaymen29/traffic-light-project

Youtube link: https://youtu.be/Z6jYNnV3z80?si=J2CzupsTc-sGIcle