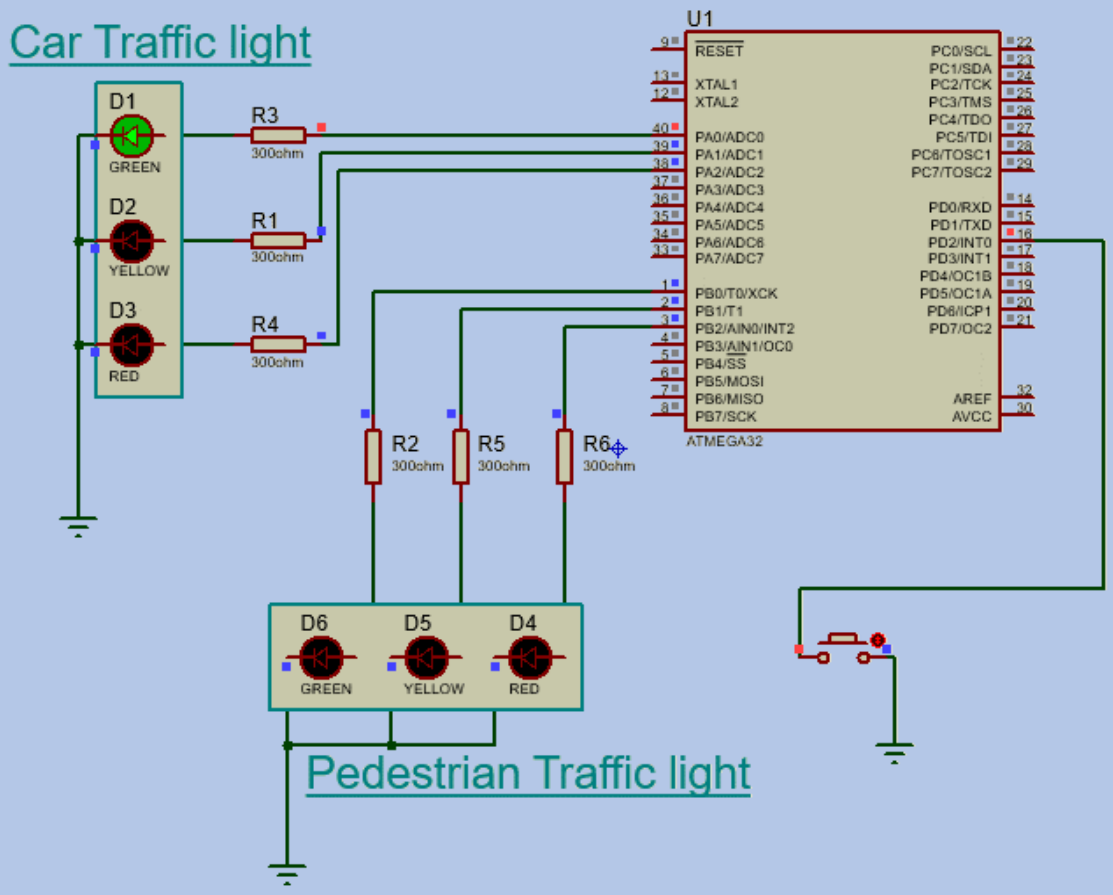
Project: on-demand Traffic light control



System description:

This traffic light system is divided into two sections:

1- Car traffic light
   Could be Green, Blinking yellow or Red.

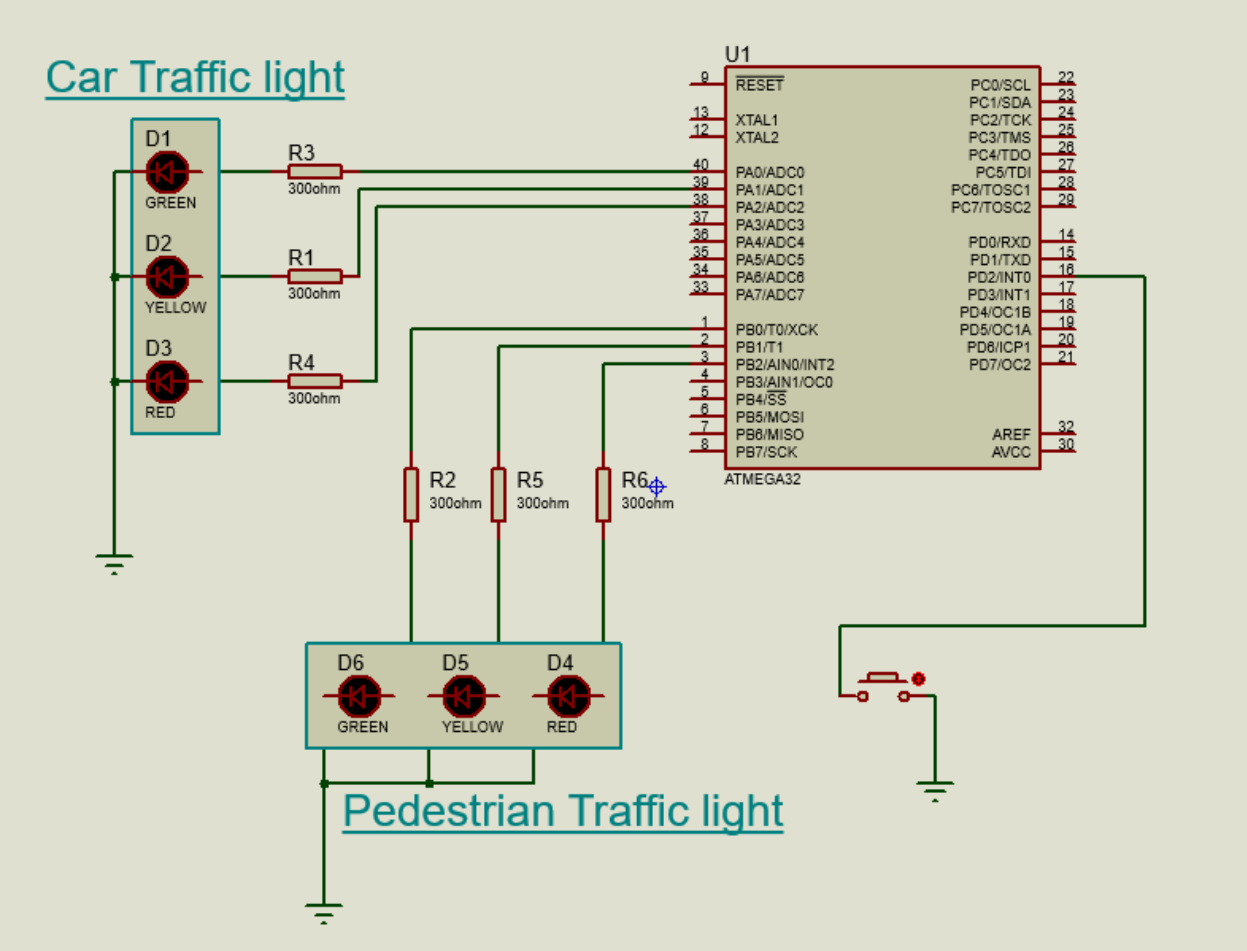
2- Pedestrian traffic light
   Could be Green, Blinking yellow or Red.

This system mainly consists of microcontroller ,6 LEDs and a Push button

1. ATmega32 microcontroller
2. One push button connected to INT0 pin for pedestrian
3. Three LEDs for cars - Green, Yellow, and Red, connected on port A, pins 0, 1, and 2
4. Three LEDs for pedestrians - Green, Yellow, and Red, connected on port B, pins 0, 1, and 2
5. Six resistors (300ohm) to constrain the current strength.

At first the Pedestrian Red light is off when the Pedestrian press the button the microcontroller gets an interrupt order to make the car red led on and the Pedestrian green order in few steps with logical order to maintain the stability of the traffic flow.

system design:



This image indicates the system design :

The car traffic light is attached to port a pin 0 ,1 and 2.

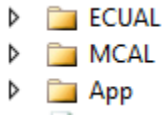
The pedestrian LEDs light is attached to port b pin 0 ,1 and 2.

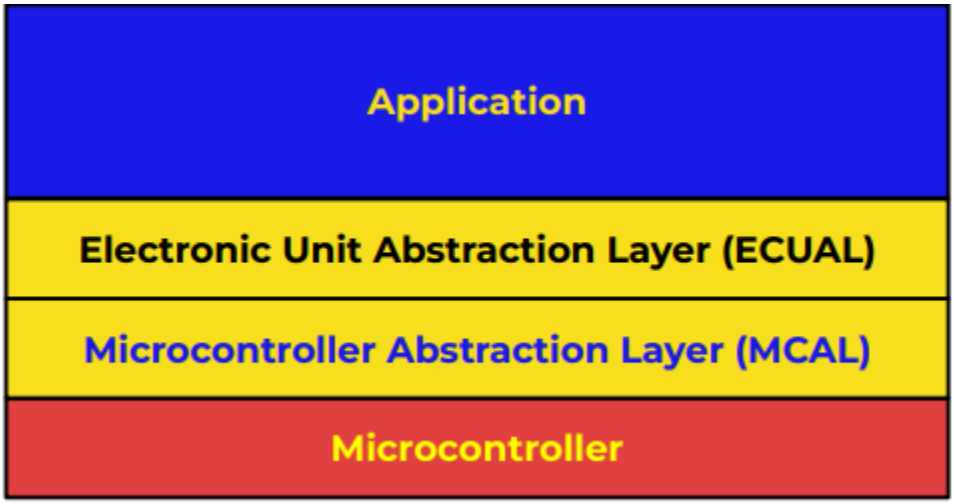
Using microcontroller, we control the flow of the traffic.

Mainly the flow of the system is divided into six states as follow:

Car Green state ,Common Yellow state , Car Red state , Pedestrian Green state , Pedestrian Red state and Pedestrian Yellow state.

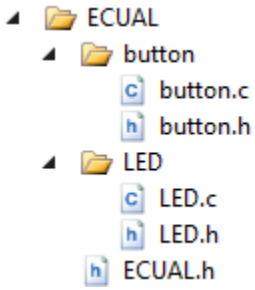
Mainly the system is divided into 3 layers as follow:





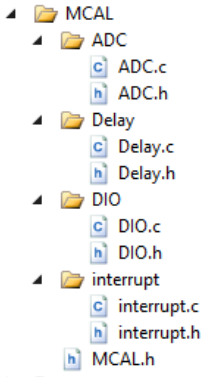
1- ECUAL layer:

Contain LEDs driver and push button driver.



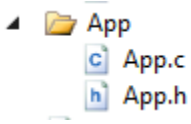
2- MCAL layer:

Contain Microcontroller driver like, Ports , I/O , set bit functions and so more.



3- App layer :

Contain the main code which control the flow of the traffic light.



The only new type that we have is traffic state.

```
typedef enum
{
    CAR_GREEN,
    COMMON_YELLOW,
    CAR_RED,
    PED_YELLOW,
    PED_GREEN,
    PED_RED
}Traffic_State;
```

It's used in the app to direct the flow of the program and it's defined in the LED.h.

Functions that are used in the project:

1- State functions

Every state has its own function which do its job.

Car Green state ,Common Yellow state , Car Red state , Pedestrian Green state ,

Pedestrian Red state and Pedestrian Yellow state.

For example :

Car Green state makes the car green led on and the other LEDs off.

2- Case director function

This function helps the program to get in the next state.

```
void case_director(void);
```

3- INT mode is used due to the lack of memory in the ISR

```
void INT_mode(void);
```

4- Traffic button init

This function initializes the input button as pull-up and Interrupt 0 sensing falling edge.

```
void Traffic_button_init(void);
```

5- LEDs init

This function initializes all the output LEDs.

```
void LEDS_init(void);
```

6- All LEDS off

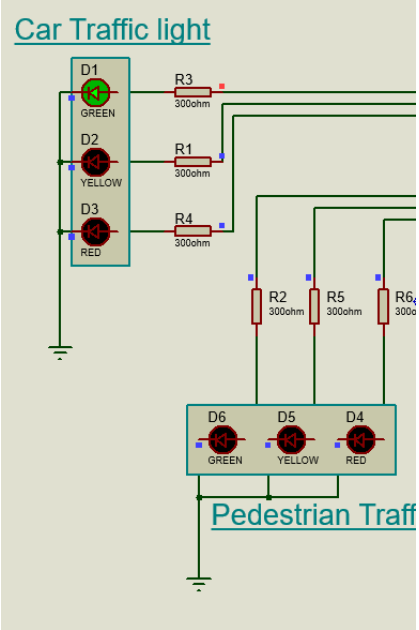
It turns all LEDs off every new state ,actually it’s not a big deal it’s just for insurance and eliminate any runtime error.

```
void ALL_LEDS_OFF(void);
```

Let’s talk about states.

1- Car Green state

It’s simple state. In this case only car green led is on for five sec and this it saves the next state in the global variable (NEXT_STATE).



2- Common Yellow state

The function is called when the previous state was either CAR_GREEN or CAR_RED. The function turns off all LEDs, then blinks the yellow LED for 5 seconds. If there was an interrupt, both yellow LEDs blinks and the red LED is turned on. After 5 seconds, the yellow LED is turned off. If there was no interrupt, the next state is either CAR_GREEN or CAR_RED depends on the previous state . if there was an interrupt, the next state is PED_GREEN

3- Car Red state

The function turns on the red LED for 5 seconds, then turns it off. If the interrupt is not triggered, the next state is the common yellow state.

4- Pedestrian Green state

Turn off all LEDs, turn on the green pedestrian LED, turn on the red car LED, wait 5 seconds, set the next state to PED_YELLOW.

5- Pedestrian yellow state

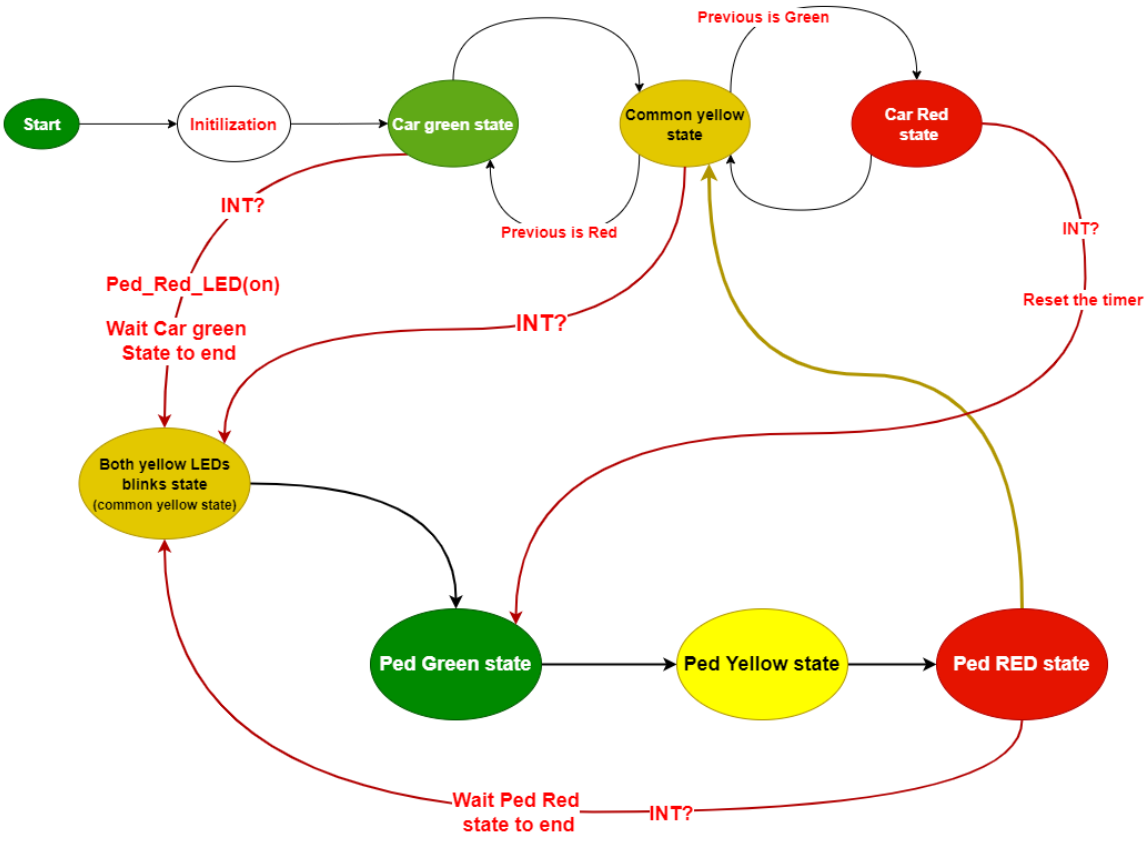
The function turns on the green pedestrian light and turns off the red car light. Then it toggles the yellow car and pedestrian lights 20 times with a 250ms delay between each toggle. Then it turns off the yellow pedestrian light and the green pedestrian light. Finally, it sets the next state to PED_RED.

6- Pedestrian Red state

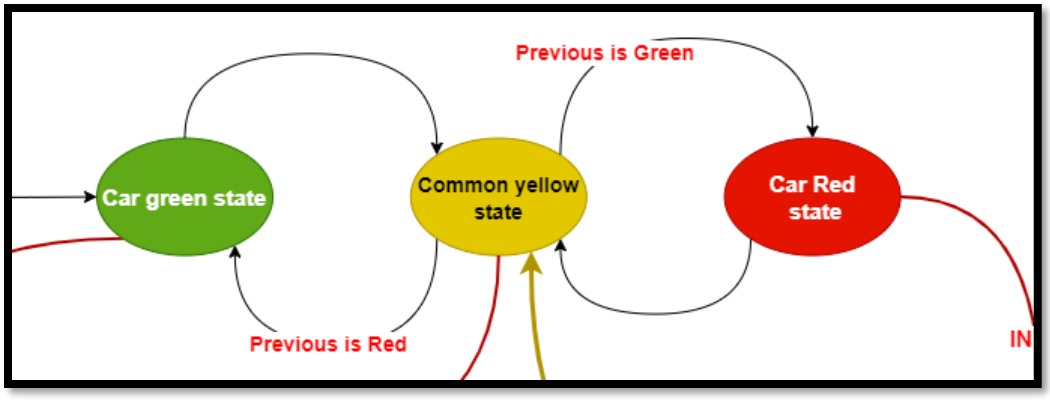
Same as Car green state but the Pedestrian red is on.

system flow

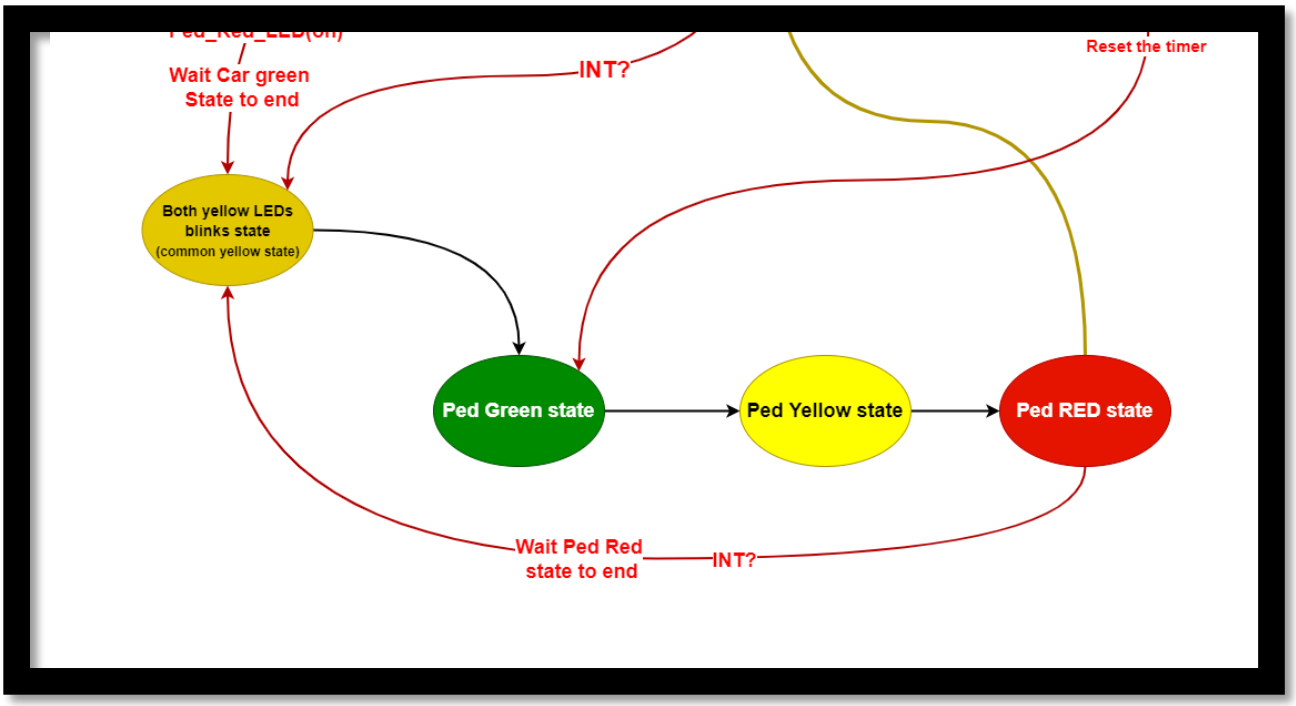
This the State diagram and flowchart describe the flow of the system.



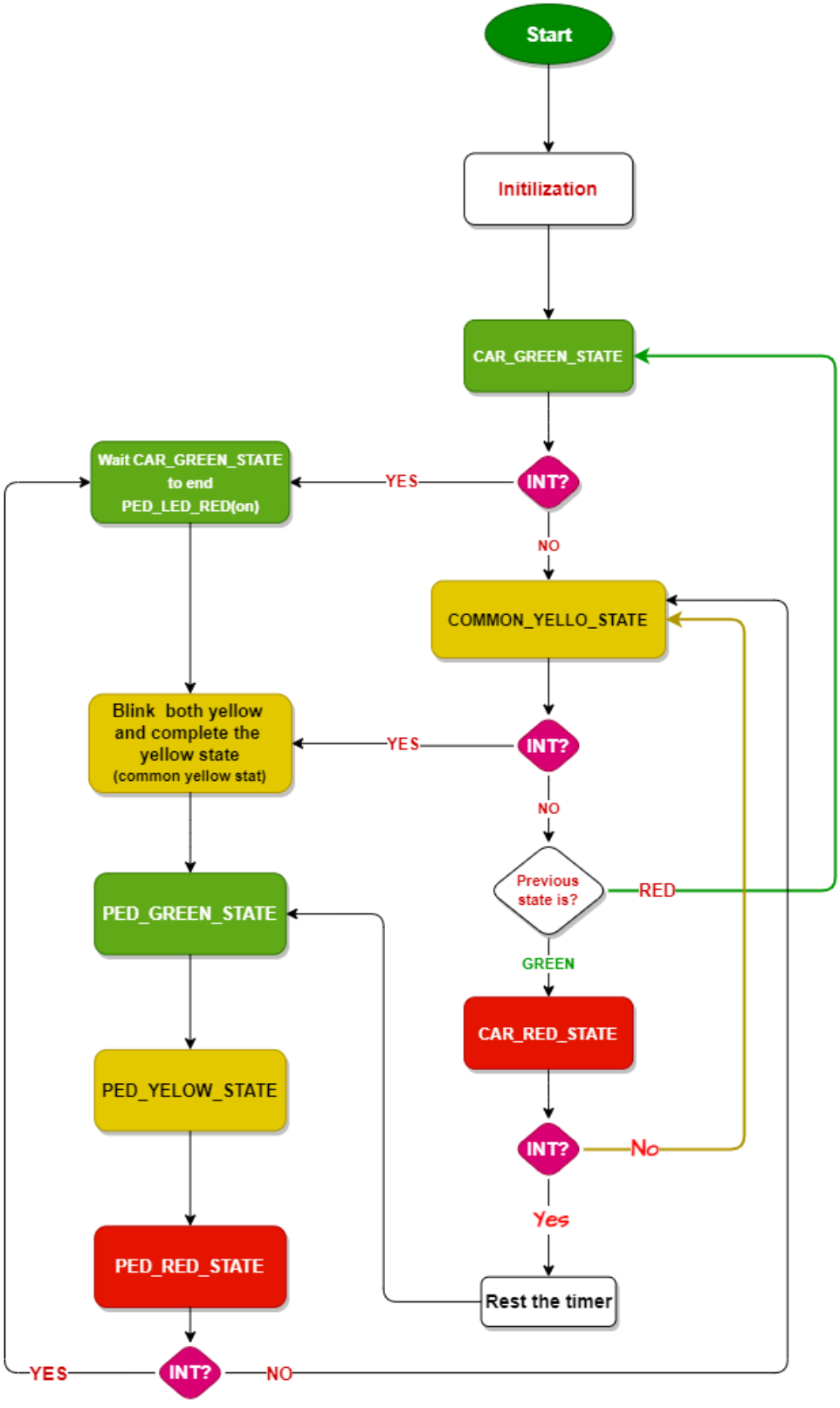
This discribes the normal mode



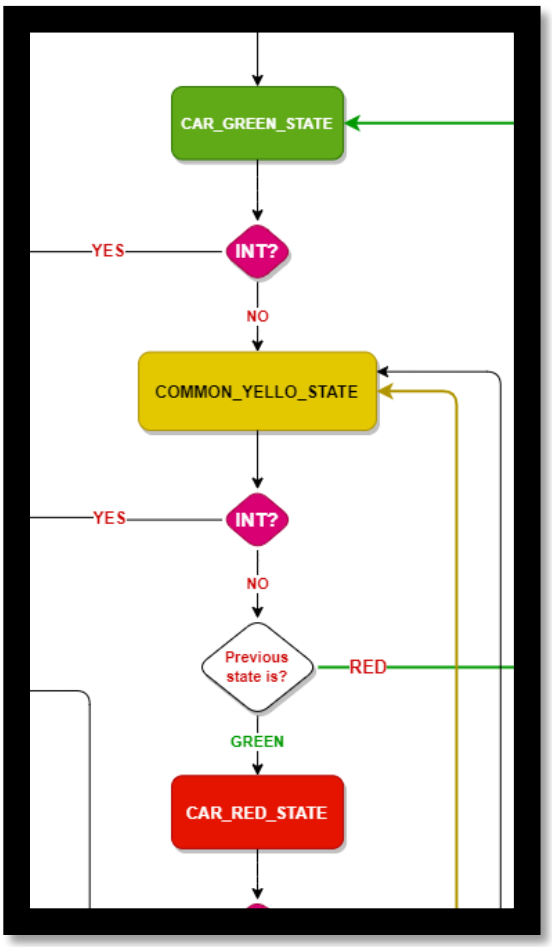
This discribes the Interrupt service mode



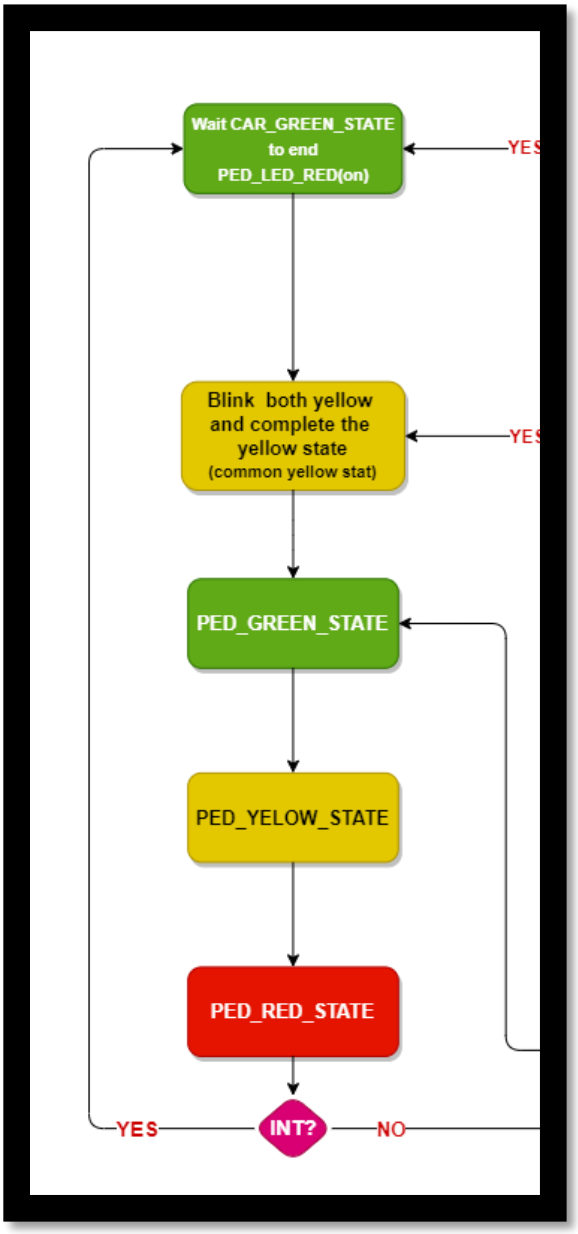
If the State diagram hard to understand, then the flowchart is easier.



This discribes the normal mode



This discribes the Interrupt service mode



The system constraints

1- When the button is pressed, the microcontroller stops receiving input until the interrupt servicing function is completed.

2- When the Car RED LED is turned on and the button is pushed, the timer is reset to give the Pedestrian sufficient time.

3- If the microcontroller receives an interrupt when the state is Yellow and is about to change to CAR GREEN, it returns to the CAR RED state, giving the pedestrian priority to pass.

4- Long press makes the microcontroller enter the ISR once.

5- Double or triple or more presses make the microcontroller enter the ISR once.