# On-demand Traffic Light Control System

## System Description:

Traffic lights are signaling devices positioned at road intersections, pedestrian crossings, and other locations to control the flow of traffic.

Traffic lights normally consist of three signals, transmitting meaning to drivers and riders through colors and symbols including arrows and bicycles.

The regular traffic light colors are red, yellow, and green arranged vertically or horizontally in that order.

Although this is internationally standardized, variations exist on national and local scales as to traffic light sequences and laws.

You are required to implement a traffic lights system with an on-demand crosswalk button.

Crosswalk buttons let the signal operations know that someone is planning to cross the street, so the light adjusts, giving the pedestrian enough time to get across.

## System Design:

Hardware requirements:

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

Software requirements:

In normal mode:

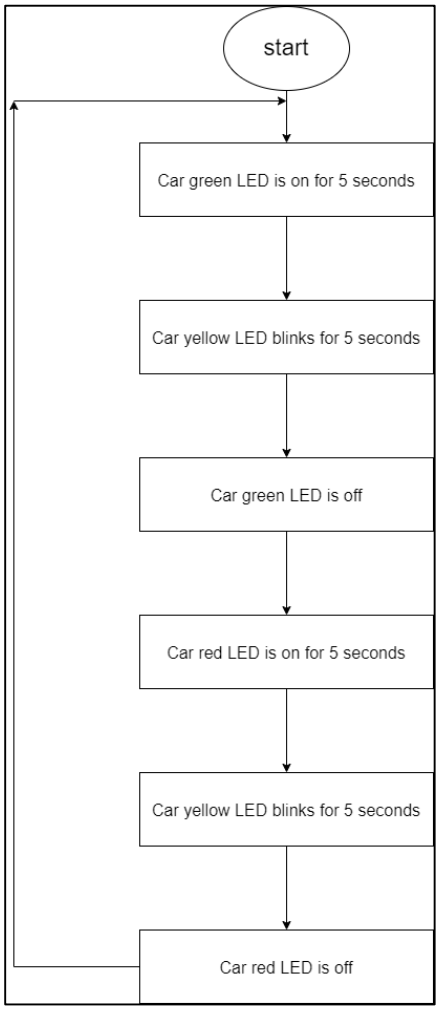
1- Cars' LEDs will be changed every five seconds starting from Green then yellow then red then yellow then Green.
2- The Yellow LED will blink for five seconds before moving to Green or Red LEDs.

In pedestrian mode:

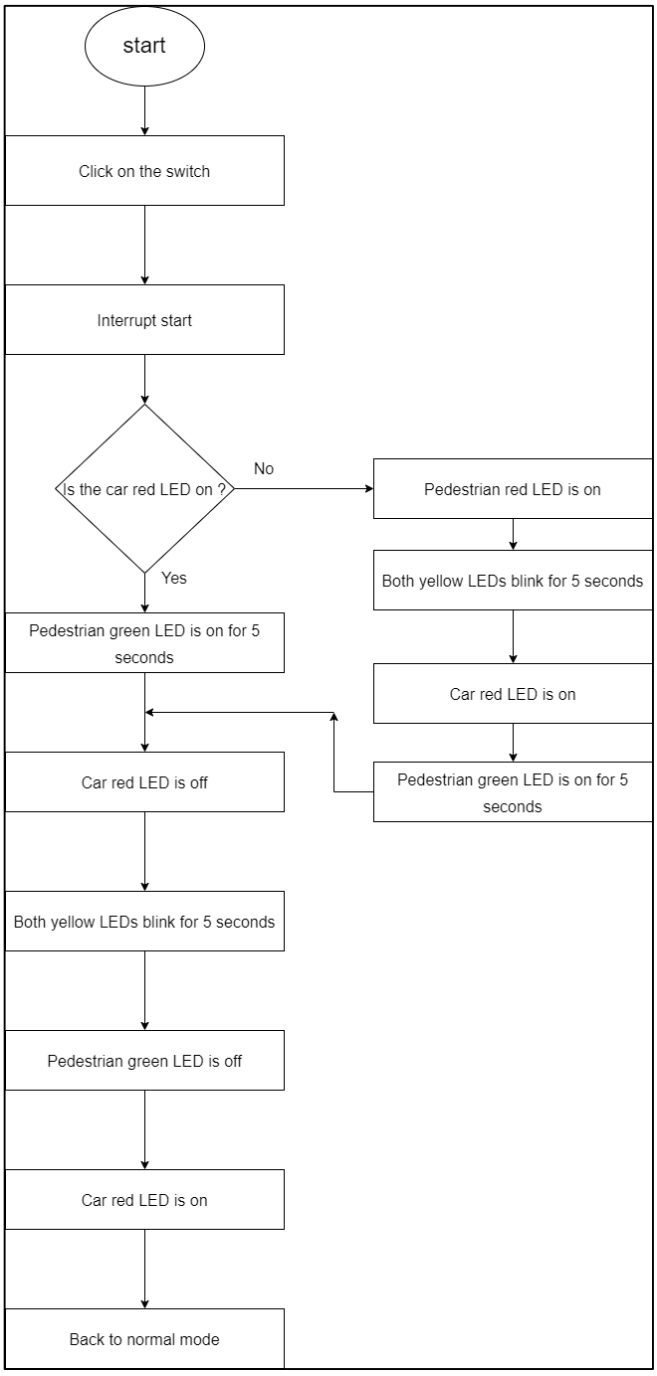
1- Change from normal mode to pedestrian mode when the pedestrian button is pressed.
2- If pressed when the cars' Red LED is on, the pedestrian's Green LED and the cars' Red LEDs will be on for five seconds, this means that pedestrians can cross the street while the pedestrian's Green LED is on.
3- If pressed when the cars' Green LED is on or the cars' Yellow LED is blinking, the pedestrian Red LED will be on then both Yellow LEDs start to blink for five seconds, then the cars' Red LED and pedestrian Green LEDs are on for five seconds, this means that pedestrian must wait until the Green LED is on.
4- At the end of the two states, the cars' Red LED will be off and both Yellow LEDs start blinking for 5 seconds and the pedestrian's Green LED is still on.
5- After the five seconds the pedestrian Green LED will be off and both the pedestrian Red LED and the cars' Green LED will be on.
6- Traffic lights signals are going to the normal mode again.

## System Flow Chart:

Normal Mode:




Pedestrian Mode



## System Summary:

| | |
|---|---|
| System Layers | This system consists of three layers which are:<br>1- MCAL.<br>2- HAL.<br>3- Application. |
| System Drivers | MCAL drivers:<br>1- DIO (digital input/output).<br>2- Timer0.<br>3- EXIT (external interrupt).<br>4- GIE (global interrupt).<br>5- WDT (watchdog timer).<br>HAL drivers:<br>1- LED.<br>2- Button. |

## Layers Description:

### 1- MCAL Layer:

### a) DIO Driver APIs:

| | |
|---|---|
| DIO_voidSetPinDirection | **Description:**<br>This function is used to determine whether the pin is used as an input or an output.<br>**Input Arguments:**<br>**1- portNumber:** A variable of u8 data type representing the chosen port.<br>**2- pinNumber:** A variable of u8 data type representing the chosen pin.<br>**3- direction:** A variable of u8 data type representing the chosen direction. |

| | |
|---|---|
| | **Return Type:** errorState data type representing whether an error occurred or not. |
| DIO_voidSetPinValue | **Description:**<br>This function is used to determine whether the pin is high or low.<br>**Input Arguments:**<br>1- **portNumber:** A variable of u8 data type representing the chosen port.<br>2- **pinNumber:** A variable of u8 data type representing the chosen pin.<br>3- **value:** A variable of u8 data type representing the chosen value (high or low).<br>**Return Type:** errorState data type representing whether an error occurred or not. |
| DIO_u8GetPinValue | **Description:**<br>This function is used to get the value of a certain pin.<br>**Input Arguments:**<br>1- **portNumber:** A variable of u8 data type representing the chosen port.<br>2- **pinNumber:** A variable of u8 data type representing the chosen pin.<br>**Return Type:** u8 data type representing the pin value. |

## b) Timer0 Driver APIs:

| | |
|---|---|
| TIMER0_voidInit | **Description:**<br>This function is used to initialize timer0. |

| | |
|---|---|
| | **Input Arguments:** None.<br>**Return Type:** errorState data type representing whether an error occurred or not. |
| TIMER0_voidStart | **Description:**<br>This function is used to start timer0.<br>**Input Arguments:** None.<br>**Return Type:** errorState data type representing whether an error occurred or not. |
| TIMER0_voidStop | **Description:**<br>This function is used to stop timer0.<br>**Input Arguments:** None.<br>**Return Type:** errorState data type representing whether an error occurred or not. |
| TIMER0_voidDelay | **Description:**<br>This function is used to generate delay time using timer0.<br>**Input Arguments:**<br>1- **delayTime:** A variable of f32 data type representing the required delay time in seconds.<br><br>**Return Type:** errorState data type representing whether an error occurred or not. |
| TIMER0_voidSetTimerValue | **Description:**<br>This function is used to set the initial value of timer0 register to start counting from it.<br>**Input Arguments:**<br>1- **value:** A variable of u8 data type representing the required initial value of timer0. |

| | |
|---|---|
| | **Return Type:** errorState data type representing whether an error occurred or not. |

## c) EXIT Driver APIs:

| | |
|---|---|
| EXIT_voidInit | **Description:**<br>This function is used to initialize external interrupt.<br>**Input Arguments:**<br>**1- peripheral:** A variable of u8 data type representing the required external interrupt peripheral (INT0, INT1 or INT2).<br>**2- mode:** A variable of u8 data type representing the required mode of operation (rising edge, falling edge, ..etc.).<br>**Return Type:** errorState data type representing whether an error occurred or not. |
| EXIT_voidSetCallbackINT0<br>EXIT_voidSetCallbackINT1<br>EXIT_voidSetCallbackINT2 | **Description:**<br>These functions are used to set the desired callback function of each external interrupt.<br>**Input Arguments:**<br>1- **ptr:** A generic pointer to function used to pass a function to each external interrupt ISR.<br>**Return Type:** errorState data type representing whether an error occurred or not. |

| | |
|---|---|
| INT0_VECT<br>INT1_VECT<br>INT2_VECT | **Description:**<br>These functions are the ISR vectors of each external interrupt.<br>**Input Arguments:** None.<br>**Return Type:** void. |

## d) GIE Driver APIs:

| | |
|---|---|
| GIE_voidEnable | **Description:**<br>This function is used to enable the global interrupt.<br>**Input Arguments:** None.<br>**Return Type:** errorState data type representing whether an error occurred or not. |
| GIE_voidDisable | **Description:**<br>This function is used to disable the global interrupt.<br>**Input Arguments:** None.<br>**Return Type:** errorState data type representing whether an error occurred or not. |

## e) WDT Driver APIs:

| | |
|---|---|
| WDT_voidEnable | **Description:**<br>This function is used to enable the watchdog timer.<br>**Input Arguments:** None.<br>**Return Type:** errorState data type representing whether an error occurred or not. |

## 2- HAL Layer:

### a) LED Driver APIs:

| | |
|---|---|
| LED_voidInit | **Description:**<br>This function is used to initialize LED through DIO driver.<br>**Input Arguments:**<br>1- **portNumber:** A variable of u8 data type representing the chosen port.<br>2- **pinNumber:** A variable of u8 data type representing the chosen pin.<br><br>**Return Type:** errorState data type representing whether an error occurred or not. |
| LED_voidOnOff | **Description:**<br>This function uses DIO driver to turn the LED on or off.<br>**Input Arguments:**<br>1- **portNumber:** A variable of u8 data type representing the chosen port.<br>2- **pinNumber:** A variable of u8 data type representing the chosen pin.<br><br>**Return Type:** errorState data type representing whether an error occurred or not. |

### b) Button Driver APIs:

| | |
|---|---|
| BTN_voidInit | **Description:**<br>This function is used to initialize the button through DIO driver and enable the internal pull up resistor. |

| | |
|---|---|
| | **Input Arguments:**<br>1- **portNumber:** A variable of u8 data type representing the chosen port.<br>2- **pinNumber:** A variable of u8 data type representing the chosen pin.<br><br>**Return Type:** errorState data type representing whether an error occurred or not. |
| BTN_u8ButtonIsPressed | **Description:**<br>This function is used to determine whether the button is pressed or not using DIO driver.<br>**Input Arguments:**<br>1- **portNumber:** A variable of u8 data type representing the chosen port.<br>2- **pinNumber:** A variable of u8 data type representing the chosen pin.<br><br>**Return Type:** u8 data type representing whether the button is pressed or not. |

## Used Data Types:

| Data type | Description |
|---|---|
| u8 | A new data type equivalent to "unsigned char" data type.<br><br>A variable of u8 data type consumes 8 bits of memory. |

| | |
|---|---|
| f32 | A new data type equivalent to "float" data type.<br><br>A variable of f32 data type consumes 32 bits of memory. |
| errorState | An enumeration data type which has two elements (OK, ERROR_OCCURED).<br><br>This data type is used in return type of functions to declare whether an error occurred in the execution of a function or not. |