On-Demand Traffic Light Control System

1 System Description

The system is a traffic light system having two modes: the normal mode and the pedestrian mode. The normal mode is used by cars, and the pedestrian mode is used by pedestrians wanting to cross the street. Each mode is represented with 3 LEDs: GREEN, YELLOW, and RED connected to two different MCU ports: PORTA and PORTB. There is a button connected to a pin in MCU PORTD to go from the normal mode to the pedestrian mode. Simply, when a pedestrian wants to cross the street, it presses the button. Then the lighting sequence of the LEDs will be managed in some way to allow the pedestrian to cross street without any accidents.

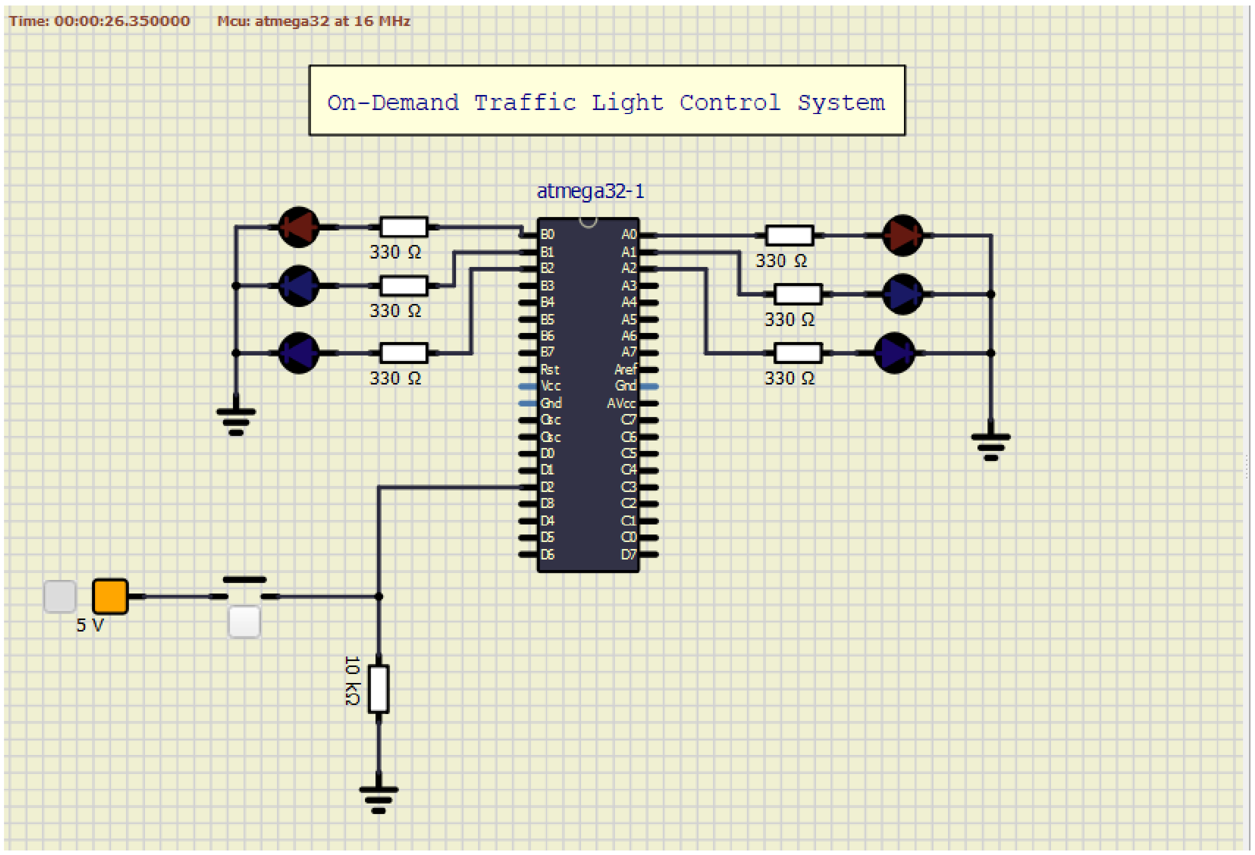


Figure 1 Circuit Design

For the sake of explanation, the LED sequence of each mode and how each mode is entered and leaved, I will use the term Cars' LED to refer to the normal mode LEDS and the term Pedestrians' LED to refer to the pedestrian mode LEDs.

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.

Pedestrian Mode

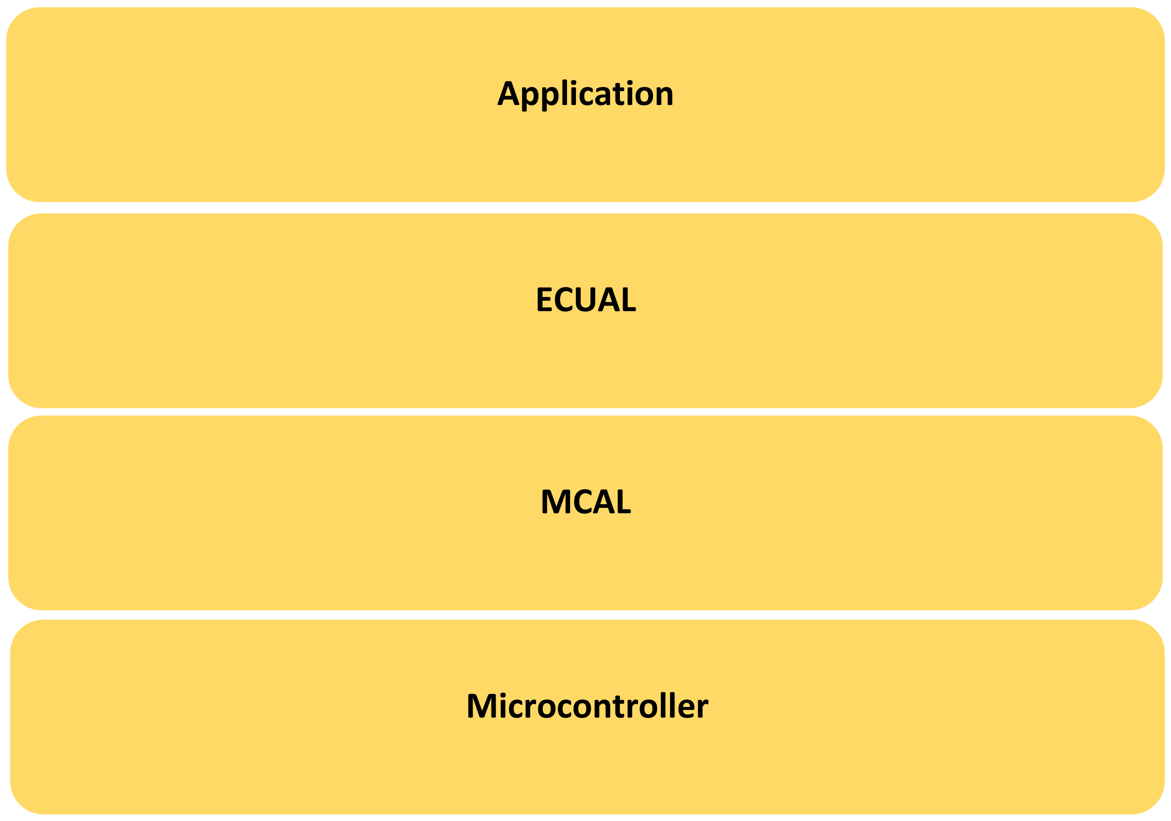
1. Change from normal mode to pedestrian mode when the pedestrian button is pressed.
2. If pressed when the cars' Red LED is on, the pedestrian's Green LED and the cars' Red LEDs will be on for five seconds, this means that pedestrians can cross the street while the pedestrian's Green LED is on.
3. If 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.

2 System Design

2.1 System Layers

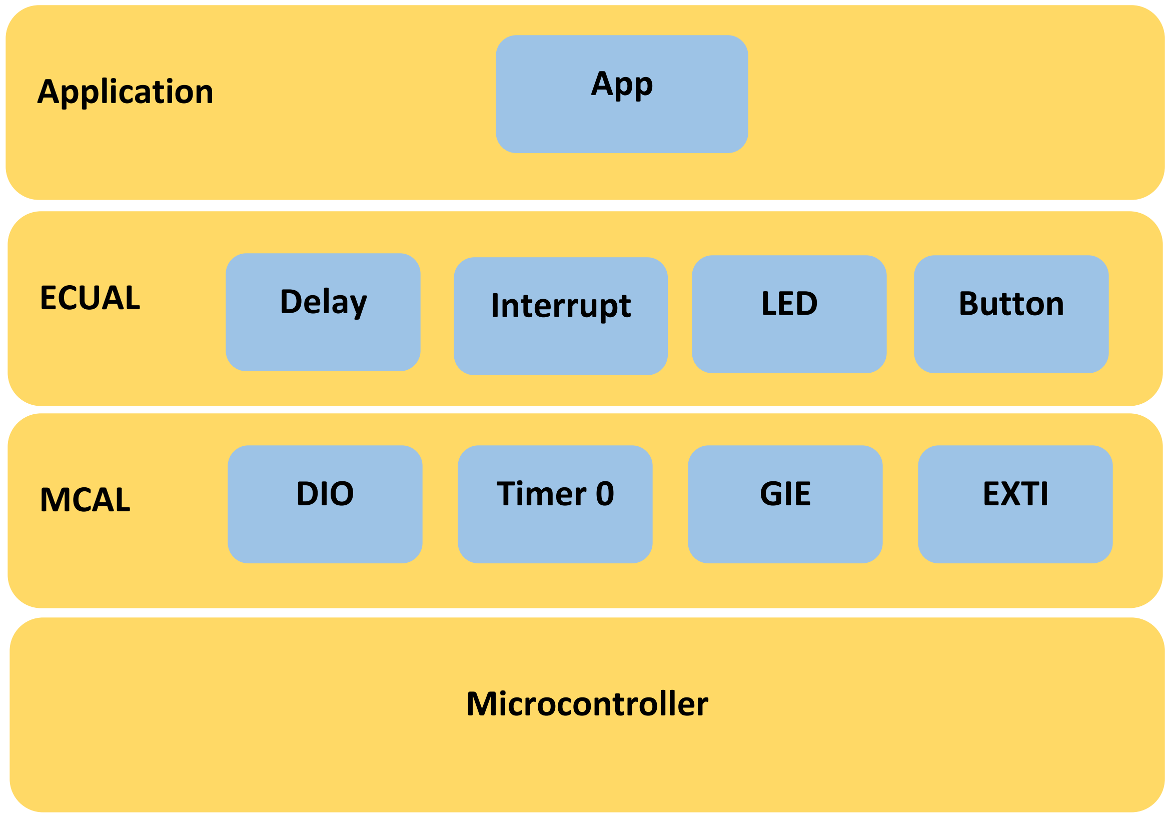
The system consists of 4 layers from top to bottom:

1. Application layer
2. Electronic control unit abstraction layer (ECUAL)
3. Microcontroller unit abstraction layer (MCAL)
4. Microcontroller layer



2.2 System Drivers

The system consists of the following drivers: DIO, Timers, Global Interrupts (GIE), External Interrupts (EXTI), LED, Button, Interrupts, Delay.



2.2.1 DIO Driver

2.2.1.1 Data Types

```
typedef enum
{
  PORT_A,
  PORT_B,
  PORT_C,
  PORT_D

} En_dioPort_t;
```

```c
typedef enum
{
  PIN0,
  PIN1,
  PIN2,
  PIN3,
  PIN4,
  PIN5,
  PIN6,
  PIN7
} En_dioPin_t;
```

```c
typedef enum
{
  IN,
  OUT
} En_dioDirection_t;
```

```c
typedef enum
{
  LOW,
  HIGH
} En_dioValue_t;
```

2.2.1.2 API

```c
En_dioError_t DIO_init(En_dioPort_t portNumber,
                       En_dioPin_t pinNumber,
                       En_dioDirection_t direction);

En_dioError_t DIO_write(En_dioPort_t portNumber,
                        En_dioPin_t pinNumber,
                        En_dioValue_t value);

En_dioError_t DIO_read(En_dioPort_t portNumber,
                       En_dioPin_t pinNumber,
                       uint8_t* value);

En_dioError_t DIO_toggle(En_dioPort_t portNumber,
                         En_dioPin_t pinNumber);
```

```
En_dioError_t DIO_init(En_dioPort_t portNumber,
                       En_dioPin_t pinNumber,
                       En_dioDirection_t direction);
```

2.2.1.2.1 Description

DIO_init initializes a given pin in a given port to be input or output.

2.2.1.2.2 Input parameters

2.2.1.2.2.1 portNumber

the number of the port containing the pin you want to initialize.

2.2.1.2.2.2 pinNumber

The number of the pin you want to initialize.

2.2.1.2.2.3 Direction

The direction with which you want to initialize the pin.

2.2.1.2.3 Return

DIO_SUCCESS. If the there is no wrong values in the input parameters.

DIO_FAIL. If there is any wrong value in the input parameters. For example, pinNumbe is greater than 7.

```
En_dioError_t DIO_write(En_dioPort_t portNumber,
                        En_dioPin_t pinNumber,
                        En_dioValue_t value)
```

2.2.1.2.4 Description

DIO_write sets a given pin of a given port to be HIGH (1) or LOW (0)

2.2.1.2.5 Input parameters

2.2.1.2.5.1 portNumber

the number of the port containing the pin you want to set its value.

2.2.1.2.5.2 pinNumber

The number of the pin you want to set its value.

2.2.1.2.5.3 value

The value with which you want to set the pin.

2.2.1.2.6 Return

DIO_SUCCESS

If the there is no wrong values in the input parameters.

DIO_FAIL

If there is any wrong value in the input parameters.

```
En_dioError_t DIO_read(En_dioPort_t portNumber,
                       En_dioPin_t pinNumber,
                       uint8_t* value)
```

2.2.1.2.7 Description
DIO_read reads the state a given pin of a given port and puts it into the **value** pointer

2.2.1.2.8 Input parameters

2.2.1.2.8.1 portNumber
the number of the port containing the pin you want to read its value.

2.2.1.2.8.2 pinNumber
The number of the pin you want to read its value.

2.2.1.2.8.3 Value
The pointer that will store the value of the pin.

2.2.1.2.9 Return
DIO_SUCCESS

If the there is no wrong values in the input parameters.

DIO_FAIL

If there is any wrong value in the input parameters.

```
En_dioError_t DIO_toggle(En_dioPort_t portNumber, En_dioPin_t pinNumber)
```

2.2.1.2.10 Description
DIO_toggle toggles the state a given pin of a given port.

2.2.1.2.11 Input parameters

2.2.1.2.11.1 portNumber
the number of the port containing the pin you want to toggle its value.

2.2.1.2.11.2 pinNumber
The number of the pin you want to toggle its value.

2.2.1.2.12 Return
DIO_SUCCESS

If the there is no wrong values in the input parameters.

DIO_FAIL

If there is any wrong value in the input parameters.

2.2.2 Timer 0 Driver

2.2.2.1 Data Types

```
typedef enum
{
	NO_PRESCALER = 1,
	PRESCALER_8 = 8,
	PRESCALER_64 = 64,
	PRESCALER_256 = 256,
	PRESCALER_1024 = 1024
}En_timerPrescaler_t;
```

```
typedef enum
{
	FAIL,
	SUCCESS
}En_timerError_t;
```

2.2.2.2 API

```
En_timerError_t TIMER0_init(uint8_t timerInitValue);
En_timerError_t TIMER0_start(En_timerPrescaler_t clockPrescaler);
En_timerError_t TIMER0_getState(void);
En_timerError_t TIMER0_stop(void);
```

```
En_timerError_t TIMER0_init(uint8_t timerInitValue);
```

2.2.2.2.1 Description

This function determines the timer mode and its initial value.

2.2.2.2.2 Input Parameters

2.2.2.2.2.1 timerInitValue

the initial value to put in the timer/counter register from which the timer start couting.

2.2.2.2.3 Return

TIMER_FAIL

If the timerInitValue is wrong. For example, the timerInitValue is larger than 255 or less than 0

TIMER_SUCCESS

If there no errors within the function or in the timerInitValue parameter

```
En_timerError_t TIMER0_start(En_timerPrescaler_t clockPrescaler);
```

2.2.2.2.4 Description
This function starts the timer with a given prescaler to start counting from the its initial value.

2.2.2.2.5 Input Parameters

2.2.2.2.5.1 clockPrescaler
it represents the by which the frequency will be divided to change the timer tick to be shorter or longer. The available values for the clock prescaler is represented by the En_timerPrescaler_t data type where the smallest value is 1 and the largest value is 1024.

2.2.2.2.6 Return
TIMER_FAIL

If, for example, the clockPrescaler is out of its specified range.

TIMER_SUCCESS

If there is no errors within the function or in the clockPrescaler value.

```
En_timerError_t TIMER0_getState(void)
```

2.2.2.2.7 Description
This function gets the timer state and returns only if the timer is overflowed.

2.2.2.2.8 Input Parameters
It doesn't take any input parameters

2.2.2.2.9 Return
TIMER_SUCCESS

If the timer is overflowed and the overflow flag is cleared.

TIMER_FAIL

If there is any error within the function.

```
En_timerError_t TIMER0_stop(void)
```

2.2.2.2.10 Description
This function stops the timer from counting.

2.2.2.2.11 Input Parameters
It doesn't take any input parameters

2.2.2.2.12 Return
TIMER_SUCCESS

If the timer is stopped properly without any problems.

TIMER_FAIL

If there is any error within the function.

2.2.3 GIE Driver

2.2.3.1 Data Types

```
typedef enum
{
        GIE_FAIL,
        GIE_SUCCESS
}En_gieError_t;
```

2.2.3.2 API

```
En_gieError_t GIE_enable(void);
En_gieError_t GIE_disable(void);
```

```
En_gieError_t GIE_enable(void);
```

2.2.3.2.1 Description

This function enables the interrupts globally.

2.2.3.2.2 Input Parameters

There are no input parameters for this function.

2.2.3.2.3 Return

GIE_FAIL

If there is any error within the function

GIE_SUCCESS

If the global interrupts are enabled successfully

```
En_gieError_t GIE_disable(void);
```

2.2.3.2.4 Description

This function disables the interrupts globally.

2.2.3.2.5 Input Parameters

There are no input parameters for this function.

2.2.3.2.6 Return

GIE_FAIL

If there is any error within the function

GIE_SUCCESS

If the global interrupts are disabled successfully.

2.2.4 EXTI Driver

2.2.4.1 Data Types

```
typedef enum
{
	FALLING_EDGE,
	RISING_EDGE,
	ANY_LOGICAL_CHANGE,
	LOW_LEVEL
}En_interruptTrigger_t;
```

```
typedef enum
{
	EXTI_FAIL,
	EXTI_SUCCESS
}En_extiError_t;
```

2.2.4.2 API

```
En_extiError_t EXTI_int0Init(void);
En_extiError_t EXTI_int0SetCallback(void(*int0Func)(void));
En_extiError_t EXTI_int0SetInterruptTrigger(En_interruptTrigger_t
interruptTrigger);
```

```
En_extiError_t EXTI_int0Init(void);
```

2.2.4.2.1 Description

This function enables the external interrupt INT0.

2.2.4.2.2 Input parameters

There are no input parameters for this function.

2.2.4.2.3 Return

EXTI_FAIL

If there is any error within the function

EXTI_SUCCESS

If the function enables the external interrupt INT0 successfully.

```
En_extiError_t EXTI_int0SetCallback(void(*int0Func)(void));
```

2.2.4.2.4 Description
This function sets upper layer function to be call backed from the interrupt service routing related to the external interrupt INT0.

2.2.4.2.5 Input parameters

2.2.4.2.5.1 Int0Func
It's a function pointer that carries the address of the function to be call backed when the interrupt happens.

2.2.4.2.6 Return
EXTI_FAIL

If there is any error within function. For example, the function pointer is NULL.

EXTI_SUCCESS

If there is no error.

```
En_extiError_t EXTI_int0SetInterruptTrigger(En_interruptTrigger_t
interruptTrigger);
```

2.2.4.2.7 Description
This function sets the trigger of the external interrupts INT0 to be triggered on rising edge, falling edge, or low-level.

2.2.4.2.8 Input parameters
interruptTrigger

it represents the trigger type of the external interrupt.

2.2.4.2.9 Return
EXTI_FAIL

If there is any error within the function. For example, the interruptTrigger value doesn't exist.

EXTI_SUCCESS

If there is no error within the function.

2.2.5 LED Driver

2.2.5.1 Data Types

```
typedef enum
{
        LED_FAIL,
        LED_SUCCESS
}En_ledError_t;
```

2.2.5.2 API

```
En_ledError_t LED_init(En_dioPort_t ledPort, En_dioPin_t ledPin);
En_ledError_t LED_on(En_dioPort_t ledPort, En_dioPin_t ledPin);
En_ledError_t LED_off(En_dioPort_t ledPort, En_dioPin_t ledPin);
En_ledError_t LED_toggle(En_dioPort_t ledPort, En_dioPin_t ledPin);
En_ledError_t LED_read(En_dioPort_t ledPort, En_dioPin_t ledPin, uint8_t* data);
```

```
En_ledError_t LED_init(En_dioPort_t ledPort, En_dioPin_t ledPin);
```

2.2.5.2.1 Description

Each LED is connected to a given pin of a given port. LED is an output device, so this function initializes the pin to which LED is connected to be output.

2.2.5.2.2 Input parameters

2.2.5.2.2.1 ledPort

this variable represents the DIO port having the pin to which the LED is connected.

2.2.5.2.2.2 ledPin

this variable represents the DIO pin to which the LED is connected.

2.2.5.2.3 Return

LED_FAIL

If there is an error within the function. For example, the ledPin or the ledPort is out of their range.

LED_SUCCESS

If the LED is initialized correctly to be an ouput without any errors.

```
En_ledError_t LED_on(En_dioPort_t ledPort, En_dioPin_t ledPin);
```

2.2.5.2.4 Description

This function turns on the LED connected to a specific DIO pin.

2.2.5.2.5 Input Parameters

2.2.5.2.5.1 ledPort

this variable represents the DIO port having the pin to which the LED is connected.

2.2.5.2.5.2 ledPin

this variable represents the DIO pin to which the LED is connected.

2.2.5.2.6 Return

LED_FAIL

If there is an error within the function. For example, the ledPin or the ledPort is out of their range.

LED_SUCCESS

If the LED is initialized correctly to be an output without any errors.

```
En_ledError_t LED_off(En_dioPort_t ledPort, En_dioPin_t ledPin);
```

2.2.5.2.7 Description

This function turns off the LED connected to a specific DIO pin.

2.2.5.2.8 Input Parameters

2.2.5.2.8.1 ledPort

this variable represents the DIO port having the pin to which the LED is connected.

2.2.5.2.8.2 ledPin

this variable represents the DIO pin to which the LED is connected.

2.2.5.2.9 Return

LED_FAIL

If there is an error within the function. For example, the ledPin or the ledPort is out of their range.

LED_SUCCESS

If the LED is initialized correctly to be an output without any errors.

```
En_ledError_t LED_toggle(En_dioPort_t ledPort, En_dioPin_t ledPin);
```

2.2.5.2.10 Description
This function toggles the LED connected to a specific DIO pin.

2.2.5.2.11 Input Parameters

2.2.5.2.11.1 ledPort
this variable represents the DIO port having the pin to which the LED is connected.

2.2.5.2.11.2 ledPin
this variable represents the DIO pin to which the LED is connected.

2.2.5.2.12 Return
LED_FAIL

If there is an error within the function. For example, the ledPin or the ledPort is out of their range.

LED_SUCCESS

If the LED is initialized correctly to be an output without any errors.

```
En_ledError_t LED_read(En_dioPort_t ledPort, En_dioPin_t ledPin, uint8_t* data);
```

2.2.5.2.13 Description
This function reads the state of the LED If it’s HIGH/ON or LOW/OFF and stores it in the pointer data.

2.2.5.2.14 Input Parameters

2.2.5.2.14.1 ledPort
This variable represents the DIO port having the pin to which the LED is connected.

2.2.5.2.14.2 ledPin
this variable represents the DIO pin to which the LED is connected.

2.2.5.2.15 Data
This pointer stores the state of the LED.

2.2.5.2.16 Return
LED_FAIL

If there is an error within the function. For example, the ledPin or the ledPort is out of their range.

LED_SUCCESS

If the LED is initialized correctly to be an output without any errors.

2.2.6 Button Driver

2.2.6.1 Data Types

```
typedef enum
{
        BUTTON_FAIL,
        BUTTON_SUCCESS

}En_buttonError_t;
```

2.2.6.2 API

```
En_buttonError_t BUTTON_init(En_dioPort_t buttonPort, En_dioPin_t buttonPin);
En_buttonError_t BUTTON_read(En_dioPort_t buttonPort, En_dioPin_t buttonPin,
uint8_t* value);
```

```
En_buttonError_t BUTTON_init(En_dioPort_t buttonPort, En_dioPin_t buttonPin);
```

2.2.6.2.1 Description

Each button is connected to a specific pin of a specific port. Button is an input device, so this function initializes the pin to which the button is connected to be input.

2.2.6.2.2 Input Parameters

2.2.6.2.2.1 buttonPort

This variable represents the DIO port having the pin to which the button is connected.

2.2.6.2.2.2 buttonPin

this variable represents the DIO pin to which the button is connected.

2.2.6.2.3 Return

BUTTON_FAIL

If there is an error within the function. For example, the buttonPin or the buttonPort is out of their range.

BUTTON_SUCCESS

If the button is initialized correctly to be an input without any errors.

```
En_buttonError_t BUTTON_read(En_dioPort_t buttonPort, En_dioPin_t buttonPin,
uint8_t* value);
```

2.2.6.2.4 Description
This function reads the state of the button If it’s pressed or not and stores it in the pointer value.

2.2.6.2.5 Input Parameters

2.2.6.2.5.1 buttonPort
This variable represents the DIO port having the pin to which the button is connected.

2.2.6.2.5.2 buttonPin
this variable represents the DIO pin to which the button is connected.

2.2.6.2.5.3 Value
This pointer stores the state of the button.

2.2.6.2.6 Return
BUTTON_FAIL

If there is an error within the function. For example, the buttonPin or the buttonPort is out of their range.

BUTTON_SUCCESS

If the button is initialized correctly to be an input without any errors.

2.2.7 Delay Driver

2.2.7.1 Data types
```
typedef enum
{
        DELAY_FAIL,
        DELAY_SUCCESS
}En_delayError_t;
```

2.2.7.2 API
```
void setDelayInMsec(uint16_t delay);
```

2.2.7.2.1 Description
This function generate a delay in millisecond.

2.2.7.2.2 Input parameters

2.2.7.2.2.1 Delay
This variable represents the amount of needed delay in milliseconds.

2.2.7.2.3 Return

DELAY_FAIL

If there is an error within the function. For example, the delay is less than 0 or greater than $2^{16} - 1$.

DELAY_SUCCESS

If the delay is generated correctly without any errors.

2.2.8 Interrupt Driver

2.2.8.1 Data Types

```
typedef enum
{
        INTERRUPT_FAIL,
        INTERRUPT_SUCCESS
}En_interruptError_t;
```

2.2.8.2 API

```
En_interruptError_t INTERRUPT_GIE_enable(void);
En_interruptError_t INTERRUPT_GIE_disable(void);
En_interruptError_t INTERRUPT_EXTI_int0Init(void);
En_interruptError_t INTERRUPT_EXTI_int0SetCallback(void(*int0Func)(void));
En_interruptError_t INTERRUPT_EXTI_int0SetInterruptTrigger(En_interruptTrigger_t
interruptTrigger);
```

```
En_interruptError_t INTERRUPT_GIE_enable(void);
```

2.2.8.2.1 Description

This function enables the interrupts globally.

2.2.8.2.2 Input Parameters

There are no input parameters for this function.

2.2.8.2.3 Return

INTERRUPT_FAIL

If there is any error within the function

INTERRUPT_SUCCESS

If the global interrupts are enabled successfully

```
En_interruptError_t INTERRUPT_GIE_disable(void);
```

2.2.8.2.4 Description

This function disables the interrupts globally.

2.2.8.2.5 Input Parameters

There are no input parameters for this function.

2.2.8.2.6 Return

INTERRUPT_FAIL

If there is any error within the function

INTERRUPT_SUCCESS

If the global interrupts are disabled successfully.

```
En_interruptError_t INTERRUPT_EXTI_int0Init(void)
```

2.2.8.2.7 Description

This function enables the external interrupt INT0.

2.2.8.2.8 Input parameters

There are no input parameters for this function.

2.2.8.2.9 Return

INTERRUPT_FAIL

If there is any error within the function

INTERRUPT_SUCCESS

If the function enables the external interrupt INT0 successfully.

```
En_interruptError_t INTERRUPT_EXTI_int0SetCallback(void(*int0Func)(void))
```

2.2.8.2.10 Description

This function sets upper layer function to be call backed from the interrupt service routing related to the external interrupt INT0.

2.2.8.2.11 Input parameters

2.2.8.2.11.1 Int0Func

It's a function pointer that carries the address of the function to be call backed when the interrupt happens.

2.2.8.2.12 Return

INTERRUPT_FAIL

If there is any error within function. For example, the function pointer is NULL.

INTERRUPT_SUCCESS

If there is no error.

```
En_interruptError_t INTERRUPT_EXTI_int0SetInterruptTrigger(En_interruptTrigger_t
interruptTrigger);
```

2.2.8.2.13 Description

This function sets the trigger of the external interrupts INT0 to be triggered on rising edge, falling edge, or low-level.

2.2.8.2.14 Input parameters

interruptTrigger

it represents the trigger type of the external interrupt.

2.2.8.2.15 Return

INTERRUPT_FAIL

If there is any error within the function. For example, the interruptTrigger value doesn't exist.

INTERRUPT_SUCCESS

If there is no error within the function.

2.2.9 App Driver

2.2.9.1 Data types

```
typedef enum
{
        APP_FAIL,
        APP_SUCCESS
}En_appError_t;
```

2.2.9.2 API

```
En_appError_t APP_init(void);
En_appError_t APP_start(void);
```

```
En_appError_t APP_init(void);
```

2.2.9.2.1 Description

This function does all the hardware initialization for the app to work properly when started.

2.2.9.2.2 Input Parameters

There are no input parameters.

2.2.9.2.3 Return

APP_FAIL

If there is any error happening during initialization.

APP_SUCCESS

The app is initialized successfully.

```
En_appError_t APP_start(void)
```

2.2.9.2.4 Descripiton

This function starts the application.

2.2.9.2.5 Input parameters

There are no input parameters.

2.2.9.2.6 Return

APP_FAIL

If there is any error happening during the run time of the application.

APP_SUCCESS

The app is running correctly.

3 SYSTEM STATE TRANSITION DIAGRAM

The system can be visualized using state transition diagram where the system has three states: normal, pedestrian waiting, and pedestrian crossing.

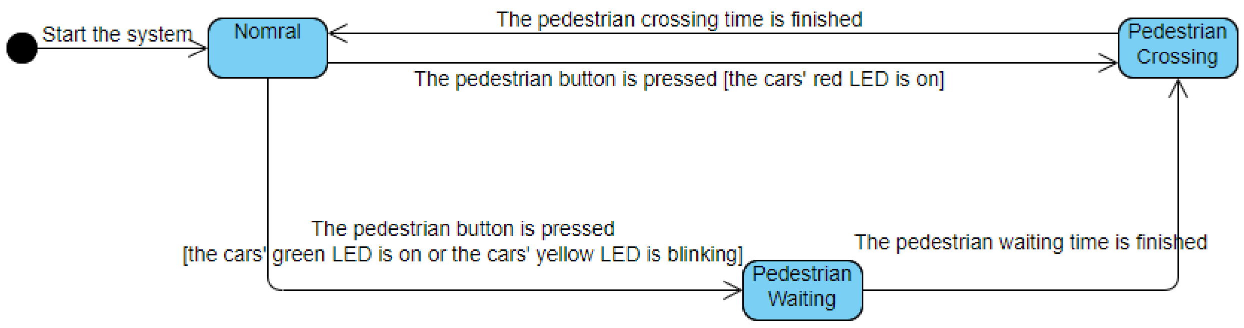

Start the system
Nomral
The pedestrian crossing time is finished
Pedestrian Crossing
The pedestrian button is pressed [the cars' red LED is on]
The pedestrian button is pressed
[the cars' green LED is on or the cars' yellow LED is blinking]
The pedestrian waiting time is finished
Pedestrian Waiting

Figure 2 On-Demand Traffic Light State Transition Diagram

When the system starts, it enters the normal state.

If the pedestrian button is pressed while the cars' red LED is on, the system enters the pedestrian crossing state where the pedestrian is crossing the street.

After the pedestrian crossing time is finished the system returns to the normal state.

If the pedestrian button is pressed while the cars' green LED is on or the cars' yellow LED is blinking, the system enters the pedestrian waiting state where the pedestrian must wait for some time before crossing the street.

After the pedestrian waiting time is finished, the system enters the pedestrian crossing state. After the pedestrian crossing time is finished, the system returns to the normal state.

4 SYSTEM CONSTRAINTS

- You can’t generate time delay more than $2^{16} – 1$ milliseconds in one call to the setDelayInMilliseconds() API.
- You can’t enter the pedestrian states if you pressed the pedestrian mode button while all the cars’ LEDs are OFF.