Traffic Light Control System Design

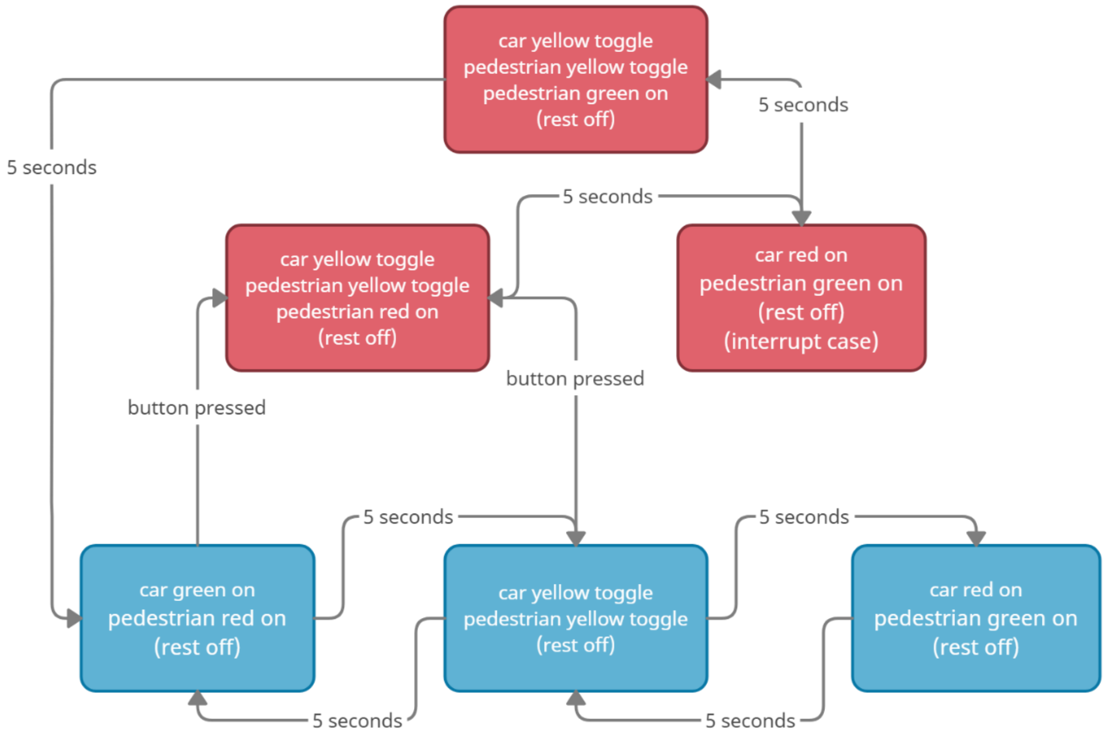
By: Ahmed Hassan Ibrahim

Table of Contents

System Description:

The system is designed to simulate traffic light system showing the flow of traffic light changes for both cars traffic light and pedestrian traffic light. The system actions change when the traffic light button is pressed by pedestrian depending on the state of the lights when pressed.

State Machine:



In the normal mode states colored blue the car lights change green – yellow – red each 5 seconds and the pedestrian traffic lights change opposite to that of car. For the yellow traffic light, it toggles each second. If the button is pressed while the car green or yellow light is on both the car and pedestrian yellow light will toggle and the pedestrian red light will be on at the same time for 5 seconds then the car red light will be on and pedestrian green light will be on allowing pedestrian to cross the road for 5 seconds then both yellow lights will toggle while pedestrian green light is still on for 5 seconds and return back to the normal state. If the button is pressed while car red is on nothing will happen as pedestrian can already cross the road.

System Layers

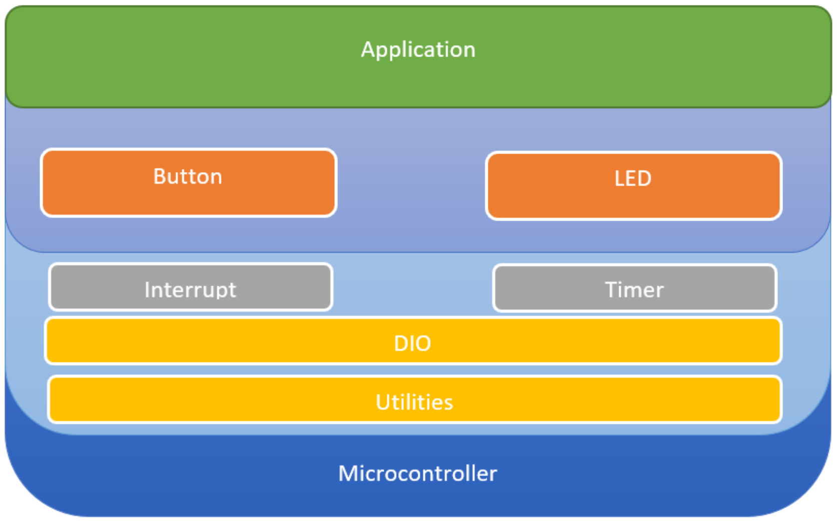
The system is divided to 4 layers:

- Microcontroller
- MCAL
- ECUAL
- Application

System Drivers

The system drivers that are used:

- DIO Driver
- Timer Driver
- Button Driver
- LED Driver
- Utilities for registers and types
- Interrupt library



DIO Driver:

Driver Functions:

void DIO_init(uint8_t pinNumber, uint8_t portNumber, uint8_t direction)

- Initialize dio direction
- Output: High on DDR pinNumber

void DIO_write(uint8_t pinNumber, uint8_t portNumber, uint8_t value)

- Write data to dio
- Output: High on PIN pinNumber

void DIO_toggle(uint8_t pinNumber, uint8_t portNumber)

- Toggle dio
- Output: toggle on PIN pinNumber

void DIO_read(uint8_t pinNumber, uint8_t portNumber, uint8_t *value)

- Read dio
- Output: store the value of the pinNumber in value.

Driver Macros

PORT_A → 'A'

PORT_B → 'B'

PORT_C → 'C'

PORT_D → 'D'

Driver Defines

direction defines for DIO_init function:

IN → 0

OUT → 1

value defines for DIO_write function:

LOW → 0

HIGH → 1

Timer Driver:

Driver Functions:

void TIMER_init()

- Initialize timer
- Output: low on TCCR0 and TCNT0

void DELAY_5_sec ()

- delay 5 seconds
- stopwatch = 5,000,000 us

void DELAY_1_sec ()

- delay 1 second
- stopwatch = 1,000,000 us

Driver Defines

NUMBER_OF_OVERFLOW_5 → 19531 (calculated overflow repeats for 5 second delay)

NUMBER_OF_OVERFLOW_1 → 3906 (calculated overflow repeats for 1 second delay)

LED Driver:

Driver Functions:

void LED_init(uint8_t ledPort, uint8_t ledPin)

- Initialize LED
- Output: High on DDR ledPin

void LED_on(uint8_t ledPort, uint8_t ledPin)

- Turn on the LED
- Output: High on PIN ledPin

void LED_off(uint8_t ledPort, uint8_t ledPin)

- Turn off the LED
- Output: low on PIN ledPin

void LED_toggle(uint8_t ledPort, uint8_t ledPin)

- Toggle the LED
- Output: toggle PIN ledPin

Button Driver:

Driver Functions:

void BUTTON_init(uint8_t buttonPort, uint8_t buttonPin)

- Initialize button
- Output: low on DDR buttonPin

void BUTTON_read(uint8_t buttonPort, uint8_t buttonPin, uint8_t *value)

- Read button
- Output: store the value of the buttonPin in value.

Driver Macros:

BUTTON_1_PORT → PORT_D

BUTTON_1_PIN → 2

LOW → 0 (state Macros)

HIGH → 1 (state Macros)

Interrupt Library:

Driver Defines

EXT_INT_0 __vector_1   (External Interrupt Requests 0)

EXT_INT_1 __vector_2   (External Interrupt Requests 1)

EXT_INT_2 __vector_3   (External Interrupt Requests 2)

sei() __asm__ __volatile__ ("sei" ::: "memory")   (set global interrupts)

cli() __asm__ __volatile__ ("cli" ::: "memory")   (clear global interrupts)

ISR(INT_VECT)void INT_VECT(void) __attribute__ ((signal,used));\

void INT_VECT(void)   (ISR definition)

Utilities:

Registers defines:

(PIN A register)

PORTA *((volatile uint8_t*)0x3B)

DDRA *((volatile uint8_t*)0x3A)

PINA *((volatile uint8_t*)0x39)

(PIN B register)

PORTB *((volatile uint8_t*)0x38)

DDRB *((volatile uint8_t*)0x37)

PINB *((volatile uint8_t*)0x36)

(PIN C register)

PORTC *((volatile uint8_t*)0x35)

DDRC *((volatile uint8_t*)0x34)

PINC *((volatile uint8_t*)0x33)

(PIN D register)

PORTD *((volatile uint8_t*)0x32)

DDRD *((volatile uint8_t*)0x31)

PIND *((volatile uint8_t*)0x30)

(Timer registers)

TCCR0 *((volatile uint8_t*)0x53)

TCNT0 *((volatile uint8_t*)0x52)

TIFR *((volatile uint8_t*)0x58)

(External interrupts registers)

MCUCR *((volatile uint8_t*)0x55)

MCUCSR *((volatile uint8_t*)0x54)

GICR *((volatile uint8_t*)0x5B)

GIFR *((volatile uint8_t*)0x5A)

Types typedef:

unsigned char uint8_t	(unsigned 8-bit character)

Application:

Driver Functions:

void APP_init()

- Initialize all drivers
- Output: run all init functions

void APP_start()

- Start the application
- Output: run the application

Application flags:

unsigned int traffic_status	(flag for button pushed at which state)
unsigned int interrupt_flag	(flag for when car red is on)
unsigned int yellow_blink	(flag to count number of times yellow light blinked)