ON-DEMAND TRAFFIC LIGHT CONTROL

1. System Description

1) Introduction

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.

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.

2) Hardware Requirements

1. ATmega32 microcontroller
2. One push button for pedestrian
3. Three LEDs for cars - Green, Yellow, and Red
4. Three LEDs for pedestrians - Green, Yellow, and Red

3) Software Requirements

There are two modes of operation:

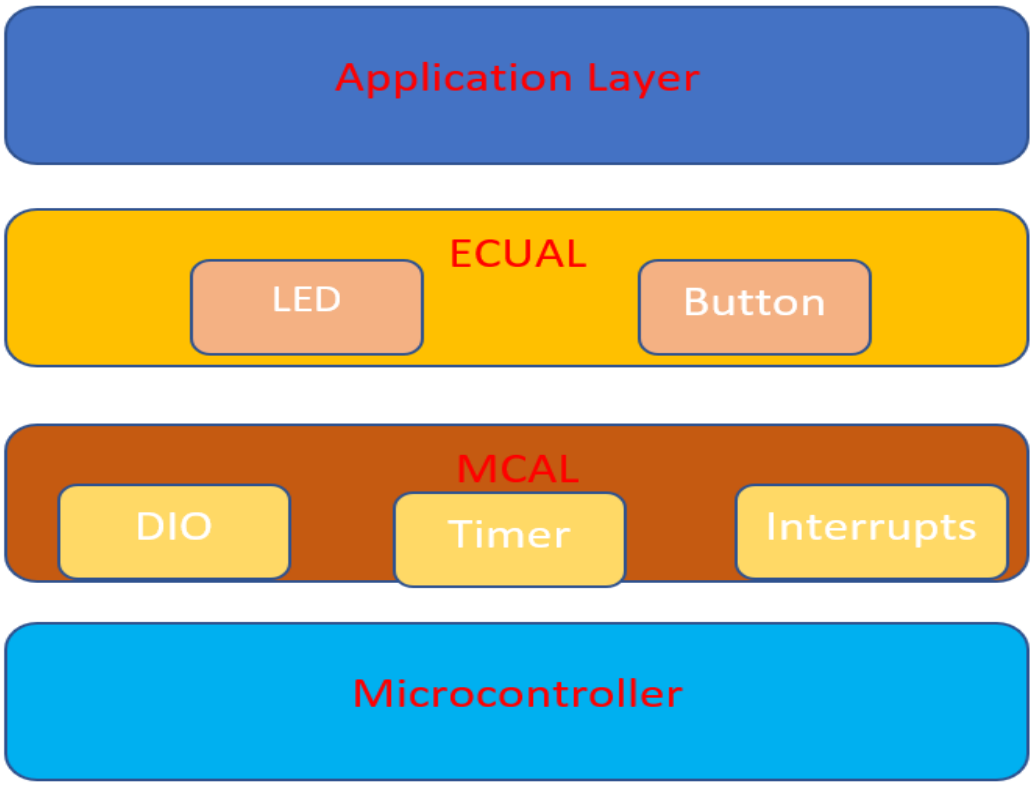
I. Normal mode:
   1. Cars' LEDs will be changed every five seconds starting from Green then yellow then red then yellow then Green.
   2. The Yellow LED will blink for five seconds before moving to Green or Red LEDs.

II. Pedestrian mode:
   1. Change from normal mode to pedestrian mode when the pedestrian button is pressed.
   2. 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. 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. 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. 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. 6.Traffic lights signals are going to the normal mode again.

2. System Design

1) System Layers

1. Microcontroller: physical controller of the system.
2. Microcontroller Abstraction Layer (MCAL): microcontroller dependent drivers.
3. Electronic Unit Abstraction Layer (ECUAL): input / output units' drivers.
4. Application Layer: software logic that control the system.



2) System Drivers

1. LED Driver:
   Initializes LED for a pin within a port. Controls the LED (ON, OFF, TOGGLING, BLINK and BLINK_2LEDS).
2. Button Driver: Initializes BUTTON for a specific pin within a specific port. Get the button status.
3. Digital Input/Output Driver (DIO Driver):
   Initializes any pin in each port to be INPUT or OUTPUT pin. Controls pins in each port to be HIGH, LOW or TOGGLE that pin. To be able get the status of any pin.
4. Interrupts Driver:
   Defines the External Interrupt vectors and Defines the Interrupt Service Routine (ISR) function prototype. Enable/ Disable the global interrupt flag. Initializes interrupts according to the interrupt sense required.
5. Timers Driver:
   Initializes the timer, Defines delay function, Timer on and off function.

3) Drivers APIs

2.3.1. DIO_Driver

```
DIO_status DIO_init  (uint8_t pin_number,uint8_t Port_number,uint8_t pin_direction);
DIO_status DIO_write (uint8_t pin_number,uint8_t Port_number,uint8_t value);
DIO_status DIO_read  (uint8_t pin_number,uint8_t Port_number,uint8_t *value);
DIO_status DIO_toggle(uint8_t pin_number,uint8_t Port_number);
```

2.3.2. Timer_Driver

```
void TimerNormal_init();
void Timer_delay(float delay);
void Timer_start(uint8_t prescaler);
void Timer_off();
```

2.3.3. Interrupts_Driver

```
Interrupt_status INT0_init(uint8_t Sense_Mode);
Interrupt_status INT1_init(uint8_t Sense_Mode);
Interrupt_status INT2_init(uint8_t Sense_Mode);
```

2.3.4. LED_Driver

```
void LED_init(uint8_t LedPin_Number, uint8_t LedPort_Number);
void LED_on(uint8_t LedPin_Number, uint8_t LedPort_Number);
void LED_off(uint8_t LedPin_Number, uint8_t LedPort_Number);
void LED_toggle(uint8_t LedPin_Number, uint8_t LedPort_Number);
void LED_blink(uint8_t LedPin_Number, uint8_t LedPort_Number,uint8_t delay);
void LED_blink_2leds(uint8_t LedPin_Number1, uint8_t LedPort_Number1,uint8_t
LedPin_Number2, uint8_t LedPort_Number2,uint8_t delay);
```

2.3.5. Button_Driver

```
void Button_init(uint8_t ButtonPin_Number, uint8_t ButtonPort_Number);
void Button_read(uint8_t ButtonPin_Number, uint8_t ButtonPort_Number, uint8_t
*value);
```

3. System Constrains

6. System doesn’t do a hardware check
7. Assuming that leds are correctly connected and in good condition
8. Assuming that button is correctly connected