Traffic Light Control

Omar Hamdy Abdelazim

❖ SYSTEM DESCRIPTION

The application that works is a traffic light, allowing pedestrians to press a specific button to cross the road whenever they want.

There are two modes :

1. The normal mode of the traffic light, which is the transition between the green, yellow and red lights every 5 seconds.
2. Pedestrian mode, which allows pedestrians to cross the road when the traffic light for cars is red

If the traffic lights were red, nothing would happen.

If the car signal was yellow, nothing would happen.

If the car signal was green, the signal would turn yellow for 5 seconds, then it would turn red so that pedestrians could cross.

❖ SYSTEM DESIGN

My system is based on Atmega32 micro-controller and contains

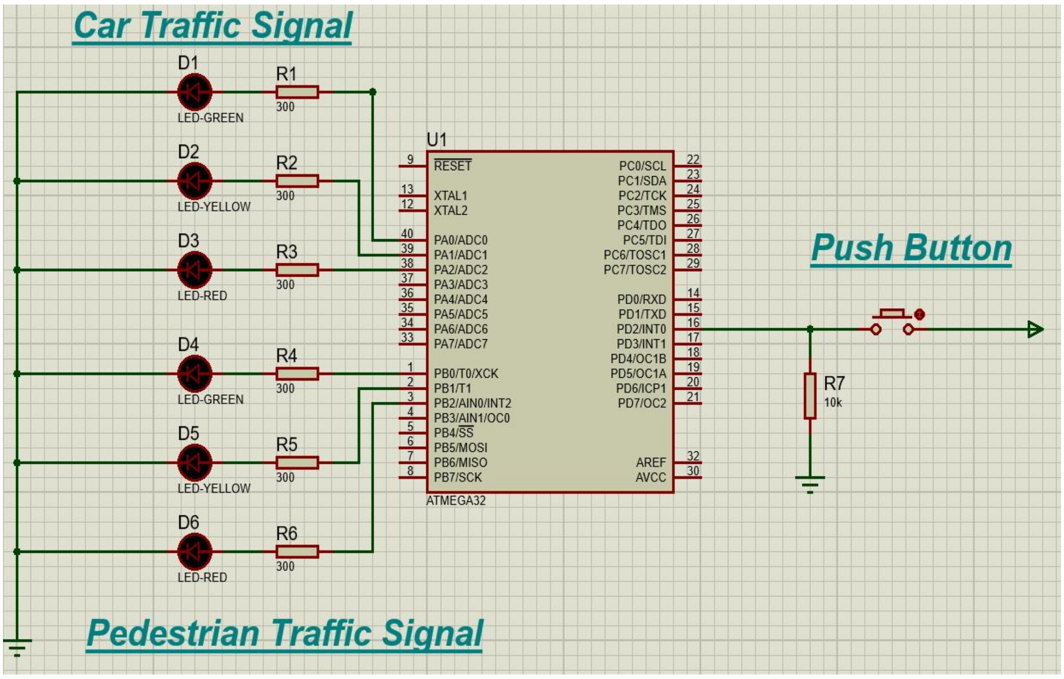
3 Leds for each signal:

Green, Yellow, Red → connected on Port_A on pins 0,1,2 respectively.

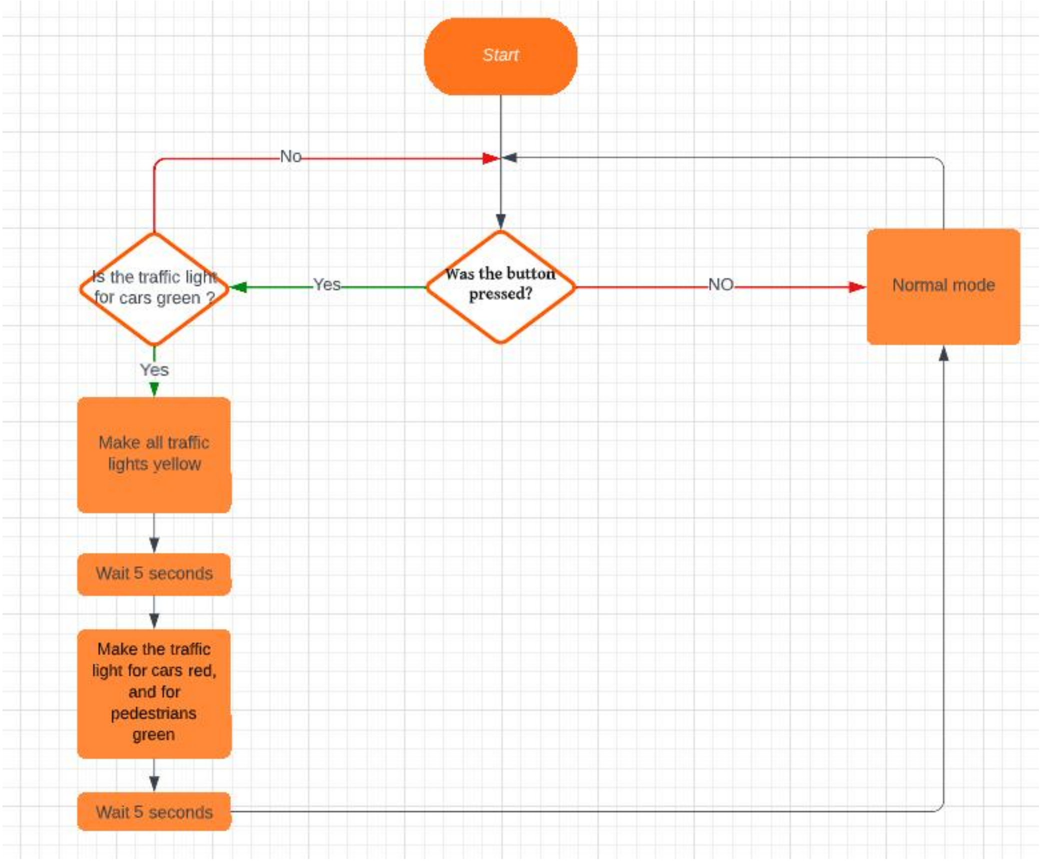
3 Leds for the pedestrian's signal:

Green, Yellow, Red → connected on Port_B on pins 0,1,2 respectively.

One push button → connected on Port_D on pin 2.



❖ SYSTEM FLOW CHART


Start
No
Is the traffic light for cars green ?
Yes
Was the button pressed?
NO
Normal mode
Yes
Make all traffic lights yellow
Wait 5 seconds
Make the traffic light for cars red, and for pedestrians green
Wait 5 seconds

❖ SYSTEM FLOW CONSTRAINTS

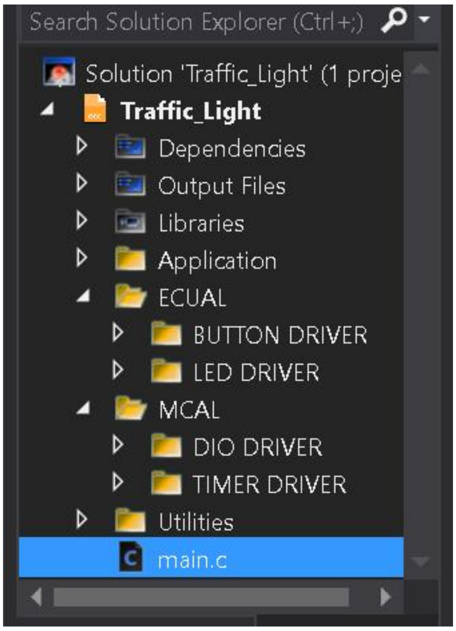
1. Pressing twice on the button will make it respond only to the first press.

2. Long pressing the button will not make it respond to anything.

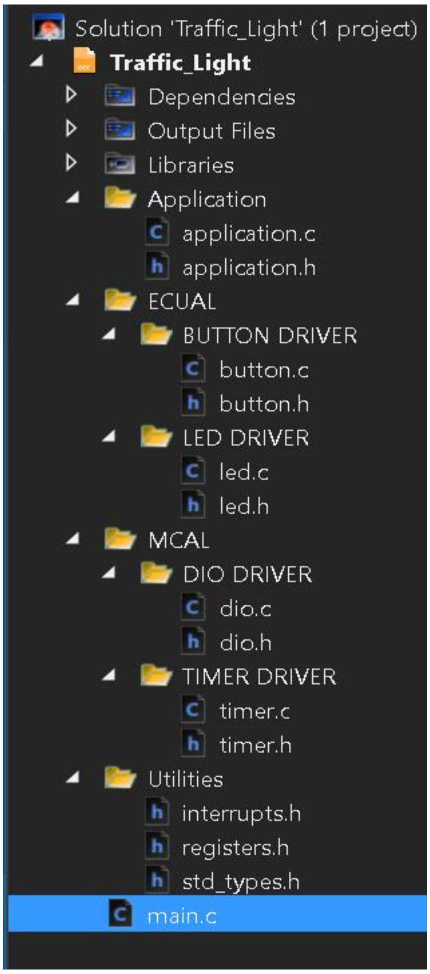
3. Pressing the button when the traffic lights are red or yellow will not make it respond to anything.

❖ DEVELOPMENT ENVIRONMENT PREPARATION

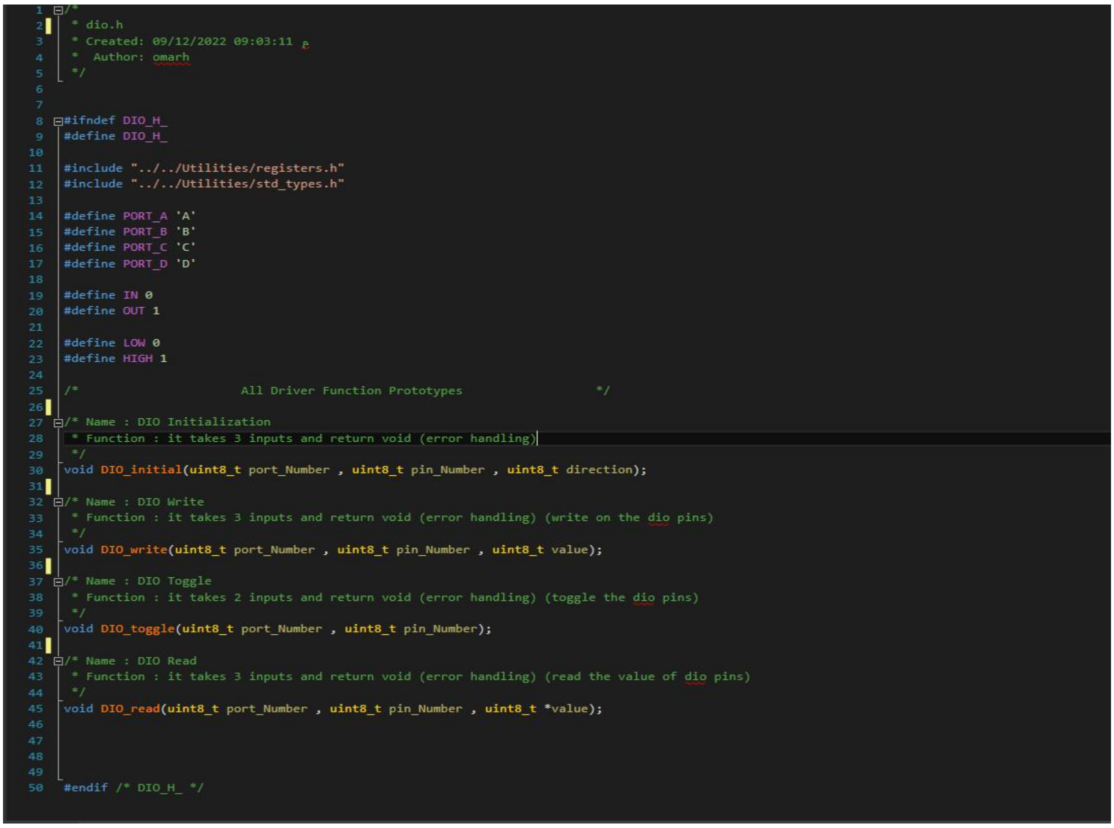
Layer Folder



.c and .h file for each folder

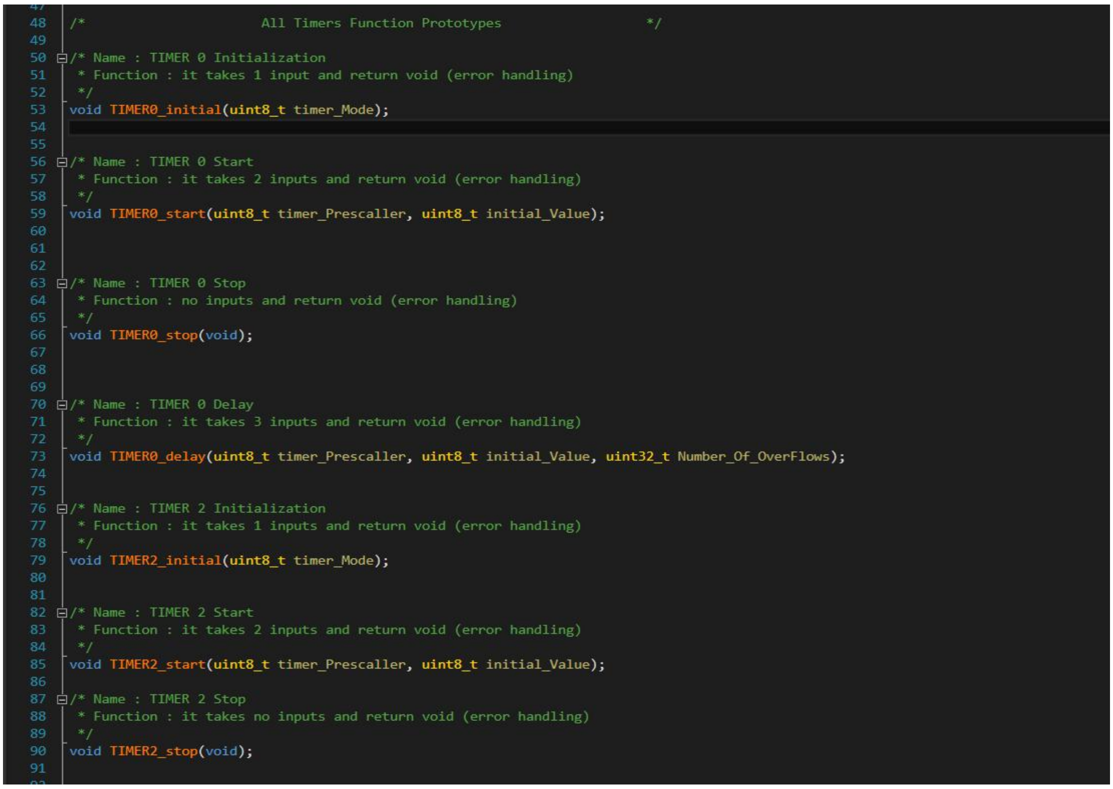


MCAL(DIO.h)



```c
/*
 * dio.h
 * Created: 09/12/2022 09:03:11 e
 *  Author: omarh
 */


#ifndef DIO_H_
#define DIO_H_

#include "../../Utilities/registers.h"
#include "../../Utilities/std_types.h"

#define PORT_A 'A'
#define PORT_B 'B'
#define PORT_C 'C'
#define PORT_D 'D'

#define IN 0
#define OUT 1

#define LOW 0
#define HIGH 1

/*                        All Driver Function Prototypes                  */

/* Name : DIO Initialization
 * Function : it takes 3 inputs and return void (error handling)
 */
void DIO_initial(uint8_t port_Number , uint8_t pin_Number , uint8_t direction);

/* Name : DIO Write
 * Function : it takes 3 inputs and return void (error handling) (write on the dio pins)
 */
void DIO_write(uint8_t port_Number , uint8_t pin_Number , uint8_t value);

/* Name : DIO Toggle
 * Function : it takes 2 inputs and return void (error handling) (toggle the dio pins)
 */
void DIO_toggle(uint8_t port_Number , uint8_t pin_Number);

/* Name : DIO Read
 * Function : it takes 3 inputs and return void (error handling) (read the value of dio pins)
 */
void DIO_read(uint8_t port_Number , uint8_t pin_Number , uint8_t *value);




#endif /* DIO_H_ */
```

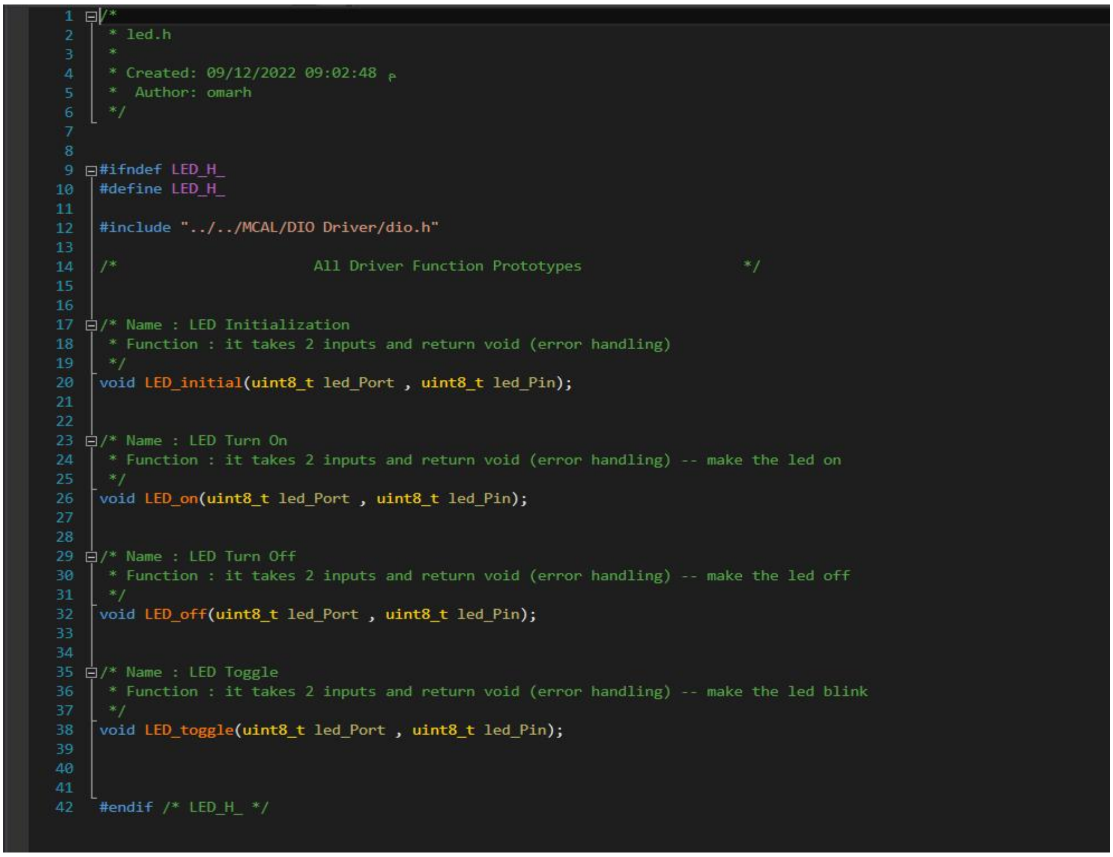
MCAL(TIMER.h)



```c
/*                       All Timers Function Prototypes                  */

/* Name : TIMER 0 Initialization
 * Function : it takes 1 input and return void (error handling)
 */
void TIMER0_initial(uint8_t timer_Mode);


/* Name : TIMER 0 Start
 * Function : it takes 2 inputs and return void (error handling)
 */
void TIMER0_start(uint8_t timer_Prescaller, uint8_t initial_Value);



/* Name : TIMER 0 Stop
 * Function : no inputs and return void (error handling)
 */
void TIMER0_stop(void);



/* Name : TIMER 0 Delay
 * Function : it takes 3 inputs and return void (error handling)
 */
void TIMER0_delay(uint8_t timer_Prescaller, uint8_t initial_Value, uint32_t Number_Of_OverFlows);


/* Name : TIMER 2 Initialization
 * Function : it takes 1 inputs and return void (error handling)
 */
void TIMER2_initial(uint8_t timer_Mode);


/* Name : TIMER 2 Start
 * Function : it takes 2 inputs and return void (error handling)
 */
void TIMER2_start(uint8_t timer_Prescaller, uint8_t initial_Value);

/* Name : TIMER 2 Stop
 * Function : it takes no inputs and return void (error handling)
 */
void TIMER2_stop(void);

```

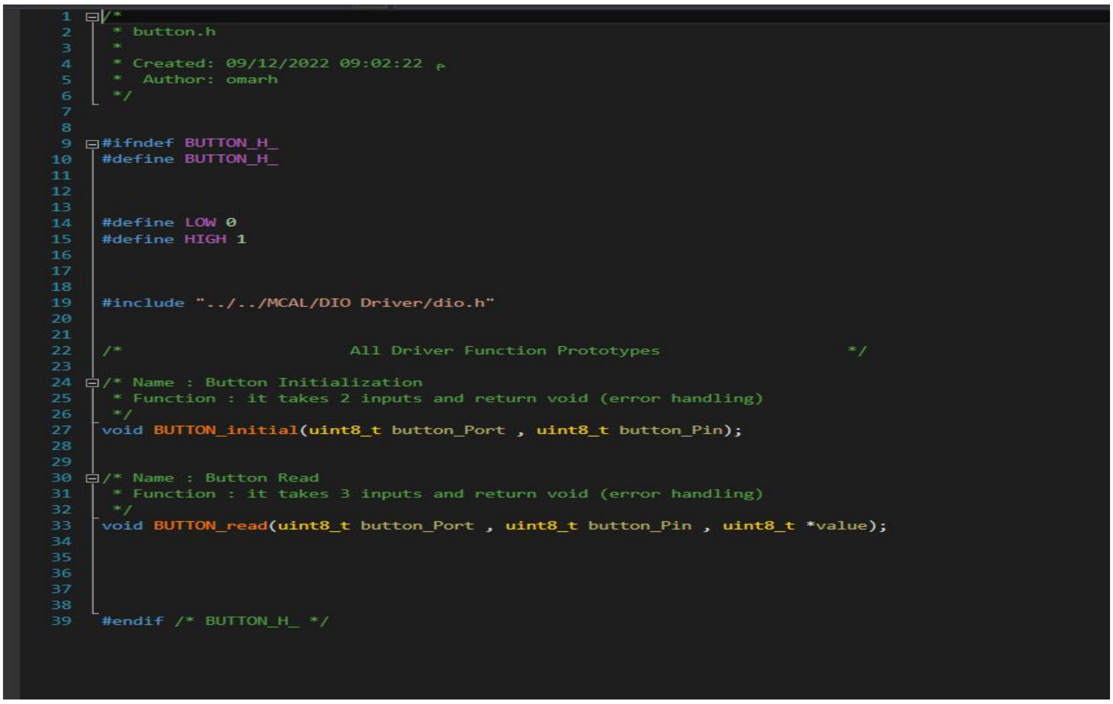
MCAL Implementation : https://youtu.be/5Z_pM7CqMUo

ECUAL(LED.h)



```c
/*
 * led.h
 *
 * Created: 09/12/2022 09:02:48 م
 *  Author: omarh
 */


#ifndef LED_H_
#define LED_H_

#include "../../MCAL/DIO Driver/dio.h"

/*                       All Driver Function Prototypes                  */


/* Name : LED Initialization
 * Function : it takes 2 inputs and return void (error handling)
 */
void LED_initial(uint8_t led_Port , uint8_t led_Pin);


/* Name : LED Turn On
 * Function : it takes 2 inputs and return void (error handling) -- make the led on
 */
void LED_on(uint8_t led_Port , uint8_t led_Pin);


/* Name : LED Turn Off
 * Function : it takes 2 inputs and return void (error handling) -- make the led off
 */
void LED_off(uint8_t led_Port , uint8_t led_Pin);


/* Name : LED Toggle
 * Function : it takes 2 inputs and return void (error handling) -- make the led blink
 */
void LED_toggle(uint8_t led_Port , uint8_t led_Pin);



#endif /* LED_H_ */
```

ECUAL(BUTTONON.h)



```c
/*
 * button.h
 *
 * Created: 09/12/2022 09:02:22 م
 *  Author: omarh
 */


#ifndef BUTTON_H_
#define BUTTON_H_



#define LOW 0
#define HIGH 1



#include "../../MCAL/DIO Driver/dio.h"


/*                       All Driver Function Prototypes                  */

/* Name : Button Initialization
 * Function : it takes 2 inputs and return void (error handling)
 */
void BUTTON_initial(uint8_t button_Port , uint8_t button_Pin);


/* Name : Button Read
 * Function : it takes 3 inputs and return void (error handling)
 */
void BUTTON_read(uint8_t button_Port , uint8_t button_Pin , uint8_t *value);





#endif /* BUTTON_H_ */
```

ECUAL Implementation : https://youtu.be/S9-Dg3lFqI8

❖ TESTING THE APPLICATION

Application and User Story 1: Passed

https://youtu.be/anBQHPeh29w

User Story 2 and 3 : Passed

https://youtu.be/qTC6rP9i9VI

User Story 4 and 5: Passed

https://youtu.be/5AuAzmOzNAk

❖ CONCLUSION

The traffic light was created successfully, the functions we needed were tested, and all the required stories were passed