

|  |
| --- |
| On-demand Traffic Light Control System  2018 |
|  |
| September 13  egFWD Embedded Systems professional track  Authored by: Adel Mostafa Kamel Helal |

# System description

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

Traffic lights consists of three signals transmitting meaningful information through colors to the cars’ drivers and the pedestrians.

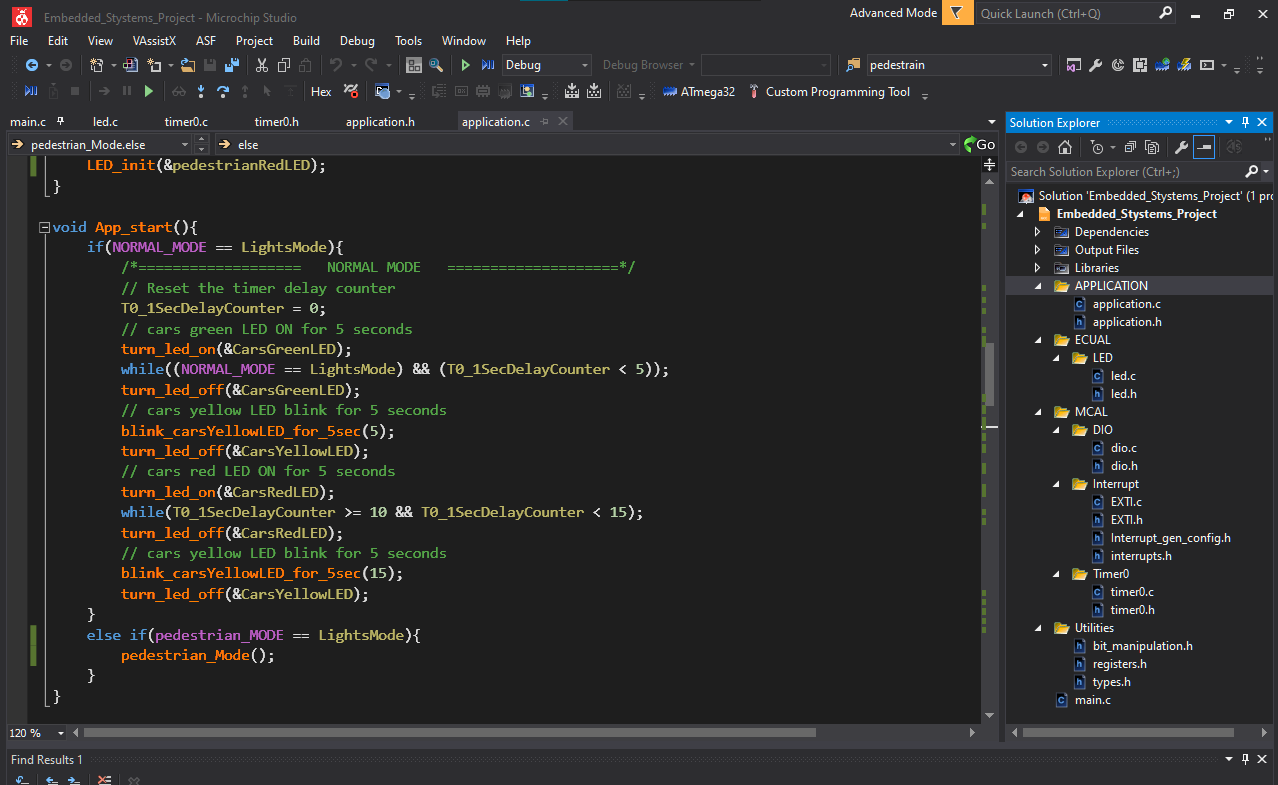
The traffic light colors are red, yellow, and green arranged vertically.

Crosswalk button lets the signal controller know that someone wants to cross the street, so the light signals adjusts, giving the pedestrian enough time to get across.

# System Design

## Software Design

Project’s folder structure is layered architecture with four layers:

1. Application layer:

Has the application code which resides in top.

1. Electrical Unit Abstraction Layer(ECUAL) layer:

Contains the LED driver.

1. Micro Controller Abstraction Layer(MCAL) layer:

Contains the DIO, timer0 and interrupt drivers.

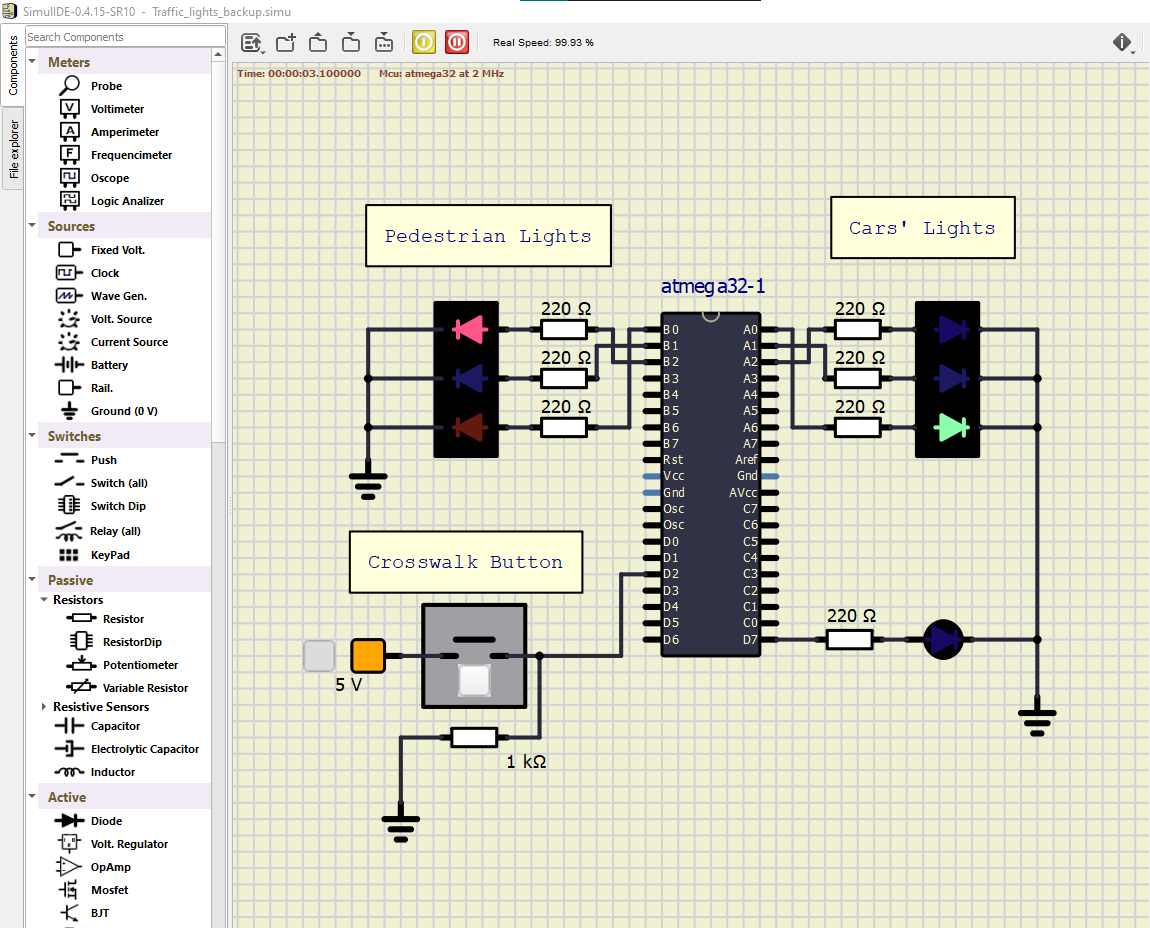
1. Utilities layer:

Contains helper header files to be support the other drivers. And it has types.h, registers.h and bit\_manipulation.h header files.

Each driver contains the used APIs and data types declared in the driver header file, defined and used in the source file.

## Hardware Design

* Atmega32 microcontroller is used with three LEDs for cars signals connected on Port A (Green to Pin 0, Yellow to Pin 1 and Red to Pin 2), three other LEDs for pedestrian signals connected on Port B (Green to Pin 0, Yellow to Pin 1 and Red to Pin 2) and one push button connected to INT0 Pin (Port D, Pin 2) for pedestrian. And another one connected to (Port D, Pin 7) for indicating that an error occurred when testing a driver API manually.
* Simulation is done using SimulIDE software to simulate the traffic light control.

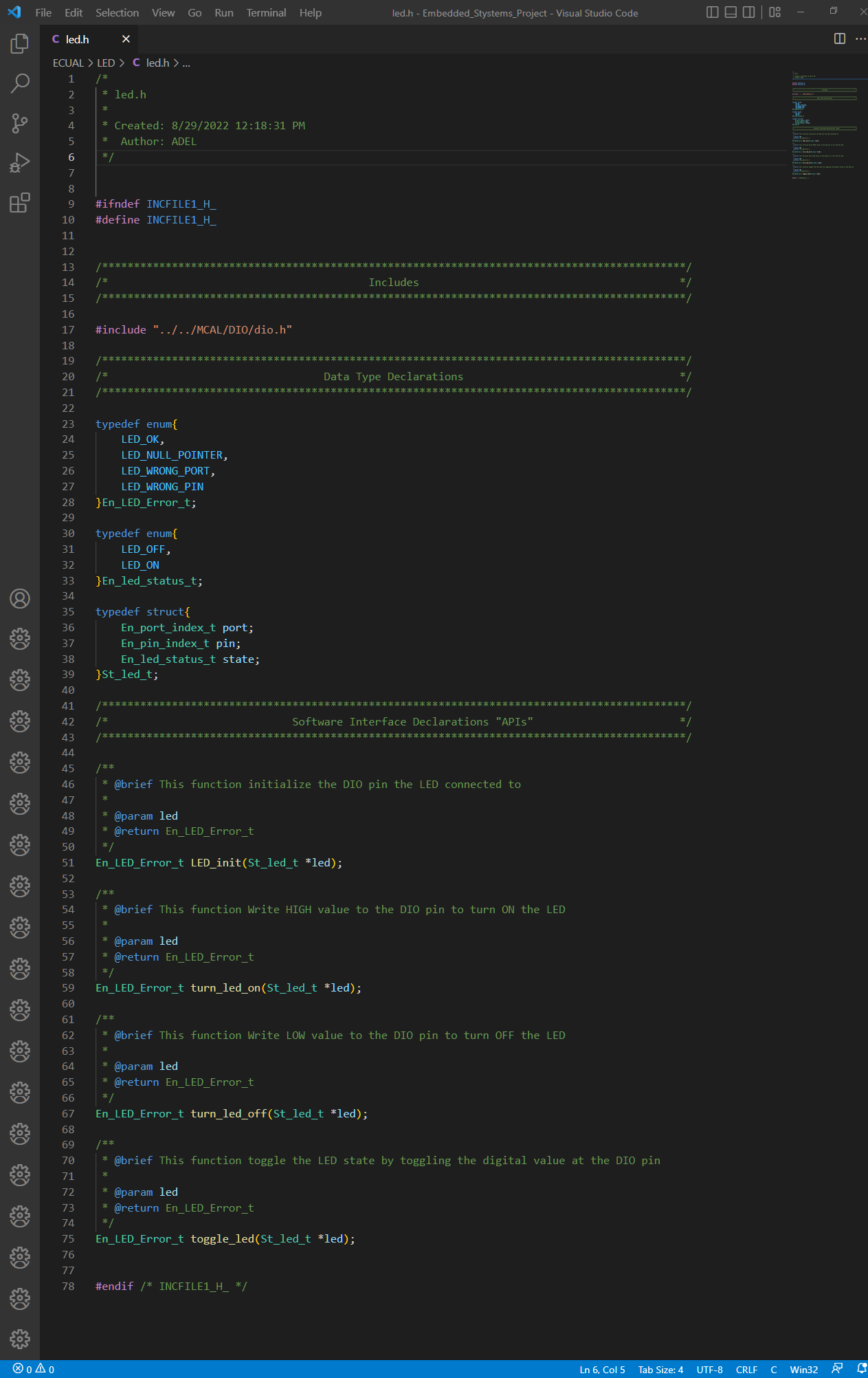


## Implementation

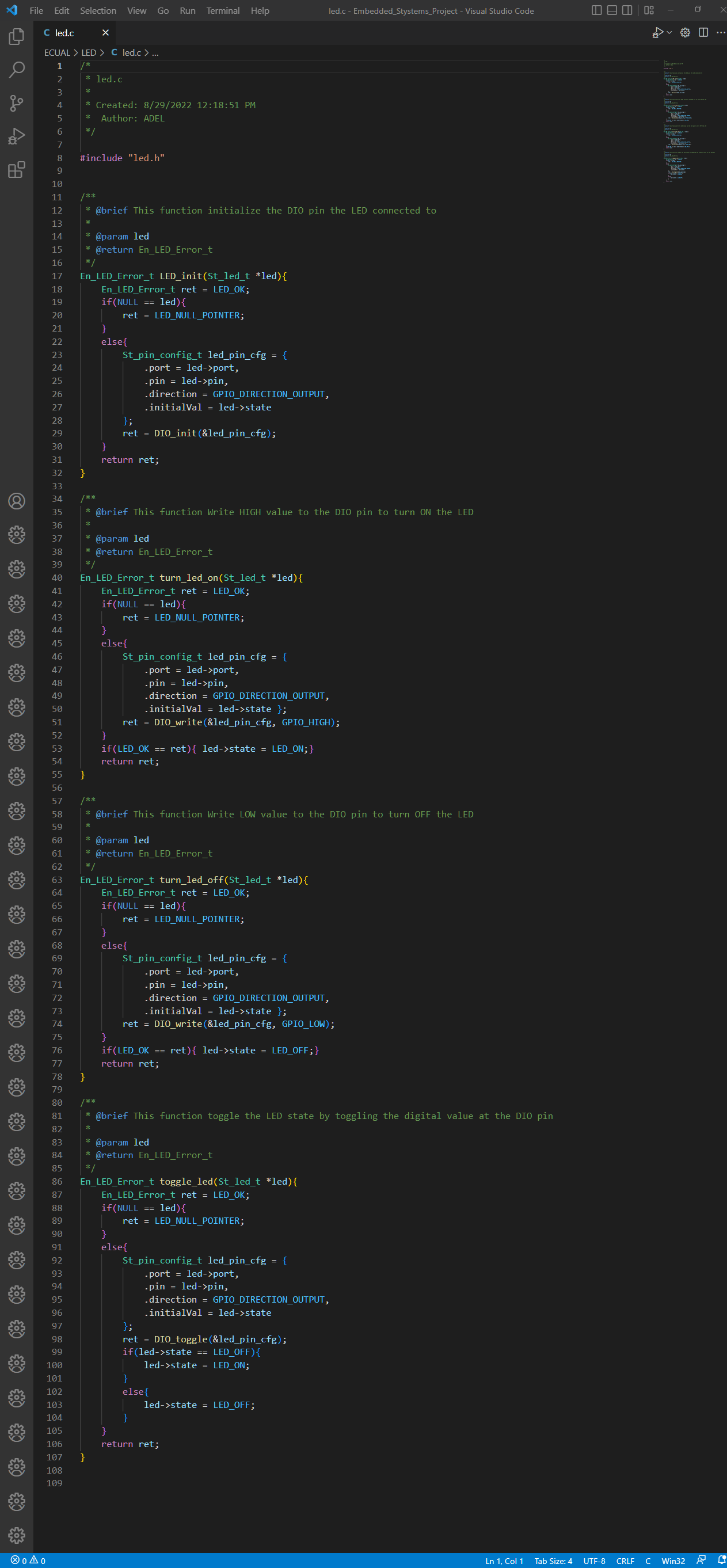
First implemented the MCAL drivers i.e. DIO, Timer0 and Interrupt drivers the I Implemented the ECUAL LED driver. I wrote a skeleton for each function using comments and then converted them into the appropriate code.

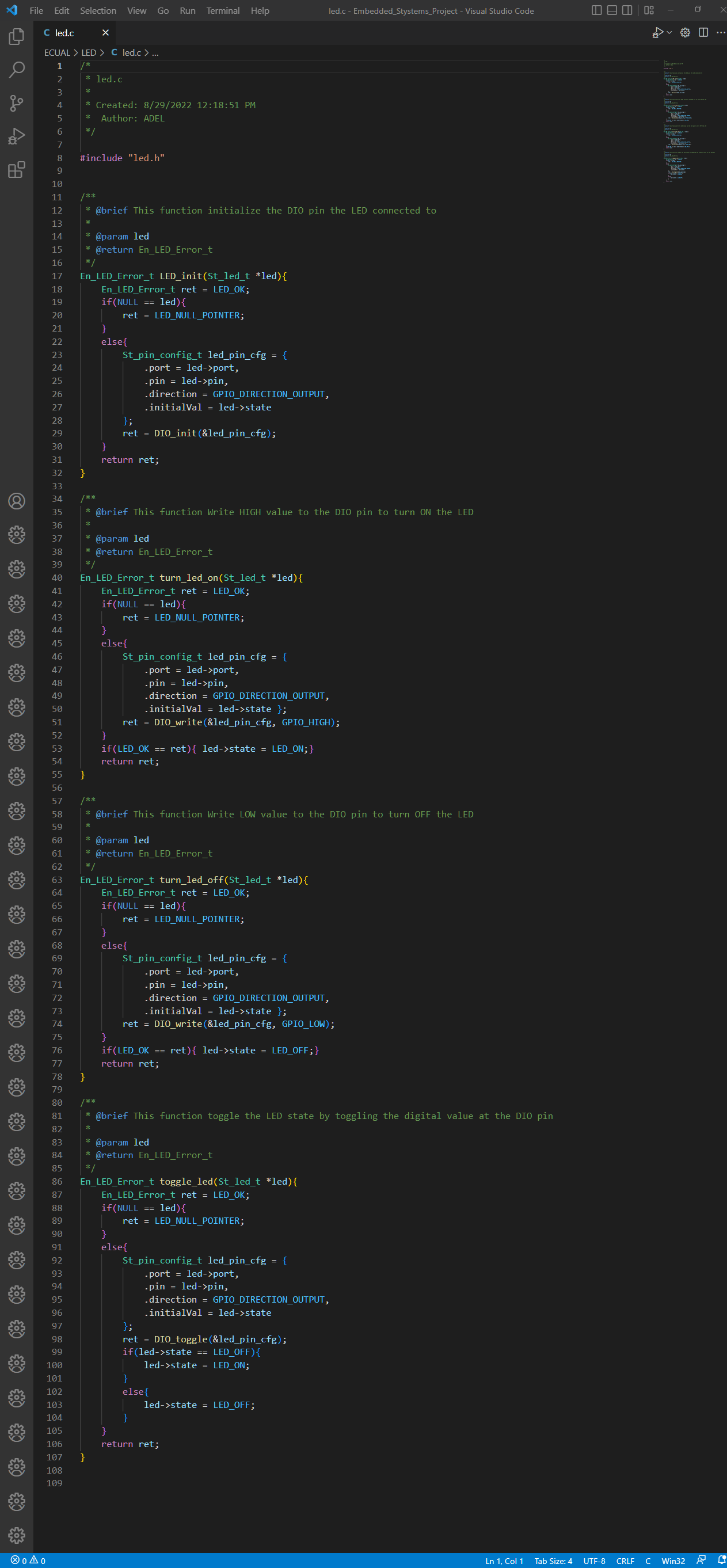
Each driver has at least two files, source file and header file.

The .h header file consists of the inclusion section of the needed header files of the lower layer, macros, and functions like macro, new defined data types and the driver APIs prototypes.



The driver .c source file includes the corresponding driver header file the definition of the driver functions.

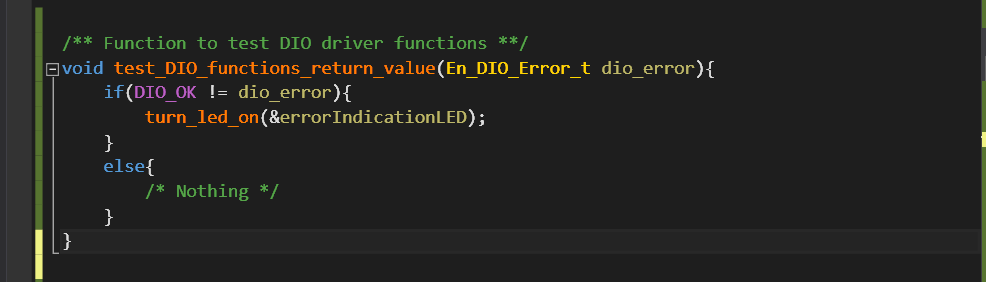




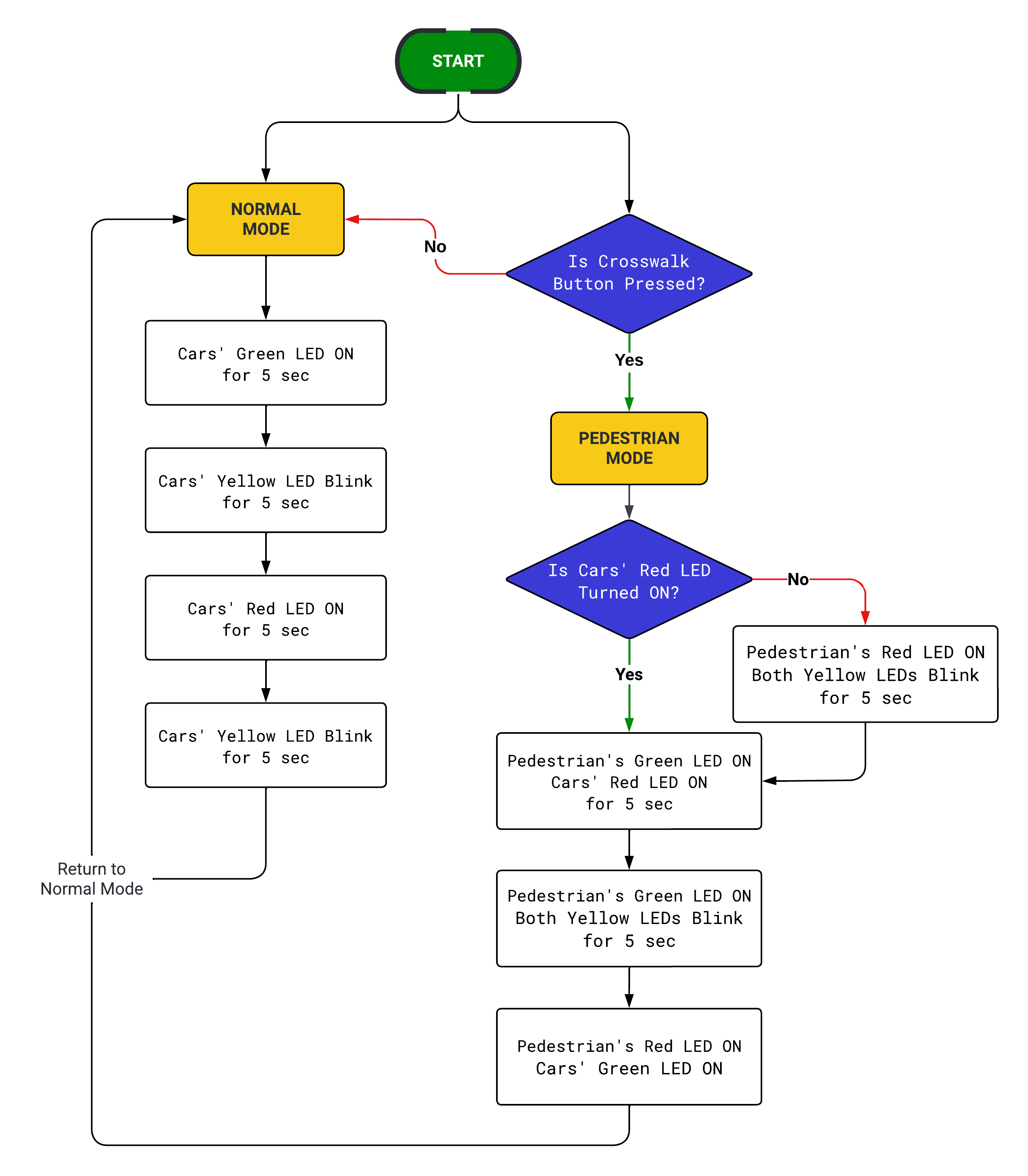
Enum data type is implemented for each driver and each function returns error state to indicate the status of the function process is it done correctly or there was an error occurred.



In the main function you can test specific driver’s functions manually through the driver test module implemented in the main source file by passing the return value of the driver function to the test module and if this value isn’t OK the function will turn the error indication LED.



# System Flow Chart



# System Constraints

There is no constrains on the system.

# Project Video