**Sprints**

**UDACITY**

**egypt fwd** initiative

Embedded Systems Professional Track
August Cohort 2022

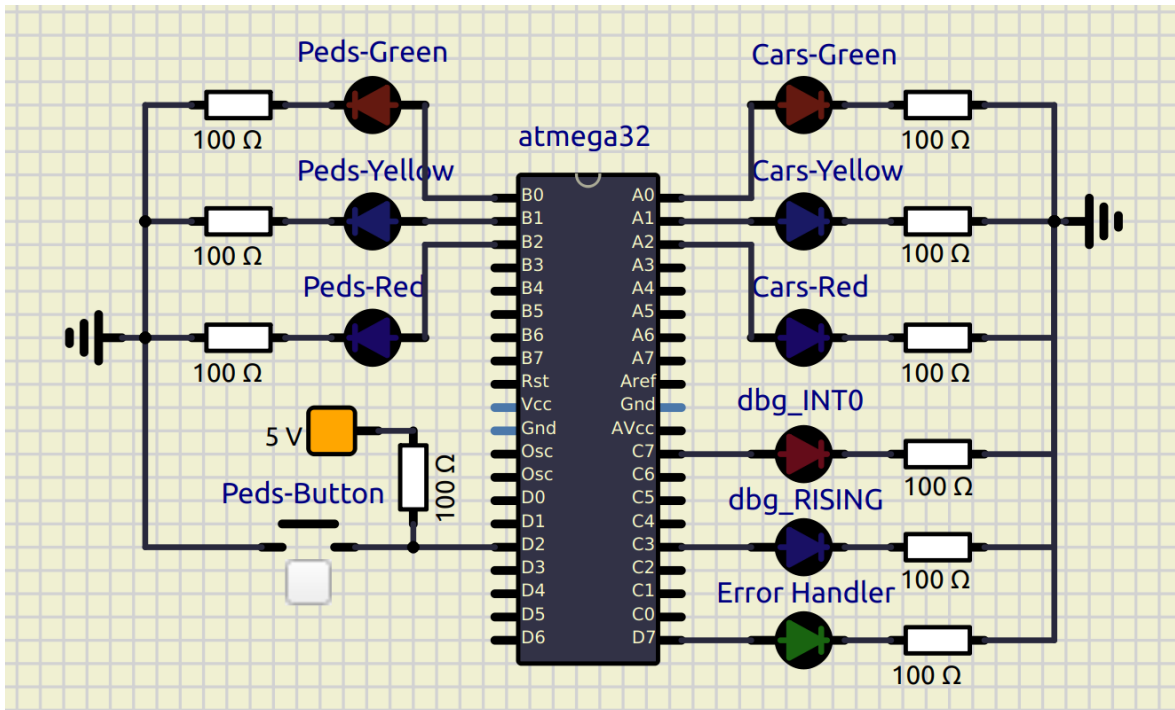# On-demand Traffic Light Control

By
**Youssef Hassan**

Cairo, Egypt

September 2022

# <u>Table of Contents</u>

1. System Description

    a. System Overview
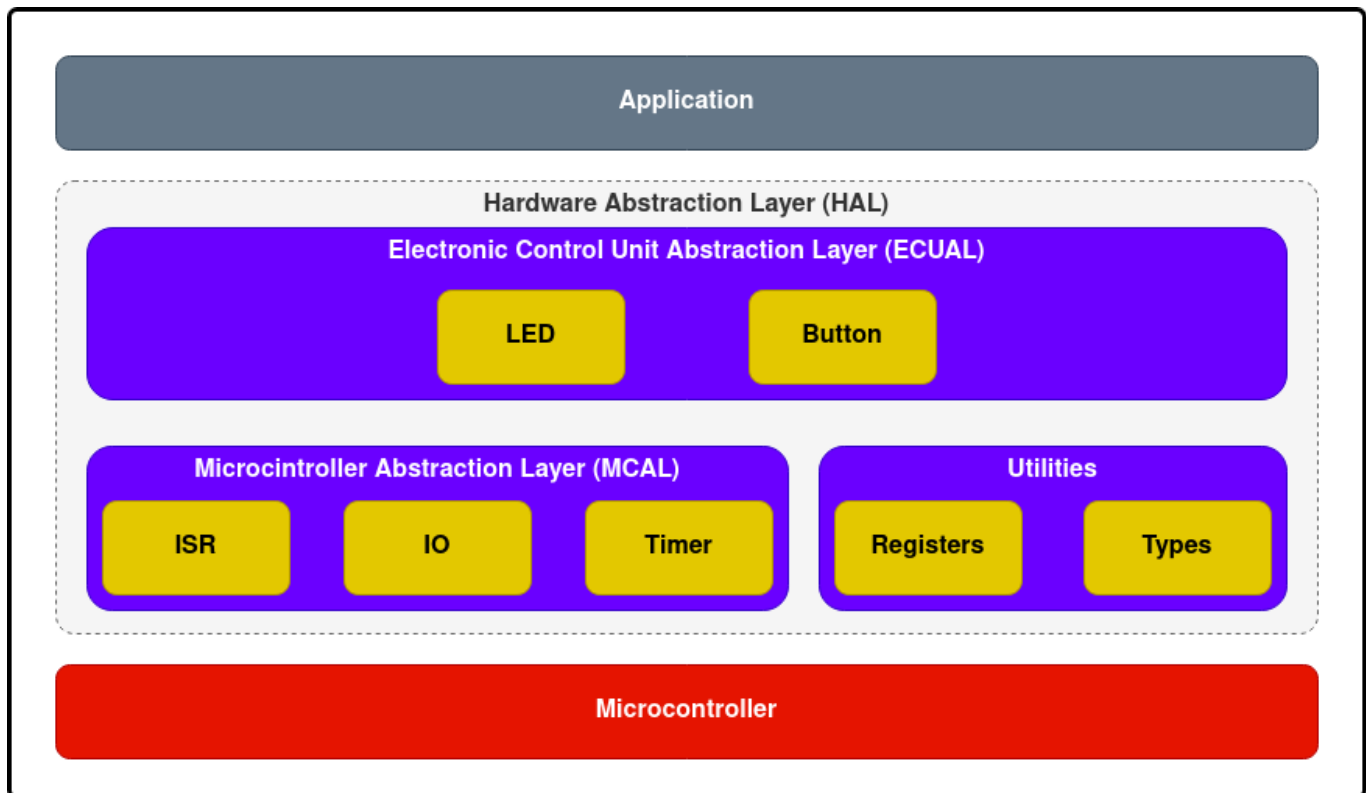


        i. System Mission

            The project aims to provide light traffic solutions for public streets, to help the government regulate traffic congestion.

        ii. Folders Structure

iii. System High-Level Design



b. System Functionality

The application has two modes which are **"normal"** mode and **"pedestrian"** mode. The normal mode will take place as long as the pedestrian button is not pressed, following all the pre-defined user stories.

c. Hardware Design

| # | Components | Quantity |
|---|---|---|
| 1 | AVR Atmega 32 | 1 |
| 2 | Red Led | 2 |
| 3 | Yellow Led | 2 |
| 4 | Green Led | 2 |
| 5 | Push Button | 1 |
| 6 | 100 Ohm Resistor | 10 |

2. System Design
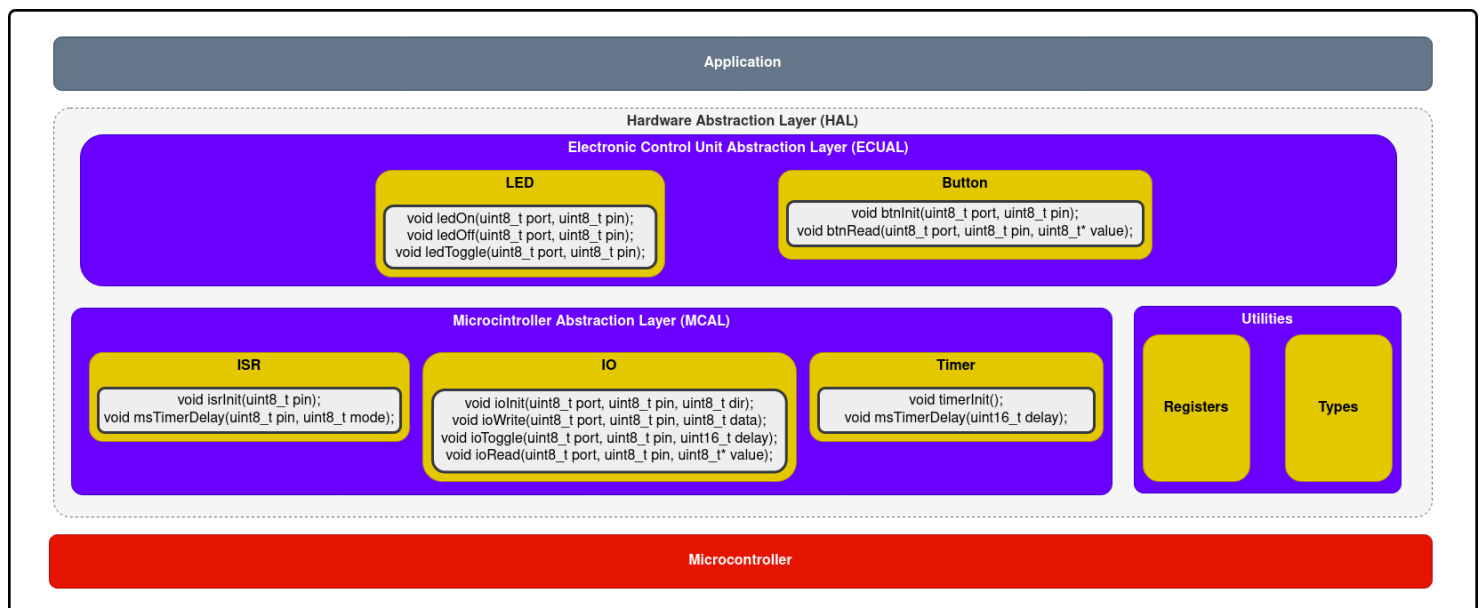
    a. Developing Environment

        i.   VScode as the system IDE.
        ii.  Makefile for compiling c modules and programs.
        iii. Simulide for simulating the circuit.
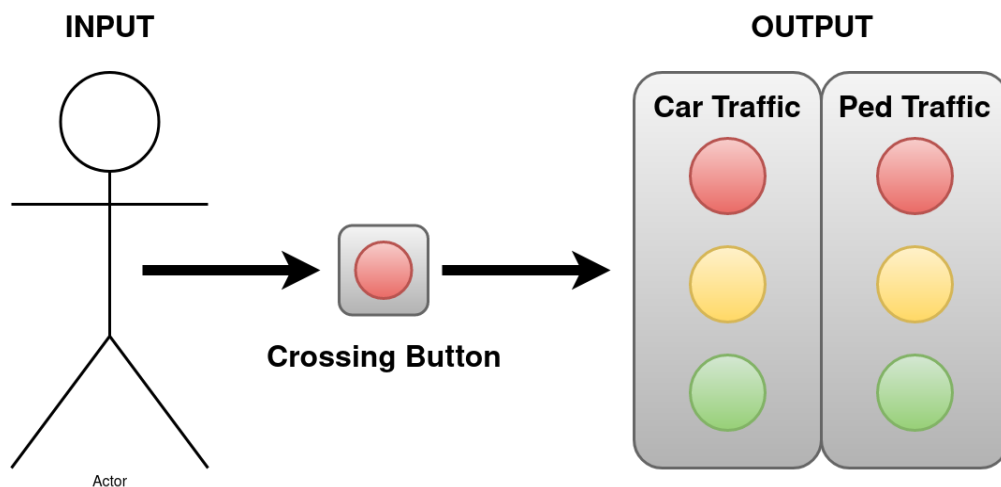        iv.  Git & GitHub for version control and collaboration.

    b. Coding Standards and Guidelines

        i.   Folders and modules naming standards are in lowercase if it's composed of a single word, separated by an underscore if multiple words.
        ii.  Variables are defined in camelCase standard.
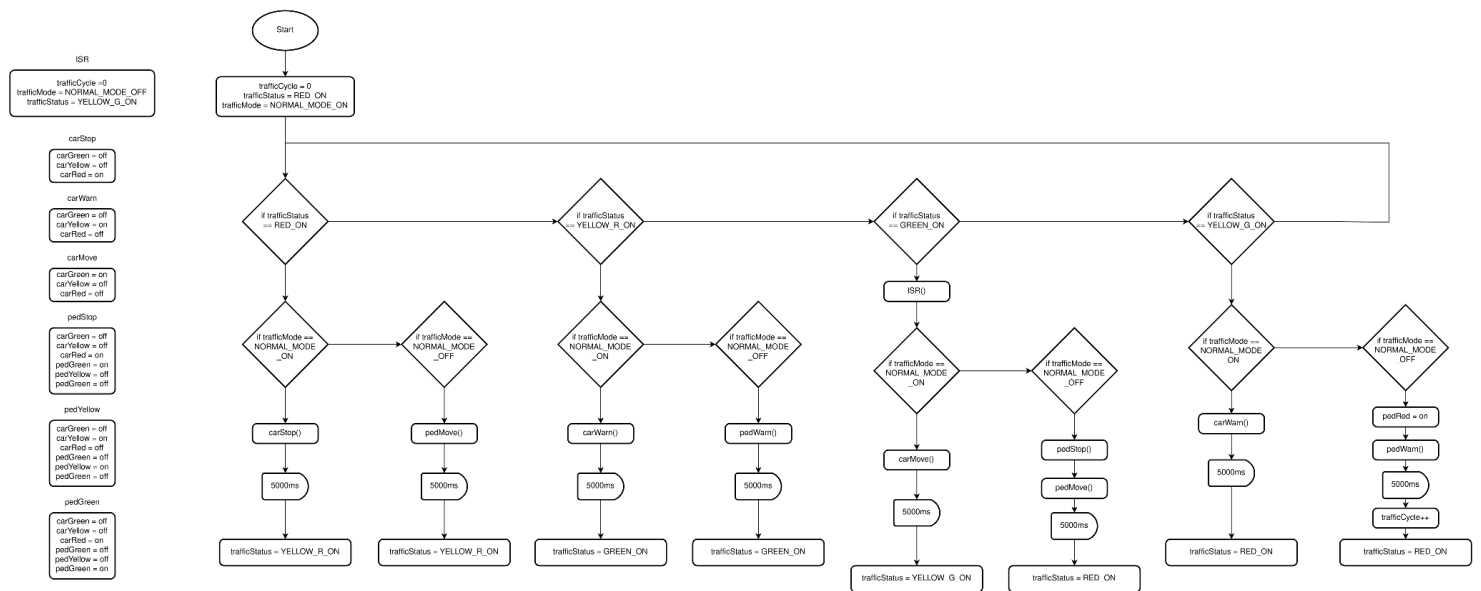        iii. Macros are defined in UPPERCASE standard.

    c. Subsystem Architecture

**Application**

**Hardware Abstraction Layer (HAL)**

**Electronic Control Unit Abstraction Layer (ECUAL)**

**LED**
```
void ledOn(uint8_t port, uint8_t pin);
void ledOff(uint8_t port, uint8_t pin);
void ledToggle(uint8_t port, uint8_t pin);
```

**Button**
```
void btnInit(uint8_t port, uint8_t pin);
void btnRead(uint8_t port, uint8_t pin, uint8_t* value);
```

**Microcintroller Abstraction Layer (MCAL)**

**ISR**
```
void isrInit(uint8_t pin);
void msTimerDelay(uint8_t pin, uint8_t mode);
```

**IO**
```
void ioInit(uint8_t port, uint8_t pin, uint8_t dir);
void ioWrite(uint8_t port, uint8_t pin, uint8_t data);
void ioToggle(uint8_t port, uint8_t pin, uint16_t delay);
void ioRead(uint8_t port, uint8_t pin, uint8_t* value);
```

**Timer**
```
void timerInit();
void msTimerDelay(uint16_t delay);
```

**Utilities**

**Registers**

**Types**

**Microcontroller**

    d. Human Machine Interface

**INPUT**

**OUTPUT**

Actor

**Crossing Button**

**Car Traffic**

**Ped Traffic**

## 3. System Flow Chart



## 4. Design Constraints

    a. Implement your own Timer library.
    b. Do not use the AVR library Timer.
    c. Do not use for-loops as a Timer delay.