On demand traffic light control

This is the second project of the fwd embedded systems professional course

System description

This is a traffic light control system.

The system has 3 traffic lights for the vehicles and another three for the pedestrians, in addition to a crosswalk button. The crosswalk button 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.

There are two modes of operation:

1. Normal mode: In this mode, the traffic lights of the cars operate as usual until the crosswalk button is pressed which transitions the system to the pedestrian mode.
2. Pedestrian mode: In this mode, the red traffic light of the cars is on while the pedestrian traffic lights operate. Once that is done, the system transitions back to the normal mode automatically.

The complete flow is described by the flow chart below

System design

System layers

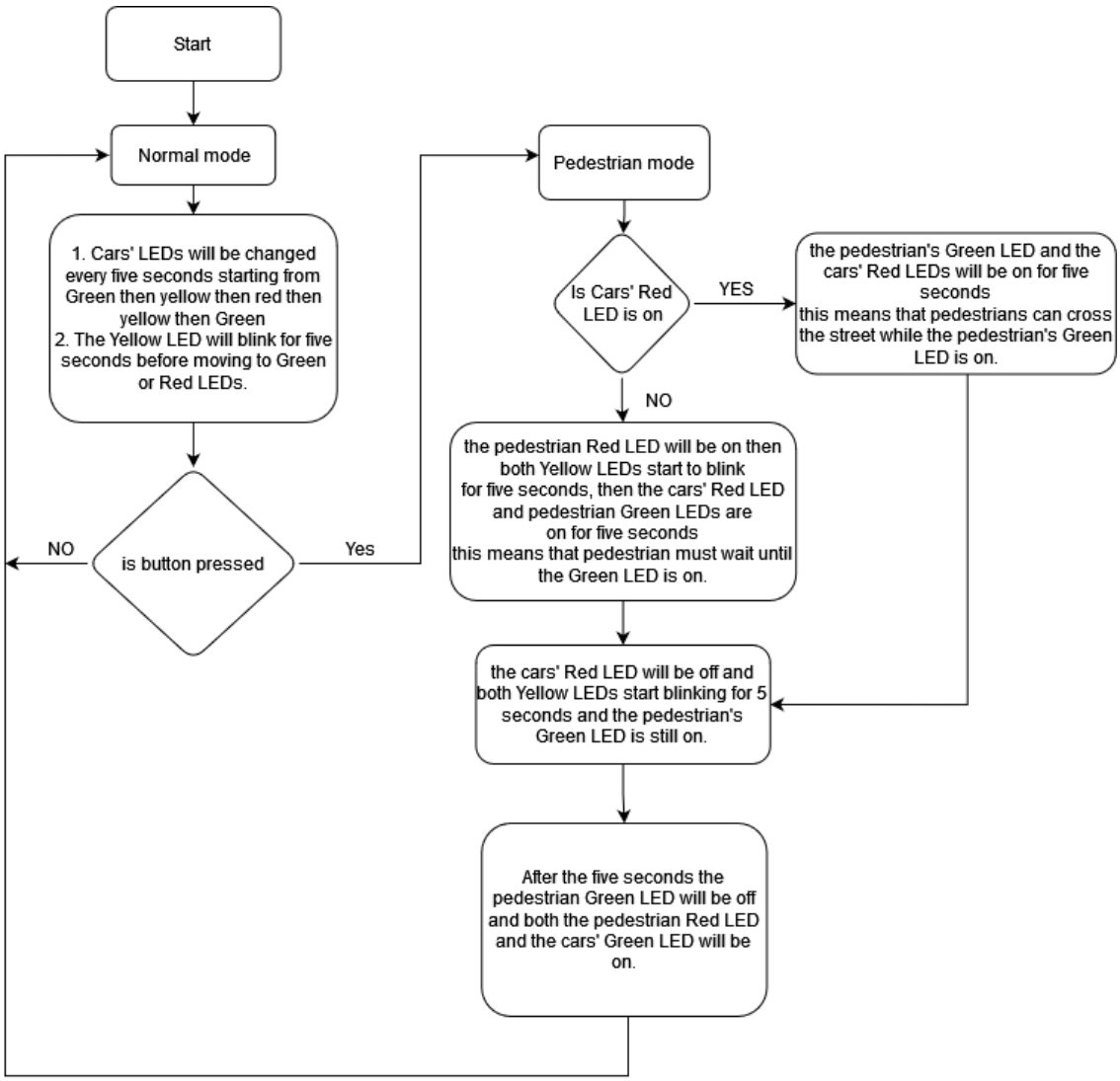
- Application
  - `void APP_init()`
  - `void APP_start()`
- ECUAL (Electronic Unit Abstraction Layer)
  - LED Driver
  - Button Driver
- MCAL (Microcontroller Abstraction Layer)
  - DIO Driver
  - Timer Driver
  - Interrupts Driver
- Utils
  - registers
    - DIO registers
    - Timer0 registers
    - External interrupts registers
  - helpers
    - Bit manipulation macros
    - Common typedefs used across all layers, e.g., `unit8_t`, `uint16_t`
    - Common macros used across all layers, e.g., `HIGH`, `LOW`

System drivers

- DIO Driver

    - `EN_port_t` , `EN_pin_t` , `EN_direction_t` typedefs
    - `EN_error_state DIO_init(EN_port_t port, EN_pin_t pin, EN_direction_t direction)`
    - `EN_error_state DIO_write(EN_port_t port, EN_pin_t pin, uint8_t value)`
    - `EN_error_state DIO_toggle(EN_port_t port, EN_pin_t pin)`
    - `EN_error_state DIO_read(EN_port_t port, EN_pin_t pin, uint8_t *value)`
- Timer Driver
    - `void TIMER_init()`
    - `EN_error_state TIMER_start(uint16_t prescaler, uint8_t initialValue)`
    - `void TIMER_stop()`
    - `EN_error_state TIMER_delay(float delay_in_ms, uint16_t prescaler)`
    - `EN_error_state TIMER_delay_5s()`
    - `EN_error_state TIMER_delay_1s()`
- Interrupts Driver
    - External interrupts vectors macros
    - `ISR` , `sei` macros
    - `void INTERRUPTS_init()`
- LED Driver
    - LEDs ports & pins macros
    - `void LEDS_init()`
    - `EN_error_state LED_on(EN_port_t port, EN_pin_t pin)`
    - `EN_error_state LED_off(EN_port_t port, EN_pin_t pin)`
    - `EN_error_state LED_toggle(EN_port_t port, EN_pin_t pin)`
    - `EN_error_state LED_blink(EN_port_t port, EN_pin_t pin)`
    - `EN_error_state LED_double_blink(EN_port_t port1, EN_pin_t pin1, EN_port_t port2, EN_pin_t pin2)`
    - `EN_error_state LED_on_only(EN_port_t port, EN_pin_t pin)`
- Button Driver
    - Button port & pin macros
    - `void BUTTON_init()`

System flow chart


Start
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.
is button pressed
NO
Yes
Pedestrian mode
Is Cars' Red LED is on
YES
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.
NO
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.
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.
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.