9/13/2022

On-demand Traffic Light Control

Ibrahim Mohamed Hamdy Hassan

1. System Description

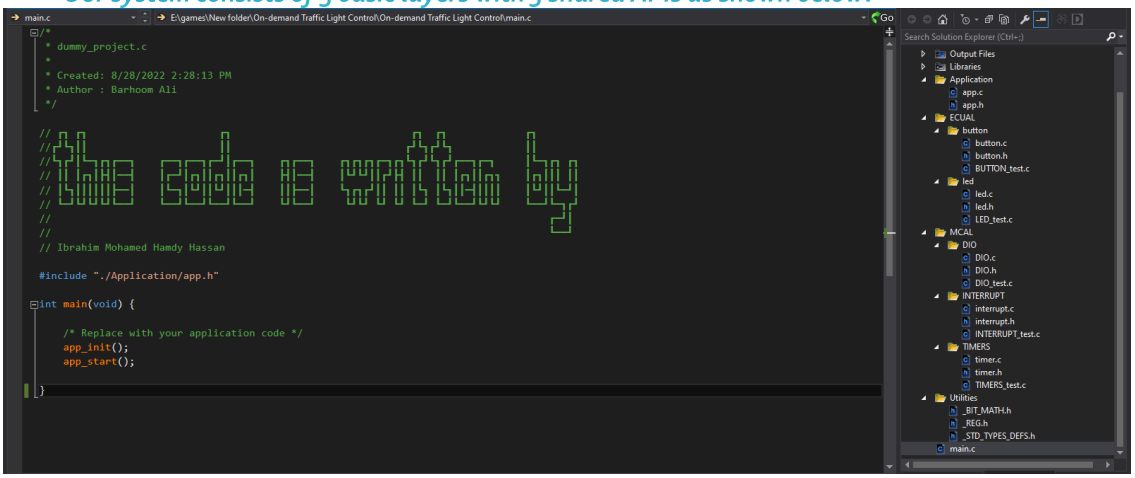
It is required to make a traffic light with a button 3 LEDs for cars and 3 LEDs for pedestrians, as when the red LED for the cars is on the green one for pedestrians is off and vice versa, when the yellow LED for cars is blinking, the yellow one for pedestrians would blink as well.

Pedestrians have the high priority in this project so if they pushed the button the controller will check for the traffic light state as if the state was green or (yellow and the next is green), it will turn off all the LEDs then start blinking both yellow LEDs then turning the red LED for cars and green LED for pedestrians then complete, but if the state is in red for cars or (yellow and the next state is red), it will ignore this action.

2. System Design

2.1. System Layers and Drivers

Our system consists of 5 basic layers with 3 shared APIs as shown below:

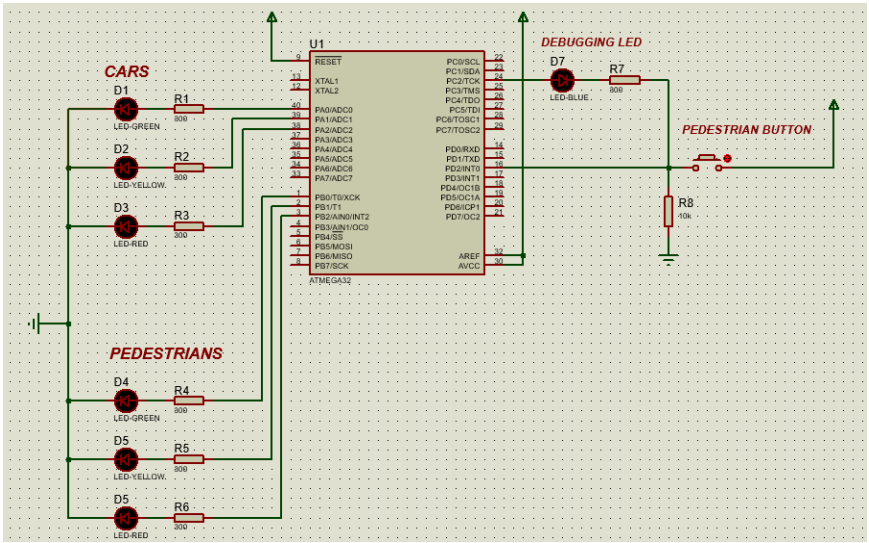


2.1.1. Microcontroller Layer:

This layer contains the hardware components that are connected to make up the circuit we need, and those components are:

- *ATMEGA32 microcontroller.*
- *6 LEDs (2 green LEDs, 2 yellow LEDs, and 2 red LEDs).*
- *Push button.*
- *7 resistors (6 of 300-ohm resistors and a 10-Kohm resistor).*

The schematic used:



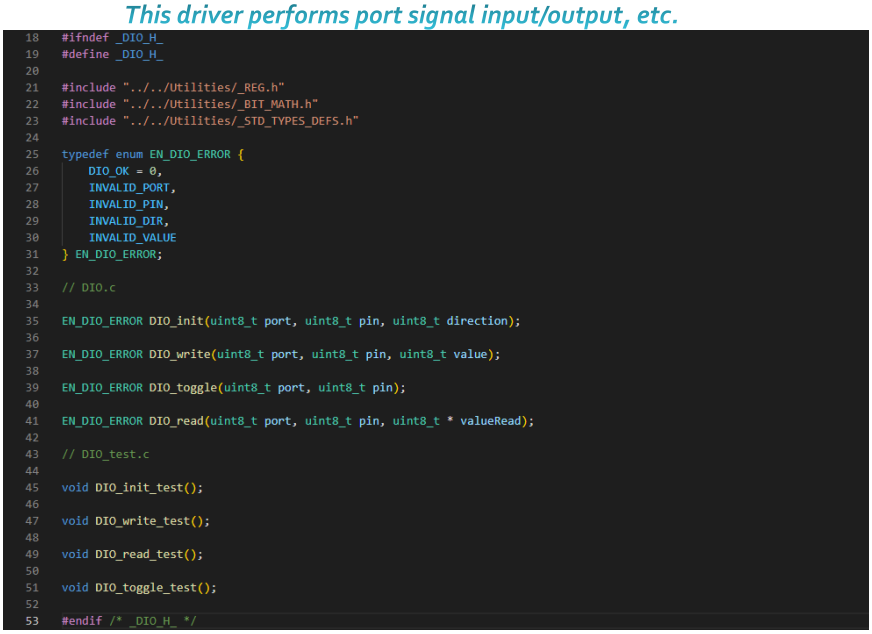
2.1.2. Microcontroller Abstraction Layer (MCAL):

This layer is a software module that directly accesses on-chip MCU peripheral modules and external devices that are mapped to memory, and makes the upper software layer independent of the MCU.

Drivers used in this project are:

- *DIO Driver:*

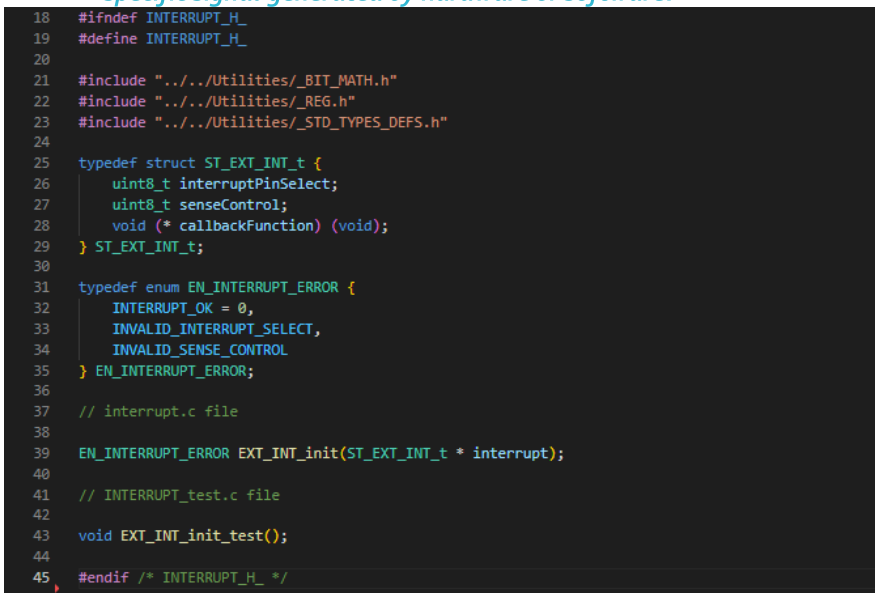
  This driver performs port signal input/output, etc.



```
#ifndef _DIO_H_
#define _DIO_H_

#include "../../Utilities/_REG.h"
#include "../../Utilities/_BIT_MATH.h"
#include "../../Utilities/_STD_TYPES_DEFS.h"

typedef enum EN_DIO_ERROR {
    DIO_OK = 0,
    INVALID_PORT,
    INVALID_PIN,
    INVALID_DIR,
    INVALID_VALUE
} EN_DIO_ERROR;

// DIO.c

EN_DIO_ERROR DIO_init(uint8_t port, uint8_t pin, uint8_t direction);

EN_DIO_ERROR DIO_write(uint8_t port, uint8_t pin, uint8_t value);

EN_DIO_ERROR DIO_toggle(uint8_t port, uint8_t pin);

EN_DIO_ERROR DIO_read(uint8_t port, uint8_t pin, uint8_t * valueRead);

// DIO_test.c

void DIO_init_test();

void DIO_write_test();

void DIO_read_test();

void DIO_toggle_test();

#endif /* _DIO_H_ */
```

- *INTERRUPT Driver:*

  This driver performs immediate attention to an event once the MCU receives a specific signal generated by hardware or software.



```
#ifndef INTERRUPT_H_
#define INTERRUPT_H_

#include "../../Utilities/_BIT_MATH.h"
#include "../../Utilities/_REG.h"
#include "../../Utilities/_STD_TYPES_DEFS.h"

typedef struct ST_EXT_INT_t {
    uint8_t interruptPinSelect;
    uint8_t senseControl;
    void (* callbackFunction) (void);
} ST_EXT_INT_t;

typedef enum EN_INTERRUPT_ERROR {
    INTERRUPT_OK = 0,
    INVALID_INTERRUPT_SELECT,
    INVALID_SENSE_CONTROL
} EN_INTERRUPT_ERROR;

// interrupt.c file

EN_INTERRUPT_ERROR EXT_INT_init(ST_EXT_INT_t * interrupt);

// INTERRUPT_test.c file

void EXT_INT_init_test();

#endif /* INTERRUPT_H_ */
```

- *TIMERS Driver:*

  This driver performs delay functions, PWM signals, counting the number of actions happening, etc.

  In this project, we use timers only in normal mode so this driver is used for performing delay and stopwatch only.

```c
#ifndef TIMER_H_
#define TIMER_H_

#include "../../Utilities/_BIT_MATH.h"
#include "../../Utilities/_REG.h"
#include "../../Utilities/_STD_TYPES_DEFS.h"

typedef struct ST_TIMER_t {
    uint8_t timerSelect;
    uint16_t prescaler;
    uint8_t interrupt_mode;
    void (* callbackFunction) (void);
} ST_TIMER_t;

typedef enum EN_TIMER_ERROR {
    TIMER_OK = 0,
    INVALID_TIMER_SELECT,
    INVALID_PRESCALER,
    INVALID_INTERRUPT_MODE
} EN_TIMER_ERROR;

// timer.c file

EN_TIMER_ERROR timer_init(ST_TIMER_t * timer);

EN_TIMER_ERROR delay_start(uint8_t timer, uint16_t delay_ms, uint16_t prescaler);

EN_TIMER_ERROR delay_stop(uint8_t timer);

uint16_t get_timer_value(uint8_t timer);

EN_TIMER_ERROR stopwatch_start(ST_TIMER_t * timer);

EN_TIMER_ERROR stopwatch_stop(ST_TIMER_t * timer, uint16_t * valueRead);

// TIMER_test.c file

void timer_init_test();

void delay_start_test();

void delay_stop_test();

void stopwatch_start_test();

void stopwatch_stop_test();

#endif /* TIMER_H_ */
```

2.1.3. *Electronics Unit Abstraction Layer (ECUAL):*

An ECUAL driver is a set of functions that initialized the MCU hardware via the MCAL and does the calling of MCAL functions, necessary calculations, algorithms, and utilities, to abstract the hardware handling from the application layer. So, the application code doesn't talk directly to DIO or PWM or whatever.

- *LED driver:*

  This driver performs a set of functions that control a specific LED.

```c
#ifndef LED_H_
#define LED_H_

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

typedef enum EN_LED_ERROR {
    LED_OK = 0,
    INVALID_LED_PORT,
    INVALID_LED_PIN,
} EN_LED_ERROR;

// led.c file

EN_LED_ERROR LED_init(uint8_t ledPort, uint8_t ledPin);

EN_LED_ERROR LED_ON(uint8_t ledPort, uint8_t ledPin);

EN_LED_ERROR LED_OFF(uint8_t ledPort, uint8_t ledPin);

EN_LED_ERROR LED_toggle(uint8_t ledPort, uint8_t ledPin);

// LED_test.c file

void LED_init_test();

void LED_ON_test();

void LED_OFF_test();

void LED_toggle_test();

#endif /* LED_H_ */
```

- *BUTTON driver:*

  This driver performs a set of functions that control a specific BUTTON.

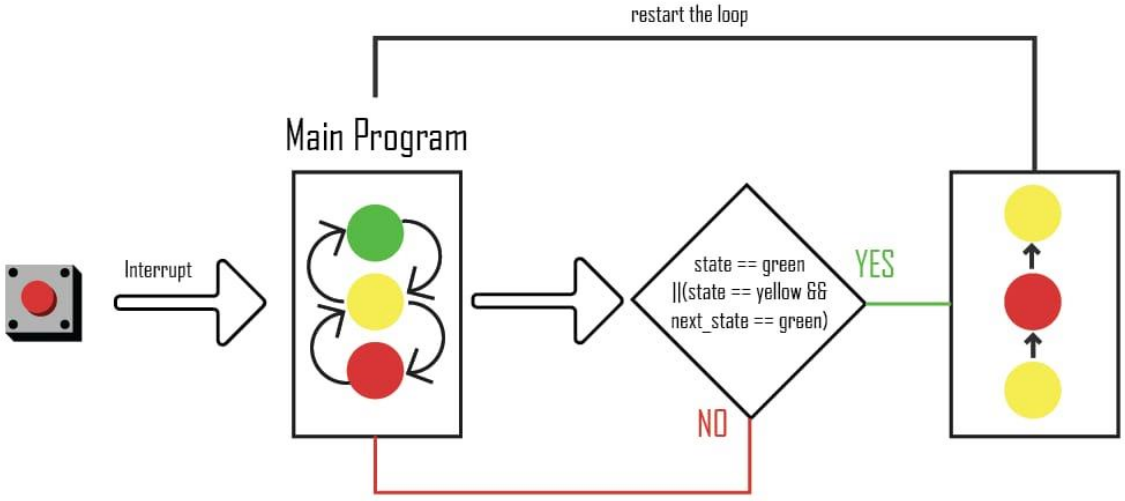
```c
#ifndef BUTTON_H_
#define BUTTON_H_

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

typedef enum EN_BUTTON_ERROR {
    BUTTON_OK = 0,
    INVALID_BUTTON_PORT,
    INVALID_BUTTON_PIN,
} EN_BUTTON_ERROR;

// button.c file

EN_BUTTON_ERROR BUTTON_init(uint8_t buttonPort, uint8_t buttonPin);

EN_BUTTON_ERROR internalPullUp(uint8_t buttonPort, uint8_t buttonPin);

EN_BUTTON_ERROR BUTTON_read(uint8_t buttonPort, uint8_t buttonPin, uint8_t * buttonState);

// BUTTON_test.c file

void BUTTON_init_test();

void internalPullUp_test();

void BUTTON_read_test();

#endif /* BUTTON_H_ */
```

2.1.4. Application Layer:

This layer performs the program's code and shows the output of it.

```c
#ifndef APP_H_
#define APP_H_

#include "../ECUAL/led/led.h"
#include "../MCAL/TIMERS/timer.h"
#include "../MCAL/INTERRUPT/interrupt.h"

void app_init();

void app_start();

#endif /* APP_H_ */
```

3. System Flowchart



4. System Constraints

Everyone should know that making a long press on the crosswalk button has no effect, and pressing it twice have the same effect of pressing it once.