TRAFFIC LIGHT CONTROL

Embedded System Project

Nour Abdel-Aziz Mohamed
Professional Embedded System.

Table of Contents

Chapter 1: 1

Abstract 2

Chapter 2: 4

Hardware 5

Chapter 3: 4

Circuit Diagram 5

Chapter 4: 4

System Layers Description 5

System Constrains 5

Chapter 5: 4

Conclusion 5

Abstract

Systems of traffic light control are frequently employed to observe and manage the movement of cars through the intersection of numerous roads. They want to make sure that cars are moving smoothly throughout the transportation corridors. However, considering the different elements involved, synchronizing several traffic signal systems at nearby junctions is a challenging challenge. Variable flows that are nearing junctions cannot be handled by conventional methods. Additionally, the current traffic system does not take into account the reciprocal interference between adjacent traffic light systems, the discrepancy in car flow over time, accidents, the passing of emergency vehicles, or pedestrian crossings. Congestion and traffic jams result from this. Thus, this project will simulate a control system using AVR ATmega 32 to control the timing when pedestrian wants to cross the street to cross it safely.

Chapter 2

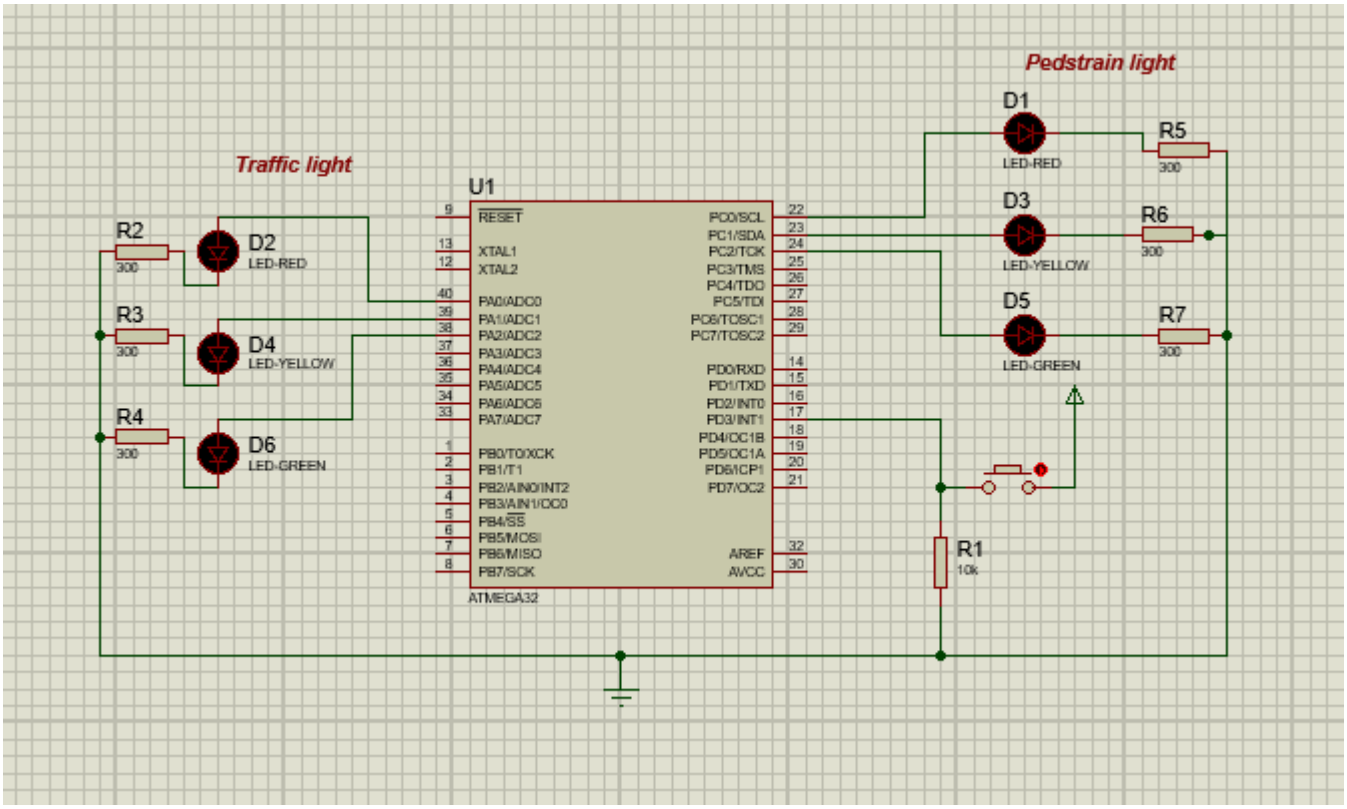
Hardware

The system consists of:

1. AVR ATmega 32.
2. (2) Red LEDs.
3. (2) Yellow LEDs.
4. (2) Green LEDs.
5. Push Button.
6. 100kohm Resistance.
7. (6) 300ohm Resistance.

Chapter 3

Circuit Diagram



Chapter 4

System Layers Description

The system is mainly consisting of three layers:

1. MCAL Layer (Microcontroller Abstraction Layer).
2. HAL Layer (Hardware Abstraction Layer).
3. LIB Layer (Library).
4. APP Layer (Application Layer).

MCAL Layer (Microcontroller Abstraction Layer):

1. DIO

a. DIO_interface.h

b. DIO_program.c

2.TIMER

a.TIMER_interface.h

b. TIMER_program.c

3.INTERRUPTS

a. INT_interface.h

b. INT_program.c

4.Global Interrupts

a. GIE_program.c

b.GIE_interface.h

c.GIE_private

HAL Layer (Hardware Abstraction Layer):

1.BUTTONN

a. BUTTONN_interface.h

b. BUTTONN_program.c

LIB Layer (Library):

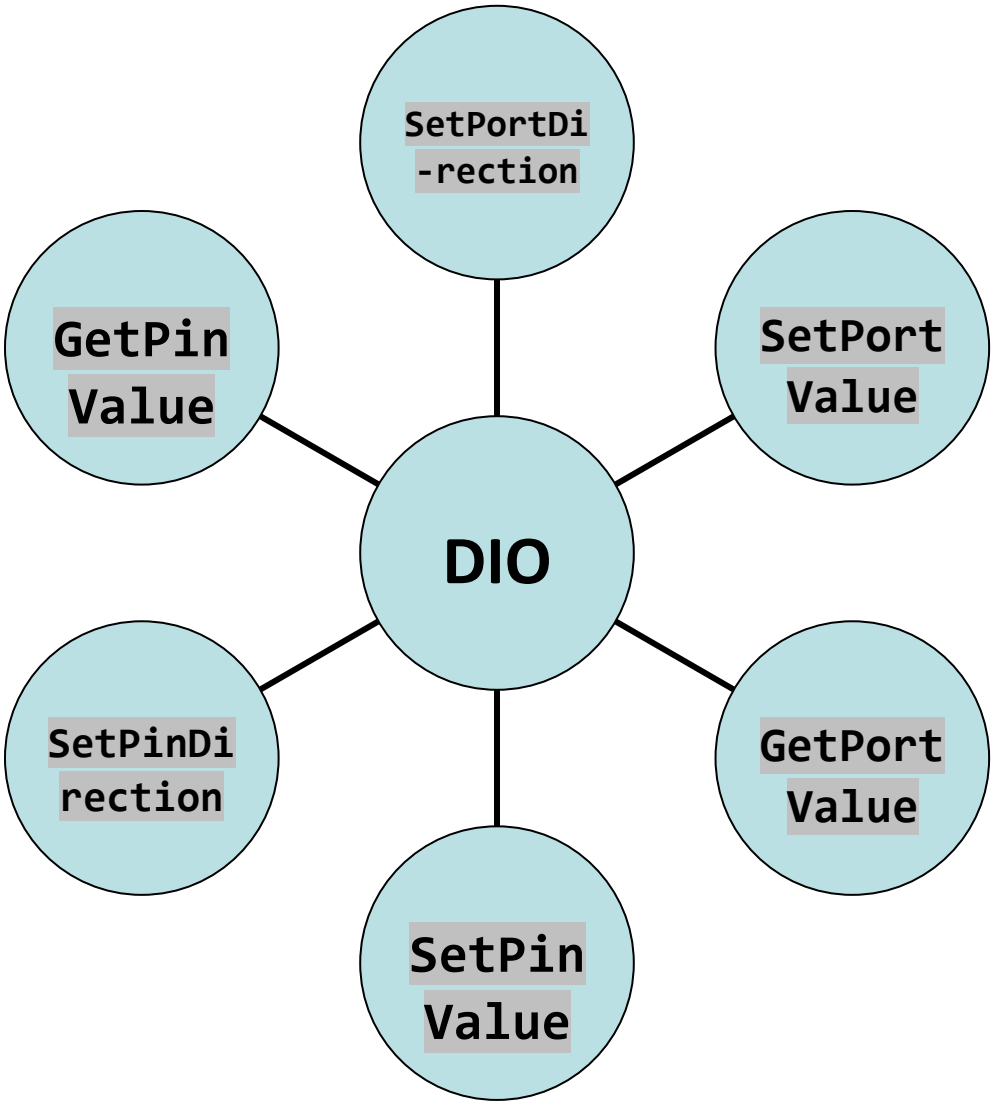
LSTD_types.h

LBIT_math.h

APP Layer:

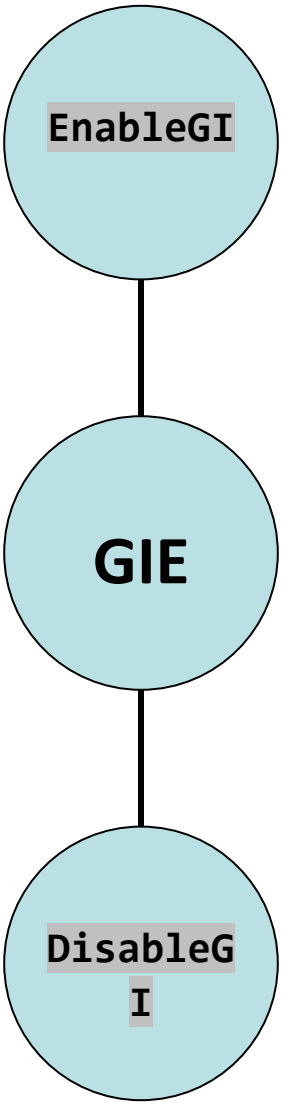
a.APP_program.c

b.APP_interface.h



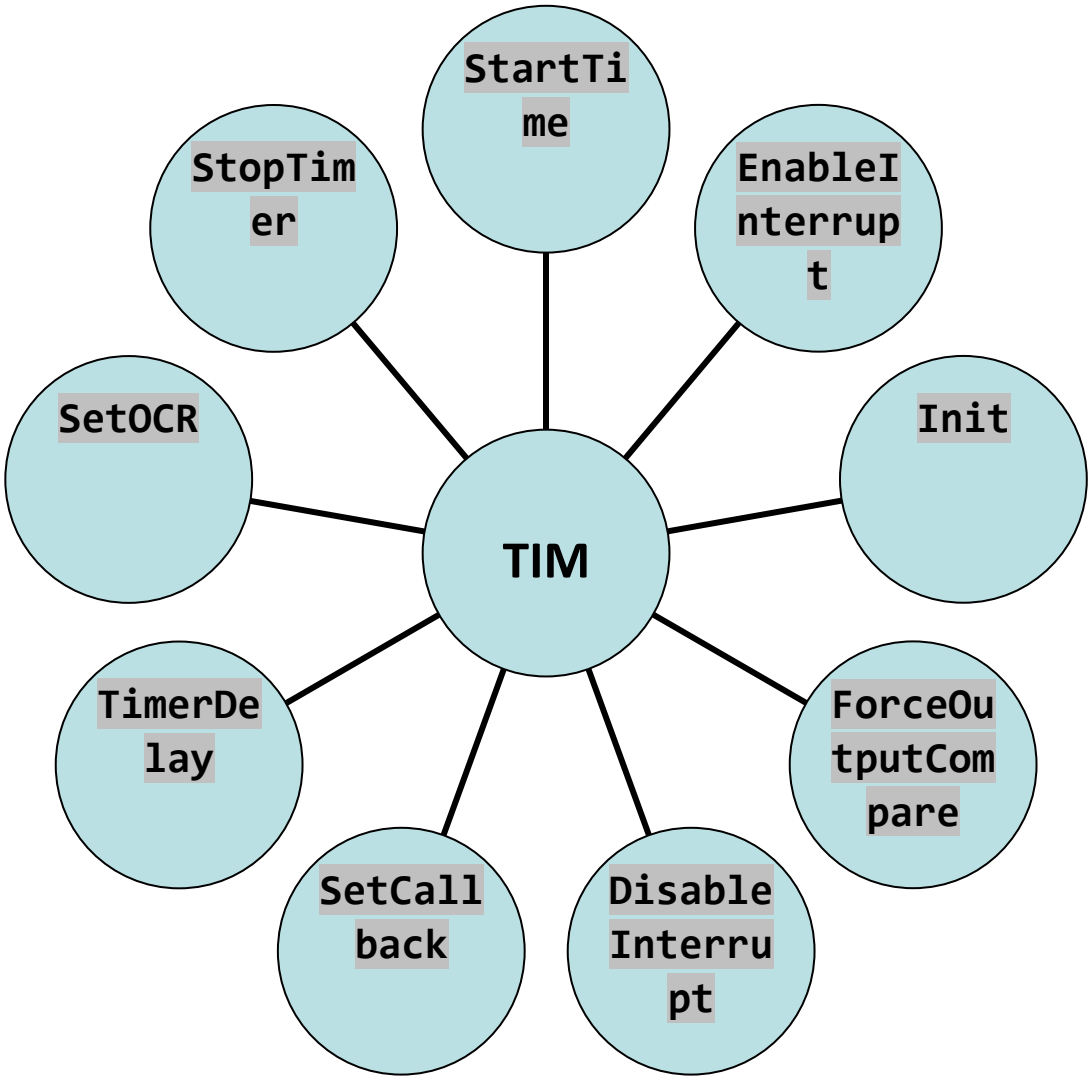

```
void DIO_SetPortDirection(u8 Port, u8 Direction);
void DIO_SetPortValue(u8 Port, u8 Value);
u8   DIO_GetPortValue(u8 Port);

void DIO_SetPinDirection(u8 Port, u8 Pin, u8 Direction);
void DIO_SetPinValue(u8 Port, u8 Pin, u8 Value);
u8   DIO_GetPinValue(u8 Port, u8 Pin);
```




```
void GIE_EnableGI(void);

void GIE_DisableGI(void);
```




```
void TIM_Init(u8 TimerNumber , u8 ClockSelect, u8 Mode, u8 HWPinMode);

void TIM_ForceOutputCompare(void);

void TIM_EnableInterrupt(u8 InterruptSource);

void TIM_DisableInterrupt(u8 InterruptSource);

void TIM_SetCallback(u8 InterruptSource, void (*UserFunction)(void));

void TIM_TimerDelay(u8 TimerNumber, u32 Delay);

void TIM_StartTimer(u8 TimerNumber);
```

```c
void TIM_StopTimer(u8 TimerNumber);

void TIM_SetOCR(u8 TimerNumber, u16 OCRValue);
```

System Constrains

In case of the long press button or double press button these cases are handled in the project by timers by using delay functions in which the system ignores very short time of pressing also the pressing exceeded the specific duration by that the press time on the button is limited by specific time delay range to avoid error from the double press button case and long or short button press so it doesn’t interrupt the system Sequence of operation.