On-demand traffic light control

prepared by student: **Omar Saeed Ali Ahmed Negm**

System description

- It is a traffic light that can be controlled manually if a pedestrian wants to cross the street, by pressing on a button changing the control mode from normal (cars') to pedestrian mode, which have 2 cases according to traffic light states:

1. If pressed when cars' is red, their red will be extended for 5 seconds and at the same time the pedestrian traffic is lit green.
2. If pressed when cars' is green or yellow, the pedestrian will wait till cars' traffic is red after changing to blinking yellow as well as pedestrian yellow blinking at the same time for 5 seconds, then both cars' and pedestrians' traffic lights changes to allow pedestrian cross for 5 seconds.

Once pedestrians' green is time out, both traffic lights change to yellow for five seconds, then change to allow cars to move with traffic control getting back to normal mode.

- This system uses Atmega32 micro controller, and for timer it uses an internal source, and for changing traffic control mode it uses an external interrupt source which to the pedestrian button is connected.

  It uses 6 LEDs: 3 for cars' traffic and 3 for pedestrian's traffic.

System design

System layers	System drivers
Microcontroller Abstraction Layer	- DIO
	- Timer
	- Interrupt
Electronic Unit Abstraction Layer	- button
	- LED
Application	None to be mentioned

APIs of each driver:

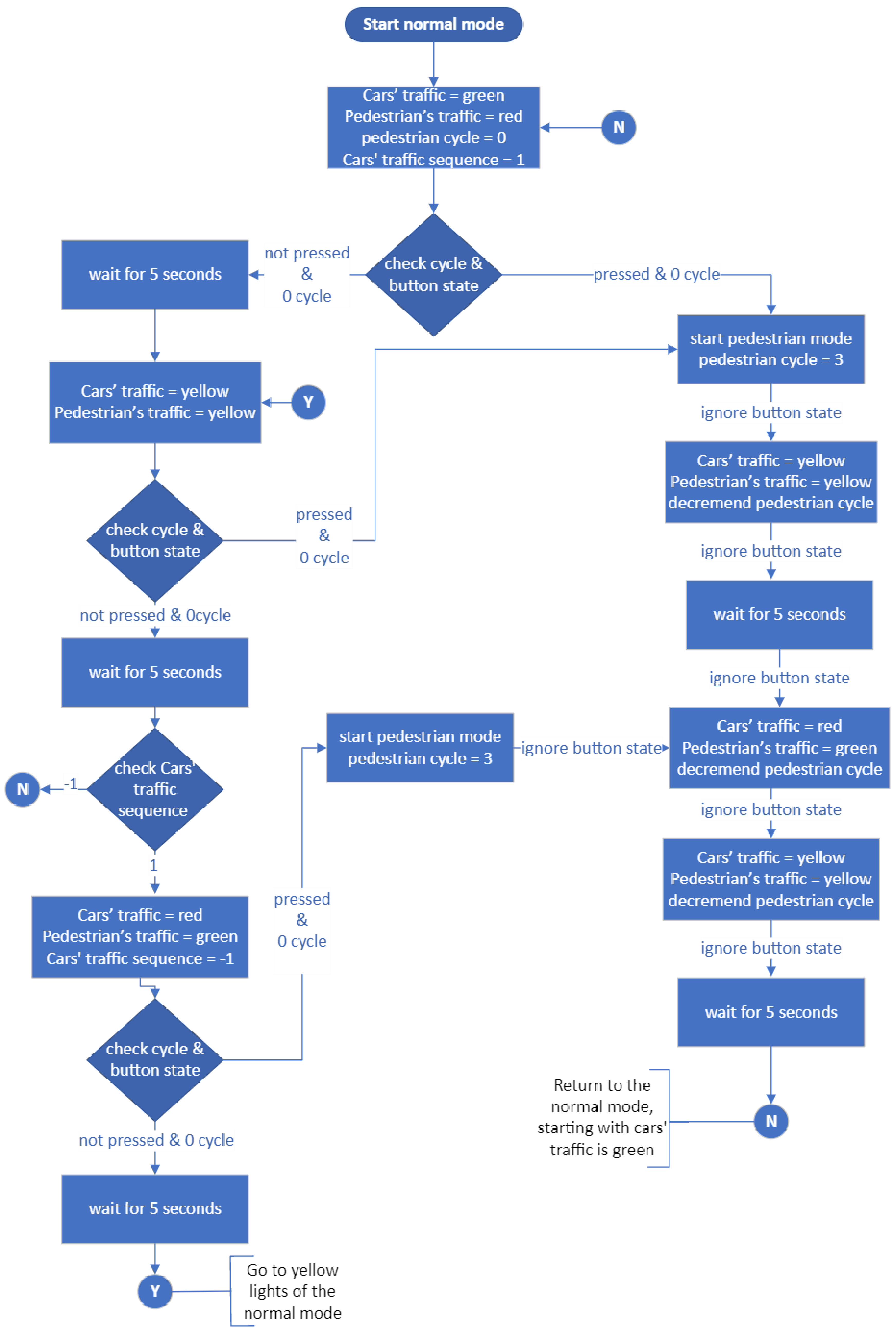
DIO	Description: contains functions to manipulate the microcontroller general purpose pins. Its registers are provided in the **register_file** in the **Utilities** folder. Input arguments: port_letter, pin_number, direction and port data value to be written in the pin of DDR and/or PORT registers. - PORT, PIN and DDR values are provided from the **bti_ops** in the **Utilities** folder. Outputs: pointer of PIN value that is read by read function, change of port data value by write function, pin_direction by initialize function. Returns: error state of each function.
Timer	Description: operates as stopwatch/timer tool of the project. Its registers are provided in the **register_file**in the **Utilities** folder. Input arguments: timer mode, start_value, prescaler of the CPU clock, overflow_state and pointer to overflow counter. Outputs: starting the timer by the start_value in the chosen timer mode with prescaler on the CPU clock, and after the delay... stopping the timer. - Values of timer source registers are provided from the **bti_ops** in the **Utilities** folder. Returns: error state of each function.
Interrupt	Description: handles the crosswalk button states to decide to enable the pedestrian mode. Its registers are provided in the **register_file** in the **Utilities** folder. - ISR function definition is in application layer. Input arguments: type of the external interrupt, control bit of GICR register that is related to external interrupt pin. Outputs: enables the sense control to generate the request when the interrupt is triggered. - Values of external interrupt registers are provided from the **bti_ops** in the **Utilities** folder. Returns: error state of each function.
button	Description: reads button actions. Input arguments: button that are used provided with port_letter and pin_number. Outputs: button state and action that occurs according to pressing the button. Returns: error state of each function.
LED	Description: controls LEDs actions. Input arguments: LED(s) that are used provided with port_letter and pin_number. Outputs: state of each LED according to the system description. Returns: error state of each function.

New datatypes:

- **unsigned char**, defined as **uint8_t**.
- **unsigned int**, defined as **uint32_t**.
- **Enumeration** types in each driver, to return error state of each function.

System flowchart

made using Microsoft Visio.



Start normal mode
Cars' traffic = green
Pedestrian's traffic = red
pedestrian cycle = 0
Cars' traffic sequence = 1
N
check cycle & button state
not pressed & 0 cycle
pressed & 0 cycle
wait for 5 seconds
start pedestrian mode
pedestrian cycle = 3
Cars' traffic = yellow
Pedestrian's traffic = yellow
Y
ignore button state
Cars' traffic = yellow
Pedestrian's traffic = yellow
decremend pedestrian cycle
check cycle & button state
pressed & 0 cycle
not pressed & 0cycle
wait for 5 seconds
wait for 5 seconds
check Cars' traffic sequence
-1
N
1
start pedestrian mode
pedestrian cycle = 3
ignore button state
Cars' traffic = red
Pedestrian's traffic = green
decremend pedestrian cycle
Cars' traffic = red
Pedestrian's traffic = green
Cars' traffic sequence = -1
pressed & 0 cycle
Cars' traffic = yellow
Pedestrian's traffic = yellow
decremend pedestrian cycle
check cycle & button state
wait for 5 seconds
not pressed & 0 cycle
Return to the normal mode, starting with cars' traffic is green
N
wait for 5 seconds
Y
Go to yellow lights of the normal mode

System constraints

1. Do not use external timer source.

2. Normal mode is the only allowed for timer in this project.

3. Do not set negative or floating-point values to any parameter of any function in this project.

4. Prescalers of CPU clock to start the timer (from ATmega32 datasheet):
   - 0 - 8 -64 - 256 -1024

5. To use DIO registers, follow these instructions:
   - port_letter format: portX - pin_number: pinN
   where: X is character , N is digit

6. HIGH_LEVEL mode of external interrupt is not available.

7. 6th bit of GICR is the only to be used for INT0 pin, to enable external interrupt.

8. If you want to use LED_blink() function, you may have to use another timer source.

9. Use only pin2 in portD for button as it is the same pin of enabling pedestrian mode.

10. You can just make short press on crosswalk button to enable the pedestrian mode then set 3 cycles for it.

11. Long press or double press on the button won’t do any change until the pedestrian mode cycles end then resetting to normal mode again.