COMPSYS 303 Assignment 1
Designing a Traffic Light Controller with Nios II
Due on 28th August

For this assignment you are required to create four traffic light controllers, with increasing functionalities for each successive revision. All four traffic light controller applications will be hosted within a single system that will allow the user to switch between either one of them at any time using the SWITCHES. The current mode must be constantly displayed to the user through the LCD screen, and any other outputs you desire. The system must ensure that a mode change only occurs when the application is in a safe state (eg R-R).

Designing a simple traffic light controller (Mode 1)

Consider a simple traffic light controller for controlling the intersection between two roads: one going in the north-south (NS) direction and another going in the east-west (EW) direction as shown in Figure 2. The traffic controller behaves as the FSM in Figure 3. R, G, and Y correspond to the red, green and yellow outputs from the traffic light respectively. The notation used in each state gives the output for the NS traffic light first, followed by that of the EW traffic light. t1 to t6 are timeout values that control the timing of the state transitions.

For implementation, the first three (from right) green LEDs are to be used as the outputs for the NS traffic light, while the next three green LEDs are to be used as the same for the EW traffic light, as illustrated in Table 1 below. The timeout conditions are to be implemented using Nios II's timer interrupt facility.

Light:	EW - R	EW - Y	EW - G	NS - R	NS - Y	NS - G
Bit:	5	4	3	2	1	0

Table 1: Traffic lights and LEDS_GREEN bits.

In order to complete this task, we *suggest* you implement the following functions:

- simple_tlc – implements the simple traffic light controller
- tlc_timer_isr – handler for the traffic light timer interrupt
- lcd_set_mode – write the current mode to the LCD

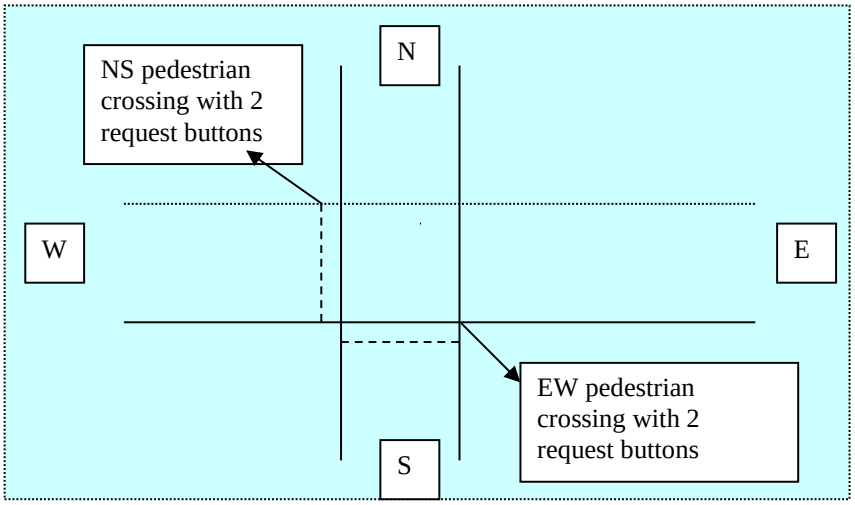

Figure 2: Model traffic intersection

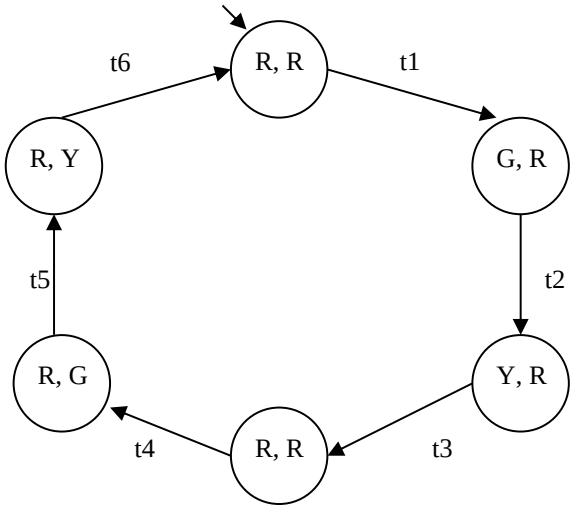

Figure 3: Simple traffic light controller

Designing a pedestrian traffic light controller (Mode 2)

The pedestrian traffic light controller is an extension of the simple traffic light controller with additional pedestrian lights and input buttons. The traffic light controller can be modelled as three concurrent threads as shown in Figure 4. In the figure, the first thread controls the change of the lights at the intersection, as well as the two pedestrian lights. The second and third threads handles the pedestrian requests through two external inputs, modelled as events NSraised and EWraised.

This approach of extending the simple traffic light controller with pedestrian function allows the simple traffic light controller developed previously to be reused with minimal modifications. Communication between the threads is achieved through shared variables.

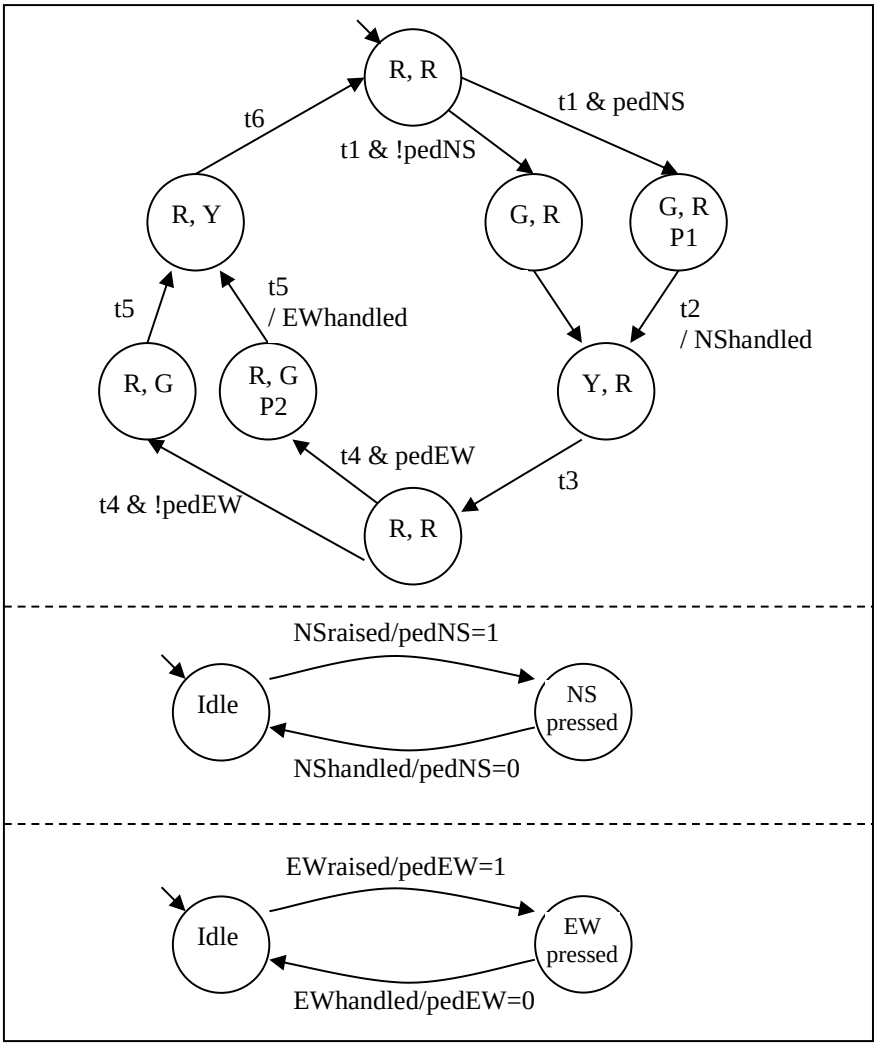

Figure 4: Pedestrian traffic light controller

For the implementation, the remaining two LEDs will be used to represent the NS and EW pedestrian lights respectively. Furthermore, KEY_0 and KEY_1 shall be used as the NS and EW pedestrian buttons respectively. External interrupts shall be used to sense the inputs from these two buttons. Accesses to shared variables are to be implemented as critical sections so that they cannot be modified by the main thread and the interrupt handlers at the same time.

Light:	EW Ped	NS Ped	EW - R	EW - Y	EW - G	NS - R	NS - Y	NS - G
Bit:	7	6	5	4	3	2	1	0

Table 2: Pedestrian and Traffic lights and LEDS_GREEN bits.

We *suggest* the following functions to complete this task:

- init_buttons_pio – initializes the interrupts for the NS and EW pedestrian buttons
- pedestrian_tlc – implements the pedestrian traffic light controller
- NSEW_ped_isr – handles the NS and EW pedestrian button interrupts

Designing a configurable traffic light controller (Mode 3)

In the previous traffic light controllers, the timeout values t1 to t6 are hard-coded. In a practical controller, there might be an initialization phase (possibly at night time, when traffic is minimal) to reconfigure the timeout durations of the traffic light.

For the purpose of this assignment, the timeout values will be entered in PuTTY or HyperTerminal and fed in to the controller through UART. The ***blocking*** UART fgetc() function, will only be used when the controller is in the Red-Red state. The timeout values do not need to be reconfigured every time, so the usage of a switch to indicate new values is advised. The string containing the timeout values shall be terminated by the line-feed character (\n), which will serve as the end-of-packet character. The string will follow the format:

- #,#,#,#,#,#[\r]\n
    - Where '#' is a 1-4 digit integer
    - ',' seperates the timeout values
    - \r is ignored (and may or may-not be received)
    - \n signals the end of the inputs

The reception of the end-of-packet character will result in a transfer of the 6 buffered timeout values, into the global timeout values. If 6 valid numbers are not received, it will wait for another string.

Once the timeout data handler completes, the controller switches back to its normal operation mode and resumes from the present {R, R} state.

The following functions are suggested for this task:

- configurable_tlc – implements the configurable traffic light controller
- timeout_data_handler – parses the configuration string and updates the timeouts

Designing a traffic light controller with a red light camera (Mode 4)

A traffic light controller can also be augmented with a red light camera to capture any violations during a change of lights. We wish to design a camera that would be activated whenever a vehicle crosses the intersection when the light is yellow. If the vehicle remains within the intersection after a predefined amount of time, the camera will take a snapshot of that vehicle. Alternatively, if the car enters on a red light then a snapshot should be taken immediately.

In this assignment, the detection of a vehicle entering and leaving the traffic intersection shall be simulated using the odd and even button presses of KEY_2 respectively. Whenever the camera timer is activated, it will print the message "Camera activated" through the UART. If a snapshot of the vehicle is taken by the camera, then the message "Snapshot taken" should be printed. When the vehicle leaves the intersection, the message "Vehicle left" is to be printed instead. An additional timer interrupt is to be used to measure the time that a vehicle was in the intersection, which should be printed out when it leaves. We shall assume here that at most only one vehicle may enter the intersection at any given time.

The following functions suggested for this task:

- camera_tlc – implements the traffic light controller with integrated camera
- handle_vehicle_button – simulates the entry and exit of vehicles at the intersection
- camera_timer_isr – handler for the red light camera timer interrupt

Due Date: The assignment is to be demonstrated on **Wednesday 28th August** in **UG2** from **1pm – 4pm**. There will be support sessions announced on Canvas in the week(s) before where students can practice their assignment and ask any unresolved questions.

Final code is to be submitted through Canvas by **Wednesday 28th August at 11:59pm**.
Please submit as a zip file,

1) The project folder for the traffic light controller,
2) The bsp project files.
3) The .sof and .sopcinfo files
4) Any documentation you have (e.g. README)

Please zip everything up into a single achieve and name your file as your project number (e.g. group_1.zip or group_2.zip).