COMPSYS 303

Assignment #1

Ho Seok AHN

hs.ahn@auckland.ac.nz

THE UNIVERSITY OF AUCKLAND
FACULTY OF ENGINEERING

1. COMPSYS 303 Assignment #1

THE UNIVERSITY OF AUCKLAND
FACULTY OF ENGINEERING

□ Designing a Traffic Light Controller with Nios II

- Lab1 (SoPC based system design) Week 2: Wed 1-4 pm in UG1, UG2
- Lab2 (Programming a SoPC) Week 3: Wed 1-4 pm in UG1, UG2
- Assignment 1 consultation Week 5: Fri 2-4 pm UG1, UG2
- **Assignment 1 demo Week 6: Wed 1-4 pm UG1, UG2 (10%)**

1. COMPSYS 303 Assignment #1

□ Designing a Traffic Light Controller with Nios II

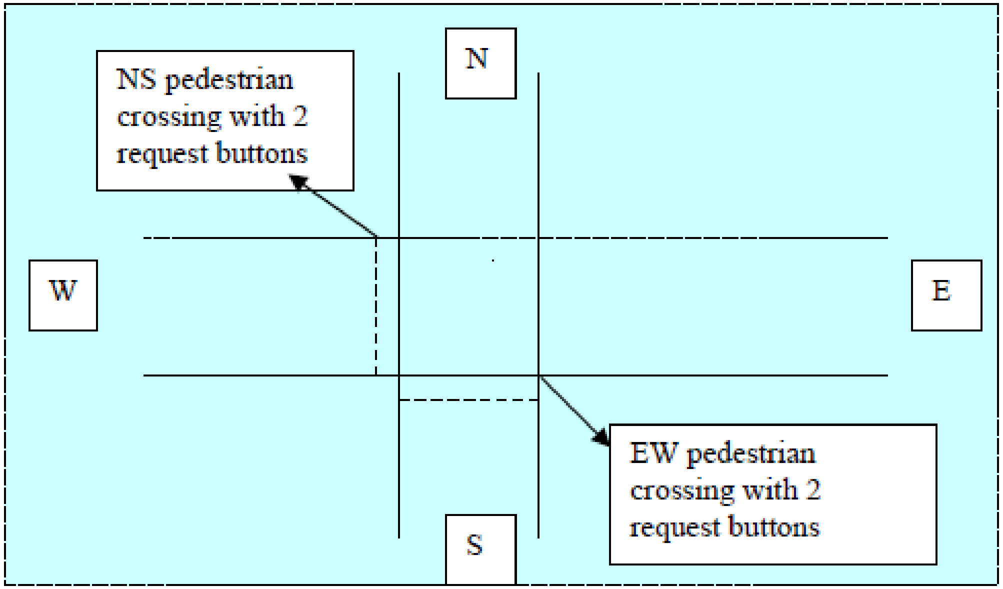
- Create **four traffic light controllers**.
- Each level needs increasing functionalities for each successive revision.
- All four traffic light controller applications will be hosted within a single system that will allow the user to switch between either one of them at any time **using the SWITCHES**.
- 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 (e.g. R-R).

2. Mode 1

THE UNIVERSITY OF AUCKLAND
FACULTY OF ENGINEERING

□ Designing a simple traffic light controller

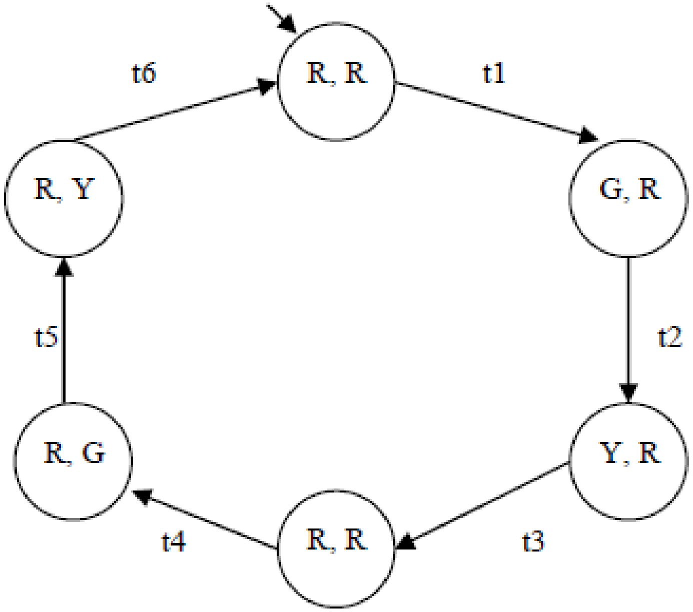
- **Mission**
  - Consider a simple traffic light controller for controlling the intersection between two roads.
  - One going in the North-South (NS) direction.
  - Another going in the East-West (EW) direction.


N
NS pedestrian crossing with 2 request buttons
W
E
S
EW pedestrian crossing with 2 request buttons

2. Mode 1

□ Designing a simple traffic light controller

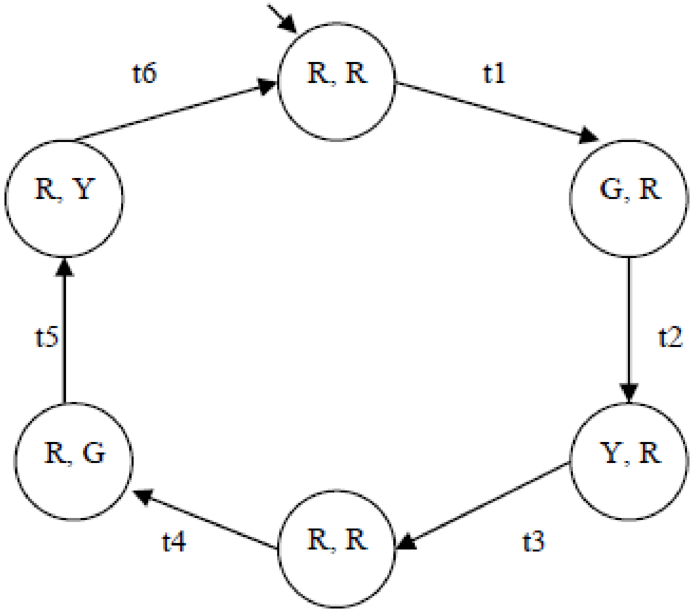
- **FSM**
  - The traffic controller behaves as this FSM.
  - Here, t1 to t6 are timeout values that control the timing of the state transitions.



2. Mode 1

□ Designing a simple traffic light controller

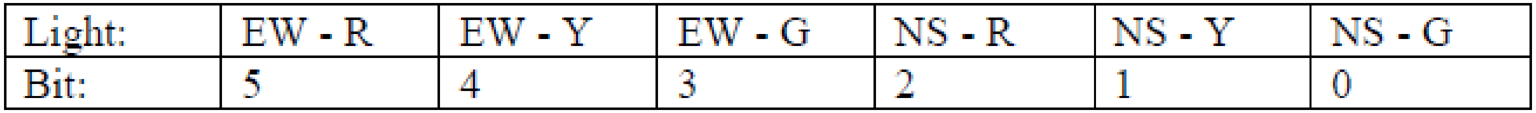
- **FSM**
  - R, G, and Y correspond to the red, green and yellow outputs from the traffic light respectively.
  - The notation used in each state gives the output for the NS traffic light first, followed by that of the EW traffic light: **state (NS, EW)**.



2. Mode 1

☐ Designing a simple traffic light controller

- **Implementation**
  - We have only same coloured LEDs on the board.
  - So, the first three (from right) LEDs are to be used as the outputs for the NS traffic light.
  - The next three LEDs are to be used as the same for the EW traffic light.
  - 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

2. Mode 1

□ Designing a simple traffic light controller

- **Implementation**
  - 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

3. Mode 2

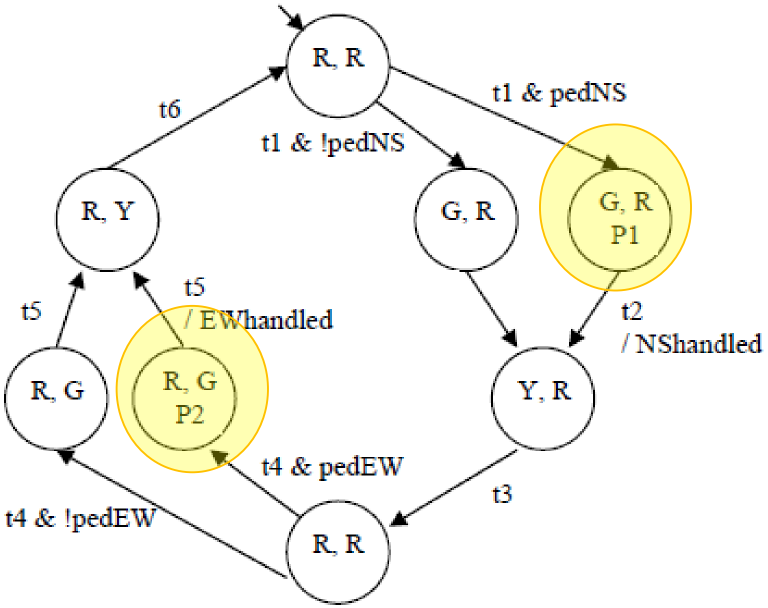
□ Designing a pedestrian traffic light controller

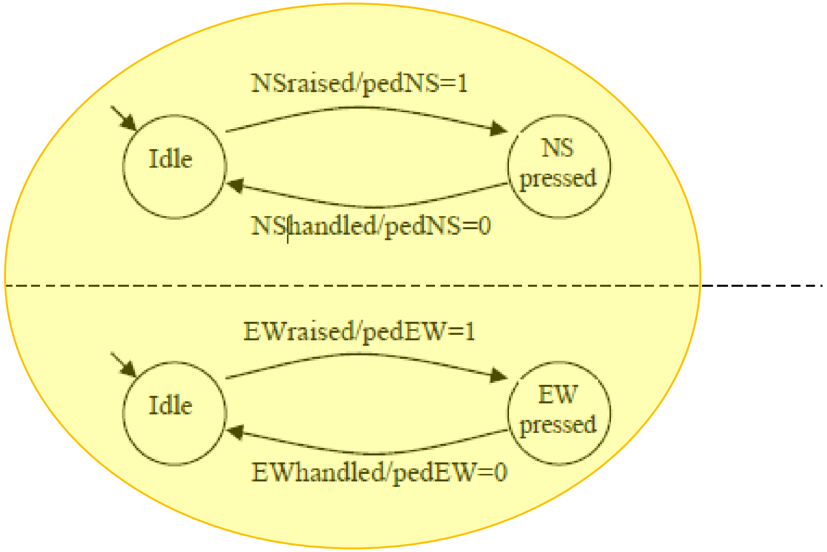
- **Mission**
  - The pedestrian traffic light controller is an extension of the simple traffic light controller with additional pedestrian lights and input buttons.

3. Mode 2

□ Designing a pedestrian traffic light controller

- **Mission**
  - 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.

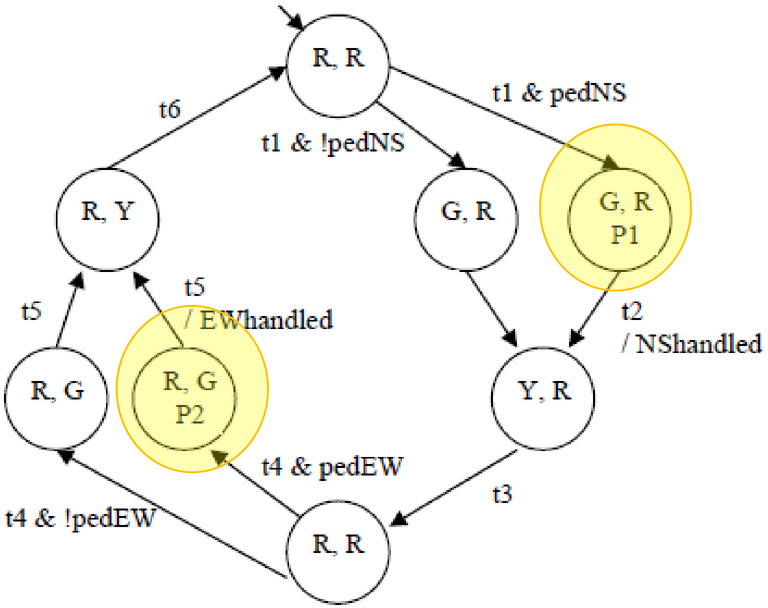


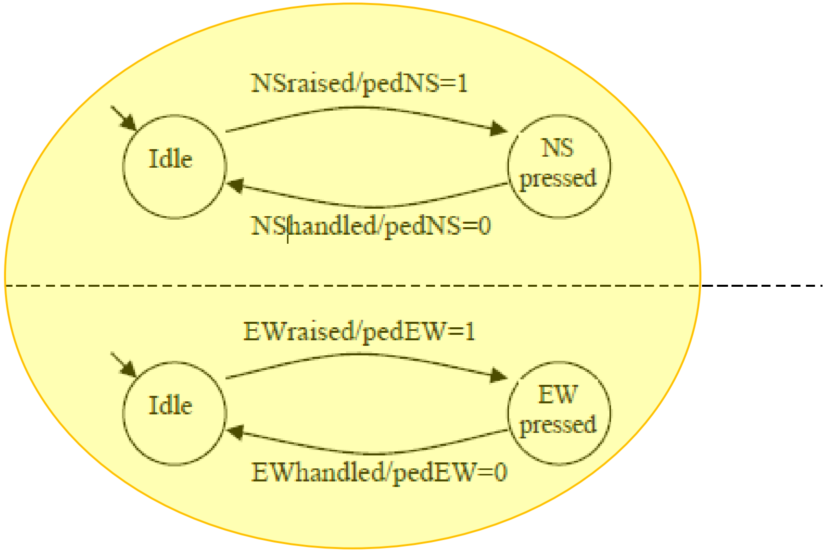


3. Mode 2

□ Designing a pedestrian traffic light controller

- **Mission**
  - The first thread controls the change of the lights at the intersection, as well as the two pedestrian lights.
  - The second and third threads handle the pedestrian requests through two external inputs, modelled as events NSraised and EWraised.

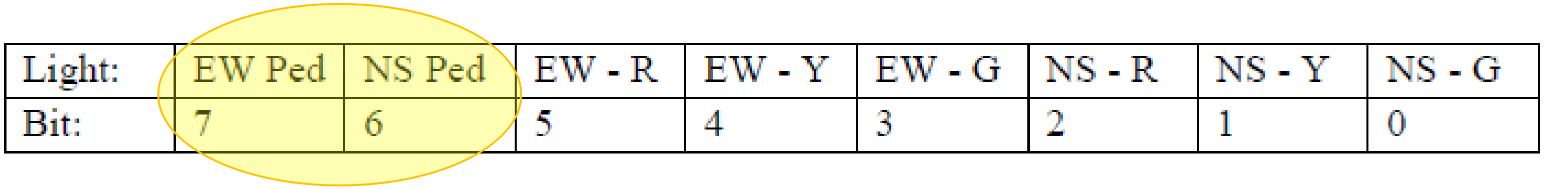




3. Mode 2

□ Designing a pedestrian traffic light controller

- **Implementation**
  - In mode 1, the first three (from right) LEDs are to be used as the outputs for the NS traffic light.
  - In mode 1, the next three LEDs are to be used as the same for the EW traffic light.
  - The remaining two LEDs will be used to represent the NS and EW pedestrian lights respectively.



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

3. Mode 2

□ Designing a pedestrian traffic light controller

- **Implementation**
  - **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.

3. Mode 2

□ Designing a pedestrian traffic light controller

- **Implementation**
  - 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

4. Mode 3

□ Designing a configurable traffic light controller

- **Mission**
  - 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.

4. Mode 3

THE UNIVERSITY OF AUCKLAND
FACULTY OF ENGINEERING

□ Designing a configurable traffic light controller

- **Implementation**
  - 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,

    ',' separates 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.

4. Mode 3

THE UNIVERSITY OF AUCKLAND
FACULTY OF ENGINEERING

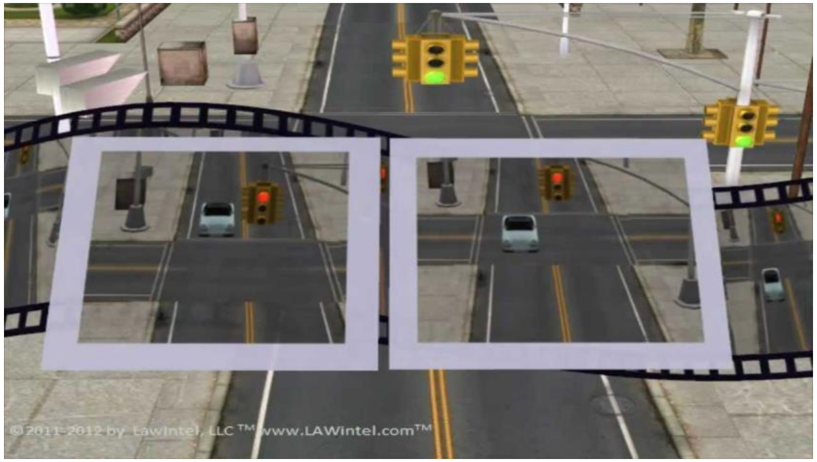
□ Designing a configurable traffic light controller

- **Implementation**
  - 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

5. Mode 4

□ Designing a traffic light controller with a red light camera

- **Mission**
  - 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 is in 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.



TRIMARC/Northrop Grumman Mission Systems

5. Mode 4

□ Designing a traffic light controller with a red light camera

- **Implementation**
  - 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.

5. Mode 4

□ Designing a traffic light controller with a red light camera

- **Implementation**
  - 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

6. Submission

□ **Designing 4 traffic light controllers (10%)**

- **Demo (3%)**
  - **Wednesday 28th August** in **UG2** from **1pm – 4pm**.
- **Files to be submitted (7%)**
  - Final code is to be submitted through Canvas by the midnight of the demo day.
  - Please submit the followings 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**).