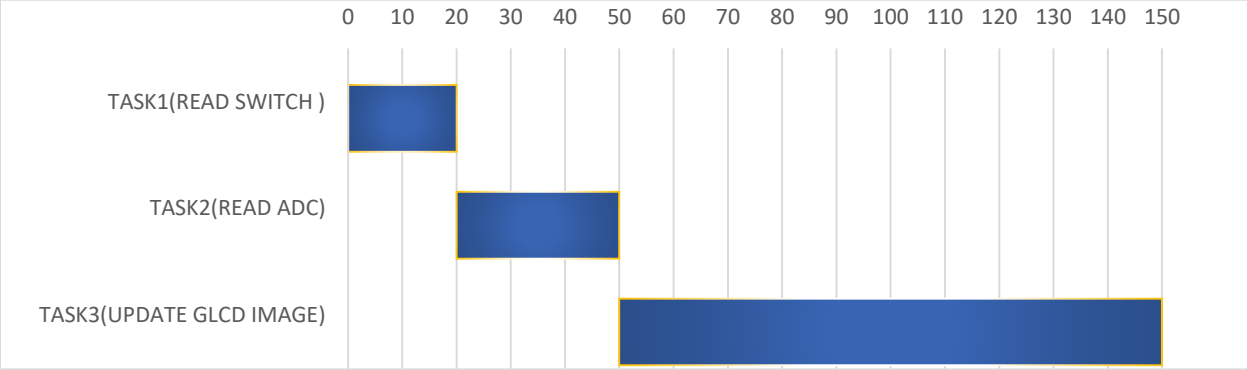
163054001 : LAB 7

The timeline for three tasks scheduled in sequential manner : Once cycle is complete, it will repeat.
Task 1: Read the switches (four push buttons plus shoot button) and update the LEDs and buzzer.
Task 2: Read the ADC value and update the delay to change the speed of Animation.
Task 3: Update the image on the GLCD.
The one cycle of three tasks would be as follows:



Time Range for which switch press will go undetected : 130millisec(Max delay). When executing task sequentially in loop, each task will get its turn as per above timeline. Each task will be complete its execution before transferring control to next task. Value of the switch will be read only when its turn come after execution of task2 and task3 being executed.

Priority of tasks for real time scheduling:
Priority1: Task 1: Read the switches (four push buttons plus shoot button) and update the LEDs and buzzer.
Priority2 :Task 2: Read the ADC value and update the delay to change the speed of Animation.
Priority3: Task 3: Update the image on the GLCD.

Reason for above priority:
a) Since, the output of the task when switch is pressed should be displayed immediately as this task takes least time and user would not want any lag in output when he presses the button. The user would like to view direct results without any delay on button press hence task1 was given first priority.
b) Since value of delay to change the speed of animation for display on GLCD is set by ADC value ie the value of delay set by task2 decides the rate of change of images in GLCD,hence this task is given more priority than Task3.

Implementation in TIVA RTOS:
Handle: readSWITCHtask Function: readSwitch Priority:3 //will give task1 highest priority.
Handle: readADCtask Function: readADC Priority:2
Handle: updateGLCDtask Function: updateGLCD Priority:1

For running the task, semaphore corresponding to Task has to be posted.
Semaphore_post(SWITCHsem);
Semaphore_post(ADCsem);.
Semaphore_post(GLCDsem); .
When semaphore is incremented by one it will unblocks the Task.
Semaphore_pend(*sem, wait/timeout) decrements the semaphore by 1.Until semaphore value is zero, task is blocked.

Notes. The switches respond in real-time when implementing the test code using RTOS. No visible lag observed on the GLCD.