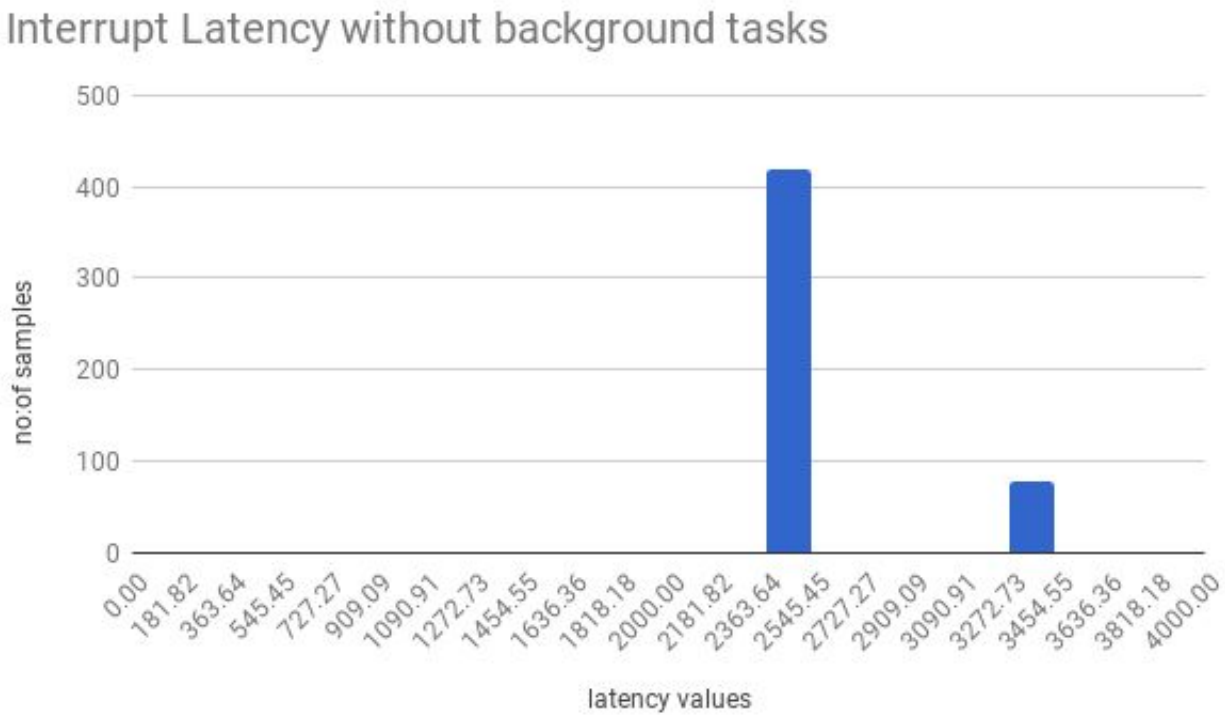
RESULTS

In this assignment we wrote a shell module to accept commands which can calculate the context switch overhead and interrupt latency values with and without background tasks in zephyr rtos. For all the measurements we took TSC register to get timestamps.

CONTEXT SWITCH OVERHEAD:
Below is the distribution of overhead values for 500 samples

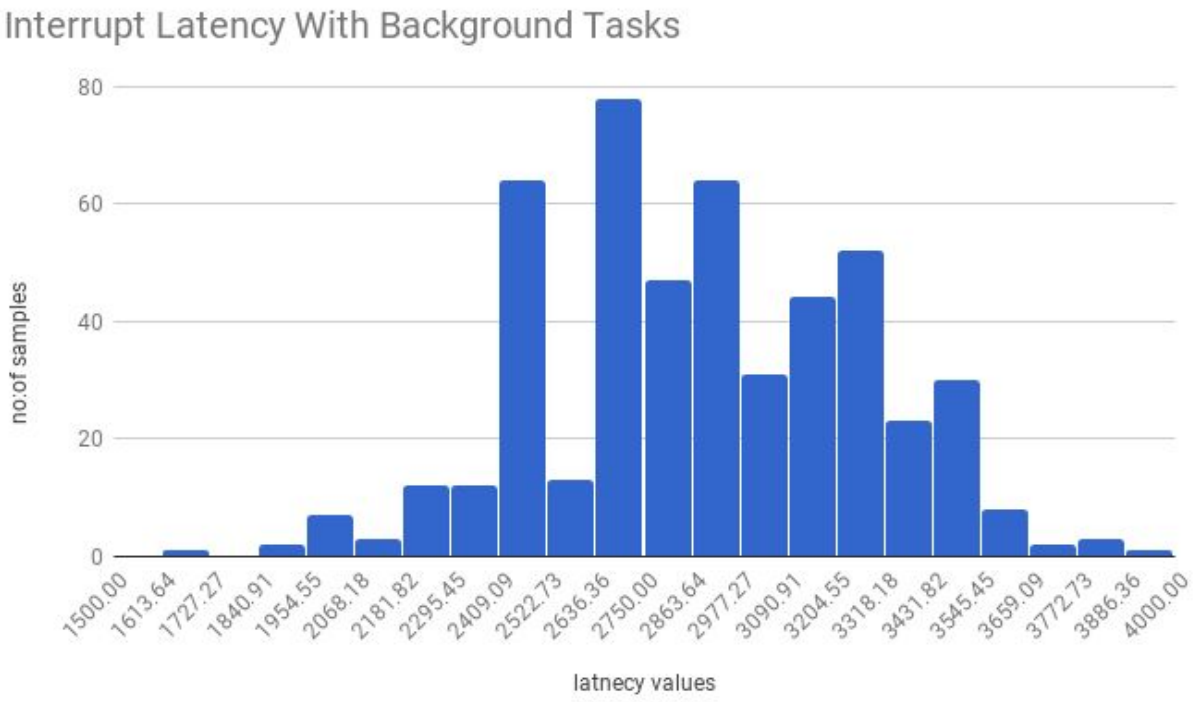
Explanation:
To calculate the context switch overhead values we used two threads which executes alternately by yielding the cpu to other thread. Just before yielding the control to other thread we take a timestamp and immediately as the control enters the second thread we take second timestamp. The difference in timestamps gives the context switch overhead. Also using a test program we calculated the overhead of calling the k_yield function and subtracted this value to get accurate overhead values.

INTERRUPT LATENCY WITHOUT BACKGROUND TASKS:



Explanation:

Above diagram shows the distribution of interrupt latency values without background tasks against 500 samples. We used the gpio functions to calculate the latency values. We used a loop back to two gpio pins one configured as output and other as input. We used a thread which generates trigger pulses by using one of gpio pin. As the signal is detected by other gpio pin interrupt is generated. We took a timestamp just before giving the trigger and then as the control enters the interrupt handler, we took other timestamp. The effective latency value is found by calculating the difference between these two. Also the overhead of gpio functions is calculated using a test program and that value is subtracted from the above value to get accurate measurements.



Explanation:

For this part we generated PWM pulse with a duty cycle of 50%. Using CRO we found out the period of PWM to be around 4.6ms. Now the PWM pulse generated by hardware blocks is looped back to one of gpio configured as input. The interrupt is received at rising edge of PWM. We took timestamps every time the control enters the interrupt handler. We used message transferring between threads as the background task. In this message queue, the interrupts are disabled while exchanging messages between threads which creates some delay in servicing the interrupts. So the interrupt latency values fluctuates as seen from the distribution above. By calculating the difference between consecutive timestamps in handler and the period of PWM, we calculated the interrupt latencies.