7. There is a preemption constraint on the control law update function, as the processing of that job cannot be interrupted and resumed later. All actuator control has to be done in the same uninterrupted block. The control law update must then be a critical region of the program.

There are no precedence constraints in this problem because no jobs must be completed before another job can be started. That is a consequence of our choice of running the threads on three machines so that jobs are independent on the three threads. Only the IMU and GPS measurements values are having locks, such that they can be correctly read and updated under any circumstances. Consequently, the GPS and IMU measured values cannot be updated in the lock while the control law function is executed.

We propose to run three processes on three different machines. The control loop process would have access to the *IMUlock* and *GPSlock*. The lock value of a measurement would be accessible to the control process only if and only if a measurement update has been made (by processes *IMUupdate*, respectively *GPSupdate*). The threads would have the following functions:

T1 Control(period=10ms)

{Control(GPSlock, IMUlock);}

The period argument of the function shows that it will be periodically called at intervals of 10ms in order to assure that the control updates have a frequency of 100Hz. Alternatively, one may understand this as the control function being called when a system timer completes a cycle of 10 ms.

T2 GPSupdate(period=100ms)

{aux=GPS\_measurement();

lockupdate(GPSlock,aux);}

In the GPS update process, one must store the measured value in an auxiliary variable to save computational resources. Only when the GPS lock is available for changing the value, the aux value will be retrieved from the computer memory, and written to the GPS lock value state variable.

T3 IMUupdate(period=10ms)

{aux=IMU\_measurement();

lockupdate(IMUlock,aux);}

Implementing the described scheduling algorithm would solve the problem.