(3)

1)

So from what I learned after going through “Publisher” and “Subscriber” node relation that Publisher publishes the commands and subscriber keeps or collects those commands and prints it or you can call shows the output. So I think while communications between a Publisher and a Subscriber was happening there was a “Queue” problem that was probably the reason for the rover getting stopped.

The serial link and processing queue’s commands probably got queued. The commands may come from hardware’s code or host USB’s commands. That makes the rover’s hardware act on old commands such as velocity commands or others instead of the most recent command “STOP” . This is why I think the rover was obeying commands sometimes and not obeying at all at other times.The problem may get worse with speed because the person who’s operating the rover is sending more frequent commands which increases queueing and makes “stale” commands.

2)

ROS node writes many twist packets quickly. The USB driver may buffer writes which Hardware reads later. No flow control means host can send faster than the hardware is processing.

Bytes are read through UART interrupt and placed in a ring buffer. If the hardware’s main loop processes packets slower than the arrival rate then packets may backlog.

Firmware can put incoming commands into an internal queue and update motor outputs only at some lower frequency.Blocking code on rover’s hardware may block interrupts or processing and delay command handling.

Also when the motor update routine is called, if it is run at a low, fixed rate or gated through a blocked timer then newer commands wait. So I think because of the commands being queued one after another the processing or execution time was longer than expected.

3)

The bottleneck causing this behavior was probably the “Serial communication buffer” between the computer (ROS node) and the hardware. Although commands were being sent instantly the serial link at 115200 bps and hardware’s limited processing speed caused incoming messages especially frequent velocity updates to queue up in the serial buffer. As a result the rover’s hardware often executed older, delayed commands instead of the most recent “STOP” signal. This hidden backlog made the rover respond inconsistently, sometimes obeying commands immediately and other times lagging behind, overshooting turns or failing to stop at all.