LinuxCNC Real-Time Evaluation

Stephen Decker, Jeremy Tang, Ethan Glassman

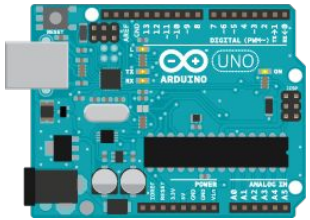
Project Recap

- We intend to evaluate real-time performance of the LinuxCNC operating system environment on a laptop.
- We intend to use an Arduino emulating a CNC mill as a workload.
- In this way we can test the real-time performance of the kernel while performing I/O operations without needing a large, complex, and dangerous real load.

Debian Linux Distribution With Real-Time Kernel

Sends directions to workload

Receives results from workload



Microcontroller

Timeline/Goals

- Pre Planning: Selecting a good data set and simulated load calculation. Complete by 9/25 ✔
- Hardware Selection and Setup: We have not yet selected between the Raspberry Pi and an Arduino. We need to finalize our hardware setup and purchase cabling, install LinuxCNC, etc. Complete by 10/2 ✔
- Initial Implementation: Code simulated load calculation on Raspberry Pi/Arduino. Setup desktop with data set and I/O operations. Complete by 10/9 ✔
- Real-time evaluation setup: Code desktop with software to time evaluation of I/O performance using high accuracy timing APIs (e.g. rdtsc), and begin to collect data on timing parameters. Complete by 10/16 ✔
- Prepare demo 1: Presentation of progress so far. Presentation on 10/21
- Real-time evaluation: Continue to collect data and refine desktop code. Perform ~1 million tests to determine if schedule is ever missed. If schedule is missed, determine a best guess of what is causing the scheduling to be missed and during what scenarios this might occur. Complete by 11/6 ✖
- Prepare demo 2: Presentation of progress so far. Presentation on 11/9 or 11/11

We hit a snag...

- Located and installed code on Arduino to allow Linux CNC to recognize Arduino as a CNC mill
  - https://github.com/dewy721/EMC-2-Arduino/tree/master/Downloads/HAL2Arduino
- We can connect the Arduino to LinuxCNC as a workload and run GCode machining programs
- It turns out that the communication from Arduino <-> Laptop over USB is not performed in a way that we can capturing timing information
- Directions sent via USB in a large group to Arduino
  - Arduino does not appear to send back data in a way we can capture it
  - Seems to be a fundamental problem with USB communication in HAL2Arduino (speed limits)
- We can’t time the Arduino receiving and executing tasks :(

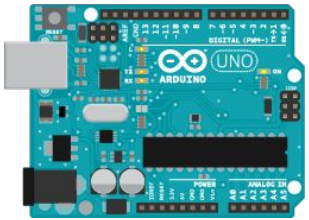
LinuxCNC Architecture

- Uses Python script to establish communication with Arduino
- Arduino expected to send certain information (firmware version, command list, etc)
  - We can see this information being sent from Aduino to Laptop
  - If we change Arduino’s response code, LinuxCNC will not initialize correctly
- However, while actually executing CNC tasks, LinuxCNC does not seem to expect or process responses (unplugging Arudino does not stop LinuxCNC execution)

Debian Linux
Distribution With
Real-Time Kernel

Sends directions to workload

~~Receives results from workload~~



Microcontroller

Remember: Linux CNC vs Vanilla Linux

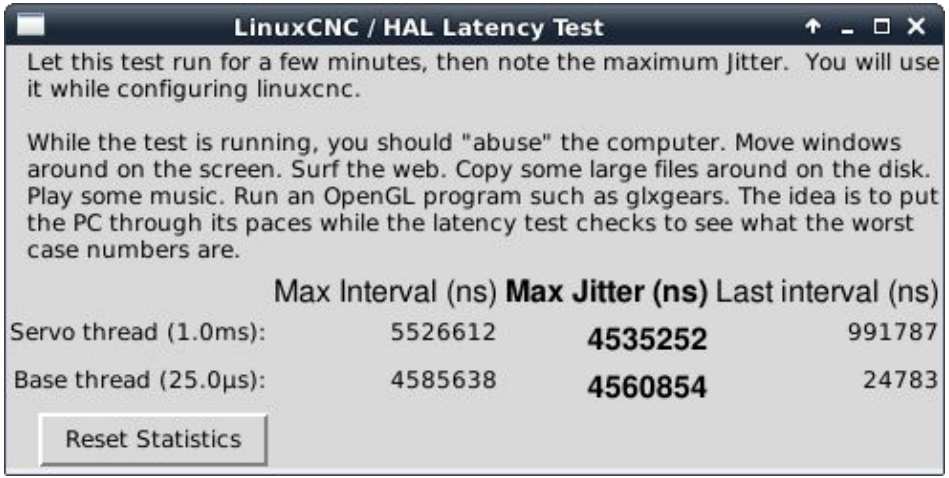
- Real-time kernel exists as a layer between hardware and normal kernel
- Hardware interrupts handled by Real-Time kernel
- Normal kernel sees hardware interrupts as software interrupts
- User services given highest priority, normal kernel services given lowest
- Other kernel config modifications (high resolution timers, no CPU frequency scaling)

We hit a snag … continued

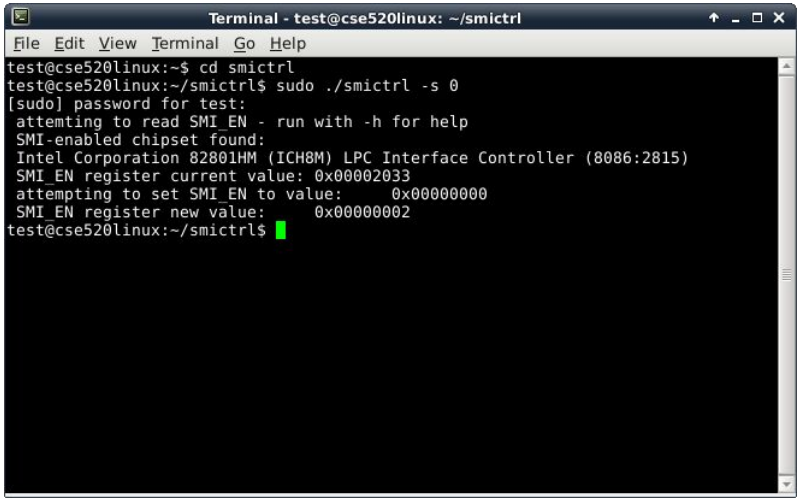
- What this means is we can’t really use standard timing analysis software like hourglass/cyclic test to time anything but realtime of the rest of the Linux
- Which is not actually testing LinuxCNC

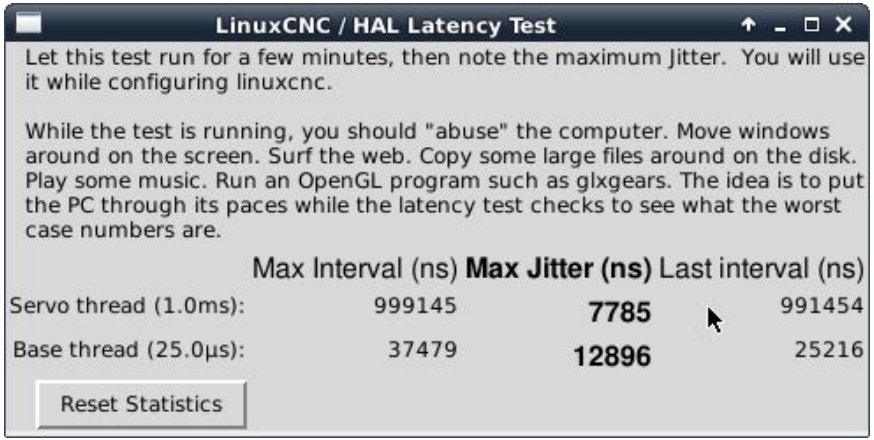
Other real-time snags: SMI

- System Management Interrupts (SMI)
  - Part of Intel's System Management Mode.
  - Power management, security, error handling.
  - BIOS-level interrupt
- Interrupts the CPU for up to milliseconds at a time.
  - Huge latency!
  - Decimates real-time performance!



Solution: disable SMI using smictrl

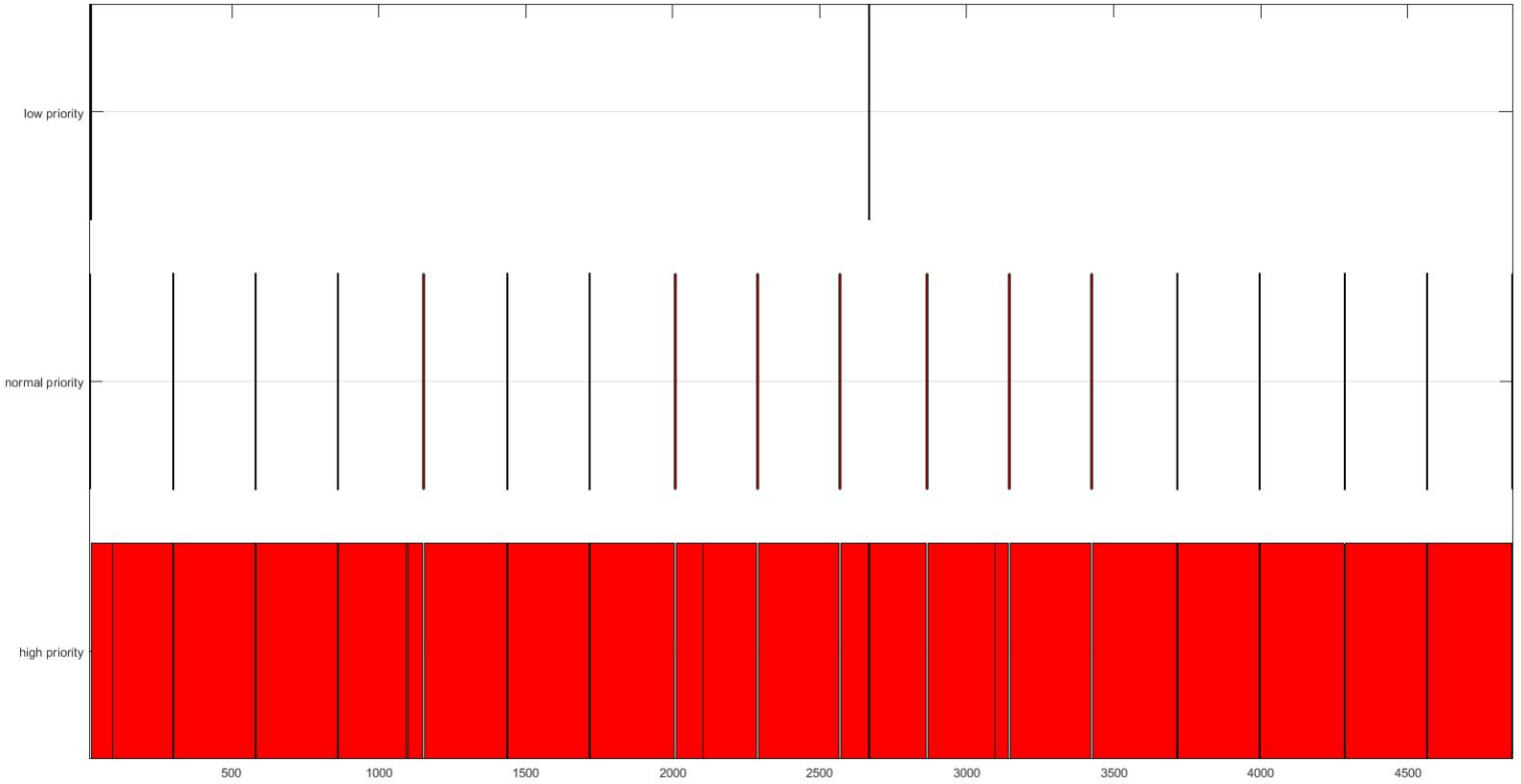




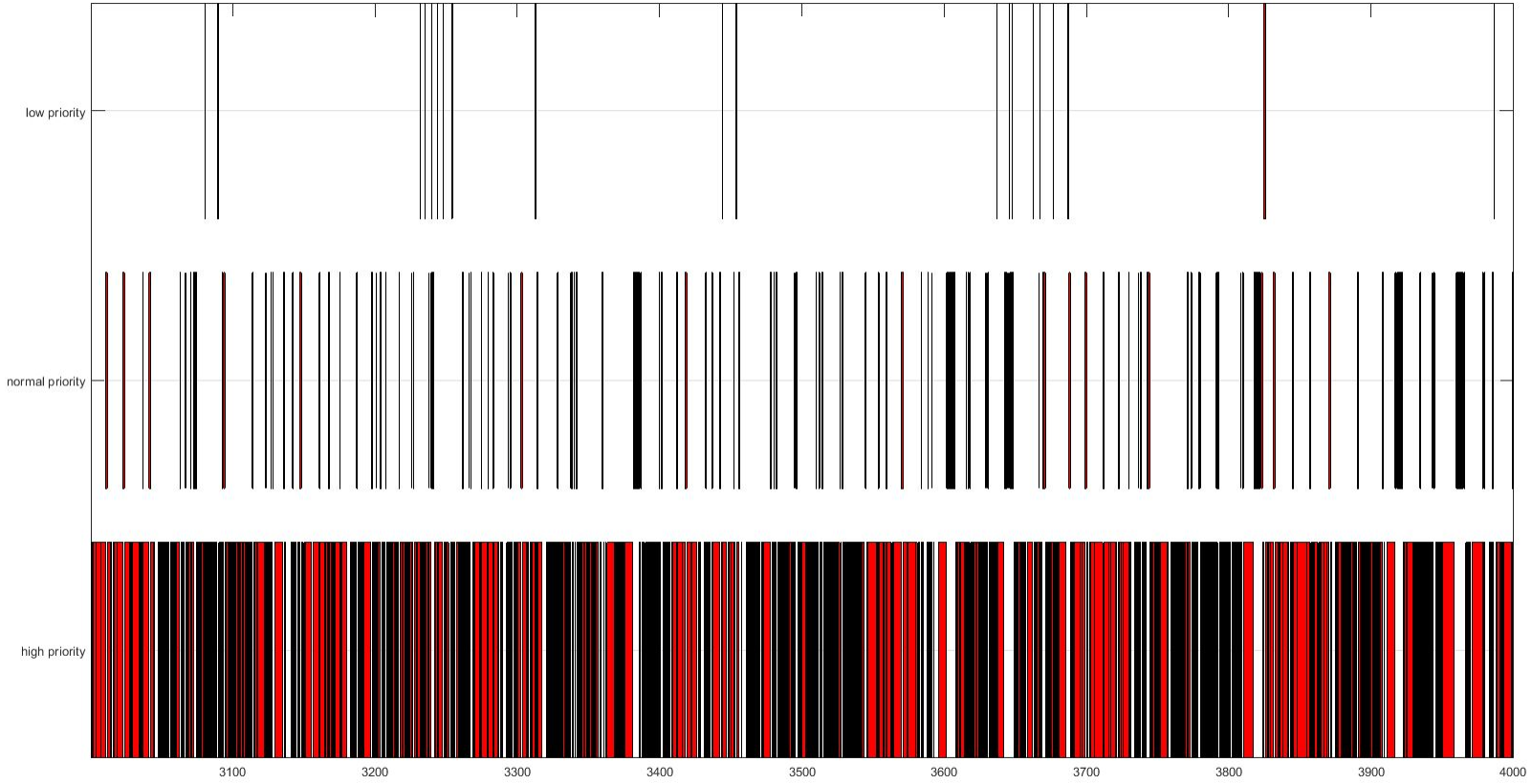
Hourglass?

- Even if we don’t intend to test the realtime-ness of the Linux OS...
- …we can still use Hourglass to observe the behavior of LinuxCNC by seeing how it interacts with other real-time processes.
- Limit to 1 CPU using taskset

Non-real-time threads without LinuxCNC



Non-real-time threads with LinuxCNC



Real-time threads with LinuxCNC

- No data output in a 5 second period.
- Real-time threads preempt other non-realtime processes such as the Window Manager
  - LinuxCNC GUI locks up
- ...but LinuxCNC takes priority over Hourglass threads with RTHIGH priority.
  - GUI resumes with the cursor location further along from when the GUI froze.
  - LinuxCNC never stopped sending commands to the Arduino.

How we are going to proceed:

- Need a test that is analysing the use of the HAL and other changes specific to Linux CNC
- Want to build our own test code. Some possibilities:
  - Write to Arduino, record time of sending message
    - Problem - no easy way to have arduino write record to file
    - Need to write our own high resolution timers for Arduino
  - Switch back to Rasperry Pi, record receiving messages
    - Problem - need high resolution timers
  - Write to disk of laptop
    - Problem - testing HAL for LinuxCNC? Probably not!
  - Keep using LinuxCNC as is, come up with ways to record indirectly
    - Problem - Are we testing the right things?

Still trying to decide best way to proceed. Will start with Arduino as we have it set up.

Questions?