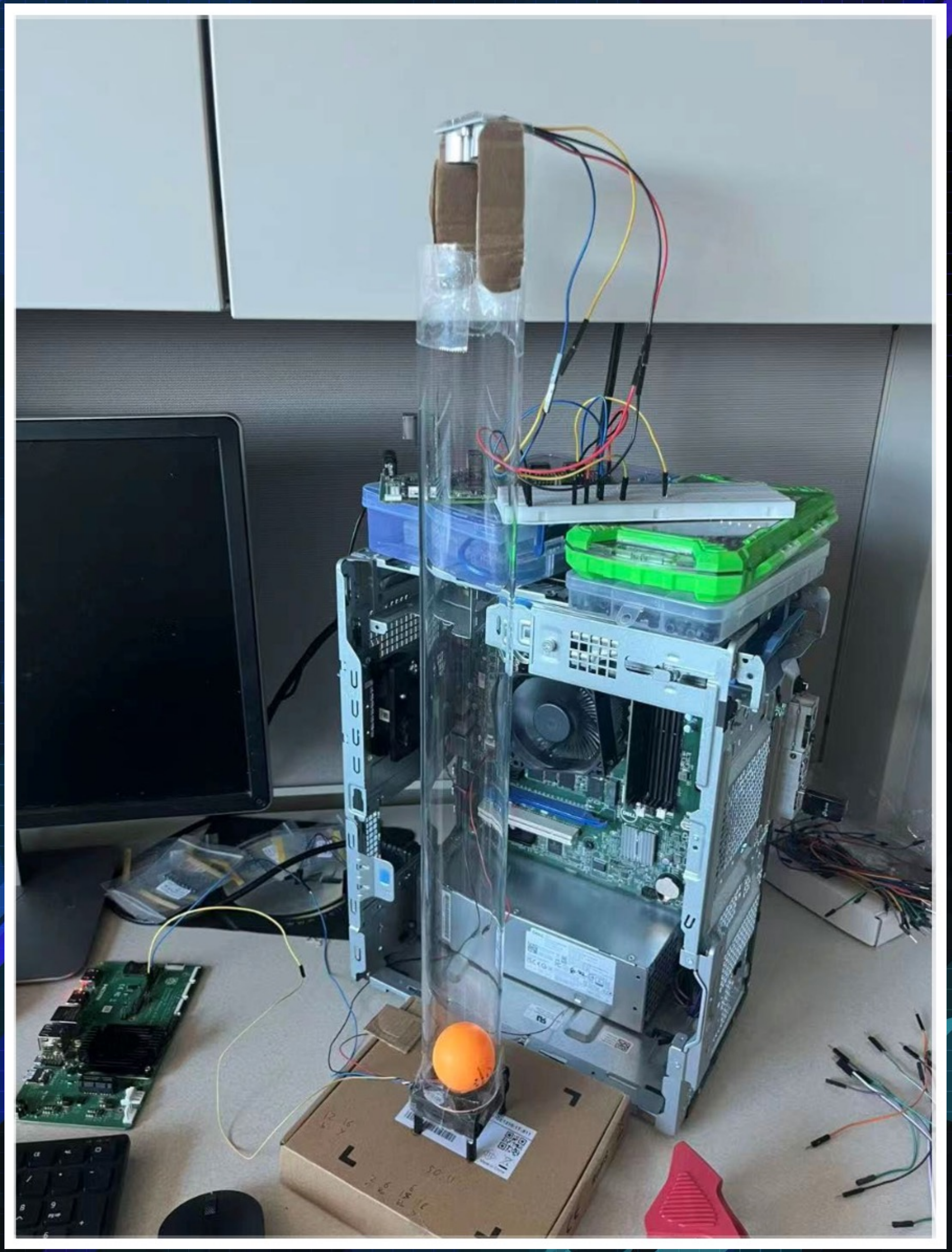
“Experimental Performance Evaluation of Real-Time Traffic Scheduler in Control Systems”

Progress Report II

SE5402/CSE5312: ARCHITECTURE OF INTERNET OF THINGS

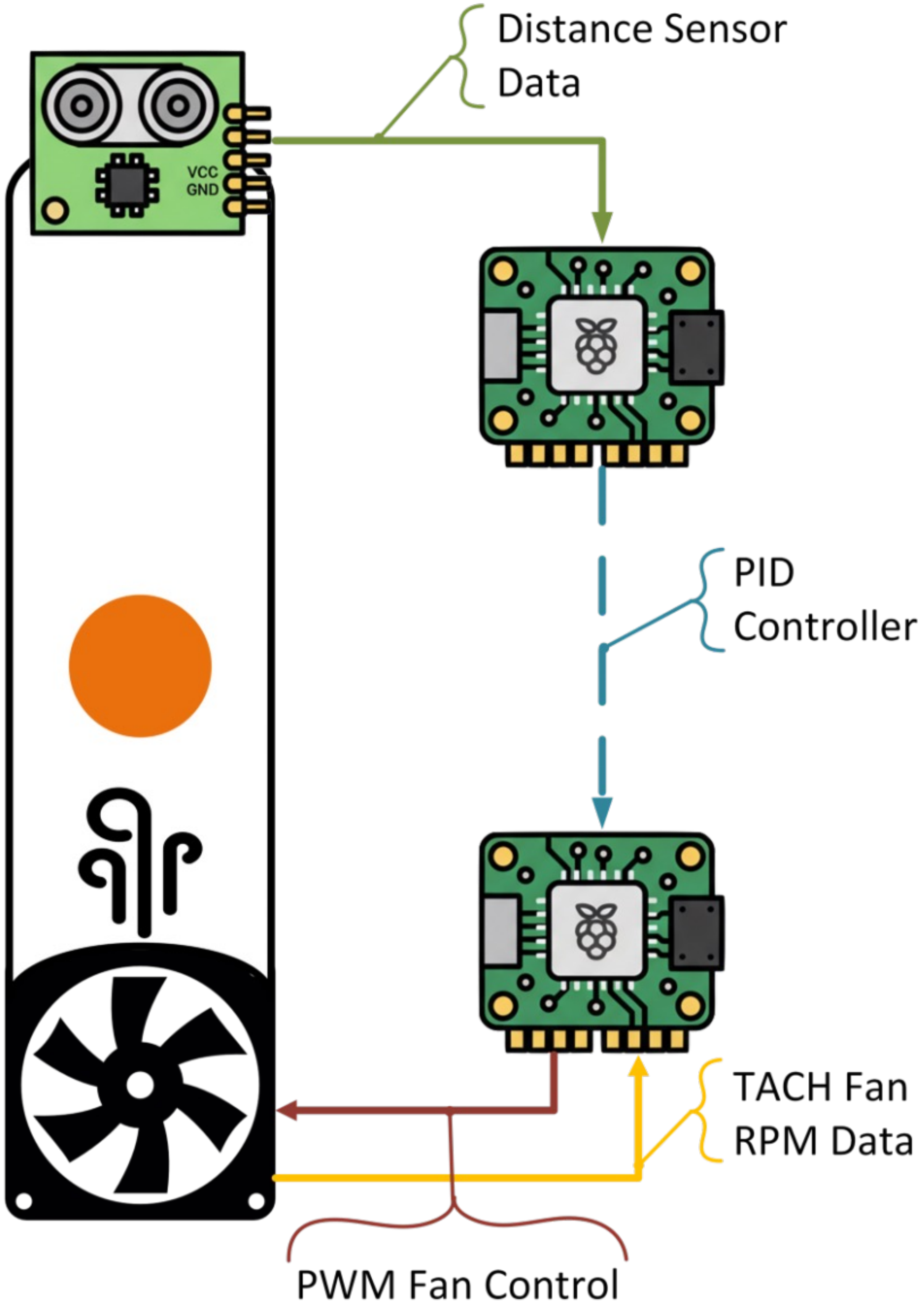
FALL 2025

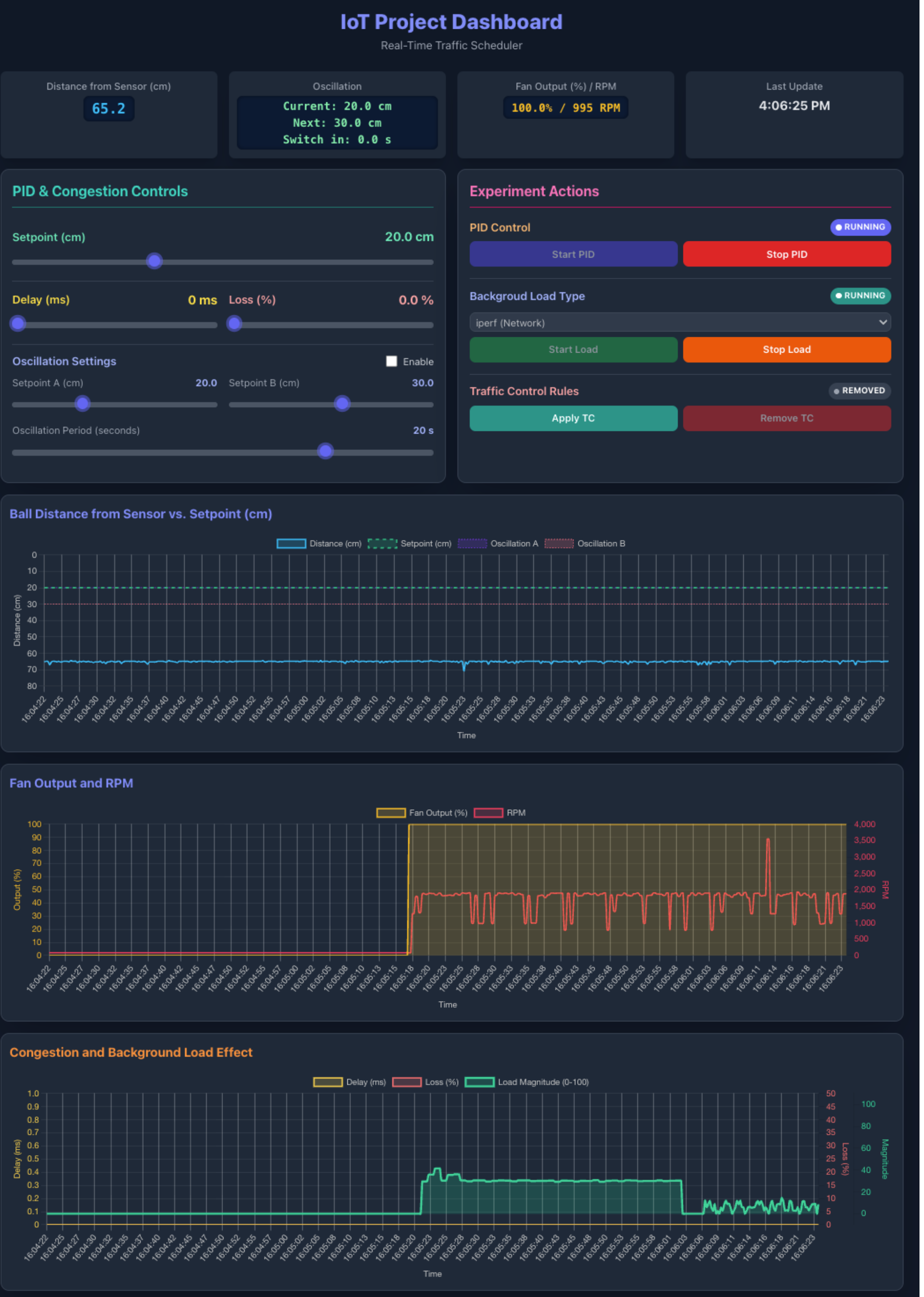
ABBY HORNING & JAKE THURMAN



Project Recap

Basic PID loop stabilizing the ball → Fan node and sensor node communicating → Simple Network Architecture → No Experiment Framework Yet → No Dashboard or Visualization → Showed a demo of single setpoint

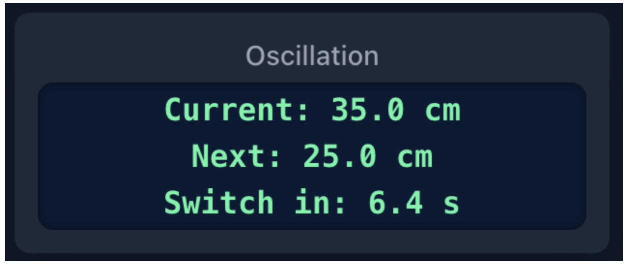


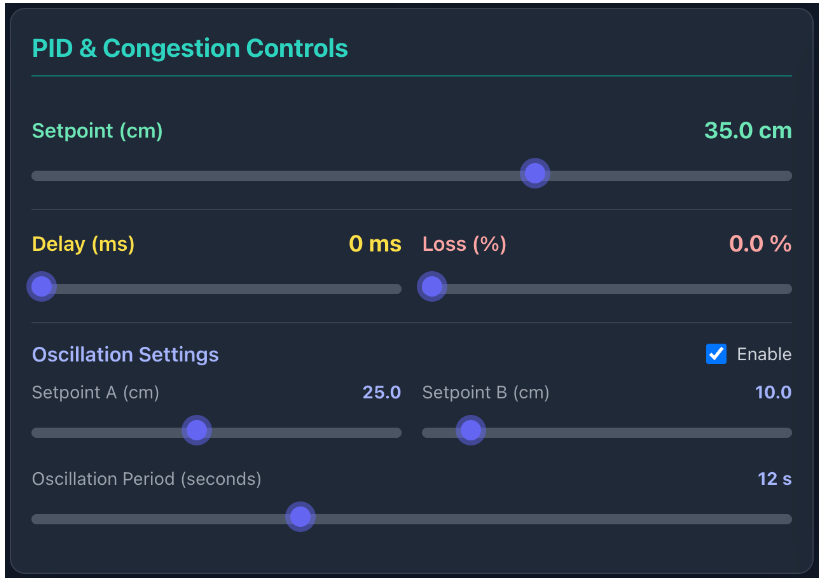


Dashboard

Master Controller (Web App):

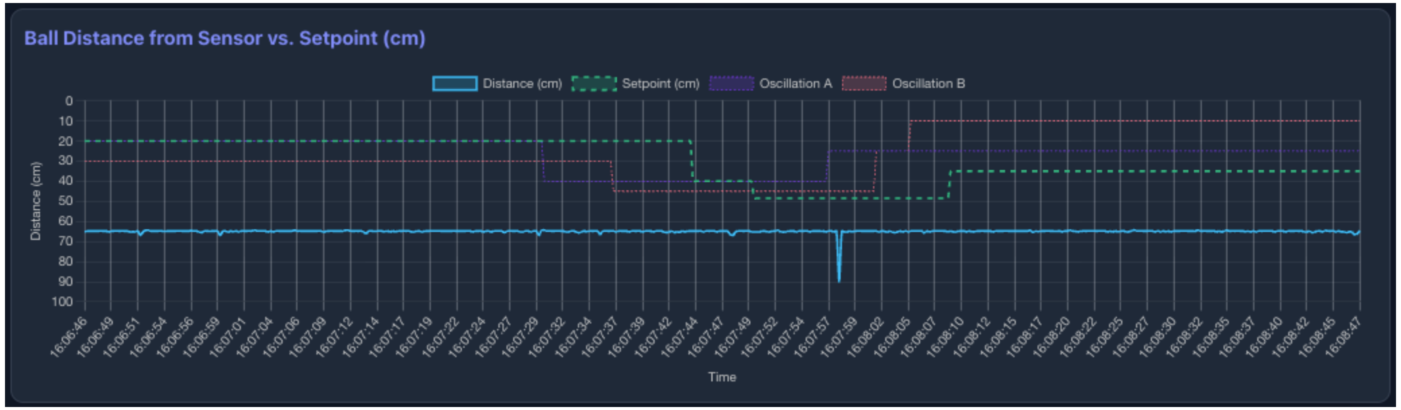
- Receives telemetry (Distance, Duty, Setpoint, Congestion).
- Provides real-time visualization and user configuration (setpoint, oscillation).

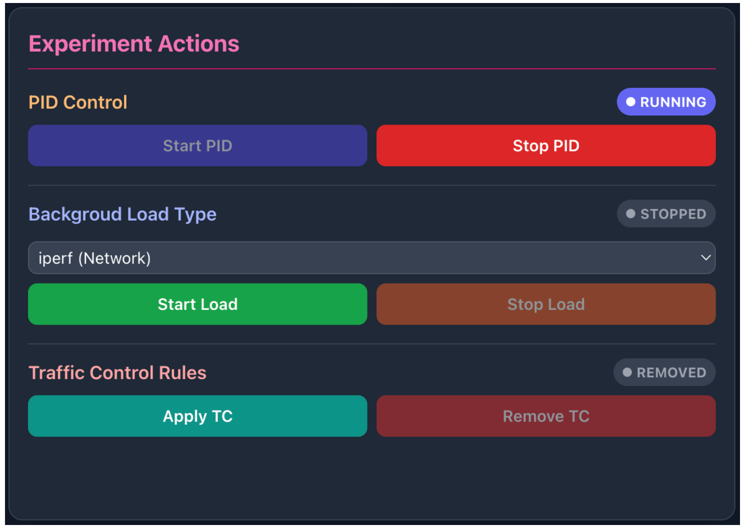


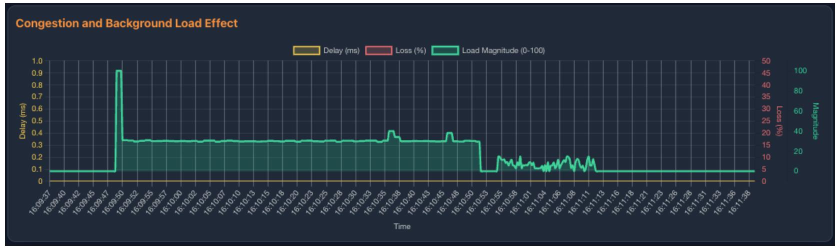


Setpoint Oscillation Implemented

- Dynamic switching between two heights (example: 20 cm ↔️ 30 cm)
- Built **setpoint_config.json** for configuration
- Added timer for oscillation period
- This is the core feature that enables *network congestion testing*
- Retuned PID parameters accounting for oscillation







Congestion Injection System

Dynamically configurable delay (latency) and loss_rate (packet loss).

Applied directly within the pid_control_thread_func just before the fan command is sent.

Custom experiment manager module with telemetry and callbacks

Network flooding (**iperf3**)

CPU stress test (**stress-ng**)

Traffic Control

Implemented with Linux **tc** utility

Multi-Threaded Architecture for Core Pipelines

Added dedicated threads for:

- PID Control Loop
- Sensor Telemetry Sender
- Experiment Manager / Congestion Updater

Ensures timing-critical tasks (e.g., PID updates at high frequency) do not get delayed by I/O or dashboard communications.

Challenges

1

Dashboard reconnection issues

2

Ball dropping rapidly when PID direction was reversed

3

Fan failing to generate expected lift

4

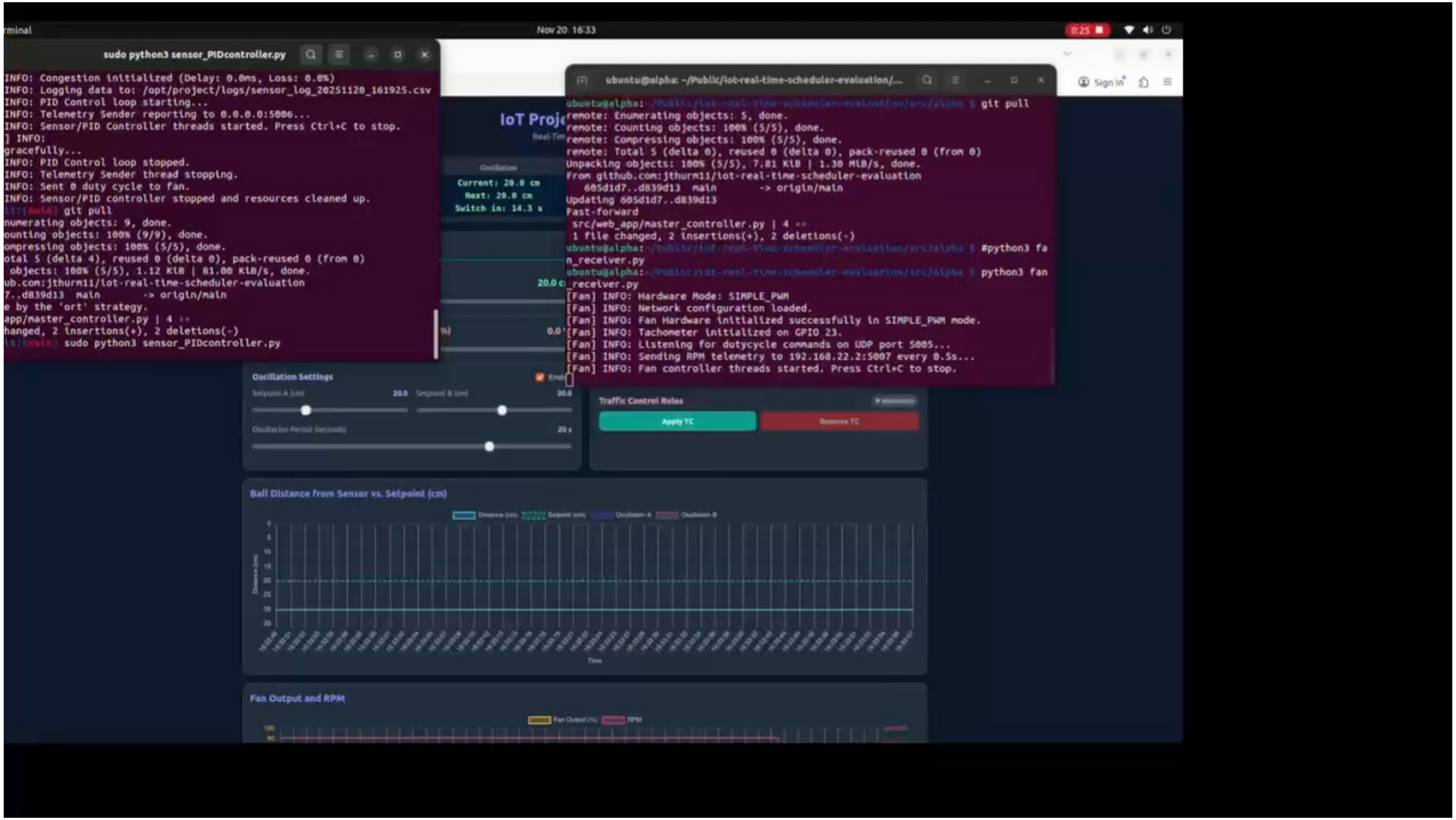
Hardware failures/concerns:

- Broken fan blade
- Sensor thread lockup

Final Steps

- Execute Congestion Experiments: Run the core experiment comparing Degraded vs. Prioritized network conditions using utility scripts.
- Final Analysis: Analyze the logged data (RPM, Duty Cycle, Time) and plot the results to visually demonstrate the benefit of the real-time scheduler.
- Final Report: Complete the final report summarizing the system design, experimental methods, results, and analysis comparing degraded and prioritized network conditions.

Oscillation Demo



References

- Salzmann et al. (2025): Hovering a ping-pong ball: A demonstration setup for teaching PID control
  - (https://doi.org/10.26434/chemrxiv-2025-328tk)
- An example of how Linux traffic scheduler improves control under traffic congestion:
  - (https://github.com/NXP/dds-tsn)
- Depth/Distance Sensors:
  - (https://randomnerdtutorials.com/complete-guide-for-ultrasonic-sensor-hc-sr04/)
- Tuning PID parameters:
  - https://davidr.no/iiav3017/papers/Ziegler_Nichols_%201942.pdf
- PWM Fan Control:
  - PWM basics: https://www.arduino.cc/en/Tutorial/Foundations/PWM
  - PWM fan control: https://github.com/folkhack/raspberry-pi-pwm-fan-2

https://github.com/jthurm11/iot-real-time-scheduler-evaluation