# SE5402/CSE5312 Architecture of Internet of Things

## "Experimental Performance Evaluation of Real-Time Traffic Scheduler in Control Systems"


### Team Members:

- Abby Horning (abigail.horning@uconn.edu)
- Jake Thurman (jacob.thurman@uconn.edu)

Project owner: Chuanyu Xue (chuanyu.xue@uconn.edu)


### 1. Project Motivation and Objective(s)

Modern day control systems including drones, autonomous vehicles, and surgical robots all rely on fast, reliable communication between sensors and controls. When network traffic becomes congested, critical control messages can be delayed, leading to instability and degraded performance. Real-time traffic scheduling addresses this challenge by prioritizing these critical messages, improving timing predictability, and enhancing network performance.

The core objective of this project is to experimentally evaluate how real-time traffic scheduling enhances control quality in a cyber-physical system. We will demonstrate this using a physical ball-floating testbed in which the sensing and control units are connected over a network (via Wi-Fi in our setup). This setup will allow us to observe the negative effects of network congestion on control performance and then apply a real-time traffic scheduler to improve system stability and resilience.

This project is a hardware-based system that will provide practical experience in several key areas:

- Understanding the inner workings of the **Linux networking stack**.
- Gaining familiarity with **real-time scheduling** mechanisms.
- Developing and applying a **PID controller** for system stabilization.
- Building practical skills in both **C and Python programming**.
- (Optional) Exploring advanced protocols such as **TSN (Time-Sensitive Networking)**.

### 2. High-level Design

Our project will utilize a pre-built ball-floating testbed, which consists of two Raspberry Pi Compute Module 4 (**RPi-CM4**) devices connected via Wi-Fi. The project will be executed in three main phases.

### System of Interest (SOI)

The testbed itself is a classic ball-and-beam system, but in a vertical orientation. The physical testbed design and baseline control are based on the "PingPongPID" system by Salzmann et al. [1], which we are extending for network evaluation. It consists of two main nodes:

- **Actuator Node** (hostname: *alpha*): An RPi-CM4 located at the bottom connected to a PWM fan. Its primary role is to receive control signals and adjust the fan's speed to control the airflow, thereby changing the ball's position.
- **Sensor Node** (hostname: *beta*): An RPi-CM4 located at the top of the testbed connected to a depth sensor. Its primary role is to monitor the ball's position and transmit this data over the network to the control node.

### Project Phases

1. **Baseline Control**: First, we will implement a PID controller to stabilize the ball at a target position. This phase establishes a performance baseline under ideal network conditions.
2. **Performance Degradation**: We will intentionally introduce network congestion (e.g., unexpected traffic) between the RPi-CM4's to demonstrate the degradation in control quality. This step will highlight the problem that real-time schedulers are designed to solve.
3. **Real-Time Scheduler Evaluation**: We will then implement a real-time traffic scheduler (e.g., Linux ETF) on the network and evaluate its effectiveness by comparing the control performance against the degraded baseline.

## 3. Hardware and Software Requirements

### Hardware Requirements

- Actuator Node (*alpha*): Raspberry Pi Compute Module 4 (**RPi-CM4**)
- Sensor Node (*beta*): Raspberry Pi Compute Module 4 (**RPi-CM4**)
- Sensors: HC-SR04 ultrasonic distance sensor
- Actuator: 12v PWM fan
- Support: Jumper wires, breadboard, resistors (for voltage divider circuit)

### Software Requirements

- Operating System: Raspberry Pi OS (64-bit) / Ubuntu 24.04
- Programming Languages: C and Python
- Libraries: Libraries for GPIO control (e.g., `gpiozero`, `pigpio`, or `RPi.GPIO`) and socket programming.
- Scheduling Tools: Linux traffic control (`tc`) for creating network congestion and potentially for scheduling.
- Version Control: Git/GitHub for collaborative development.

## 4. Team Members and Task Assignments

Our team will divide responsibilities to establish clear ownership over each activity and to ensure the project's success.

### Abby Horning (Primary Tasks)

- **PID Controller Implementation**: Implement and tune the core Proportional-Integral-Derivative (PID) algorithm to achieve stable control of the ball's position. This includes programming the control logic, handling sensor input, and generating the fan's PWM output.
- **Data Logging and Visualization**: Develop Python scripts to capture and log all necessary experimental data (ball position, network metrics, control signals). Create charts and plots for performance visualization and analysis used in the final report.

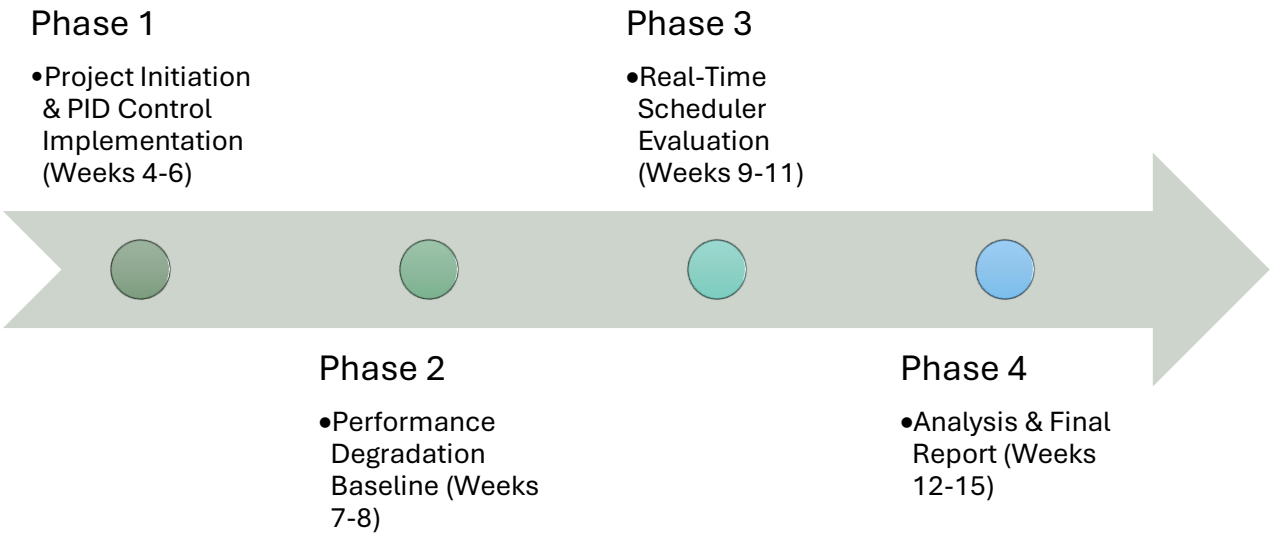
### Jake Thurman (Primary Tasks)

- Network Congestion Mechanism: Develop and implement low-level C code and Linux scripts to generate synthetic traffic on the Wi-Fi link, intentionally causing performance degradation for the baseline comparison.
- **Real-Time Scheduler Configuration**: Research, configure, and apply the chosen real-time traffic scheduler (e.g., Linux ETF) using Linux Traffic Control (`tc`) utilities on both Raspberry Pi Compute Modules.

### Collaborative Tasks (Abby & Jake)

- **System Assembly and Configuration**: Handle the physical setup of the testbed, including all necessary hardware wiring, and the initial configuration of the Raspberry Pi Compute Module operating systems (OS) and Wi-Fi networking.
- **Integration Testing**: Jointly test the assembled hardware and integrated software components (controller and networking stack) at each phase milestone.
- **Data Analysis and Interpretation**: Conduct a joint analysis of all collected data to draw final conclusions regarding the effectiveness of the real-time scheduler.
- **Documentation and Presentation**: Collaborate on writing the final project report and preparing all required midterm and final presentations.

## 5. Deliverables and Timeline




- Phase 1: Project Initiation & PID Control Implementation (Weeks 4-6)
    - Milestone: Proposal Submission (September 27)
    - Activities: Finalize the project plan, acquire necessary materials, and implement the baseline PID controller to stabilize the ball.
- Phase 2: Performance Degradation Baseline (Weeks 7-8)
    - Milestone: Successful demonstration of network congestion and performance degradation.
    - Activities: Introduce network congestion on the network link and record the resulting control quality degradation. This establishes our baseline for performance.
- Phase 3: Real-Time Scheduler Evaluation (Weeks 9-11)
    - Milestone: Implementation of the real-time traffic scheduler and data collection.
    - Activities: Implement and apply a real-time traffic scheduler (e.g., Linux ETF or KeepON) to the network. Conduct experiments to compare the improved control performance against the degraded baseline.
- Phase 4: Analysis & Final Report (Weeks 12-15)
    - Milestone: Final project report and presentation submission (December 12).
    - Activities: Analyze all collected data, focusing on key trade-offs in cost, power, and performance. Prepare the final report and presentation for submission.

## 6. References


[1] C. G. Salzmann, S. M. Vecchi Marsh, J. Li, and L. Slater, “Hovering a ping-pong ball: A demonstration setup for teaching PID control,” *Univ. College London, Dept. of Chemistry*, London, U.K., 2025.

[2] C. Xue, "Experimental Performance Evaluation of Real-Time Traffic Scheduler in Control System," Couse Project Proposal, 2025.

[3] Arduino, “Analog Output — Arduino Documentation,” *Arduino Docs*. [Online]. Available: https://docs.arduino.cc/learn/microcontrollers/analog-output/.

[4] Folkhack, “raspberry-pi-pwm-fan-2,” *GitHub repository*. [Online]. Available: https://github.com/folkhack/raspberry-pi-pwm-fan-2.

[5] N. Nguyen and M. Mondal, “Mobile Sink Trajectory Planning in WPCNS,” *Sample Proposal*, 2021.

[6] S. Wrótniak, “Improving Wireless Firmware Update Protocol for IoT Devices,” *Sample Proposal*, 2021.

[7] NXP, “DDS-TSN: Example project of DDS-TSN integration,” *GitHub repository*. Apache-2.0 License, 2022. [Online]. Available: https://github.com/NXP/dds-tsn.

[8] R. Santos, “Complete Guide for Ultrasonic Sensor HC-SR04 with Arduino,” *Random Nerd Tutorials*, 2019. [Online]. Available: https://randomnerdtutorials.com/complete-guide-for-ultrasonic-sensor-hc-sr04/.