Smart Room Control System

Modeling, Simulation and Embedded Control using STM32 and Python

Author: **Juan Marin**
Institution: ASIMOV

November 4, 2025

Abstract

This document describes the design, modeling and implementation of a closed-room environmental control system. The physical plant (temperature, humidity and CO_2) is simulated in Python, while the estimation and control run on an STM32F767ZI microcontroller. Systems communicate through UART over USB. A Python-based simulation acts as the plant and logger; **no GUI was used** in the experiments reported here.

Contents

1 Introduction

This project aims to design a smart indoor climate control system with a realistic yet computationally simple dynamic model. The goal is to maintain comfortable levels of temperature, humidity and CO_2 concentration using two actuators: an electric heater and a ventilation fan.

1.1 System Overview

The system consists of:

- **Python Plant:** Simulates environmental dynamics and sensors; sends measurements and receives control via UART.
- **STM32 Controller (NUCLEO-F767ZI):** Implements a 3-state Kalman Filter, a 1D Kalman Filter for occupancy, and the control law executed at 10 Hz.
- **UART Communication:** ASCII packets at 115 200 bps.

1.2 Firmware and Repository Layout

- Board: **NUCLEO-F767ZI**
- Firmware path: `/stm32_controller/smart_room_control_v1`
- Report: `docs/smart_control_room.pdf`

1.3 Experiment Command (Online with MCU)

The online experiment was executed with:

```
python python_plant/main.py --mode online --port COM11 --baud 115200 --
    Ts 0.01 --duration 60 --sim-speed 10 --noise-mult-T 20 --noise-mult-
    w 2 --noise-mult-c 20 --heater-scale 5 --fan-scale 20
```

1.4 Setpoints

Unless otherwise stated, the reference values used were:

$$T_{\text{ref}} = 21\,^{\circ}\text{C}, \quad w_{\text{ref}} = 0.006\ \text{kg/kg}, \quad c_{\text{ref}} = 800\,\text{ppm}.$$

2 Physical Model and State Variables

The room is modeled as a well-mixed air volume. The state vector is defined as:

$$x = \begin{bmatrix} T \\ w \\ c \end{bmatrix}, \qquad u = \begin{bmatrix} u_h \\ u_f \end{bmatrix}, \qquad d = \begin{bmatrix} T_o \\ w_o \\ c_o \\ N \end{bmatrix}. \tag{1}$$

3 Mathematical Modeling

3.1 Continuous-Time Dynamics

$$\dot{T} = \frac{q}{V}(T_o - T) + \frac{\eta_h P_h}{\rho c_p V} u_h + \frac{Q_{pers}}{\rho c_p V} N, \tag{2}$$

$$\dot{w} = \frac{q}{V}(w_o - w) + \frac{G_w}{\rho V} N, \tag{3}$$

$$\dot{c} = \frac{q}{V}(c_o - c) + \gamma_c N, \tag{4}$$

where $q = q_{\max} u_f + k_{\text{stack}}(T - T_o)$.

3.2 Discretization and Linearization

Using forward Euler with sample time T_s:

$$x_{k+1} = x_k + T_s\, f(x_k, u_k, d_k). \tag{5}$$

Around an operating point $(x^\star, u^\star, d^\star)$ we obtain

$$\delta x_{k+1} = A\delta x_k + B\delta u_k + E\delta d_k, \tag{6}$$

with $A = I + T_s \frac{\partial f}{\partial x}$, $B = T_s \frac{\partial f}{\partial u}$, and $E = T_s \frac{\partial f}{\partial d}$.

4 State Estimation (Kalman Filters)

4.1 3-State Kalman Filter for (T, w, c)

A discrete-time linear Kalman Filter estimates the vector $x = [T, w, c]^T$. The measurement is $z = [T, w, c]^T$ (direct but noisy). The implemented filter uses the linearized, discretized model at each step:

$$\text{Predict:} \quad \hat{x}_{k|k-1} = A_k \hat{x}_{k-1|k-1} + B_k u_{k-1} + E_k d_{k-1}, \tag{7}$$

$$P_{k|k-1} = A_k P_{k-1|k-1} A_k^\top + Q, \tag{8}$$

$$\text{Update:} \quad K_k = P_{k|k-1} H^\top (H P_{k|k-1} H^\top + R)^{-1}, \quad H = I_3, \tag{9}$$

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k(z_k - H\hat{x}_{k|k-1}), \tag{10}$$

$$P_{k|k} = (I - K_k H) P_{k|k-1}. \tag{11}$$

Initialization used (matching firmware):

$$x_0 = [20.0,\ 0.006,\ 800]^T, \quad R = \text{diag}(0.10^2,\ (10^{-4})^2,\ 20^2), \quad Q = \text{diag}(4{\times}10^{-4},\ 10^{-8},\ 25).$$

4.2 1D Kalman Filter for Occupancy N

Occupancy is slowly varying. A scalar KF is applied:

$$N_{k+1} = N_k + w_k, \qquad w_k \sim \mathcal{N}(0, Q_N), \tag{12}$$

$$y_k = N_k + v_k, \qquad v_k \sim \mathcal{N}(0, R_N). \tag{13}$$

Parameters used: $Q_N = 10^{-6}$, $R_N = 10^{-2}$, with an initial variance $P_0 = 1$.

5 Embedded Control Law

5.1 Implemented PI-Based Control (Firmware)

In the reported experiments the embedded controller uses two PI loops with anti-windup and rate limits:

- Temperature loop $\rightarrow u_h$: gains $K_p^T = 0.80$, $K_i^T = 0.05\ \mathrm{s}^{-1}$, back-calculation $K_{aw}^T = 0.5$.
- Ventilation loop $\rightarrow u_f$: a combined error e_F mixes humidity and CO_2:

$$e_F = K_w \frac{w_{\mathrm{ref}} - \hat{w}}{0.002} + K_c \frac{\hat{c} - c_{\mathrm{ref}}}{400}, \tag{14}$$

  with $K_w = 0.7$, $K_c = 0.3$, then PI gains $K_p^F = 2.0$, $K_i^F = 0.10\ \mathrm{s}^{-1}$, $K_{aw}^F = 0.5$.

Actuator limits: $u_h, u_f \in [0, 1]$ with per-step rate limits $\Delta u_h \leq 0.10$, $\Delta u_f \leq 0.20$.

5.2 MPC Formulation (Design Reference)

For completeness, the MPC design considered uses the linearized model (A, B) and solves over horizon N_p:

$$\min_{\{\Delta u\}} \sum_{k=0}^{N_p-1} \left\| x_{k+1|t} - x_{\mathrm{ref}} \right\|_Q^2 + \left\| \Delta u_{k|t} \right\|_R^2 \tag{15}$$

$$\text{s.t. } x_{k+1|t} = A x_{k|t} + B u_{k|t} + E d_{k|t}, \quad u_{\min} \leq u_{k|t} \leq u_{\max},$$
$$\Delta u_{\min} \leq \Delta u_{k|t} \leq \Delta u_{\max}. \tag{16}$$

In the current firmware, the PI strategy above was executed during tests; MPC remains as a drop-in controller for future experiments.

6 Communication Protocol (UART over USB)

- Baud rate: 115200 bps, 8N1.
- Python $\rightarrow$ STM32: `<MEAS, T, w, c, N>`
- STM32 $\rightarrow$ Python: `<CTRL, uh, uf>` and optional `<KF, T, w, c, N>`

Example:

```
<MEAS, 22.4, 0.0062, 750, 2>
<CTRL, 0.45, 0.70>
```

7 Results

The online experiment (60 s, $T_s = 0.01\,\mathrm{s}$, control at 10 Hz) tracked the setpoints $T_{\mathrm{ref}} = 21\,^{\circ}\mathrm{C}$, $w_{\mathrm{ref}} = 0.006$, $c_{\mathrm{ref}} = 800\,\mathrm{ppm}$ using the PI controller with the Kalman estimates.

7.1 Artifacts

- Video/Audio summary: `results.wav`
- Figure: see Figure 1.

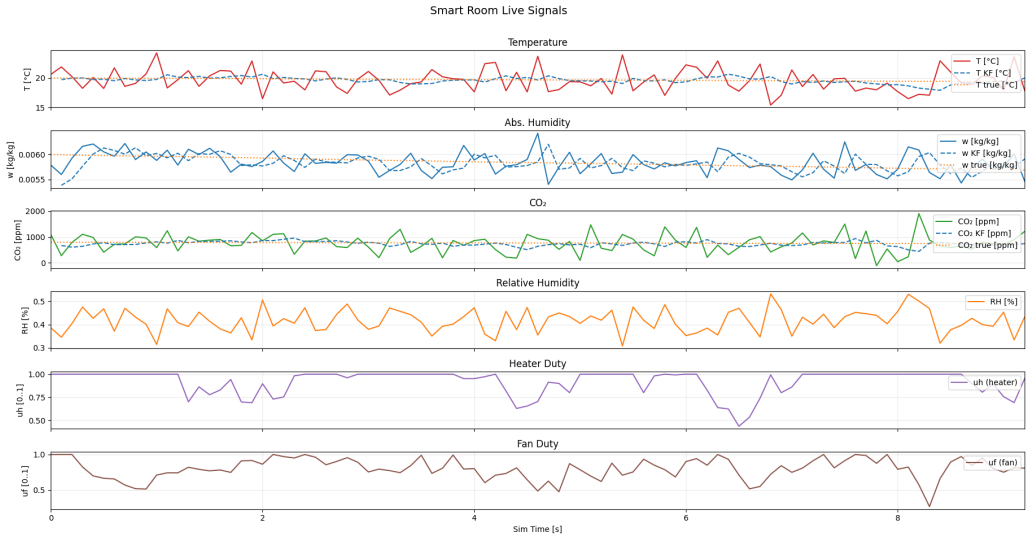

Figure 1: Time histories of states and control inputs during the 60 s online test.

8 Conclusions and Future Work

The combination of a 3-state KF (for T, w, c) and a scalar KF (for N) provided robust state estimates under measurement noise. The PI-based embedded controller yielded stable regulation towards $(T_{\text{ref}}, w_{\text{ref}}, c_{\text{ref}})$ with simple tuning and low computational cost. Future steps include:

- Validate and benchmark the MPC controller under varying occupancy profiles (N time-varying).
- Incorporate additional actuators for humidity and CO_2.
- Long-duration tests and energy-use analysis.