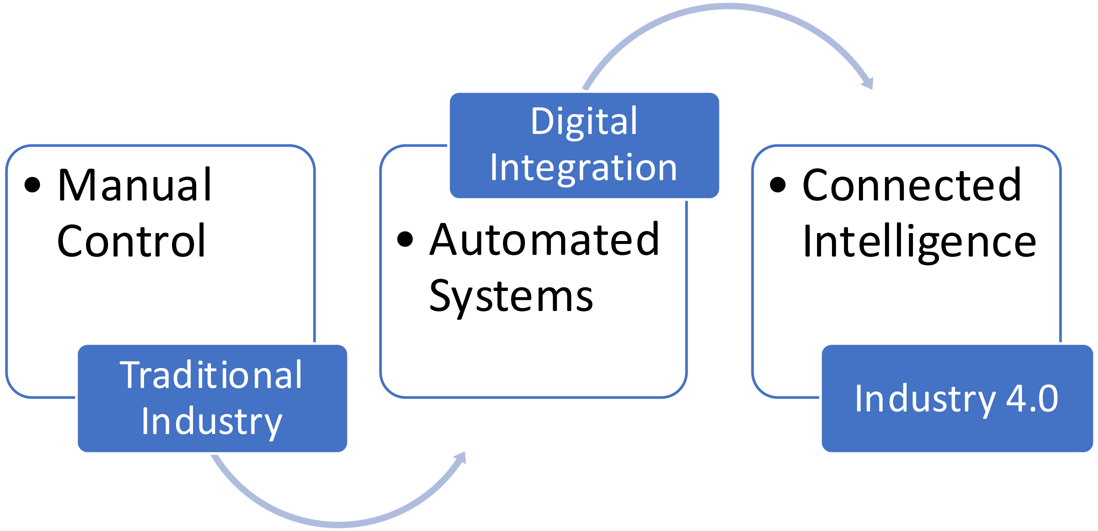
Air Heater Control and Monitoring System

Presented by – Choukha Ram

Industry 4.0 Context



- **Key Transformations**
  - From isolated systems to connected devices
  - From manual monitoring to real-time data
  - From basic security to cyber-security focus

Project Overview

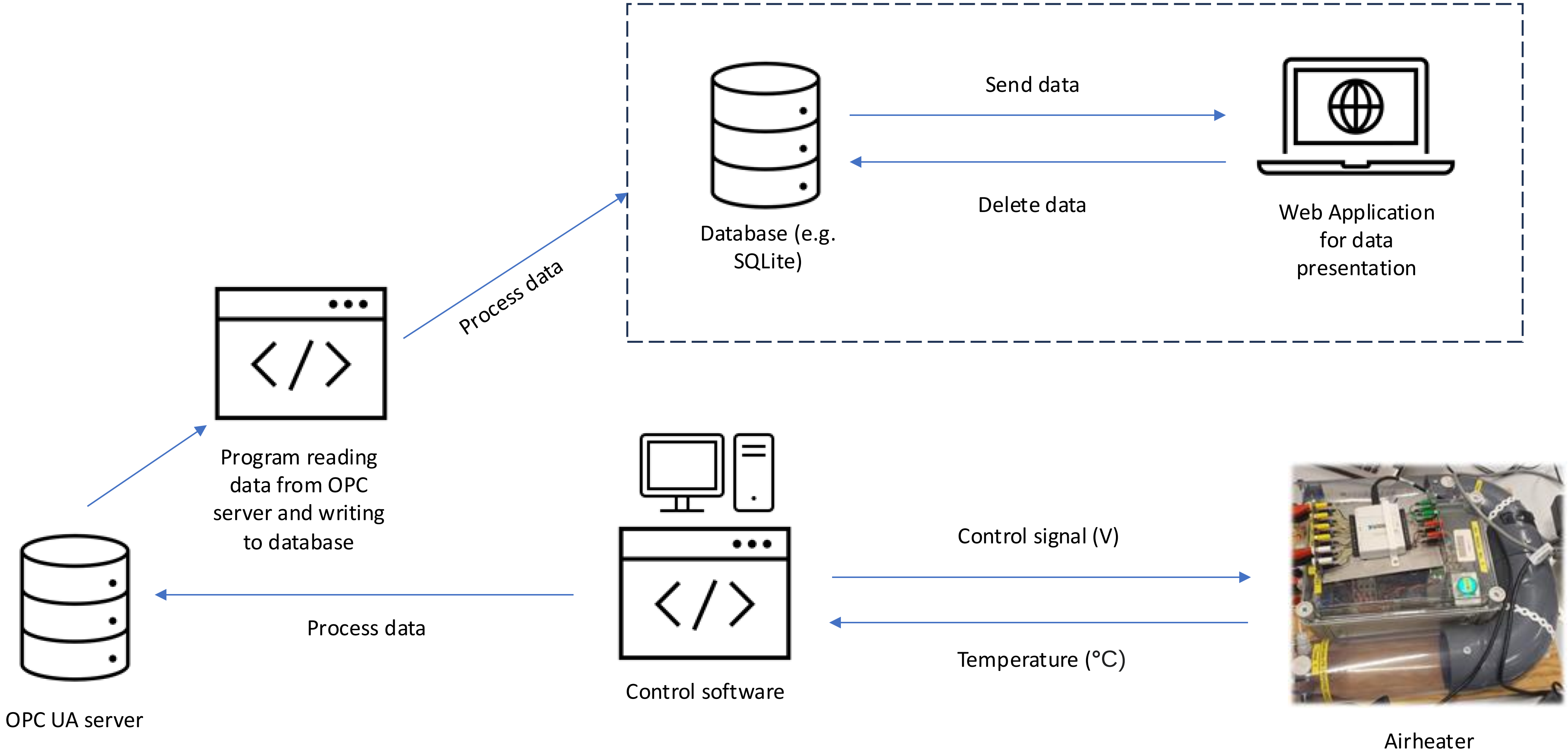
Core components

Air Heater Process → Control System → Monitoring Interface

Scope

- Temperature control implementation
- Real-time monitoring system
- Secure data communication
- Web-based visualization

System design



Airheater model

Air heater model used for simulation of the real process.

This model is discretized using Forward Euler.

$$T_{out}(t) = T_{out}(t) + \frac{T_s}{\theta_t}[-T_{out}(t) + K_h u(t - \theta_d) + T_{env}]$$

Discretization using FE[1]

$$T_{out}(t_{k+1}) = T_{out}(t_k) + \frac{T_s}{\theta_t}[-T_{out}(t_k) + K_h u(t_{k-N}) + T_{env}]$$

$$N = \left\lceil \frac{\theta_d}{T_s} \right\rceil$$

Variables/ parameters	**Description**	**Value**
$T_{out}(t)$	Temperature output from the air heater at time t [°C]	-
u_k	Control signal to heater [V]	-
θ_t	Time constant [s]	22
θ_d	Time delay due to air transportation [s]	2
K_h	Heater gain [°C/V]	3.5
T_{env}	Environmental temperature [°C]	21.5
T_s	Samling time [s]	0.1

(1) FE: Forward Euler

Discrete PI controller

The discrete PI controller is responsible for maintaining the temperature setpoint in the air heater

The controller tuning parameters were found using the ZN method:

$$K_p = 2.0$$

$$T_i = 7.5$$

$$u(t_k) = u(t_{k-1}) + K_p\big(e(t_k) - e(t_{k-1})\big) + \frac{K_p}{T_i} T_s e(t_k)$$

$$e(t_k) = r(t_k) - y(t_k)$$

Variables	Description
$u(t_k)$	Control signal at time t_k [V]
$e(t_k)$	Error at time t_k [°C]
$r(t_k)$	Setpoint at time t_k [°C]
K_p	Proportional term in [illegible]roller [V/°C]
T_i	Integral time in PI controller [s]
T_s	Samling time [s]

Discrete low-pass filter

The low-pass filter will remove high frequent noise from output data (from the air heater). This will result in a more smoothed time series of outputs.

In this presentation a filter time constant of 0.5 seconds were used.

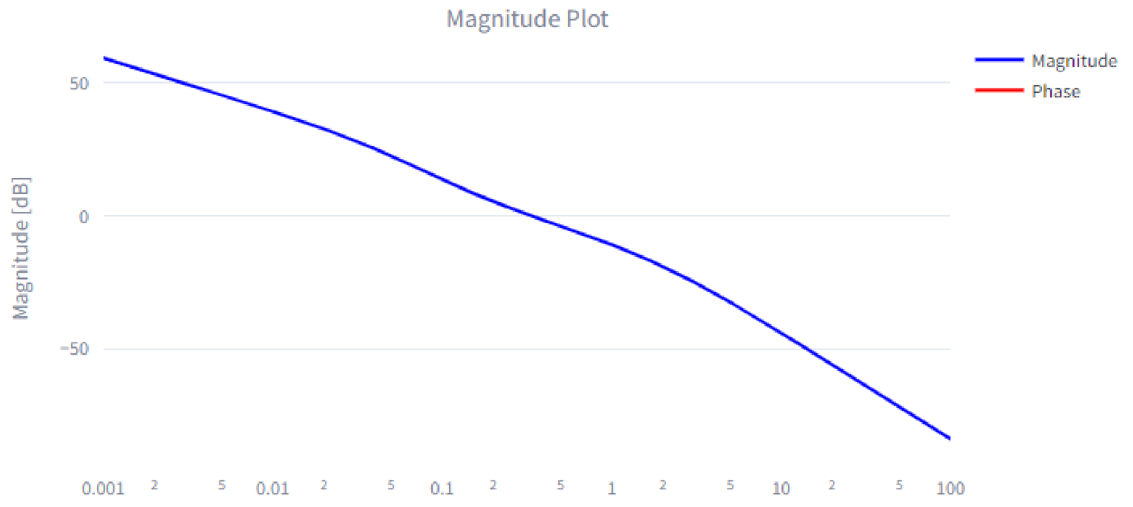
$$y_f(t_k) = (1 - a)y_f(t_{k-1}) + a y_m(t_k)$$

$$a = \frac{T_s}{T_f + T_s}$$

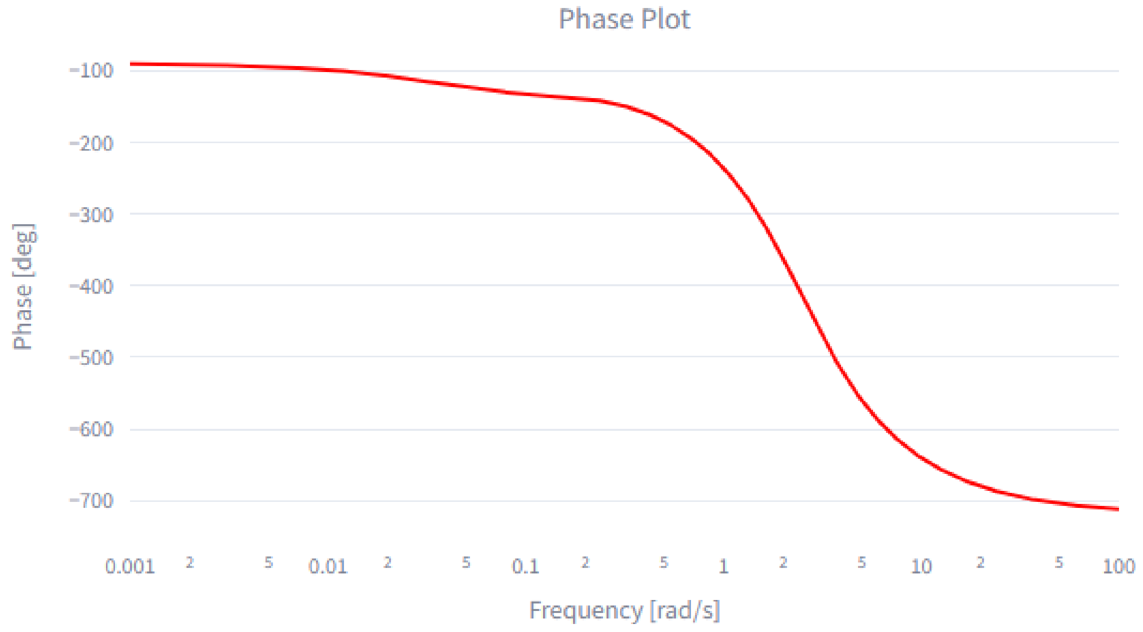
Variables	**Description**
$y_f(t_k)$	Filtered output value at time t_k
$y_m(t_k)$	Measured output value at time t_k
T_s	Samling time [s]
T_f	Filter time constant [s]

Stability and Frequency Analysis

- **Stability Margins**
- Gain Margin: 5.22 dB
- Phase Margin: 28.16 degrees
- Critical Gain: 3.65



- **Crossover Frequencies**
- Gain Crossover: 0.33 rad/s
- Phase Crossover: 0.57 rad/s



Airheater control library

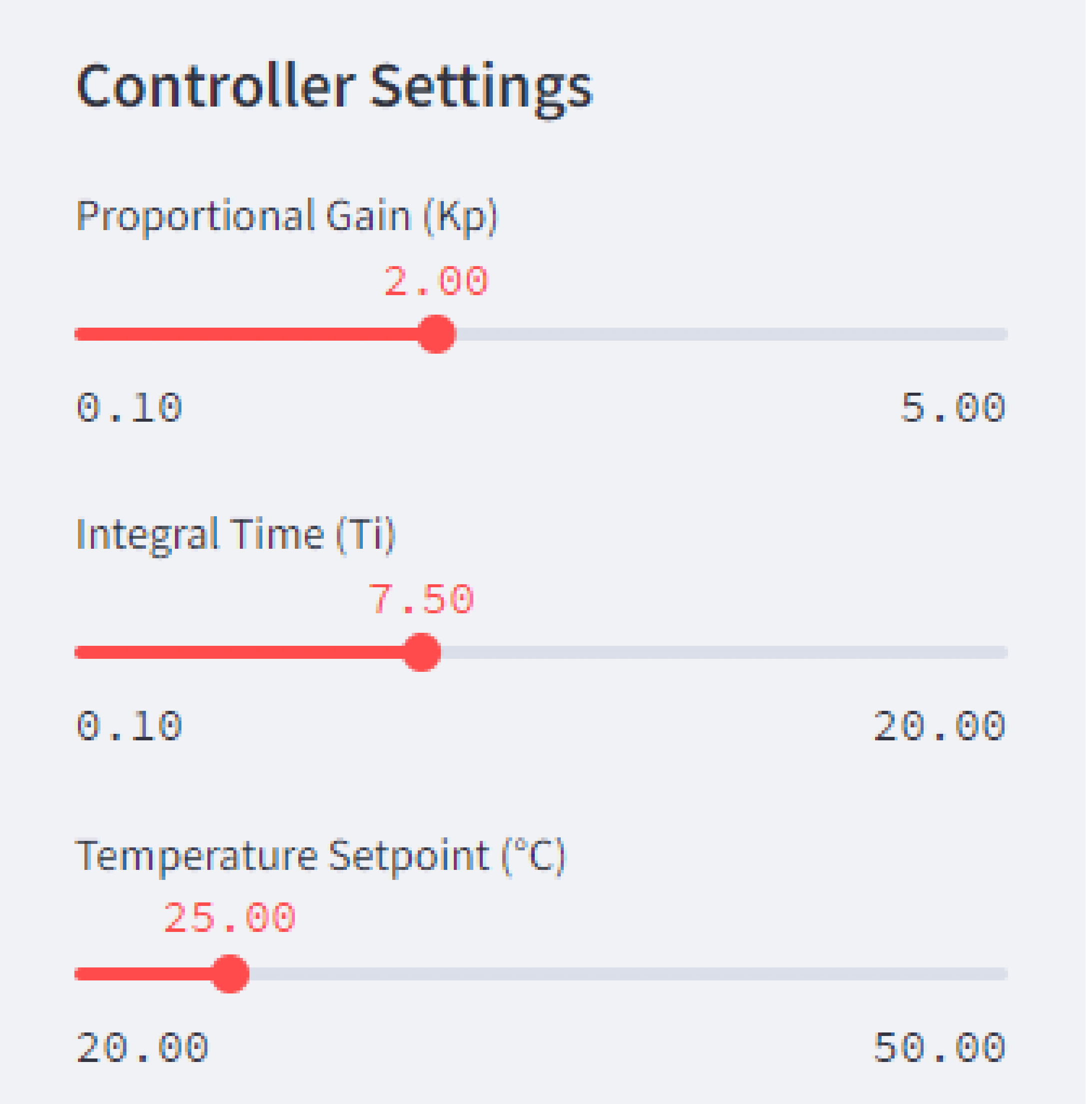
A simple airheater control library was developed to ensure easy reuse of code and reduce maintenance.

The classes and functions are depicted on the right.

```python
'''
    Airheater class

    Implements the airheater model
'''
class Airheater:

    def __init__(self, Kh: float, theta_t: float, theta_d: float, Tenv: float, Tout_init: float, Ts: float):

    def update_heater(self, u: float) -> float:

'''
    LowPassFilter class

    Simple implementation of lowpass filter

    Rule of thumb: Tf >= 5*Ts
'''
class LowPassFilter:

    def __init__(self, y_init: float, Tf: float, Ts: float):

    def update_filter(self, y: float) -> float:

'''
    PI controller class

    Simple impementation of PI controller
'''
class PI_controller:

    def __init__(self, Kp: float, Ti: float, Ts: float, u_min: float, u_max: float):

    def update_pi_controller(self, r: float, y: float) -> float:

'''
    scaling_input function

    Simple function for performing scaling form voltage signal to process variable
'''
def scaling_input(ymax: float, ymin: float, umax: float, umin:float , u:float ) -> float:
```

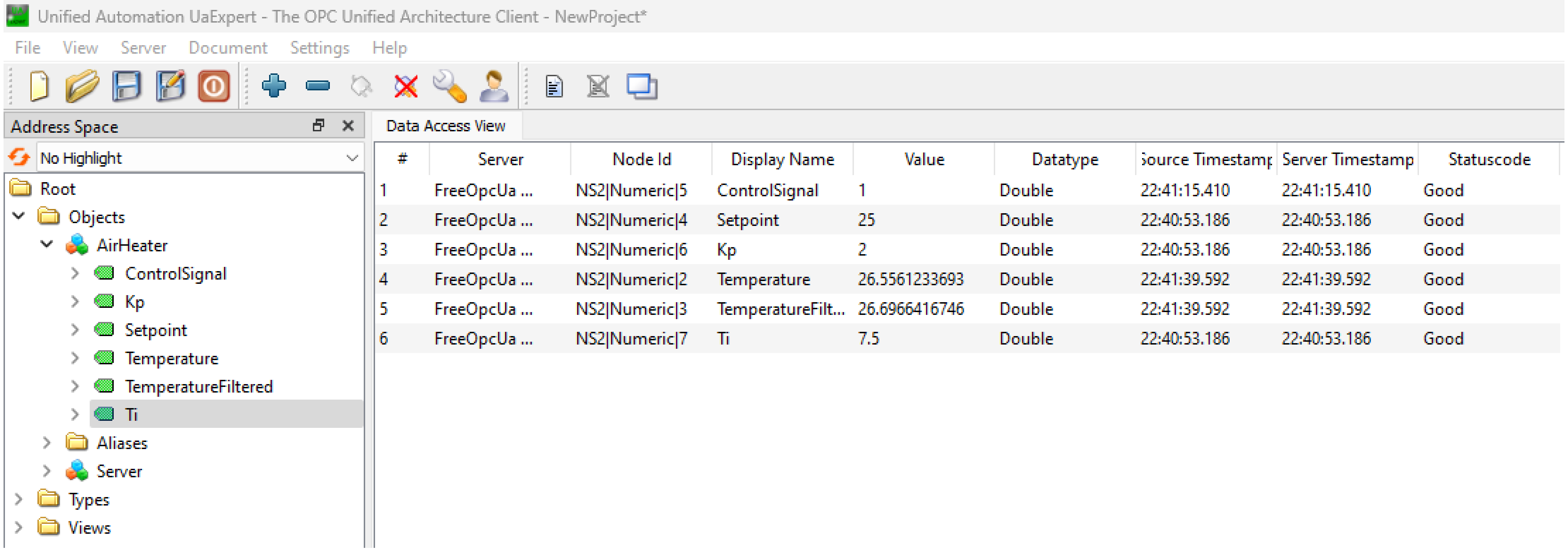
Controller application

- Communicates directly with airheater through DAQ device.
- Make process data available for other devices by sending it to the OPC UA server.
- Can adjust the setpoint in the application.
- Note that the controller parameters and the environmental temperature can be changed in the GUI.
- The GUI is programmed in Python using the Streamlit library.



OPC UA server

The server runs in the background using asyncua library in python.



Database Application

tructure | Browse Data | Edit Pragmas | Execute SQL

Table | Create Index | Modify Table | Delete Table | Print | Refresh

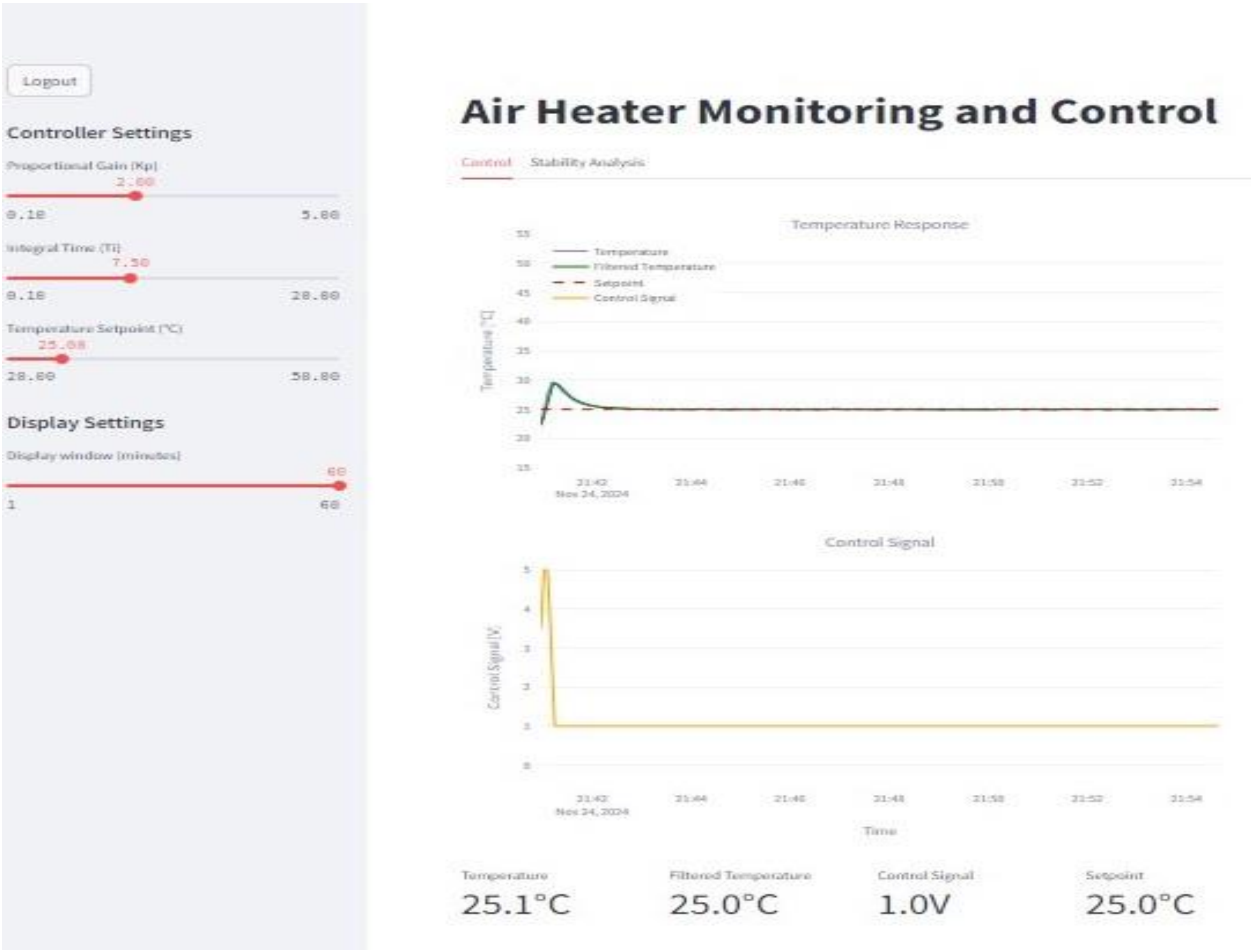
	Type	Schema
les (3)		
measurements		CREATE TABLE measurements (id INTE
id	INTEGER	"id" INTEGER
timestamp	DATETIME	"timestamp" DATETIME DEFAULT CURR
temperature	REAL	"temperature" REAL
temperature_filtered	REAL	"temperature_filtered" REAL
setpoint	REAL	"setpoint" REAL
control_signal	REAL	"control_signal" REAL
kp	REAL	"kp" REAL
ti	REAL	"ti" REAL
sqlite_sequence		CREATE TABLE sqlite_sequence(name,s
users		CREATE TABLE users (id INTEGER PRI
id	INTEGER	"id" INTEGER
username	TEXT	"username" TEXT NOT NULL UNIQUE
password_hash	TEXT	"password_hash" TEXT NOT NULL

Filter in any column

mp	temperature	temperature_filtered	setpoint	control_signal
	Filter	Filter	Filter	Filter
21:40:58	22.8745703654656	22.3403882373775	25.0	3
21:40:59	23.212748346846	22.6311749405337	25.0	4
21:41:00	23.7207072017867	22.9943523609514	25.0	4
21:41:01	24.2795747942477	23.4227598387168	25.0	5
21:41:02	24.7822868582401	23.8759355118912	25.0	5
21:41:03	25.5014350521822	24.4177686919882	25.0	5
21:41:04	26.1731301910966	25.002889191691	25.0	5
21:41:05	26.7117075205726	25.5724953013182	25.0	5
21:41:06	27.2299797994691	26.1249901340352	25.0	5
21:41:07	27.8053351776093	26.6851051485599	25.0	4.78310719223

Airheater website

- Shows process data in two plots (one for temperature data and reference temperature, another for the control signal).
- Also a tab for Stability Analysis
- Controller settings and logout button on the left menu

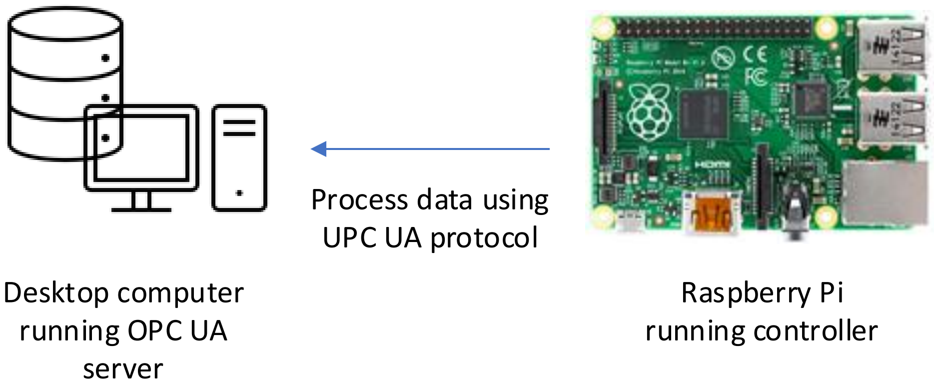


Running control system on Raspberry PI

The controller code was tested on a raspberry PI device.

It successfully communicated process data to the OPC server running on the desktop computer using WiFi.

Raspberry PI could further be extended to communicate directly with airheater using an DAC/ADC-chip[1].



(1) Raspberry Pi only have digital input/output pins.

Demonstration

Discussion

Cybersecurity and safety concerns

Cybersecurity and safety concerns (part 1/2)

- Using a big platform like azure for deploying the webapp greatly enhances the security since Microsoft reputation depends on their ability to keep their services safe from malicious third parties.
- Since databases are used in this system it is important to protect against SQL injection. This is done using parameters.
- OPC UA data should be limited to parties that need access. Implement authentication. If a malicious third-party gain access to the OPC UA server, this can cause great damage.
- Since much of the applications are run on a desktop computer malware is always a concern. Install anti-virus software and configure the firewall properly.

Cybersecurity and safety concerns (part 2/2)

- Keep OS up to date.
- Implement user authentication for access to the website. Two factor-authentication is an even better security measure.
- It is assumed that Microsoft has implemented good protection measures against DDOS[1] attacks.

(1) DDOS: Denial-of-Service (DDOS) attacks

Conclusion

- Process data from a control process (in this case an airheater) can easily be shared with other devices using an OPC UA server.
- Python was shown to be potent tools for control and supervision of the airheater system.
- Important to be aware of different cyber security threats facing these kinds of systems.

Future enhancements

- Advanced control algorithms
- Cloud integration
- Mobile interface
- Advanced analytics
- Predictive maintenance