Dissertation Implementation approach

## Data-Collection: from heat probes

### Option 1: Vendor Data-Logger (No Coding Required)

1. Plug in the probe to the USB/RS-485/Modbus interface box.
2. Install the vendor’s software on your PC.
3. Configure sampling (e.g. 1 Hz), choose a file path, click Start, and you’re collecting data.
4. Export the CSV or JSON when you’re done.

### Option 2: Raspberry Pi + Python (More Flexible)

1. A Raspberry Pi (any model with GPIO pins; Pi 3/4/Zero W are common).
2. A digital temperature probe (e.g. DS18B20 for 1-Wire, or an RTD with an SPI/ADC breakout).
3. A micro-SD card (with Raspberry Pi OS installed).
4. A USB power supply or battery pack.

**Step-by-step**

1. **Set up the Pi**
   * Flash Raspberry Pi OS to your SD card (use the Raspberry Pi Imager).
   * Boot the Pi, connect to your network (Ethernet or Wi-Fi).
2. **Connect your sensor**
   * For a DS18B20 1-Wire probe:
     + Plug its red wire to **5 V**, black to **GND**, yellow (data) to **GPIO 4**.
     + Add a 4.7 kΩ pull-up resistor between 5 V and GPIO 4.
   * For other sensors (Modbus, SPI ADC, etc.), wire per the breakout’s instructions.
3. **Enable the interface**
   * Open a terminal on the Pi and run sudo raspi-config.
   * Under **Interface Options**, enable **1-Wire** (for DS18B20) or **SPI/I2C** as needed.
   * Reboot.
4. Install Python libraries

sudo apt update

sudo apt install python3-pip

pip3 install adafruit-circuitpython-ds18x20 adafruit-circuitpython-pureio (Or, for Modbus: pip3 install minimalmodbus

1. Write a simple Python script
2. Run the script to log the temperature reading into a CSV file
3. **Automate at boot (optional)**

Edit the crontab: crontab -e

@reboot /usr/bin/python3 /home/pi/read\_temp.py & (Pi will start logging whenever it powers on)

## Implementation flow

[Raw CSV]

↓

[1. Load & Preprocess]

↓

[2. Anomaly Detection (Isolation Forest)]

↓

[3. Noise Injection & Masking]

↓

[4. Encryption]

↓

[5. HVAC Control (Decrypt & De-fuzz)]

## Dissertation Roadmap

### Putting It All Together: Dissertation Roadmap

1. **Introduction** – Why thermal side-channels matter.
2. **Background** – What others have done for power and sound side-channels.
3. **Method**
   * Data loading & visualization
   * Spike detection algorithm
   * Noise-injection and smoothing
   * Encryption and key handling
4. **Implementation** – Show your Python (or pseudocode) and how you ran tests.
5. **Results** – Before/after graphs, cooling system logs, standard-compliance table.
6. **Discussion** – Trade-offs: more noise = more privacy but slightly rougher control.
7. **Conclusion** – You’ve built a magic filter that hides door-openings but keeps the room cool.

## Steps that I did

As a sample data, I took data from outside assuming that is the temperature from data center.

Step1 : I loaded that data into CSV file and plotted the graph to see the dips. (png file)

Step 2: Now **automatically spot and timestamp those dips**—i.e. run your anomaly detector on the underlying temperature data that produced the PNG.

Install python – I already have python installed

*pip --version*

***pip 24.3.1*** *from C:\Python313\Lib\site-packages\pip* ***(python 3.13)***