Digital Thermometer (IR based) Simulation

Somantha Manuranga, s.mandalawalli-acharige@oth-aw.de[1]

[1] *OTH Amberg-Weiden, Study programme "Artificial Intelligence for Industrial Applications"*

Report written on June 24, 2025

Abstract

This report shows the simulation of a Digital thermometer (Infrared Thermometer) to measure the body temperature.The main objective of this simulation is the capture the temperature values during a specific period of time with real world disturbances. Here 60 seconds of time was considered.The simulation helps to see how the temperature readings can change slightly each and every time because of environmental errors and noises.Finally, an implementation was done to make the simulation more realistic by taking into account temperature drift, outliers, and spikes.

Keywords: *Digital Thermometer, Infrared radiation, Temperature measurement*

1. Simulation Aspect

The main goal of this simulation is to measure the temperature readings with a digital (IR) thermometer over a 60 seconds of period of time , by considering how real world errors effect the raw readings and demonstrate how signal processing techniques (used signal processing techniques are moving average filter and outlier rejection) [1] to get more accurate and reliable temperature readings.

2. Formula as the Basis of the Simulation

The fundamental physical principle relating the temperature of an object to the thermal radiation it emits is the Stefan-Boltzmann Law. For a perfect blackbody,the power (P) radiated per unit area (A) is given by:[2] [3]

$$\frac{P}{A} = \sigma T^4$$

where:

- $\frac{P}{A}$ is the radiant power emitted per unit area (W/m^2)
- σ is the Stefan–Boltzmann constant ($\approx 5.67 \times 10^{-8}$ W/m^{2}K^4)
- T is the absolute temperature of the object in kelvins (K)

$$S \propto \varepsilon T^4$$

Real objets do not emit their all heat. Emissivity (0-1) shows how they emit compared to a blackbody (which means 100%). In real world device the detected signal is affected by the object's true emissivity and temperature as well as the sensor noise,

$$S_{\text{detected}} = K \times \epsilon_{\text{true}} \times (T_{\text{true}})^4 + \text{Noise}$$

Where K is a constant related to the sensor characteristics.But the thermometer has fixed emissivity , ϵ_{calib} during the calibration and calculates the measured temperature using,

$$T_{\text{measured}} = \left(\frac{S_{\text{detected}}}{K \times \epsilon_{\text{calib}}} \right)^{1/4}$$

The core formula for first step of this simulation is much simpler.
Measured Temperature = Ideal Temperature + Noise Value

3. Simulation Code Environment

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

print("Libraries imported: numpy, matplotlib.
    pyplot, pandas.")

# Simulation parameters
simulation_duration_seconds = 60
measurement_interval_seconds = 0.5
ideal_body_temperature_celsius = 37.0
noise_celsius = 0.15

# Total number of simulated readings
num_readings = int(simulation_duration_seconds /
     measurement_interval_seconds) + 1
```

Code 1. Setting up of Initial Simulation Parameters

```
# Generate evenly spaced time points.
time_points = np.linspace(0,
    simulation_duration_seconds, num_readings)

# Add normally distributed noise to each reading
measurement_noise = np.random.normal(
    loc=0,
    scale=noise_celsius,
    size=num_readings)

simulated_temperatures =
    ideal_body_temperature_celsius +
    measurement_noise

print("Simulated data generation complete.")
print(f"Generated {len(time_points)} time points
     and {len(simulated_temperatures)}
    temperature readings.")

# Plot the simulated temperature readings over
    time
plt.figure(figsize=(14, 7))

plt.plot(time_points, simulated_temperatures,
         marker='o',
         linestyle='-',
         markersize=4,
         label='Simulated Reading')

plt.axhline(y=ideal_body_temperature_celsius,
            color='r',
            linestyle='--',
            label='Ideal Temperature')

plt.title('Simulated Digital IR Thermometer
    Readings Over 60 Seconds')
plt.xlabel('Time (seconds)')
plt.ylabel('Temperature ( C )')
plt.grid(True, which='both', linestyle='--',
    linewidth=0.5)

# Set y-axis limits around the ideal temperature
        3 times noise std dev, with a small
    buffer
y_lower_limit = ideal_body_temperature_celsius -
     3 * noise_celsius - 0.2
y_upper_limit = ideal_body_temperature_celsius +
     3 * noise_celsius + 0.2
plt.ylim(y_lower_limit, y_upper_limit)

plt.legend()
```

```
plt.show()
```

Code 2. Simulating and plotting temperature readings

This code visualizes the Digital IR thermometer readings over 60 seconds. It generates noisy readings by adding rondom noise to a constant ideal temperature. This plot shows that noisy readings relative to the ideal temperature baseline over time.

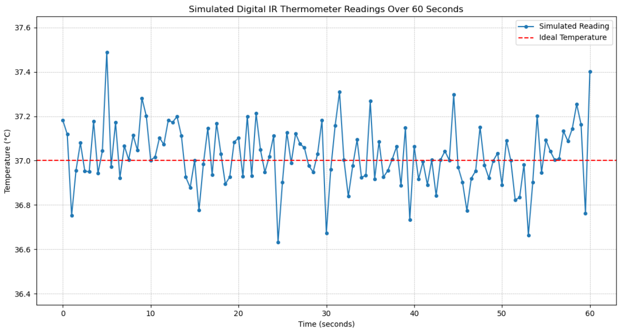

Figure 1. Simulated Digital IR Thermometer Readings Over 60 Seconds

Over a 60-second time period simulation was completed for 121 number of readings.Each blue dot here represents the temperature reading taken at a certain time. Those readings were not constant because of the random noise was added to make it more like real life measurements.At the final step we can observe the average of all readings(Mean Temp. value) , typical spread of the readings around the mean(Standard Deviation) and Min,Max temperature values. Also overall variation can be found from max and min temperature difference.

4. Advanced Simulation with Signal Processing

But in a real scenario, the objects will cool down over time (drift temp,) , emissivity mismatch can happen because surface properties can affect the readings (bias) and sudden spikes (outliers) can happen during these temperature readings also. Based on these reasons we can make a close simulation about this IR thermometers.

```
# Modeling Errors
noise_celsius = 0.15
drift_celsius_per_minute = -0.1
emissivity_offset_celsius = -0.1
outlier_time_seconds = 30
outlier_magnitude_celsius = 1.0

# Signal Processing Parameters
moving_average_window_size = 5
outlier_rejection_threshold_std = 3.0

# Derived Values
num_readings = int(simulation_duration_seconds /
    measurement_interval_seconds) + 1
time_points = np.linspace(0,
    simulation_duration_seconds, num_readings)
total_drift = np.linspace(0,
    drift_celsius_per_minute, num_readings)
measurement_noise = np.random.normal(loc=0,
    scale=noise_celsius, size=num_readings)

# Raw Signal Generation
base_temperature =
    ideal_body_temperature_celsius + total_drift
     + emissivity_offset_celsius
raw_simulated_temperatures = base_temperature +
    measurement_noise

# Insert Outlier
outlier_index = np.argmin(np.abs(time_points -
    outlier_time_seconds))
raw_simulated_temperatures[outlier_index] +=
    outlier_magnitude_celsius

# Applying signal processing
raw_series = pd.Series(
    raw_simulated_temperatures)
# Apply a moving filter
filtered_temperatures = raw_series.rolling(
    window=moving_average_window_size,
    min_periods=1).mean()

# Outlier Rejection
mean_raw = raw_series.mean()
std_raw = raw_series.std()
threshold = outlier_rejection_threshold_std *
    std_raw
is_not_outlier = (raw_series - mean_raw).abs() <
     threshold
cleaned_temperatures = raw_series.where(
    is_not_outlier)
```

Code 3. Advanced simulation setup with noise, drift, bias, and outlier

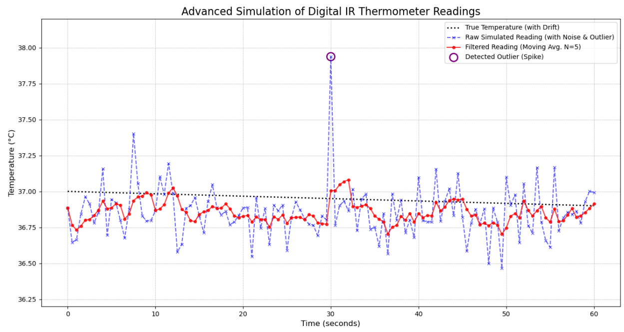

Figure 2. Advanced Simulation Readings in IR thermometer

5. Simulation Results and Conclusion

The core idea behind this simulation was how these thermometer readings are affected by measurement noise, sensor errors , temp.drift , and occasional outliers which lead to inaccurate raw data.Also normally sensors introduce random variations due to environmental factors and internal noises.To overcome these challenges signal processing techniques like moving average filter is very important. These methods helps to minimize the noises and impact of outliers and allow measured temperature to better true temperature values.This graph shows that without filtering, raw measurements can mislead, but with proper processing, we can generate more accurate and reliable temperature readings.

References

[1] D. N. Purnamasari, K. A. Wibisono, and H. Sukri, "Digital moving average filter application for echo signals and temperature", 2021. [Online]. Available: https://www.e3s-conferences.org/articles/e3sconf/abs/2021/104/e3sconf_icstunkhair2021_02007/e3sconf_icstunkhair2021_02007.htmll.

[2] J. Prosper, "Fundamentals of non-contact temperature measurement", *ResearchGate*, 2023, Accessed on June 2025. [Online]. Available: https://www.researchgate.net/publication/389776500.

[3] N. Bin and W. Yuchuan, "Research on non-contact infrared temperature measurement", in *2025 IEEE Conference on Infrared Temperature Measurement Systems*, IEEE, 2025. [Online]. Available: https://ieeexplore.ieee.org/document/5677034.