Task 14: Fire Detection Algorithm v1 – Technical Document

Title: Fire Detection Decision Algorithm v1

1. Pseudocode

```
Input: sensor_data (temperature, smoke, ir_flame, proximity)

Output: fire_alert (True/False)

 A. Preprocess sensor_data:
     - Fill missing keys with default values (0 for numbers, None for proximity)
 B. Calculate normalized scores:
     - temp_score = clamp((temperature – temp_thresh)/temp_thresh, 0, 1)
     - smoke_score = clamp((smoke – smoke_thresh)/smoke_thresh, 0, 1)
     - ir_score = 1 if ir_flame >= ir_thresh else 0
 C. Compute weighted global score:
     - global_score = temp_score * weight_temp + smoke_score * weight_smoke + ir_score * weight_ir
 D. Compare global_score with alert_threshold:
     - if global_score >= alert_thresh:
           Fire_alert = True
       else:
           Fire_alert = False
 E. Return fire_alert
```

2. Example Scenarios

temp	smoke	ir_flame	temp_score	smoke_score	ir_score	global_score	fire_alert
55°C	350	1	0.1	0.1667	1	0.68	True
30°C	100	0	0	0	0	0.0	False

Explanation:

- Global score combines each metric with its weight.
- Thresholds were tuned v1 experiments for balanced detection.

3. Chosen Thresholds and Weights

Parameter	**Value**	**Justification**
Temperature threshold	50°C	Fires usually raise temp above 50°C in small rooms.
Smoke threshold	300 ppm	Average background smoke is < 250 ppm.
IR flame threshold	1	Detects actual flame presence.
Weight Temp	0.4	Balanced influence in global score.
Weight Smoke	0.4	Smoke is as important as temperature.
Weight IR	0.2	Flame detection is secondary but critical.
Alert threshold	0.5	Midpoint ensures early detection but limits false positives.

4. Notes and Future Improvements

- This is v1, based on testing with simulated data.
- Future enhancements:

- Use **sensor fusion** techniques (Kalman filter, moving average).

- Add **time-based smoothing** to reduce false positives.

- Fine-tune thresholds for specific environments