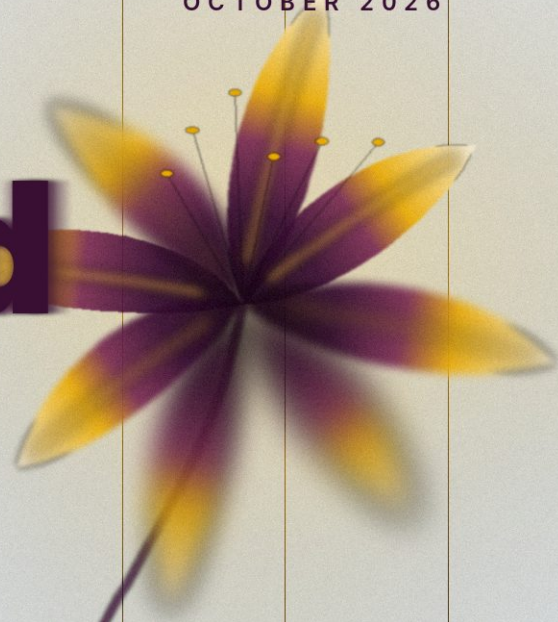


DairyGuard Kinetic.



Architecture of an open cold-chain spoilage monitor: data, models, estimator, evaluation and deployment

Suryansh Pratap Singh

Department of Mathematics and Computing, Shri Mata Vaishno Devi University, India

WHAT THIS DOCUMENT IS

A white paper explains how a system is built and why each part is the way it is. This one describes DairyGuard Kinetic as it exists and has been tested: where its data came from, which models it uses and how they were fitted and trained, the algorithm that turns a temperature log into a verdict, how it was evaluated, and how it is packaged as Python software, a command-line tool, microcontroller firmware and a browser demo. Every component named here is in the public repository, with tests. The results themselves are reported in the book *Beyond the 4 °C Alarm* [1]; this paper is the map of the machinery behind them.

1 The problem the system solves

A cold-chain logger records the temperature of a product as it travels. The question asked of it is whether the product is still within a microbiological limit, for milk the FSSAI aerobic plate count of 3×10^4 CFU mL⁻¹ [3]. Five requirements follow from where the system must run and who must trust it.

1. **Use only what a logger records:** a temperature every few minutes, from a cheap probe.
2. **Run on a microcontroller:** a few kilobytes of memory, no network required.
3. **Say how sure it is:** a probability that the limit has been crossed, not only a red or green light.
4. **Be checkable:** every parameter traced to a source, every result reproducible from code.
5. **Be honest about its limit:** report how much of the error comes from the biology a log cannot see.

2 System overview

DairyGuard Kinetic has five layers (Fig. 1). A data layer holds the growth records and the parameter file. A model layer turns temperature into a growth rate and a growth rate into shelf life. The estimator, the monitor, accumulates shelf life along a log and turns it into a probability and an estimate of hours left. An evaluation harness, a simulator in which the truth is known, measures how well the monitor works. Three

deployment targets run the same estimator: a Python package with a command-line tool, C firmware for an ESP32, and a browser demo.

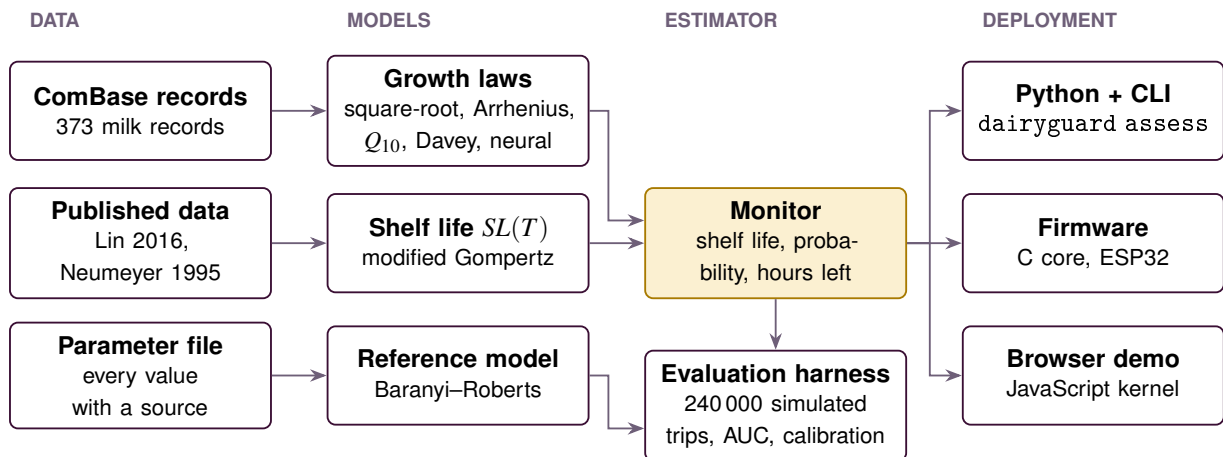


Figure 1 The architecture of DairyGuard Kinetic. Every box is a module in the repository with its own tests

3 Data layer

3.1 How the growth data was collected

ComBase is the public database of microbial responses to food environments, assembled from published studies and laboratory data [4, 5]. A search for growth records with milk as the food matrix returned records for spoilage flora, *Listeria*, *Bacillus cereus*, *Escherichia coli*, *Salmonella* and others. For each record we transcribed the organism, the food, temperature, water activity, pH, the conditions, the source study and the maximum growth rate, and marked whether a rate was stated, could only be obtained by fitting the curve, had not been transcribed, or showed no growth. This gave 373 records (data/combase_milk_records.csv), of which 179 have a stated rate. A separate file holds the 51 records of natural spoilage flora in pasteurised milk measured at the Tasmanian Institute of Agriculture, 2.7–14.9 °C (data/combase_tasmania_pasteurised_milk.csv).

3.2 Units

ComBase reports rates in decimal logarithms, $\log_{10}$ CFU mL⁻¹ per hour. The models use the natural-log rate, so every rate is multiplied by $\ln 10$ before fitting, $\mu = \ln(10) \mu_{10}$. A test checks that this conversion is applied (tests/test_units_and_headlines.py).

3.3 Parameter provenance

Every parameter of every model is a row of data/params.csv with its value, unit and source, marked SOURCED (from a publication or ComBase) or ASSUMED (a modelling choice to be measured). The assumed values are the between-strain spread, the range of lag and starting count, and the frequency of warm events on a trip.

4 Model layer

4.1 Growth rate against temperature

The default law is the square-root law of Ratkowsky et al. [6],

$$\sqrt{\mu(T)} = b(T - T_{\min}), \quad (1)$$

with $b = 0.030 \text{ h}^{-1/2} \text{ K}^{-1}$ and $T_{\min} = -7.7^\circ \text{C}$ for psychrotrophic spoilage flora in milk [7]. Four alternatives are implemented in `dairyguard.growthlaws` with one interface (`fit`, `predict`): the Arrhenius law, a constant- Q_{10} law, the Davey law, and a monotone neural network (Section 5).

4.2 Reference model

The Baranyi–Roberts model [8] describes the count N along any temperature history through $y = \ln N$ and a physiological state q :

$$\frac{dy}{dt} = \mu(T) \frac{q}{1+q} (1 - e^{y-y_{\max}}), \quad \frac{dq}{dt} = \mu(T) q. \quad (2)$$

It is integrated minute by minute and serves as the truth in the evaluation harness.

4.3 Shelf life and the estimator

The estimator uses a closed form that a microcontroller can evaluate: the modified-Gompertz time to reach the limit N_L at a constant temperature,

$$SL(T) = \frac{h_0}{\mu} + \frac{A \ln 10}{\mu e} \left[1 - \ln \left(-\ln \frac{L}{A} \right) \right], \quad A = \log_{10} \frac{N_{\max}}{N_0}, \quad L = \log_{10} \frac{N_L}{N_0}, \quad (3)$$

and adds the fraction used by each reading,

$$\text{CSL} = \sum_i \frac{\Delta t_i}{SL(T_i)}. \quad (4)$$

The reference and the estimator are deliberately different models, so the evaluation does not mark the estimator against itself.

4.4 Why the design works: the generalized time

Dividing Eq. (2) by μ removes temperature: with $\tau(t) = \int_0^t \mu(T(s)) ds$, both y and q are functions of τ alone, so the limit is reached exactly when τ reaches a value set by the biology. Because $1/SL(T) = \mu(T)/K$ with K constant, CSL is a discretised τ/K . This is the analogue of Ozawa’s generalized time in thermal analysis [9, 10], and it is why the architecture can summarise any log by one running sum.

5 Fitting and training

- **Classical laws.** Fitted by least squares on $\ln \mu$, because the error of a growth rate grows with its size. Bootstrap intervals give the uncertainty of the parameters.
- **Monotone neural law.** $\sqrt{\mu} = \sum_{k=1}^4 e^{u_k} \text{softplus}(e^{v_k}(T - c_k)/10)$. Positive weights on both layers make it increasing in temperature by construction, so it cannot learn a physically impossible shape. It is trained by least squares on $\ln \mu$ with an L2 penalty on (u, v) , five times from independent random starts; the five fits are averaged, and their spread estimates the model’s own uncertainty.
- **Validation protocol.** A law is judged on data it was not fitted to. For *Listeria*, whose records come from many studies, each study is left out in turn; for the single-source spoilage flora, five-fold cross-validation over records is used. Scores are the bias and accuracy factors of Ross [11].
- **Result.** On held-out studies all five laws fall within a narrow band (accuracy factor 1.37–1.45 for *Listeria*, 1.13–1.15 for spoilage flora), and growth rates differ between studies with a coefficient of variation of 0.25 [1].

6 The monitor

The monitor (`dairyguard.monitor.ColdChainMonitor`) holds the consumed shelf life of a nominal batch and of K plausible batches drawn once from the product's between-batch spread: b lognormal with a coefficient of variation of 10 %, $T_{\min}$ normal with a standard deviation of 1.5 K, lag and starting count uniform over their ranges.

ALGORITHM: ONE READING

```

input temperature  $T$ , time since the last reading  $\Delta t$ 
 $CSL_0 \leftarrow CSL_0 + \Delta t / SL(T; \theta_0)$  nominal batch
for  $k = 1 \dots K$ :  $CSL_k \leftarrow CSL_k + \Delta t / SL(T; \theta_k)$  plausible batches
 $P \leftarrow \frac{1}{K} \# \{k : CSL_k \geq 1\}$  probability the limit is crossed
 $h_k \leftarrow \max(1 - CSL_k, 0) SL(4^\circ\text{C}; \theta_k)$  hours left at 4 °C
output  $CSL_0$ ,  $P$ , median of  $h_k$  (typical), 10th percentile of  $h_k$  (cautious), and a state:
fresh ( $P < 0.15$ ), use soon ( $< 0.40$ ), check before use ( $< 0.75$ ), likely over the limit

```

Each reading costs $K + 1$ evaluations of Eq. (3). Calibration stabilises from $K = 80$ (calibration error 0.014 against 0.160 at $K = 10$), so the Python monitor uses 80 to 200 and the firmware 64.

7 Evaluation harness

The harness (`dairyguard.simulate`, `dairyguard.benchmark`) generates trips in which the truth is known: a fridge at 2.5–5 °C, random warm events of 10 min to 8 h reaching 12–30.7 °C, a product that follows the air with a time constant of minutes (pouch) or hours (crate), and hidden biology drawn per trip. The reference model decides whether each trip crossed the limit. The monitor sees only sensed readings (DS18B20, PT1000 or ideal) at 1, 15 or 60 min, or on an adaptive schedule. Five rules are scored by AUC; differences between rules are tested by a paired bootstrap on identical trips; probabilities are scored by Brier score and expected calibration error. Removing information step by step, from an oracle that knows each trip's biology to a cheap probe read hourly, gives the loss budget. Five parameter sets of 12 settings and 4 000 trips each give 240 000 trips; every random number comes from a fixed seed, and a full rerun reproduces the saved results exactly.

8 Software and deployment

Table 1 The repository

Path	Responsibility
<code>src/dairyguard/kinetics.py</code>	Supply root law, Gompertz shelf life, look-up tables
<code>src/dairyguard/growth.py</code>	Supply growth laws, held-out validation, between-study variation
<code>src/dairyguard/products.py</code>	Supply parameter sets for milk, ComBase-fitted milk flora and <i>Listeria</i>
<code>src/dairyguard/monitor.py</code>	The monitor: shelf life used, probability, hours left, state
<code>src/dairyguard/simulate.py</code> , <code>benchmark.py</code>	The evaluation harness and its statistics
<code>src/dairyguard/cli.py</code>	Command line and log reading
<code>io.py</code>	
<code>experiments/</code>	Every study in the book, one script each
<code>firmware/</code>	C core of the monitor and the ESP32 sketch
<code>demo/</code>	The browser demo and its JavaScript kernel
<code>generated/</code> , <code>report/</code> , <code>whitepaper/</code>	Scripts that write every number, table and figure; the book and this paper
<code>tests/</code>	27 automated tests, run on every push

Python. `pip install -e .` installs the package and the `dairyguard` command. `dairyguard assess trip.csv` reads any log with a time and a temperature column and prints the verdict; `-json` gives machine-readable output for a server.

Firmware. The C core (`firmware/dairyguard_monitor/dg_core.c`) keeps 64 plausible batches in under 2 kB of memory with no dynamic allocation. A test compiles it on a desktop and checks it against the Python monitor at every reading to a relative error below 2×10^{-4} . The ESP32 sketch reads a DS18B20 every 15 min, drives a three-colour light and reports over serial.

Browser. The demo runs a JavaScript copy of the kernel, checked against Python by a parity test, so anyone can try the monitor without installing anything.

9 Limits of this architecture, and what comes next

The architecture does what a log allows, and the evaluation shows where that stops. The monitor's probability is as well calibrated as a logistic model trained on 4 000 labelled trips (calibration error 0.026 against 0.026), but its estimate of the hours left is imprecise (median error 16 h against a median of 22 h), because the time a batch can afford depends on biology the log does not record. Real growth rates differ between studies by 0.25 in the square-root slope, and a flexible neural law fitted to growth rates alone does not predict unseen studies better than the classical laws. The constraint is therefore the data, not the model: the next step needs complete growth curves of real milk under changing temperatures, each paired with simple measurements of the batch at loading.

DairyGuard's next model is being designed for that data. It keeps the generalized-time core of Section 4.4 and the monitor's interface of Section 6, and replaces the fixed between-batch spread with an estimate of each batch's biology learned from those measurements. Its blueprint, with the data plan and the tests it must pass fixed before any data are collected, is White Paper WP-02 [2].

Acknowledgements

The author thanks the ComBase consortium and the laboratories whose published growth records made this work possible.

Note on tools

Parts of the code were developed with an AI coding assistant; every result is produced by tested code that the author has checked.

References

- [1] Singh SP. Beyond the 4 °C alarm: what a temperature log can and cannot tell about milk spoilage. Katra: the author; 2026. <https://github.com/CopyNinja07-07/dairyguard>
- [2] Singh SP. DairyGuard Batch: blueprint of a batch-aware spoilage model. DairyGuard White Paper WP-02; 2026. <https://github.com/CopyNinja07-07/dairyguard>
- [3] Food Safety and Standards Authority of India. Appendix B: microbiological requirements, version 2 (4 November 2024), Tables 2A and 2B. [https://fssai.gov.in/upload/uploadfiles/files/Appendix%20B\(1\).pdf](https://fssai.gov.in/upload/uploadfiles/files/Appendix%20B(1).pdf)
- [4] Baranyi J, Tamplin ML. ComBase: a common database on microbial responses to food environments. J Food Prot. 2004;67:1967–71. <https://doi.org/10.4315/0362-028X-67.9.1967>
- [5] ComBase. Database of microbial responses to food environments. <https://www.combase.cc>
- [6] Ratkowsky DA, Olley J, McMeekin TA, Ball A. Relationship between temperature and growth rate of bacterial cultures. J Bacteriol. 1982;149:1–5. <https://doi.org/10.1128/jb.149.1.1-5.1982>

- [7] Neumeyer K. Modelling pseudomonad growth in milk and milk-based products. Research Master thesis. Hobart: University of Tasmania; 1995. <https://eprints.utas.edu.au/21013/>
- [8] Baranyi J, Roberts TA. A dynamic approach to predicting bacterial growth in food. *Int J Food Microbiol.* 1994;23:277–94. [https://doi.org/10.1016/0168-1605\(94\)90157-0](https://doi.org/10.1016/0168-1605(94)90157-0)
- [9] Ozawa T. Non-isothermal kinetics and generalized time. *Thermochim Acta.* 1986;100:109–18. [https://doi.org/10.1016/0040-6031\(86\)87053-8](https://doi.org/10.1016/0040-6031(86)87053-8)
- [10] Málek J, Koga N, Pérez-Maqueda LA, Criado JM. The Ozawa's generalized time concept and YZ-master plots as a convenient tool for kinetic analysis of complex processes. *J Therm Anal Calorim.* 2013;113:1437–46. <https://doi.org/10.1007/s10973-013-2939-0>
- [11] Ross T. Indices for performance evaluation of predictive models in food microbiology. *J Appl Bacteriol.* 1996;81:501–8. <https://doi.org/10.1111/j.1365-2672.1996.tb03539.x>