International Journal
Information Technology Applications (ITA)

Volume 15, Number 2, August 2026

MATHEMATICAL AND DIGITAL METHODS USED IN THE GARDENER FIRMWARE

Jozef Dubovec

Abstract:

The article describes the mathematical and digital methods used in the Gardener firmware from an embedded-system point of view. The goal is not to present a new mathematical theory, but to define how measured values, bitmasks, resources, normalized sensor data, recipe deviations, consumption, carbon footprint and economic indicators are represented in a reproducible way. The method combines deterministic integer encoding, binary floating-point interpretation, weighted aggregation, threshold comparison, time-based accumulation and basic statistical evaluation. The result is a firmware-oriented model that can store heterogeneous sensor and process data in the same digital format while keeping a clear path back to engineering units and business indicators.

Keywords:

Embedded firmware, data normalization, bitmask, controlled environment agriculture, Gardener, statistical evaluation.

ACM Computing Classification System:

Computer systems organization - Embedded systems;
Software and its engineering - Software notations and tools;
Information systems - Database design and models; Applied computing - Agriculture [8].

Introduction

Gardener is designed as an edge-oriented automation system for controlled environment agriculture and related technical processes. Its practical role is similar to a small, affordable SCADA-like layer placed close to the real process. It observes sensor values, keeps the meaning of resources stable, compares the real state with a recipe or plan, and gives the user a clear operational view. In this article the term SCADA480 is used as a compact interface and data-control concept for small portrait displays and small automation installations, rather than as a separate industrial SCADA product.

The motivation for Gardener is practical. Many small farms, schools, laboratories, community growers and small technical installations need automation, logging and decision support, but classical industrial SCADA solutions are often too expensive, too complex, or too dependent on specialised engineering services. A low-cost edge system can make automation more reachable. It cannot replace a full industrial platform in all cases, but it can provide a structured path from a simple sensor board to reproducible data, local control, dashboard presentation and later scientific or economic evaluation.

Affordability is not only a matter of hardware price. It also depends on whether the system can be configured, repaired, expanded and understood by the user. For that reason, the mathematical layer of Gardener must be transparent.

A temperature value, a pump state, a plant stage, an alarm, an energy counter and a return-on-investment indicator must be stored and interpreted by clear rules. If the rule is explicit, the same value can be used by firmware, database, web dashboard and publication-style reporting without changing its meaning.

The firmware point of view is important because the first interpretation of a value usually happens at the edge. A sensor driver may produce a floating-point number, a binary input may produce a single bit, a recipe may define a target range, and a user interface may require a normalized percentage. These representations are different, but they must be connected. Gardener therefore treats conversion, normalization, bitmasks, resource identifiers, aggregation, statistics and economic indicators as parts of one digital method, not as unrelated programming details.

The main contribution of this article is a compact description of the methods used to keep this stability. This article focuses on bitmasks, resource identification, numeric conversion, normalized value computation, statistical aggregation, recipe comparison, energy and consumption calculation, and economic indicators such as unit production cost, return on investment and payback. This document is intentionally general. It gives the mathematical logic and firmware interpretation, but it does not depend on one exact board, one exact display size, or one exact crop.

The purpose is not to introduce a new mathematical theory, but to describe how known and established methods are combined in an embedded agriculture-automation context.

1 State of the Art and Firmware Scope

1.1 General role of Gardener and SCADA480

Gardener can be described as a practical edge automation layer for smaller controlled processes. It is intended to connect sensors, actuators, recipes, data storage and user interaction in a form that remains understandable for the operator. In this sense, it follows the general idea of supervisory control and data acquisition, but it is focused on smaller installations where affordability, modularity and local decision making are more important than large industrial scale.

The SCADA480 idea represents a compact supervisory interface suitable for portrait displays such as 480 x 800 pixels. It is not defined here by one graphical layout only. It is a functional concept: the same values that are processed by firmware must also be visible to the operator as stable states, warnings, recipe matches and indicators. A small display can therefore become a meaningful local control point when the data model behind it is consistent.

This general role explains why mathematical representation is important. If the system is to be affordable and widely deployable, it cannot depend on manual interpretation of every sensor or every board variant. The firmware must transform different input forms into repeatable internal forms, so that an inexpensive device can still produce reliable logs, useful alarms and comparable reports.

1.2 Embedded representation of measured values

Embedded systems usually combine integer logic, floating-point sensor values and compact status flags. In a small automation controller this is not only a programming detail. It is also a data model problem. A temperature value, a pump state, a warning flag and a crop-stage mask may have different meanings, but they must travel through the same communication and database pipeline without ambiguity.

For this reason, Gardener separates the semantic identity of a value from the physical sensor that produced it. The semantic identity is represented by resource identifiers and domain-element-structures, while the physical origin is represented by board, hardware and origin information. This makes the firmware less dependent on one exact sensor and allows the same mathematical processing to be reused in agriculture, water treatment, laboratory or other application areas.

1.3 Relation to existing standards and methods

The numerical part of the model is related to established computing and engineering practice. Floating-point values follow the idea of binary floating-point representation, as standardized by IEEE 754 [1]. The firmware may also store a float as the same 32-bit payload that was used in memory, when exact reconstruction is required. Integer masks follow common embedded C practice, where one integer value can represent many Boolean states [4]. For environmental and economic evaluation, the model is compatible with product-level greenhouse-gas accounting and life-cycle assessment principles [2], [3], [9].

2 Mathematical and Digital Model

2.1 Bitmasks and discrete states

A bitmask is used when several independent states must be stored in one compact integer. Each bit represents one property, stage, permission, equipment role or condition. If the bit is zero, the property is absent. If the bit is one, the property is active. Multiple properties may be active at the same time. In firmware this is useful for lifecycle stages, lighting classes, board roles, resource groups, user permissions and status flags [4].

$$\text{mask} = \Sigma\, bi \cdot 2\text{^}i, \text{ where } bi \in \{0,1\} \quad (1)$$

Formula (1) describes the general meaning of a bitmask. The resulting integer can be stored in a database, transferred by message and decoded later without losing the individual flags. This method is deterministic, compact and suitable for microcontroller firmware.

2.2 Resource identity and normalized addressing

Gardener uses a compact resource identity to describe what a value means. In the agricultural configuration, this can be understood as a combination of domain, element and hardware identifier. The domain expresses a high-level group such as aeration, watering, lighting or medium. The element expresses the measured or controlled quantity inside that domain, such as temperature, humidity, EC or pH. The hardware identifier links the value to the concrete sensor or component family.

$$\text{resource\_id} = \text{domain\_id} \cdot 8192 + \text{element\_id} \cdot 256 + \text{hardware\_id} \quad (2)$$

Formula (2) is the normalized address formula used to create a stable 32-bit style identifier from smaller fields. The exact interpretation of domain and element may depend on the application area, but the computation keeps the same structure. This allows the database and communication layer to store one resource value while the configuration layer explains its human meaning.

2.3 Digital representation of numeric values

The firmware receives values in different numeric forms. Some values are Boolean, some are integers, some are decimal engineering values and some are floats from sensor drivers. Gardener stores or transports these values through a common digital payload when possible. For integers, the conversion to an unsigned 32-bit value is direct if the original range is known. For signed integers, the original cast type must be stored together with the payload so the receiver knows whether the number is unsigned or signed.

$$u32 = \text{reinterpret\_uint32}(\text{float32\_value}) \quad (3)$$

Formula (3) describes the case where a 32-bit floating-point number is not rounded into an integer, but its memory representation is preserved as a 32-bit payload. This method is useful when the exact original float must be reconstructed later.

This method is different from ordinary numeric conversion. Numeric conversion changes the mathematical value into an integer. Reinterpretation keeps the same bits and requires the cast type to be known during decoding [1].

$$value = stored\_integer / scale \quad (4)$$

Formula (4) describes fixed-scale decimal storage. For example, if a pH value is stored with scale 100, the database may store 615 and the reconstructed engineering value is 6.15. This method is deterministic, easy to compare, and often safer for long-term database reproducibility than repeatedly formatting decimal strings.

2.4 Normalized aggregation and recipe comparison

When several sensors measure the same resource, Gardener can calculate one normalized value for the section or system. The simple method is an arithmetic average. The more useful firmware method is weighted aggregation, because different sensors can have different reliability, placement or priority. The statistical background of this type of aggregation and variability description is consistent with common engineering statistics [5], [6].

$$\bar{x} = (w1x1 + w2x2 + ... + wnxn) / (w1 + w2 + ... + wn) \quad (5)$$

Formula (5) represents weighted aggregation. The result can be stored as the current normalized value of one domain-element slot in the measurement matrix. A weight can represent sensor confidence, calibration quality or expected importance. If one sensor is missing, the computation can continue with the remaining valid sources.

$$deviation = actual\_value - target\_value \quad (6)$$

$$match = 100 - normalized\_penalty(deviation, tolerance) \quad (7)$$

Formulas (6) and (7) describe the general recipe-comparison method. A recipe contains the expected target or range for a stage of the growing cycle. The firmware compares the actual normalized value to this target. The deviation can then be converted into a percentage match. The exact penalty function may be linear, threshold-based or stage-specific. The important point is that the computation is explicit and repeatable.

3 Statistical, Consumption and Economic Evaluation

3.1 Statistical description of cycles

Gardener cycle evaluation uses basic descriptive statistics. The purpose is to compare runs, sections or devices without hiding the variability of biological processes. For measured output such as fresh biomass, dry biomass, energy consumption or water use, the mean gives the central value. Standard deviation describes how much the individual cycles differ from that mean. Coefficient of variation may be used when comparison between different units or crop types is needed [5], [6].

$$mean = (x1 + x2 + ... + xn) / n \quad (8)$$

$$s = sqrt(\Sigma(xi - mean)^2 / (n - 1)) \quad (9)$$

Formulas (8) and (9) describe mean and sample standard deviation. These are not complicated methods, but they are important because they make reports reproducible. When a publication states a mean yield or a mean electricity consumption, the reader should also understand the variation between cycles or units.

3.2 Energy, water and carbon footprint model

Consumption values are accumulated over time. For electrical energy, the basic quantity is power multiplied by time. In firmware, power may come from a meter, a driver state estimate, or an external measurement. Energy consumption can be accumulated per device, per section, per crop cycle or per kilogram of fresh biomass.

$$E = P \cdot t \tag{10}$$

$$CO2e = E \cdot emission\_factor \tag{11}$$

Formula (10) gives the basic energy relation. Formula (11) converts energy to carbon dioxide equivalent using an emission factor. This is compatible with the general logic of product carbon footprint accounting, where the system boundary and emission factor must be declared [2], [3], [9]. Water, nutrient and material consumption can be treated in a similar way: total use is accumulated for a cycle and then divided by the functional output, such as kilograms of edible fresh biomass.

3.3 Economic model

The economic model connects operating costs, amortized device cost and production output. Unit production cost is the total cycle cost divided by usable yield. Return on investment compares the gain or saving with the investment. Payback period expresses how long it takes until cumulative savings compensate the initial investment. In firmware or edge reporting, these formulas do not replace accounting, but they create a consistent technical estimate [7].

$$UPC = total\_cycle\_cost / edible\_yield \tag{12}$$

$$ROI = (gain - investment) / investment \tag{13}$$

$$payback\_period = investment / annual\_net\_saving \tag{14}$$

Formulas (12), (13) and (14) give the basic economic relations. In practice, the result depends strongly on electricity cost, yield, equipment lifetime, number of cycles per year, market price of the crop and waste rate. Therefore Gardener should store not only final indicators, but also the assumptions used to compute them.

4 Discussion

The mathematical model described in this article is intentionally simple. This is an advantage in firmware. A deterministic conversion rule is easier to test than an implicit one. A bitmask is easier to decode than a long text status. A weighted average is easier to explain than a black-box model. For a distributed edge system such as Gardener, transparency is often more valuable than mathematical complexity.

This simplicity is also important from the point of view of affordability. A small grower or a school installation should not need an industrial engineering team to understand why a value changed, why a warning was generated, or why a recipe match decreased. When a system can express its reasoning through clear values, thresholds, deviations and accumulated counters, it becomes more suitable for education, maintenance and gradual improvement. Gardener is therefore not only a controller, but also a method for making technical and biological processes more readable.

The SCADA480 concept is useful in this context because it connects compact visualization with firmware semantics. A small display cannot show everything, but it can show the most important normalized states. The dashboard percentage, section state rings, warnings, plant layout indicators and system health values become meaningful only when the underlying data are stable.

The visual layer should not invent meaning; it should expose the meaning already present in the resource model, recipe model and measurement matrix.

The main risk is that compact digital storage can become ambiguous if metadata are not stored together with the value. A 32-bit payload is not enough by itself. The receiver must know whether the payload means a Boolean value, signed integer, unsigned integer, fixed-scale decimal or raw floating-point representation. Therefore the cast type, unit, scale and semantic resource identity are part of the model, not optional decoration.

Another risk is overprecision. Sensor values, biological yield and economic calculations can look exact when written with many decimal places, but the real process contains uncertainty. For this reason, reports should use appropriate rounding and should include standard deviation or another variability indicator when cycle data are compared. The firmware can support this by storing raw values, normalized values and calculation assumptions.

From an engineering point of view, the strongest part of the approach is reproducibility across layers. The same measured value can be used by the embedded firmware for local decision making, by the database for history, by the web application for visualization, and by a publication or business report for later evaluation. This reduces the risk that every layer creates its own version of the truth. It also makes the system easier to audit, because the path from measurement to indicator can be followed step by step.

From an agricultural point of view, the approach is important because biological systems are variable. Two cycles may be operated with the same recipe and still produce different results. A firmware model that stores assumptions, target ranges, deviations, consumption and cycle outcomes allows this variability to be studied rather than ignored. The user can compare recipes, sections and devices over time and decide whether a difference is caused by environment, hardware, crop stage, human action or natural variation.

The advantage of the Gardener approach is that the same digital logic can support real-time control, database storage, dashboards and later scientific evaluation. The method is therefore useful not only for one growing unit, but also for comparing devices, recipes, crops and operating conditions over time. This is where affordability and scientific structure meet: a low-cost system becomes more valuable when it produces data that are consistent enough to support technical learning and practical decisions.

Conclusion

The article presented a simplified scientific description of mathematical and digital methods used in the Gardener firmware. The methods include bitmask representation, resource identity computation, deterministic conversion of numeric values, fixed-scale decimal storage, raw float reinterpretation, weighted aggregation, recipe deviation analysis, statistical cycle description, consumption calculation, carbon footprint estimation and economic indicators.

The central principle is reproducibility. A value should be stored in a compact form, but it must remain possible to recover its meaning and engineering unit. A computed indicator should be useful for the user, but the assumptions behind the computation must remain available. With this structure, Gardener can connect embedded firmware, database storage, cloud dashboards and publication-style reporting without changing the meaning of the data.

The general importance of Gardener is in making structured automation more accessible. When the same mathematical logic can be used on inexpensive edge devices and on larger server systems, the user can start with a small installation and later expand without losing the data model. This makes the approach suitable for small agriculture, education, experimentation and gradual professionalization of controlled environment growing.

Acknowledgement

The author acknowledges the practical development of the Gardener system, the experimental work around controlled-environment growing units, and the broader open-source and embedded-systems communities whose tools and standards make reproducible edge automation possible.

The author also acknowledges the value of accessible automation for small growers, educational projects and technical experimentation, where affordable systems can support learning, documentation and better process understanding.

Fig.1. Greenhouse and Units are currently tested on SPU Nitra.

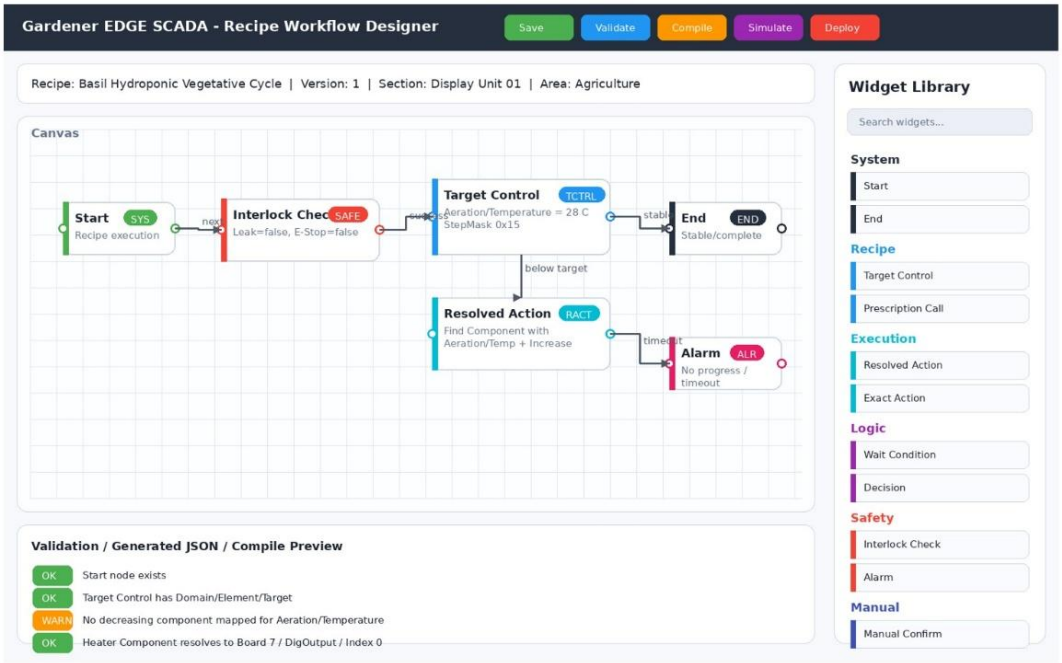

Fig.2. GUI of the server WebApp Gardener, accessible on-line on the SandBox.

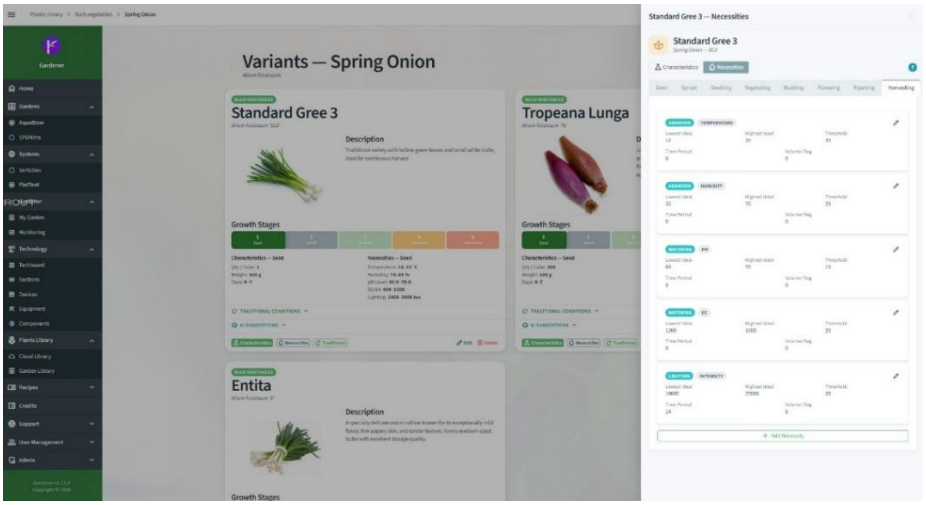

Fig.3. GUI of the server WebApp Gardener, accessible on-line on the SandBox.

References

[1] IEEE Standards Association. (2019). IEEE 754-2019 - IEEE Standard for Floating-Point Arithmetic. IEEE.

[2] International Organization for Standardization. (2018). ISO 14067:2018, Greenhouse gases - Carbon footprint of products - Requirements and guidelines for quantification. ISO.

[3] World Resources Institute and World Business Council for Sustainable Development. (2011). Product Life Cycle Accounting and Reporting Standard. Greenhouse Gas Protocol.

[4] Kernighan, B. W., Ritchie, D. M. (1988). The C Programming Language. 2nd edition. Prentice Hall.

[5] Welford, B. P. (1962). Note on a method for calculating corrected sums of squares and products. Technometrics, vol. 4, no. 3, pp. 419-420.

[6] Montgomery, D. C., Runger, G. C. (2014). Applied Statistics and Probability for Engineers. 6th edition. Wiley.

[7] Boardman, A. E., Greenberg, D. H., Vining, A. R., Weimer, D. L. (2018). Cost-Benefit Analysis: Concepts and Practice. 5th edition. Cambridge University Press.

[8] Association for Computing Machinery. (2012). ACM Computing Classification System. ACM.

[9] International Organization for Standardization. (2006). ISO 14040:2006, Environmental management - Life cycle assessment - Principles and framework. ISO.

Author

Jozef Dubovec
AQUASTOW s.r.o., Bratislava, Slovakia
jozef.dubovec@aquastow.eu

He studies theoretical cybernetics, working as a business and technical consultant for various institutions in banking area, ESA, Monsha'At, architect of the Gardener and SCADA480.