Chemical Supremacy: Automated Laboratory Systems and Strategic Boosting Protocols in Screeps Infrastructure

The transition of a decentralized autonomous colony within the Screeps environment from the mid-game energy economy to the late-game chemical hegemony represents the most significant architectural hurdle for any automated system. While the early stages of development are characterized by the acquisition of energy and the expansion of Room Controller Levels (RCL), the attainment of RCL 6 introduces the mineral economy, and RCL 7 and 8 introduce the complex laboratory systems required for high-level competition. The fundamental goal of these systems is the automated production of Tier 3 (T3) compounds, which can amplify creep performance by factors of up to four hundred percent, transforming a standard unit into a specialized agent capable of dismantling entire unboosted armies. This report provides an exhaustive engineering analysis of the best practices for laboratory automation, mineral logistics, and strategic boosting, contrasting the legacy patterns of the grgisme/screeps era with the high-performance, process-driven architectures of the modern ecosystem.

The Physical Architecture of Laboratory Systems: Spatial Logic and Geometric Optimization

The deployment of laboratory structures is governed by a rigid geometric constraint: a laboratory performing a reaction must be within a range of two squares of its two reagent sources. This simple rule necessitates a highly optimized spatial layout to maximize throughput while minimizing the logistics overhead of the "Scientist" creep responsible for resource shuffling.

Geometric Optimization and Layout Paradigms

In a 10-lab environment—the maximum allowed at RCL 8—the objective is to designate two "Source" or "Input" labs that can supply reagents to the remaining eight "Sink" or "Output" labs. If the source labs are centrally located, the output labs can be clustered around them such that all eight outputs are within range two of both inputs. This allows for a massive parallelization of the runReaction command, producing up to 80 units of mineral compound every 10 to 160 ticks, depending on the reaction type and cooldown period.
Two primary layouts have emerged as the industry standard: the "Flower" and the "Diamond." The Flower layout centers two labs and surrounds them in a hexagonal or circular pattern, which is often favored for its accessibility to logistics creeps. The Diamond layout, often integrated into a 13x13 or 15x15 "Bunker" stamp, aligns labs in a grid that is easier to protect with ramparts but may introduce slight pathing inefficiencies for the Scientist. A variation of these, the "Figure-8" layout, facilitates scaling from RCL 6 (3 labs) to RCL 8 (10 labs) by adding

labs in a predictable progression that maintains range-two integrity.

Layout Feature	Flower Cluster	Diamond Grid	Figure-8 Progression
Input Lab Placement	Centralized / Dual-Hub	Grid-Aligned	Centralized / Linear
Output Lab Capacity	8 (Parallel)	8 (Parallel)	1 to 8 (Scalable)
Scientist Access Tiles	1 to 2	2 to 4	1 to 2
Rampart Coverage	Circular / Tight	Square / Efficient	Variable
Recommended Use	Compact Bases	Bunker-Style Planning	Progressive Growth

Algorithmic Lab Placement: Distance Transform and Floodfill

For a top-tier bot, manual placement of labs is an architectural dead-end. The system must utilize automated room planning algorithms to identify the optimal lab site during the "Colony Boot" phase. The primary algorithm used is the Distance Transform, which calculates the distance of every walkable tile from the nearest wall or exit. By targeting tiles with a distance value of at least 3, the bot ensures there is a 5x5 or larger open space suitable for a lab cluster. Once a candidate center point is found, a Floodfill algorithm is executed to categorize surrounding tiles by their accessibility and proximity to core structures like the Terminal and Storage. Because the Scientist creep must move frequently between the Terminal and the Labs, the "Logistics Cost" (the product of distance and transfer frequency) must be minimized. The most efficient systems place the lab cluster immediately adjacent to the Terminal, often sharing a single standing tile for the Scientist creep to minimize move intents.

The Science Overlord: A Process-Driven Architectural Framework

The transition from legacy "Role" based scripts to a "Kernel/Process" architecture is nowhere more critical than in laboratory management. A "Science Overlord" acts as a high-level manager that orchestrates the execution of "Science Processes," abstracting the complexity of chemical stoichiometry away from the general creep logic.

The Kernel Model and Lab Processes

In the modern kernel architecture, the Science Overlord does not iterate through creeps; instead, it manages a persistent state machine for each room's laboratory complex. This state machine tracks the "Lab Order"—a data structure containing the target product, the required amount, and the current progress. The Overlord divides this mission into several discrete sub-processes:

1. **Reaction Scheduler:** Calculates the necessary precursors and invokes runReaction whenever a lab's cooldown reaches zero.
2. **Logistics Manager:** Interfaces with the room's logistics queue to request a Scientist creep to fulfill reagent needs.
3. *Boost Requester:* Monitors the colony's defensive and offensive status to trigger the production of combat compounds.

This separation of concerns allows the bot to handle complex, long-running chemical chains without blocking the main execution loop. If a specific reaction requires 5,000 ticks to complete, the process simply persists in the heap memory, waking up only when the Scientist has

deposited new reagents or a reaction has completed.

Recursive Dependency Management and Stoichiometry

The most significant challenge in lab automation is the multi-tiered nature of the reaction graph. To produce a T3 compound like Catalyzed Ghodium Acid (XGH_{2}O), the system must first produce Ghodium (G), Hydroxide (OH), and Ghodium Acid (GH_{2}O). A top-tier Science Overlord utilizes a recursive formulation function that takes a target resource and quantity and returns a list of "Missing Precursors."
The stoichiometry of these reactions is a 1:1 ratio: 1 unit of Reagent A + 1 unit of Reagent B = 1 unit of Product C. However, the time required is not constant. For example, producing Hydroxide takes 20 ticks, while producing Catalyzed Ghodium Acid takes 80 ticks. The Science Overlord must account for these time deltas to prevent the lab complex from idling.

Compound Tier	Example Formula	Reaction Time (Ticks)	Key Utility
Base	Z + K \rightarrow ZK	5	Precursor only
Tier 1	G + H \rightarrow GH	10	Moderate WORK boost
Tier 2	GH + OH \rightarrow GH_{2}O	15	Significant WORK boost
Tier 3	GH_{2}O + X \rightarrow XGH_{2}O	80	Maximum WORK boost

The recursive logic must also check the current Terminal and Storage inventory. If the bot already possesses 500 units of GH_{2}O, it should skip the precursor reactions and move directly to the final tier with Catalyst (X). This dynamic order generation ensures that the lab complex always operates at maximum economic efficiency.

The Scientist Creep: State Machines and Inventory Logistics

The "Scientist" is a specialized logistics creep whose sole function is to move minerals between the Terminal, Storage, and Labs. Unlike legacy haulers that use simple "Pull" logic, the Scientist is a high-precision agent driven by the Science Overlord's current orders.

Finite State Machine Implementation

To ensure deterministic behavior and minimize CPU overhead, the Scientist operates on a Finite State Machine (FSM). This avoids the "re-deciding" problem where a creep recalculates its target every tick. The core states include:

1. **RENEW:** Moving to a spawn to increase ticksToLive before starting a high-value transfer, preventing the creep from dying while carrying 3,000 units of expensive minerals.
2. **REAGENT_SUPPLY:** Withdrawing the required quantities of Ingredient_{1} and Ingredient_{2} from the Terminal and depositing them into the designated input labs.
3. **PRODUCT_COLLECTION:** Withdrawing finished minerals from the output labs once they reach a threshold (e.g., 100 units) and returning them to the Terminal.
4. **CLEANUP (FLUSHING):** The most critical state. If an order changes or a lab contains the wrong mineral type, the Scientist must purge the labs to prevent a "Lab Jam".

The Lab Jam and Purging Protocols

A "Lab Jam" occurs when a lab contains a residual amount of a mineral that is not part of the current reaction order, preventing the input of new reagents. This often happens after a global reset or a change in high-level strategy. The Scientist must be programmed with a "Purge" protocol that scans all 10 labs against the current LabOrder. If a lab's store contains an unauthorized resource, the Scientist immediately prioritizes emptying that lab into the Terminal. Failure to implement this cleanup logic is a primary reason why lower-tier lab automation systems fail over long durations.

Empire-Wide Mineral Logistics: The Terminal Network

In the endgame, no room is an island. Minerals are distributed unevenly across the world map; for instance, Zynthium (Z) and Keanium (K) are found in different quadrants than Utrium (U) and Lemergium (L). To sustain a continuous T3 production pipeline, an empire must establish a robust Terminal logistics network.

The Quota and Delta System

The industry-standard for inter-room logistics is the Quota System. Each room's OfficeMemory defines a target quantity for every resource type. The Logistics Overlord then calculates the "Delta" for each room.
A positive delta indicates a surplus, while a negative delta indicates a deficit. The system then matches "Providers" (Surplus > 0) with "Requesters" (Deficit < 0) using a stable matching algorithm that prioritizes the shortest linear distance between rooms to minimize the energy cost of the Terminal.send() command. This ensures that catalysts (X) and reagents are distributed across the empire where they are most needed.

Terminal vs. Storage Concentration

There is an architectural debate regarding where to store minerals. Storing minerals in the StructureStorage allows for massive capacity (up to 1,000,000 units), but transferring them to the Terminal for shipping requires multiple Scientist ticks. High-performance bots often concentrate minerals in the Terminal until it reaches a specific threshold (e.g., 50,000 units) before moving the overflow to Storage. This "Terminal-First" approach ensures that resources are always ready for immediate inter-room transfer or market sale.

Strategic Boosting: The Request Protocol and Body Optimization

The ultimate objective of laboratory automation is the application of boosts. A boost is a permanent modification that enhances a specific body part type for the duration of the creep's 1,500-tick life.

The Boost Request Protocol

Legacy bots often "Push" boosts to creeps, which is brittle and inefficient. Modern bots use a "Request" protocol. When a creep is spawned, its logic identifies its mission requirements (e.g., "I am a siege attacker, I need TOUGH and ATTACK boosts"). It then enters a BOOSTING state and registers a request with the room's Science Overlord.
The Science Overlord then reserves the necessary minerals and energy in the designated labs. The creep moves to the lab and calls StructureLab.boostCreep(creep). Each boost application costs 30 mineral units and 20 energy units per body part. This request-based system allows the Science Overlord to prioritize boosts for high-priority units, such as defenders, over routine production.

Body Part Sequencing and Effective Health

The efficacy of boosts is non-linear when combined with strategic body part sequencing. Damage in Screeps is applied to body parts in the order they appear in the array. Therefore, "Tough" parts should always be placed at the front of the body. When boosted with Catalyzed Ghodium Oxide (XGHO_{2}), these parts take 70% less damage.
This effectively triples the health of the creep's front-line "Armor" parts. Furthermore, since empty CARRY parts and boosted MOVE parts generate less fatigue, a bot can design "Capacious" haulers that move at full speed with significantly fewer MOVE parts, freeing up energy for more productive work.

Boost Type	Compound	Efficiency Multiplier	Primary Application
Upgrade	XGH_{2}O	+100% (No extra energy)	Power-leveling GCL
Heal	XLHO_{2}	+300% Effectiveness	Quad formation survival
Tough	XGHO_{2}	-70% Damage Taken	Siege tanks and defenders
Attack	XUH_{2}O	+300% Effectiveness	Dismantling enemy ramparts
Move	XZHO_{2}	+300% Fatigue reduction	Rapid response and logistics

Military Architecture: Quads, Tower Draining, and Siege Logic

In high-level play, an unboosted creep is effectively a liability. Combat is won by the side with the more efficient chemical supply chain.

The Quad Formation

The "Quad" is a specialized squad of four T3-boosted creeps moving in a 2x2 square. These creeps coordinate their actions to act as a single unit. They use "Cross-Healing," where each creep heals the squad member with the lowest HP. To sustain a Quad, the laboratory system must be capable of providing 1,200 units of various T3 compounds (30 units x 10 parts x 4 creeps) in a single burst. A failure in the lab automation pipeline (e.g., missing a MOVE boost

for just one creep) will break the quad's formation, leading to immediate tactical failure.

Tower Draining and Threat Analysis

A common defensive tactic is to use "Tower Draining". An attacker sends a T3-boosted TOUGH/HEAL creep to stand in range of an enemy's towers, forcing them to spend energy. A modern defensive bot must be able to perform "Threat Potential" analysis. If the incoming creep has T3 HEAL parts that out-heal the tower's damage, the tower should stop firing to conserve energy for a more viable target. This highlights the need for a Lab system that can pivot production from "Economy" boosts to "Combat" boosts in a single tick when an invasion is detected.

Economic Engineering: Market Arbitrage and Credit Liquidity

The laboratory system is not only a military asset but a primary driver of the colony's credit balance. A bot that can refine raw minerals into T3 compounds can realize massive profits on the public market.

The "Make vs. Buy" Decision Matrix

A top-tier bot must constantly evaluate the market price of minerals versus their production cost. If the price of XGH_{2}O is less than the combined price of G, H, and X plus the CPU cost of refining, the bot should simply buy the finished product.
Modern bots implement "Automated Trading" logic that places "Buy" orders at low prices for rare minerals and "Sell" orders for surplus local minerals. This ensures that the colony's credits are always being reinvested into the chemical pipeline, maintaining a high level of "Credit Liquidity."

CPU Efficiency in Chemical Operations

The StructureLab.runReaction method and the boostCreep method each cost 0.2 CPU per intent. In a large empire with 20 rooms, running 160 reactions per tick (8 per room) would consume 32 CPU—nearly a third of the standard limit. To optimize this, the Science Overlord must:

1. **Batch Reactions:** Only run reactions in rooms where the product is actually needed or when the bucket is full.
2. **Heap Caching:** Store all lab object references in the global heap to avoid expensive find calls.
3. **Order Prioritization:** Halt T1 precursor reactions if the T3 final product quota is already met, saving both CPU and energy.

Migration Roadmap for grgisme/screeps

To modernize the legacy grgisme repository for top-tier lab performance, the following roadmap is recommended:

Phase 1: Spatial Automation (RCL 6)

Abandon manual lab placement. Implement the Distance Transform planner to identify a 5x5 area for the lab cluster. Integrate a basic 3-lab "Starter" reaction loop to begin producing Ghodium for Safe Modes.

Phase 2: The Science Overlord (RCL 7)

Implement the Kernel/Process model. Transition the Scientist from a simple "Role" to an FSM-driven process managed by the Science Overlord. Introduce the recursive dependency resolver to handle the multi-tiered reaction graph.

Phase 3: The Global Mineral Network (RCL 8)

Establish the Quota/Delta system for Terminal-to-Terminal transfers. Implement the Request Protocol for boosting, allowing combat creeps to "Order" T3 boosts during their spawning phase. Integrate automated market trading to fill gaps in the catalyst (X) supply.

Conclusion: Chemical Supremacy as the Ultimate Benchmark

The implementation of a top-tier laboratory automation and boosting system represents the pinnacle of Screeps engineering. It is the point where a bot transcends simple strategy and enters the realm of complex systems management. By adopting a process-driven architecture, optimizing spatial layouts through automated planning, and integrating empire-wide mineral logistics, a developer can achieve a level of tactical dominance that is unreachable through standard unboosted play.

The transition from the legacy, monolithic loops of 2017 to the high-efficiency, chemical-hegemony models of 2025 is not merely an upgrade; it is a fundamental shift in how the Screeps environment is perceived. The bot is no longer a collection of scripts, but a distributed industrial complex, converting raw matter into the refined catalysts of empire expansion. In the endgame of Screeps, those who control the labs control the world.

Final Metrics and Strategic Comparison Table

Metric	Legacy (grgisme Era)	Modern (Top-Tier OS)	Impact on Competitive Play
Lab Management	Manual / Hard-coded	Science Overlord / Process	100% uptime of reactions
Reaction Logic	Single-tier / Simple	Recursive / Hierarchical	Access to T3 compounds
Scientist Logic	Static Pull-based	FSM with Purge Protocols	Zero "Lab Jams" or downtime
Logistics	Room-local only	Empire-wide Quota System	Unlimited mineral accessibility
Boosting	"Push" (Hard to	"Request"	Precision-enhanced

Metric	Legacy (grgisme Era)	Modern (Top-Tier OS)	Impact on Competitive Play
	manage)	(Just-in-Time)	units
Combat Power	Unboosted / Weak	T3 Quad-form / 4x Stats	Tactical invincibility
CPU Efficiency	Low (Redundant searches)	High (Heap caching)	Scaling to 50+ rooms

The path forward for the user is the systematic re-engineering of the science pipeline, treating every mineral unit as a potential multiplier for empire-wide success. By following the best practices outlined in this report—specifically the use of FSM-driven scientists, recursive dependency solvers, and inter-room terminal balancing—the colony will achieve the chemical supremacy necessary to dominate the Screeps endgame..

Works cited

1. Resources | Screeps Documentation, https://docs.screeps.com/resources.html 2. Screeps #22: For Science | Field Journal, https://jonwinsley.com/notes/screeps-for-science#terminal-logistics 3. Lab runReaction() CPU Too High | Screeps Forum, https://screeps.com/forum/topic/1149/lab-runreaction-cpu-too-high 4. Base Building Considerations : r/screeps - Reddit, https://www.reddit.com/r/screeps/comments/87ed6x/base_building_considerations/ 5. Automating Base Planning in Screeps – A Step-by-Step Guide, https://sy-harabi.github.io/Automating-base-planning-in-screeps/ 6. Screeps #1: Overlord overload - Ben Bartlett, https://bencbartlett.com/blog/screeps-1-overlord-overload/ 7. Screeps after one year - Pedantic Orderliness, https://www.pedanticorderliness.com/posts/screeps 8. Great Filters - Screeps Wiki, https://wiki.screepspl.us/Great_Filters/ 9. Screeps #14: Decision Making - Field Journal, https://jonwinsley.com/notes/screeps-decision-making 10. Strange Reactions constant | Screeps Forum, https://screeps.com/forum/topic/1065/strange-reactions-constant 11. Creep Boost discussion | Screeps Forum, https://screeps.com/forum/topic/471/creep-boost-discussion 12. Screeping | World Domination via JavaScript, https://screeping.wordpress.com/ 13. Safety and Design Handbook.pdf - OSU Chemistry, https://chemistry.osu.edu/sites/chemistry.osu.edu/files/Safety%20and%20Design%20Handbook.pdf 14. P U R I F I C A T I O N L A B O R A T O R Y C H E M I C A L S, https://rexresearch1.com/LabDesignSafetyTechnique/Purification%20Laboratory%20Chemicals PERRIN.pdf 15. Creeps | Screeps Documentation, https://docs.screeps.com/creeps.html 16. Help me! | Screeps Forum, https://screeps.com/forum/topic/2524/help-me/21?lang=en-GB&page=1 17. Using Modern A.I. Paradigms To Improve Paradox A.I. | Page 3 | Paradox Interactive Forums, https://forum.paradoxplaza.com/forum/threads/using-modern-a-i-paradigms-to-improve-paradox-a-i.1142483/page-3 18. Creep Body Setup Strategies - Screeps Wiki, https://wiki.screepspl.us/Creep_body_setup_strategies/ 19. [Power] Power creep ideas | Screeps Forum, https://screeps.com/forum/topic/388/power-power-creep-ideas 20. Workflow tips and prioritization for new players? | Screeps Forum, https://screeps.com/forum/topic/2556/workflow-tips-and-prioritization-for-new-players 21. Tips - Screeps Wiki - Fandom, https://screeps.fandom.com/wiki/Tips