Swarm Dialogues: Communication policies that Shape Embedded Evolution in ESP32 Collectives

Luis Yallico Ylquimiche

***Abstract*—We contribute a hardware-in-the-loop swarm that enables peer-to-peer ESP-NOW transmissions and quantifies the impact of communication policies in ESP32 swarms (3-13 robots). We compare stochastic peer ordering with link-aware scheduling and test two contention controls: per-agent message budgets and latency-bounded random micro-delays. Link-aware scheduling lowers latency and jitter by X%, stochastic ordering spreads information faster by Y% but increases collision errors by Z% as density rises. Micro-delays stabilize latency with minimal throughput cost. Design rule: use stochastic scheduling with micro-delays for rapid diffusion, use link-aware scheduling when latency/jitter robustness is a priority.**

I. Introduction

This study profiles swarm communication on ESP32 hardware and demonstrates how agent density, locomotion, topology inference, imposed message budgets, and stochastic transmission policies shape both network quality of service (QoS) and embedded evolution performance. Focusing on direct peer-to-peer links, we contribute with an empirical mapping that probes communication bottlenecks flagged in recent surveys, and evidence for extracting tangible design rules for swarm networks.

Swarm engineers draw inspiration from social biological systems such as ants, bees or termites to build decentralised robot collectives that are inherently robust to failure, flexible across tasks and scalable in number [1]. In swarm systems, collective intelligence emerges when individual robots trade packets of information among neighbouring robots. Classic ant-colony-optimisation work in the early 2000s has already proven that an indirect information exchange of "virtual-pheromones" can lead to agents collectively discovering optimal routing formations [2]. Hence, communication design is a first-order determinant of emergent behavior in swarms.

While coordination and task allocation have been widely studied in swarm robotics, two recent surveys agree that communication bandwidth, latency and energy usage are the main blockers to real-world swarm deployments [3][4]. These issues become more pronounced as swarms sizes scale, resulting in an increase in data volumes being transmitted among peers, often overwhelming individual agents' limited compute capabilities. Beyond sheer capacity, the architecture of the communication also matters. Many swarms rely on blind broadcast communication schemes that scale poorly, with collision rates rising sharply beyond a few dozen peers which tend to reduce the reliability of the network [4].

Communication is equally critical when controllers evolve on-line. Recent embodied evolution studies explain that less communication can enhance swarm performance, as trimming neighbourhood size helps populations forget outdated beliefs and re-adapt faster [5][3]. While our context is controller evolution, we instantiate it as a distributed global optimisation problem. Fixing behaviours and evolving solutions to a known objective, isolates the communication effects and makes adaptation dynamics comparable across conditions.

To support our investigation, we implement a hardware-in-the-loop framework by leveraging over-the-air (OTA) updates. For embedded evolution we use ESP-NOW with an island-model genetic algorithm (GA). Moving beyond simulation-heavy work, we experiment with Pololu 3Pi+ robots, as shown in Fig. 1b, and adopt a state-of-the-art cloud enabled system that captures granular packet and evolution data in real-time, yielding reproducible datasets and enabling controlled parameter sweeps.

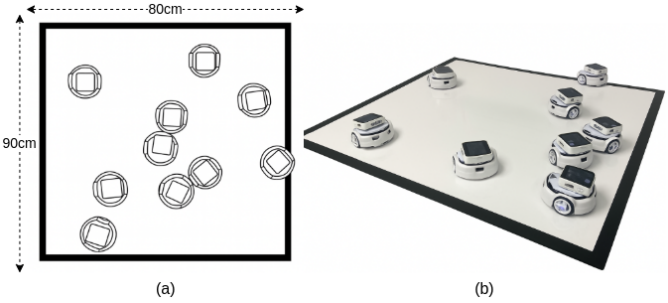

Fig. 1. Experimental setup: (a) arena dimensions and (b) example of a Brownian locomotion experiment with 8 robots.

We explore three communication factors and their measurable effects (i) peer ordering, (ii) fixed per-robot message budgets, and (iii) per-packet stochastic transmission delays. We ask, **RQ1**: whether communication-aware peer ordering using *link-quality* as a lightweight proxy for topology (see Section IV-D1) changes network QoS, connection formation patterns, and time-to-consensus relative to stochastic ordering; **RQ2**: whether these factors influence information diffusion or convergence speed compared with one another; and **RQ3**: whether micro-delays between transmissions can mitigate collision bursts without harming delivery by measuring proxies such as latency and throughput.

II. Related Work

Island-model evolutionary algorithms in [6], show that migration (genome exchange) topology and connectivity structure govern convergence dynamics not just the migration rate, with the Rastrigin function Eq. 1, used to benchmark the performance of different asynchronous evolutionary algorithms.

In [7], controllers are evolved for ad-hoc aerial relays without positional information, using periodic single-hop broadcasts to neighbours. This concept is leveraged on our experimental setting where the agents exploit link-quality proxies to bias their connections. In non-local communication schemes such as [8], network-wide epidemic broadcasting known as *flooding* is used to accelerate the diffusion of information, for environmental mapping experiments. In contrast, our study uses *unicast* peer-to-peer links at the data-link layer, enabling direct round-trip time measurements.

Constrained connectivity repeatedly appears beneficial. The *"less is more"* effect in [9], demonstrates that fewer links and lower swarm densities improve adaptation in swarm consensus tasks, by helping robots discard stale beliefs. A similar effect is also observed by [5] over embodied evolution, where limiting the genome-exchange range can support the evolutionary search maintain higher diversity for longer and escape local optima. We explore similar properties which are discussed in Section-VI, the main difference being the data-links used and our message prioritisation algorithm.

Transmission timing also matters. Experiments from [10] show that limiting the frequency of communication stabilises swarm behavior, while [11] explains that the speed of consensus among peers tolerates random transmission delays even when agents receive multiple out-of-order messages. Furthermore, non-radio data links have also been explored by [12], which shows that infra-red (IR) line-of-sight local communication can be used in swarms, some benefits include lower energy requirements per bit, moreover we note that IR has limits on bandwidth and bit error probability that scales with range.

Complementary work by [13] examines real-time communication middleware for swarms at the application layer, addressing peer-to-peer IP protocols. In contrast we explore direct peer-to-peer exchange managed without any middleware.

III. Experimental Setting

In this study, we use Eq. 1 to benchmark the evolutionary performance of the swarm. This global optimisation task was chosen to emulate evolutionary controller optimisation, while no controller was actually evolved the concept of robots sending and receiving genomes from their peers remains the same. Making swarm behaviour a control variable.

The environment for the experiments was a rectangular arena measuring 80x90cm without obstacles (Fig. 1a), where the initial positions, and number of the robots was determined by the experiment schedule (Table IV). Across the study, we evaluated swarm under different communication and system configurations. Each experiment manipulates a specific independent variable while holding all other conditions constant. These include:

- **Swarm density**: The number of agents deployed simultaneously, ranging from 3 to 13.
- **Locomotion**: Robots were either *stationary* or navigated using a *Brownian motion* gait.
- **Topology inference**: Peer transmission priority was governed by either a *stochastic* shuffle or a *comm-aware* ranking strategy based on link quality metrics.
- **Message limits**: A *message budget* controlled how frequently agents could communicate with one another.
- **Transmission frequency**: Each transmission was optionally delayed by a random interval derived from the maximum observed peer latency.

Note that a detailed description of the communication independent variables is available in Section IV-D.

A. Rastrigin Function

The Rastrigin function is defined as follows:

$$f_R(\mathbf{x}) = 10n + \sum_{i=1}^{n} \left(x_i^2 - 10\cos(2\pi x_i)\right) \tag{1}$$

Where n is the number genes migrating between islands (robots). Whereas, x is the individual genome being evaluated. The function has a global minimum at $f_R(\mathbf{x}) = 0$ [6]. Early single-robot trials showed that $n = 10$ created enough search difficulty for performance to be measurable across swarm sizes.

B. Robot Platform

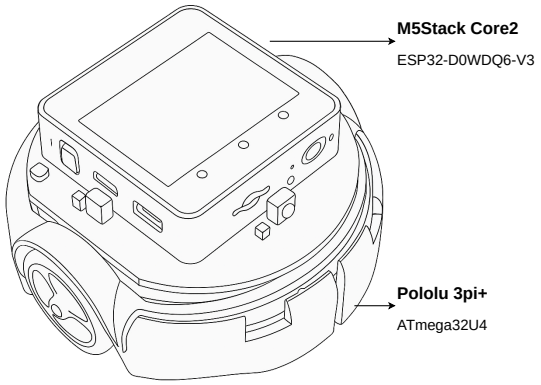

Fig. 2. The Swarm-B2 platform: M5Stack Core2 and Pololu 3Pi+

Table I summarizes the main hardware components of the Swarm-B2 platform used in this study [REF]. Devices coordinate via a I^2C 100kHz bus. The ESP32 microcontroller runs dual-core FreeRTOS tasks, that logs real-time data to the local SD card, and manages communication using the ESP-NOW data link (2.4 GHz).

TABLE I
SWARM-B2 HARDWARE STACK

Component	Interface	Function
M5Stack Core2 (240 MHz)	I^2C Master	Embedded evolution, ESP-NOW communication, data logging and user interface
Pololu 3Pi+ (16 MHz)	I^2C Slave	Locomotion using bumper and line following sensors

The Pololu 3Pi+ is equipped with a line following sensor array and two bump sensors, which were used to navigate the around arena as the robots have no positional sensing. A *Brownian-motion* gait was selected to maintain unbiased mobility across the arena and ensure constant movement. The gait code exposes the Pololu 3pi+ slave to the ESP32 master node, this interface lets the ESP32 act as a passenger with override, that can set speed scaling factors or raise `START`/`STOP` flags without touching the low-level control loop. Figure 15 shows the on-board display, used only for debugging and device status during development. A real-time clock (RTC) synchronised experiment runs across the swarm.

IV. IMPLEMENTATION

Reliable and precise data logging is a pre-requisite for evaluating the communication performance and evolution dynamics of the swarm. To achieve this, our framework implements several mechanisms to capture and record key metrics which are explained in this section.

A. System Architecture

The firmware was implemented using the Espressif's ESP-IDF framework to provide real-time multitasking. Firmware updates and telemetry data capture are cloud enabled. Code that is written locally in VSCode (ESP-IDF v5.1.4) is rebuilt by GitHub Actions for validation. Only binaries that pass tests are signed and uploaded to an Amazon Web Services (AWS) S3 OTA bucket with a versioned register. On boot, each robot fetches the version register, streams any newer release into an idle OTA slot, and then resets with the latest firmware.

Experiment schedules ran in batches. During execution, time-stamped messages and event logs are written to a local SD card. After execution, the robot uses WiFi to upload telemetry into an AWS S3 bucket. These files are serialised as JSON, tagged with the `experiment_id`, and later processed with AWS Glue/Athena and visualised in PowerBI. This data processing pipeline, allowed us to harvest high-volumes of traceable log data per-robot and per-experiment (see Section IV-E), while keeping the ESP-NOW data link isolated from Wi-Fi interference during experiments.

B. Embedded Evolution

The swarm uses a distributed genetic algorithm to find the global minimum for Eq. 1. A visual representation of this is shown in Fig. 4. As the local population in each agent evolves, the swarm begins to communicate their local fitness and corresponding genome to their peers.

Our implementation employs an elitist migration strategy, this happens when the local GA reaches a patience threshold and the agent pushes its "best" (lowest fitness score) genome to other swarm members via ESP-NOW. The incoming remote genes from another peer are integrated into the local population by replacing the worst performing 5% individuals, this value was chosen to preserve genomic high fitness locally.

To avoid stagnation over a local-minima and promote exploration across the swarm, a mass extinction event together with a hyper-mutation mechanism tracks consecutive non-improving generations [14]. In each agent, if the local population's fitness remains stale for over 3 seconds, the mutation probability is temporarily increased and the lowest performing half of the population is re-initialised.

Evolution continues locally until the experiment ends, triggered either by (i) discovering the global optimum (fitness = 0) or (ii) a 60 s timeout with no convergence. Two FreeRTOS tasks run in parallel, one executes the GA and the other handles ESP-NOW messaging. During shutdown, the `espnow_task` flushes any queued packets. Then evolutionary and communication files are uploaded to the telemetry S3 over HTTPS.

C. Message Structure

We evaluate robot-to-robot communication with a fixed ESP-NOW message schema (Table II), floats are rounded to $3d.p.$ to send the message in a single transmission. Implications of this design choice are discussed in Section VI.

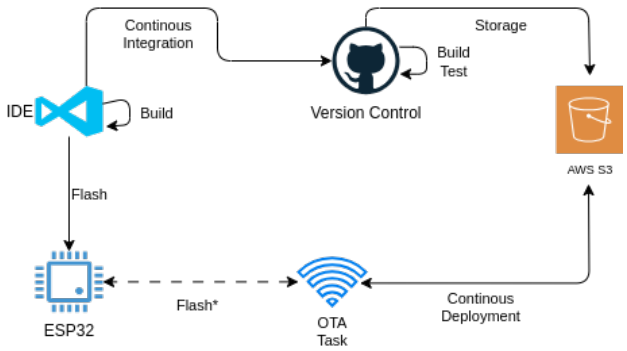

Fig. 3. Integrated architecture: Local development, GitHub repository, AWS S3 storage, and OTA updates.

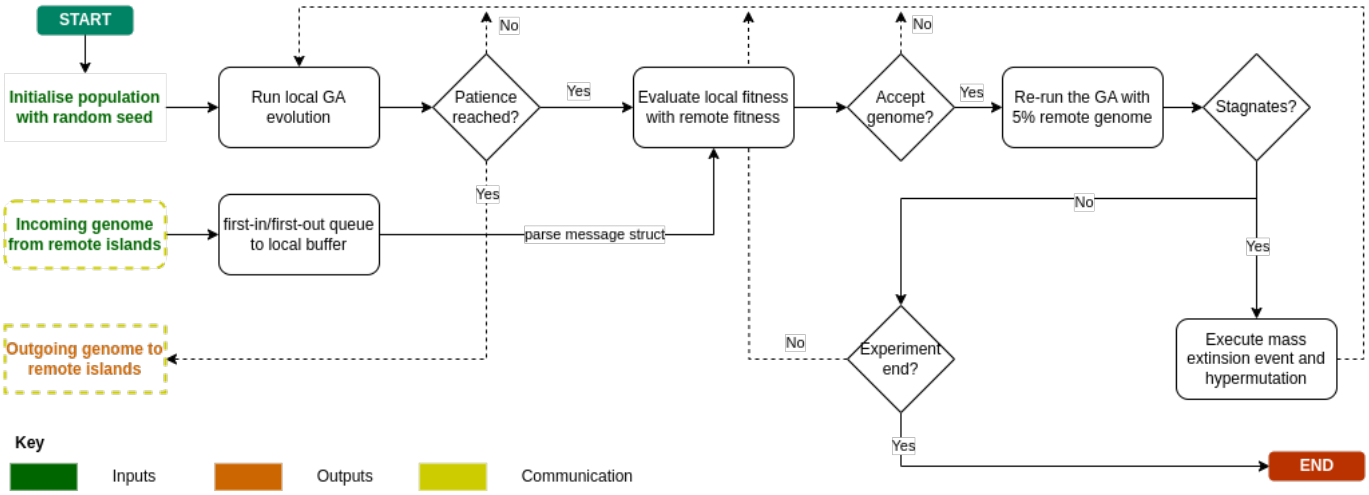

Fig. 4. Distributed island-model GA executed on the ESP32 swarm

TABLE II
MESSAGE STRUCTURE USED FOR ESP-NOW FITNESS VALUE AND GENOME EXCHANGE

Field	Type	Description
`log_id`	uint32_t	Unique internal identifier of the event.
`robot_id[5]`	char	MAC identifier for the sender robot plus a null terminator.
`log_datetime`	time_t	Timestamp based on the internal RTC.
`message[128]`	char	Content of the message including the fitness value and the 10-gene genome delimited by "|".

D. Communication Layer

In contrast to previous work by [13] that explored decentrilised IP communication protocols using middleware in swarms, we exploit the ESP32s Medium Access Control (MAC) layer to keep packet timing and load directly measurable.To do this, we had to implement ESP-NOW by pre-registering the MAC addresses of all peers in each agent's firmware, enabling direct unicast communication between robots. We do not view this as a hard scalability limit in the swarm paradigm, as [15] explains that gossip-style OTA propagation can disseminate firmware across the swarm, allowing MAC registers to scale without central coordination.

Our design uses push only event-driven peer-to-peer messaging, where each device transmits data to peers (migration) without waiting for requests. Using this scheme avoids the complexity of pull-based communication protocols. Recall from Section IV-B that only local elite genomes migrate, this takes place asynchronously between islands as each robot's GA reaches migration thresholds at different times. Because of this, incoming messages are handled by a first-in/first-out queue and buffered locally for later processing. This allowed internal task operations to continue uninterrupted, at the cost of internal memory, from pilot experiment we set the queue size to 40 (8KB).

Social evidence, in our case is present in the form of the communication quality that arises from migration. Specifically from the latency and Received Signal Strength Indicator (RSSI) that a peer perceives relative to other peers. We use this social evidence to inform our communication strategy and adapt the following policies.

1) Topology Inference

We investigate the impact of two communication schemes, based on the social evidence retained locally per-peer. These schemes were designed to influence the priority and order with which each agent sends messages. We use the social evidence as a proxy to infer the topology, under the assumption that distant islands would exhibit different link quality characteristics. These are implemented as follows:

- *Stochastic*: Each agent calls a random seed to apply a Fisher-Yates shuffle [16] over the list of peer MAC addresses, sorting them before sending messages to *all peers*. No social evidence bias is used, every order permutation is equally likely.
- *comm-aware*: Each agent ranks its peers based on the most recent measurements of latency and RSSI. Peers with unknown metrics (at initialisation) are prioritised first to ensure social evidence is available. Then, peers are scored by normalizing both latency and RSSI, and those only in the *worst half* (highest latency, lowest RSSI) are messaged.

For *stochaistic*, the shuffle is defined in Eq. 2, where (j_i) is a random index drawn from a uniform distribution. σ is the random permutation of MAC addresses for each i peer. We can think of this scheme as a *pseuso-broadcast* using unicast communication, as the genome is sent to all peers with negligible delays between transmissions.

$$\sigma \;=\; \prod_{i=n}^{2} \big(i,\, j_i\big), \qquad j_i \sim \mathrm{Uniform}\{1,\ldots,i\} \tag{2}$$

Whereas for *comm-aware*, the scoring for peer i ranking is defined in Eq. 3. Here, the round-trip latency l can be measured via acknowledgements (ACKS). The RSSI r is reported by the ESP-NOW send callback after each successful transmission. If multiple messages have been sent, only the most recent metrics stay in memory. Note that both RSSI and latency are scored equally.

$$Score_i = \frac{l_i - l_{\min}}{l_{\max} - l_{\min}} + \frac{r_{\max} - r_i}{r_{\max} - r_{\min}} \tag{3}$$

2) Message Budgets

Inspired by the reported "less-is-more" effects, we cap the number of migration messages that each robot may broadcast (Eq. 4). Each agent is given a message budget B per sliding time window W. Once the budget is exhausted, the robot stays silent until the window advances, no matter how often its onboard genetic algorithm requests migration. Different from earlier IR-based systems that relied on a short-range *narrowcast* channel [10], we deliver the message to all peers using the topology inference schemes discussed previously.

$$\forall t: \quad M_W(t) = \sum_i \mathbf{1}_{\{t-W<t_i\le t\}} \le B \tag{4}$$

Here, t_i are the send times, $M_W(t)$ is the number of broadcast messages sent in the last $W = 8$s, and the budget is fixed at $B = 1$. These values were chosen after pilot tests showed that, without the budget, each peer emitted roughly 15 messages per experiment, effectively limiting migration by half.

3) Transmission Frequency

To further explore the communication behaviour of the swarm under simultaneous transmission conditions, we modulate the migration interval between islands. Each migration is postponed by random delay drawn from the highest peer-to-peer latency observed so far, which is estimated from the latest ACK round-trip times on every link, described by Eq. 5:

$$\Delta t \sim \mathrm{Uniform}\big(0,\, L_{\max}\big), \qquad L_{\max} = \max_j\{\tau_j\} \tag{5}$$

where (τ_j) is the one-hop latency to peer (j). If random mode is disabled, $(\Delta t = 0)$ and messages are sent immediately by calling `esp_now_send`. The intention of this approach is to control the transmission of messages in a stochastic manner to reduce unintended channel contention in the swarm network.

E. Communication Performance Metrics

To evaluate the quality of the communication in the swarm, each robot records granular *sender-receiver*: latency l (ms), RSSI r (dBm), net kilobyte rate t (Kbps), transmission-error flags e, and per-core CPU load u. These raw logs are post-processed by the data pipeline previously detailed in Section IV-A. AWS then computes the aggregated per-device and per-experiment metrics, (i) L *mean latency*, the time taken for a message to be sent and acknowledged, (ii) J *mean jitter (ms)*, the standard deviation of successive latencies, (iii) E *mean error rate (%)*, the percentage of messages that were sent but not acknowledged, and (iv) T *mean throughput (kbps)*, the data is successfully transferred by each agent. To assess overall network performance, a QoS score was computed as:

$$\mathrm{QoS} = w_0 \cdot (1-\hat{L}) + w_1 \cdot (1-\hat{J}) + w_2 \cdot (1-\hat{P}) + w_3 \cdot \hat{T} \tag{6}$$

Where, $\hat{L}$, $\hat{J}$, $\hat{E}$, $\hat{T}$ are the normalised global metrics aggregated at the experiment-level in the range of $[0, 1]$ and the weights w_i are defined in Table III. A higher QoS value indicates better overall network communication performance.

TABLE III
QOS UTILITY FUNCTION WEIGHTS FOR DIFFERENT SWARM ROBOTICS APPLICATIONS

QoS	w_0	w_1	w_2	w_3	**Application**
QoS_c	0.5	0.25	0.15	0.1	Swarm consensus and voting
QoS_s	0.15	0.15	0.50	0.2	Swarm sparse deployments

V. RESULTS

As shown in Fig. 5a, the latency and jitter of the swarm network differ significantly between topology inference schemes. Across experiments, *comm-aware* generally achieves lower latency and more stable delivery compared to the *stochastic* mode. Fig. 5b-c show throughput vs topology/message limits. *Stochastic* yields higher raw throughput with wider variability.

Larger swarms reach lower mean fitness scores. *Stochastic* schemes accelerate information diffusion, where ≈80% of agents share the temporal "best" solution 20 seconds earlier than under *comm-aware*. Fig. 6a-b. The degree of connectivity per-peer across time shows that *stochastic* forms more links compared to *comm-aware*, both increasing over time. Fig. 7.

Error rates are small and comparable for smaller density swarms, rising during 13-agent swarm experiments and when the network operates under the *stochastic* scheme. Fig. 8a-b. With regards to locomotion Fig. 9a shows that Brownian motion results in slightly improved RSSI values, indicating more stable connections compared to static swarms. Whereas 9b suggests no statistical significance between RSSI and topology inference schemes.

Random micro-delays during transmission stabilise latency over time with negligible throughput loss, albeit mean jitter and latency values increase compared to the *comm-aware* scheme (Fig. 10a-c). The transmissions delays do not block convergence and achieve comparable adaptation rates

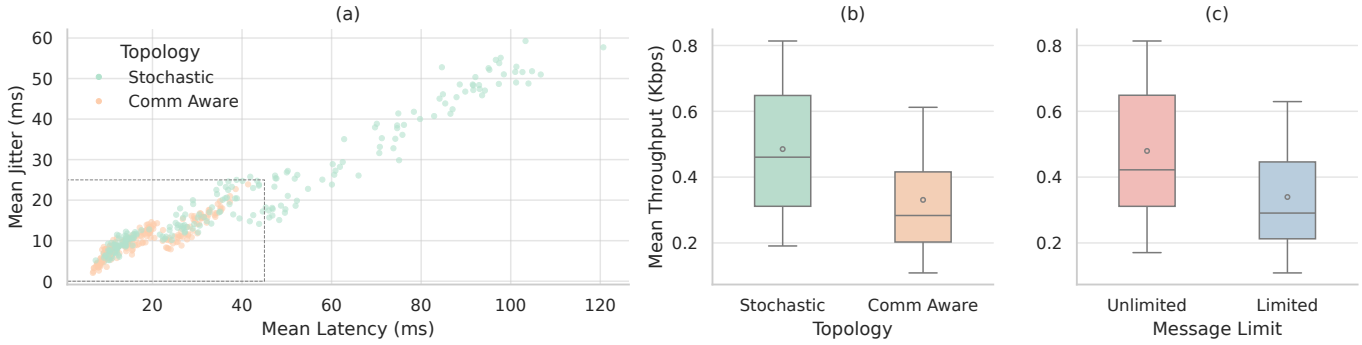

Fig. 5. (a) Latency and Jitter relationship under varying topology schemes, (b)(c) throughput distribution under topology and message limit variables

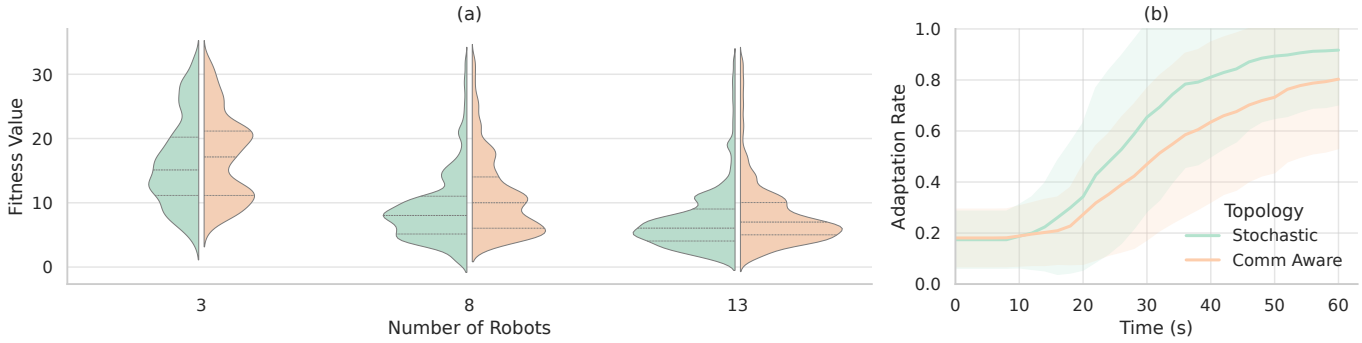

Fig. 6. (a) Mean device fitness values across different swarm sizes, (b) mean cumulative percentage of agents converging to the lowest fitness value across $2s$ windows

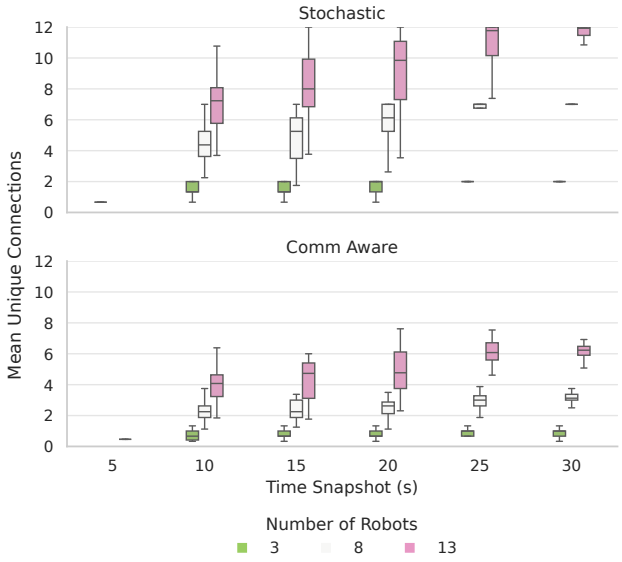

Fig. 7. Per-peer number of mean connections formed by time snapshot up to 30 seconds for both topology inference schemes

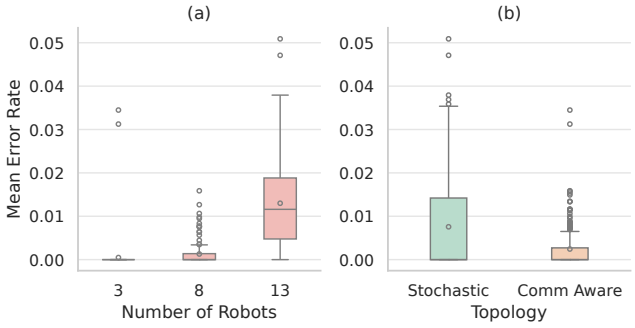

Fig. 8. (a) Mean error rates by device across varying swarm sizes, (b) mean error rate for different topology inference schemes

compared to the *stochastic* scheme (Fig. 11a-b).

An evaluation of the characteristics of the QoS metrics for the 13-agent swarm, show that (i) *comm-aware* achieves high QoS_c scores, (ii) modulated transmission shifts toward QoS_s but scores less in QoS_c, (iii) message limits do not have a clear effect on QoS, both kernel densities overlap each other. Fig. 12a-d.

VI. Discussion

Across conditions, lower end-to-end latency is driven primarily by limiting concurrent links, not by message limits, achieving sub-50ms comparable to [7] for swarm communication relay scenarios. Under *stochastic* scheduling, pseudo-broadcast rapidly increases unique connections,

TABLE IV
EXPERIMENTAL CONFIGURATIONS

Robots	Topology	Locomotion	Message Limit	Transmission	Latency (ms)	Jitter (ms)	Error Rate (%)	Throughput	QoS_c	QoS_s
2 (Baseline)	Stochastic	Static	Unlimited	None	–	–	–	–	–	–
3	Stochastic	Static	Unlimited	None	–	–	–	–	–	–
			Unlimited	Modulated	–	–	–	–	–	–
			Limited	None	–	–	–	–	–	–
		Brownian	Unlimited	None	–	–	–	–	–	–
			Unlimited	Modulated	–	–	–	–	–	–
			Limited	None	–	–	–	–	–	–
	comm-aware	Static	Unlimited	None	–	–	–	–	–	–
			Limited	None	–	–	–	–	–	–
		Brownian	Unlimited	None	–	–	–	–	–	–
			Limited	None	–	–	–	–	–	–
8	Stochastic	Static	Unlimited	None	–	–	–	–	–	–
			Unlimited	Modulated	–	–	–	–	–	–
			Limited	None	–	–	–	–	–	–
		Brownian	Unlimited	None	–	–	–	–	–	–
			Unlimited	Modulated	–	–	–	–	–	–
			Limited	None	–	–	–	–	–	–
	comm-aware	Static	Unlimited	None	–	–	–	–	–	–
			Limited	None	–	–	–	–	–	–
		Brownian	Unlimited	None	–	–	–	–	–	–
			Limited	None	–	–	–	–	–	–
13	Stochastic	Static	Unlimited	None	–	–	–	–	–	–
			Unlimited	Modulated	–	–	–	–	–	–
			Limited	None	–	–	–	–	–	–
		Brownian	Unlimited	None	–	–	–	–	–	–
			Unlimited	Modulated	–	–	–	–	–	–
			Limited	None	–	–	–	–	–	–
	comm-aware	Static	Unlimited	None	–	–	–	–	–	–
			Limited	None	–	–	–	–	–	–
		Brownian	Unlimited	None	–	–	–	–	–	–
			Limited	None	–	–	–	–	–	–

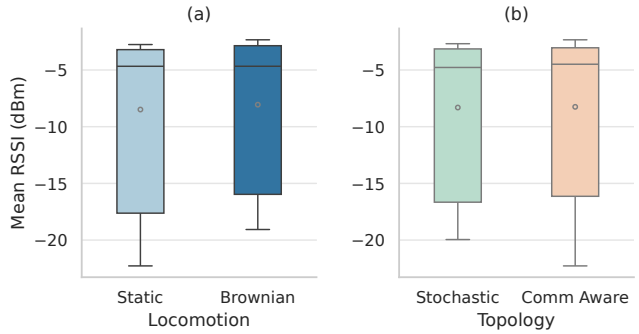

Fig. 9. (a) Mean RSSI values across different locomotion modes, (b) mean RSSI values accross different topology inference schemes

raising contention. *Comm-aware* limits link formation, avoiding burst traffic and yielding lower median latency and jitter at throughput cost. Time-snapshots of inferred topologies in Fig. 13 illustrate this mechanism, with *stochastic* creating denser transient link sets than *comm-aware* over comparable intervals. In this elitist island-model setup, slower mixing between islands can be useful. Limiting link formation in *comm-aware* reduces premature convergence, allowing for a more diverse genome to persist and be explored longer. This aligns with the observation that overall fitness improves primarily with agent count (Fig. 6a), while the fastest adaptation rates arise under the *stochastic* scheme where information diffuses quickly (Fig. 11b).

Comm-aware attains consistently high QoS values by suppressing jitter and keeping low latencies, while *stochastic* achieves higher raw throughput but with higher error rates as agent density grows. Kernel-density plots of QoS_c against QoS_cs in Fig. 12 show *comm-aware* achieving an overall improved service relative to the overall distribution, whereas frequency modulated *stochastic* shifts the distribution toward QoS_s. Message limits show no clear effect.

Error rate increases with swarm size, consistent with higher collision probability (Fig. 8a-b). This effect is independent of locomotion, and becomes most pronounced under *stochastic* at 13-robot swarms. RSSI does not identify topology schemes in this arena, which means that the *comm-aware* scheme is effectively ranking peers by latency. Brownian motion yields slightly lower RSSI than static (Fig. 9a-b), plausibly by incidental repositioning into stronger links. Verifying this, would require sparser layouts in larger arenas.

Introducing frequency modulation under the *stochastic* scheme smooths latency over time (Fig. 10b) without impacting throughput, and preserves fast adaptation rates relative to *comm-aware*. Validating the findings of [11], the

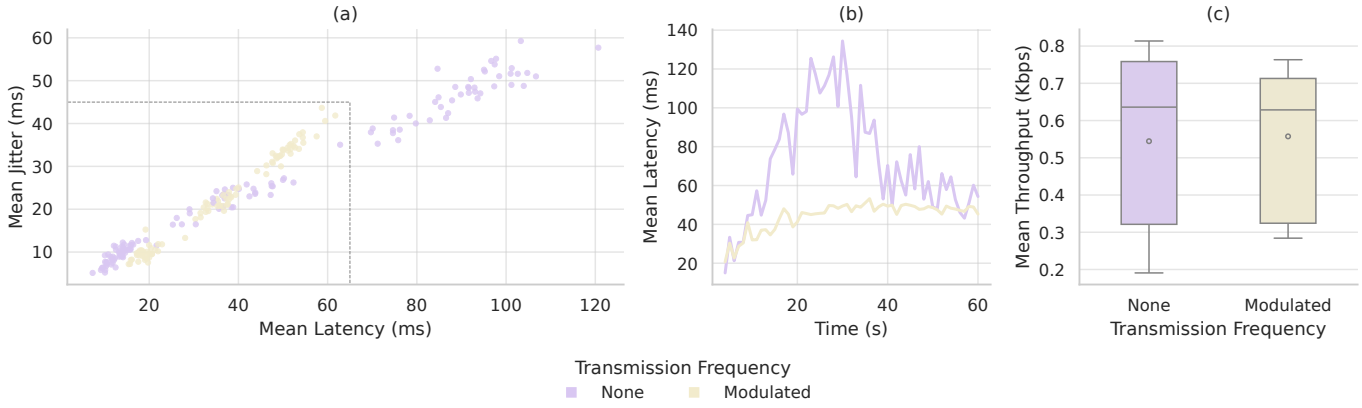

Fig. 10. (a) Latency and Jitter relationship under varying transmission policies, (b) latency across time by transmission setting, (c) throughput distribution by transmission setting

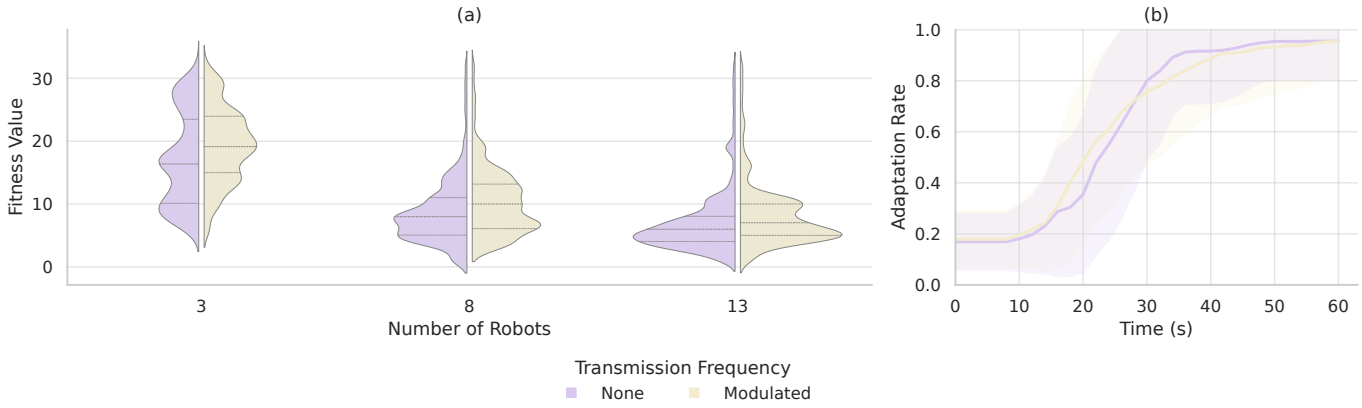

Fig. 11. (a) Mean device fitness values across different swarm sizes, (b) mean cumulative percentage of agents converging to the lowest fitness value across $2s$ windows

swarm can tolerate transmission delays and attain similar fitness scores compared to having no modulation. For tasks where rapid environmental updates matter such as search and rescue [8], this policy provides a low complexity alternative to flooding.

Using the fixed genome structure (Table II) in a single-channel setup, we observed no detectable effects of mean swarm throughput on GA performance. However, we have not tested whether alternative payload schemas (e.g., partial-genome transfer or non-elitist migration) could change this relationship. The generality beyond the ESP-NOW 250-byte packet limit also remains unverified, larger messages that require fragmentation would likely complicate asynchronous processing, requiring deeper local buffers to avoid packet loss, and also increase compute overhead.

Energy consumption matters. Broadcast-heavy policies raise CPU utilisation, whereas link-constraining policies lower processing load (Fig. 14). Energy budgets therefore favour selective link formation and moderated transmissions. For context, ESP-NOW transmit currents are typically an order of magnitude higher than Bluetooth, so depending on the application swarms may need to prioritise partial link formation over fast information diffusion. This trade-off is not abstract, in the author's current role at Unilode Aviation Solutions (the operator of the largest globally distributed aviation IoT network), battery longevity outranks other QoS metrics due to the economics of battery replacement.

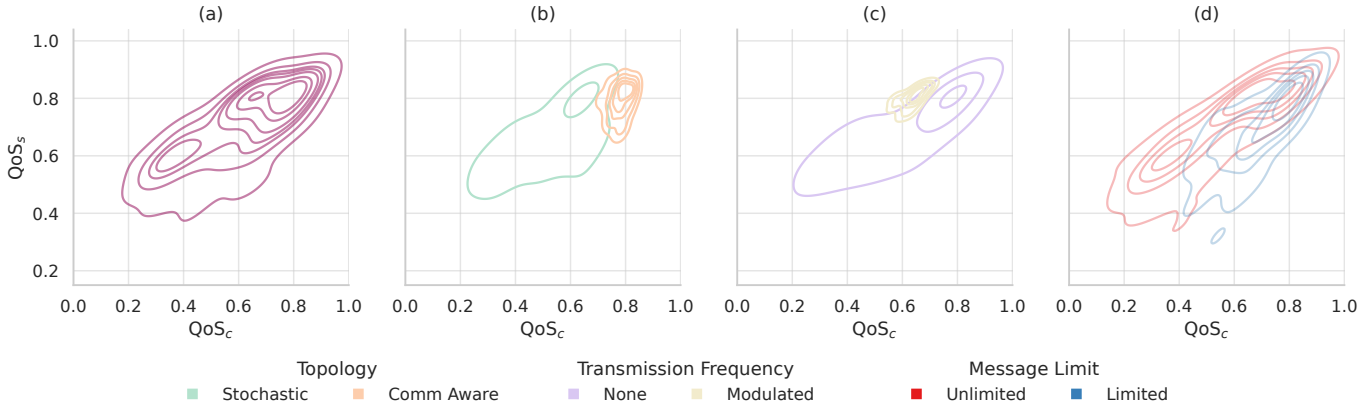

Fig. 12. Kernel-density contours showing the joint distribution of two QoS metrics (x: QoS_c, y: QoS_s) for a 13-agent swarm. (a) pooled across all experimental factors, (b) split by communication topology, (c) split by transmission frequency policy, (d) split by message limit setting

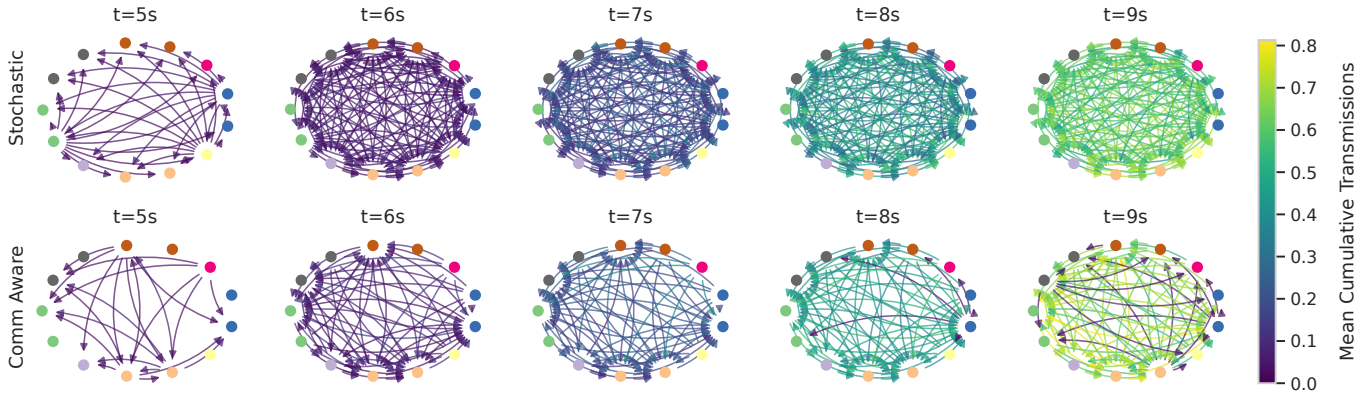

Fig. 13. Temporal snapshots of inferred communication topologies for a 13-agent swarm. Rows: inference scheme, columns: time samples. Nodes are on a fixed circular layout, edges show inferred links and its colour indicates mean cumulative transmissions per node.

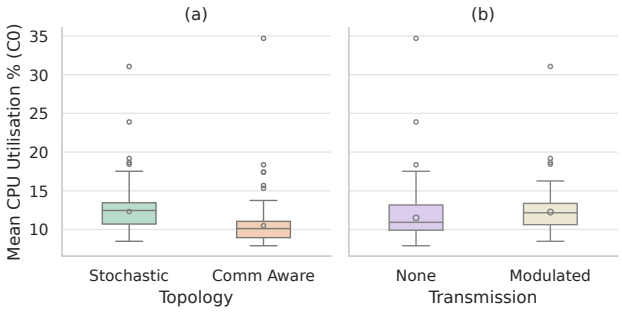

Fig. 14. Core 0 utilisation by communication policy, excluding message limited experiments

Answers to the research questions: **RQ1**: Link-aware peer ordering reduces latency and jitter by constraining link formation, at a small throughput cost (Fig. 5a-c; Fig. 13). **RQ2**: *Stochastic* ordering accelerates diffusion and earlier convergence, larger swarms achieve lower mean fitness throughout (Fig. 6a-b). **RQ3**: Latency-bounded random micro-delays smooth latency with negligible throughput loss and preserve adaptation rates (Fig. 10b-c; Fig. 11b).

A. *Limitations*

Time resolution is bounded by device RTC drift (roughly $\pm 1s$), so message timestamps are only loosely synchronised. Over time results should therefore be interpreted with that tolerance in mind. A second limitation is the environmental variability. Although all trials ran in the same room, they spanned several weeks, and ambient Wi-Fi activity may have varied. We set the ESP-NOW link to channel 6 to avoid nearby access points, but environmental noise cannot be entirely ruled out.

VII. Conclusion

This work profiled peer-to-peer communication for robot swarms on ESP32 hardware. Across 3 to 13 robot swarms, we found that link formation and transmission timing, drive QoS and adaptation rates on this platform, with throughput volume being secondary. Link formation schemes based on latency social evidence, reduced adaptation rates and information diffusion but did not prevent convergence. Conversely, introducing a small, randomized per-packet transmission delay bounded by the recent peer latency, stabilised network latency without materially impacting throughput.

These findings translate into pragmatic design rules. When rapid network-wide diffusion and high throughput are the priority (e.g., environmental updates or consensus), prioritise stochastic scheduling with randomized transmission micro-delays. When preventing premature convergence and achieving low latencies is of concern, favor limiting effective link formation that will lower contention and support diversity in evolutionary search. This study supports the *"less-is-more"* effects observed in prior swarm studies, where well-scoped constraints can be a feature in decentralized decision making.

A. Future Work

Future work will stress-test larger and dynamic payloads to probe the limits of our conclusions. We aim to explore how these findings translate to other hardware platforms and combination of protocols, such as IR and Bluetooth. By leveraging the OTA framework, it could also be possible to explore the meta-evolution (e.g., *alpha-evolve*) of the communication policies at the firmware level, enabling task-conditioned tuning for applications in long-term swarm deployments where energy efficiency is critical.

References

[1] Heiko Hamann. *Swarm Robotics: A Formal Approach*. Cham: Springer International Publishing, 2018.

[2] Marco Dorigo et al. "Ant algorithms and stigmergy". In: *Future Generation Computer Systems* 16.9 (June 1, 2000). MAG ID: 2151758915, pp. 851–871.

[3] Tan Jian Ding et al. "Advancements and Challenges of Information Integration in Swarm Robotics". In: *2023 IEEE International Conference on Cybernetics and Intelligent Systems (CIS) and IEEE Conference on Robotics, Automation and Mechatronics (RAM)*. 2023 IEEE International Conference on Cybernetics and Intelligent Systems (CIS) and IEEE Conference on Robotics, Automation and Mechatronics (RAM). ISSN: 2326-8239. June 2023, pp. 89–95.

[4] Xing An et al. "Multi-Robot Systems and Cooperative Object Transport: Communications, Platforms, and Challenges". In: *IEEE Open Journal of the Computer Society* 4 (2023). Conference Name: IEEE Open Journal of the Computer Society, pp. 23–36.

[5] Motoaki Hiraga et al. "When Less Is More in Embodied Evolution: Robotic Swarms Have Better Evolvability with Constrained Communication". In: *Journal of Robotics and Mechatronics* 35.4 (Aug. 20, 2023), pp. 988–996.

[6] M. Ruciński, D. Izzo, and F. Biscani. "On the impact of the migration topology on the Island Model". In: *Parallel Computing*. Parallel Architectures and Bioinspired Algorithms 36.10 (Oct. 1, 2010), pp. 555–571.

[7] Sabine Hauert, Jean-Christophe Zufferey, and Dario Floreano. "Evolved swarming without positioning information: an application in aerial communication relay". In: *Autonomous Robots* 26.1 (Jan. 1, 2009), pp. 21–32.

[8] Dimitri Perrin and Hiroyuki Ohsaki. "Decentralised Communication in Autonomous Agent Swarms". In: *2012 26th International Conference on Advanced Information Networking and Applications Workshops*. 2012 26th International Conference on Advanced Information Networking and Applications Workshops. Mar. 2012, pp. 1143–1146.

[9] Mohamed S. Talamali et al. "When less is more: Robot swarms adapt better to changes with constrained communication". In: *Science Robotics* 6.56 (July 28, 2021). Publisher: American Association for the Advancement of Science, eabf1416.

[10] Till Aust et al. "The Hidden Benefits of Limited Communication and Slow Sensing in Collective Monitoring of Dynamic Environments". In: *Swarm Intelligence*. Ed. by Marco Dorigo et al. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2022, pp. 234–247.

[11] Konstantinos I. Tsianos and Michael G. Rabbat. *The Impact of Communication Delays on Distributed Consensus Algorithms*. July 24, 2012.

[12] Stefan M. Trenkwalder et al. "SwarmCom: an infra-red-based mobile ad-hoc network for severely constrained robots". In: *Autonomous Robots* 44.1 (Jan. 1, 2020), pp. 93–114.

[13] Mahmoud Almostafa Rabbah et al. "Real Time Distributed and Decentralized Peer-to-Peer Protocol for Swarm Robots". In: *International Journal of Advanced Computer Science and Applications (IJACSA)* 12.11 (Jan. 30, 2021). Number: 11 Publisher: The Science and Information (SAI) Organization Limited.

[14] T. Krink and R. Thomsen. "Self-organized criticality and mass extinction in evolutionary algorithms". In: *Proceedings of the 2001 Congress on Evolutionary Computation (IEEE Cat. No.01TH8546)*. 2001 Congress on Evolutionary Computation. Vol. 2. Seoul, South Korea: IEEE, 2001, pp. 1155–1161.

[15] Vivek Shankar Varadharajan et al. "Over-the-Air Updates for Robotic Swarms". In: *IEEE Software* 35.2 (Mar. 2018), pp. 44–50.

[16] Ronald Aylmer Fisher and Frank Yates. *Statistical tables for biological, agricultural, and medical research*. 6th ed., rev. and enl. OCLC: 550706. New York: Hafner Pub. Co, 1963.

VIII. Appendix

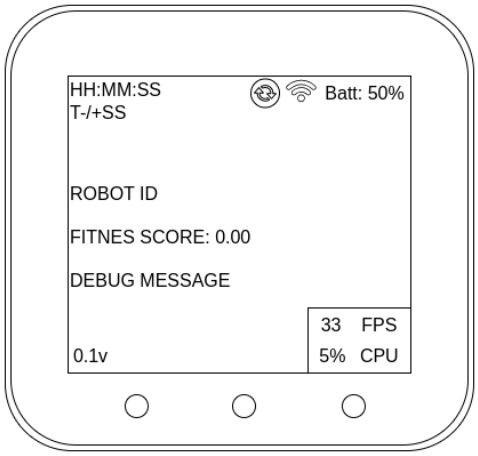

Fig. 15. On-board debug/status display