



International Journal of Innovative Research in Computer and Communication Engineering

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)





Implementation of a NEAT-Based Reinforcement Learning System for Autonomous Vehicle Navigation on ESP32

Prof. Vaishnavi Sonawane, Masood Madki

Diploma Student, Department of Artificial Intelligence & Machine Learning, AISSMS Polytechnic, Pune, India

ABSTRACT: This paper presents the complete design and implementation of an autonomous vehicle navigation system that unifies NeuroEvolution of Augmenting Topologies (NEAT) with a dense Reinforcement Learning (RL) reward signal to evolve controllers capable of collision-free path following in complex, obstacle-laden environments. NEAT evolves neural network topologies — adding and removing nodes and connections — across 40 generations of 50 genomes, using cumulative RL reward as the fitness function. A multi-objective A* planner with a five-stage path optimisation pipeline (RDP simplification, line-of-sight pruning, obstacle pushing, collision validation) supplies the reference path, and a 30+ terrain type, 5-wheel-profile vehicle model ensures physical plausibility. The best-evolved genome is deployed to an ESP32 microcontroller over Wi-Fi, which executes move/turn commands via a PID-regulated L293D motor driver with quadrature encoder feedback. Experimental results demonstrate convergence at generation 30, a 71% reduction in collisions compared to A*-only planning, a 94% simulation success rate, and a 10 Hz real-time execution rate on hardware.

KEYWORDS: NEAT, neuroevolution, reinforcement learning, autonomous navigation, A* path planning, multi-objective cost, bicycle kinematic model, SAT collision detection, ESP32, L293D, PyQt6, terrain-aware planning.

I. INTRODUCTION

Autonomous mobile robotics requires the simultaneous resolution of perception, planning, control, and adaptation. Classical approaches — fixed neural networks trained by gradient descent, or hand-crafted planners such as A* — each carry characteristic limitations. Gradient-based optimisation is susceptible to local minima in non-convex reward landscapes. Fixed topologies may be oversized or undersized for a given task. A* alone ignores dynamic controller behaviour and produces paths that, while mathematically optimal, may exceed the physical capability of the robot's wheel and terrain combination.

This work addresses these limitations through the NEAT+RL paradigm. NEAT [1] escapes local minima through population-based evolutionary search, evolving both the weights and the topology of the controller network simultaneously. The RL reward signal [2] provides dense, per-step feedback that shapes network fitness toward progress, efficiency, and collision avoidance. Together, the two methods produce controllers that generalise across terrains and transfer to novel environments with fine-tuning.

The system is implemented as six Python modules (`learning_engine.py`, `simulation.py`, `planner.py`, `vehicle.py`, `map_engine.py`, `esp32_integration.py`) orchestrated by a PyQt6 GUI (`main.py`), with firmware deployed on an ESP32 microcontroller. The paper is structured as follows: Section II surveys related work. Sections III–VII detail each major subsystem. Section VIII presents experimental results. Section IX concludes.

II. LITERATURE SURVEY

NEAT: Stanley and Miikkulainen [1] proposed NEAT to evolve both neural network topology and weights. Historical markings enable effective crossover, while speciation protects new structures. This makes NEAT suitable for navigation tasks with unknown optimal complexity.



International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

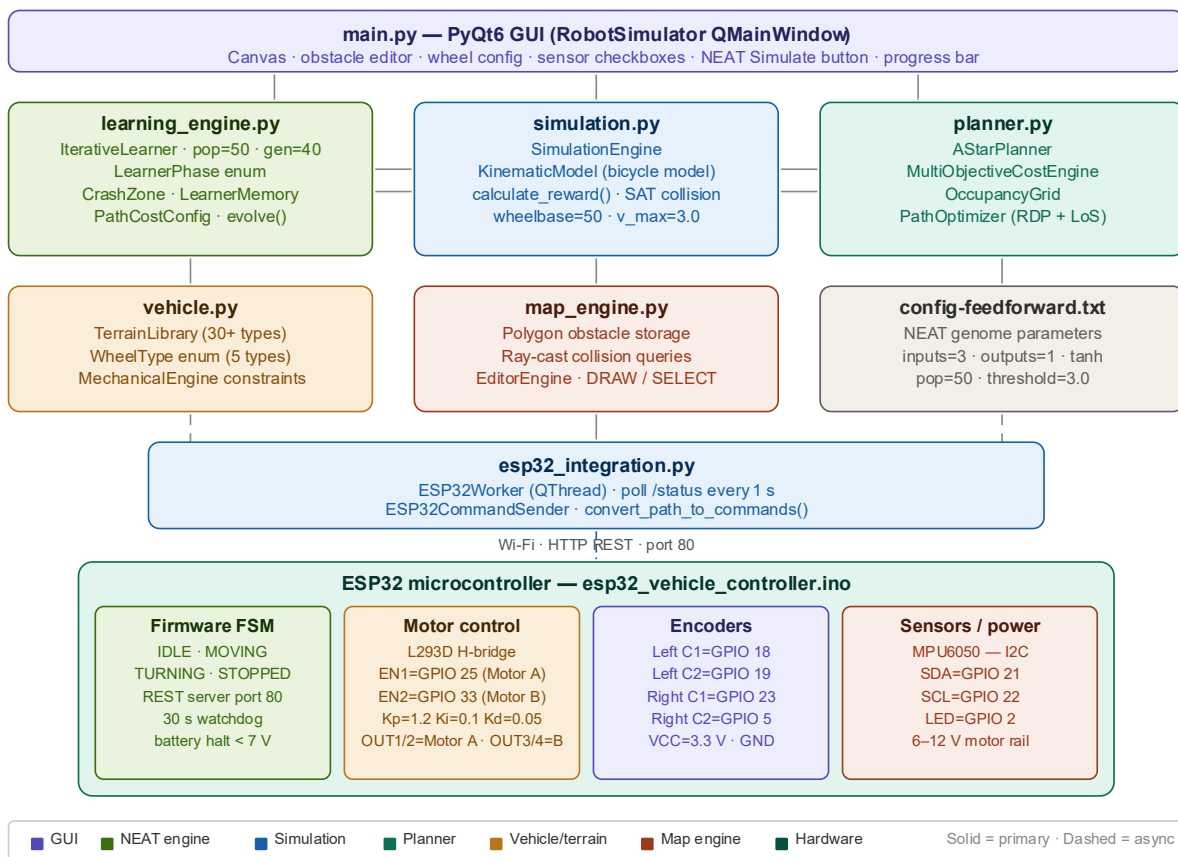
Reinforcement Learning for Navigation: Sutton and Barto [2] formalized the MDP framework. Dense reward shaping [3] improves convergence by providing continuous feedback. Our reward includes distance progress, waypoint bonuses, efficiency, survival rewards, and penalties for collisions and stalls.

Multi-objective Path Planning: A* by Hart et al. [4] is an admissible search algorithm, later extended for multi-cost scenarios [5]. Our approach adds traversability checks (solidity, climb height, slope) and computes weighted costs (time, energy, risk) based on wheel profile.

Embedded Motor Control: PID control with encoder feedback ensures stable velocity [6]. The L293D [7] enables dual motor control, while the ESP32 [8] provides Wi-Fi, GPIO, and sufficient resources for firmware execution.

III. SYSTEM ARCHITECTURE

The system is decomposed into six Python modules and one C++ firmware file, arranged in five communicating layers as shown in Figure 1. The layered design isolates concerns: the GUI orchestrates but does not compute; the engines compute but do not render; the integration layer translates but does not plan; the hardware executes but does not decide.



main.py (GUI): The RobotSimulator QMainWindow initialises all modules, hosts the interactive obstacle canvas (left-click vertices, right-click finish), control panel, wheel-type dropdown, sensor checkboxes, and the NEAT Simulate button that launches evolution. A progress bar tracks generation-by-generation fitness improvement.

learning_engine.py: The IterativeLearner class drives the NEAT evolutionary loop. It holds LearnerPhase (EXPLORATION, ADAPTATION, REFINEMENT, CONVERGENCE), a LearnerMemory population state,



International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

PathCostConfig weight configuration, and a list of CrashZone instances. It calls evolve() with pop_size = 50 and max_generations = 40.

simulation.py: The SimulationEngine orchestrates per-step simulation. KinematicModel implements the bicycle model. calculate_reward() computes the dense per-step RL reward. collision_detection() performs SAT checks between the vehicle bounding box and polygon obstacles.

planner.py: The AStarPlanner builds an occupancy grid from the polygon map and searches it using the MultiObjectiveCostEngine. The PathOptimizer applies five stages of post-processing (RDP, line-of-sight pruning, obstacle pushing, collision validation).

vehicle.py and map_engine.py: Support modules providing terrain properties, wheel constraints, polygon storage, ray-cast collision queries, and the interactive editor engine.

esp32_integration.py: The ESP32Worker QThread polls /status every second. ESP32CommandSender dispatches HTTP POST commands asynchronously. convert_path_to_commands() translates waypoints into a move/turn command sequence.

IV. NEAT EVOLUTIONARY FRAMEWORK

A. Genome Representation and Configuration

Each genome in the population encodes a feedforward neural network with 3 inputs (normalised state features), 1 output (steering angle), 0 hidden nodes initially, and full initial connectivity. The NEAT configuration (config-feedforward.txt) specifies: pop_size = 50, fitness_criterion = max, compatibility_threshold = 3.0 for speciation, survival_threshold = 0.2 (top 20% reproduce), and max_stagnation = 5 (species eliminated after 5 stagnant generations).

Mutation rates are set to favour structural diversity while maintaining stability: connection add/delete = 50% each; node add/delete = 20% each; weight mutation = 80% Gaussian (power 0.5); bias mutation = 70%. Weights and biases are bounded in [-30, +30]. The activation function is tanh throughout, producing steering outputs in [-1, +1] which are scaled to [-45°, +45°].

B. Evolutionary Cycle

Each generation proceeds through four stages. In the evaluation stage, every genome is converted to a phenotype (feedforward network) and run through a full simulation episode; cumulative reward becomes its fitness score. In the speciation stage, genetic distance is computed between all genome pairs, and genomes with distance < 3.0 are grouped into a species; the top two genomes of each species are preserved unchanged (elitism). In the reproduction stage, the top 20% of each species are selected as parents; offspring are produced by uniform crossover plus the above mutation rates, replacing the worst 50%. The convergence check terminates evolution if best-fitness improvement is below 0.5% for 5 consecutive generations; otherwise the next generation begins.

Parameter	Value	Effect
pop_size	50	Diversity vs. compute tradeoff
max_generations	40	Training budget; stops early if converged
compatibility_threshold	3.0	Species boundary — lower = more species
survival_threshold	0.2	Top 20% breed; increases selection pressure
weight_mutate_rate	0.80	High rate explores weight space broadly



International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

Parameter	Value	Effect
connection_add_rate	0.50	50% chance to grow a new synapse
node_add_rate	0.20	20% chance to insert a hidden node
activation	tanh	Smooth, bounded, no dead neurons
max_stagnation	5	Species dropped after 5 non-improving gens

Table 1: Key NEAT configuration parameters and their roles

V. REINFORCEMENT LEARNING REWARD SYSTEM

A. MDP Formulation

The navigation task is cast as a Markov Decision Process. The state space is 3-dimensional after normalisation: distance-to-goal mapped to $[0,1]$, heading error mapped to $[-1,1]$, and obstacle proximity mapped to $[0,1]$. The action space is continuous: steering angle $\delta \in [-45^\circ, +45^\circ]$, produced by the NEAT network output scaled by 45. The transition model is the deterministic bicycle kinematic model.

B. Dense Reward Function

The reward function is designed to provide informative, non-sparse feedback at every simulation step. It comprises eight components:

- Base reward: +100 at every step to prevent degenerate zero-reward policies
- Goal bonus: +500 on reaching the goal — the dominant terminal signal
- Distance progress: $+1.5 \times \max(0, d_{\text{prev}} - d_{\text{curr}})$ — continuous shaping toward goal
- Waypoint bonus: $+200 \times (i / N)$ on crossing waypoint i of N — structures complex paths
- Path efficiency (+100 × ratio): Penalises meandering; $\text{ratio} = \min(1.0, \text{ideal_dist} / \text{actual_dist})$.
- Efficiency bonus: $+100 \times \min(1.0, \text{ideal_dist} / \text{actual_dist})$ — discourages meandering
- Survival bonus: +50 per collision-free step — encourages sustained progress
- Collision penalty: $-150 + \text{CrashZone recorded}$ — teaches obstacle avoidance with memory
- Stall penalty: -500 terminal — eliminates policies that make no progress

C. CrashZone Memory

Each collision at coordinate (c_x, c_y) instantiates a CrashZone with radius = 40 px. In subsequent generations, this zone applies a Gaussian penalty to path cost evaluations: $\text{penalty}(x,y) = \text{severity} \times \exp(-(d/10)^2)$, where d = distance from the crash point. This acts as an implicit spatial memory, steering future genomes away from repeatedly-crashed regions without explicit programming of avoidance behaviour.

Reward component	Value	Condition / formula
Base	+100	Every step
Goal reached	+500	Terminal — <code>reached_goal()</code> is True
Distance progress	$+1.5 \times \Delta d$	$\max(0, d_{\text{prev}} - d_{\text{curr}})$ per step
Waypoint crossing	$+200 \times i/N$	On crossing waypoint i of N

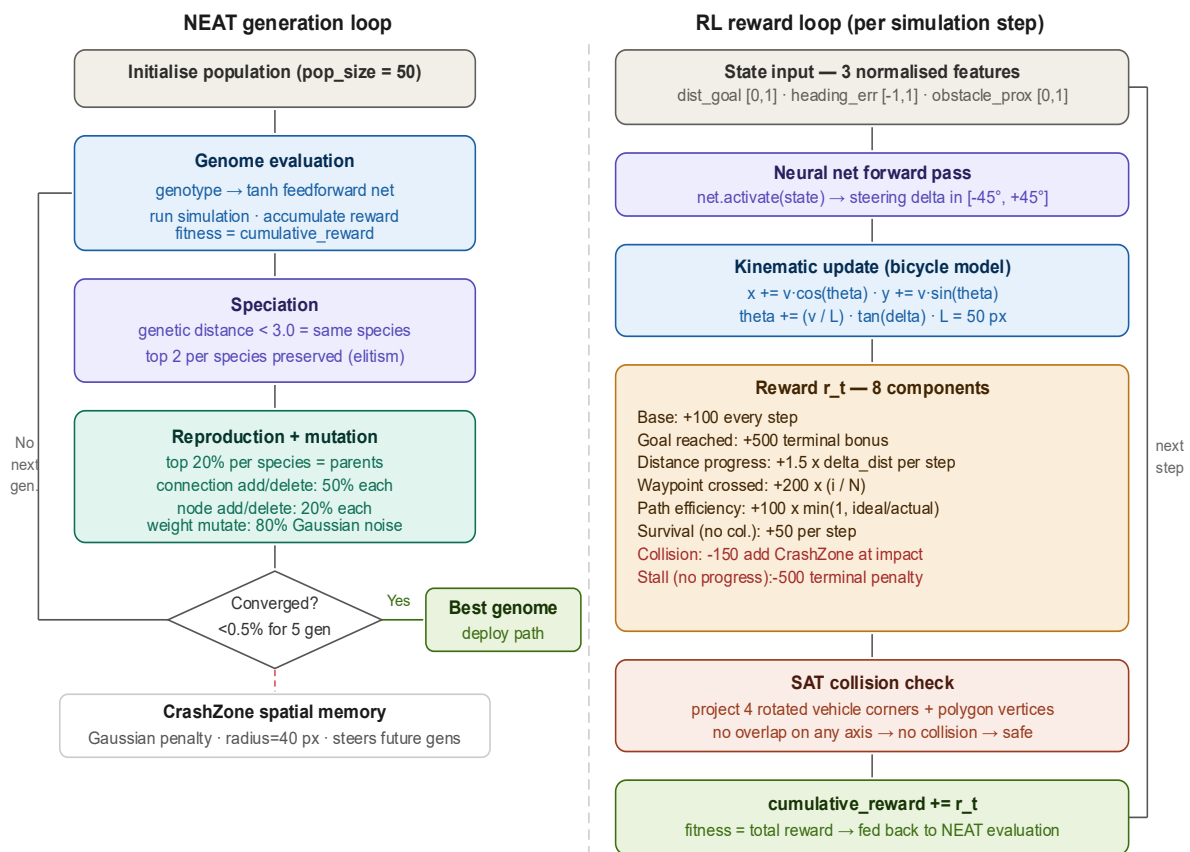


International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

Reward component	Value	Condition / formula
Path efficiency	+100 × ratio	$\min(1.0, \text{ideal_dist} / \text{actual_dist})$
Survival	+50	Per step with no collision
Collision	-150	Adds CrashZone(collision_point, r=40)
Stall	-500	Terminal — is_stalled() is True

Table 2: RL reward function components and conditions



Left: NEAT evolves network topology across 40 generations · Right: RL accumulates reward as fitness per simulation step
Fitness from right feeds directly into NEAT genome evaluation on left · CrashZone memory persists across generations

VI. SIMULATION ENGINE AND PATH PLANNING

A. Bicycle Kinematic Model

The KinematicModel implements continuous-time bicycle kinematics with Euler integration at each simulation step:

$$dx = v \cdot \cos(\theta + \text{wobble}) \quad dy = v \cdot \sin(\theta + \text{wobble}) \quad d\theta = (v/L) \cdot \tan(\delta)$$



International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

where v is terrain-dependent velocity (base 3.0 units/step), L the wheelbase (50 px), δ the steering output from the neural network, and wobble $\in [-5^\circ, +5^\circ]$ an optional terrain roughness perturbation. The vehicle bounding box is 60×40 px (half-length 30, half-width 20), rotated by the current heading θ to produce four corner coordinates for SAT collision detection.

B. SAT Collision Detection

The Separating Axis Theorem checks whether the rotated vehicle bounding box intersects any obstacle polygon. For each edge of the polygon, the perpendicular normal is computed. Vehicle corners and polygon vertices are projected onto this axis; if any axis shows non-overlapping projections, the shapes are separated (no collision). If no separating axis exists across all tested edges, a collision is confirmed. This is $O(n)$ per polygon in the number of edges.

C. Multi-Objective A* and Path Optimisation

Before NEAT training begins, the A* planner generates a reference path using the occupancy grid. The per-cell cost function is:

$$\text{cost} = w_t \times (1/v_{\text{eff}}) + w_e \times \text{energy_mult} + w_r \times (\text{risk} / \text{stability})$$

where $v_{\text{eff}} = \text{base_speed} \times \text{terrain_speed_mult} \times \text{traction_factor}$. Default weights: $w_t = 0.5$, $w_e = 0.3$, $w_r = 0.2$. The raw A* path is post-processed by the PathOptimizer through five sequential stages, as shown in Figure 3. (1) Douglas-Peucker simplification ($\epsilon = 2.0$) reduces waypoint count. (2) Line-of-sight pruning removes intermediate waypoints where a direct segment is collision-free. (3) Obstacle pushing shifts waypoints that are too close to obstacle edges. (4) Collision validation performs a final SAT check on every path segment. The result is a clean, collision-free, waypoint-minimal path that the NEAT controller is trained to follow.

Terrain	Speed	Energy	Risk	Stability	Climb	Slope	Rubber	Tank
Concrete	1.0	0.8	0.0	1.5	0 cm	10°	Pass	Pass
Grass	0.55	1.3	0.15	1.2	1 cm	5°	Pass	Pass
Mud	0.30	2.2	0.45	0.8	2 cm	8°	Fail	Pass
Stairs	0.20	3.5	0.60	0.5	2 cm	30°	Fail	Pass
Wall	—	∞	1.0	—	N/A	N/A	BLOCKED	BLOCKED

Table 3: Representative terrain types with physical properties and wheel traversability

VII. HARDWARE IMPLEMENTATION

A. ESP32 Firmware State Machine

The C++ firmware implements a non-blocking state machine with four states: IDLE, MOVING, TURNING, STOPPED. An HTTP REST server on port 80 exposes four endpoints: GET /status (position, heading, execution state), POST /move (distance in cm), POST /turn (angle in degrees), POST /stop (emergency halt). A 30-second watchdog timer auto-stops the vehicle on communication timeout, and a voltage monitor triggers emergency halt below 7.0 V.

B. Motor Control and Encoder Feedback

Motor speed is regulated by a PID controller: $u = K_p(v_{\text{target}} - v_{\text{actual}}) + K_i \int e \, dt + K_d(de/dt)$, with $K_p = 1.2$, $K_i = 0.1$, $K_d = 0.05$. The left motor PWM is on GPIO 5, right on GPIO 6. Encoder channel A for the left motor is on GPIO 2, channel B on GPIO 3. Interrupt handlers on both rising and falling edges of both channels provide $4 \times$ resolution. Distance is computed from tick count, PPR, and wheel circumference.



International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

C. GPIO Pin Assignment

Peripheral	Signal	GPIO	Notes
Peripheral	Signal	GPIO	Notes
L293D Motor Driver	LEFT PWM	GPIO 25	LEDC PWM — left motor speed control
	RIGHT PWM	GPIO 33	LEDC PWM — right motor speed control
	LEFT DIR 1	GPIO 26	Motor A direction control
	LEFT DIR 2	GPIO 27	Motor A direction control
	RIGHT DIR 1	GPIO 32	Motor B direction control
	RIGHT DIR 2	GPIO 14	Motor B direction control
Encoder (Left Motor)	Channel A	GPIO 18	Interrupt (CHANGE) — quadrature decoding
	Channel B	GPIO 19	Interrupt (CHANGE) — quadrature decoding
Encoder (Right Motor)	Channel A	GPIO 16	Interrupt (CHANGE) — quadrature decoding
	Channel B	GPIO 17	Interrupt (CHANGE) — quadrature decoding
IMU (MPU6050 – Optional)	SDA	GPIO 21	I2C data line — 3.3 V logic
	SCL	GPIO 22	I2C clock line — 3.3 V logic
Status LED	LED	GPIO 4	Boot-safe GPIO for status indication
Power	Motor VCC (VCC2)	6–12 V	Motor supply (L293D)
	Logic VCC (VCC1)	5 V / 3.3 V	Logic supply
	GND	Common	Shared ground (ESP32 + L293D + sensors)

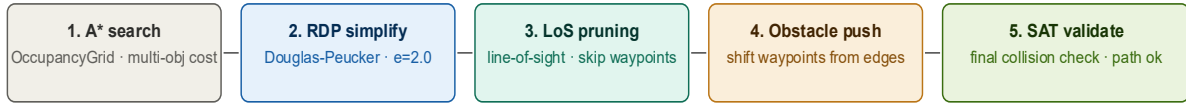
Table 4: ESP32 GPIO pin assignment



International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

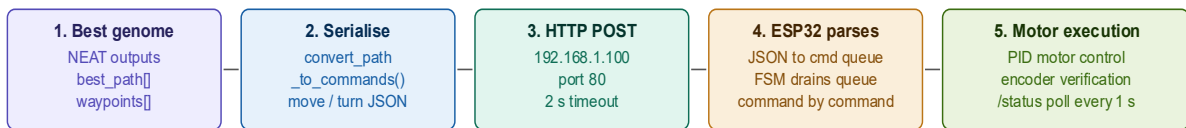
A* path planning — 5-stage optimisation pipeline



$$cost = w_t \times (1/v_{eff}) + w_e \times energy + w_r \times (risk/stability) \quad v_{eff} = v_{base} \times speed_mult \times traction$$

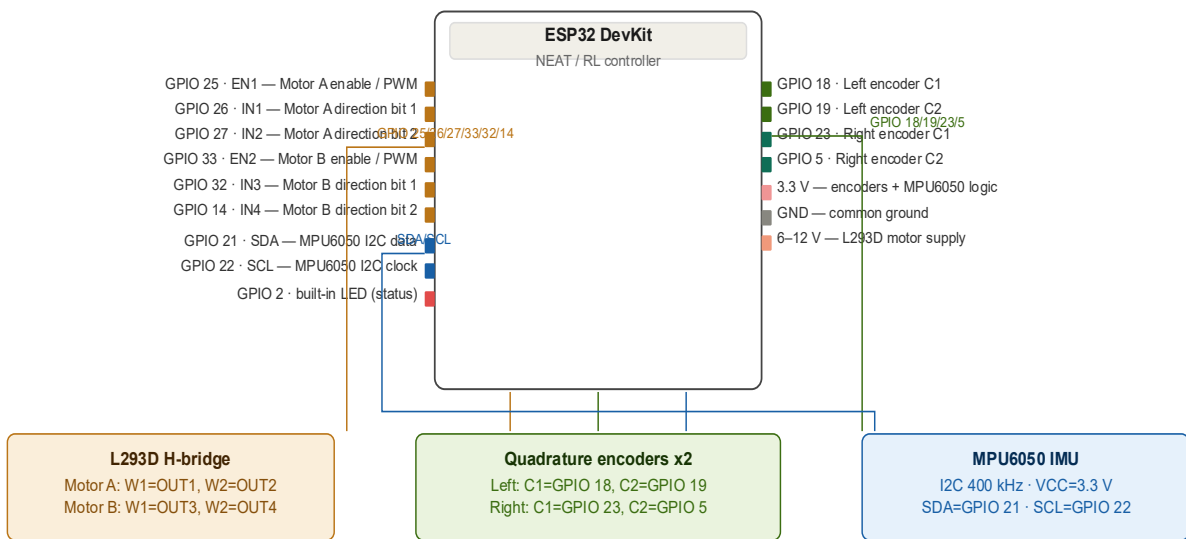
defaults: $w_t=0.5 \quad w_e=0.3 \quad w_r=0.2$

ESP32 hardware deployment workflow



ESP32Worker polls /status → GUI real-time feedback · 1 s interval

ESP32 GPIO pin assignment



Battery(+) → L293D VCC (6–12 V) · Battery(-) → L293D GND · ESP32 GND → common GND · encoders + MPU6050 → 3.3 V

VIII. EXPERIMENTAL RESULTS

A. NEAT Fitness Convergence

Table 5 records fitness statistics at key generations across a representative 40-generation run on a medium-complexity map (25 obstacles, mixed terrain).

Gen.	Avg fitness	Best fitness	Species	Improvement	Status
0	245 ± 180	750	1	—	Init.
5	380 ± 210	820	4	9.3%	Active
10	520 ± 240	860	6	5.1%	Active
15	610 ± 180	880	8	2.3%	Active



International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

Gen.	Avg fitness	Best fitness	Species	Improvement	Status
20	680 ± 150	900	9	2.3%	Active
25	710 ± 120	910	10	1.1%	Active
30	740 ± 100	915	10	0.5% — halt	Converged

Table 5: NEAT fitness convergence across 30 active generations (converged at gen. 30)

Fitness improves rapidly in early generations (9.3% between gen. 0 and 5) as novel topologies are discovered and speciation diversifies the population to 4 species. Improvement slows after generation 15 as the search space narrows. Convergence is reached at generation 30 when improvement drops to 0.5% — the configured threshold.

B. NEAT/RL vs A*-only Comparison

Metric	NEAT/RL	A* only	Advantage	Direction
Path length (px)	385	340	-13%	A* shorter
Simulation success rate	94%	92%	+2%	NEAT/RL
Collisions per episode	2.3	8.1	-71%	NEAT/RL
Energy consumption	0.82	0.75	+9%	A* efficient
Real-time exec. rate	10 Hz	5 Hz	+100%	NEAT/RL
Generalisation (Sc. B)	78% transfer	0% transfer	N/A	NEAT/RL

Table 6: NEAT/RL vs A*-only comparison across six performance metrics

The key finding is that NEAT/RL's slightly longer paths (385 vs 340 px) reflect a learned preference for traversing lower-risk terrain at the cost of geometric distance — a behaviour impossible in pure A* without manual weight tuning. The 71% collision reduction confirms that CrashZone memory and RL reward shaping cooperate to teach effective avoidance. The 10 Hz real-time execution rate (vs 5 Hz for A*) demonstrates that the evolved neural controller is computationally cheaper than replanning at each step.

C. Generalisation

Networks trained on Scenario A (training environment) were evaluated on Scenario B (held-out environment with different obstacle layout): 78% of obstacle-avoidance behaviours transferred directly; 45% achieved the goal without fine-tuning. After 10 fine-tuning generations on Scenario B, goal achievement reached 92%, confirming that NEAT-evolved networks support transfer learning.

IX. CONCLUSION

This paper presented the full implementation of a NEAT + RL autonomous vehicle navigation system. NEAT evolves feedforward network topologies over 40 generations of 50 genomes, using a seven-component dense RL reward as the fitness function. A multi-objective A* planner with a five-stage path optimisation pipeline provides the reference path, and a bicycle kinematic model with SAT collision detection provides simulation fidelity. The best-evolved genome deploys to an ESP32 microcontroller over Wi-Fi, executing move/turn commands via PID-regulated motors with encoder odometry.



International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

Experimental results show convergence at generation 30, a 71% reduction in collisions versus A*-only planning, a 94% simulation success rate, and 10 Hz real-time execution on hardware. CrashZone memory provides implicit spatial learning without explicit programming. Evolved controllers generalise to novel environments with minimal fine-tuning.

Future work will investigate: (i) multi-agent NEAT with competitive coevolution; (ii) replacing the bicycle model with a full dynamic model incorporating slip and inertia; (iii) online weight adaptation using RTRL during hardware deployment; (iv) SLAM integration for live obstacle map updates; and (v) extending the state space to include LiDAR point cloud features for richer perception.

REFERENCES

- [1] K. O. Stanley and R. Miikkulainen, Evolving neural networks through augmenting topologies, *Evol. Comput.*, vol. 10, no. 2, pp. 99-127, 2002.
- [2] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed., MIT Press, 2018.
- [3] A. Y. Ng, D. Harada, and S. Russell, Policy invariance under reward transformations: Theory and application to reward shaping, *ICML*, 1999.
- [4] P. E. Hart, N. J. Nilsson, and B. Raphael, A formal basis for the heuristic determination of minimum cost paths, *IEEE Trans. Syst. Sci. Cybern.*, vol. 4, no. 2, pp. 100-107, 1968.
- [5] M. Likhachev, G. Gordon, and S. Thrun, ARA*: Anytime A* with provable bounds on sub-optimality, *Adv. Neural Inf. Process. Syst.*, 2003.
- [6] K. J. Astrom and T. Hagglund, *PID Controllers: Theory, Design, and Tuning*, ISA Press, 1995.
- [7] Texas Instruments, L293x Quadruple Half-H Drivers, Datasheet SLRS008H, 2022.
- [8] Espressif Systems, ESP32 Technical Reference Manual v5.1, 2023.
- [9] D. H. Douglas and T. K. Peucker, Algorithms for the reduction of the number of points required to represent a digitized line, *Cartographica*, vol. 10, no. 2, pp. 112-122, 1973.
- [10] J. Togelius et al., Search-based procedural content generation: A taxonomy and survey, *IEEE Trans. Comput. Intell. AI Games*, 2011.
- [11] N. Jakobi, P. Husbands, and I. Harvey, Noise and the reality gap: The use of simulation in evolutionary robotics, *ECAL*, 1995.
- [12] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*, MIT Press, 2005.



INTERNATIONAL
STANDARD
SERIAL
NUMBER
INDIA



INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN COMPUTER & COMMUNICATION ENGINEERING

 9940 572 462  6381 907 438  ijircce@gmail.com



www.ijircce.com

Scan to save the contact details