International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 13 Issue: 10 | Oct 2026 www.irjet.net p-ISSN: 2395-0072

Drift-Aware Hybrid Cost Models for Micromouse Speed-Run Planning: A Monte Carlo Study

Saransh Baid[1], Vidit Choudhary[2], Sakshi Shankar Bhongle[3], Chinmay Jain[4], Yashitha K[5]

[1]Student (1BI23RI043), Dept. of Robotics and AI, Bangalore Institute of Technology, Bengaluru, India
[2]Student (1BI23RI060), Dept. of Robotics and AI, Bangalore Institute of Technology, Bengaluru, India
[3]Student (1BI23RI040), Dept. of Robotics and AI, Bangalore Institute of Technology, Bengaluru, India
[4]Student (1BI23RI010), Dept. of Robotics and AI, Bangalore Institute of Technology, Bengaluru, India
[5]Assistant Professor, Dept. of Robotics and AI, Bangalore Institute of Technology, Bengaluru, India

---

Abstract - *Micromouse speed runs are planned with movement-cost tables calibrated on a fully charged robot. As the battery sags and floor friction or motor heating grow, the robot can no longer execute these commands and the run fails. Voltage feedforward corrects only the electrical part of this drift. We propose a hybrid cost model that multiplies voltage feedforward by a per-primitive residual learned with a median estimator from exploration timings, and we analyse when such learning succeeds. In a reproducible C++ Monte Carlo simulator with braided 16-by-16 mazes, a turn-aware Dijkstra planner and non-uniform degradation (958 500 runs), the model is compared with a fixed table, a 30 percent safety margin, voltage compensation with and without a friction allowance, and an oracle. The fixed table and plain voltage compensation fail under friction, and the two margin baselines succeed but are 3 to 27 percent slower than the oracle. The hybrid model matches the oracle's success and its mean time stays within 0.04 percent of the oracle's in all static scenarios. It tolerates outliers and continuous discharge, and it absorbs voltage-sensor gain errors when exploration reveals the full slowdown; when exploration hides friction, a fixed allowance is safer. A closed-form failure model predicts the simulated failure rates and gives design rules for the detection threshold, command margin, estimator, sensor calibration and required exploration transfer.*

Key Words: adaptive motion model, battery degradation, Dijkstra path planning, micromouse, mobile robots, model–reality gap, Monte Carlo simulation, residual learning, timing drift, voltage feedforward

1. INTRODUCTION

In a micromouse contest a small robot explores an unknown 16 × 16 maze, maps it, and then attempts the fastest possible run from the start cell to the central goal [1]. The task combines mapping, path planning and high-speed motion control, and it is a long-standing benchmark for navigation algorithms [2], [3].

Speed-run planners weight straights, turns and pivots by how long they take [4], and the motion profiles that execute them assume a known acceleration capability [5], [6]. Both are tuned on a fully charged robot on a known floor. During a contest this changes. A two-cell lithium-ion pack falls from about 8.4 V when fully charged to below 7 V near the end of its discharge [7], and the steady-state speed of a DC motor scales roughly with supply voltage [8]; floor friction and motor heating add further, non-electrical slowdown. When the controller is asked to execute a profile the motors can no longer track, the robot overshoots or strikes a wall and the run fails. This is an instance of the model–reality gap [9], [10].

The margin for error is small. With a 2% command margin, a fixed table tolerates only about 4% of slowdown (Section 4), well inside the voltage range of a single discharge. The usual engineering response is voltage feedforward: measure the battery with an analogue-to-digital converter and scale the drive commands. It compensates the electrical part of the drift but is blind to friction and heating. Another response is a generous safety margin, which costs time on every run.

This paper asks how a micromouse can keep its cost model valid without paying for a large margin, and when timings measured during exploration are sufficient for that purpose. The contributions are:

1) A **hybrid cost model** that multiplies voltage feedforward by a per-primitive residual learned from exploration timings with a median estimator, together with a purely learned variant, the Adaptive Movement Strategy (AMS), that isolates the value of learning.

2) A **closed-form failure model** that predicts the simulated failure rates and yields explicit rules for the command margin, the detection threshold and the exploration-to-speed-run transfer required.

3) A **lightweight Monte Carlo benchmark**, complementary to physics-based micromouse simulators [11], with braided mazes, separate exploration and speed-run timings, non-uniform degradation, voltage-sensor errors and a common command rule, and an evaluation of 958 500 runs against five baselines, including voltage compensation

with a friction allowance and an oracle, that reports where each approach fails as well as where it succeeds.

2. RELATED WORK

2.1 Maze Solving and Speed-Run Planning

Wall following is simple but does not guarantee reaching a central goal [1]. Flood fill gives shortest paths in cell count and is the standard micromouse planner [2]. Cai et al. added an expected toll for unknown walls to reduce turns [12], and Zawadniak et al. included rotation cost in flood fill [4]. Surveys [3], [13] note that classical planners, including Dijkstra's algorithm [14], assume deterministic execution.

2.2 Motion Control and Voltage Compensation

Speed runs are executed with closed-loop velocity profiles whose accelerations are chosen within the drive's capability [5]; practitioners tune turn profiles carefully because small timing errors cause wall strikes [6]. Voltage feedforward follows from the DC motor model [8], and micromouse builders routinely account for the dependence of motor speed on battery voltage [15]; it compensates supply sag, but not changes in friction or motor temperature. Friction-aware dynamic models can be calibrated from wheel-speed data [16], and a recent model couples battery state with motor and robot dynamics and shows that a lower state of charge degrades trajectory tracking [17].

2.3 Adaptation and Learned Cost Models

Simulation noise and domain randomisation improve transfer to real robots [9], [10]. Online alternatives include adaptive control [18], composite adaptation [19], and recursive identification and Kalman filtering [20], [21]. Closest to this work, traversal times of navigation-graph edges have been estimated from observed traversals [22], edge costs have been discovered during execution [23], and terrain costs have been learned from the robot's own experience [24]. Correcting an analytic model with a learned residual is also established [25].

2.4 Research Gap

Table 1 positions this work. Learning traversal costs from experience and correcting a physical model with a learned residual are not new in themselves. What has not been studied, to our knowledge, is the feasibility of time-optimal speed-run commands under degradation: how fixed tables fail, how voltage feedforward and exploration learning should be combined, how the threshold, margin, estimator and exploration transfer decide success, and how sensitive the result is to voltage-sensor errors.

Table -1: Positioning Against Related Work

Work	Focus	Turn-aware	Adapts costs
[2]	Flood fill, modified flood fill	No	No
[4]	Rotation-aware flood fill	Yes	No
[6]	Turn execution in practice	Yes	Manual tuning
[22]	Learned edge traversal times	–	Yes
[25]	Physics + learned residual	–	Yes[a]
Volt. FF [15]	Supply compensation	–	Voltage only
This work	Learned, hybrid costs	Yes	Yes

[a]Robotic throwing.

3. SYSTEM MODEL AND METHODS

Fig. 1 shows the simulation pipeline. Each trial generates a maze, explores it, returns to the start, lets the strategy under test build its cost model, plans the speed run and executes it.

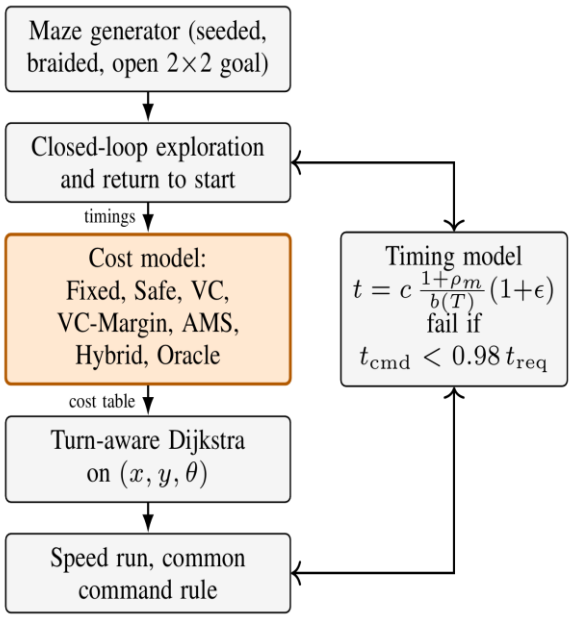


Fig -1: Simulation framework. Only the highlighted block differs between strategies.

3.1 Mazes

Mazes are 16×16, generated by randomised depth-first backtracking from a Mersenne Twister [26] seeded with the maze index. The start cell is closed on three sides and open to the east. As in contest mazes, the 2×2 goal is open inside, and a fraction β of the remaining interior

walls is removed to create loops ($\beta = 0.05$ unless stated), so routes differ in length and number of turns.

3.2 Exploration and Return

The robot starts with an empty map. In each cell it senses the local walls and mirrors them to its neighbours, runs a breadth-first search (BFS) through confirmed-open edges to the nearest unvisited cell, and takes the first step of that path: a forward move or a pivot of 90° or 180°. When every reachable cell is visited, it drives back to the start, as contest rules require, and the speed run begins there, from the heading in which the return ended. The start cell is a dead end, so every speed run opens with a 180° pivot, which counts towards its time and can fail like any other primitive. All exploration times reported below include this return, and its primitives are logged with the others. The explored map equalled the true maze in every trial. Exploration is slow and closed-loop, so it cannot crash; each primitive takes the time it physically needs, and these times are logged.

3.3 Timing and Degradation Model

Four primitives $m \in \{F, R, L, U\}$ are modelled: forward one 180 mm cell, pivot right or left 90°, and pivot 180°, with calibrated speed-run costs c_m^S and exploration costs c_m^E (Table 2). The forward cost corresponds to 1.8 m/s in the speed run and 0.6 m/s during exploration, which represents a faster class of mouse than the N20-based platform of Section 7. Success and relative overheads depend only on time ratios (Section 4), so scaling all costs by a common factor changes absolute times but not these results, except through the elapsed time in the discharge experiment E8.

Two degradation mechanisms are modelled. The speed factor $b(T) = \max(0.3, b_0 - \kappa T)$ is the ratio of the available to the calibrated motor speed, set mainly by battery voltage; it is not the state of charge, and it may fall with elapsed motion time T. The residual ρ_m is a per-primitive slowdown from floor friction or motor heating that a voltage measurement cannot reveal; pivots, which scrub the tyres, are affected more (ρ_P) than straights (ρ_F). The time a speed-run primitive physically requires is

$$t_{req}(m) = c_m^S f_m(T)(1+\epsilon), \qquad f_m(T) = \frac{1+\rho_m}{b(T)}, \tag{1}$$

where $\epsilon \sim \mathcal{N}(0, \sigma^2)$ is drawn independently for every execution.

At exploration speed the controller has spare authority, so only part of the slowdown may be visible. A transfer factor $\chi \in [0,1]$ models this, and an exploration primitive takes

$$t^E(m) = c_m^E[1 + \chi(f_m(T) - 1)](1+\epsilon). \tag{2}$$

With probability q a logged sample is an outlier, multiplied by a factor drawn uniformly from [1.5,3], emulating wheel slip or a bump. VC and the hybrid models read the speed factor through a voltage sensor with gain error g and relative noise σ_v, i.e. $\tilde{b} = b\,g\,(1+\eta)$, $\eta \sim \mathcal{N}(0, \sigma_v^2)$. The main experiments use $\chi = 1$, $q = 0$, $g = 1$ and $\sigma_v = 0$; all four are varied in Section 6.

3.4 Speed Run and Common Command Rule

Each strategy commands primitive m for $t_{cmd} = \hat{c}_m(1+\mu)$, where $\hat{c}_m$ is its own cost estimate and μ a common margin. Executing a motion profile that is too short for the drive is modelled as a crash: if $t_{cmd} < 0.98\,t_{req}$ the run fails, with no retry. Otherwise the primitive takes $\max(t_{cmd}, t_{req})$. This abstracts actuator saturation: a closed-loop profile follower absorbs a small shortfall, not a large one.

3.5 Strategies

Fixed: $\hat{c}_m = c_m^S$. **Fixed-Safe:** $\hat{c}_m = c_m^S$ with margin $\mu_s = 30\%$ instead of μ. **VC (voltage-compensated):** $\hat{c}_m = c_m^S/\tilde{b}(T)$, using the measured speed factor at each command. **VC-Margin:** $\hat{c}_m = c_m^S(1+a_m)/\tilde{b}(T)$ with fixed friction allowances $a_F = 5\%$ and $a_P = 20\%$ for pivots; this is the padded voltage compensation a practitioner would use, and it covers the residuals of S2 and S3 with noise. **AMS (Adaptive Movement Strategy):** learns the ratio $r_m = t^E/c_m^E$ from the logged exploration samples (mean, or median for AMS-Med), and uses $\hat{c}_m = c_m^S \hat{r}_m$ if the drift $\delta = \max_m|\hat{r}_m - 1|$ exceeds a threshold τ, otherwise c_m^S. It follows Algorithm 1 without the voltage readings and the floor at 1, so it isolates the value of learning. **Hybrid / Hybrid-Med (proposed):** learns the voltage-normalised residual $\hat{r}_m^{res} = \max\left(1, \text{agg}\left(t^E\tilde{b}/c_m^E\right)\right)$, where agg is the mean (Hybrid) or the median (Hybrid-Med), and commands $\hat{c}_m = c_m^S \hat{r}_m^{res}/\tilde{b}(T)$ with the current reading, i.e. voltage feedforward times a learned residual (Algorithm 1). The lower bound of 1 reflects that friction and heating cannot make the robot faster. **Oracle (reference):** $\hat{c}_m = c_m^S f_m(T)$; it knows the true degradation and bounds what a strategy using the common margin μ can achieve.

Algorithm 1 Hybrid-Med cost model and speed-run commands (proposed)

Require: exploration samples $\{t^{E}_{m,k}\}$ with voltage readings $\{\tilde{b}_k\}$, costs c^{E}_{m}, c^{S}_{m}, margin μ
Ensure: command time of every speed-run primitive
1: **for** each primitive m with samples **do**
2: $\hat{r}_m \leftarrow \mathrm{median}_k\left(t^{E}_{m,k}\,\tilde{b}_k / c^{E}_{m}\right)$ ▷ voltage-norma
3: **end for**
4: $\hat{r}_m \leftarrow$ mean of the observed $\hat{r}$ for primitives without samples
5: $\hat{r}_m \leftarrow \max(1, \hat{r}_m)$ for all m ▷ residual cannot spe
6: plan with costs $c^{S}_{m}\hat{r}_m/\tilde{b}(T_0)$ ▷ T_0: start o
7: **for** each primitive m of the planned path **do**
8: read $\tilde{b}(T)$; command $t_{cmd} = c^{S}_{m}\hat{r}_m(1+\mu)/\tilde{b}(T)$
9: **end for**

The planner uses each strategy's cost table, so a learned table with non-uniform ratios can change the route. In the main experiments $\tau = 0$ (always use the learned table); Section 6 shows that a positive threshold only adds failures.

3.6 Speed-Run Planning and Computational Cost

Dijkstra's algorithm [14] runs over the state (x, y, θ), $\theta \in \{N, E, S, W\}$, i.e. 1024 states, from all goal states backwards; a state's successors are the forward move and the three pivots, each weighted by the strategy's cost. The robot follows the decreasing time-to-goal. Planning is $O(V \log V)$ once per run. VC-Margin needs no memory; the mean-based variants need four running sums and counters; the median-based variants store the exploration samples, about 750 values or 3 kB per maze, which is small next to the 520 kB SRAM of the RP2350 microcontroller [27].

4. FAILURE CONDITION ANALYSIS

A primitive fails when $\hat{c}_m(1+\mu) < 0.98\, c^{S}_{m} f_m (1+\epsilon)$. Define the effective tolerance $\gamma = (1+\mu)/0.98 - 1$ ($\gamma = 4.1\%$ for $\mu = 2\%$) and the estimate ratio $e_m = \hat{c}_m/(c^{S}_{m} f_m)$. Then a primitive fails if

$$\epsilon > e_m(1+\gamma) - 1. \qquad (3)$$

With $p_m = 1 - \Phi\big((e_m(1+\gamma)-1)/\sigma\big)$, a run of n primitives succeeds with probability $\prod(1-p_m) \approx (1-p)^n$.

Fixed: $e = b/(1+\rho_m)$; without noise it fails as soon as $f_m > 1+\gamma$, e.g. for $b < 0.961$ or $\rho_P > 4.1\%$.

Margin rule: for an unbiased estimate ($e = 1$), success probability P^* over n primitives requires $\gamma \geq z\sigma$ with $z = \Phi^{-1}\big((P^*)^{1/n}\big)$. For $P^* = 99\%$, $z = 3.55$ at the mean path length here ($n = 53$) and $z = 3.78$ for perfect mazes ($n = 127$). Hence $\mu \gtrsim 0.98(1 + z\sigma) - 1 \approx z\sigma - 2\%$.

Threshold rule: AMS keeps the calibrated table whenever $\delta \leq \tau$, so a drift just below τ is uncorrected. To avoid a dead band, τ must lie below the smallest slowdown that already causes failures, about $\gamma - z\sigma \approx 0.5\%$ for $\mu = 2\%$, $\sigma = 1\%$, yet above the estimation noise of $\hat{r}$ (0.05–0.2%, Section 6). This window is so narrow that $\tau = 0$ is the practical choice.

Transfer rule: with $\chi < 1$, AMS learns $\hat{r}_m = 1 + \chi(f_m - 1)$, so $e_m = \hat{r}_m/f_m$. Without noise it fails when $\chi < \chi^* = (f_m/(1+\gamma) - 1)/(f_m - 1)$; for the pivots in scenario S3 ($f = 1.244$), $\chi^* = 0.80$, and noise raises the requirement further. More generally, with noise an estimate with relative shortfall $s = 1 - e_m$ keeps success probability P^* only if $s \lesssim \gamma - z\sigma$. Partial transfer gives $s = (1-\chi)(f_m - 1)/f_m$, so the requirement grows with the slowdown: for the S3 pivots, $\gamma - z\sigma = 0.5\%$ requires $\chi \gtrsim 0.975$. For the hybrid model with an exact voltage reading and a purely electrical slowdown, the learned residual falls below 1 and is clamped, so χ has no effect.

Sensor-gain rule: with gain error g the hybrid model learns $\hat{r}^{res}_m = \max(1, g\,b\,[1 + \chi(f_m - 1)])$ and divides by $g\,b$ at command time. With full transfer g cancels or, for $g < 1$, only lengthens the commands. With partial transfer and a purely electrical slowdown the shortfall is $s = \min\big(1 - 1/g,\ (1-\chi)(1-b)\big)$, the smaller of the over-reading and the AMS shortfall, so over-reading must stay below $\gamma - z\sigma \approx 0.5\%$ unless transfer is nearly complete.

Check: using each maze's path length, the model predicts 0.2%, 34.6%, 94.0% and 99.9% success for Fixed at $b_0 = 0.97$, 0.98, 0.99 and 1.0; the simulation gives 0.2%, 35.2%, 94.6% and 100% (500 trials each, from E4). The model predicts 99.9% for an unbiased estimate; AMS achieved 99.9%.

5. EXPERIMENTAL SETUP

5.1 Implementation and Reproducibility

The simulator is written in C++17 (standard library only, g++ -O2); analysis uses Python with SciPy [28]. All strategies in a trial share one noise stream seeded by maze and repetition (common random numbers). Results are bit-identical for the same compiler and standard library. Source code, run scripts and all results are available at https://github.com/vidit-16/AMS-Micromouse.

Table -2: Simulation Parameters

Parameter	Symbol	Value
Mazes		100 (seeds 0–99)
Repetitions		10 (E1, E9); 5 (sweeps)
Braiding	β	0.05
Speed-run costs F / R,L / U	c_m^S	100 / 180 / 300 ms
Explore costs F / R,L / U	c_m^E	300 / 360 / 540 ms
Execution noise	σ	1%
Margin; Fixed-Safe margin	μ; μ_s	2%; 30%
VC-Margin allowances F / pivots	a_F; a_P	5%; 20%
Crash tolerance		$t_{cmd} \geq 0.98\, t_{req}$
AMS threshold	τ	0
Transfer; outlier rate	χ; q	1; 0
Voltage gain; noise	g; σ_v	1; 0

5.2 Scenarios and Experiments

The main experiment **E1** uses four scenarios: S0 nominal ($b_0 = 1$); S1 voltage sag ($b_0 = 0.85$); S2 friction ($b_0 = 1$, $\rho_F = 3\%$, $\rho_P = 15\%$); S3 combined ($b_0 = 0.9$, $\rho_F = 3\%$, $\rho_P = 12\%$), each with 100 mazes × 10 noise realisations. The sweeps are: **E2** $b_0 = 0.70, 0.72, \ldots, 1.00$; **E3** $\rho_P = 0, 2.5, \ldots, 30\%$ at $b_0 = 1$, $\rho_F = 0$; **E4** $\tau \in \{0, 0.5, 1, 2, 3, 5\}\%$ for $b_0 = 0.95, 0.96, \ldots, 1.00$; **E5** $\sigma = 0, 0.5, \ldots, 3\%$ and $\mu = 0, 1, \ldots, 6\%$ in S1; **E6** $\chi \in \{0.80, 0.85, 0.90, 0.925, 0.95, 0.975, 1\}$ in S3 and $\chi \in \{0, 0.25, 0.5, 0.75, 0.9, 1\}$ in S1; **E7** $q \in \{0, 1, 2, 5, 10, 20\}\%$ in S1 and S3; **E8** $\kappa \in \{0, 0.5, 1, 1.5, 2, 3\} \times 10^{-4}\, s^{-1}$ at $b_0 = 1$, $\rho_F = 0$, with $\rho_P = 0$ and $\rho_P = 10\%$; **E9** $\beta \in \{0, 0.05, 0.10\}$ in S3; **E10** $g - 1 \in \{-6, -4, -2, -1, 0, 1, 2, 3, 4, 6, 8\}\%$ in S1 and S3, and $\sigma_v \in \{0, 0.25, 0.5, 1, 1.5, 2\}\%$ in S1; **E11** $g - 1 \in \{0, 1, 2, 4\}\%$ with $\chi \in \{0.5, 0.75, 0.9, 0.95, 1\}$ in S1. A trial is one maze, noise realisation and parameter setting; a run is one strategy in one trial. All nine strategies run in every trial, giving 106 500 trials and 958 500 runs.

Parameter choices. $b_0 = 0.85$ corresponds to a two-cell pack at about 7.1 V against a calibration at 8.4 V [7]. The residuals in S2 and S3 are representative values inside the range swept in E3 (up to 30%). The Fixed-Safe margin of 30% is the smallest multiple of 5% that covers the largest slowdown in E1 (24.4% for the S3 pivots) plus 3.55σ of noise, which needs 26.3%. The pivot residuals are assumed, not measured; E3 shows that VC already fails for $\rho_P \geq 5\%$, so the need for a friction correction does not hinge on their exact value. The crash tolerance and the margin enter the results only through γ (Section 4), so other values shift the thresholds predictably. The noise level $\sigma = 1\%$ is a representative timing repeatability and is swept in E5.

5.3 Metrics and Statistics

The primary metric is speed-run success, with Wilson 95% confidence intervals (CIs) [29]. Because strategies are evaluated on the same trials, success is compared with McNemar's exact test on paired outcomes [30]. Times of successful runs are compared on paired trials, relative to the oracle, with the Wilcoxon signed-rank test [31].

6. RESULTS

6.1 Main Comparison

Table 3 and Fig. 2 summarise E1. Exploration, including the return to start, took 246 s in S0 (range 215–317 s), and the speed run 7.14 s on average (1.92–22.95 s) over paths of 16 to 171 primitives (mean 53).

Table -3: Main Results (E1, 1000 Runs per Cell): Success and Time Overhead vs. Oracle

Strategy	Success (%)				Time overhead (%)[a]			
	S0	S1	S2	S3	S0	S1	S2	S3
Fixed	99.9	0	0	0	0.0	–	–	–
Fixed-Safe	100	100	100	100	27.4	8.3	16.6	6.5
VC	99.9	99.9	0	0	0.0	0.0	–	–
VC-Margin	100	100	100	100	12.9	12.9	3.3	4.8
AMS	99.9	99.9	99.9	99.9	0.0	0.0	0.0	0.0
AMS-Med	99.9	99.9	99.9	99.9	0.0	0.0	0.0	0.0
Hybrid	99.9	99.9	99.9	99.9	0.0	0.0	0.0	0.0
Hybrid-Med	99.9	99.9	99.9	99.9	0.0	0.0	0.0	0.0
Oracle	99.9	99.9	99.9	99.9	–	–	–	–

[a]Mean over trials where both succeed; values below 0.05% shown as 0.0. 99.9% = 999/1000, 95% CI 99.4–100; 0% has CI 0–0.4.

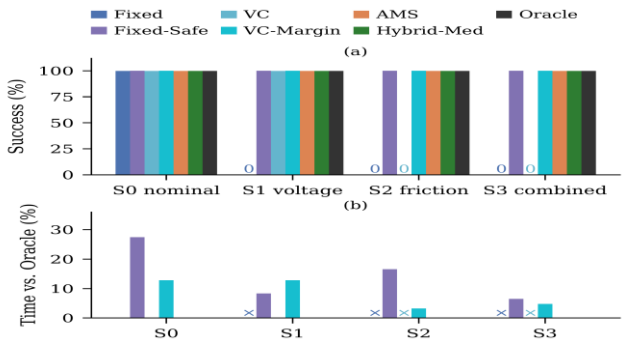


Fig -2: E1: (a) speed-run success in the four scenarios; (b) mean speed-run time relative to the oracle for successful runs (×: no successful run). Hybrid-Med is the proposed model.

The fixed table failed every run in S1–S3. Voltage compensation solved the pure voltage case (S1) but failed every run once friction was present (S2, S3), because pivots needed 12–15% more time than it allowed. The two margin baselines always succeeded. Fixed-Safe was 6.5–27.4% slower than the oracle, most in the nominal case where its margin is unused; VC-Margin was 3.3–12.9% slower, least in S2 and S3 where real friction consumes its allowance. AMS, Hybrid and their median variants succeeded in 999 of 1000 runs in every scenario, with mean speed-run times within 0.04% of the oracle, and VC-Margin was slower than Hybrid-Med in every scenario (by 3.3–12.8% on average, Wilcoxon $p < 10^{-160}$). In S2 and S3, AMS and Hybrid-Med each beat VC and Fixed in 999 paired trials and lost in none (McNemar $p < 10^{-300}$), and succeeded in all ten realisations for 99 of 100 mazes. The single failure, shared by every strategy that uses the 2% margin, including the oracle, is one noise draw above 4σ in maze 71; the two margin baselines survive it, a difference of 1 in 1000 (McNemar $p = 1$).

6.2 Learned-Model Accuracy and Routing

Exploration supplied on average 464 forward, 127 right, 132 left and 26 U-turn samples per maze. The AMS estimate of each primitive's true slowdown had a standard deviation of 0.05%, 0.09%, 0.09% and 0.20%, respectively, with negligible bias.

Route choice matters but rarely changes. With $\beta = 0.05$, the turn-aware planner found a faster route than the cell-count shortest path (flood fill with fewest turns on ties) in 7% of mazes, saving up to 5.8%; with $\beta = 0.10$, in 14% of mazes and up to 10.3%; in perfect mazes ($\beta = 0$) never, as expected (Table 4). The non-uniform learned table changed the planned route in only 2% of S2 trials, with a 0.03% predicted gain. The benefit of learning is therefore almost entirely in command timing, not routing. All adaptive strategies kept 99.7–100% success at every braid level in S3, VC-Margin and Fixed-Safe 100%, while Fixed and VC stayed at 0%.

Table -4: Effect of Maze Braiding (E9, Scenario S3, 1000 Trials Each)

β	Path prims. (mean)	Explore (s) (mean)	Turn-aware faster (%)	Max gain (%)	AMS / Hybrid success (%)
0	126.8	320.1	0	0	99.8 / 99.8
0.05	52.9	292.4	7	5.8	99.9 / 99.9
0.10	38.8	277.0	14	10.3	100 / 100

6.3 Voltage and Friction Sweeps

Fig. 3(a) sweeps the speed factor. Fixed dropped to 35.2% at $b_0 = 0.98$ and to 0% for $b_0 \leq 0.96$. Fixed-Safe failed below its margin, at 98.4% for $b_0 = 0.78$, 0.2% for 0.76 and 0% below. VC, VC-Margin, AMS and Hybrid stayed at 100%. Fig. 3(b) sweeps the pivot residual with $\rho_F = 0$: Fixed and VC (identical, overlapping lines) fell to 32.0% at $\rho_P = 2.5\%$ and to 0% from 5%. VC-Margin held 100% up to its 20% allowance, then fell to 60.0% at 22.5% and 0.2% at 25%, and Fixed-Safe to 65.0% at 30%, whereas AMS and Hybrid stayed at 100% up to 30%.

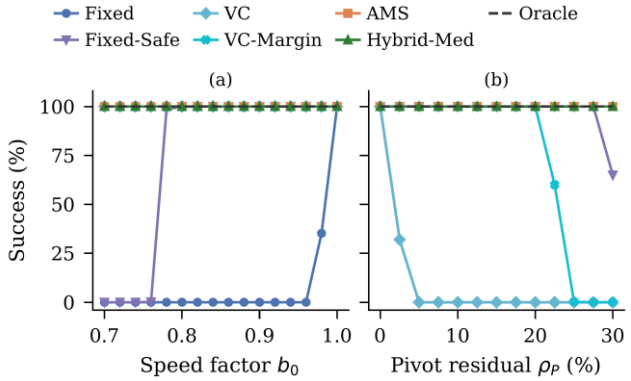


Fig -3: Success against (a) speed factor (E2) and (b) pivot residual at $b_0 = 1$ (E3); 500 trials per point. AMS, Hybrid-Med and Oracle overlap at 100%, with VC-Margin in (a); in (b) Fixed and VC overlap.

6.4 Detection Threshold

Experiment E4 confirms the threshold rule. With $\tau = 0$, AMS always used its learned table and succeeded in 100% of runs at every b_0; at $b_0 = 1$ its runs were 0.001% slower than Fixed (Wilcoxon $p = 0.98$), so always learning costs nothing. Positive thresholds created a dead band, as predicted: success fell to 94.6% at $b_0 = 0.99$ for $\tau = 2\%$, to 35.2% at $b_0 = 0.98$ for $\tau = 3\%$, and to 0% at $b_0 = 0.96$ for $\tau = 5\%$.

6.5 Execution Noise and Margin

Fig. 4 shows AMS success against noise and margin in S1. Near-certain success required $\gamma \approx 3.5$–4σ, in line with the margin rule: at $\sigma = 1\%$, $\mu = 0$ gave 37.6% and $\mu = 2\%$ ($\gamma = 4.1\sigma$) gave 100%; at $\sigma = 2\%$, $\mu = 5\%$ ($\gamma = 3.6\sigma$) gave 99.0%. The oracle matched AMS within 1.0 percentage point in every cell, so these failures belong to the margin, not to learning. The margin costs time: at $\sigma = 0$ the mean run grew from 8.24 s at $\mu = 0$ to 8.73 s at $\mu = 6\%$.

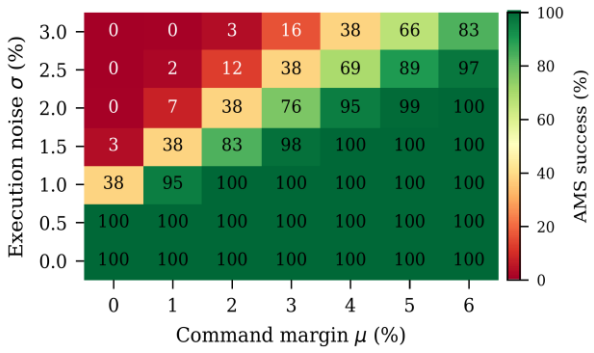


Fig -4: AMS success (%) against execution noise and command margin in S1 (E5, 500 trials per cell).

6.6 Transfer from Exploration and Outliers

Fig. 5(a) varies how much of the slowdown is visible during exploration. AMS and Hybrid-Med needed nearly full transfer: success was 0% at $\chi = 0.80$, about 60% at 0.90, 97% at 0.95 and above 99% at 0.975, consistent with the transfer rule ($\chi \gtrsim 0.975$). Voltage feedforward cannot help in S3 because friction is visible only to the learned residual. VC-Margin, which learns nothing, succeeded at every χ at its 4.8% overhead, so when friction is hidden during exploration a fixed friction allowance is the safer choice. In S1, where the slowdown is purely electrical, the picture reverses: AMS failed for every $\chi \leq 0.75$ and reached only 74.4% at $\chi = 0.9$, whereas Hybrid-Med stayed at 100% for every χ, because its voltage term is measured rather than learned; this holds for an accurate sensor (Section 6.7). Fig. 5(b) adds outliers in S3. Success stayed at 100%, because outliers inflate the estimate, but the mean-based models became slower: 1.2% at q = 1%, 6.3% at 5% (slower than VC-Margin at 4.8%), 12.4% at 10% (slower than Fixed-Safe at 6.5%) and 24.9% at 20%. Hybrid-Med kept the overhead at or below 0.32%; the same holds in S1.

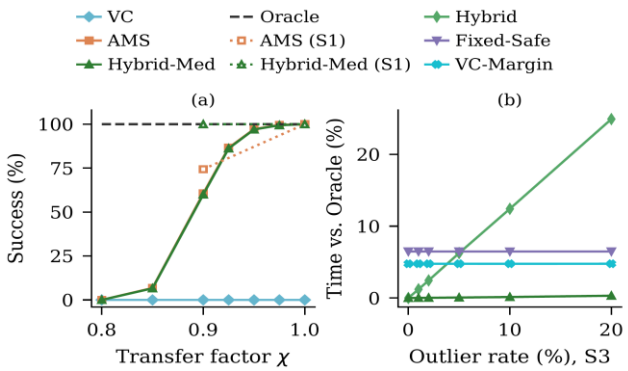


Fig -5: (a) Success against exploration transfer factor χ in S3 (solid; AMS and Hybrid-Med overlap) and S1 (dotted) (E6). (b) Time overhead against outlier rate in S3 (E7); AMS (not shown) overlaps Hybrid. 500 runs per point and strategy.

6.7 Voltage-Sensor Errors

Fig. 6 examines the voltage measurement on which VC and the hybrid models rely. A sensor that over-reads the speed factor makes VC command motions that are too short: in S1 its success fell to 94.6% at +1% gain error, 38.2% at +2% and 0% from +4%. Hybrid-Med stayed at 100% across the whole range −6% to +8% in both S1 and S3, because the same biased reading enters the residual learned during exploration and cancels in the command, given full transfer. An under-reading sensor is safe for both and costs time, 6.4% at −6% in S1. VC-Margin tolerates over-reading only up to the part of its allowance that friction leaves free: 97.0% at +6% and 5.0% at +8% in S1, but 95.6% at +3%, 55.2% at +4% and 0% from +6% in S3. Random per-reading noise is different: it does not cancel, and VC and Hybrid-Med fell to about 90% at $\sigma_v = 1\%$ and to 20–24% at 2%, while AMS, which uses no sensor, was unaffected, and VC-Margin stayed at 100% because its allowance absorbs the noise, at 12.9% overhead. The voltage reading must therefore be filtered to about 0.5% noise, which averaging several ADC samples can typically provide.

Experiment E11 combines an over-reading sensor with partial transfer in S1. Hybrid-Med held 100% with an exact sensor at every χ and with full transfer at every gain error, but not with both. At +1%, +2% and +4% gain error its success was 94.6%, 38.2% and 0% for $\chi \leq 0.75$, where it behaves like VC, 94.6%, 74.4% and 74.4% for $\chi = 0.9$, and 97.4% in all three cases for $\chi = 0.95$. The sensor-gain rule reproduces all 20 cells within 0.5 percentage points. VC-Margin stayed at 100% throughout. The voltage divider should therefore over-read by less than about 0.5%, or be biased to under-read, which costs the same percentage in time.

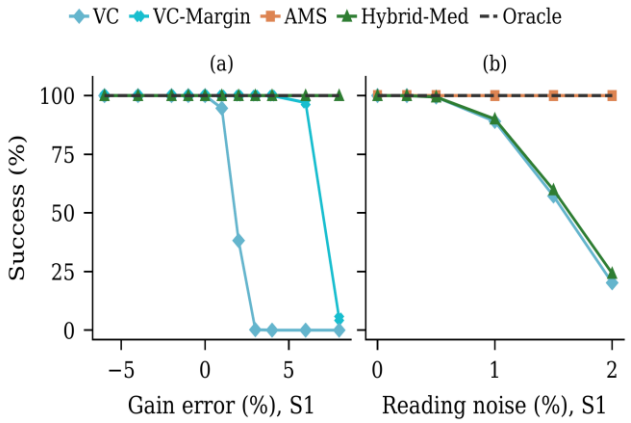


Fig -6: Voltage-sensor errors in S1 (E10, 500 trials per point): (a) gain error; (b) per-reading noise. Hybrid-Med and Oracle overlap in (a); AMS, VC-Margin and Oracle in (b).

6.8 Continuous Discharge

In E8 the battery keeps discharging with motion time ($\rho_F = 0$). Exploration and return took 246–256 s with $\rho_P = 0$ and 257–268 s with $\rho_P = 10\%$, because pivots are slower; in the latter case the speed factor at the start of the speed run was 0.987, 0.974, 0.961, 0.947 and 0.920 for $\kappa = 0.5$ to $3 \times 10^{-4}\ s^{-1}$, below the average seen during exploration (Fig. 7). With $\rho_P = 0$, VC, VC-Margin, both hybrid variants and the oracle stayed at 100%, whereas AMS fell to 82.8% at $\kappa = 10^{-4}$ and 2.4% at 2×10^{-4}, because its table averages the whole exploration. With $\rho_P = 10\%$, VC also failed every run; VC-Margin and the hybrid variants stayed at 100%, VC-Margin at 7.2% overhead and the hybrid variants within 0.02% of the oracle. Neither voltage sensing nor exploration learning alone is sufficient when both kinds of drift are present.

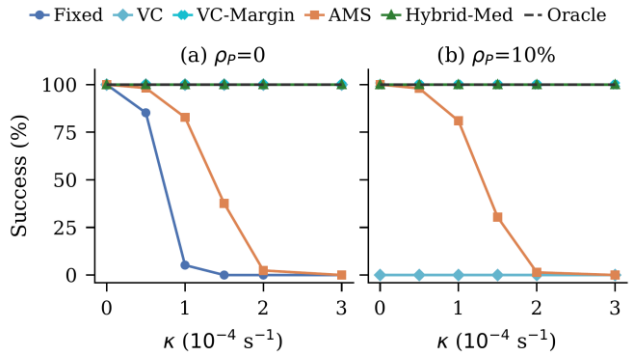


Fig -7: Success under continuous discharge (E8, 500 trials per point): (a) without and (b) with pivot residual. VC, VC-Margin, Hybrid-Med and Oracle overlap at 100% in (a); VC-Margin, Hybrid-Med and Oracle in (b).

7. DISCUSSION

7.1 Implications

A fixed table tolerates only about 4% of slowdown at a 2% margin, so it must be corrected or padded. Padding works but costs 3–27% of speed-run time and fails beyond its margin. VC-Margin is the strongest baseline: it matched the adaptive models in success except for residuals beyond its allowance and over-reading sensors, and it is safer when exploration hides friction, but it was 3–13% slower in every scenario. Voltage feedforward follows a draining battery but not friction; exploration learning captures any static slowdown that exploration reveals but not continuing discharge. Hybrid-Med combines the two and came within 2 percentage points of the oracle's success in every experiment except S3 of E6 (friction hidden during exploration), the voltage-noise sweep of E10 and the over-reading sensor with partial transfer in E11. Table 5 lists the resulting design rules.

Table -5: Design Rules Derived From the Experiments

Choice	Recommendation	Evidence
Cost model	Hybrid-Med: voltage feedforward × median residual	Table 3, Figs. 5, 7
Threshold	None ($\tau = 0$)	Sec. 4, E4
Margin	Effective tolerance $\gamma \geq z\sigma$, $z \approx 3.5$–3.8, i.e. $\mu \approx z\sigma - 2\%$	Sec. 4, Fig. 4
Estimator	Median (robust to outliers)	Fig. 5(b)
Voltage	Filter to $\lesssim 0.5\%$ noise; gain error is absorbed only with full transfer, otherwise keep over-reading below $\gamma - z\sigma$	Fig. 6, E11
Transfer	Shortfall $(1-\chi)(f-1)/f < \gamma - z\sigma$ ($\chi \gtrsim 0.975$ in S3); otherwise add a friction allowance	Sec. 4, Fig. 5(a)

7.2 Scope

The study is a simulation at the level of primitive timings: degradation scales durations, noise is Gaussian, a crash is a timing threshold, the robot turns by in-place pivots rather than smooth or diagonal turns, and wall sensing and localisation are ideal. Within this scope, the transfer factor and the pivot residual decide whether exploration-based learning applies and how much it gains over a friction allowance, and they are the first parameters to measure on hardware, for example from 20 pivots on two floor surfaces with cold and warm motors. A Raspberry Pi Pico 2 W micromouse [27] with a TB6612FNG driver [32], N20 encoder motors, infrared wall sensors and an MPU6500 gyroscope has been designed for this purpose. Its motors run from the unregulated two-cell pack, so they see the sag studied here, and the encoders and pack ADC supply the timings and voltage the hybrid model needs.

8. CONCLUSION

Fixed micromouse cost tables fail beyond about 4% slowdown, voltage compensation fails once friction is present, and margins large enough to cover both cost 3–27% of speed-run time. A hybrid model that multiplies voltage feedforward by a median-estimated residual learned during exploration matched an oracle in success across voltage sag, friction, their combination, outliers and continuous discharge; its mean time was within 0.04% of the oracle's in the static scenarios and within 0.32% with up to 20% outliers. It also absorbed voltage-sensor gain errors that defeat voltage compensation, provided the reading is filtered and exploration reveals the full slowdown; with partial transfer, over-reading must stay below about 0.5%. A closed-form failure model predicted the simulated failure rates and fixed the remaining design choices: no detection threshold, an effective tolerance γ of

about 3.5 noise standard deviations ($\mu \approx 3.5\sigma - 2\%$), and exploration that reveals enough of the slowdown to keep the residual shortfall below $\gamma - z\sigma$, about 97.5% transfer for the S3 pivots. If exploration hides friction, a fixed friction allowance is safer. Future work will measure the transfer and the pivot residual on the Pico 2 W platform and add recursive estimators that update the residual during the speed run [20], [21].

ACKNOWLEDGMENT

The authors thank the Department of Robotics and Artificial Intelligence, Bangalore Institute of Technology, Bengaluru, for its facilities and support.

REFERENCES

[1] S. Mishra and P. Bande, "Maze solving algorithms for micro mouse," in Proc. IEEE Int. Conf. Signal Image Technology and Internet Based Systems (SITIS), 2008, pp. 86–93, doi: 10.1109/SITIS.2008.104.

[2] G. Law, "Quantitative comparison of flood fill and modified flood fill algorithms," Int. J. Computer Theory and Engineering, vol. 5, no. 3, pp. 503–508, 2013, doi: 10.7763/IJCTE.2013.V5.738.

[3] L. Lu, P. You, X. Liu, Z. Zhang, and Z. Du, "Overview of maze path planning algorithms for micromouse," in Proc. SPIE 9th Int. Conf. Computing, Control and Industrial Engineering (CCIE), vol. 13976, 2025, art. no. 139760X, doi: 10.1117/12.3074338.

[4] P. V. F. Zawadniak et al., "A micromouse scanning and planning algorithm based on modified floodfill methodology with optimization," in Proc. IEEE Int. Conf. Autonomous Robot Systems and Competitions (ICARSC), 2020, pp. 245–250, doi: 10.1109/ICARSC49921.2020.9096193.

[5] L. Biagiotti and C. Melchiorri, Trajectory Planning for Automatic Machines and Robots. Berlin, Germany: Springer, 2008.

[6] P. Harrison, "Reliable turns," Presented at MINOS 2015 micromouse conference, 2015. [Online]. Available: https://micromouseonline.com/wp-content/uploads/2015/06/MINOS15-Reliable-Turns-Peter-Harrison.pdf, accessed: Oct. 9, 2026.

[7] T. B. Reddy, Ed., Linden's Handbook of Batteries, 4th ed. New York, NY, USA: McGraw-Hill, 2011.

[8] A. Hughes and B. Drury, Electric Motors and Drives: Fundamentals, Types and Applications, 4th ed. Oxford, U.K.: Newnes, 2013.

[9] N. Jakobi, P. Husbands, and I. Harvey, "Noise and the reality gap: The use of simulation in evolutionary robotics," in Advances in Artificial Life (ECAL), Lecture Notes in Computer Science, vol. 929. Springer, 1995, pp. 704–720.

[10] E. Salvato, G. Fenu, E. Medvet, and F. A. Pellegrino, "Crossing the reality gap: A survey on sim-to-real transferability of robot controllers in reinforcement learning," IEEE Access, vol. 9, pp. 153 171–153 187, 2021, doi: 10.1109/ACCESS.2021.3126658.

[11] P. V. F. Zawadniak et al., "Micromouse 3D simulator with dynamics capability: a Unity environment approach," SN Applied Sciences, vol. 3, no. 2, Feb. 2021, art. no. 259, doi: 10.1007/s42452-021-04239-7.

[12] Z. Cai, L. Ye, and A. Yang, "FloodFill maze solving with expected toll of penetrating unknown walls for micromouse," in Proc. IEEE 14th Int. Conf. High Performance Computing and Communications (HPCC), 2012, pp. 1428–1433, doi: 10.1109/HPCC.2012.209.

[13] J. A. Abdulsaheb and D. J. Kadhim, "Classical and heuristic approaches for mobile robot path planning: A survey," Robotics, vol. 12, no. 4, 2023, art. no. 93, doi: 10.3390/robotics12040093.

[14] E. W. Dijkstra, "A note on two problems in connexion with graphs," Numerische Mathematik, vol. 1, pp. 269–271, 1959.

[15] P. Harrison, "Motor equations," Micromouse Online. [Online]. Available: https://micromouseonline.com/micromouse-book/motors/motor-equations/, accessed: Oct. 9, 2026.

[16] J. J. Park, S. Lee, and B. Kuipers, "Discrete-time dynamic modeling and calibration of differential-drive mobile robots with friction," in Proc. IEEE Int. Conf. Robotics and Automation (ICRA), 2017, pp. 6510–6517, doi: 10.1109/ICRA.2017.7989769.

[17] H. Hellani, P. Chauchat, and G. Graton, "Autonomous mobile robot modeling including battery consumption," in Proc. Int. Conf. Integrated Modeling and Analysis in Applied Control and Automation (IMAACA), Fès, Morocco, Sep. 2025, HAL Id: hal-05314427. [Online]. Available: https://amu.hal.science/hal-05314427v1.

[18] K. J. Åström and B. Wittenmark, Adaptive Control, 2nd ed. Reading, MA, USA: Addison-Wesley, 1995.

[19] K. Guo and Y. Pan, "Composite adaptation and learning for robot control: A survey," Annual Reviews in Control, vol. 55, pp. 279–290, 2023, doi: 10.1016/j.arcontrol.2022.12.001.

[20] L. Ljung, System Identification: Theory for the User, 2nd ed. Upper Saddle River, NJ, USA: Prentice Hall, 1999.

[21] R. E. Kalman, "A new approach to linear filtering and prediction problems," Trans. ASME, J. Basic Engineering, vol. 82, no. 1, pp. 35–45, 1960.

[22] C. Mühlbacher, S. Gspandl, M. Reip, and G. Steinbauer, "Adapting edge weights for optimal paths in a navigation graph," in Advances in Service and Industrial Robotics (RAAD 2017), Mechanisms and Machine Science. Springer, 2017, pp. 372–380, doi: 10.1007/978-3-319-61276-8_41.

[23] J. J. Chung, A. J. Smith, R. Skeele, and G. A. Hollinger, "Risk-aware graph search with dynamic edge cost discovery," Int. J. Robotics Research, vol. 38, no. 2–3, pp. 182–195, 2019, doi: 10.1177/0278364918781009.

[24] L. Wellhausen, A. Dosovitskiy, R. Ranftl, K. Walas, C. Cadena, and M. Hutter, "Where should I walk? predicting terrain properties from images via self-supervised learning," IEEE Robotics and Automation Letters, vol. 4, no. 2, pp. 1509–1516, 2019, doi: 10.1109/LRA.2019.2895390.

[25] A. Zeng, S. Song, J. Lee, A. Rodriguez, and T. Funkhouser, "TossingBot: Learning to throw arbitrary objects with residual physics," IEEE Trans. Robotics, vol. 36, no. 4, pp. 1307–1319, 2020, doi: 10.1109/TRO.2020.2988642.

[26] M. Matsumoto and T. Nishimura, "Mersenne twister: A 623-dimensionally equidistributed uniform pseudo-random number generator," ACM Trans. Modeling and Computer Simulation, vol. 8, no. 1, pp. 3–30, 1998.

[27] Raspberry Pi Ltd, "Raspberry Pi Pico 2 W datasheet," Raspberry Pi Ltd, Cambridge, U.K., 2024. [Online]. Available: https://datasheets.raspberrypi.com/picow/pico-2-w-datasheet.pdf, accessed: Oct. 9, 2026.

[28] P. Virtanen et al., "SciPy 1.0: Fundamental algorithms for scientific computing in Python," Nature Methods, vol. 17, no. 3, pp. 261–272, 2020, doi: 10.1038/s41592-019-0686-2.

[29] E. B. Wilson, "Probable inference, the law of succession, and statistical inference," J. American Statistical Association, vol. 22, no. 158, pp. 209–212, 1927.

[30] Q. McNemar, "Note on the sampling error of the difference between correlated proportions or percentages," Psychometrika, vol. 12, no. 2, pp. 153–157, 1947.

[31] F. Wilcoxon, "Individual comparisons by ranking methods," Biometrics Bulletin, vol. 1, no. 6, pp. 80–83, 1945.

[32] Toshiba Corporation, "TB6612FNG driver IC for dual DC motor, datasheet," Toshiba Corp., Tokyo, Japan, 2014. [Online]. Available: https://toshiba.semicon-storage.com/info/docget.jsp?did=10660, accessed: Oct. 9, 2026.