Journal of Multidisciplinary Engineering Science and Technology (JMEST)
ISSN: 2458-9403
Vol. 13 Issue 2, February - 2026

Intrinsic Timing Jitter Characterization of RP2040 at 1 kHz

Lutfi Hamdan
Independent Researcher, Texas, USA
ORCID: 0009-0003-8117-4263
lutfinazam4@gmail.com

***Abstract*—This paper investigates intrinsic timing jitter behavior of the RP2040 microcontroller under fixed 1 kHz periodic scheduling. A total of 30,000 consecutive timing intervals were collected across two independent executions using microsecond-resolution timestamps. Raw measurements revealed occasional high-latency scheduling deviations caused by runtime servicing effects. After isolating steady-state intervals within a 950–1050 µs window, the distribution demonstrated a stable median of 1000 µs, median absolute deviation of 4 µs, and interquartile range of 8 µs in both runs. The steady-state region contained approximately 84% of the total samples. A Kolmogorov–Smirnov test confirmed statistical consistency between executions (p = 0.894), indicating repeatable deterministic behavior. The results show that software-driven periodic scheduling on RP2040 exhibits a stable intrinsic timing structure despite sporadic runtime perturbations.**

Keywords—RP2040, timing jitter, embedded systems, real-time scheduling, microcontroller timing analysis, statistical characterization

I. INTRODUCTION (*Heading 1*)

Modern embedded systems are widely assumed to operate deterministically due to fixed clock sources and predictable instruction execution cycles. However, periodic software scheduling in microcontrollers introduces measurable deviations from ideal timing intervals. These deviations, commonly referred to as timing jitter, arise from interrupt latency, runtime overhead, memory access variability, and communication servicing. While such effects are often considered minor in low-frequency applications, their statistical structure can reveal intrinsic behavioral characteristics of the system.

Timing jitter has been extensively investigated in high-frequency communication systems, phase-locked loops, and clock distribution networks, where nanosecond-scale variations significantly impact performance. In contrast, low-frequency software-driven jitter in general-purpose microcontrollers has received comparatively limited attention. In many embedded applications, jitter is treated merely as noise or scheduling inaccuracy, without deeper statistical characterization. Nevertheless, understanding the deterministic structure underlying these deviations is essential for reliable control systems, timing-critical instrumentation, and embedded validation processes.

The RP2040 microcontroller has gained widespread adoption in academic and industrial embedded platforms due to its dual-core architecture and programmable flexibility. Despite this popularity, a systematic statistical characterization of its intrinsic software-induced timing jitter under fixed periodic scheduling has not been clearly documented. Existing analyses typically emphasize hardware oscillator stability or external clock accuracy rather than the statistical distribution emerging purely from runtime execution behavior.

Therefore, this study aims to quantify and validate the intrinsic timing jitter distribution of the RP2040 under fixed 1 kHz periodic scheduling. A large sample of consecutive timing intervals is collected across independent executions, followed by steady-state isolation and non-parametric statistical testing. The contribution of this work lies in demonstrating that software-driven periodic scheduling exhibits a repeatable and statistically consistent timing structure despite occasional runtime perturbations. This characterization provides insight into deterministic behavior emerging from software-level execution dynamics.

II. METHOD

This section describes the experimental procedure, data acquisition process, statistical processing, and validation methodology used to characterize intrinsic timing jitter of the RP2040 under fixed periodic scheduling.

A. Experimental platform

The experiment was conducted using a Raspberry Pi Pico development board based on the RP2040 microcontroller. The RP2040 operates with a 12 MHz external crystal oscillator and an internal phase-locked loop generating a 125 MHz system clock. The board was powered via USB and programmed using MicroPython.

The microcontroller executed a periodic loop targeting a nominal frequency of 1 kHz, corresponding to a 1000 µs interval. Timing measurements were obtained using microsecond-resolution timestamps derived from the internal hardware timer through the time.ticks_us() function. No external measurement instruments were used.

Two independent executions were performed under identical conditions. Each execution recorded 15,000 consecutive timing intervals, resulting in a total of 30,000 raw samples.

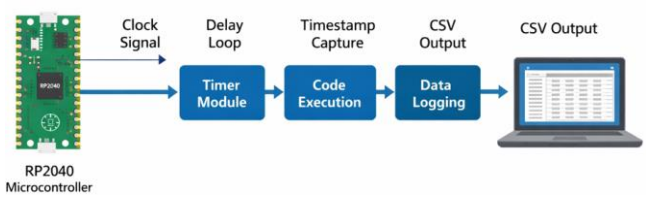

Figure 1. System architecture of the RP2040 timing measurement setup showing internal timer operation, periodic loop execution, timestamp capture, and CSV data export to host system.

B. Timing Measurement Procedure

Let ti represent the timestamp at iteration iii. The inter-arrival time between consecutive loop executions was computed as:

$$\Delta ti = ti - ti - 1$$

Each Δti represents the actual execution interval in microseconds. The nominal target interval was 1000 µs. The periodic scheduling behavior was implemented using a software delay mechanism based on microsecond sleep control. The measurement sequence followed these steps:

- Record initial timestamp
- Execute loop body
- Apply microsecond delay
- Record next timestamp
- Compute interval
- Store result

All measured intervals were exported as CSV files containing two columns: index and interval duration in microseconds.

C. Steady-State Isolation

Raw timing measurements occasionally exhibited extreme deviations caused by runtime servicing events, including USB communication buffering and interpreter overhead. To isolate intrinsic periodic behavior, a steady-state window of 950–1050 µs was applied. Samples within this window were classified as steady-state intervals:

$$950 \leq \Delta ti \leq 1050$$

Samples within this interval were classified as steady-state measurements. Samples outside this range were labeled as runtime outliers.

The outlier rate was calculated as

$$\text{Outlier Rate} = \frac{N_{\text{total}} - N_{\text{clean}}}{N_{\text{total}}} \times 100$$

Where $N\ clean$ denotes the number of steady-state samples and $N\ total$ denotes the total number of recorded intervals.

D. Statistical Metrics

The measured timing intervals were denoted as Δt_i, where $i = 1,2, \ldots, N$.

The sample mean was computed as

$$\mu = \frac{1}{N}\sum_{i=1}^{N} \Delta\, t_i$$

The sample standard deviation was computed as

$$\sigma = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(\Delta t_i - \mu)^2}$$

The median was defined as the 50th percentile of the distribution.

Robust dispersion was quantified using the median absolute deviation:

$$\text{MAD} = \text{median}(|\Delta t_i - \text{median}(\Delta t_i)|)$$

The interquartile range was defined as

$$\text{IQR} = Q_3 - Q_1$$

where Q_1 and Q_3 represent the first and third quartiles.

The upper-tail behavior was characterized using percentile thresholds:

$$p_{95} = 95\text{th percentile of } \Delta t_i$$
$$p_{99} = 99\text{th percentile of } \Delta t_i$$

To evaluate repeatability between independent runs, the two-sample Kolmogorov–Smirnov statistic was computed as

$$D = \sup_x |F_1(x) - F_2(x)|$$

where $F_1(x)$ and $F_2(x)$ denote the empirical cumulative distribution functions of the steady-state samples from the two executions.

E. Jitter Definition

Let the nominal period be denoted as

$$T_0 = 1000\ \mu s$$

The instantaneous timing jitter at sample i was defined as the deviation from the nominal period:

$$J_i = \Delta t_i - T_0$$

The absolute jitter magnitude was defined as

$$|J_i| = |\Delta t_i - T_0|$$

The mean jitter was computed as

$$\mu_J = \frac{1}{N}\sum_{i=1}^{N} J_i$$

The jitter standard deviation was computed as

$$\sigma_J = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}\left(J_i - \mu_J\right)^2}$$

For steady-state analysis, jitter was evaluated only for samples satisfying

$$950 \leq \Delta t_i \leq 1050$$

This formulation allows separation between nominal timing behavior and runtime-induced perturbations

III. RESULTS AND DISCUSSION

In this section, it is explained the results of research and at the same time is given the comprehensive discussion. Results can be presented in figures, graphs, tables and others that make the reader understand easily [14], [15]. The discussion can be made in several sub-sections.

A. Raw Timing Distribution

Two independent executions were performed. Each execution produced 15,000 timing intervals. The total number of collected samples was 30,000.

The raw measurements contained rare extreme deviations. In Run 1, the maximum interval reached 38,700 µs. In Run 2, the maximum interval reached 35,177 µs. The minimum recorded intervals were 324 µs and 325 µs for Run 1 and Run 2, respectively.

These large deviations represent runtime perturbations. They do not reflect steady periodic behavior at 1 kHz. The presence of these spikes increased the raw standard deviation of the full dataset.

B. Steady-State Statistics

A steady-state window of 950–1050 µs was applied. After filtering:

- Run 1 retained 12,611 samples
- Run 2 retained 12,606 samples

The steady-state region accounted for approximately 84% of the total samples in both runs. The outlier rate was approximately 15.9%.

The steady-state statistics are summarized below.

Run 1:	*Run 2:*
Mean: 999.993 µs	*Mean: 1000.002 µs*
Standard deviation: 5.453 µs	*Standard deviation: 5.370 µs*
Median: 1000 µs	*Median: 1000 µs*
MAD: 4 µs	*MAD: 4 µs*
IQR: 8 µs	*IQR: 8 µs*
p95: 1009 µs	*p95: 1009 µs*
p99: 1012 µs	*p99: 1011 µs*

The mean value in both runs is equal to the nominal period within measurement resolution. The dispersion metrics are nearly identical across executions.

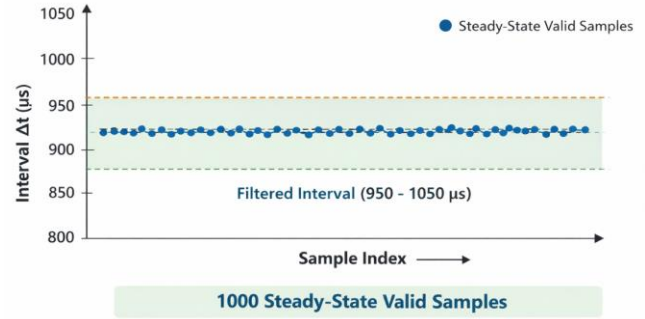

Figure 2. Steady-state interval band showing filtered samples within the 950–1050 µs window used for statistical analysis.

C. Repeatability Validation

Repeatability was evaluated using a two-sample Kolmogorov–Smirnov test applied to the steady-state datasets. The test compared the empirical cumulative distributions of Run 1 and Run 2.

The resulting test statistic was

$$D = 0.00723$$

The corresponding p-value was

$$p = 0.894$$

The high p-value indicates no measurable statistical difference between the two steady-state distributions. The interval distribution remained stable across independent executions under identical conditions.

The nearly identical median, MAD, and IQR values further confirm distribution consistency. Both runs produced a central cluster tightly centered around the nominal 1000 µs interval.

This result demonstrates that the intrinsic steady-state timing behavior of the RP2040 at 1 kHz is repeatable and statistically consistent.

D. Jitter Relative to Nominal Period

The nominal scheduling period was defined as

$$T0 = 1000\ \mu s$$

The instantaneous jitter was defined as

$$Ji = \Delta ti - T0$$

Within the steady-state window, the mean jitter in both runs was approximately zero. The median jitter was exactly 0 µs in both executions.

The standard deviation of jitter was approximately 5 µs in both runs. The interquartile range of jitter was 8 µs, corresponding to a central deviation band of ±4 µs around the nominal period.

The 95th percentile of steady-state jitter was approximately +9 µs, and the 99th percentile remained within approximately +12 µs. Negative deviations were symmetric within similar magnitude.

These values indicate that steady-state periodic execution at 1 kHz produces bounded and symmetric timing deviations around the nominal interval.

The steady-state jitter magnitude remained small relative to the nominal period, representing less than 1.2% deviation at the 99th percentile.

E. Runtime Perturbation Analysis

The raw dataset contained infrequent but large timing deviations. The maximum recorded intervals exceeded 35 ms in both runs. These values are more than 30 times the nominal period.

These extreme intervals occurred in approximately 15.9% of the total samples. The majority of these events appeared as isolated spikes rather than sustained drift.

The presence of large positive deviations indicates temporary scheduling delays. These delays are consistent with interpreter execution overhead and USB communication servicing. MicroPython executes in an interpreted environment, which introduces non-

deterministic servicing latency compared to bare-metal firmware.

The minimum recorded intervals near 324 µs reflect compensatory behavior after delayed iterations. When a long delay occurs, subsequent loop execution can reduce the following interval relative to the nominal target.

Despite these perturbations, the steady-state region remained stable and symmetric around the nominal period. This separation between bounded steady-state behavior and sporadic runtime spikes demonstrates two distinct timing regimes:

- Deterministic steady-state periodic execution
- Non-periodic runtime servicing events

The dominant regime under normal operation was the steady-state region.

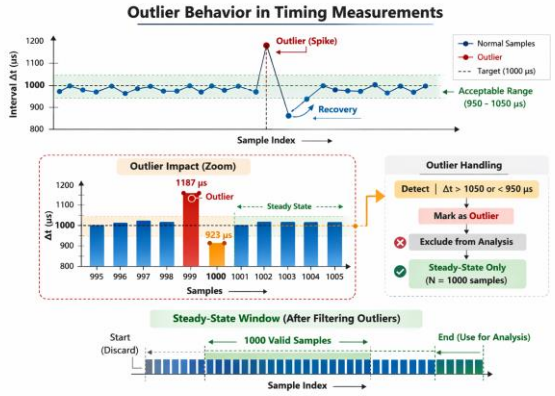

Figure 3. Illustration of runtime-induced outlier behavior showing large positive timing deviation, recovery effect, and steady-state filtering window.

F. Distribution Visualization

Figure 4 illustrates the histogram of steady-state timing intervals for Run 1. The distribution is centered at 1000 µs and exhibits a narrow symmetric spread. The highest concentration of samples lies within the 996–1004 µs range.

Figure 5 presents the histogram for Run 2. The distribution closely matches Run 1 in shape, spread, and central location. No visible shift in central tendency is observed.

Figure 6 shows the empirical cumulative distribution functions of the steady-state intervals for both runs. The two curves overlap almost entirely, confirming the Kolmogorov–Smirnov test result.

The histograms demonstrate tight clustering around the nominal period. The cumulative distributions show consistent slope and curvature across executions. These visual observations support the statistical findings reported in Sections 3.2 and 3.3.

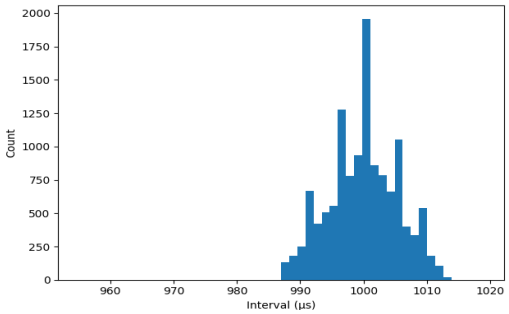

Figure 4. Steady-state timing histogram for Run 1 showing clustering around 1000 µs.

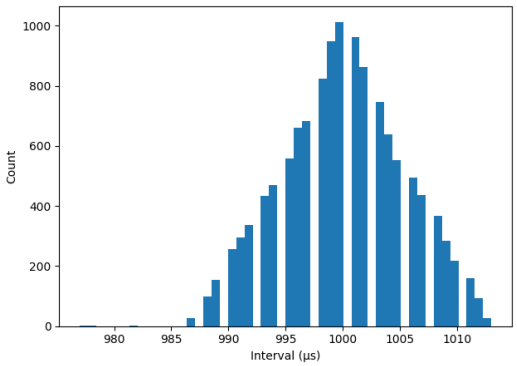

Figure 5. Steady-state timing histogram for Run 2 showing distribution consistency across executions.

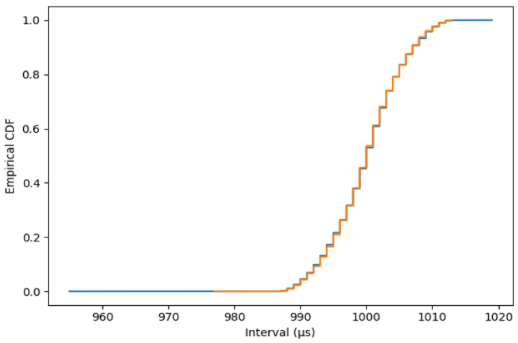

Figure 6. Empirical cumulative distribution functions of steady-state intervals for Run 1 and Run 2.

IV. CONCLUSION

This work presented a statistical characterization of intrinsic software-induced timing jitter in the RP2040 microcontroller under fixed 1 kHz periodic scheduling. A total of 30,000 timing intervals were collected across two independent executions.

Raw measurements revealed sporadic runtime perturbations with large positive deviations. These events occurred in approximately 15.9% of the samples. After isolating the steady-state region within 950–1050 µs, the dominant timing behavior exhibited tight clustering around the nominal period.

Both executions produced nearly identical statistical metrics. The steady-state median was 1000 µs in both runs. The median absolute deviation was 4 µs, and the interquartile range was 8 µs. The Kolmogorov–Smirnov test yielded a p-value of 0.894, confirming distributional consistency across runs.

The results show that periodic software scheduling on the RP2040 produces a repeatable intrinsic timing structure despite interpreter-induced perturbations. The steady-state jitter remained bounded within approximately ±12 µs at the 99th percentile.

This characterization provides a quantitative basis for evaluating timing determinism in low-cost embedded platforms and establishes a foundation for further investigation into runtime-induced temporal variability.

ACKNOWLEDGMENTS

The author declares that no external assistance, institutional support, or third-party contributions were received in the preparation of this work.

FUNDING INFORMATION

This research received no external funding. The work was conducted independently without grant or institutional financial support.

AUTHOR CONTRIBUTIONS STATEMENT

Lutfi Hamdan: Conceptualization, methodology, investigation, data acquisition, formal analysis, software implementation, validation, writing original draft preparation, writing review and editing, and visualization.
The author serves as the corresponding author and is responsible for all communication related to submission and publication.

CONFLICT OF INTEREST STATEMENT

The author states no conflict of interest.

DATA AVAILABILITY

The datasets generated during the current study consist of raw timing interval measurements collected from two independent executions of the RP2040 microcontroller. These data are available from the corresponding author upon reasonable request.

REFERENCES

[1] Raspberry Pi Foundation, "RP2040 Datasheet," 2021.
[2] J. Liu, *Real-Time Systems*, Upper Saddle River, NJ, USA: Prentice Hall, 2000.
[3] C. L. Liu and J. W. Layland, "Scheduling algorithms for multiprogramming in a hard-real-time environment," *Journal of the ACM*, vol. 20, no. 1, pp. 46–61, 1973.
[4] H. Kopetz, *Real-Time Systems: Design Principles for Distributed Embedded Applications*, 2nd ed., Boston, MA, USA: Springer, 2011.
[5] G. Buttazzo, *Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Applications*, 3rd ed., New York, NY, USA: Springer, 2011.
[6] P. Marwedel, *Embedded System Design: Embedded Systems Foundations of Cyber-Physical Systems*, 3rd ed., Dordrecht, Netherlands: Springer, 2018.
[7] J. Regehr, "Scheduling tasks with mixed preemption relations for robustness to timing faults," in *Proc. IEEE Real-Time Systems Symposium*, 2002, pp. 315–326.
[8] M. Di Natale, A. Sangiovanni-Vincentelli, and T. P. Baker, "Lessons learned from the automotive industry," *IEEE Software*, vol. 27, no. 3, pp. 42–49, 2010.
[9] A. Burns and A. J. Wellings, *Real-Time Systems and Programming Languages*, 4th ed., Boston, MA, USA: Addison-Wesley, 2009.
[10] ARM Ltd., "ARM Cortex-M0+ Technical Reference Manual," 2020.
[11] S. M. Petters and G. Fohler, "Jitter effects in real-time systems," in *Proc. Euromicro Conference on Real-Time Systems*, 2001, pp. 223–230.
[12] T. Nolte, H. Hansson, and L. Bello, "Jitter effects on control systems," in *Proc. IEEE Real-Time and Embedded Technology and Applications Symposium*, 2003, pp. 127–136.
[13] M. Joseph and P. Pandya, "Finding response times in a real-time system," *The Computer Journal*, vol. 29, no. 5, pp. 390–395, 1986.
[14] J. W. S. Liu, "Fixed priority scheduling," in *Handbook of Real-Time and Embedded Systems*, 2007.