Indonesian Journal of Computer Science
ISSN 2549-7286 (*online*)
Jln. Khatib Sulaiman Dalam No. 1, Padang, Indonesia
Website: ijcs.stmikindonesia.ac.id | E-mail: ijcs@stmikindonesia.ac.id

Developing a Gantry Robot with Pre-calculated pure S-curve Motion Profiles for Delicate Egg Handling: Utilizing ESP32, FreeRTOS, and AS5600 Encoders

Ju Ju Naing[1], War War Min Swe[2], Htay Htay Win[3]
doublejue@gmail.com, warwarminswe288@gmail.com, htayhtayw@gmail.com
[1, 2, 3]Department of Mechanical Engineering, Mandalay Technological University, Mandalay, Myanmar

Article Information

Submitted : 27 Mar 2024
Reviewed: 30 Mar 2024
Accepted : 8 Apr 2024

Keywords

Gantry robot, S-curve motion profile, FreeRTOS, egg handling, automation

Abstract

The poultry industry faces challenges due to egg breakage during transfer. This paper describes a gantry robot specifically designed for delicate egg handling. The robot utilizes pre-calculated pure S-curve motion profiles to achieve smooth and precise movements, minimizing stress on the eggs. This approach leverages the computational efficiency of pre-calculation, making it suitable for low-power microcontrollers like the ESP32. FreeRTOS(Free Real-Time Operating System) ensures real-time task management for profile execution and data collection every 4 milliseconds from the AS5600 encoders. These encoders provide high-resolution angular position feedback, allowing for comparison with the planned S-curve profile after each movement step. This system offers advantages such as reduced egg breakage, improved transfer efficiency, and a simpler design compared to real-time control. However, limitations include limited adaptability to significant environmental changes and disturbances. Future work may investigate incorporating real-time feedback control for enhanced robustness.

A. Introduction

The poultry industry relies heavily on efficient and safe egg transfer processes. Traditional methods, often involving manual handling, can be imprecise and lead to significant egg breakage. This breakage results in economic losses and waste, highlighting the need for a more automated and controlled solution.

Gantry robots offer a promising alternative for automated egg transfer. These robots provide greater control and consistency compared to manual handling. This paper introduces a novel gantry robot specifically designed for delicate egg handling. The robot utilizes pre-calculated pure S-curve motion profiles to achieve smooth and precise movements during transfers, minimizing stress on the eggs.

This approach leverages the computational efficiency of pre-calculation, making it suitable for low-power microcontrollers like the ESP32. FreeRTOS, a real-time operating system, ensures efficient task scheduling and communication between the control software and hardware components. Additionally, AS5600 encoders are implemented to provide high-resolution angular position feedback after each movement step. This feedback allows for verification of the planned movements against the pre-calculated S-curve profile, further enhancing accuracy.

B. System Design

This section explores the design considerations for a gantry robot system intended for delicate egg handling. The focus will be on open-loop control using pre-calculated S-curve motion profiles, encompassing both the mechanical design and the selection of key electronic components that ensure its functionality.

1. Mechanical Design

The gantry robot employs a Cartesian coordinate system with three linear axes (X, Y, Z). This configuration allows for precise positioning and controlled movement of the end effector across the workspace, making it suitable for delicate tasks like egg manipulation.

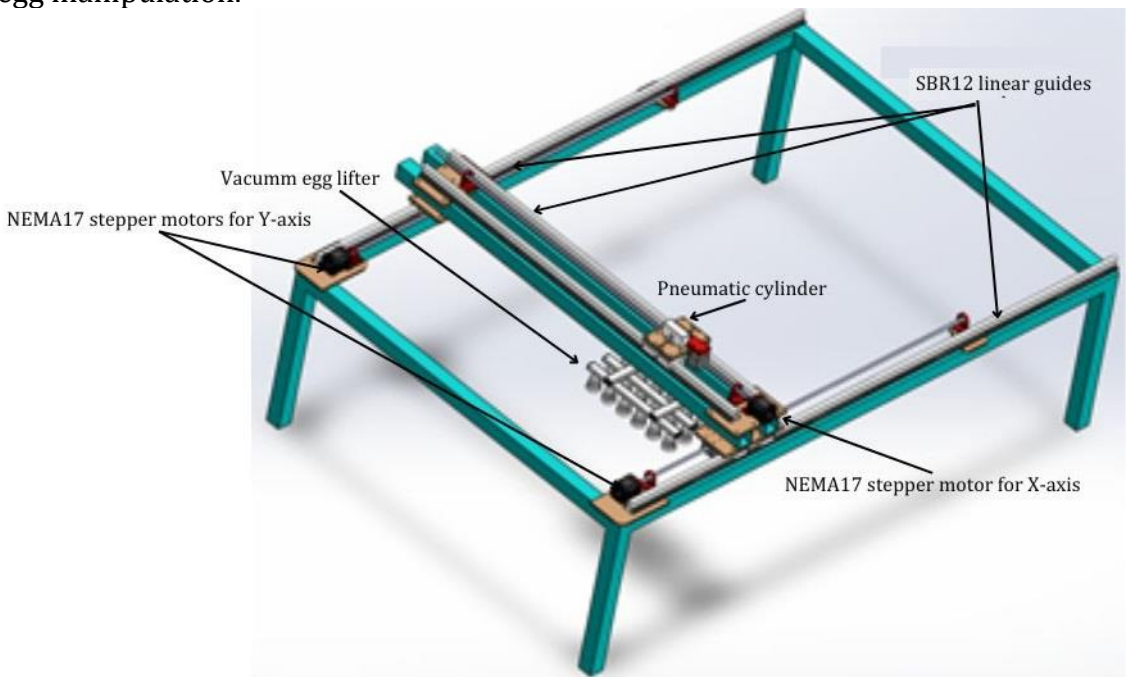


Figure1. Mechanical design of the gantry robot

2. Electronic Components

The experimental setup utilizes the ESP32 microcontroller as its core. This choice is motivated by the ESP32's robust computational capabilities and integrated Wi-Fi functionality, making it well-suited for real-time motion control applications. Its dual-core architecture further enhances its suitability by enabling efficient multitasking, a critical feature for the intended application[1].

The A4988 stepper motor driver is employed due to its precise current regulation capabilities. It supports microstepping, which can further improve the smoothness of movement by dividing each full step of the stepper motor into smaller increments. This can be beneficial for very precise egg handling. It can operate stepper motors in full-step, half-step, quarter-step, eighth-step, and sixteenth-step modes[2]. This ensures accurate control of the torque delivered to the motor windings.

The robot utilizes NEMA 17 stepper motors on each axis for precise movement. Resolutions for NEMA 17 motors are 1.8 degrees or 200 steps per revolution.

AS5600, the 12-bit magnetic encoder with measurement accuracy 0.087890625°, are mounted on each stepper motor of the robot's axes[3].

3. Interconnections and Wiring Diagram

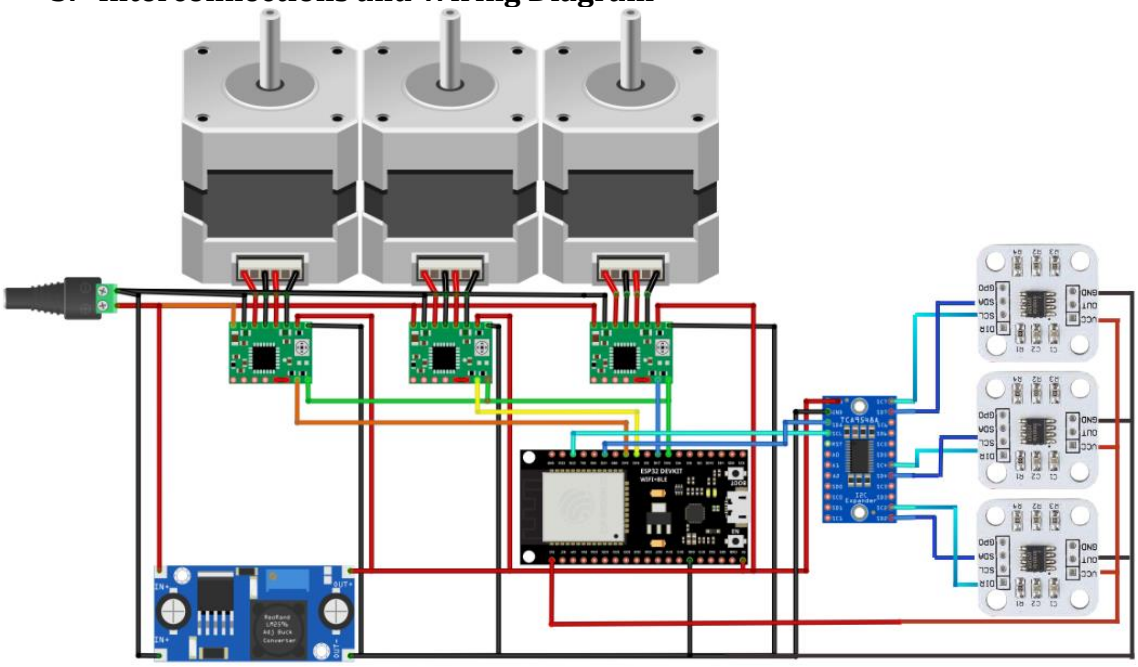


Figure2. Wiring Schematic for ESP32, A4988, and NEMA17

The successful operation of the system hinges on the meticulous interconnection of its components to facilitate smooth data exchange and control. The ESP32 microcontroller communicates with the A4988 stepper driver using dedicated General-Purpose Input/Output (GPIO) pins. This establishes a seamless channel for transmitting control signals to the driver. In response, the stepper driver modulates the current supplied to the coils of the NEMA17 stepper motor, thereby governing its motion. AS5600 magnetic encoders for measuring the rotation of stepper motors are connected through tca9548a i2c multiplexer module.

A comprehensive wiring diagram, depicted in Figure 2, illustrates these connections. The diagram details the pin-to-pin correspondence between the ESP32 and the A4988 drivers, the connections between the driver and the NEMA17 stepper motors, as well as the connections of AS5600 magnetic encoders to ESP32 through i2c multiplexer module. This meticulously planned arrangement ensures a clear and organized hardware configuration, laying the foundation for precise motion control.

4. Communication Protocol and System Design

The system architecture is designed for efficient data exchange and synchronization between the ESP32 microcontroller and the A4988 stepper driver. The ESP32 generates control signals based on the pre-calculated S-curve velocity profile algorithm. These control signals, including step pulses and direction signals, are transmitted to the A4988 driver, instructing it to execute the desired motion.

This meticulous design integrates the ESP32 microcontroller, A4988 stepper driver, and NEMA17 stepper motor seamlessly. A well-defined communication protocol facilitates the smooth transmission of control signals, enabling the implementation of the S-curve velocity profile. This collaborative synergy establishes the foundation for precise motion control. The subsequent section will explore the implementation of the inverse displacement function-based delay mechanism.

Additionally, an AS5600 magnetic encoder is integrated to monitor the system's efficiency and provide feedback on the motor's position.

C. Background Theory

In the field of mechatronics, achieving rapid and precise movement of actuators across diverse applications necessitates mitigating shocks within the mechanism, particularly during the initial and final phases of motion. This challenge becomes more pronounced when utilizing stepper motors, whose inherent discrete movement can exacerbate these effects. A key solution lies in implementing a soft-start operation, characterized by a gradual initiation of movement with controlled speed and acceleration. One well-established method for achieving a soft-start is the S-curve profile, which limits instantaneous speed, acceleration, and jerk[4]. This profile ensures smooth operation by starting with slow movement that gradually accelerates to a constant velocity, followed by a mirrored deceleration pattern near the end of the trajectory. Studies suggest that an S-curve profile duration of approximately 0.5 seconds can be effective for typical mechanical movements[5].

Stepper motors, electromechanical devices that convert digital electrical pulses into precise shaft rotations, offer several advantages. These include accurate positioning, repeatable movements, high torque at standstill, reliability, and ease of maintenance. However, they also present challenges such as resonance and limitations in achieving extremely high speeds. This paper focuses on the critical issue of speed control in stepper motors. Since stepper motors operate in discrete time steps, adjusting the time interval between consecutive pulses is the primary method for achieving different speeds. Determining the optimal timing for pulse generation within these intervals remains a key challenge in optimizing stepper motor speed control. This work explores and evaluates a dedicated variable pulse control architecture, with a particular focus on the variable pulse algorithm. The

objective is to achieve enhanced motion profiling and smoother operation for stepper motor applications in contemporary settings[6].

D. Trajectory Planning

In mechatronic systems, velocity profiles are crucial for achieving precise positioning while minimizing vibrations and energy consumption. The chosen profile significantly impacts the system's behavior, particularly regarding acceleration and jerk (the rate of change in acceleration). Abrupt velocity changes introduce discontinuities in the trajectory, leading to high jerk values that can cause vibrations and potential structural damage. Several velocity profiles are employed for various applications, including triangular, parabolic, trapezoidal, and S-curve profiles[7].

The triangular profile, characterized by sharp changes in acceleration, results in high jerk values. The parabolic profile offers a smoother curve but lacks a constant velocity phase, leading to immediate acceleration and deceleration[7].

The trapezoidal velocity profile, characterized by a constant velocity segment flanked by parabolic blends, is widely employed in industrial robots due to its computational efficiency[8]. However, this profile presents a trade-off between achieving fast motion and maintaining smooth operation within mechanical systems. The inherent discontinuity in acceleration associated with the trapezoidal profile leads to theoretically infinite jerk values, which can induce undesirable vibrations and compromise precision [9].

However, a recent paradigm shift has emerged with the seven-segment (S-curve) velocity profile, recognized for its superior performance. This profile, defined by a third-degree polynomial, ensures a constant jerk value, effectively mitigating damage caused by high-frequency vibrations within the system's structure. While implementing the S-curve profile in embedded systems might require a more complex architecture, its benefits in terms of reduced structural damage and enhanced precision make it a compelling choice for trajectory planning in modern mechatronic systems[7].

A streamlined variant of the conventional seven-phase S-curve, the five-phase model(also called pure S-curve model) strategically discards the uniform acceleration and deceleration segments, transforming the curve into five distinct segments. This transformation reduces the complexity of the piece-wise function, streamlining the motion planning process and achieving a balance between computational efficiency and accuracy. The inherent symmetry between acceleration and deceleration phases is leveraged to further simplify motion planning. This strategic adoption of the five-phase S-curve model represents a nuanced approach to trajectory planning, particularly suitable for resource-constrained scenarios. It offers a harmonious blend of computational efficiency and precision, aligning with the needs of applications with limited processing power[10], [11].

E. Mathematical Representation of the Pure S-Curve Velocity Profile

This section explores the mathematical foundation of the S-curve velocity profile. The following equations represent the acceleration (a_A), velocity (v_A), and

displacement (s_A) of the S-curve during its acceleration phase ($0 \leq t \leq t_a$), where t_a denotes the acceleration time.

$$t_a = \frac{2v_m}{a} \quad (1)$$

$$C_1 = \frac{a^2}{2v_m} \quad (2)$$

$$s_A(t) = C_1 \frac{t^3}{3} \quad (3)$$

$$v_A(t) = C_1 t^2 \quad (4)$$

$$a_A(t) = 2C_1 t \quad (5)$$

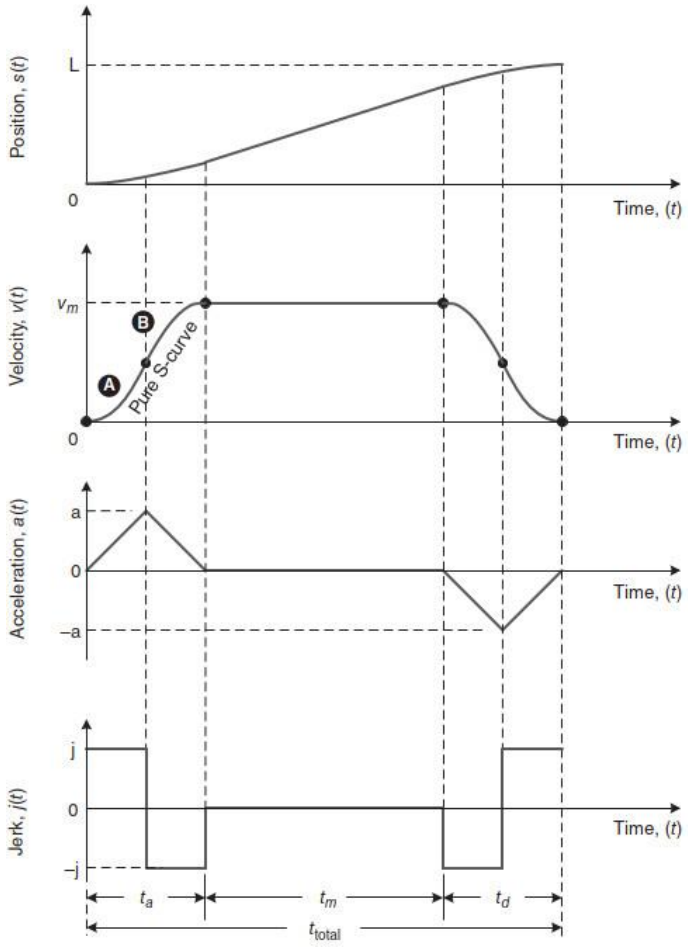


Figure 3. Pure S-curve velocity profile[10]

The subsequent deceleration phase ($t_a \leq t$) is characterized by the equations:

$$s_B(t) = C_1 \frac{t_a^3}{24} + v_m \left(t - \frac{t_a}{2}\right) - C_1 \left\{t_a^2 \left(t - \frac{t_a}{2}\right) - t_a \left(t^2 - \left(\frac{t_a}{2}\right)^2\right) + \frac{1}{3}\left(t^3 - \left(\frac{t_a}{2}\right)^3\right)\right\} \quad (6)$$

$$v_B(t) = v_m - C_1(t_a - t)^2 \quad (7)$$

$$a_B(t) = 2C_1(t_a - t) \quad (8)$$

This mathematical representation of the S-curve velocity profile establishes a solid foundation for its implementation and comprehension. By providing a clear understanding of the relationships between acceleration, velocity, and displacement

throughout the motion, this framework empowers precise trajectory planning within mechatronic applications.

F. Pulse Generation Algorithms for S-Curve Motion Profiles

Precise pulse generation is crucial in mechatronic systems for achieving accurate positioning while minimizing vibrations and optimizing energy consumption. Two primary methodologies govern the calculation of pulse timing: "time per step" and "steps per time."

The "time per step" algorithms iteratively calculate the time interval until the next pulse is required. This involves calculating the waiting period, generating the pulse, and repeating until the desired number of pulses is reached. Conversely, "steps per time" algorithms operate by continuously checking the current time, calculating the expected position based on speed, and generating a pulse if the actual position deviates significantly from the expected one. This process repeats until the final target position is attained.

For segment A of the acceleration phase (governed by the displacement equation (3)), the inverse function (shown in equation (9)) is used to determine the delay time between the current step and the next step based on the current step count.

$$dt(s) = \sqrt[3]{\frac{3s}{C_1}} \tag{9}$$

However, segment B of the acceleration phase presents a challenge. The inverse function for equation (6) is mathematically difficult to calculate. In this scenario, the velocity equation (7) can be used as an alternative to determine the motion dynamics. Subsequently, the delay time between steps can be obtained using the "steps per time" algorithm.

G. Precise Stepper Motor Control with FreeRTOS

Precise stepper motor control underpins various mechatronic applications. However, managing the complexities of achieving this precision often necessitates running tasks concurrently. This is where FreeRTOS (Free Real-Time Operating System) shines as a powerful tool for efficient control.

FreeRTOS, a popular open-source real-time operating system designed for embedded systems, boasts a core strength: **preemptive scheduling**. This feature allows the system to automatically switch between tasks based on priority[12]. In this application, preemptive scheduling becomes particularly valuable because it enables:

1. Concurrent Task Execution: The overall control logic can be divided into independent tasks, each with a specific function. This modular approach simplifies development and enhances maintainability.

2. Non-Blocking Stepper Control: By separating the stepper motor control logic from other tasks, delays introduced by functions like wait or delay are avoided. This ensures uninterrupted and smooth motor operation.

By utilizing FreeRTOS and a mailbox for data exchange, the system achieves efficient, non-blocking control of the stepper motor while simultaneously collecting real-time data for performance evaluation. This decoupled approach ensures smooth motor operation and facilitates comprehensive performance analysis.

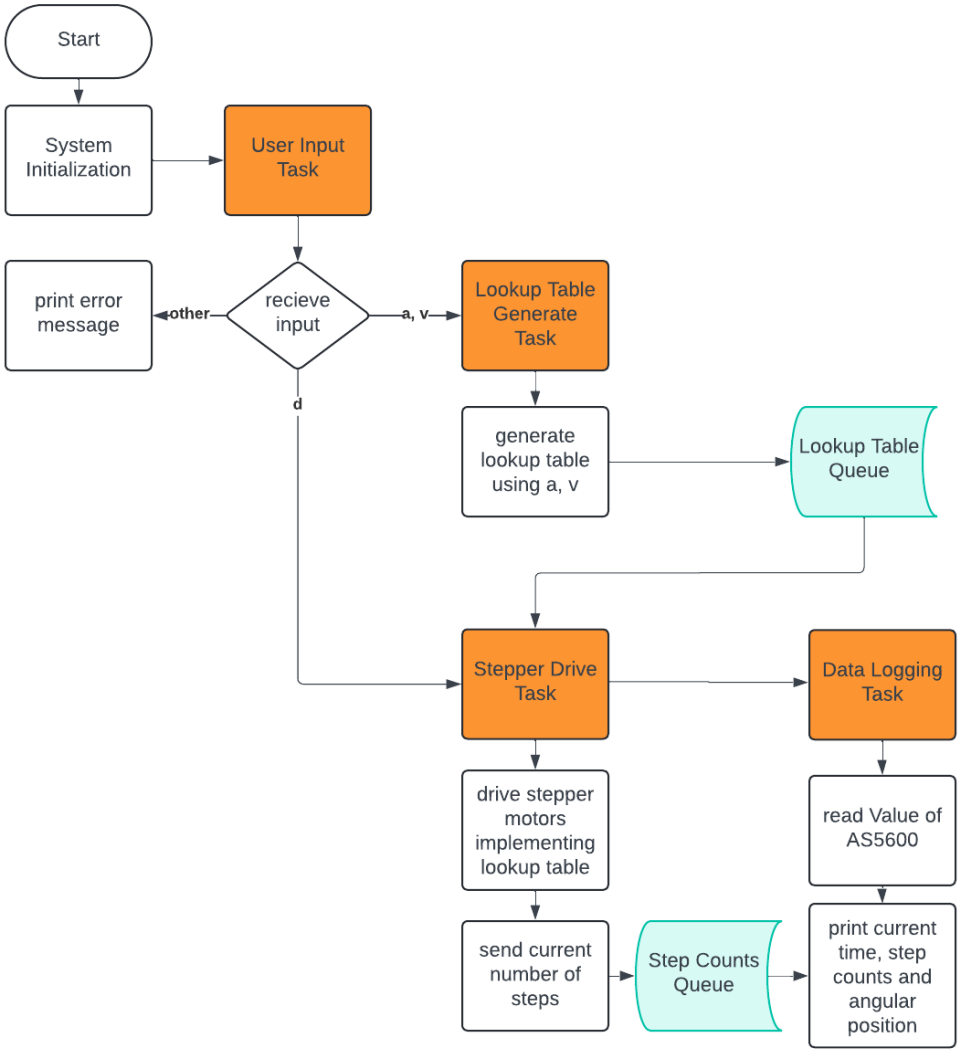


Figure 4. Flow diagram of the FreeRTOS program of the gantry robot

The flowdiagram in figure 4 depicts the execution sequence of the FreeRTOS program, which governs the precise control of stepper motors using pre-defined velocity and acceleration profiles. Here's a detailed breakdown of the steps:

- **System Initialization (Start):**
  - The system powers up, and the FreeRTOS program begins execution.
  - Essential components like hardware and tasks are initialized.
  - Two queues are created:
    - **lookup_table_queue**: To store calculated delay times for the acceleration phase segments.
    - **step_count_queue**: To transmit the current step count of the stepper motor(s).

- **Lookup Table Generation Task:**

This task employs the proposed pulse generation algorithm to calculate the delay times for each segment of the acceleration phase. These delay times are crucial for controlling the stepper motor's movement smoothly.

The calculated delay times are stored sequentially within the lookup_table_queue.

- **User Input Task:** This task waits for user input.
  - **User Input for desired velocity *v* and acceleration *a* values:**

Once received, the user input task triggers the lookup_table_generation_task.

  - **User Input for Distance *d*:**

The user specifies the desired travel distance *d* for the stepper motor(s). This distance will guide the motor movement within the defined velocity and acceleration parameters.

    - **Concurrent Task Activation:**

Upon receiving the travel distance *d*, the system concurrently initiates two tasks, **Stepper Drive Task** and **Data Logging Task.**

- **Stepper Drive Task:**

This task continuously:

Retrieves delay times from the lookup_table_queue.

Drives the stepper motor(s) using the retrieved delay times and the desired travel distance *d*.

Monitors and increments the accumulated step count of the stepper motor(s).

The updated step count is then sent to the step_count_queue.

- **Data Logging Task:**

This task executes concurrently with the stepper_drive_task.

It periodically:

  - Fetches the current step count from the step_count_queue.
  - Reads the instantaneous motor angle value from the AS5600 magnetic encoder.
  - Captures the current timestamp using system time functions.

The retrieved step count, motor angle value, and timestamp are then logged for later analysis and evaluation. This logged data provides insights into the motor's performance, potentially including factors like positioning accuracy, vibration levels, and efficiency.

H. Experimental Investigation For Precise Stepper Motor Control

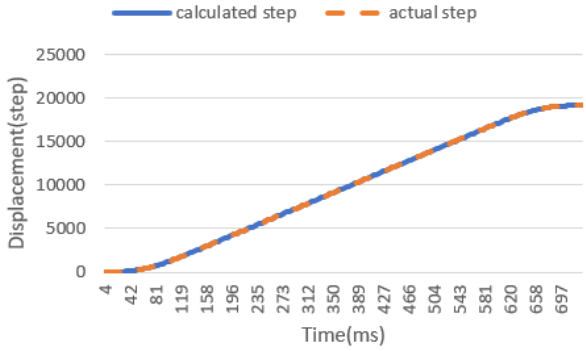
This section details an experimental investigation aimed at evaluating the impact of varying speeds and accelerations on stepper motor performance using the S-curve velocity profile. Understanding these factors is crucial for optimizing motion control across diverse applications, as discussed in the previous article.

Figure 5 depicts the experimental setup utilized in this investigation.

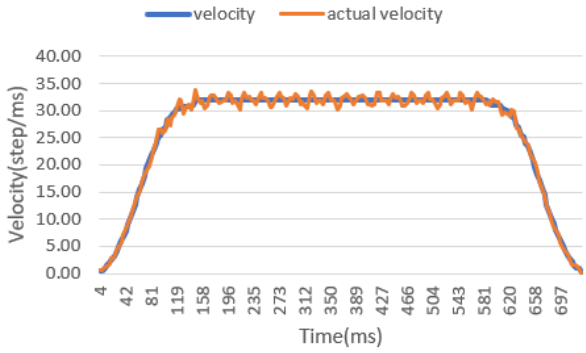
During the experiments, a constant acceleration of 0.4 steps/ms^2 was maintained. The stepper motor underwent test runs at speeds of 160 mm/s, 180 mm/s, and 200 mm/s, covering a distance of 100 mm in each test.

Figure 5. Experimental configuration of the gantry robot

I. Result And Discussion



(a)



(b)

Figure 6. Comparative analysis of calculated and actual displacement and velocity at a coasting velocity of 160 mm/s (32 step/ms)

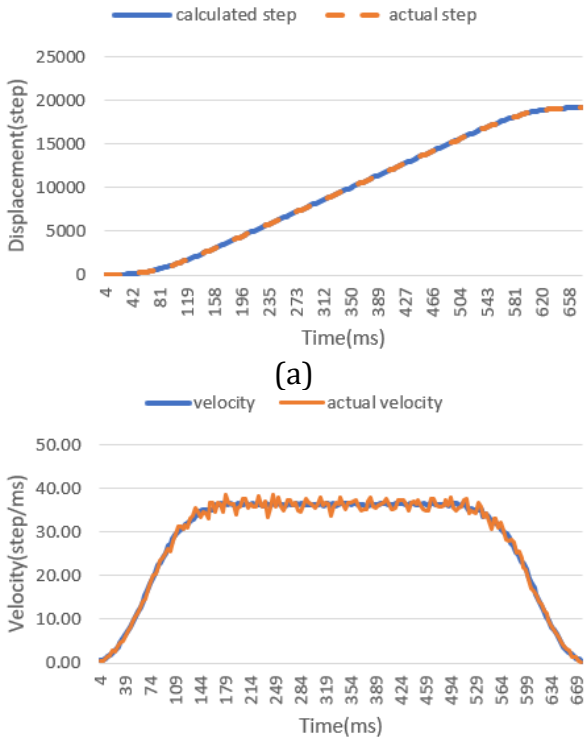


Figure 7. Comparative analysis of calculated and actual displacement and velocity at a coasting velocity of 180 mm/s (36 step/ms)

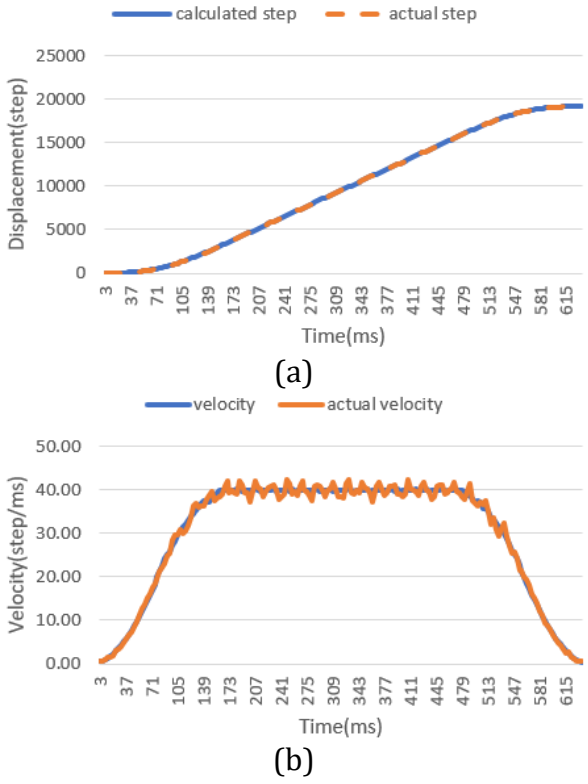


Figure 8. Comparative analysis of calculated and actual displacement and velocity at a coasting velocity of 200 mm/s (40 step/ms)

The comprehensive experimental evaluation involving various coasting velocities revealed sporadic occurrences of step losses. These losses manifested as a maximum deviation of two steps from the targeted position. Notably, initial slipping was observed at coasting velocities exceeding 200 mm/s.

J. Conclusion

This work presented a system for precise control of stepper motors within a gantry robot designed for delicate object handling. The system leverages S-curve motion profiles generated using a simple and efficient algorithm. This algorithm demonstrated commendable positional accuracy, making it suitable for various applications requiring smooth and precise movement. Notably, this implementation represents one of the simplest approaches for achieving the S-curve profile, offering a straightforward method for improved motor performance and smooth robot operation.

While the system currently operates as an open-loop control system, future investigations could explore the benefits of incorporating closed-loop control with real-time adjustments for enhanced accuracy. Additionally, the integration of additional sensors, such as accelerometers, could provide valuable data for comprehensive motion monitoring and potential performance optimization. Furthermore, transitioning from lead screws to a ball screw mechanism holds the potential to unlock higher speed capabilities, opening doors to further advancements in both system performance and a broader range of potential applications.

K. Acknowledgment

All the authors of the research paper are highly acknowledged to Mechanical Engineering Department, Mandalay Technological University, Myanmar for encouraging and supporting to do this research.

L. References

[1] Espresssif Systems., “ESP32.” Espresssif Systems., 2024. [Online]. Available: https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf

[2] Y. Zhang, B. Feng, and S. Sang, “Design of Intelligent Wireless Laser Engraving Machine,” *J. Phys.: Conf. Ser.*, vol. 2216, no. 1, p. 012005, Mar. 2022, doi: 10.1088/1742-6596/2216/1/012005.

[3] S. Tang and Y. Yu, “Research on Closed-loop Control of Step Motor Based on Magnetic Encoder,” *HSET*, vol. 1, pp. 351–356, Jun. 2022, doi: 10.54097/hset.v1i.486.

[4] M. Blejan, “Mathematics for Real-Time S-Curve Profile Generator,” *HIDRAULICA, Magazine of Hydraulics, Pneumatics, Tribology, Ecology, Sensorics, Mechatronics*, no. 4, pp. 7–25, 2020.

[5] A. Drumea, C. I. Marghescu, M. Pantazică, G. Jitianu, and A. Vlad, “S-CURVE MOTION CONTROL IMPLEMENTATION USING 32-BIT MICROCONTROLLER,” in *Proceedings of 2022 International Conference on Hydraulics and Pneumatics*, Băile Govora, Romania, 2022, pp. 160–165.

[6] M. Y. Stoychitch, "Generate stepper motor linear speed profile in real time," *IOP Conf. Ser.: Mater. Sci. Eng.*, vol. 294, p. 012055, Jan. 2018, doi: 10.1088/1757-899X/294/1/012055.
[7] J. R. García-Martínez, J. Rodríguez-Reséndiz, and E. E. Cruz-Miguel, "A New Seven-Segment Profile Algorithm for an Open Source Architecture in a Hybrid Electronic Platform," *Electronics*, vol. 8, no. 6, p. 652, Jun. 2019, doi: 10.3390/electronics8060652.
[8] V. Montalvo, A. A. Estévez-Bén, J. Rodríguez-Reséndiz, G. Macias-Bobadilla, J. D. Mendiola-Santíbañez, and K. A. Camarillo-Gómez, "FPGA-Based Architecture for Sensing Power Consumption on Parabolic and Trapezoidal Motion Profiles," *Electronics*, vol. 9, no. 8, p. 1301, Aug. 2020, doi: 10.3390/electronics9081301.
[9] T. Liu, J. Cui, Y. Li, S. Gao, M. Zhu, and L. Chen, "Time-Optimal Asymmetric S-Curve Trajectory Planning of Redundant Manipulators under Kinematic Constraints," *Sensors*, vol. 23, no. 6, p. 3074, Mar. 2023, doi: 10.3390/s23063074.
[10] H. Gurocak, *Industrial Motion Control*. John Wiley & Sons, Ltd, 2016.
[11] K. Yu, Z. Zhang, Z. Zhou, and M. Dai, "Application of the Five-phase S-curve Velocity Model on FDM Three-dimensional Printer," in *2020 IEEE 5th Information Technology and Mechatronics Engineering Conference (ITOEC)*, Chongqing, China: IEEE, Jun. 2020, pp. 1365–1371. doi: 10.1109/ITOEC49072.2020.9141587.
[12] W. Gay, *FreeRTOS for ESP32-Arduino: practical multitasking fundamentals*, 1. Auflage. London: Elektor International Media BV, 2020.