Proyecto: Circuito Vertical de Canicas con Comunicación por Protocolo RS232

Esteban Vargas Monge *2022208944*
Bernardo Chacon Montero *2022053512*
Emily Flores Rojas *2021475927*
Ismael Bermudez Mora *2022000119*

***Resumen*—Este proyecto presenta un sistema mecatrónico para el transporte vertical de canicas, compuesto por una estructura modular, motores a paso y control distribuido entre una Raspberry Pi y un microcontrolador STM32 mediante RS232. Se implementaron modos de operación manual y programado, con validación de movimientos y control en tiempo real usando interrupciones. Aunque algunas funciones automáticas no se completaron debido a limitaciones mecánicas, el sistema cumple los objetivos principales y demuestra un funcionamiento estable y coordinado entre hardware y software.**

***Index Terms*—Control Automático, Mecatrónica, Motores a Paso, Raspberry Pi, STM32, UART.**

I. Introducción

El desarrollo de sistemas mecatrónicos integrados requiere la convergencia de componentes mecánicos, electrónicos y de software capaces de trabajar en conjunto de forma precisa y confiable. Bajo este enfoque, se diseñó un sistema de transporte vertical de canicas cuyo propósito es desplazar una canasta dentro de una matriz mediante motores a paso controlados electrónicamente y coordinados a través de comunicación serial RS232. Este proyecto combina una estructura física construida con piezas Lego Technic —elegidas por su modularidad, disponibilidad y facilidad de modificación— junto con un sistema de control distribuido entre una Raspberry Pi y un microcontrolador STM32.

La Raspberry Pi funciona como unidad de alto nivel, ejecutando una interfaz gráfica en Python que permite seleccionar rutas, definir secuencias de movimiento y validar desplazamientos antes de enviarlos al microcontrolador. Por su parte, el STM32 se encarga del control en tiempo real mediante interrupciones, administrando los motores a paso, el servomotor y la comunicación UART sin bloquear el sistema. Esta arquitectura facilita la ejecución precisa de movimientos verticales y horizontales mientras se mantiene una comunicación continua y eficiente entre los dispositivos.

Para facilitar la colaboración y el desarrollo ágil, se creó un repositorio en GitHub. Esto resultó de gran utilidad para trabajar desde distintos dispositivos y enviar cambios a la Raspberry Pi rápidamente, permitiendo que todos los integrantes del grupo actualizaran el código de manera eficiente y centralizada.

El sistema ofrece tres modos de operación: uno manual, en el cual el usuario interactúa directamente con los controles direccionales, otro programado, que permite crear secuencias completas de movimientos automáticos y uno de calibración. Aunque algunas funcionalidades, como las zonas de carga automatizadas, no fueron implementadas por limitaciones en el diseño inicial, el proyecto logra integrar de manera efectiva los elementos esenciales de un sistema mecatrónico de transporte controlado.

II. Enfoque e implementación

II-A. Estructura Mecánica y Materiales

Para los requisitos mecánicos, se planteó una estructura parecida a la que tendría una impresora 3D, utilizando motores a paso, correas dentadas y llantas. Los materiales utilizados para el desarrollo del proyecto fueron:

- 3 motores a paso 28BYJ-48 con módulo controlador ULN2003.
- Estructura principal construida con piezas LEGO Technic.
- 1 Servomotor para la apertura de la canasta.
- 1 motor DC para el tornillo sinfín (elevador de canicas).
- 2 sensores ultrasónicos para el conteo de canicas.
- Raspberry Pi 5.
- STM32 Nucleo F446RE.

Los motores a paso fueron elegidos porque son actuadores digitales que tienen mejor precisión que un motor DC; cada impulso eléctrico mueve el motor a una posición o "paso" sin necesidad de un sensor que esté monitoreando constantemente [1]. Además, no tienen las limitaciones físicas que presentan muchos servomotores y su precio es bastante accesible.

Las correas dentadas se colocaron de forma vertical a ambos lados de la matriz principal de la estructura. Con los motores a paso, estas correas permiten mover la canasta hacia arriba o hacia abajo según se necesite. A este sistema se le añade una llanta que se desplaza horizontalmente sobre una barra unida a las correas, logrando así el movimiento lateral requerido.

Por razones de presupuesto y disponibilidad, se decidió construir la estructura usando piezas Lego Technic (Fig. 1). Esto resultó ideal por su facilidad de implementación, disponibilidad y capacidad de expansión.

II-B. Interfaz y Lógica (Raspberry Pi)

El software de la interfaz se desarrolló en Python y está pensado para ejecutarse en la Raspberry Pi. Este programa implementa los modos de movimiento y maneja la comunicación con el microcontrolador STM32. La clase principal, *MarbleInterfaceFinal*, contiene todo el comportamiento del sistema.

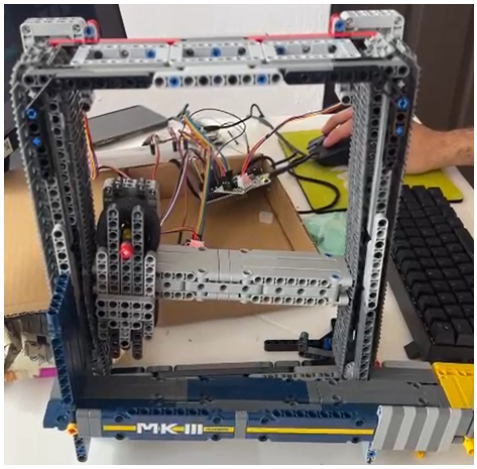

Figura 1. Estructura mecánica implementada.

II-B1. Modo Manual: En el modo manual, el usuario tiene botones de dirección. Cada vez que se presiona un botón, se valida si el movimiento es posible dentro de la matriz y se envía el comando.

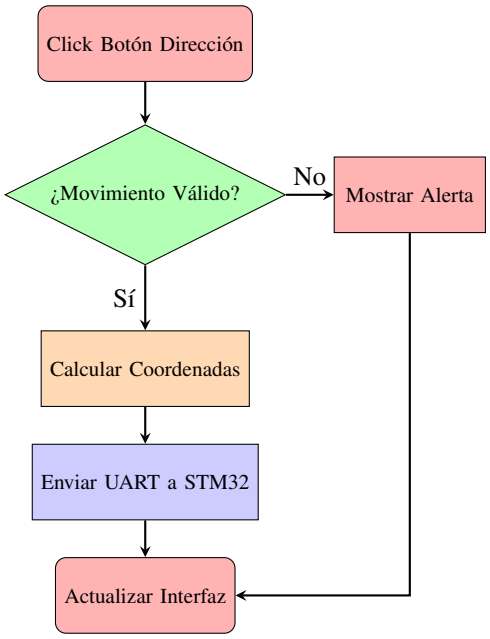

Figura 2. Diagrama de Flujo: Modo Manual.

II-B2. Modo Programado: En el modo programado, el usuario construye rutas seleccionando zonas. El sistema ejecuta automáticamente las acciones recorriendo la lista de pasos.

II-C. Firmware y Conexiones (STM32)

El STM32 trabaja en tiempo real, ejecutando los movimientos mientras la Raspberry Pi solo envía órdenes. Se utilizan interrupciones UART para recibir comandos sin bloquear el movimiento de los motores [2]. La conexión de pines utilizada se detalla a continuación:

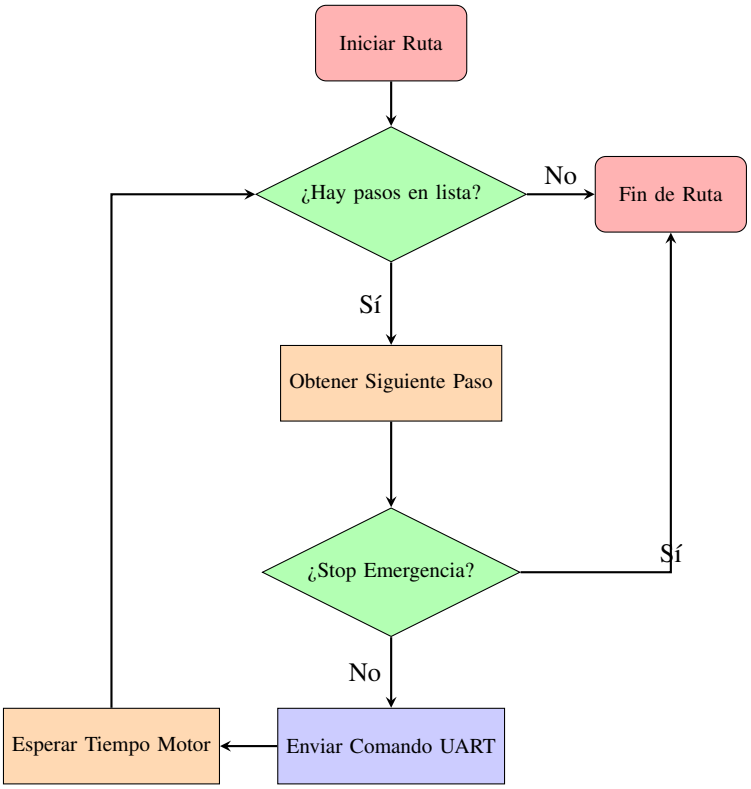

Figura 3. Diagrama de Flujo: Modo Programado.

Cuadro I
CONEXIÓN DE PINES ENTRE STM32 Y ACTUADORES

Actuador	**Función**	**Pines STM32**
Motor 0	Horizontal (X)	PA0, PA1, PA4, PB0
Motor 1	Vertical Izq.	PA5, PA6, PA7, PA8
Motor 2	Vertical Der.	PB4, PA10, PB3, PB1
Servomotor	Volcado	PB6 (TIM4_CH1)

III. ANÁLISIS

En este proyecto, las interrupciones en el STM32 resultaron indispensables. El sistema necesita que los motores a paso avancen en intervalos de tiempo muy específicos, y además se debe llevar el conteo de canicas y recibir comandos al mismo tiempo. Si se hubiera usado *polling*, podrían haberse perdido pasos o datos de comunicación.

Las pruebas de validación se realizaron inicialmente enviando señales individuales a cada motor para confirmar el cableado y la secuencia de pasos. Posteriormente, se probó el sistema en ”modo simulación”dentro del código, comprobando que la lógica de la interfaz enviaba los comandos correctos antes de conectar la potencia.

Durante las pruebas integrales, se comprobó que la parte eléctrica y el protocolo de comunicación funcionan correctamente sin latencia perceptible. Sin embargo, se identificó un problema mecánico importante: una pérdida gradual de altura en el eje vertical debido a la diferencia de tensión y fuerza entre las dos correas laterales. Aunque el código envía

la cantidad correcta de pasos, la física de la estructura provoca que la canasta no siempre alcance la altura teórica tras varios movimientos. En una iteración futura, se debería medir esta pérdida y modificar el código para añadir una compensación automática en la subida.

IV. Conclusiones

El desarrollo de este proyecto integró exitosamente componentes mecánicos, electrónicos y de software. El diseño final cumple con los requisitos solicitados y ofrece un funcionamiento estable.

La construcción con Lego Technic y motores a paso resultó ser una solución modular y accesible, aunque presentó retos mecánicos. La interfaz en la Raspberry Pi demostró ser intuitiva y robusta para controlar el sistema, mientras que el STM32 manejó con precisión los tiempos críticos mediante interrupciones, validando la arquitectura distribuida elegida para este sistema mecatrónico.

V. Recomendaciones

Si se realizara de nuevo este proyecto, se invertiría en motores con mayor torque (como NEMA 17) en lugar de los 28BYJ-48, para evitar pérdidas de pasos bajo carga y mejorar la velocidad. También se trabajaría con un modelo a mayor escala para facilitar la manipulación de componentes.

Finalmente, se recomienda mantener un orden estricto en la programación y utilizar control de versiones (como Git) desde el inicio del proyecto, ya que esto facilitó enormemente la integración del trabajo de los diferentes miembros del equipo.

Referencias

[1] V. V. Athani, *Stepper Motors: Fundamentals, Applications and Design*, Referencia técnica sobre motores paso a paso, 2014.

[2] *STM32F446xx Reference Manual*, Documentación oficial del protocolo y periféricos, STMicroelectronics, 2024.

[3] *STM32CubeIDE User Guide*, Guía de usuario del entorno de desarrollo integrado STM32CubeIDE, STMicroelectronics, 2024.

[4] *STM32 Microcontrollers Overview*, Información general y hojas de datos de la familia de microcontroladores STM32, STMicroelectronics, 2024.

[5] *Getting started with STM32Cube MCU Package and HAL*, Guía de uso de la capa de abstracción de hardware (HAL/LL) para STM32, STMicroelectronics, 2024.