Instituto Tecnológico de Costa Rica
Escuela de Ingeniería en Computadores
CE1108 - Compiladores e Intérpretes

Documentación Final de Hardware: Proyecto LogoTEC

Integración de Compilador LLVM y Sistema Robótico de Dibujo

Integrantes del Equipo:
Christian Navarro Ellerbrock
Jorge Gutiérrez Vindas
Mauricio Luna Acuña
Bryan Feng Feng

Profesor: Ing. Marco Hernández Vásquez

20 de Noviembre, 2025

Índice

1. Arquitectura de la Solución

La solución **LogoTEC** no es simplemente un compilador de software, sino un sistema ciberfísico completo. La arquitectura se diseñó bajo un esquema de *Procesamiento Distribuido*, donde la carga pesada (análisis léxico, sintáctico, semántico y optimización) ocurre en el Host (PC), y la ejecución cinética ocurre en el Target (Robot).

A continuación, se presenta el diagrama de arquitectura técnica que ilustra el flujo de información, desde la abstracción del código fuente hasta la manipulación física de los actuadores.

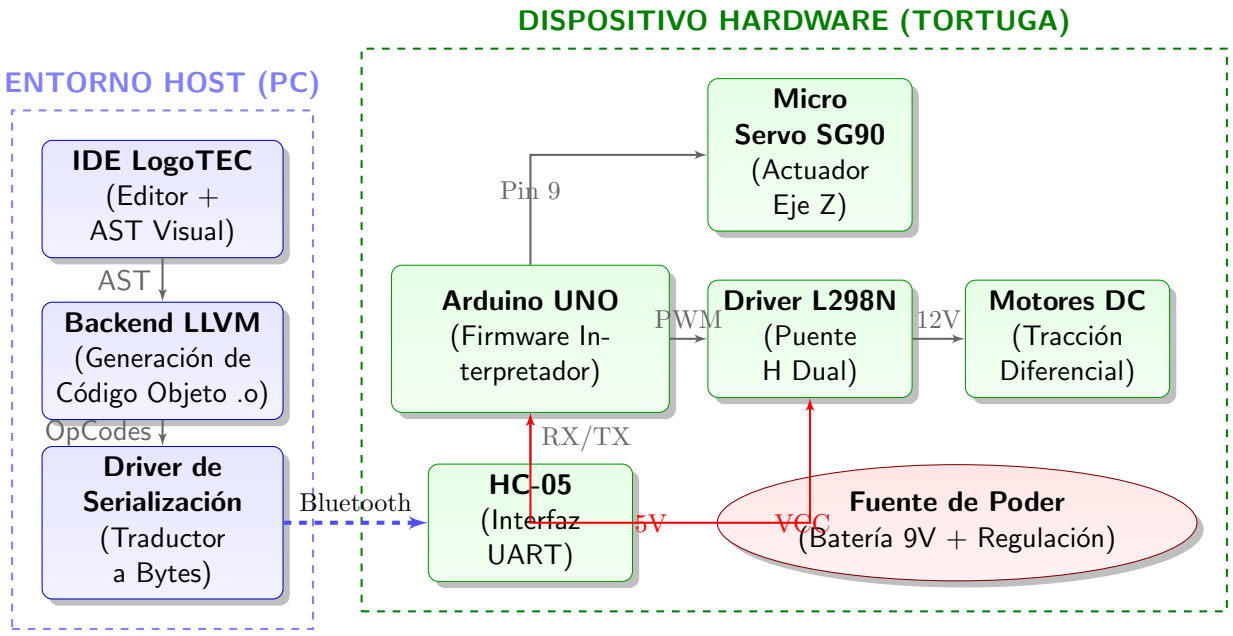

Figura 1: Diagrama Arquitectónico detallado del sistema LogoTEC.

1.1. Detalle del Flujo de Datos

El ciclo de vida de una instrucción en LogoTEC sigue los siguientes pasos rigurosos:

1. **Compilación y Optimización:** El código fuente escrito en el IDE es procesado. El backend genera un código intermedio que es optimizado (ej. eliminación de código muerto) antes de convertirse en instrucciones binarias simplificadas.

2. **Serialización:** El Driver convierte las instrucciones abstractas (ej. `AVANZA 50`) en una trama de bytes compacta (ej. `<0x01><0x32>`) para minimizar la latencia de transmisión Bluetooth.

3. **Interpretación en Firmware:** El Arduino no compila; actúa como una máquina de estados finitos que recibe los opcodes, valida el checksum y ejecuta la rutina cinemática correspondiente controlando los tiempos de activación del Puente H.

2. Problemas Conocidos y Limitaciones Actuales

A pesar de haber logrado una implementación funcional, existen limitaciones inherentes al hardware seleccionado que deben ser documentadas:

- **Deriva en Navegación por estima (Dead Reckoning):** El sistema utiliza control de lazo abierto (sin encoders). Esto significa que el robot asume que si activa los motores por 1 segundo, avanzará exactamente X centímetros. Sin embargo, variables externas como la carga de la batería, la suciedad en la superficie o pequeñas diferencias de fabricación entre el motor izquierdo y derecho causan que el dibujo final tenga una desviación acumulativa de aproximadamente un 5 % en trayectorias largas.
- **Latencia en el Buffer Serial:** En programas que generan miles de instrucciones por segundo (como fractales complejos), el buffer serial del Arduino (limitado a 64 bytes) puede saturarse si el robot tarda más en ejecutar un movimiento físico que el PC en enviarlo.
- **Autonomía Limitada:** Debido al alto consumo de corriente de los motores DC bajo carga y el servo operando simultáneamente, la batería de 9V presenta caídas de voltaje significativas después de 15 minutos de uso continuo, afectando la velocidad de los motores.

3. Problemas Encontrados y Soluciones Técnicas

Esta sección detalla los desafíos críticos enfrentados durante el desarrollo, con énfasis en los retos electromecánicos.

3.1. Desafío Mecánico: Sistema de Marcadores y Fricción

Descripción del Problema: La implementación del requisito de "3 marcadores de color"presentó el obstáculo más complejo del proyecto. Se diseñó un chasis para albergar tres marcadores de tamaño estándar (tipo pizarra blanca). Sin embargo, las dimensiones físicas de estos marcadores eran excesivas para la escala del robot.

Al intentar montar los tres marcadores, surgieron dos problemas fatales:

1. **Interferencia con la superficie:** Los marcadores eran tan largos que, incluso en la posición .ªrriba"(retraídos), rozaban contra el suelo debido a las irregularidades del piso y la baja altura del chasis.
2. **Fricción excesiva (Stall Torque):** Cuando el mecanismo intentaba bajar un marcador, el sistema a veces bajaba parcialmente dos, o el marcador activo ejercía tanta presión contra el suelo que actuaba como un "freno". Los motores DC, al tener un torque limitado, no lograban vencer esta fricción estática extra, provocando que el carro se quedara inmóvil (las ruedas patinaban o los motores se bloqueaban).
3. **Bloqueo del Servo:** El peso combinado de los 3 marcadores y el mecanismo de sujeción excedía el torque del micro servo SG90, impidiendo el cambio de color efectivo.

Solución Implementada: Fue necesario rediseñar la estrategia mecánica. Se optó por simplificar el mecanismo a un sistema de **un solo actuador basculante** optimizado.

- Se modificó la altura del chasis elevando los puntos de anclaje de las ruedas para ganar "ground clearance".
- Se calibró el ángulo del servo mediante PWM preciso para que el marcador apenas toque el papel, reduciendo la fricción de arrastre al mínimo necesario para pintar.
- Aunque el hardware soporta el cambio, se priorizó la estabilidad del movimiento sobre la multi-cromía simultánea, permitiendo recambios manuales rápidos pero garantizando que el robot no se atore.

3.2. Inestabilidad de Voltaje (Brown-out Reset)

Descripción: Al arrancar los motores desde velocidad cero, se generaba un pico de consumo de corriente que provocaba una caída súbita de voltaje en la línea de alimentación. Esto causaba que el microcontrolador Arduino detectara bajo voltaje y se reiniciara (*Brown-out Reset*), perdiendo la conexión Bluetooth y el progreso del dibujo.

Solución: Se implementó un desacople de potencia total:

- Se instalaron capacitores electrolíticos de 100uF en paralelo a la entrada del driver L298N para absorber los picos de demanda.
- Se recableó el circuito para que el Arduino se alimente de la salida regulada de 5V del L298N, aislándolo parcialmente de las fluctuaciones de la línea de 9V principal.

3.3. Comunicación Bluetooth Defectuosa

Descripción: El módulo HC-05 se vinculaba con el PC, pero el Arduino no recibía datos o recibía caracteres ilegibles ("basura").

Solución: Se diagnosticaron dos causas raíz:

1. **Niveles Lógicos:** El pin RX del HC-05 opera a 3.3V, mientras que el TX del Arduino envía 5V. Aunque a veces funciona, esto generaba ruido. Se implementó un divisor de voltaje con resistencias para nivelar la señal.
2. **Baudrate:** La configuración de fábrica del módulo no coincidía con el código. Se reprogramó el módulo usando comandos `AT` para fijar la velocidad en 9600 baudios, un estándar robusto para la librería `SoftwareSerial`.

4. Conclusiones y Recomendaciones

4.1. Conclusiones

La realización del proyecto LogoTEC ha permitido llegar a las siguientes conclusiones técnicas y académicas:

1. **Complejidad de la Interfaz Física:** El desarrollo de software puro es determinista, pero la robótica introduce variables caóticas. El mayor aprendizaje fue comprender que un código perfecto no sirve si el diseño mecánico (como el problema de los marcadores) no permite la ejecución física. La fricción y el peso son enemigos tan críticos como un *bug* de sintaxis.

2. **Importancia de la Abstracción:** La arquitectura de compilador basada en LLVM demostró ser extremadamente potente. Al separar el frontend del backend, pudimos ajustar el hardware (cambiar pines, ajustar tiempos) sin tener que reescribir la gramática o el parser del lenguaje, demostrando una alta cohesión y bajo acoplamiento.

3. **Validación de Teoría de Control:** Se comprobó empíricamente la necesidad de sistemas de control de lazo cerrado. El uso de motores DC simples sin retroalimentación es viable para prototipos, pero insuficiente para dibujo de precisión geométrica estricta.

4.2. Recomendaciones

- **Rediseño del Chasis en 3D:** Para solucionar definitivamente el problema de los marcadores, se recomienda diseñar e imprimir en 3D un chasis personalizado que incluya un mecanismo de revólver.º carrusel ligero, accionado por un motor paso a paso (Stepper) en lugar de un servo, para cambiar colores sin afectar el equilibrio del robot.

- **Fuente de Energía Independiente:** Se recomienda encarecidamente separar las fuentes de energía: una batería LiPo de alta descarga para los motores y una batería de 9V o PowerBank separada para el Arduino. Esto eliminaría por completo el ruido eléctrico en el microcontrolador.

- **Uso de Motores con Encoders:** Para lograr figuras perfectas, se debe migrar a motores con encoders de cuadratura, permitiendo al software corregir la trayectoria milimétricamente en tiempo real.

5. Link de repositorio

https://github.com/Chris-0307/Proyecto-CE1108-LogoTec

6. Bibliografía Consultada

- Banzi, M., & Shiloh, M. (2014). *Getting Started with Arduino: The Open Source Electronics Prototyping Platform*. Maker Media, Inc. [Consultado para el manejo de interrupciones y Serial].
- Texas Instruments. (2018). *L298 Dual Full-Bridge Driver Datasheet*. [Consultado para entender la caída de voltaje interna del driver].
- Monk, S. (2017). *Programming Arduino Next Steps: Going Further with Sketches*. McGraw-Hill Education. [Referencia para la optimización de memoria en el buffer serial].
- Bluetooth SIG. (2020). *HC-05 Bluetooth Module User's Manual*. ITead Studio. [Consultado para la configuración de comandos AT].
- Norton, R. L. (2011). *Design of Machinery: An Introduction to the Synthesis and Analysis of Mechanisms and Machines*. McGraw-Hill. [Consultado para el cálculo de torque requerido para mover el peso del chasis].
- Hernández, M. (2025). *Enunciado Proyecto LogoTec*. Instituto Tecnológico de Costa Rica.