Instituto Tecnológico de Costa Rica

Módulo de Compiladores

Proyecto: Tambarduine

Profesor:

Marco Hernandez Vazques

Estudiantes:

Rachel Pereira González 2020186432

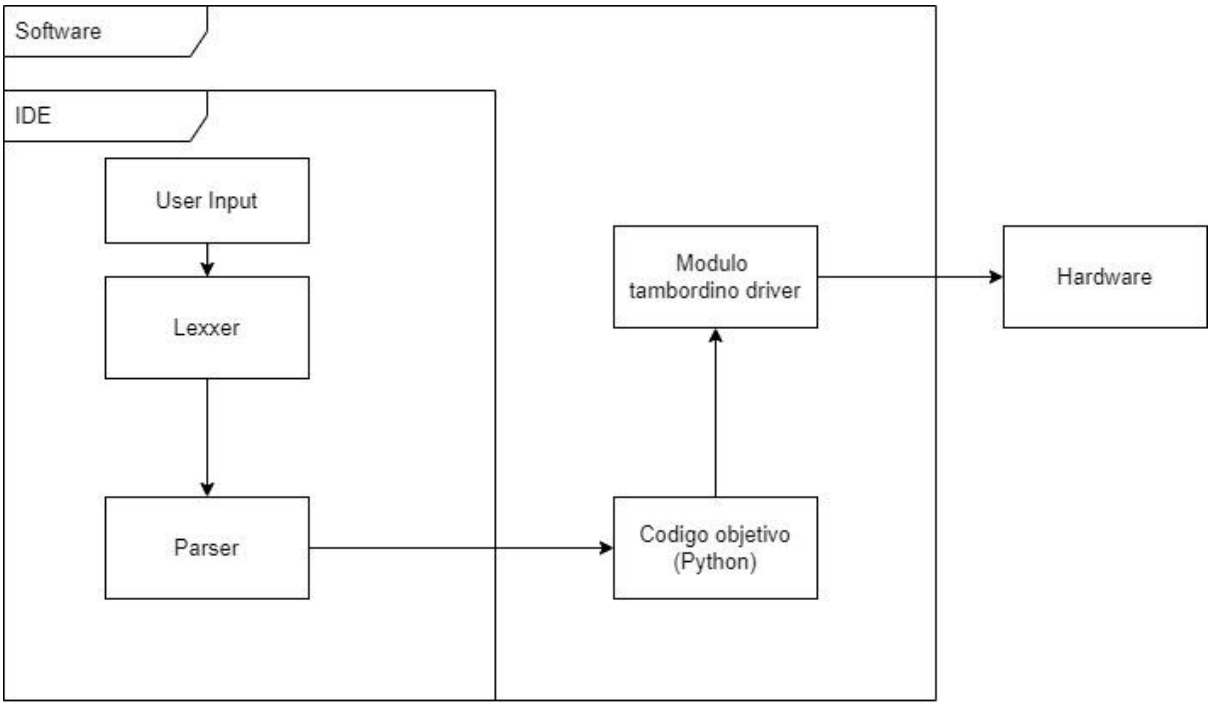
Isaac Barrios Campos 2020425901

Bryan Gomez Matamoros 2020211778

Daniel Zúñiga Barahona 2019043724

3 abril, 2022

Diagrama de arquitectura



Alternativas de solución y solución seleccionada

Hardware:

Para el hardware, se optó inicialmente por el microcontrolador Arduino UNO, pero al final se utilizó el Wemos D1 R1 WiFi, controlado utilizando el lenguaje de Arduino y la biblioteca de adaptación para placas ESP8266. El Wemos ofrece mayor comodidad al trabajar con comunicación serial, además de mayor cantidad de puertos PWM.

El equipo consta de una pandereta de cartón, dos servo motores MG995 para agitar la pandereta, cinco servo motores MG90S para los percutores y un LED blanco que cumple la función del metrónomo.

La comunicación entre el microcontrolador y el programa se realiza por medio de comunicación serial, a través de un puerto USB.

Software:

Para el software se eligió el uso de python como lenguaje de programación ya que el mismo es un lenguaje de fácil uso y con la capacidad de ser mutable, lo que nos permitiría agregar parámetros según se requiriese en tiempo de ejecución.

Para el lexxer y el parser se seleccionó un sabor de lex y yac conocido como PLY, otras opciones que se estuvieron planteando fue la de hacer uso de SLY, pero se escogió PLY ya que contaba con mayor documentación oficial, de la cual se hizo uso para su respectivo aprendizaje.

Para el IDE se seleccionó como módulo gráfico Tkinter para la interfaz de usuario, otra opción planteada fue pygame pero esta se descarto Tkinter ofrece más facilidades pertinentes para el proyecto.

Para el manejo de archivos, dícese el cargado y guardado de código, se utilizaron respectivas funciones basadas en la escritura y lectura de archivos de texto (.txt), esto mediante las opciones sobre el manejo de archivos planos que python maneja de base.

Para la comunicación Hardware-Software se eligió una comunicación serial.

Para la detección de errores Sintácticos se pretende realizar una función más que indicará con mayor detalle el porqué del error, pero se decide utilizar una más sencilla ya que la más específica por diferentes inconvenientes que no se utilizó.

Problemas conocidos:

1- Al hacer scroll sobre una de las áreas de texto del IDE es posible que se des-sincronice por unos milímetros al llegar a la parte superior o inferior, es posible volver a sincronizarlo haciendo scroll en la parte des-sincronizada

2- La gramática libre de contexto contiene una regla ambigua (98) que aunque no afecta al momento de llegar a un resultado correcto, sí muestra un error en el compilador al solucionarse la ambigüedad.

3- Al enviar gran cantidad de instrucciones al microcontrolador en un intervalo corto de tiempo (menor a 0.3 segundos), la ejecución puede verse interrumpida, dado que el buffer del puerto serial se llena, incluso al hacer clear y flush.

Actividades realizadas por estudiante: Bitácora

Fecha	Encargado	Actividad	Tiempo Invertido
26/2	Rachel, Bryan, Isaac, Daniel	Primera reunión	30m
6/3	Bryan, Isaac	Reunión para hablar acerca de las tareas y actividades a realizar relacionadas con el análisis léxico y sintáctico	30m
6/3	Isaac	Instalación de librería y prueba de la misma	1h30m
6/3	Rachel	Creación de la ventana básica del IDE y inicio de la lógica	2h
8/3	Isaac	IF ELSE reglas en gramática libre de contexto	2h
8/3	Rachel	Pruebas sobre la lógica de IDE	3h
8/3	Isaac	Escritura de Gramática libre de contexto para análisis de síntesis	1h
9/3	Rachel	Adaptación de la lógica básica creada a lo requerido por el Lex	1h30m
9/3	Daniel	Planificación y diseño inicial de hardware	2h30m
11/3	Rachel	Finalización de la lógica del IDE con sus pruebas necesarias	2h
12/3	Isaac	Reedición de gramática libre de contexto, integración de lexxer y parser y inicio de trabajo en scopes	6h
12/3	Bryan, Isaac	Reunión sobre revisión de avances relacionados con los análisis gramaticales	1h
12/3	Bryan	Estudio de la documentación sobre PLY	1h10m
12/3	Bryan	Trabajo de la primera versión del Lexxer para el proyecto	3h
12/3	Bryan	Adición de nuevos Tokens y funciones para el Lex	2h
12/3	Daniel	Compra de equipo a utilizar	1h
13/3	Rachel	Creación de cuadro de impresión en consola	2h
13/3	Isaac	Término de reescritura de gramática para coincidir con Lexxer, inicio de manejo de errores	4h30m
14/3	Rachel	Construcción de la pandereta	2h
14/3	Daniel	Primer prototipo y pruebas con motores	4h
15/3	Rachel	Creación de la una barra de numeros básica	2h
16/3	Rachel	Agregar más características a la barra de número y detalles de diseño	2h
16/3	Bryan	Prototipo de función de reconocimiento de Scopes	1h30m
17/3	Isaac	Script de traducción de código fuente a python usando el ast	4h
17/3	Isaac	Lógica de funcionamiento de los scopes	3h
20/3	Rachel	Revisión sobre todo lo relacionado con la Gramática libre	4h

		de contexto para detectar los errores Sintácticos y la instalación de las librerías necesarias	
21/3	Daniel	Desarrollo de Comunicación Serial	3h
22/3	Isaac	Inicio de análisis semántico	8h30m
22/3	Rachel	Pruebas de una posible función de detección de errores	2h
25/3	Bryan	Funcionalidad del manejo de archivos (Carga/Guardado)	1h30m
26/3	Isaac	Bug fixes del análisis semántico y comprobaciones con test unitarios	4h
27/3	Rachel	Prueba de adaptabilidad de la función de errores al proyecto actualizado	3h
29/3	Rachel	Integración de las funciones para el File Management al IDE	2h
30/3	Rachel	Corrección de error que se presenta en la barra de números	2h
31/3	Rachel, Bryan, Isaac, Daniel	Reunión para analizar el estado del proyecto y aclarar últimos detalles	30m
2/4	Daniel	Montaje de hardware final, pruebas y correcciones	6h

Problemas encontrados y solucionados:

1- Al momento de crear el ast no era posible también agregar las órdenes necesarias para el análisis semántico ya que se perdía el scope donde se encontraba el singleton al entrar en un exec(). Se soluciona eliminando el singleton y manteniendo el scope manualmente por medio de la mutabilidad de python. Se realizan tres intentos de solución por medio de singleton, hasta identificar el problema correctamente, ya que solucionarlo directamente podría cambiar el comportamiento de la librería que lo causaba se decide cambiar el enfoque entregando una referencia a los parámetros globales que entrarán en el exec().
Bibliografia del problema: https://docs.python.org/3/library/functions.html#exec

2- Declaración de parámetros fuera de scope ya que su primera aparición era antes de la declaración del scope donde se usaría. Se soluciona creando un Scope especial que maneja estos casos, ya que sus errores crearán error en tiempo de ejecución no pertenecientes al compilador. Se soluciona en el segundo intento. Aunque la solución evitaría que se den los problemas que después se comprobaría en el análisis semántico para tratar las variables como parámetros, se recomienda mantener los checks correspondientes como segunda comprobación. No se consulta bibliografía al ser un problema de implementación interna

3- Al momento de reconocer palabras reservadas había un problema con aquellas que aparecían de una manera diferente a su respectivo token, por ejemplo aquellas escritas únicamente en minúscula, pero esto se soluciona con la utilización del método upper que Python brinda. Se soluciona al primer intento.

4-Al realizar pruebas en el programa se observa que la funcionalidad de algunos botones como el Undo, Redo y Clear estaban produciendo errores en la barra de número, por lo que en el primer intento de solución luego de un análisis del procedimiento de cada función se logra observar que se estaba llamando a una función que provocaba estos errores, entonces se crea una función aparte para sea llamada en Undo y Redo sin afectar la funcionalidad de botón Clear.

5- La comunicación serial entre el microcontrolador y el programa se maneja a tiempo real (una vez el programa se compila y ejecuta, envía las instrucciones al microcontrolador), esto ocasiona que el tiempo de respuesta sea alto. Para disminuirlo, se aumentó el baudrate a 115200 baudios y se cambió el microcontrolador de un Arduino UNO a un Wemos D1 R1, el cual ofrece un poder de procesamiento mayor. Sería ideal aumentar aún más el baudrate, pero esto implica una comunicación más inestable.

Conclusiones y Recomendaciones del proyecto

1- Estandarizar el proceso por el que se van creando los subárboles de un árbol AST permite simplificar altamente los posteriores procesos en el que se use el AST.
2- Es posible modificar parcialmente el orden en que se van realizando los análisis de un compilador para así simplificar pasos posteriores.
3- La programación POO permite una baja cohesión, ya que cada módulo del proyecto puede trabajar de forma independiente, pero a su vez ha generado problemas al comprobar funcionamientos de todas las partes.
4- La implementación de workflows y CLI permite un aumento en la detección de bugs y del tiempo efectivo dedicado al proyecto.
5- Es posible trabajar aspectos de programación en otras herramientas donde resulta más simple la modificación gracias a la integración continua, como se realizó con la gramática libre de contexto.

Bibliografía

Python Software Foundation,(2022). *Python 3.10.4 Documentation*. Tomado de: 3.10.4 Documentation (python.org) el 25/03/22

Beazley D,(2020). *PLY (Python Lex-Yacc)*. Tomado de: PLY (Python Lex-Yacc) (dabeaz.com) el 25/02/22

LINK DE GITHUB:
isriom/Proyecto-compiladores-Tambarduine (github.com)

Link de la gramática libre de contexto
Gramatica libre de contexto - Hojas de cálculo de Google