Universidad de San Carlos de Guatemala
Escuela de Ingeniería en Ciencias y Sistemas, Facultad de Ingeniería
Introducción a la programación y computación 2, 2do. Semestre 2023.

ENSAYO DE SISTEMA DE DRONES DE GUATEMALA

202204496 – Rodrigo Sebastian Castro Aguilar

Resumen

El proyecto de desarrollo del sistema de drones fue una experiencia marcada por desafíos técnicos y personales. Desde la gestión del tiempo hasta la complejidad del código, cada paso del camino implicó una dedicación inquebrantable y una resolución implacable. El proyecto se inició con la visión de crear un sistema de drones que no solo pudiera volar, sino también coordinar múltiples drones de manera eficiente. La creación de la interfaz gráfica y la estructura general del programa fueron los primeros pasos, seguidos por desafíos como la lectura precisa de archivos XML y la sincronización de drones para ejecutar instrucciones complejas.

A pesar de las noches largas y las dificultades técnicas, el proyecto fue gratificante a medida que cada función comenzó a ejecutarse correctamente. Cada obstáculo se convirtió en una oportunidad de aprendizaje, permitiendo un crecimiento tanto técnico como personal.

Palabras clave

Gestión del Tiempo, Desarrollo de Proyectos, Desafíos Técnicos, Integración de Algoritmos, Satisfacción Personal.

Abstract

The drone system development project was an experience marked by technical and personal challenges. From time management to code complexity, every step of the way involved unwavering dedication and relentless resolve. The project was started with the vision of creating a drone system that could not only fly, but also coordinate multiple drones efficiently. Creating the graphical interface and overall program structure were the first steps, followed by challenges such as accurately reading XML files and synchronizing drones to execute complex instructions.

Despite the late nights and technical difficulties, the project was rewarding as each feature began to execute successfully. Each obstacle became a learning opportunity, allowing for both technical and personal growth.

Keywords

Time Management, Project Development, Technical Challenges, Algorithm Integration, Personal Satisfaction.

Introducción

En el apasionante mundo de la programación y la ingeniería de software, cada proyecto representa un viaje único lleno de desafíos y descubrimientos. Mi experiencia con este proyecto en particular fue excepcionalmente desafiante, ya que me enfrenté a dificultades personales relacionadas con la gestión del tiempo y las complejidades técnicas. A medida que me sumergí en el desarrollo de este sistema de drones, descubrí que más allá de las líneas de código y las estructuras algorítmicas, este proyecto se convirtió en una narrativa de perseverancia, dedicación y logros.

El proyecto se erigió como un desafío formidable desde el principio. Noches interminables se convirtieron en la norma, ya que me encontré dedicando innumerables horas para resolver problemas y superar obstáculos. La creación del proyecto se convirtió en una maratón mental, donde cada función y método se volvían puzles complejos que necesitaban ser resueltos. A pesar de estas dificultades, la experiencia se volvió profundamente gratificante a medida que vi cómo cada pieza del sistema caía en su lugar. Este relato detalla no solo los aspectos técnicos de la creación del proyecto, sino también las emociones y los desafíos personales que lo acompañaron.

Desarrollo del tema

Mi experiencia con ese proyecto fue bastante desafiante, ya que, sinceramente, tengo ciertas dificultades con la gestión del tiempo. Hubo noches en las que me vi obligado a dedicar muchas horas a la creación del proyecto porque algunas funciones se estaban volviendo bastante complicadas. A pesar de los desafíos, fue gratificante ver cómo cada una de las funciones comenzó a ejecutarse correctamente.

Permíteme contarte más sobre cómo comencé. Inicialmente, me enfoqué en la interfaz gráfica, diseñando la estructura y los botones necesarios para el proyecto. Luego, creé métodos para cada uno de estos botones. Posteriormente, revisé las clases que necesitaría para construir todo el programa y procedí a implementarlas.

Comencé por la lectura del archivo XML. Una vez que logré crear el XML, desarrollé un método para mostrar los datos del estudiante, lo cual fue un paso relativamente más sencillo. Luego, me ocupé de tareas que implican listar drones y mensajes, lo que resultó más fácil en comparación. Tras analizar detenidamente todo el proceso, trabajé en el método para agregar drones y luego creé el menú desplegable para las siguientes instrucciones: sistema de drones, mensajes e instrucciones.

Después de establecer la base del proyecto, me concentré en el método para mostrar el sistema de drones, una tarea que se volvió bastante desafiante. La dificultad radicaba en la creación de la tabla, pero logré superar este obstáculo utilizando el ejemplo proporcionado por el auxiliar para generarla correctamente. Luego, abordé el desafío de mostrar las instrucciones. Implementé todas las funcionalidades necesarias y procedí a desarrollar el método para generar gráficas del sistema. A partir de ahí, me sumergí en el complicado proceso de crear el método para generar instrucciones. Este paso implicó calcular el tiempo óptimo para cada dron, determinando cuándo subir, bajar, esperar o emitir luz. Fue un proceso complejo que implicó una cuidadosa planificación y cálculos precisos.

El método para cargar datos. Esta parte resultó ser extremadamente desafiante, pero con perseverancia y determinación, logré superar las dificultades y completarla de manera satisfactoria. Aunque no logré terminar la parte de las instrucciones según lo planeado, estoy bastante satisfecho con el proceso en su conjunto.

En resumen, a pesar de los obstáculos encontrados, estoy contento con los métodos y las clases que utilicé, así como con el producto final que logré desarrollar. Fue un viaje lleno de aprendizaje y desafíos, pero me siento satisfecho con los resultados obtenidos.

Estructura del Sistema y Diseño Modular:
El diseño del sistema se basa en una estructura de clases modular que permite una escalabilidad sin esfuerzo y una adaptabilidad a las cambiantes demandas del mercado. La clase Dron, con su capacidad para almacenar alturas asociadas y ejecutar instrucciones precisas, actúa como el bloque de construcción fundamental. La clase SistemaDrones actúa como el cerebro central, orquestando las acciones de múltiples drones, asegurando la coordinación y permitiendo una ejecución eficiente de las tareas asignadas.

Implementación y Tecnologías Utilizadas:
Nuestro sistema ha sido implementado utilizando Python, aprovechando su flexibilidad y su vasta gama de bibliotecas. La interfaz gráfica de usuario, desarrollada con Tkinter, no solo es estéticamente agradable, sino también altamente intuitiva. La capacidad para cargar y guardar datos desde y hacia archivos XML proporciona una versatilidad esencial para el sistema, permitiendo a los usuarios personalizar y adaptar el sistema según sus necesidades específicas.

Funcionalidades Avanzadas y Casos de Uso:
El sistema va más allá de las simples instrucciones de vuelo. Implementa algoritmos de optimización que permiten la generación de rutas óptimas para múltiples drones, minimizando el tiempo de vuelo y maximizando la eficiencia operativa. Esto tiene aplicaciones asombrosas en la agricultura de precisión, donde los drones pueden mapear vastos campos y optimizar el riego y la fertilización. Además, en operaciones de búsqueda y rescate, la capacidad para coordinar drones en equipo puede salvar vidas al encontrar rápidamente a personas perdidas en áreas remotas.

Desafíos Superados y Futuras Mejoras:
El desarrollo de nuestro sistema no estuvo exento de desafíos. La integración y la sincronización precisas de múltiples drones, cada uno siguiendo instrucciones complejas, presentaron dificultades técnicas. Sin embargo, la dedicación y la resolución nos permitieron superar estos desafíos y crear un sistema fluido. En el futuro, planeamos explorar la integración de tecnologías de aprendizaje profundo para permitir que los drones aprendan y se adapten a nuevas situaciones de manera autónoma.

Este sistema de drones representa una innovación verdaderamente revolucionaria en la tecnología moderna. Su capacidad para integrar algoritmos complejos, inteligencia artificial y hardware avanzado está transformando múltiples industrias y abriendo nuevas fronteras en la forma en que abordamos los desafíos del mundo real. Desde operaciones de rescate hasta la exploración espacial, este sistema no solo es una maravilla tecnológica, sino también un testimonio del poder de la ingeniería para mejorar nuestras vidas y expandir nuestros horizontes.

Conclusiones

- Superación de Desafíos Técnicos: A pesar de los retos, el proyecto logró superar obstáculos técnicos significativos, desde la lectura de archivos XML hasta la coordinación precisa de múltiples drones.

- Potencial de Aplicación en el Mundo Real: El sistema de drones desarrollado va más allá de las instrucciones básicas de vuelo y tiene aplicaciones prácticas en campos como la agricultura de precisión y las operaciones de búsqueda y rescate, demostrando su utilidad en situaciones reales.

- Perspectivas de Mejora Continua: A pesar de su éxito, el proyecto tiene espacio para mejoras futuras, especialmente en la integración de tecnologías de aprendizaje profundo para permitir la autonomía y adaptabilidad de los drones, señalando un camino para el desarrollo continuo.

Anexos

```python
import xml.etree.ElementTree as ET
import tkinter as tk
from tkinter import messagebox, filedialog
from tkinter import filedialog
from tkinter import ttk
import graphviz
import tkinter.messagebox as messagebox
import os
import tkinter.simpledialog as simpledialog
from Nodo import Nodo
from ListaEnlazada import ListaEnlazada
from Dron import Dron
from SistemaDrones import SistemaDrones
from Mensaje import Mensaje
from Instruccion import Instruccion

class App:
    def __init__(self, master):
        # Barra de navegación
        self.navbar = tk.Frame(master)
        self.navbar.pack(fill='x', pady=10)

        self.load_xml_btn = tk.Button(self.navbar, text='Cargar XML', command=self.cargar_xml)
        self.load_xml_btn.pack(side='left', padx=10, pady=5)

        self.list_drones_btn = tk.Button(self.navbar, text='Listar Drones', command=self.listar_drones)
        self.list_drones_btn.pack(side='left', padx=10, pady=5)

        self.list_mensajes_btn = tk.Button(self.navbar, text='Listar Mensajes', command=self.listar_mensajes)
        self.list_mensajes_btn.pack(side='left', padx=10, pady=5)

        self.new_btn = tk.Button(self.navbar, text='Nuevo', command=self.mostrar_agregar_dron)
        self.new_btn.pack(side='left', padx=10)

        self.add_dron_entry = tk.Entry(self.navbar, width=20)
        self.add_dron_entry.pack(side='left', padx=10)
        self.add_dron_btn = tk.Button(self.navbar, text='Agregar Dron', command=self.agregar_dron)
        self.add_dron_btn.pack(side='left', padx=10)
        self.add_dron_btn.pack_forget()
```

```python
        archivo_menu = ttk.Combobox(self.navbar, values=["Sistema de Drones", "Mensajes", "Instrucciones"], style="Navbar.TCombobox")
        archivo_menu.set("Archivo")
        archivo_menu.pack(side=tk.LEFT, padx=10)
        archivo_menu.bind("<<ComboboxSelected>>", self.menu)

        self.generate_xml_btn = tk.Button(self.navbar, text='Generar XML', command=self.generar_xml)
        self.generate_xml_btn.pack(side='left', padx=10)

        self.help_btn = tk.Button(self.navbar, text='Ayuda', command=self.ayuda)
        self.help_btn.pack(side='left', padx=10)

        self.exit_btn = tk.Button(self.navbar, text='Salir', command=master.quit)
        self.exit_btn.pack(side='left', padx=10)

        self.graph_label = tk.Label(master, text='Aquí se mostrarán las gráficas', font=('Arial', 14))
        self.graph_label.pack(pady=20)

        self.add_dron_entry.pack_forget()
        self.add_dron_btn.pack_forget()


    def mostrar_agregar_dron(self):
        self.add_dron_entry.pack(side='left', padx=10)
        self.add_dron_btn.pack(side='left', padx=10)
        self.new_btn.config(command=self.ocultar_agregar_dron)
        self.new_btn.config(text='Cancelar')

    def ocultar_agregar_dron(self):
        self.add_dron_entry.pack_forget()
        self.add_dron_btn.pack_forget()
        self.new_btn.config(command=self.mostrar_agregar_dron)
        self.new_btn.config(text='Nuevo')
```

```python
    def cargar_xml(self):
        filepath = filedialog.askopenfilename(filetypes=[("XML files", "*.xml")])
        if filepath:
            self.drones, self.sistemas_drones, self.mensajes = cargar_datos(filepath)
            messagebox.showinfo("Carga exitosa", "Los datos se han cargado exitosamente")

    def generar_xml(self):
        root = ET.Element("respuesta")
        lista_mensajes = ET.SubElement(root, "listaMensajes")

        for mensaje in self.mensajes.obtener_mensajes():
            mensaje_elem = ET.SubElement(lista_mensajes, "mensaje", nombre=mensaje.nombre)
            sistema_drones_elem = ET.SubElement(mensaje_elem, "sistemaDrones")
            sistema_drones_elem.text = mensaje.sistema_drones

            tiempo_optimo_elem = ET.SubElement(mensaje_elem, "tiempoOptimo")
            tiempo_optimo_elem.text = str(self.tiempo_optimo(mensaje))

            mensaje_recibido_elem = ET.SubElement(mensaje_elem, "mensajeRecibido")

            instrucciones_elem = ET.SubElement(mensaje_elem, "instrucciones")

            for instruccion in mensaje.instrucciones:
                tiempo_elem = ET.SubElement(instrucciones_elem, "tiempo", valor=str(instruccion.altura))
                acciones_elem = ET.SubElement(tiempo_elem, "acciones")

                dron_elem = ET.SubElement(acciones_elem, "dron", nombre=instruccion.dron)

        ET.indent(root, space="  ")

        tree = ET.ElementTree(root)
        file_path = filedialog.asksaveasfilename(defaultextension=".xml", filetypes=[("XML files", "*.xml")])

        if file_path:
            tree.write(file_path, xml_declaration=True, encoding="utf-8")
            messagebox.showinfo("XML Generado", f"El archivo XML ha sido generado correctamente en {file_path}")
```

```python
    def ayuda(self):
        nombre = "Rodrigo Sebastian Castro Aguilar"
        carne = "202204496"
        curso = "Introduccion a la Programacion y Computacion 2"
        seccion = "A"
        mensaje = f"Nombre: {nombre}\nCarné: {carne}\nCurso: {curso}\nSección: {seccion}"
        messagebox.showinfo("Datos del Estudiante", mensaje)

    def listar_drones(self):
        nombres_drones = [dron.nombre for dron in self.drones]
        drones_ordenados = sorted(nombres_drones)
        messagebox.showinfo("Drones Ordenados", "\n".join(drones_ordenados))

    def listar_mensajes(self):
        nombres_mensajes = [mensaje.nombre for mensaje in self.mensajes.obtener_mensajes()]
        messagebox.showinfo("Mensajes", "\n".join(nombres_mensajes))

    def agregar_dron(self):
        nombre_dron = self.add_dron_entry.get()

        dron_existente = any(dron.nombre == nombre_dron for dron in self.drones.obtener_drones())

        if not dron_existente:
            nuevo_dron = Dron(nombre_dron)
            self.drones.insertar_fin(nuevo_dron)
            messagebox.showinfo("Dron agregado", f"Se ha agregado el dron '{nombre_dron}'")
        else:
            messagebox.showerror("Error", f"El dron '{nombre_dron}' ya existe")
```

Universidad de San Carlos de Guatemala
Escuela de Ingeniería en Ciencias y Sistemas, Facultad de Ingeniería
Introducción a la programación y computación 2, 2do. Semestre 2023.

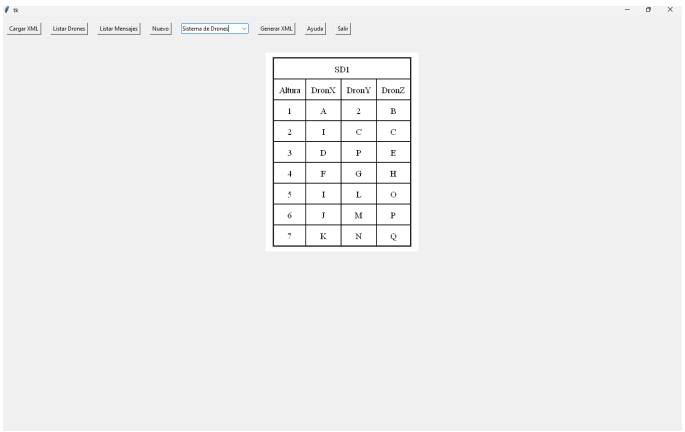
```python
    def menu(self, event):
        option = event.widget.get()
        if option == "Sistema de Drones":
            self.mostrar_sistemas_drones()
        elif option == "Mensajes":
            mensaje_elegido = self.prompt_choice("Selecciona un mensaje:", [mensaje.nombre for mensaje in self.mensajes.obtener_mensa
            if mensaje_elegido:
                self.mostrar_instrucciones(mensaje_elegido)
        elif option == "Instrucciones":
            mensaje_elegido = self.prompt_choice("Selecciona un mensaje:", [mensaje.nombre for mensaje in self.mensajes.obtener_mensa
            if mensaje_elegido:
                    self.generar_instrucciones( mensaje_elegido)
            else:
                messagebox.showerror("Error", "No se seleccionó un mensaje.")

    def mostrar_sistemas_drones(self):
        sistemas_drones_nombres = [sistema.nombre for sistema in self.sistemas_drones]
        selected_sistema = self.prompt_choice("Selecciona un sistema de drones:", sistemas_drones_nombres)
        if selected_sistema:
            sistema_drones = next((sistema for sistema in self.sistemas_drones if sistema.nombre == selected_sistema), None)
            if sistema_drones:
                self.generar_grafica_sistema_drones(sistema_drones)
            else:
                messagebox.showerror("Error", "No se encontró el sistema de drones seleccionado.")

    def mostrar_instrucciones(self, nombre_mensaje):
        mensaje = next((mensaje for mensaje in self.mensajes.obtener_mensajes() if mensaje.nombre == nombre_mensaje), None)
        if mensaje:
            instrucciones = mensaje.instrucciones
            tabla = "Instrucciones:\n"
            for instruccion in instrucciones:
                tabla += f"{instruccion.dron},{instruccion.altura}\n"
            messagebox.showinfo("Instrucciones del Mensaje", tabla)
        else:
            messagebox.showerror("Error", "No se encontró el mensaje seleccionado.")

    def prompt_choice(self, mensaje, opciones):
        """Muestra un cuadro de diálogo para que el usuario elija una opción."""
        return simpledialog.askstring("Selección", mensaje, initialvalue=opciones[0])
```

```python
    def generar_grafica_sistema_drones(self, sistema_drones):
        dot_string = 'digraph G {\n'
        dot_string += sistema_drones.to_dot()
        dot_string += "}\n"
        with open("sistema_drones.dot", "w") as archivo:
            archivo.write(dot_string)
        os.system('dot -Tpng sistema_drones.dot -o sistema_drones.png')
        img = tk.PhotoImage(file='sistema_drones.png')
        self.graph_label.config(image=img)
        self.graph_label.image = img

    def generar_instrucciones(self, nombre_mensaje):
        mensaje = next((mensaje for mensaje in self.mensajes.obtener_mensajes() if mensaje.nombre == nombre_mensaje), None)
        if mensaje:
            lista_instrucciones = mensaje.instrucciones
            sistema_drones = next((sds for sds in self.sistemas_drones if sds.nombre == mensaje.sistema_drones), None)
            if sistema_drones:
                tiempo_total = 0
                for instruccion in lista_instrucciones:
                    dron = next((dron for dron in sistema_drones.drones if dron.nombre == instruccion.dron), None)
                    if dron:
                        dron.agregar_instruccion(instruccion.altura)


                        tiempo_total += self.ejecutar_instruccion(dron, instruccion.altura,instruccion.dron)
                    else:
                        messagebox.showerror("Error", f"No se encontró el dron '{instruccion.dron}' en el sistema de drones.")
                        return
                messagebox.showinfo("Instrucciones Completadas", f"Todas las instrucciones se han completado en {tiempo_total} segund
            else:
                messagebox.showerror("Error", "No se encontró el sistema de drones especificado en el mensaje.")
        else:
            messagebox.showerror("Error", "No se encontró el mensaje especificado.")
```

```python
    def tiempo_optimo(self, mensaje):
        tiempo_optimo = 0
        tiempo_actual = self.drones.inicio.dato.tiempo
        for instruccion in mensaje.instrucciones:
            dron = next((dron for dron in self.drones.obtener_drones() if dron.nombre == instruccion.dron), None)
            if dron:
                droninstru = instruccion.dron
                dron.tiempo += self.ejecutar_instruccion(dron, instruccion.altura,droninstru)
        for time in self.drones:
            tiempo_optimo = time.tiempo
            if tiempo_optimo > tiempo_actual:
                tiempo_actual = tiempo_optimo
        print(tiempo_actual)
        return tiempo_actual

    def ejecutar_instruccion(self, dron, altura_objetivo, dron_instruccion):
        tiempo = 0
        while dron.naltura != altura_objetivo:
            if dron.naltura < altura_objetivo:
                dron.naltura += 1
                tiempo += 1
                dron.movimientos.insertar_fin("subir")
                print(f"{dron.nombre} sube a la altura {dron.naltura}")
            elif dron.naltura > altura_objetivo:
                dron.naltura -= 1
                tiempo += 1
                dron.movimientos.insertar_fin("bajar")
                print(f"{dron.nombre} baja a la altura {dron.naltura}")

        if dron.naltura == altura_objetivo and dron.nombre == dron_instruccion:
            tiempo += 1
            dron.movimientos.insertar_fin("EmitirLuz")
            print(f"{dron.nombre} emitit la luz a la altura {dron.naltura}")
        elif dron.naltura == altura_objetivo:
            print(f"{dron.nombre} Esperar {dron.naltura}")
            tiempo +=1
        return tiempo
```

```python
def cargar_datos(filepath):
    drones = ListaEnlazada()
    sistemas_drones = ListaEnlazada()
    mensajes = ListaEnlazada()

    tree = ET.parse(filepath)
    root = tree.getroot()

    for dron_elem in root.findall('.//listaDrones/dron'):
        nombre_dron = dron_elem.text
        nuevo_dron = Dron(nombre_dron)
        drones.insertar_fin(nuevo_dron)

    for sistema_elem in root.findall('.//listaSistemasDrones/sistemaDrones'):
        nombre_sistema = sistema_elem.get('nombre')
        altura_maxima = int(sistema_elem.find('alturaMaxima').text)
        cantidad_drones = int(sistema_elem.find('cantidadDrones').text)
        drones_sistema = ListaEnlazada()

        for contenido_elem in sistema_elem.findall('.//contenido'):
            nombre_dron = contenido_elem.find('dron').text
            alturas = ListaEnlazada()
            for altura_elem in contenido_elem.findall('.//alturas/altura'):
                valor_altura = int(altura_elem.get('valor'))
                letra = altura_elem.text
                alturas.insertar_fin(valor_altura, letra)
            dron = Dron(nombre_dron, alturas)
            drones_sistema.insertar_fin(dron)

        sistema_drones = SistemaDrones(nombre_sistema, altura_maxima, cantidad_drones, drones_sistema)
        sistemas_drones.insertar_fin(sistema_drones)

    for mensaje_elem in root.findall('.//listaMensajes/Mensaje'):
        nombre_mensaje = mensaje_elem.get('nombre')
        sistema_drones_nombre = mensaje_elem.find('sistemaDrones').text
        instrucciones = ListaEnlazada()
```

```python
        for instruccion_elem in mensaje_elem.findall('.//instrucciones/instruccion'):
            nombre_dron = instruccion_elem.get('dron')
            altura = int(instruccion_elem.text)
            instruccion = Instruccion(nombre_dron, altura)
            instrucciones.insertar_fin(instruccion)

        mensaje = Mensaje(nombre_mensaje, sistema_drones_nombre, instrucciones)
        mensajes.insertar_fin(mensaje)

    return drones, sistemas_drones, mensajes

if __name__ == '__main__':
    root = tk.Tk()
    app = App(root)
    root.mainloop()
```



SD1			
Altura	DronX	DronY	DronZ
1	A	2	B
2	I	C	C
3	D	P	E
4	F	G	H
5	I	L	O
6	J	M	P
7	K	N	Q