CNN-Based Retinal Disease Classification - Source Code

```python
import numpy as np
import os
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
from flask import Flask, request, render_template, jsonify
import serial
import serial.tools.list_ports
import platform
import time

app = Flask(__name__)

# Load the model
model = load_model("EyeModel.h5", compile=False)

# Global variables for UART
ser = None
last_port = None

def get_port_name():
    system = platform.system()
    if system == "Windows":
        return 'COM1'
    elif system == "Linux":
        return '/dev/ttyUSB0'
    else:
        return None

def list_available_ports():
    ports = serial.tools.list_ports.comports()
    for port in ports:
        print(f"Found port: {port.device}")
        print(f"Description: {port.description}")
        print(f"Hardware ID: {port.hwid}")
        print("---")
    return ports

def find_silicon_labs_port():
    ports = list_available_ports()
    for port in ports:
        if "CP210x" in port.description:
            return port.device
    return None

def initialize_serial():
    global ser
    try:
        if ser is not None and ser.is_open:
            ser.close()
            ser = None

        port = find_silicon_labs_port() or get_port_name()
        if not port:
            print("UART Debug: No suitable port found")
            return False

        print(f"UART Debug: Attempting to connect to {port}...")
        ser = serial.Serial(
            port=port,
            baudrate=9600,
            bytesize=serial.EIGHTBITS,
            parity=serial.PARITY_NONE,
            stopbits=serial.STOPBITS_ONE,
            timeout=1,
            write_timeout=1
        )

        if ser.is_open:
            print(f"UART Debug: Successfully connected to {port}")
            return True
        return False

    except serial.SerialException as e:
        print(f"UART Debug: Serial Error: {e}")
        return False
    except Exception as e:
        print(f"UART Debug: Unexpected error: {e}")
        return False

# Initialize UART connection
uart_enabled = False
try:
    if initialize_serial():
        uart_enabled = True
        print("UART Connected Successfully!")
except Exception as e:
    print(f"UART Connection Failed: {e}")
    print("Application will run without UART communication")

@app.route('/')
def index():
    return render_template('index.html')

@app.route('/predict', methods=['GET', 'POST'])
def upload():
    if request.method == 'POST':
        f = request.files['image']
        basepath = os.path.dirname(__file__)
        filepath = os.path.join(basepath, 'uploads', f.filename)

        os.makedirs(os.path.join(basepath, 'uploads'), exist_ok=True)
        f.save(filepath)

        img = image.load_img(filepath, target_size=(128, 128))
        x = image.img_to_array(img)
        x = np.expand_dims(x, axis=0)

        pred = np.argmax(model.predict(x))
        index = {0: 'Cataract', 1: 'Diabetic Retinopathy', 2: 'Glaucoma', 3: 'Normal'}

        if uart_enabled:
            try:
                disease_map = {0: '0', 1: '1', 2: '2', 3: '3'}
                uart_message = disease_map[pred]

                if not ser or not ser.is_open:
                    initialize_serial()
                if ser and ser.is_open:
                    ser.write(uart_message.encode())
                    ser.flush()
                    print(f"UART Message Sent: {uart_message}")
                    time.sleep(0.1)

                if ser.in_waiting:
                    response = ser.read(ser.in_waiting)
                    print(f"UART Debug: Received response: {response}")
            except Exception as e:
                print(f"Failed to send UART message: {e}")

        text = f"The classified Disease is : {index[pred]}"
        return text

@app.teardown_appcontext
def cleanup(error):
    global ser
    if ser is not None and ser.is_open:
        ser.close()
        ser = None

if __name__ == '__main__':
    print("UART Debug: Starting application...")
    print(f"Running on: {platform.system()}")
    print("Available ports:")
    list_available_ports()
    app.run(debug=True, use_reloader=False)
```