sundial_plots.py

```python
import mysql.connector
import matplotlib.pyplot as plt
import matplotlib.animation as animation
import numpy as np
from collections import deque

# === CONFIG ===
DB_CONFIG = {
    'host': 'localhost',
    'user': 'esp_user',
    'password': 'ESPtesting123!!!',
    'database': 'esp_user'
}

MAX_POINTS = 100  # For LDR history
SMOOTHING_DELTA = 0.0  # δ smoothing coefficient, adjust for smoothing

# === DATA BUFFERS ===
ldr_left_data = deque(maxlen=MAX_POINTS)
ldr_right_data = deque(maxlen=MAX_POINTS)
sun_azimuth = [0]  # For polar chart
gnomon_angle = [0]  # Simulated gnomon tilt
system_state = ["UNKNOWN"]

# === DATABASE CONNECTION ===
def fetch_latest_data():
    try:
        conn = mysql.connector.connect(**DB_CONFIG)
        cursor = conn.cursor(dictionary=True)
        cursor.execute("""
            SELECT * FROM sun_tracking_data
            ORDER BY timestamp DESC
            LIMIT %s
        """, (MAX_POINTS,))
        rows = cursor.fetchall()
        cursor.close()
        conn.close()
        return rows[::-1]  # Return in chronological order
    except mysql.connector.Error as e:
        print("DB Error:", e)
        return []

# === SMOOTHING FUNCTION ===
def smooth_series(data, delta):
    if not data: return []
    smoothed = [data[0]]
    for i in range(1, len(data)):
        a = smoothed[-1]
        b = data[i]
        ab = b - a
```

```
        c = b - ab * delta
        smoothed.append(c)
    return smoothed

# === MATPLOTLIB SETUP ===
fig = plt.figure(figsize=(10, 8))

# Polar plot: Sun Azimuth
ax1 = fig.add_subplot(221, polar=True)
azimuth_line, = ax1.plot([], [], marker='o')
ax1.set_title("Sun Azimuth (North = 0°)")

# Set custom compass direction labels
ax1.set_theta_zero_location('N')  # 0° at the top
ax1.set_theta_direction(-1)       # Clockwise
ax1.set_thetagrids(
    angles=[0, 90, 180, 270],
    labels=['N', 'E', 'S', 'W']
)

# Gnomon tilt polar chart
ax2 = fig.add_subplot(222, polar=True)
gnomon_line, = ax2.plot([], [], marker='o')
ax2.set_title("Gnomon Tilt Angle")

# LDR plot
ax3 = fig.add_subplot(223)
ldr_left_plot, = ax3.plot([], [], label='LDR Left')
ldr_right_plot, = ax3.plot([], [], label='LDR Right')
ax3.set_title("LDR Sensor Values (Smoothed)")
ax3.legend()
ax3.set_ylim(0, 5096)

# System state display
ax4 = fig.add_subplot(224)
ax4.axis('off')
state_text = ax4.text(0.5, 0.5, "", fontsize=18, ha='center', va='center')

# === UPDATE FUNCTION ===
def update_plot(frame):
    data = fetch_latest_data()
    if not data:
        return

    latest = data[-1]
    sun_azimuth[0] = latest['sun_azimuth']
    gnomon_angle[0] = latest['latitude']  # Adjust if needed
    system_state[0] = latest['system_state']

    ldr_left_raw = [row['ldr_left'] for row in data]
    ldr_right_raw = [row['ldr_right'] for row in data]

    smoothed_left = smooth_series(ldr_left_raw, SMOOTHING_DELTA)
```

```
    smoothed_right = smooth_series(ldr_right_raw, SMOOTHING_DELTA)

    ldr_left_data.clear()
    ldr_left_data.extend(smoothed_left)
    ldr_right_data.clear()
    ldr_right_data.extend(smoothed_right)

    # Sun azimuth polar update
    az_rad = np.deg2rad(sun_azimuth[0])
    azimuth_line.set_data([0, az_rad], [0, 1])

    # Gnomon tilt polar update
    gn_rad = np.deg2rad(gnomon_angle[0])
    gnomon_line.set_data([0, gn_rad], [0, 1])

    # LDR plot update
    x_vals = list(range(len(ldr_left_data)))
    ldr_left_plot.set_data(x_vals, list(ldr_left_data))
    ldr_right_plot.set_data(x_vals, list(ldr_right_data))
    ax3.set_xlim(0, len(ldr_left_data))

    # System state display
    state_text.set_text(f"System State:\n{system_state[0]}")

# === START ANIMATION ===
ani = animation.FuncAnimation(fig, update_plot, interval=1000)
plt.tight_layout()
plt.show()

```