



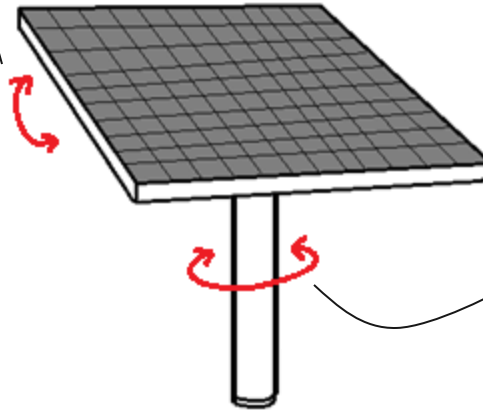
PROYECTO: **SEGUIDOR SOLAR**

ENRIQUEZ MICHAEL
CONTRERAS ANTHONY
JIMENEZ YASID
OÑATE IAN

Seguidor solar de dos grados de libertad

ÁNGULO PITCH:

Permite que el panel se incline hacia arriba o hacia abajo, como si estuviera asintiendo con la cabeza.



ÁNGULO ROLL:

Permite al panel "girar" sobre su propio eje, como si estuviera girando la cabeza para mirar hacia un lado.



01

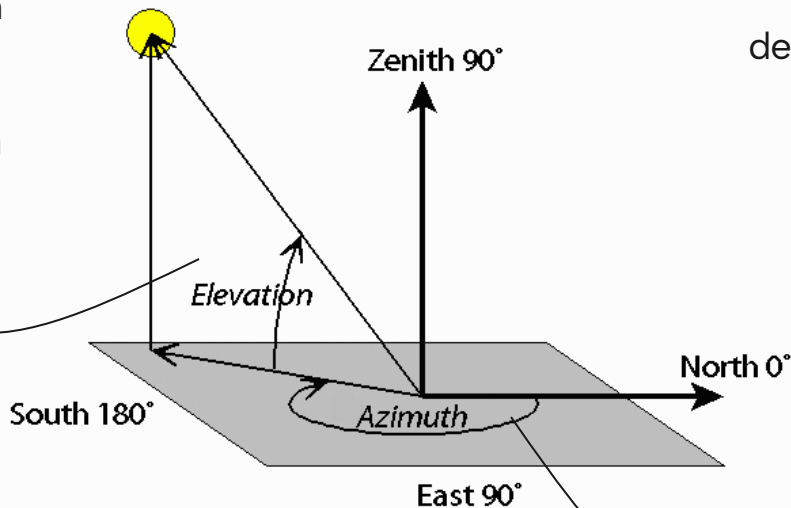
Desarrollo matemático

Rotación compuesta – Ángulos de Euler



Cálculo del 'azimuth' y 'elevation'

Es el ángulo de **elevación** del sol con respecto a su proyección en la superficie y se relaciona con el ángulo 'pitch'



def getSolarPosition(...)

```
az = get_azimuth(latitude, longitude, current_time)
el = get_altitude(latitude, longitude, current_time)
```

Es el ángulo **azimutal** de la proyección del sol en la superficie con respecto al norte y se relaciona con el ángulo 'roll'

Rotación Compuesta

Se define como el resultante de aplicar sucesivamente dos o más rotaciones a un objeto o sistema de coordenadas. Una forma común de parametrizar las rotaciones es mediante...

Ángulos de Euler

Estos ángulos
suelen denominarse
roll, pitch y yaw

$$R_{xyz}(\alpha, \beta, \gamma) = \begin{bmatrix} c\beta c\gamma & s\alpha s\beta c\gamma - c\alpha s\gamma & c\alpha s\beta c\gamma + s\alpha s\gamma \\ c\beta s\gamma & s\alpha s\beta s\gamma + c\alpha c\gamma & c\alpha s\beta s\gamma - s\alpha c\gamma \\ -s\beta & s\alpha c\beta & c\alpha c\beta \end{bmatrix}$$

Representan tres
rotaciones sucesivas
alrededor de ejes
específicos.

Es una matriz de
rotación

Se calcula
multiplicando
las matrices de
rotación de los
tres ejes

Cálculo de 'Roll'

$$\alpha = \arctan \left(\frac{R_{32} / \cos(\beta)}{R_{33} / \cos(\beta)} \right)$$

$$\alpha = \arctan \left(\frac{R_{32}}{R_{33}} \right)$$

$$\alpha = \arctan 2(R_{32}, R_{33})$$

$$R_{32} = s\alpha c\beta$$

$$R_{33} = c\alpha c\beta$$

Cálculo de 'Pitch'

$$\beta = \arctan \left(\frac{-R_{31}}{\pm \sqrt{R_{11}^2 + R_{21}^2}} \right)$$

$$\beta = \arctan 2(-R_{31}, \sqrt{R_{11}^2 + R_{21}^2})$$

$$R_{31} = -s\beta$$

$$R_{11} = c\beta c\gamma$$

$$R_{21} = c\beta s\gamma$$

Cálculo de 'Yaw'

$$\gamma = \arctan \left(\frac{R_{21} / \cos(\beta)}{R_{11} / \cos(\beta)} \right)$$

$$\gamma = \arctan \left(\frac{R_{21}}{R_{11}} \right)$$

$$\gamma = \arctan 2(R_{21}, R_{11})$$

$$R_{21} = c\beta s\gamma$$

$$R_{11} = c\beta c\gamma$$

Matriz de rotación compuesta

```
# Matrices de rotación individuales
Rx = np.array([
    [1, 0, 0],
    [0, np.cos(alpha_rad), -np.sin(alpha_rad)],
    [0, np.sin(alpha_rad), np.cos(alpha_rad)]
])

Ry = np.array([
    [np.cos(beta_rad), 0, np.sin(beta_rad)],
    [0, 1, 0],
    [-np.sin(beta_rad), 0, np.cos(beta_rad)]
])

Rz = np.array([
    [np.cos(gamma_rad), -np.sin(gamma_rad), 0],
    [np.sin(gamma_rad), np.cos(gamma_rad), 0],
    [0, 0, 1]
])

# Multiplicar las matrices en el orden correcto: Rz( $\gamma$ ) * Ry( $\beta$ ) * Rx( $\alpha$ )
R = np.dot(Rz, np.dot(Ry, Rx))
```

def Rxyz(alpha, beta, gamma)

```
# Convertir los ángulos a radianes
alpha_rad = np.radians(alpha)
beta_rad = np.radians(beta)
gamma_rad = np.radians(gamma)
```

Retorna un matriz de rotación 3D
de manera general

Cálculo de 'Roll' y 'Pitch'

```
# Calculamos beta y alpha
val = np.cos(el_rad) * np.sin(az_rad)
beta_rad_temp = np.arcsin(val)
beta_deg_temp = np.degrees(beta_rad_temp)

val_fi1 = -(np.cos(el_rad) * np.cos(az_rad)) / np.cos(beta_rad_temp)
alpha_rad_temp = np.arcsin(val_fi1)
alpha_deg_temp = np.degrees(alpha_rad_temp)

R = Rxyz(alpha_deg_temp, beta_deg_temp, gamma)
```

2. Obtener los elementos de la matriz

```
R31 = R[2, 0]
```

```
R11 = R[0, 0]
```

```
R21 = R[1, 0]
```

```
R32 = R[2, 1]
```

```
R33 = R[2, 2]
```

3. Calcular beta (pitch)


```
beta_rad = np.arctan2(-R31, np.sqrt(R11**2 + R21**2))
```

```
beta_deg = np.degrees(beta_rad)
```


4. Calcular alpha (roll)

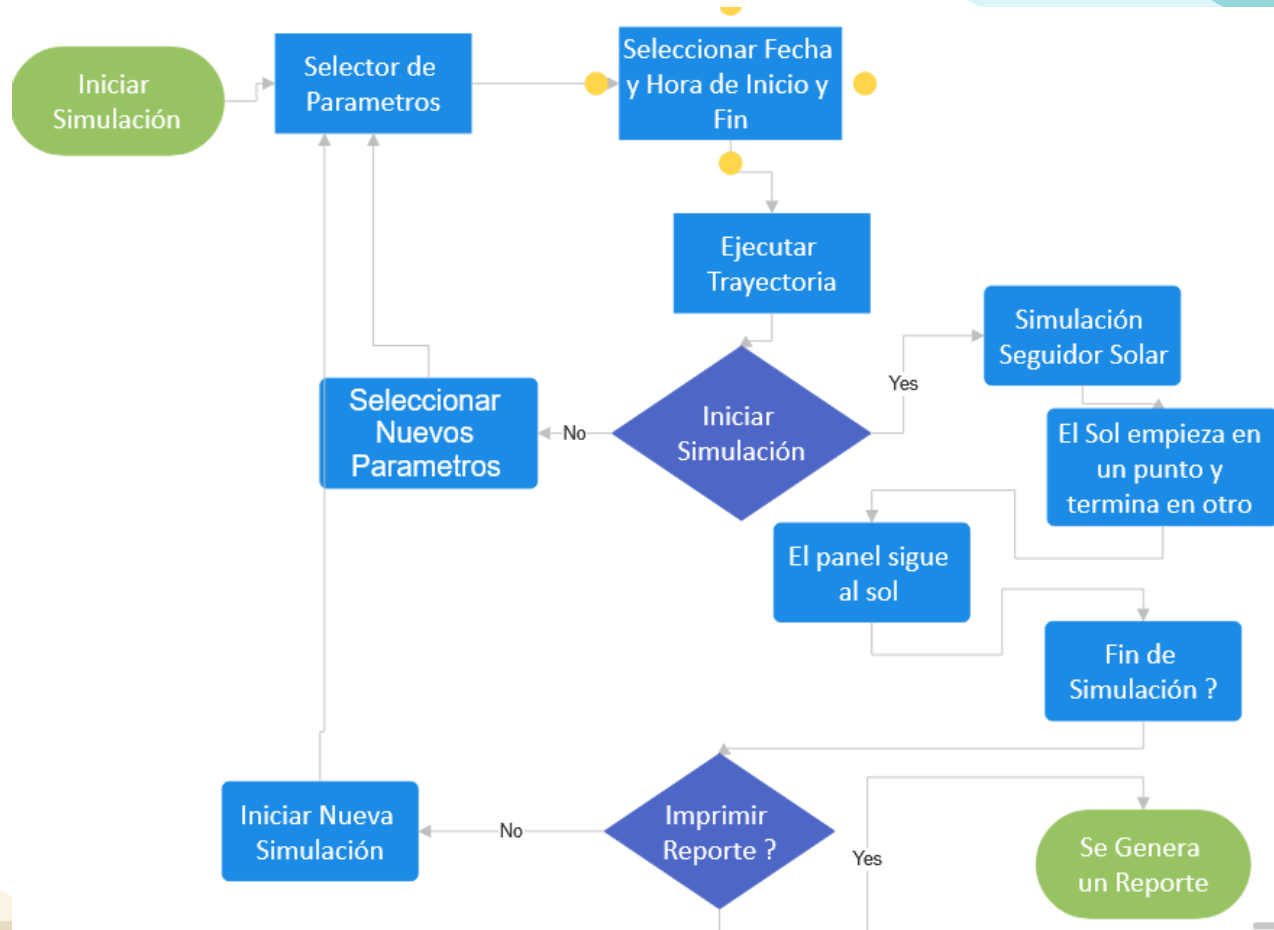
```
alpha_rad = np.arctan2(R32, R33)
```

```
alpha_deg = np.degrees(alpha_rad)
```

02 Diagrama de flujo







03 Programación Backend

Posición Panel Solar y Sol.



Obtención de parámetros

```
st_fecha = None
start_t = None
end_t = None

def obtener_fecha_y_horas(callback): 2 usages  ImYasid +1
    global st_fecha, start_t, end_t, ventana

    fecha_seleccionada = calendario.get_date()
    st_fecha = datetime.strptime(fecha_seleccionada, format: "%m/%d/%y").strftime("%Y-%m-%d")

    hora_inicio = combo_inicio.get()
    hora_fin = combo_fin.get()

    try:
        start_t = datetime.strptime(hora_inicio, format: "%H").time()
        end_t = datetime.strptime(hora_fin, format: "%H").time()
    except ValueError:
        messagebox.showerror(title: "Error", message: "Por favor ingrese las horas en formato HH")

    # Llamar al callback para actualizar la interfaz con los valores seleccionados
    if callback:
        callback(st_fecha, start_t.strftime("%H:%M"), end_t.strftime("%H:%M"))

    ventana.destroy()
```

Solar Position

```
def getSolarPosition( 3 usages  ImYasid +1
    start_date: datetime,
    start_hour: int = 6,
    end_hour: int = 18,
    latitude: float = -0.2105367,
    longitude: float = -78.491614
):
    """
    Calcula posiciones solares y ángulos para una fecha específica y rango de horas.

    Args:
        start_date (datetime): Fecha y hora de inicio para la simulación.
        start_hour (int): Hora de inicio del cálculo.
        end_hour (int): Hora de fin del cálculo.
        latitude (float): Latitud para la posición geográfica.
        longitude (float): Longitud para la posición geográfica.

    Returns:
        tuple: Tupla con los siguientes elementos:
            - times (list): Lista de tiempos de simulación.
            - azimuths (list): Lista de ángulos azimutales.
            - elevations (list): Lista de ángulos de elevación.
            - beta (list): Lista de ángulos de pitch.
            - alpha (list): Lista de ángulos de roll.
    """
```

Generar Informe

```
def generar_reporte(fecha, hora_inicio, hora_fin):  
    """  
    Genera un reporte con los cálculos y gráficas en formato PDF sin guardar imágenes permanentes,  
    guardado en el escritorio con nombre de archivo incremental.  
    """  
  
    # Obtener datos  
    times, azimuths, elevations, beta, alpha = getSolarPosition(  
        start_date=fecha, start_hour=hora_inicio, end_hour=hora_fin  
    )  
  
    # Crear la primera gráfica (Azimut y Elevación)  
    plt.figure(figsize=(10, 5))  
    plt.plot(times, azimuths, label="Azimut", marker="o")  
    plt.plot(times, elevations, label="Elevación", marker="s")  
    plt.xlabel("Tiempo")  
    plt.ylabel("Ángulo (°)")  
    plt.title("Azimut y Elevación a lo largo del tiempo")  
    plt.legend()  
    plt.xticks(rotation=45)  
  
    # Guardar la figura en un archivo temporal  
    img_path1 = "grafico_azimut_elevacion.png"  
    plt.savefig(img_path1, format='png')  
    plt.close()  
  
    # Crear la segunda gráfica (Pitch y Roll)  
    plt.figure(figsize=(10, 5))  
    plt.plot(times, beta, label="Beta (Pitch)", marker="o", color="r")  
    plt.plot(times, alpha, label="Alpha (Roll)", marker="s", color="g")  
    plt.xlabel("Tiempo")  
    plt.ylabel("Ángulo (°)")  
    plt.title("Pitch y Roll a lo largo del tiempo")  
    plt.legend()  
    plt.xticks(rotation=45)
```



04

Ejecución

Simulador del seguidor solar





Seguidor Solar

¡Bienvenido al simulador de un Seguidor Solar!

Visualiza y analiza el movimiento de un seguidor solar de 2 grados de libertad.

[Iniciar Simulación](#)



05 Repositorio Público





landres10188 Merge branch 'main' of https://github.com/TheAnsw3r/Seguidor_Solar

0fb37e6 · 30 minutes ago



54 Commits

Experimentacion	cambio de nombres	9 hours ago
Imagenes	segundo avance interfaz	3 days ago
Interfaz	Código con comentarios y pydocs	30 minutes ago
Desarrollo Funcional.ipynb	Control de valores	2 weeks ago
PROYECTO-SEGUIDOR SOLAR-Presentación.pdf	Presentación final	41 minutes ago
Proyecto MN-Seguidor Solar.pdf	Informe	1 hour ago
README.md	Original Main	last week
Video de demostracion.mp4	video de demostración	8 hours ago
experimentacionMain.py	xd	9 hours ago
requirements.txt	Mejoras en requirements	10 hours ago
seguidorSolar.py	semiFinal	8 hours ago



06 Desafíos



1. Desafíos Matemáticos

- ✓ Cálculo preciso de los ángulos de control (Pitch y Roll).
- ✓ Obtención de la posición solar.

2. Desafíos Computacionales

- ✓ Manejo de listas grandes de datos.
- ✓ Compatibilidad con diferentes versiones de Python y dependencias.

3. Desafíos Gráficos y Visuales

- ✓ Visualización precisa del sol y el panel solar en 3D.
- ✓ Sincronización de la animación.

4. Desafíos en la Interfaz de Usuario

- ✓ Selección de fecha y rango de horas en tkinter.
- ✓ Gestión de eventos en tkinter.

5. Desafíos en la Generación de Reportes

- ✓ Generación correcta del PDF con fpdf.
- ✓ Automatización del nombre del archivo para evitar sobrescritura.