Algoritmo Para Determinar la Orientación Óptima de Paneles Solares

Ajila Arelys, Armijos Luis, Carreño Douglas, Córdova Viviana, Naula Gerardo, Jessica Rivas

I.Definición del Problema

En vista del aumento del calentamiento global provocado por las emisiones de gases de efecto invernadero, implementar fuentes generadoras de energía sustentable se vuelve cada vez más urgente. En vista de esta necesidad, una de las alternativas por la que han apostado varios gobiernos es la energía solar. Hoy en día el uso de la energía solar está siendo implementado a escalas cada vez mayores a nivel mundial. Es preciso especificar que la radiación solar puede ser aprovechada de dos formas: por conversión fotovoltaica en energía eléctrica a través de paneles o por conversión térmica empleando dispositivos que colectan y concentran la radiación, ya sea para calentar agua o generar electricidad, y de esa forma calentar un fluido que al evaporarse mueva turbinas implementadas en generadores eléctricos [1]. Para aprovechar esta fuente de energía, este proyecto tiene como finalidad la creación de un algoritmo que, a través de fórmulas y cálculos matemáticos calcule con precisión la ubicación del sol en las diferentes horas del día. Este algoritmo será implementado en un sistema de seguimiento solar para que permita la movilidad del panel solar de acuerdo a la ubicación del sol previamente calculada.

Es comúnmente conocido que el rumbo aparente del sol cambia con los días. Esto se debe principalmente a la duración del día a lo largo del año y los ángulos de declinación, de altitud, de azimut entre otros [2]. La variación del rumbo del sol se convierte en la principal problemática a resolver para que nuestro prototipo sea eficiente. La radiación solar es captada de mejor manera cuando la placa solar está en dirección perpendicular a los rayos del sol [3]. Para conseguir lo anteriormente mencionado, es necesario idear un algoritmo que considere todas las variaciones anteriormente descritas y consiga que el prototipo se mueva siguiendo las diferentes rotaciones del sol para lograr una dirección perpendicular a lo largo del día todos los meses del año.

II.Código en C:

```
#include <stdio.h>

#include <math.h>

#include <time.h>


// Función para contar los días de un mes

int conteoDiasMes(int mes, int anio) {

    int diasPorMes = 31;  // Inicialmente se asume que el mes tiene 31 días

    if (mes == 2) {

        // Comprobar si el año es bisiesto

        if ((anio % 4 == 0 && anio % 100 != 0) || (anio % 400 == 0)) {

            diasPorMes = 29;  // Año bisiesto

        } else {

            diasPorMes = 28;  // Año no bisiesto

        }

    } else if (mes == 4 || mes == 6 || mes == 9 || mes == 11) {

        diasPorMes = 30;  // Meses con 30 días (Abril, junio, septiembre y noviembre)

    }

    return diasPorMes;  // Devuelve el número de días del mes

}



// Función para calcular los días totales entre meses

int calculoEntreMeses(int mesInicial, int mesActual, int diaInicial, int diaActual, int anio) {

    int diasTotalesEntreMeses = 0;

    int diasEntreMeses = 0;

    int mesRecorrido = mesInicial;



    for (mesRecorrido = mesInicial; mesRecorrido < mesActual; mesRecorrido++) {

        diasEntreMeses += conteoDiasMes(mesRecorrido, anio);

    }
```

```
    diasTotalesEntreMeses = diasEntreMeses + diaActual;
    return diasTotalesEntreMeses;
}

int main() {
    double Latid, Longi;
    char buffer[100];

    printf("Calcularemos la orientacion del Sol e inclinacion del Panel Solar\n");

    // Obtener la Latitud
    while (1) {
        printf("Ingresa la Latitud, el valor debe ser entre -90 grados y +90 grados:");
        if (fgets(buffer, sizeof(buffer), stdin)) {
            if (sscanf(buffer, "%lf", &Latid) == 1) {
                break;  // Si se pudo leer correctamente, salir del bucle
            } else {
                printf("El Dato ingresado no es valido.\n");
            }
        }
    }

    // Obtener la Longitud
    while (1) {
        printf("Ingresa la Longitud, el valor debe ser entre -180 grados a +180 grados:");
        if (fgets(buffer, sizeof(buffer), stdin)) {
            if (sscanf(buffer, "%lf", &Longi) == 1) {
                break;  // Si se pudo leer correctamente, salir del bucle
            } else {
                printf("El Dato ingresado no es valido.\n");
            }
        }
    }

    // Obtener la fecha y la hora actual
    time_t t = time(NULL);
    struct tm *tm = localtime(&t);

    int anioActual = tm->tm_year + 1900;
    int mesActual = tm->tm_mon + 1;
    int diaActual = tm->tm_mday;

    int hora = tm->tm_hour;
    int minutos = tm->tm_min;
    int segundos = tm->tm_sec;

    int mesInicial = 1;
    int diaInicial = 1; // Suponiendo que queremos contar desde el primer día del primer mes

    int n = calculoEntreMeses(mesInicial, mesActual, diaInicial, diaActual, anioActual);

    // Declinación del sol
    double PI = 3.14159265358979323846; // Mejor usar M_PI definido en math.h
    double dcoseno = ((360.0 / 365.0) * (n + 10)) * PI / 180.0;
    double declinacionsolar = -23.44 * cos(dcoseno);

    // Anomalía solar
    double B = ((360.0 / 365.0) * (n - 81)) * PI / 180.0;

    // Ecuación del tiempo
    double Eot = 9.87 * sin(2 * B) - 7.53 * cos(B) - 1.5 * sin(B);
```

```
    // Hora local
    double HoraLocal = hora + (minutos / 60.0);

    // Zona horaria y longitud estándar
    int ZonaHoraria = -5;
    int LongituEstandar = ZonaHoraria * 15;

    // Tiempo solar verdadero
    double tsvPre = 4 * (Longi - LongituEstandar) + Eot;
    double tsvPree = tsvPre / 60.0;
    double Tsv = HoraLocal + tsvPree;

    // Ángulo horario solar
    double H = ((15 * (Tsv - 12)) * PI / 180.0);

    // Ángulo de inclinación del sol
    double Latitudrad = Latid * PI / 180.0;
    double declinacionsolarRad = declinacionsolar * PI / 180.0;
    double angulo = asin(sin(declinacionsolarRad) * sin(Latitudrad) + cos(declinacionsolarRad) * cos(Latitudrad) * cos(H));
    double angulogrados = angulo * 180.0 / PI;

    // Azimuth
    double azimuth = (sin(declinacionsolarRad) - sin(angulo) * sin(Latitudrad)) / (cos(angulo) * cos(Latitudrad));
    double azimuth2 = acos(azimuth);
    if (H > 0) {
        azimuth2 = (2 * PI) - azimuth2;
    }
    double azimuthgrados = azimuth2 * 180.0 / PI;

    // Resultados
    printf("La altura del sol es %.2f grados y tiene una orientacion de %.2f grados respecto al norte\n", angulogrados, azimuthgrados);

    return 0;
}
```

III.Código En Arduino Para Orientar Los Paneles Solares Siguiendo Una Fuente De Luz.

```
#include <Servo.h>

// Variables
int pinServo = 7; // Indicamos el valor del pin del motor
Servo motor; // Servo pasa a ser una variable al estar definido con la importación de la librería
int sensor; // Sensor

// Configuración de la programación
void setup (){
  Serial.begin(9600); // Comunicación entre el puerto serial y el Arduino
  motor.attach(pinServo); // Vinculamos el servo al pin
  motor.write(0); // Inicializa el servo en la posición 0 grados
  delay(1000); // Espera 1 segundo para asegurarse de que el servo se mueva a la posición inicial
}

// Programa
void loop (){
  sensor = analogRead(A0); // Se lee el valor del sensor

  // Remapeo en base a la intensidad de luz del ambiente, ajustado para moverse de 0 a 180 grados
  sensor = map(sensor, 300, 999, 0, 180);

  motor.write(sensor); // En función del sensor
  Serial.print("Posición del servo: ");
```

```
Serial.println(sensor); // Indicamos la posición del servo

delay(2000); // Intervalo de 2 segundos

}
```

IV. Análisis de Variables de Algoritmo en Lenguaje Arduino.

A. Librerías

1) Servo.h

Esta librería es necesaria para controlar los servos pues permite enviar señales para que los motores se muevan correctamente.

B. Variables necesarias

1) *pinServo:* asignada con valor 7, que significa el pin al que está conectado el servo, tipo entero.
2) *motor:* representa el servo, dato tipo servo sensor tipo de dato entero, el cual almacena
3) *sensor:* almacena la lectura analógica del control de luz, tipo entero.

C. Funciones

1) Función setup

Se encarga de inicializar la comunicación serial, esta función se encarga de vincular el pin al servo, inicializando así la posición del mismo. Inicia con la comunicación serial de la computadora a través de Serial.begin(9600), el motor.attach(pinServo),asocia a la variable motor, a el pin de la variable pinServo, y así lograr recibir comandos de posición, el motor.write(o), es la posición inicial del servo, partiendo de cero grados, delay(1000)espera un segundo para que el servo se acomode en la posición inicial.

2) Función loop

La función corresponde a que cada interacción lee las señales del sensor y así este ajusta el valor a el rango, lo que permite que este mueva a el servo a la posición enviada por el sensor, su primera variable visible es anologRead(A0), valor analógico del pin, que se conecta al sensor de luz, el map(sensor, 300, 999, 0, 180), esta establece que los primeros dígitos (300, 999)es el rango inicial del sensor, y el tercer y cuarto valor (0 y a 180), es su rango de movimiento. motor.write(sensor), trabaja con la variable sensor por lo tanto la ubicación del mismo, Serial.print, Serial.println, presenta en pantalla la posición del servo. delay(2000), retraso de dos segundo antes de que el bucle empiece nuevamente.

V. Análisis de Variables de Algoritmo en Lenguaje C

A. Librerias

1) Standard input-output header

Esta librería es estándar porque posee las declaraciones básicas de funciones, macros, etc.

2) Math.h

Esta librería integra operaciones básicas y se la integra para realizar operaciones necesarias en el algoritmo como seno y coseno.

3) time.h

Esta librería permite manipular la hora y fecha del sistema. Es necesaria para obtener la hora y fecha que se utilizarán en las fórmulas de la localización del sol.

B. Variables Necesarias.

1) *Longi:* Almacena el valor de longitud que es ingresado por el usuario y es de tipo double
2) *Latid:* Almacena el valor de latitud que es ingresado por el usuario y es de tipo double
3) *t:* Es una variable de tipo time_t. almacena el valor del tiempo actual en base a la función “time” que con el argumento “NULL” devuelve el tiempo actual.
4) *anioActual:* Fecha del año actual. Es de tipo entero.
5) *MesActual:* devuelve el mes actual. Es de tipo entero.
6) *diaActual:* devuelve el dia actual. Es de tipo entero.
7) *hora:* almacena la hora actual. Es de tipo entero.
8) *minutos*: almacena los minutos actuales. Es de tipo entero.
9) *segundos*: almacena los segundos actuales. Es de tipo entero.
10) *mesInicial*: Inicializa el “mesInicial” en 1, representando a enero. Es de tipo entero.
11) *diaInicial*: Inicializa el “diaInicial” en 1 representando el dia inicial del año. Es de tipo entero.
12) *n*: se le asigna el valor que retorna la función “calculoEntreMeses”. Es de tipo entero.
13) *PI*: tiene el valor de 3,14 y es de tipo double.
14) *dcoseno*: almacena el resultado de una parte de la fórmula de la declinación solar en grados. Es de tipo double.
15) *Declinaciónsolar*: almacena el resultado de la fórmula de la declinación solar implementando la constante de -23,44 y el valor de la variable dcoseno. Es de tipo double.

16) *B*: almacena el valor de la fórmula de la anomalía solar. Es de tipo double.
17) *Eot*: Almacena el valor de la ecuación del tiempo. Es de tipo double.
18) *HoraLocal*: Almacena el valor de la hora local basándose en las variables “hora” y “minutos” anteriormente mencionadas. Es de tipo double.
19) *ZonaHoraria*: Almacena el valor de -5 que es el valor del meridiano local en relación al meridiano de Greenwich. Es de tipo entero.
20) *LongitudEstandar*: Almacena el valor de la “ZonaHoraria” multiplicada por la constante 15. Es de tipo double.
21) *tsvPre*: almacena la primera parte de la fórmula del tiempo solar verdadero utilizando las variables “Longit”, “LongitudEstandar” y la “Eot”. Es de tipo double.
22) *tsvPree*: almacena el valor de la variable “tsvPre” dividida entre constante 60. Es de tipo double.
23) *Tsv*: almacena el valor del tiempo solar verdadero utilizando las variables “HoraLocal” y “tsvPree”. Es de tipo double.
24) *H*: almacena el valor del ángulo horario solar implementando las variables “Tsv” y “PI”. Es de tipo double.
25) *Latitudrad*: Almacena el valor de la Latitud en radianes utilizando la variable “Longi”. Es de tipo double.
26) *declinacionsolarRad*: Almacena el valor de la variable “Declinacionsolar” en radianes. Es de tipo double.
27) *angulo*: almacena el valor de la fórmula del ángulo de inclinación en radianes. Es de tipo double.
28) *Angulogrados*: almacena el valor de la variable “angulo” convertida a grados. Es de tipo double.
29) *Azimuth*: Almacena el valor de la primera parte de la fórmula para calcular el Azimuth solar en radianes. Es de tipo double.
30) *azimuth2*: Almacena el valor del Azimuth solar en radianes, tomando en consideración el arco coseno de la variable “Azimuth”. Es de tipo double.
31) *Azimuthgrados*: Almacena el valor del azimuth solar en grados, tomando en referencia la variable “azimuth2”. Es de tipo double.

C. Estructuras De Datos

En este algoritmo se usará una sola estructura llanada “tm”, necesaria para almacenar la fecha actual en años, meses y días al igual que el tiempo actual en horas, minutos y segundos. Además, almacena el valor del mes y día iniciales necesarios para calcular el número de días que han pasado desde el inicio del año, y cuyo valor también se almacena en esta estructura.

D. Funciones, estructuras de control y bucles

El algoritmo cuenta con 2 funciones:

1) conteoDiasMes

Esta función es de tipo entero y toma como argumento un entero que representa un mes. Hace uso de una estructura “if” para condicionar en caso de que el mes sea febrero cuente 29 días y en caso de que sea abril, junio, septiembre o noviembre cuente 30 días. En caso de que no se cumpla esta condición entonces contará 31 días por mes. Retorna el valor de la variable “diasPorMes”.

2) CalculoEntreMeses

Es de tipo entero y toma como argumentos enteros que representen el mes Inicial, mes Actual, el día Inicial y el día Actual.

Sirve para calcular los días entre el mes inicial(enero) y la fecha actual.

Tiene implementada un bucle for para que el mes recorrido vaya aumentando mientras sea menor al mes actual y el resultado de los días de esos meses se vaya sumando en cada iteración. Luego se agrega los días del mes Actual. Retorna el valor de la variable “diasTotalesEntreMeses”.

El algoritmo también cuenta con un “if” dentro de la función principal:

En la línea 92 se hace uso de la estructura de control “if” para poder reflejar de manera correcta el azimuth o posición del sol. La estructura condiciona para que en caso de que el ángulo horario sea mayor que 0(esto ocurre después del mediodía solar), el resultado del azimuth sea restado a 2π para mostrar el valor correcto de la posición del sol.

VI.Pseudocódigo Y Diagrama De Flujo Comparados Con El Implementado En C

Pseudocódigo

```
// Funcion para conteo de dias //

Funcion diasPorMes <- conteoDiasMes ( mes )

   diasPorMes=31

   si mes==2 Entonces

      diasPorMes=29

   sino
```

```
      si mes==4 o mes==6 o mes==9 o mes=11
Entonces
         diasPorMes=30
      fin    si
   FinSi
fin funcion


Funcion diasTotalesEntreMeses <- calculoEntreMeses
( mesInicial, mesActual, diaInicial, diaActual)
      diasTotalesEntreMeses=0
      diasEntreMeses=0
      mesRecorrido=mesInicial
      diasMesInicial=conteoDiasMes(mesInicial);
      Repetir
                mesRecorrido=mesRecorrido+1

      diasEntreMeses=diasEntreMeses+conteoDiasMes(
mesRecorrido)
      Hasta Que (mesActual-1)=mesRecorrido
      diasTotalesEntreMeses=diasEntreMeses+diasMesI
nicial+diaActual
fin funcion



// inicio de Algoritmo //
Algoritmo proyectopis
      Escribir "Calcularemos La orientación del Sol e
inclinación del Panel Solar"
      Escribir "Ingresa primero la Latitud y Longitud
para proseguir con el cálculo"
      Escribir "Ingresa la Latitud"
      Leer Latid
      Escribir "Ingresa la Longitud"
      Leer Longi


      // convertir la fecha a numero para calcularlos en
las formulas//
      fechaHoy=ConvertirATexto(FechaActual())

anioActual=ConvertirANumero(SubCadena(fechaHoy,
1,4))

mesActual=ConvertirANumero(SubCadena(fechaHoy,
5,6))

diaActual=ConvertirANumero(SubCadena(fechaHoy,7
,8))
    MesInicial= 1
    mesActual=MesActual
    diaActual=diaActual



      // convertir la hora a numeros para calcularlos en
las formulas//
      tiempo = HoraActual()
      f2 = ConvertirATexto(tiempo)
      horass = Subcadena(f2,1,2)
      minutoss = Subcadena(f2, 3, 4)
      segunds = Subcadena(f2,5,6)
      hora = ConvertirANumero(horass)
      minutos = ConvertirANumero(minutoss)
      segund = ConvertirANumero(segunds)


      n = calculoEntreMeses ( mesInicial, mesActual,
diaInicial, diaActual)
```

$$\delta = -23.45° x \cos(360/365 x(d + 10))$$

//Formula de declinación del sol [4]

```
      // declinacion del sol//
      dcoseno = ((360/365 * (n + 10 )) * PI/180)
      declinaciónsolar = -23.44 * cos(dcoseno)
```

$$B = \frac{360}{365}(D - 81)$$

//formula de anomalía solar [5]

// anomalia solar//

B = ((360/365 * (n - 81)) * PI/180)

$$EoT = 9.87(2B) - 7.53cos(B) - 1.5\sin(B)$$

//Formula de la ecuación del tiempo [6]

// formula de la ecuacion del tiempo //

Eot = 9.87 * sen(2 * B) - 7.53 * cos(B) - 1.5 * sen(B)

// sacar la hora local para usarla en la formula de la ecuación del tiempo//

HoraLocal = hora + (minutos / 60)

// formula de la ecuacion del tiempo //

ZonaHoraria = -5

LongituEstandar = -5 * 15

$$TC = hora\ local + \frac{4(longitud - Longitud\ estandar)}{60} + Eot$$

//formula de tiempo solar [6]

// tuve que dividir la formula de La Ecuación del tiempo por partes //

tsvPre = 4 * (Longi - LongituEstandar) + Eot

tsvPree = tsvPre / 60

Tsv = HoraLocal + tsvPree

$$H = 15° X (TSV - 12)$$

//formula de ángulo horario [6]

// h es angulo horario solar //

H = ((15 * (Tsv - 12))* PI/180)

$$a = \sin\ ^1(\sin(\delta)\, x\, \sin(\phi) + \cos(\delta) * \cos(\phi) * \cos(H)$$

//formula de inclinación solar [7]

// Formula de angulo de inclinacion del sol //

Latitudrad = Latid * PI/180

declinacionsolarRad = declinacionsolar * PI/180

angulo = asen(sen(declinacionsolarRad) * sen(Latitudrad) + cos(declinacionsolarRad) * cos(Latitudrad) * cos(H))

angulogrados = angulo * 180/PI

$$Azimut = \cos \frac{(\sin(\delta) - \sin(\alpha) * \sin(\phi)}{\cos(\alpha) * \cos(\phi)}$$

//Formula ángulo de azimut [8]

// Formula del Zimuth//

azimuth = (sen(declinacionsolarRad) - sen(angulo) * sen(Latitudrad)) / (cos(angulo) * cos(Latitudrad))

azimuth2 = acos(azimuth)

Si H > 0 Entonces

azimuth2 = (2*PI)-azimuth2

FinSi

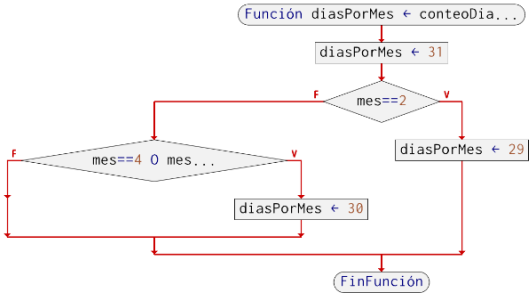
azimuthgrados = azimuth2 * 180/PI

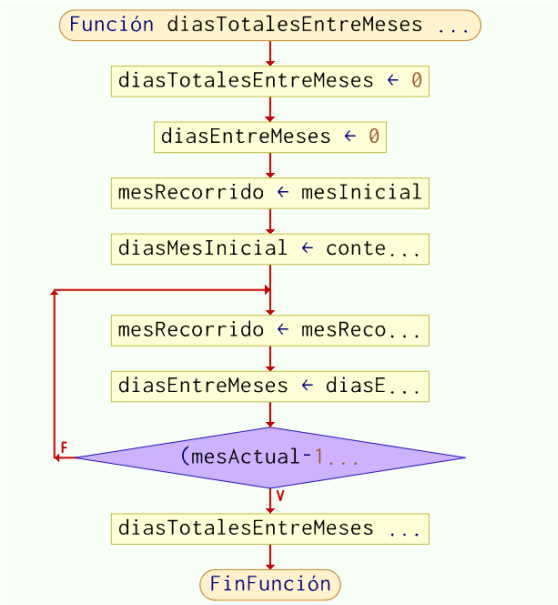
// pongo una condicion de si para cuando sea positivo o negativo//

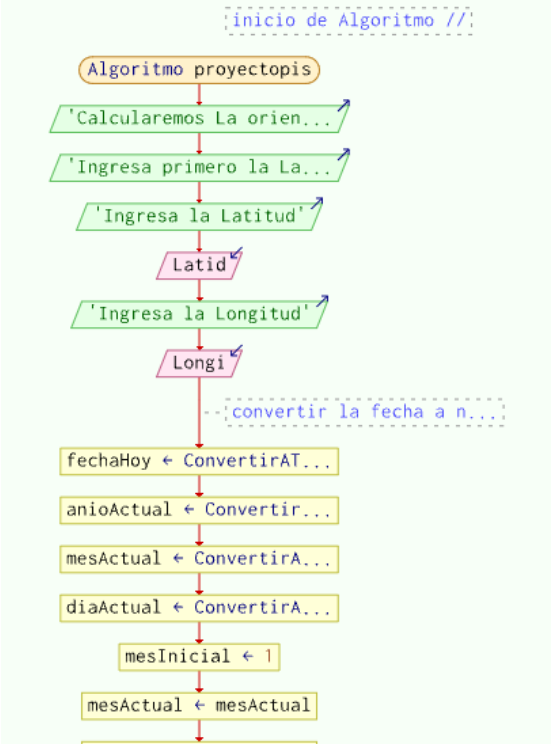
Escribir "La altura del sol es " angulogrados " y tiene una orientación de " azimuthgrados " grados respecto al norte"

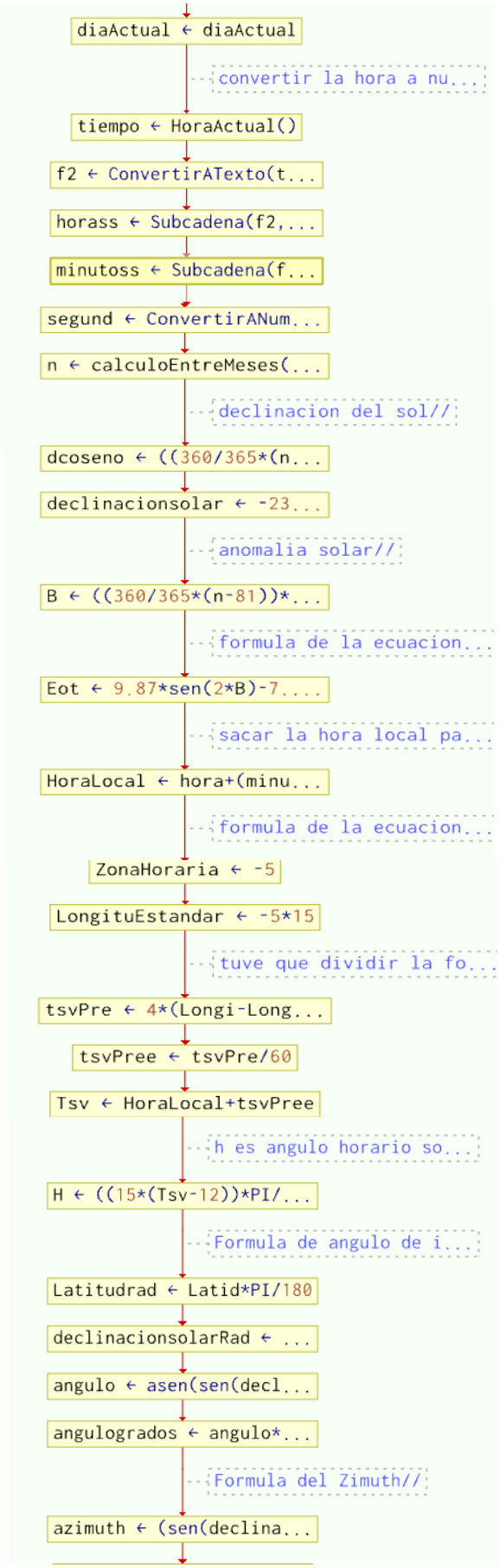
FinAlgoritmo

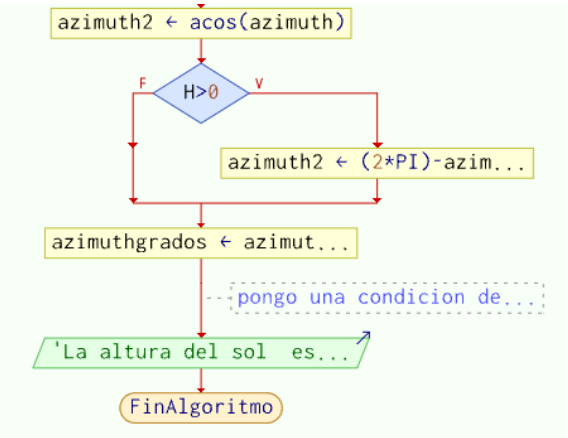
Diagramas de flujo:











Además del uso de una estructura para almacenar el valor de algunas variables de fecha y hora, el algoritmo no presenta cambios en su estructura, lógica o funcionamiento. En vista de lo anterior, se usará el mismo diagrama de flujo para ambos

VII.Discusión

En función del tiempo, se ha elaborado un algoritmo que cumple con las funciones esenciales para lograr una precisión de seguimiento solar usando la longitud y latitud de la ubicación de interés. Sin embargo, aún se requieren mas ajustes y no se ha considerado todos los escenarios posibles. Por ejemplo, se necesita ajustar el conteo de días para años no bisiestos que para el presente proyecto no era necesario ya que el año actual es bisiesto, pero sería necesario en caso que se deseara implementar a una mayor escala. En conclusión, es un buen prototipo que se podría implementar en un sistema real, pero debería tener algunos ajustes en su lógica al igual que implementar mas funciones y/o variables que consideren otros escenarios y así el sistema sea más eficiente y autónomo.

VIII.Bibliografía

[1] M. Mayer, M. Cáceres, A. Firman y L. Horacio, «Desarrollo de algoritmos de control de un sistema seguidor para la medición de los componentes de la radiación solar,» *Extensionismo, Innovación Y Transferencia Tecnológica,* vol. 4, pp. 198-210, 2018.

[2] J. Blanco R y E. Perez, «La variación anual de los ángulos solares en la latitud de Santa Marta y su importancia local,» *Boletin de Investigaciones Marinas y Costeras,* vol. 38, nº 1, pp. 151-169, 2009.

[3] J. Romero Castillo, «UPCommons,» Enero 2015. [En línea]. Available: http://hdl.handle.net/2099.1/26396. [Último acceso: 13 Mayo 2024].

[4] C. Honsberg y S. Bowden, «Pveductaion,» [En línea]. Available: https://www.pveducation.org/es/fotovoltaica/2-propiedades-de-la-luz-del-sol/angulo-de-declinaci%C3%B3n. [Último acceso: 14 junio 2024].

[5] C. Honsberg y S. Bowden, «Pveducation,» [En línea]. Available: https://www.pveducation.org/es/fotovoltaica/2-propiedades-de-la-luz-del-sol/angulo-de-declinaci%C3%B3n. [Último acceso: 15 junio 2024].

[6] C. Honsberg y S. Bowden, «Pveducation,» [En línea]. Available: https://www.pveducation.org/es/fotovoltaica/2-propiedades-de-la-luz-del-sol/hora-solar. [Último acceso: 15 junio 2024].

[7] C. Honsberg y S. Bowden, «Pveducation,» [En línea]. Available: https://www.pveducation.org/es/fotovoltaica/2-propiedades-de-la-luz-del-sol/el-%C3%A1ngulo-de-elevaci%C3%B3n. [Último acceso: 15 junio 2024].

[8] C. Honsberg y S. Bowden, «Pveducation,» [En línea]. Available: https://www.pveducation.org/es/fotovoltaica/2-propiedades-de-la-luz-del-sol/%C3%A1ngulo-acimut. [Último acceso: 15 junio 2024].