

# **Informe final: Proyecto Integrador de Saberes (PIS)**

## **Realizado por el Grupo 6:**

Integrante 1: Anthony Yaguana

Correo: [anthony.yaguana@unl.edu.ec](mailto:anthony.yaguana@unl.edu.ec)

Integrante 2: Narcisa Pinzón

Correo: [narcisa.pinzon@unl.edu.ec](mailto:narcisa.pinzon@unl.edu.ec)

Integrante 3: Camila Chimbo

Correo: [camila.chimbo@unl.edu.ec](mailto:camila.chimbo@unl.edu.ec)

Integrante 4: Santiago Villamagua

Correo: [santiago.villamagua@unl.edu.ec](mailto:santiago.villamagua@unl.edu.ec)

Integrante 5: Darwin Correa

Correo: [darwin.correa@unl.edu.ec](mailto:darwin.correa@unl.edu.ec)

## **Ciclo:**

Primer Ciclo de la carrera (Ingeniería en Sistemas)

## **Materias Involucradas:**

Matemáticas Discretas

Algebra Lineal

Teoría de la Programación

Electricidad

Comunicación y Redacción Técnica

## ÍNDICE DEL CONTENIDO

<b>1. Introducción.....</b>
<b>1.1 Definición del problema.....</b>
<b>1.2 Propósito general.....</b>
<b>1.3 Trascendencia y limitaciones.....</b>
<b>2. Objetivos.....</b>
<b>2.1 Objetivo General.....</b>
<b>2.2 Objetivos Específicos.....</b>
<b>3. Desarrollo del proyecto.....</b>
<b>3.1 Algoritmo de control.....</b>
<b>3.1.1 Descripción del algoritmo de control.....</b>
<b>3.1.2 Discusión para la implementación del código en paneles reales.....</b>
<b>3.1.3 Resultados esperados.....</b>
<b>3.2 Código realizado en C.....</b>
<b>3.2.1 Función principal del programa.....</b>
<b>3.2.2 Resultados y pruebas.....</b>
<b>3.3 Materiales utilizados en el proyecto.....</b>
<b>3.3.1 Materiales y Componentes del panel solar.....</b>
<b>3.4 Diseño del circuito.....</b>
<b>3.4.1 Esquema del circuito.....</b>
<b>4. Enfoque en la sostenibilidad.....</b>
<b>4.1 Impacto Ambiental.....</b>
<b>4.2 Eficiencia energética.....</b>
<b>5. Mejoras futuras.....</b>
<b>6. Conclusiones.....</b>
<b>7. Recomendaciones.....</b>
<b>8. Bibliografía.....</b>
<b>9. Anexos .....</b>

# 1. Introducción

## 1.1 Definición del problema:

Uno de los problemas más conocidos y resaltados en la sociedad actualmente es la contaminación del medio ambiente, donde se han propuesto múltiples soluciones en varios campos ya sean tecnológicos y ambientales, en el cual el campo tecnológico se ha propuesto implementar el uso y aprovechamiento de energías que sean amigables con el medio ambiente. Para ello en el presente ciclo se propuso como objetivo de Proyecto Integrador de Saberes (PIS) construir un Cargador Solar Automatizado, que haga uso de energía solar como medio sostenible, económico, amigable y favorable para la sociedad y el medio ambiente, implementando los conocimientos adquiridos a lo largo del ciclo académico en las tres unidades que constan del mismo.

El siguiente informe indicará el proceso efectuado para la construcción del proyecto, resaltando los conocimientos empleados en cada una de las asignaturas, y especificando la relación que mantiene cada materia en la implementación del cargador solar.

## 1.2 Propósito general

El proyecto presentado a continuación tiene como objetivo diseñar un panel solar enfocado principalmente en la sostenibilidad el cual contenga materiales sostenibles que proporcionen un cambio en el impacto de la contaminación ambiental.

## 1.3 Trascendencia y limitaciones

El proyecto se basa en la construcción de un panel solar con materiales de bajo impacto y componentes eléctricos específicos para llevar a cabo la eficiencia en la captación de energía solar.

## 2. Objetivos

### 2.1 Objetivo general:

Crear un algoritmo que permita calcular la altitud y orientación solar, en la que exista mayor captación para que sea orientado y sea colocado el panel solar para obtener mejores resultados, utilizando los siguientes datos: altitud y latitud de la posición actual, la fecha y hora del sistema. Además implementar un diseño electrónico el cual sea comunicado con el algoritmo realizado para que este sea dirigido automáticamente.

### 2.2 Objetivos específicos

- Utilización y comprensión de conceptos básicos aplicados en la lógica de programación, además de la aplicación de algoritmos en proyectos de baja escala programables y observar la eficiencia de los mismos.

- Manejo de componentes electrónicos básicos con integración de microcontroladores en proyectos básicos ya simulados anteriormente en plataformas de diseño y simulación.

## 3. Desarrollo del proyecto

### 3.1 Algoritmo de control

#### 3.1.1 Descripción del algoritmo de control

##### Análisis de variables implementadas en pseudocódigo

Para el presente algoritmo se han identificado varias variables que se utilizarán en el transcurso en el que se desarrolla el algoritmo.

##### Variables principales:

-Latitud y Longitud: Estas se pueden obtener, dirigiéndose a la aplicación de Google Maps ya sea desde una computadora o un celular dando un clic o un touch en la ubicación que se encuentra así serán proporcionadas automáticamente y exactas para que sea más eficiente la información ingresada al programa.

-Fecha y Hora actual: Estos datos cambian constantemente se pueden calcular de una manera automática, solo se debe leer la fecha y hora del sistema para usarlas de forma conveniente. Dentro de la fecha actual se tienen en cuenta variables como: (dia, mes y año).

En la hora actual sólo se utilizaron dos variables para guardar la hora actual y los minutos actuales de estas variables:(hora y minutos).

Variables requeridas a lo largo del algoritmo: Estas variables se utilizaron para guardar los datos obtenidos de los diferentes cálculos realizados en el algoritmo.

##### Variables del programa:

###### • Variables decimales.

-double: Longitud, Latitud, x2, y2, z2, x1, y1, z1, angulo.  
-double: Declinacion\_Solar, EoT, TSV, Angulo\_Horario, Altitud\_Solar, Orientacion\_Solar, Azimut, Hora\_Solar, Minutos\_Solar.

###### • Variables enteras.

-int: horas, minutos, segundos, N\_Dias, años, meses, días, opción, contador, servidor.

**Anexo 1: Explicación del pseudocódigo del algoritmo:** Este es el funcionamiento general del algoritmo para calcular la altitud y orientación solar en la que se debe colocar y dirigir el panel fotovoltaico.

Primero se obtienen los datos necesarios para hacer los respectivos cálculos y así obtener los ángulos necesarios, se tienen en cuenta los datos de longitud y altitud, además de la fecha y hora actual del sistema, separándolas en sus respectivas variables.

Con estos datos iniciales se procede a hacer los cálculos con apoyó de las funciones antes ya definidas que se explicarán a continuación del documento.

Cabe recalcar que la forma en que se obtiene cada dato es secuencial, es decir, calculamos un dato necesario para después usarlo en otra ecuación y así obtener los datos necesarios.

Entonces los pasos del algoritmo son:

1.Definir la longitud y latitud del lugar en que se encuentra.

2.Recuperación de la fecha y hora actual del sistema.

3.Calcular el número de días que han transcurrido (N) desde que inició el año hasta el día actual usando las variables de mes, dia y anio.

4.Con los días totales calculados se procede a hacer el respectivo cálculo de la declinación solar (Declinacion).

5.Hacer uso también de los días totales transcurridos (N) para proceder a resolver la ecuación del tiempo (EoT).

6.Calcular el tiempo solar verdadero (TSV), utilizando la hora actual (hora), los minutos actuales (minutos), la longitud de nuestra ubicación (Longitud) y el resultado de la ecuación del tiempo (EoT).

7.Hacer el cálculo de la hora solar (hora\_solar) y los minutos solares (minutos\_solar), para la hora solar utilizando la parte entera del tiempo solar verdadero (TSV) y para los minutos solares.

8.Hacer el cálculo del ángulo horario (H), con ayuda del resultado del tiempo solar verdadero (TSV).

9.Acontecer al cálculo de la declinación solar (Declinación) y la latitud (Lati\_tud) a radianes usando la fórmula general de conversión a radianes que sería: variable \* (PI/180).

10.Con todos los datos obtenidos anteriormente se procede a la realización de cada uno de los dos datos finales de nuestro algoritmo, la altitud solar (AS), que se denota como los grados de inclinación que debe tener nuestro panel solar obtenido con la declinación solar (Declinacion), la latitud (Latitud) y el ángulo horario (H). Cabe recalcar que todos los datos usados para calcular la altitud solar deben estar en radianes.

11.Finalmente se realiza la obtención de la orientación solar (Azimut) que son los grados en que debe estar dirigido el panel fotovoltaico con respecto al norte, para calcular la orientación solar (Azimut) usamos los datos respectivos de la declinación solar (Declinación), la altitud solar (AS), la latitud con respecto a la ubicación actual (Latitud) y el ángulo horario (H). En este caso se utilizó todos los ángulos en radianes.

## Anexo 2

### Funciones definidas en el algoritmo

Función Calcular\_numero\_dias (N): En esta función se pasan como parámetros el mes actual (mes), el día actual(dia) y el año actual (anio), tomando un bucle “Para” y un “Según” para hacer una iteración desde el mes 1 que sería enero hasta un mes anterior al actual esto debido a que el mes actual no está totalmente cursado para determinar el número de mes y por cada número de mes se suma la cantidad de días que tiene cada mes almacenando esta cantidad de días en la variable (N).

Al finalizar se realiza un análisis para verificar si el año actual es un año bisiesto, de ser así se suma un 1 día a la cantidad de días calculados anteriormente ya que los años bisiestos cuentan con un día más que los años normales.**Anexo 3**

Función Calcular\_angulo\_declinación (Declinacion): Esta función recibe como parámetro el número de días (N) que han transcurrido desde que inició el año en el mes de enero hasta el día actual mediante la fórmula:

$$\delta = -23.45^\circ \times \cos\left(\frac{360}{365} \times (d + 10)\right)$$

[imagen 1]

A esta fórmula se le complementan ciertos parámetros que se manifiestan de la siguiente manera:

Término \_1: Este valor se debe a que es el ángulo de inclinación máximo de la tierra respecto al plano de su órbita alrededor del sol.

Término \_2: Se establece una división de  $360^\circ$  entre el número de días del año multiplicado al número de días que han pasado más 10, para compensar cierto tiempo que se pudo haber perdido

Finalmente, al realizar estos cálculos se obtiene el ángulo de la declinación solar para el día N. **Anexo 4**

Función Calcular\_ecuacion\_tiempo (EoT): Para llevar a cabo este cálculo se tiene como referencia la Ecuación del Tiempo (EoT) en minutos denotada como una ecuación que corrige la excentricidad de la órbita de la tierra y la inclinación del eje de la tierra donde:

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

[imagen 2]

Esta fórmula se utiliza para calcular el ángulo del horario del sol restando al día 81 para ajustar el día del año para que se centre alrededor del solsticio de invierno.

$$B = 360 * (N - 81) / 365$$

[imagen 3]

Luego se convierte el ángulo del horario del sol de grados a radianes multiplicando ( $B * \pi/180$ ) y finalmente se utiliza el ángulo calculado para determinar la Ecuación del tiempo.**Anexo 5**

Función Calcular\_TSV (TSV): Esta función se encarga de calcular la hora solar verdadera (TSV) en función de la hora local, longitud, zona horaria y ecuación del tiempo (EoT).**Anexo 6**

$$TSV = \text{Hora local} + \frac{4 \cdot (\text{Longitud local} - \text{Longitud estándar}) + EoT}{60}$$

[imagen 4]

Función Calcular\_angulo\_horario (H): Pasamos como parámetros el tiempo solar verdadero (TSV) calculado en la anterior función, para calcular este ángulo horario nos ayudamos de la fórmula:

$$H = 15^\circ \times (TSV - 12)$$

[imagen 5]

Luego convertimos en radianes con la fórmula general de conversión a radianes para así poder trabajar este dato adecuadamente. **Anexo 7**

Función Calcular\_AS: En esta función se usan como parámetros los valores anteriormente calculados, para hacer uso de la declinación solar (Declinacion), la latitud de nuestra ubicación actual (Lati\_tud) y el ángulo horario (H).

Con los parámetros mencionados se procede a utilizar la siguiente fórmula para poder calcular la altura solar (AS):

De esta manera obtenemos la inclinación correspondiente del panel solar (AS).

$$\alpha = \sin^{-1}[\sin \delta \sin \varphi + \cos \delta \cos \varphi \cos(HRA)]$$

[imagen 6]

Función Calcular\_Azimut: Con esta función se procede a calcular la orientación (Azimut) que debe tener nuestro panel solar con respecto al norte, pasando los parámetros correspondientes que serían: la declinación solar (Declinacion), la altitud solar (AS), la latitud de nuestra ubicación actual (Lati\_tud) y el ángulo horario (H). De la misma manera que en las anteriores funciones, tomando la fórmula correspondiente para calcular el azimut. **Anexo 8**

$$\text{Azimut} = \cos^{-1} \left( \frac{\sin(\delta) - \sin(\alpha) \cdot \sin(\phi)}{\cos(\alpha) \cdot \cos(\phi)} \right)$$

[imagen 7]

#### Coordenadas de posición

$$x = \cos(\alpha) \cdot \sin(\theta)$$

$$y = \cos(\alpha) \cdot \cos(\theta)$$

$$z = \sin(\alpha)$$

[ imagen 8]

Las coordenadas de posición se utilizan para representar la posición del sol en un momento dado mediante x, y, z, además de hacer uso de la altitud y el azimut.

#### Ángulo entre vectores

$$\theta = \cos^{-1} \left( \frac{\mathbf{a} \cdot \mathbf{b}}{\|\mathbf{a}\| \|\mathbf{b}\|} \right)$$

[imagen 9]

El ángulo entre vectores es el arco que forman dos segmentos de vectores previamente unidos por un punto. Al encontrar este ángulo se pudo obtener el movimiento del panel solar en un momento dado propiciando un movimiento efectivo.

#### 3.1.2 Discusión para la implementación del código en paneles reales.

El software utilizado para realizar el panel solar puede extenderse ampliamente beneficiando a todo aquel que lo necesite y sea un amante de los recursos naturales que ofrece la naturaleza como fuente de energía propiciando al ahorro de la energía eléctrica, además no solo puede un panel solar tener múltiples usos, empezando como un cargador solar hasta como un implemento de alumbrado en las viviendas, como aire acondicionado o inclusive como una fuente de energía para duchas de agua caliente, el software puede ser modificado para infinidad de usos y no solo destinarse a uno solo. Pero para lograrlo debemos considerar algunos aspectos como:

- **Interfaz con hardware:** El código proporcionado debe comunicarse eficientemente con los controladores de los paneles solares.
- **Gestión de datos:** Debe manejar los datos de manera eficiente y segura. Esto incluye el almacenamiento local en el sistema de control y la transmisión segura de datos a través de redes.
- **Control y monitoreo en tiempo real:** El sistema debe ser capaz de monitorear constantemente el rendimiento de los paneles, la energía generada, y otros parámetros relevantes.
- **Seguridad y redundancia:** Se deben implementar medidas de seguridad robustas para proteger el sistema contra accesos no autorizados y para garantizar que el sistema pueda manejar situaciones de emergencia de manera adecuada.
- **Optimización de eficiencia:** El código debe ser optimizado para minimizar el consumo de recursos computacionales y energéticos, asegurando que el sistema funcione de manera eficiente tanto en términos de hardware como de energía.

#### 3.1.3 Resultados esperados

Se espera que el proyecto presentado cumpla con los requerimientos, objetivos y resultados planteados inicialmente.

Ya que así se demuestra la capacidad para satisfacer las especificaciones y expectativas ya establecidas, además que dentro del proyecto se podrán realizar mejoras para optimizarlo para obtener una eficiencia energética óptima para su uso en la sociedad.

### 3.2 Código realizado en C

El código ha sido desarrollado en C un lenguaje de alto nivel ideal para dar inicio en el mundo de programación. Ya que C ayuda a entender cómo aplicar la lógica de programación necesaria para el entendimiento de cómo es el funcionamiento de sistemas informáticos; además ofrece funciones y herramientas necesarias para cumplir con los estándares de un software eficiente.

#### 3.2.1 Función principal del programa.

El programa realizado a continuación tiene como objetivo optimizar la orientación de un panel solar para maximizar la captación de energía solar. La captación eficiente de energía solar depende de la orientación precisa del panel respecto al sol en cualquier momento del día y del año.

#### Inicio del Programa en C.

**Definición de bibliotecas y constantes utilizadas en el código:**

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <time.h>
#include <unistd.h>
#include <windows.h> // Biblioteca para manejar comunicación serial en Windows
#ifndef M_PI
#define M_PI 3.14159265358979323846
#endif
```

<stdio.h> .- Librería empleada para la entrada y salida de datos.

<math.h>.- Librería utilizada para resolver funciones matemáticas.

<time.h>.- Librería para obtener la fecha y hora en tiempo real.

<stdlib.h>.- Librería para administrar memoria.

<unistd.h>.- Librería POSIX utilizada en sistemas de UNIX y Linux para acceder al API del sistema operativo.

<windows.h>.- Librería específica para Windows para acceder al API del mismo.

#define .- Se implementa para tomar una variable como constante en el programa.

#### Función con una variable entera para calcular el número de días.

Esta función permite calcular el número total de días que han transcurrido hasta la fecha actual, la cual ayudará para los cálculos pertinentes y exactos que se llevaron a cabo a lo largo del programa.

```
int Calcular_Numero_Dias(int meses, int dias, int anios)
{
    int N_Dias = 0;

    for (int i = 1; i <= meses; i++)
    {
        if (i == 1 || i == 3 || i == 5 || i == 7 || i == 8 || i == 10 || i == 12)
        {
            N_Dias += 31;
        }
        else if (i == 4 || i == 6 || i == 9 || i == 11)
        {
            N_Dias += 30;
        }
        else
        {
            N_Dias += 28;
        }
    }

    N_Dias = N_Dias + dias;

    if (anios % 4 == 0)
    {
        if (anios % 100 == 0)
        {
            if (anios % 400 == 0)
            {
                N_Dias = N_Dias + 1;
            }
        }
        else
        {
            N_Dias = N_Dias + 1;
        }
    }
    N_Dias+= 1;
    return N_Dias;
}
```

**Función con una variable (double) esta variable maneja decimales esta función nos ayudará para calcular el ángulo de Declinación incluyendo la variable (int N\_Dias).**

Esta función calcula la declinación solar para un día específico del año, representado por (N\_Dias), considerando el efecto de la inclinación del eje terrestre y la variación anual del ángulo solar.

```
double Calcular_Angulo_Declinacion_Solar(int N_Dias)
{
    double Termino1, Termino2, DS_Radianes, Coseno, Declinacion_Solar;
    Termino1 = -23.44;
    Termino2 = (360.0 / 365.0) * (N_Dias + 10);
    DS_Radianes = Termino2 * (M_PI / 180.0);
    Coseno = cos(DS_Radianes);
    Declinacion_Solar = Termino1 * Coseno;
    return Declinacion_Solar;
}
```

**Función con una variable (double) esta variable maneja decimales, esta función nos ayudará para calcular la Ecuación del tiempo incluyendo la variable (int N\_Dias).**

Calcula la ecuación del tiempo teniendo en cuenta las variaciones a la excentricidad de la órbita e inclinación de su eje.

```
double Ecuacion_Del_Tiempo(int N_Dias)
{
    double B, EoT;

    B = ((360.0 / 365.0) * (N_Dias - 81));
    B = B * (M_PI / 180.0);
    EoT = 9.87 * sin(2 * B) - 7.53 * cos(B) - 1.5 * sin(B);
    return EoT;
}
```

## Función con una variable (double) esta variable maneja decimales esta función nos ayudará para calcular el Tiempo Solar.

Esta función permitirá ajustar la orientación de los paneles solares de manera más precisa y eficiente, siguiendo el movimiento del sol en el cielo a lo largo del día.

```
double Calcular_Tiempo_Solar_Verdadero(int horas, int minutos, double Longitud, double EoT)
{
    int Zona_Horaria;
    double Hora_Local, Longitud_Estandar, TSV;

    Hora_Local = horas + (minutos / 60.0);
    Zona_Horaria = -5;
    Longitud_Estandar = Zona_Horaria * 15;
    TSV = Hora_Local + (((4 * (Longitud - Longitud_Estandar)) + EoT) / 60.0);
    return TSV;
}
```

## Función con una variable (double) esta variable maneja decimales esta función nos ayudará para calcular el Ángulo .

Esta función calcula el ángulo horario se utiliza para ajustar la inclinación de los paneles de manera que estén orientados hacia el sol para captar la máxima cantidad de radiación solar.

```
double Calcular_Angulo_Horario(double TSV)
{
    double Angulo_Horario;
    Angulo_Horario = 15 * (TSV - 12);
    Angulo_Horario = Angulo_Horario * (M_PI / 180.0);
    return Angulo_Horario;
}
```

## Función con una variable (double) esta variable maneja decimales esta función nos ayudará para calcular el Altitud.

Con esta función podremos obtener la altitud solar que se utiliza para calcular la intensidad de la radiación solar incidente en los paneles y ajustar la inclinación de estos paneles para maximizar la captación de energía solar.

```
double Calcular_Altitud_Solar(double Declinacion_Solar, double Latitud, double Angulo_Horario)
{
    double Altitud_Solar;
    Altitud_Solar = asin(sin(Declinacion_Solar) * sin(Latitud) + cos(Declinacion_Solar) * cos(Latitud) * cos(Angulo_Horario));
    return Altitud_Solar;
}
```

## Función con una variable (double) esta variable maneja decimales esta función nos ayudará para calcular la Azimut.

En sistemas de seguimiento solar, el azimut solar se utiliza para ajustar continuamente la orientación de los paneles solares a lo largo del día y del año.

```
double Calcular_Azimut(double Declinacion_Solar, double Altitud_Solar, double Latitud, double Angulo_Horario)
{
    double Azimut;
    Azimut = acos((sin(Declinacion_Solar) - sin(Altitud_Solar) * sin(Latitud)) / (cos(Altitud_Solar) * cos(Latitud)));
    if (Angulo_Horario > 0)
    {
        Azimut = 2 * M_PI - Azimut;
    }
    return Azimut;
}
```

## Funcion (Pedir\_Datos).

Solicita al usuario ingresar un valor de tipo (double) si no es así este se reiniciará hasta que el usuario ingrese un valor válido en el sistema.

```
double Pedir_Datos(const char *mensaje)
{
    double valor;
    int resultado;

    while (1)
    {
        printf("%s", mensaje);
        resultado = scanf("%lf", &valor);

        // Limpiar el dato de entrada para manejar las entradas incorrectas
        while (getchar() != '\n');

        if (resultado == 1)
        {
            return valor;
        }
        else
        {
            printf("**** Dato invalido, por favor ingrese bien el dato que se a solicitado ****\n");
        }
    }
}
```

## Función (vector\_final).

Convierte los ángulos de azimut y altitud en las componentes de un vector tridimensional almacenando sus resultados en las variables ya declaradas.

```
void vector_final(double *x2, double *y2, double *z2, double Azimut, double Altitud_Solar)
{
    *x2 = cos(Altitud_Solar) * sin(Azimut);
    *y2 = cos(Altitud_Solar) * cos(Azimut);
    *z2 = sin(Altitud_Solar);
}
```

## Función (vector\_inicial).

Calcula las componentes de un vector tridimensional en su posición inicial basado en la fecha, hora y coordenadas actuales usando funciones auxiliares para calcular sus ángulos y almacenarlos en las variables correspondientes.

```
void vector_inicial(double *x1, double *y1, double *z1, double Longitud, double Latitud)
{
    int horas, minutos, segundos, anios, meses, dias;
    time_t tiempoahora;
    time(&tiempoahora);
    struct tm *mitiempo = localtime(&tiempoahora);

    anios = mitiempo->tm_year + 1900;
    meses = mitiempo->tm_mon + 1; // Meses desde 0
    dias = mitiempo->tm_mday;

    printf("Por favor ingrese los siguientes datos para obtener la posicion inicial\n");
    printf("Ingrese la hora: ");
    scanf("%d", &horas);
    printf("Ingrese los minutos: ");
    scanf("%d", &minutos);
    printf("Ingrese los segundos: ");
    scanf("%d", &segundos);

    int NDias = Calcular_Numero_Dias(meses - 1, dias, anios);
    double Declinacion_Solar = Calcular_Angulo_Declinacion_Solar(NDias);
    double EoT = Ecuacion_Del_Tiempo(NDias);
    double TSV = Calcular_Tiempo_Solar_Verdadero(horas, minutos, Longitud, EoT);
    double Angulo_Horario = Calcular_Angulo_Horario(TSV);
    double Altitud_Solar = Calcular_Altitud_Solar(Declinacion_Solar, Latitud, Angulo_Horario);
    double Azimut = Calcular_Azimut(Declinacion_Solar, Altitud_Solar, Latitud, Angulo_Horario);

    *x1 = cos(Altitud_Solar) * sin(Azimut);
    *y1 = cos(Altitud_Solar) * cos(Azimut);
    *z1 = sin(Altitud_Solar);
}
```

## Función (angulo\_placa).

Calcula en ángulo entre dos vectores tridimensionales en grados usando el producto punto además de las magnitudes vectoriales asegurando que el coseno esté dentro del rango

```
double angulo_placa(double x2, double x1, double y2, double y1, double z2, double z1) {
    double producto_punto = x2 * x1 + y2 * y1 + z2 * z1;
    double magnitud_inicial = sqrt(x1 * x1 + y1 * y1 + z1 * z1);
    double magnitud_final = sqrt(x2 * x2 + y2 * y2 + z2 * z2);
    double cos_angulo = producto_punto / (magnitud_final * magnitud_inicial);
    cos_angulo = fmax(-1.0, fmin(1.0, cos_angulo));
    return acos(cos_angulo) * (180.0 / M_PI);
}
```

### Función (Calcular\_Angulo\_Base).

Garantiza que el ángulo del azimut se mantenga en el rango de (0,360) grados.

```
double Calcular_Angulo_Base(double azimutSolar) {
    double anguloBase = azimutSolar;
    if (anguloBase < 0) anguloBase = 0;
    if (anguloBase >= 360) anguloBase = 360;
    return anguloBase;
}
```

### Función (ingreso\_de\_usuario)

Almacena los datos de hora, minutos y segundos ingresados por el usuario en las variables.

```
void ingreso_de_usuario(int *horas, int *minutos, int *segundos) {
    printf("Ingrese la hora: ");
    scanf("%d", horas);
    printf("Ingrese los minutos: ");
    scanf("%d", minutos);
    printf("Ingrese los segundos: ");
    scanf("%d", segundos);
}
```

### Función (opcion\_menu).

Muestra un menú con dos opciones en la cual el usuario tendrá la posibilidad de orientar el cargador solar de manera automática o manual.

```
void opcion_menu() {
    printf("MENU\n");
    printf("Opcion 1: Posicion automatica [1]\n");
    printf("Opcion 2: Posicion manual [2]\n");
}
```

### Función (calcularhora).

Obtiene la fecha y hora actual ajustando desde el año 1900 y los meses los almacena con valores de (0-11) luego los almacena en las variables correspondientes.

```
void calcularhora(int *horas, int *minutos, int *segundos, int *anios, int *meses, int *dias) {
    time_t tiempoahora;
    time(&tiempoahora);
    struct tm *mitiempo = localtime(&tiempoahora);
    *horas = mitiempo->tm_hour;
    *minutos = mitiempo->tm_min;
    *segundos = mitiempo->tm_sec;
    *anios = mitiempo->tm_year + 1900;
    *meses = mitiempo->tm_mon;
    *dias = mitiempo->tm_mday;
}
```

### Funcion (incluir\_resultados).

Ayuda a gestionar la entrada de datos, además de realizar los cálculos correspondientes integrados en la función y finalmente muestra los resultados finales.

```
void incluir_resultados(double longitud, double latitud, double x2, double y2, double z2, double x1, double y1, double z1, double angulo, int horas, int minutos, int segundos, int N_Dias, int anios, int meses, int dias, double Declinacion_Solar, double EoT, double TSV, double Angulo_Horario, double Altitud_Solar, double Orientacion_Solar, double Azimut, double Hora_Solar, double Minutos_Solar, int opcion) {
    if (opcion == 2) {
        ingreso_de_usuario(&horas, &minutos);
    } else {
        calculaHora(&horas, &minutos, &segundos, &anios, &meses, &dias);
    }
    printf("\n");
    printf("// FECHA Y HORA ACTUAL DEL SISTEMA\n");
    printf("Horas => %d\n", horas);
    printf("Minutos => %d\n", minutos);
    printf("Segundos => %d\n", segundos);
    printf("Anios => %d\n", anios);
    printf("Mes => %d\n", meses);
    printf("Dia => %d\n", dias);
    printf("\n");
    printf("// Iniciamos el Programa llamando y asiendo los valores a cada una de las funciones antes definidas //\n");
    printf("// VERIFICACION DE DATOS //\n");
    N_Dias = Calcular_Numeros_Dias(meses, dias, anios);
    printf("// Dias transcurridos desde que inicio el año hasta el dia actual usando la funcion correspondiente antes definida //\n");
    Declinacion_Solar = Calcular_Angulo_Declinacion_Solar(N_Dias);
    printf("// Declinacion Solar usando la funcion correspondiente y pasando los parametros por valores necesarios para la operacion //\n");
    Declinacion_Solar = Declinacion_Solar * M_PI;
    printf("// Calculamos la ecuacion del tiempo (EoT) y pasamos los parametros necesarios para el calculo //\n");
    EoT = Calcular_Del_Tiempo(N_Dias);
    printf("// Ecuacion del tiempo => %f\n", EoT);
    printf("// Calculamos el tiempo solar verdadero pasando como referencia los datos necesarios para el calculo //\n");
    TSV = Calcular_Tiempo_Solar_Verdeadero(horas, minutos, Longitud, EoT);
    horas_decimal = trunc(TSV);
    horas_decimal = round((TSV - horas_decimal) * 60);
    printf("// Tiempo solar verdadero => %f\n", TSV);
    printf("// Calculamos el angulo horario usando como parametro el tiempo solar verdadero (TSV) //\n");
    Angulo_Horario = Calcular_Angulo_Horario(horas_decimal);
    printf("// Angulo Horario => %f\n", Angulo_Horario);
    printf("// Pasamos la declinacion solar y la latitud de grados a radianes //\n");
    Declinacion_Solar = Declinacion_Solar * (M_PI / 180.0);
    Latitud = Latitud * (M_PI / 180.0);
    printf("// Declinacion Solar en radianes => %f\n", Declinacion_Solar);
    printf("// Latitud en radianes => %f\n", Latitud);
    printf("// Calculamos la Altitud Solar en radianes //\n");
    Altitud_Solar = Calcular_Altitud_Solar(Declinacion_Solar, Latitud, Angulo_Horario);
    printf("// Altitud Solar a grados //\n");
    Altitud_Solar = Altitud_Solar * (180.0 / M_PI);
    printf("// Calculamos la orientacion Solar o Azimut en radianes //\n");
    Azimut = Calcular_Orientacion_Solar(Declinacion_Solar, Altitud_Solar * (M_PI / 180.0), Latitud, Angulo_Horario);
    printf("// Pasamos la orientacion Solar a grados //\n");
    Azimut = Azimut * (180.0 / M_PI);
    printf("// Finalmente presentando los resultados de altitud y orientacion del sol actual //\n");
    printf("// RESULTADOS FINALES DE ALTITUD DEL SOL Y SU ORIENTACION //\n");
    printf("// La posicion de la altitud del sol es de %f grados de elevacion desde el piso\n", Altitud_Solar);
    printf("// La posicion de la orientacion del sol es de %f grados desde el norte\n", Azimut);
    printf("//\n");
    printf("\n");
    printf("// ROTACION DE LOS SERVOS MOTORES //\n");
    vector_final(x2, y2, z2, Azimut, Altitud_Solar);
    angulo = angulo_placa(x2, x1, y2, z2, z1);
    printf("// El angulo de rotacion de la placa fotovoltaica es: %f\n", angulo);
    printf("\n");
    printf("// ROTACION DE LA BASE //\n");
    Azimut_Calcula_Angulo_Base(Azimut);
    printf("// El angulo de rotacion de la base es: %f\n", Azimut);
}
```

### Función principal del programa (main).

Se declaran las variables a utilizar, además de organizar el flujo del programa de manera ordenada conforme las funciones sean llamadas para poder presentar en la consola los datos y resultados de manera entendible para los usuarios que lo manipularon.

```
int main() {
    double Longitud, Latitud, x2, y2, z2, x1, y1, z1, angulo;
    int horas, minutos, segundos, N_Dias, anios, meses, dias, opcion, contador, servidor;
    double Declinacion_Solar, EoT, TSV, Angulo_Horario, Altitud_Solar, Orientacion_Solar, Azimut, Hora_Solar, Minutos_Solar;
    char salida;

    printf("// Bienvenido. Por favor ingrese su Longitud y Latitud actual //\n");
    Longitud = Pedir_Datos("1. Longitud (En un rango de -180 a 180 grados) => ");
    Latitud = Pedir_Datos("2. Latitud (En un rango de -90 a 90 grados) => ");
    vector_inicial(x1, y1, z1, Longitud, Latitud);
    char salida;

    while (salida != 'S' || salida != 's')
    {
        opcion_menu();
        printf("Seleccione una opcion: ");
        scanf("%d", &opcion);

        if (opcion == 1) {
            while (contador < 200)
            {
                incluir_resultados(Longitud, Latitud, x2, y2, z2, x1, y1, z1, angulo, horas, minutos,
                    segundos, N_Dias, anios, meses, dias, Declinacion_Solar, EoT, TSV, Angulo_Horario, Altitud_Solar, Orientacion_Solar, Azimut,
                    Hora_Solar, Minutos_Solar, opcion);
                contador++;
                sleep(10);
            }
        } else if (opcion == 2) {
            incluir_resultados(Longitud, Latitud, x2, y2, z2, x1, y1, z1, angulo, horas, minutos,
                segundos, N_Dias, anios, meses, dias, Declinacion_Solar, EoT, TSV, Angulo_Horario, Altitud_Solar, Orientacion_Solar, Azimut,
                Hora_Solar, Minutos_Solar, opcion);
        } else {
            printf("Opcion no valida. Intento de nuevo.\n");
        }
    }
    printf("Si desea salir presione S o sino presione cualquier tecla para continuar\n");
    scanf("%s", &salida);
    servidor++;
}

return 0;
```

### 3.2.2 Resultados y pruebas

El algoritmo previamente planificado e implementado en un lenguaje de programación para poder ser llevado a cabo arroja los siguientes resultados:

```
PS C:\Proyecto Final\Proyecto-Integrador-de-Saberes> .\Calcular_Posicion_Solar.exe
// Bienvenido, Por favor ingrese su Longitud y Latitud actual
//
1. Longitud (En un rango de -180 a 180 grados) => -79.28422
2. Latitud (En un rango de -90 a 90 grados) => .3.99313
Por favor ingrese los siguientes datos para obtener la posicion inicial
Ingrese la hora: 10
Ingrese los minutos: 23
Ingrese los segundos: 11
MENU
Opcion 1: Posicion automatica [1]
Opcion 2: Posicion manual [2]
Seleccione una opcion: 2
Ingrese la hora: 17
Ingrese los minutos: 23
Ingrese los segundos: 10
=====
// FECHA Y HORA ACTUAL DEL SISTEMA //
Horas => 17
Minutos => 23
Segundos => 10
Anio => 32758
Mes => -667211407
Dia => 347
=====
// VERIFICACION DE DATOS //
> Dias transcurridos desde que inicio el anio => 348
> Declinacion Solar => -23.270030
> Ecuacion del tiempo => 4.635990
> Tiempor solar verdadero => 17.180318
> Angulo Horario => 1.356204
> Declinacion Solar en radianes => -0.406139
> Latitud en radianes => 0.005236
=====
```

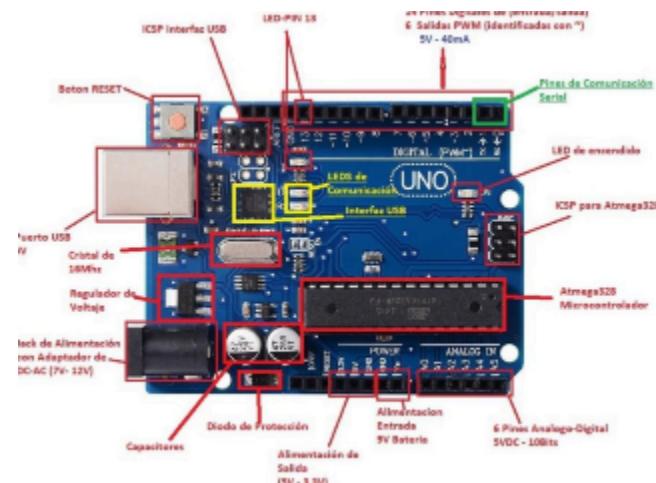
```
=====
// VERIFICACION DE DATOS //
> Dias transcurridos desde que inicio el anio => 213
> Declinacion Solar => 17.970483
> Ecuacion del tiempo => -6.017670
> Tiempor solar verdadero => 21.686091
> Angulo Horario => 2.535813
> Declinacion Solar en radianes => 0.313644
> Latitud en radianes => 0.005236
=====
// RESULTADOS FINALES DE ALTITUD DEL SOL Y SU ORIENTACION //
> La posicion de la altitud del sol es de => -51.290656 grados de elevacion desde el piso
> La posicion de la orientacion del sol es de => 299.992816 grados desde el norte
=====
// ROTACION DE LOS SERVO MOTORES //
> El angulo de rotacion de la placa fotovoltaica es: 114.099429
=====
// ROTACION DE LA BASE //
> El angulo de rotacion de la base es: 299.992816
=====
```

Como se muestra en la ejecución los resultados que se han obtenido es la orientación e inclinación en una hora específica, así mismo se puede observar la posición inicial y final en la que se encuentra el panel solar.

### **3.3 Materiales utilizados en el proyecto**

### **3.3.1 Materiales y Componentes electrónicos usados en el panel solar.**

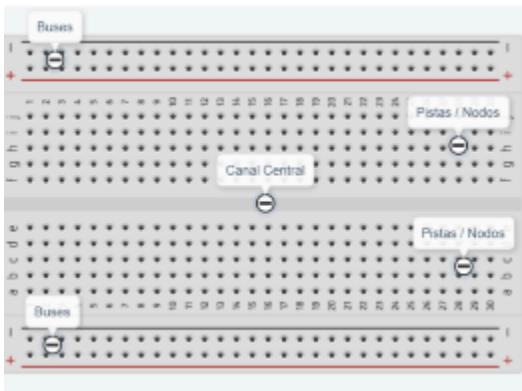
## 1. Arduino Uno R3.



[imagen 10]

Es un microcontrolador conocido por su facilidad de uso, además de ser una opción perfecta para la introducción a la programación de software y controlar hardware usado en proyectos electrónicos: Se basa en un chip ATmega328P, cuenta con 14 pines (entrada/salida), 6 entradas analógicas, un cristal de cuarzo de 16MHz, conexión USB para su programación y un jack de alimentación.

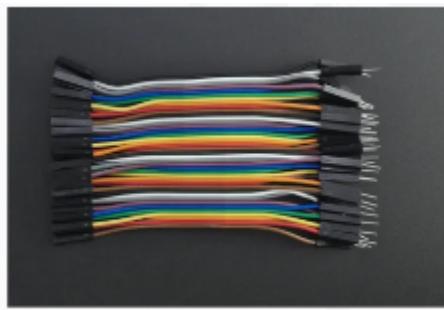
## 2. Protoboard.



[imagen 11]

Es una placa de prueba esencial para la creación de circuitos electrónicos además de ser muy útil para realizar pruebas a los circuitos antes de que estos sean presentados en una versión final está compuesta por pequeños orificios conectados entre sí mediante láminas metálicas.

## 3. Cables Jumpers.



[imagen 12]

Son cables flexibles usados en las conexiones desde un protoboard a otros dispositivos como por ejemplo a las placas Arduino, estos cables se dividen en tres tipos: **Macho a Macho** .- Cuentan con dos conectores conocidos como pines ideales para conexiones en un protoboard.

**Macho a Hembra** .-Cuenta con un extremo que integra un pin para hacer la conexión a un protoboard y el otro extremo integra un receptáculo para una conexión ajustable para un conector macho.

**Hembra a Hembra** .-Sus dos extremos están formados por un receptáculo ideal para crear una conexión más extensa ya que en sus extremos se puede conectar cables macho así la conexión entre un protoboard a otro dispositivo como puede ser una placa Arduino es más cómoda y eficiente.

## 4. Resistencias 10k.



[imagen 13]

Son usadas en muchos circuitos electrónicos para poder ajustar y controlar el flujo de corriente, además de que limita la cantidad de corriente protegiendo de daños en el circuito.

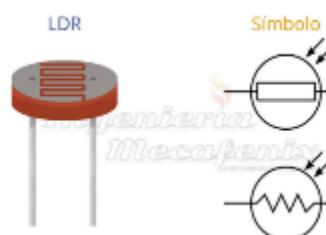
## 5. Servos de 360.



[imagen 14]

Son componentes muy útiles para los proyectos que requieren una rotación continua y control de su velocidad este puede girar indefinidamente en cualquier dirección estos se pueden controlar mediante señales PWM.

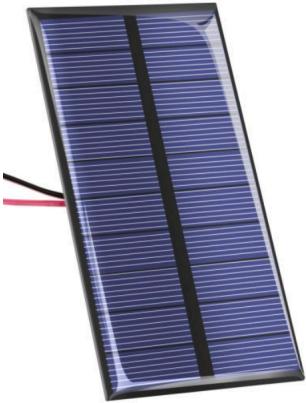
## 6. LDR.



[imagen 15]

También conocidos como fotoresistor el cual cambia su resistencia por la cantidad de luz que se establezca sobre él, es usado para detectar y captar la mayor intensidad de luz .Con aplicación en sistemas de iluminación automática y sistemas de seguimiento solar.

## 7. Panel.



[imagen 16]

Como ya es de conocimiento los paneles solares son utilizados en dispositivos que convierten la luz solar en energía eléctrica, son útiles en estos dispositivos para realizar el monitoreo y control en los sistemas solares obteniendo la mayor captación solar ya sea de manera manual o automática.

## 8. Estructura MDF.

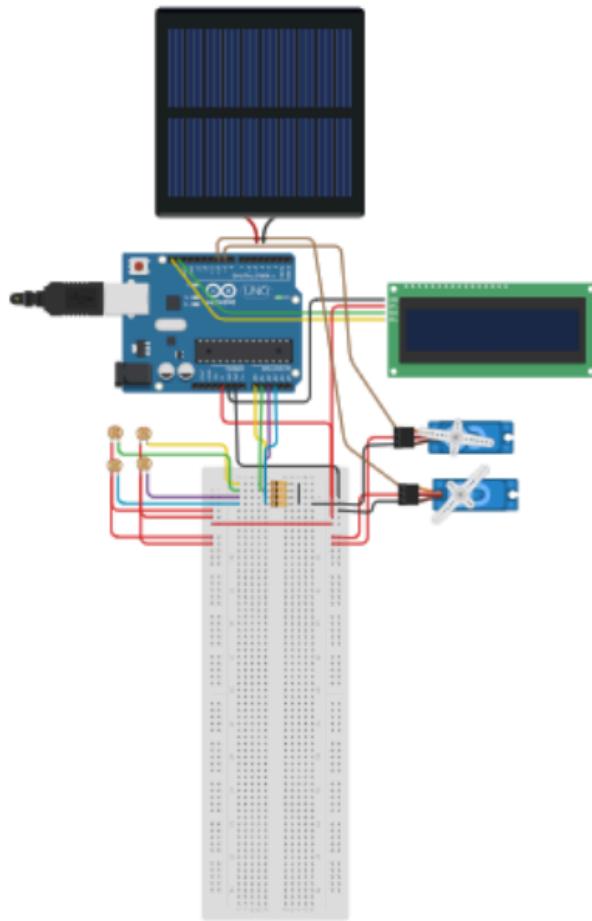


[imagen 17]

Está hecho de fibras de madera reciclada fácil de trabajar además de ser usado en construcción y diseño en proyectos de baja escala. Cuenta con algunas limitaciones ya que este no es resistente al agua.

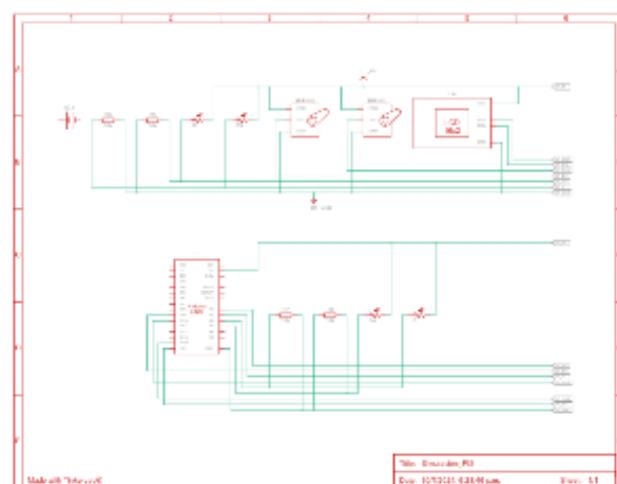
## 3.4 Diseño del circuito

El circuito realizado incluye la conexión entre los componentes necesarios para mover el panel solar en función de los servomotores e incluyendo el procesamiento de las señales de las fotorresistencias para optimizar la energía de forma inteligente.



### 3.4.1 Esquema del circuito

El siguiente esquema muestra la interconexión de los diferentes componentes electrónicos para definir su posición, dirección y estructura propiciando un circuito real acorde a lo planeado.



## 4. Enfoque en la sostenibilidad

## 4.1 Impacto Ambiental

La energía solar no genera ningún impacto ambiental ya que es amigable con el medio ambiente por lo que no genera contaminación al momento de crearse ni al momento de utilizarse como energía eléctrica, pero no hay que olvidar que los materiales con los que se ha construido en panel solar si generan contaminación al momento de ser fabricados y cuando son desechados más sin duda al tener un objetivo estructurado se convierte en una solución óptima para obtener energía.

## 4.2 Eficiencia energética

### 5. Mejoras futuras:

- **Integración con IoT y sistemas de energía inteligente:** Utilizar IoT para una integración más completa con otros dispositivos del hogar y para permitir la gestión inteligente de la energía generada y consumida.
- **Optimización del sistema de almacenamiento de energía:** Mejorar los algoritmos de gestión de la batería para optimizar la carga y descarga, considerando patrones de consumo y generación.
- **Automatización avanzada:** Implementar lógica de control avanzada para ajustar automáticamente la orientación de los paneles solares y optimizar el seguimiento del sol.
- **Monitoreo remoto y gestión de mantenimiento predictivo:** Utilizar análisis de datos avanzados para detectar anomalías en el rendimiento antes de que causen problemas mayores, permitiendo intervenciones de mantenimiento predictivo.

### 6. Conclusiones:

- El uso de funciones y procedimientos es importante para proporcionar un estilo más claro en la realización del programa.
- Para mantener una ubicación precisa del lugar en donde se encuentra el sol fue necesario la aplicación de distintas fórmulas para calcular la inclinación y orientación a la que se va a dirigir el panel solar.
- La aplicación de un código eficiente proporciona resultados adecuados y óptimos para su debida implementación en el panel solar.

### 7. Recomendaciones:

- Definir de una manera clara y precisa el propósito y que problema que resolverá el programa.
- Dividir el programa en pequeños módulos manejables y con una función específica.

- Tener en cuenta que las funciones a usar deben cumplir con tareas específicas las cuales ayudan a la resolución del problema.
- Realiza pruebas a cada función de manera individual para asegurar su funcionamiento.
- Mantener una documentación clara en tu código con pequeñas notas claves para que sea más fácil identificar algún error y su mantenimiento sea más fácil de realizar.

## 8. Bibliografía.

[1] Tutorialspoint. (s.f.). C Programming - Tutorialspoint. Recuperado de <https://www.tutorialspoint.com/cprogramming/index.htm>

[2] PV Education. (s.f.). Ángulo de Declinación Solar. [Imagen 1] Recuperado de <https://www.pveducation.org/es/fotovoltaica/2-propiedades-de-la-luz-del-sol/angulo-de-declinacion>

[3] Algoritmos. (s.f.). Tiempo Solar. Recuperado de [https://algoritmos9511.gitlab.io/\\_downloads/e1ac04d57c11925f0283040c533417bb/tiempo.pdf](https://algoritmos9511.gitlab.io/_downloads/e1ac04d57c11925f0283040c533417bb/tiempo.pdf)

[4] Universidad Santo Tomás, "Título del documento," en *Repositorio Institucional de la Universidad Santo Tomás*, año. [En línea]. Disponible en: <https://repository.usta.edu.co/handle/11634/16519>.

[5] M. Álvarez Álvarez, D. Ramírez Fonseca, y L. Pérez Caballero, "Fotovoltaic technology as an alternative energy source," \*Ingeniería Energética\*, vol. 36, no. 2, pp. 138-145, 2015. [En línea]. Disponible en: [http://scielo.sld.cu/scielo.php?pid=S1815-59012015000200008&script=sci\\_arttext&tlang=en](http://scielo.sld.cu/scielo.php?pid=S1815-59012015000200008&script=sci_arttext&tlang=en). [Consultado: 17 junio 2024].

[6] "¿Qué es y cómo funciona una LDR?," tecnosalva. [En línea]. Available: <https://www.tecnosalva.com/que-es-y-como-funciona-una-ldr/#:~:text=Una%20LDR%20o%20resistencia%20dependiente,que%20cae%20sobre%20su%20superficie>

[7] "factorenergia," [En línea]. Available: <https://www.factorenergia.com/es/blog/autoconsumo-electrico/que-es-la-resistencia-de-una-placa-solar/#:~:text=La%20resistencia%20de%20una%20placa%20solar%20es%20un%20factor%20clave,solar%20tendrá%20un%20mejor%20rendimiento>

[8] "Proyecto Arduino," [En línea]. Available: <https://proyectoarduino.com/arduino-uno-r3/>

[9] "¿Qué es el MDF? Ventajas y desventajas," majofesa. [En línea]. Available: <https://www.majofesa.com/mdf-o-tablero-dm-ventajas-y-desventajas/#:~:text=%C2%BFQu%C3%A9%20es%20el>

[%20MDF%3F.densidad%20que%20la%20madera%20contrachapada.](#)

[10] "Cargadores solares para dispositivos móviles," sabermas.umich.mx. [En línea]. Available: <https://www.sabermas.umich.mx/archivo/tecnologia/249-numero-28/443-cargadores-solares-para-dispositivos-moviles.shtml#:~:text=Los%20cargadores%20solares%20obtiene,n%20energ%C3%A9tica,poderla%20consumir%20cuando%20se%20requiere>

[11] "Simulación PIS," Tinkercad. [En línea]. Available: <https://www.tinkercad.com/things/7Oh93C7u2Le-simulacionpis?sharecode=6Im-tmjrubbVxKwk78lyMyZZdjbxHSvsn5qXCjKaQmk>.

Imagen 1:  
<https://www.solarweb.net/forosolar/formacion-energia-solar/3229-declinacion-solar.html>

Imagen 2: <https://susdesign.com/popups/sunangle/eot.php>

Imagen 3:  
<https://www.e-education.psu.edu/eme810/node/531>

Imagen 4:  
<https://fcaglp.fcaglp.unlp.edu.ar/~egiorgi/ag/Apuntes/tiempos.pdf>

Imagen 5:  
<https://solarsena.com/solar-hour-angle-calculator-formula/>

Imagen 6:  
<https://www.pveducation.org/es/fotovoltaica/2-propiedades-de-la-luz-del-sol/posicion%C3%B3n-del-sol>

Imagen 7:  
<https://www.pveducation.org/es/fotovoltaica/2-propiedades-de-la-luz-del-sol/%C3%A1ngulo-acim>

Imagen 8:  
<https://rodin.uca.es/bitstream/handle/10498/9873/teap.pdf?sequence=1&isAllowed=y>

Imagen 9:  
[https://d1wqxts1xzle7.cloudfront.net/62068123/calcu-vectorial-tromba20200211-29642-17w01fq.pdf?1581459347=&response-content-disposition=inline%3B+filename%3DCA\\_LCULO\\_VECTORIAL\\_Tromba.pdf&Expires=1721665998&Signature=PKbMwOdBVpgxNVqNpS1~PjUsJ1p6wHo-zLFtqEt2T9-oO2y5d2aoqy9WEuLYZLSvsVv7qw4Fxu3eumKIGvX-koCSYAOFvUWOB-iqOm2CZGjJn68-FazFZEc](https://d1wqxts1xzle7.cloudfront.net/62068123/calcu-vectorial-tromba20200211-29642-17w01fq.pdf?1581459347=&response-content-disposition=inline%3B+filename%3DCA_LCULO_VECTORIAL_Tromba.pdf&Expires=1721665998&Signature=PKbMwOdBVpgxNVqNpS1~PjUsJ1p6wHo-zLFtqEt2T9-oO2y5d2aoqy9WEuLYZLSvsVv7qw4Fxu3eumKIGvX-koCSYAOFvUWOB-iqOm2CZGjJn68-FazFZEc)

[Z2a6MA~9Bu~wuqAeMExQVrBcNKbNE9PfHSCBH83xEb2bPF~1ZdYnHng3RoXYNtBGeTAFNJRkrBfqR3RvG4Dcofdc2~qRHpTzUMpVMkDdpZ8a7zWEiWxplu4SfcHFOEMJ074-JX9~PrqrWalZf2rB11-bT9ei4NNYoVngzoatWrbh-EzcewDQwFKdmc3c-zh6On5OA9X~sb0v7TnAkGAzLJqZ7zUfov\\_g\\_&Key-Pair-Id=APKAJLOHF5GGSLRBV4ZA](#)

Imagen 10:  
<https://tecmikro.com/content/17-arduino-uno-r3-caracteristicas>

Imagen 11: [PlayKodo - https://www.playkodo.com/protoboard-circuitos-electricos/](https://www.playkodo.com/protoboard-circuitos-electricos/)

Imagen 12:  
<https://www.vistronica.com/conectores-cables-y-switches/cable-dupont-macho-hembra-x40-10-cm-detail.html>

Imagen 13: <https://arm32.cl/producto/resistencias/>

Imagen 14:  
<https://novatronicec.com/index.php/product/servomotor-s3003-12kg-360-grados/>

Imagen 15:  
<https://www.ingmecafenix.com/automatizacion/fotoresistencia/>

Imagen 16:  
<https://www.steren.com.ec/panel-solar-de-5-vcc-y-160-mah.html>

Imagen 17:  
<https://es.quora.com/Qu%C3%A9-es-la-madera-MDF>

## 9. Anexos

### Anexo 1

DIAGRAMA DE FLUJO GENERAL DEL ALGORITMO:

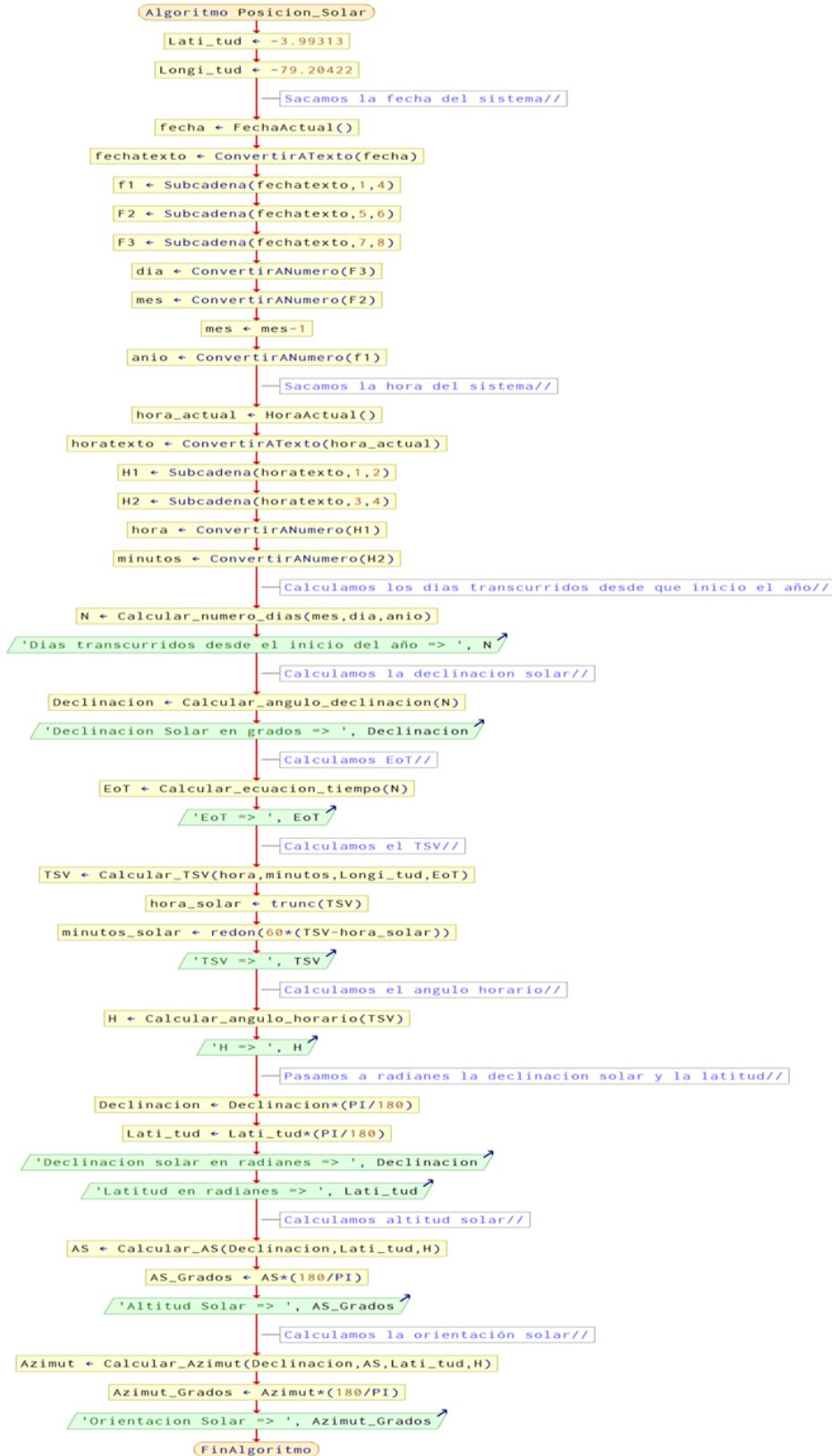


DIAGRAMA DE FLUJO DE LA FUNCIÓN PARA CALCULAR LOS DÍAS TRANSCURRIDOS DESDE EL INICIO DEL AÑO HASTA EL DIA ACTUAL:

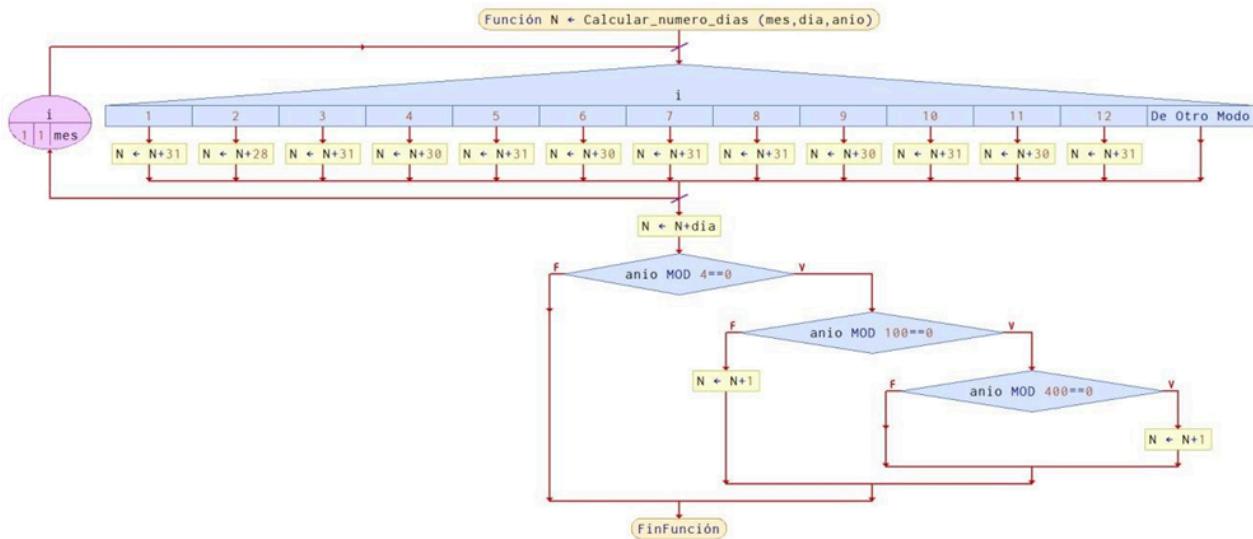


DIAGRAMA DE FLUJO DE LA FUNCIÓN PARA CALCULAR EL ÁNGULO DE DECLINACIÓN:

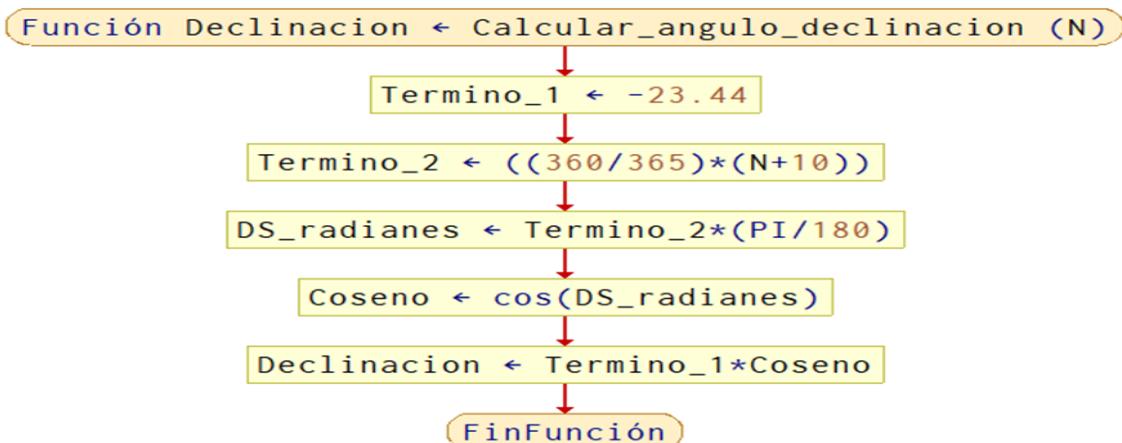


DIAGRAMA DE FLUJO DE LA FUNCIÓN DE LA ECUACIÓN DEL TIEMPO:

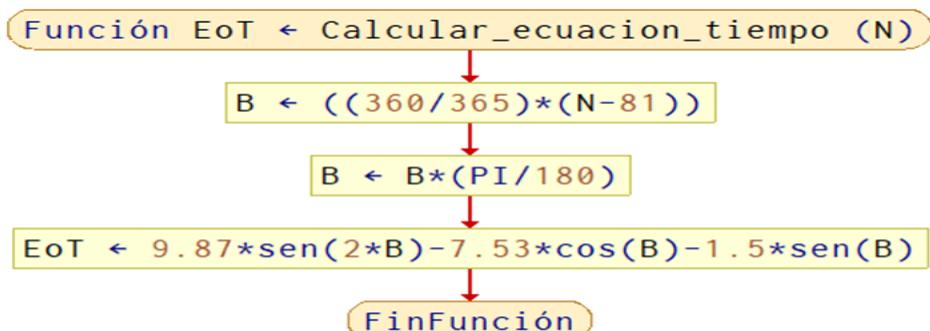


DIAGRAMA DE FLUJO DE LA FUNCIÓN PARA CALCULAR EL TSV:

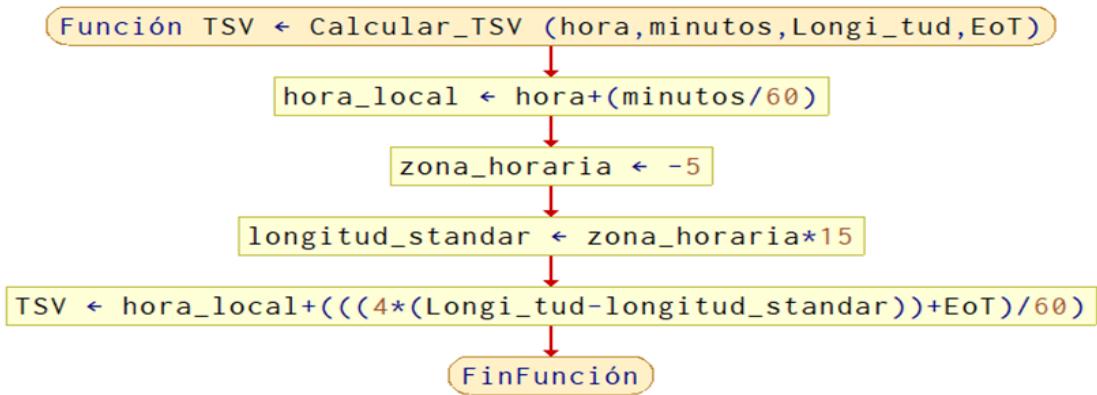


DIAGRAMA DE FLUJO DE LA FUNCIÓN PARA CALCULAR EL ÁNGULO HORARIO:

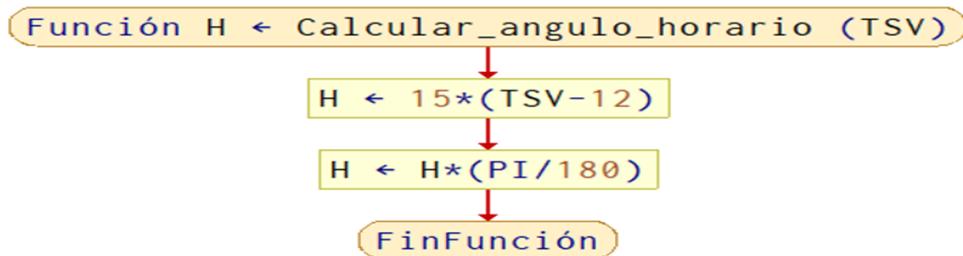


DIAGRAMA DE FLUJO DE LA FUNCIÓN PARA CALCULAR LA ALTITUD SOLAR:

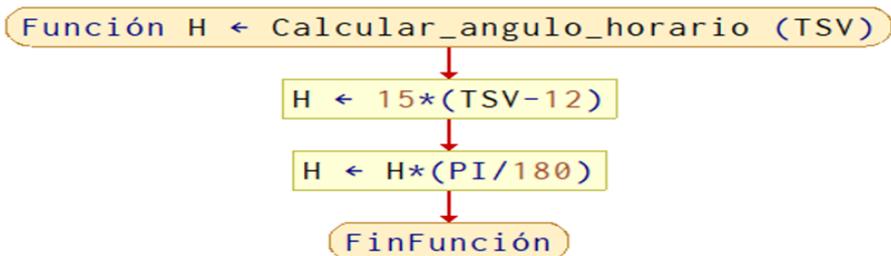
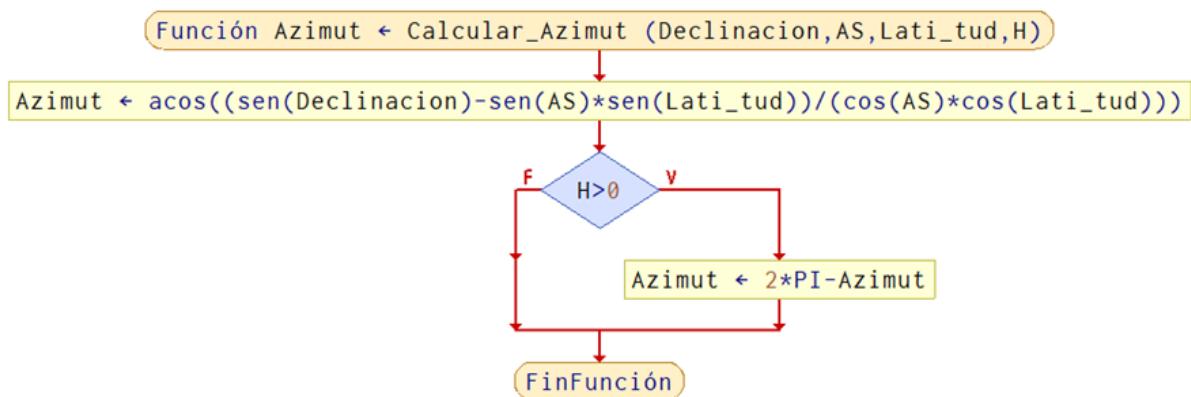


DIAGRAMA DE FLUJO DE LA FUNCIÓN PARA CALCULAR LA ORIENTACIÓN SOLAR:



## Anexo 2

Pseudocódigo principal:

```
Algoritmo Posicion_Solar
    Lati_tud ← -3.99313
    Longi_tud ← -79.20422
    // Sacamos la fecha del sistema//
    fecha ← FechaActual()
    fechatexto ← ConvertirATexto(fecha)
    f1 ← Subcadena(fechatexto,1,4)
    F2 ← Subcadena(fechatexto,5,6)
    F3 ← Subcadena(fechatexto,7,8)
    dia ← ConvertirANumero(F3)
    mes ← ConvertirANumero(F2)
    mes ← mes-1
    anio ← ConvertirANumero(f1)
    // Sacamos la hora del sistema//
    hora_actual ← HoraActual()
    horatexto ← ConvertirATexto(hora_actual)
    H1 ← Subcadena(horatexto,1,2)
    H2 ← Subcadena(horatexto,3,4)
    hora ← ConvertirANumero(H1)
    minutos ← ConvertirANumero(H2)
    // Calculamos los días transcurridos desde que inicio el año//
    N ← Calcular_numero_dias(mes,dia,anio)
    Escribir 'Días transcurridos desde el inicio del año => ', N
    // Calculamos la declinación solar//
    Declinacion ← Calcular_angulo_declinacion(N)
    Escribir 'Declinación Solar en grados => ', Declinacion
    // Calculamos EoT//
    EoT ← Calcular_ecuacion_tiempo(N)
    Escribir 'EoT => ', EoT
    // Calculamos el TSV//
    TSV ← Calcular_TSV(hora,minutos,Longi_tud,EoT)
    hora_solar ← trunc(TSV)
    minutos_solar ← redon(60*(TSV-hora_solar))
    Escribir 'TSV => ', TSV
    // Calculamos el ángulo horario//
    H ← Calcular_angulo_horario(TSV)
    Escribir 'H => ', H
    // Pasamos a radianes la declinación solar y la latitud//
    Declinacion ← Declinacion*(PI/180)
    Lati_tud ← Lati_tud*(PI/180)
    Escribir 'Declinación solar en radianes => ', Declinacion
    Escribir 'Latitud en radianes => ', Lati_tud
    // Calculamos altitud solar//
    AS ← Calcular_AS(Declinacion,Lati_tud,H)
    AS_Grados ← AS*(180/PI)
    Escribir 'Altitud Solar => ', AS_Grados
    // Calculamos la orientación solar//
    Azimut ← Calcular_Azimut(Declinacion,AS,Lati_tud,H)
    Azimut_Grados ← Azimut*(180/PI)
    Escribir 'Orientación Solar => ', Azimut_Grados
FinAlgoritmo
```

### Anexo 3

```
Funcion N ← Calcular_numero_dias (mes, dia, anio)
    Para i<1 Hasta mes Con Paso 1 Hacer
        Segun i Hacer
            1:
                N = N + 31
            2:
                N = N + 28
            3:
                N = N + 31
            4:
                N = N + 30
            5:
                N = N + 31
            6:
                N = N + 30
            7:
                N = N + 31
            8:
                N = N + 31
            9:
                N = N + 30
            10:
                N = N + 31
            11:
                N = N + 30
            12:
                N = N + 31
        Fin Segun
    Fin Para
    N = N + dia
    si anio % 4 == 0 Entonces
        si anio % 100 == 0 Entonces
            si anio % 400 == 0 Entonces
                N = N + 1
            Fin si
        SiNo
            N = N + 1
        FinSi
    FinSi
Fin Funcion
```

Anexo 4

```
Funcion Declinacion ← Calcular_angulo_declinacion ( N )
    Termino_1 = -23.44
    Termino_2 = ((360/365) * (N + 10))
    DS_radianes = Termino_2 * (PI/180)
    Coseno = cos(DS_radianes)
    Declinacion = Termino_1 * Coseno
Fin Funcion
```

Anexo 5

```
Funcion EoT ← Calcular_ecuacion_tiempo ( N )
    B = ((360/365) * (N - 81))
    B = B * (PI/180)
    EoT = 9.87 * sen(2 * B) - 7.53 * cos(B) - 1.5 * sen(B)
Fin Funcion
```

Anexo 6

```
Funcion TSV ← Calcular_TSV (hora, minutos, Longi_tud, EoT)
    hora_local = hora + (minutos/60)
    zona_horaria = -5
    longitud_standar = zona_horaria * 15
    TSV = hora_local + (((4*(Longi_tud - longitud_standar)) + EoT)/60)
Fin Funcion
```

Anexo 7

```
Funcion H ← Calcular_angulo_horario (TSV)
    H = 15 * (TSV - 12)
    H = H * (PI/180)
Fin Funcion
```

Anexo 8

```
Funcion Azimut ← Calcular_Azimut (Declinacion, AS, Lati_tud, H)
    Azimut = acos((sen(Declinacion) - sen(AS) * sen(Lati_tud)) / (cos(AS) * cos(Lati_tud)))
    si H > 0 Entonces
        |   Azimut =2 * PI - Azimut
    FinSi
Fin Funcion
```