# Automatización de Paneles Solares

## I INTRODUCCIÓN

El uso de paneles solares se ha convertido en el principal medio para aprovechar la energía solar, y la automatización de los paneles es una herramienta esencial para maximizar dicha eficiencia. Aquí se explica un modelo integral de ensayo que utiliza diferentes estudios y proyectos y ofrece un enfoque unificado, combinando ideas de algoritmos, simulaciones y otros componentes relevantes para formar un sistema de automatización de paneles solares.

Con la demanda creciente de fuentes de energía renovable, la automatización de la orientación de los paneles solares es crítica para mejorar la eficiencia en el aprovechamiento de energía y es esencial para el desarrollo sostenible en cuanto a la generación de energía.

## II OBJETIVOS

**General:** Diseñar e implementar un sistema de automatización de paneles solares que maximice el aprovechamiento de la energía solar mediante la orientación automática.

**Específicos:**

- Desarrollar un algoritmo con base a luz solar en tiempo real para la orientación de los paneles solares.
- Implementar el algoritmo utilizando microcontroladores y sensores adecuados.
- Revisar la eficiencia del sistema de automatización comparándolo con sistemas estáticos.
- Realizar una simulación final previa a la implementación física en todo su conjunto.
- Determinar la rentabilidad del sistema antes de la implementación.

## III METODOLOGÍA

Desarrollo del Algoritmo: Basado en cálculos astronómicos para determinar la posición del sol en cualquier parte y lugar del día y geo-posición.

## IV ECUACIONES UTILIZADAS

- Azimut: el ángulo horizontal del norte.
- Altura Solar: el ángulo vertical del horizonte.
- Declinación: el ángulo de los rayos solares con el plano del ecuador terrestre.
- Ángulo Horario: tiempo solar en horas angulares.

## V ANÁLISIS DETALLADO DE LAS VARIABLES INVOLUCRADAS

El código proporcionado realiza cálculos relacionados con la posición del sol en función de la fecha, hora, latitud y longitud proporcionadas por el usuario. Aquí está el análisis detallado de las variables utilizadas:

### Variables Globales y Constantes

PI: Una constante definida para representar el valor de Pi con precisión extendida.

### Variables Locales en `main()`

a, m, d1, cont, d, db: Variables enteras utilizadas para almacenar el año actual, mes actual, día actual, un contador para iteraciones, días transcurridos desde el inicio del año y un indicador de año bisiesto respectivamente.

hour, min, hor_loc: Variables de punto flotante utilizadas para representar la hora actual, minutos actuales y la hora local en formato decimal (hora + minutos/60).

Long, lat, ds, b, eot, long_zon, long_est, tsv, h, alt_sol, alt_sol_gra, azim: Variables de punto flotante utilizadas para almacenar la longitud, latitud, declinación solar, ecuación del tiempo, longitud zonal, longitud estándar, tiempo solar verdadero, ángulo de altura solar, altura solar en grados, y azimut (dirección solar) en grados respectivamente.

Detalle del Código y Uso de Variables

- Fecha y Hora Actuales:

- `time_t now; struct tm *local;`: Variables relacionadas con la obtención de la hora actual del sistema.

- `localtime(&now);`: Convierte el tiempo en segundos (`now`) en una estructura `tm` local (`local`) que contiene los detalles de fecha y hora.

- Cálculo de Días Transcurridos:

- Utiliza un bucle para contar los días transcurridos desde el inicio del año hasta la fecha actual, considerando si el año es bisiesto para ajustar el número de días en febrero.

- Cálculo de la Declinación Solar:

- Utiliza la fórmula astronómica para calcular la declinación solar basada en el día del año (`d`).

- Cálculo de la Ecuación del Tiempo:

- Determina la variación entre la hora solar verdadera y la hora solar estándar debido a la órbita elíptica de la Tierra alrededor del Sol.

- Cálculo del Tiempo Solar Verdadero:

- Ajusta la hora local (`hor_loc`) para calcular el tiempo solar verdadero teniendo en cuenta la longitud estándar y la ecuación del tiempo.

- Cálculo de la Altura Solar y Azimut:

- Utiliza fórmulas trigonométricas para determinar la altura del sol sobre el horizonte y su dirección (azimut) en función de la latitud, longitud y declinación solar.

Entrada y Salida

-Entrada de Latitud y Longitud: El usuario ingresa la latitud y longitud para calcular la posición solar.

-Salida: Se imprimen en consola los valores calculados de la declinación solar, altura solar y azimut.

En conclusión, el código proporciona un ejemplo de cómo calcular la posición solar en función de la ubicación geográfica y el tiempo actual. Utiliza fórmulas astronómicas estándar para determinar la posición del sol, lo cual es útil en aplicaciones de astronomía, navegación y energía solar.

## VI EXPLICACIÓN DE LAS ESTRUCTURAS DE DATOS Y FUNCIONES UTILIZADAS EN EL CÓDIGO

El código proporcionado utiliza principalmente estructuras de datos simples y algunas funciones estándar de la biblioteca estándar de C para realizar cálculos relacionados con la posición solar. Aquí está la explicación detallada de las estructuras de datos y funciones utilizadas:

Estructuras de Datos

- Estructura `tm` (de `<time.h>`):

Esta estructura se utiliza para almacenar la fecha y hora descompuesta en sus componentes individuales como año, mes, día, hora, etc. Es útil para trabajar con fechas y horas de manera fácil y organizada.

En el código proporcionado, se utiliza principalmente para almacenar la fecha y hora actuales obtenidas del sistema mediante la función `localtime`.

- Tipo `time_t` (de `<time.h>`):

Es un tipo de dato que se utiliza para representar tiempos y fechas en forma de segundos desde la época (1 de enero de 1970 en la mayoría de los sistemas). Se utiliza para almacenar el tiempo actual del sistema antes de convertirlo en una estructura `tm` utilizando `localtime`.

Funciones Utilizadas

- `time()` (de `<time.h>`):

Esta función devuelve el tiempo actual en segundos desde la época. Se utiliza para inicializar la variable `now` de tipo `time_t`.

- `localtime()` (de `<time.h>`):

Esta función convierte el tiempo representado por `time_t` en una estructura `tm` local que contiene la fecha y hora descompuesta en componentes individuales (como año, mes, día, etc.).

- `printf()` y `scanf()` (de `<stdio.h>`):

Estas funciones se utilizan para la entrada y salida estándar en la consola.

- `printf()` se utiliza para imprimir mensajes y resultados en la consola.

- `scanf()` se utiliza para recibir entrada del usuario, en este caso, para obtener la latitud y longitud.

- Funciones matemáticas como `cos()`, `sin()`, `acos()`, `asin()`, `trunc()`** (de `<math.h>`):

Se utilizan para realizar cálculos matemáticos necesarios para determinar la posición del sol.

- Ejemplos de uso en el código:

```
ds = -23.44 * cos((((360.0 / 365) * (d + 10)) * pi) / 180);

alt_sol = asin(sin(ds) * sin(lat) + cos(ds) * cos(lat) * cos(h));

azim = acos((sin(ds) - sin(alt_sol) * sin(lat)) / (cos(alt_sol) * cos(lat)));
```

En conclusión, el código combina el uso de estructuras de datos (`struct tm` y `time_t`) para manejar fechas y horas, con funciones matemáticas (`cos()`, `sin()`, etc.) para calcular posiciones astronómicas como la declinación solar, la altura solar y el azimut. La entrada y salida se manejan con `printf()` y `scanf()`, asegurando una interacción adecuada con el usuario para ingresar la ubicación geográfica necesaria. En conjunto, estas estructuras de datos y funciones permiten al código calcular y mostrar la posición del sol con precisión astronómica utilizando datos de tiempo y ubicación del usuario.

VII PSEUDOCÓDIGO Y DIAGRAMAS DE FLUJO DEL ALGORITMO ORIGINAL COMPARADOS CON LA IMPLEMENTACIÓN EN C

```
Algoritmo azim_sol
    //determinio de la fecha actual
    a = ConvertirANumero(Subcadena(ConvertirATexto(FechaActual()), 1, 4))
    m = ConvertirANumero(Subcadena(ConvertirATexto(FechaActual()), 5, 6))
    dl = ConvertirANumero(Subcadena(ConvertirATexto(FechaActual()), 7, 8))
    //determinio de la hora local
    hour = trunc(HoraActual()/10000)
    min = trunc(((HoraActual()/10000) - hour) * 100)
    hor_loc = hour + (min / 60)
    //conteo de los dias transcurridos desde que inicio el año
    cont = 0
    d = 0
    Repetir
        si cont == 1 o cont == 5 o cont == 7 o cont == 8 o cont == 10 o cont == 12 Entonces
            d = d + 31
        FinSi
        si cont == 2 Entonces
            d = d + 28
        FinSi
        si cont == 4 o cont == 6 o cont == 9 o cont == 11 Entonces
            d = d + 30
        FinSi
        cont = cont + 1
    Hasta Que cont > m
    si a % 4 == 0 y a % 100 ≠ 0 Entonces
        db = 1
    sino
        si a % 100 == 0 y a % 400 == 0 Entonces
            db = 1
        SiNo
            db = 0
        FinSi
    FinSi
    d = d + dl + db
    //recibir datos de latitud y longitud
    Escribir "----------------------------------------------"
    Escribir "escriba la longitud"
    Leer Long
    Escribir "----------------------------------------------"
    Escribir "escriba la latitud"
    Leer lat
    //calculo de la declinacion solar
    ds = -23.44 * cos((((360/365) * (d + 10)) * pi) / 180)
    Escribir "----------------------------------------------"
    Escribir "el angulo de declinacion solar es de ", ds, "°"
    //formula de la ecuacion del tiempo
    b = (((360/365) * (d - 81)) * pi) / 180
    eot = 9.87 * sen(2 * b) - 7.53 * cos(b) - 1.5 * sen(b)
    //calculo del tiempo solar verdadero
    long_zon = trunc(trunc(Long) / 15)
    long_est = long_zon * 15
    tsv = hor_loc + (((4 * (Long - long_est)) + eot) / 60)
    //calculo de la altura solar
    h = (15 * (tsv - 12)) * (pi / 180)
    ds = ds * (pi / 180)
    lat = lat * (pi / 180)
    alt_sol = asen(sen(ds) * sen(lat) + cos(ds) * cos(lat) * cos(h))
    alt_sol_gra = alt_sol * (180 / pi)
    Escribir "----------------------------------------------"
    Escribir "la altura solar es de ", alt_sol_gra, "°"
    //calculo del azimut
    Escribir "----------------------------------------------"
    azim = (acos((sen(ds) - sen(alt_sol) * sen(lat)) / (cos(alt_sol) * cos(lat)))) * (180 / pi)
    Escribir "el azimut o direccion solar es de ", azim, "°"
FinAlgoritmo
```

Figura 1. Código desarrollado por los estudiantes de primer ciclo

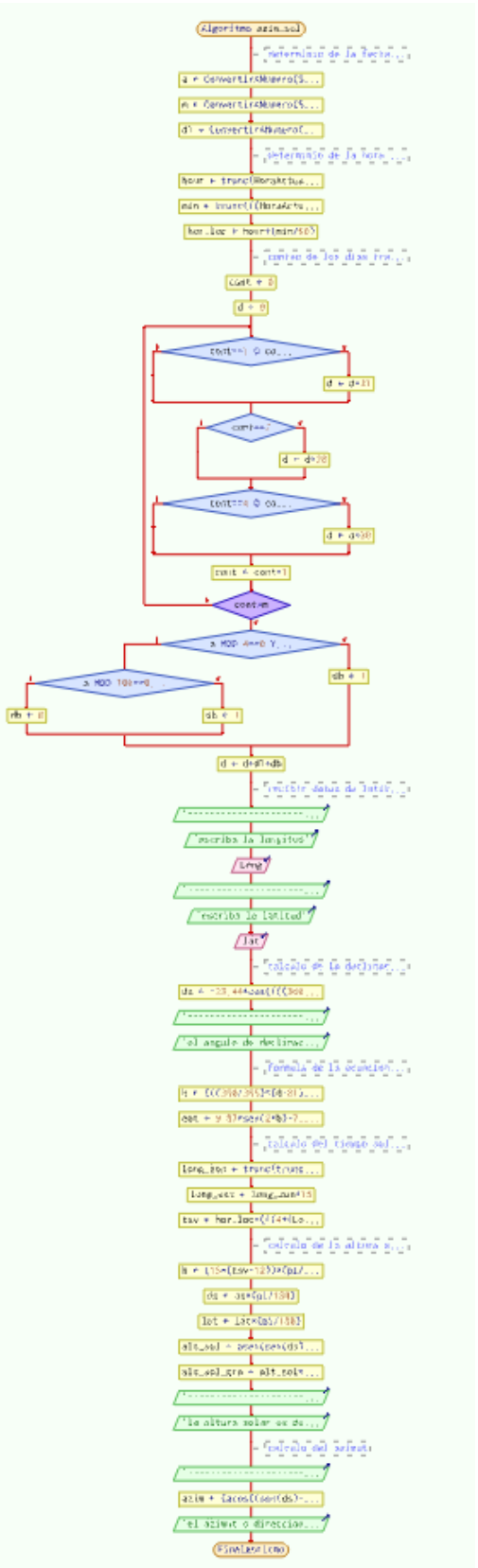

Figura 2. Diagrama de flujo del algoritmo desarrollado por los estudiantes de primer ciclo

```c
#include <stdio.h>
#include <math.h>
// Libreria para las operaciones matematicas
#include <time.h>
// Libreria para las funciones de fecha y hora

#define pi 3.14159265358979323846 // Definimos pi como un valor aproximado a "Pi"

int main() {
    int a, m, d1, cont = 0, d, db = 0;
// Declaramos las variables enteras de: año, mes y dia actuales, asi como un contador y el numero de dias desde el 1 de enero
    float hour, min, hor_loc; // Declaramos las variables de la hora, minutos y hora local
    /*Declaramos las variables de longitud, latitud, declinacion solar, la ecuacion del
    tiempo, el tiempo solar verdadero, la altura solar y el azimuto direccion solar*/
    float Long, lat, ds, b, eot, long_zon, long_est, tsv, h, alt_sol, alt_sol_gra, azim;

    // Obtener la fecha y hora actuales

    time_t now; // Variable para almacenar el tiempo en segundos
    struct tm *local; // Estructura que
    time(&now);
    local = localtime(&now);

    // Determinación de la fecha actual
    a = local->tm_year + 1900;
// tm_year devuelve años desde 1900
    m = local->tm_mon + 1;
// tm_mon devuelve meses desde 0
    d1 = local->tm_mday;
// tm_mday devuelve el día del mes

    // Determinación de la hora local
    hour = local->tm_hour;
    min = local->tm_min;
    hor_loc = hour + (min / 60);

    // Conteo de los días transcurridos desde que inicio el año
    d = 0;
    cont = 0;
    do {
        if (cont == 1 || cont == 3 || cont == 5 || cont == 7 || cont == 8 || cont == 10 || cont == 12) {
            d = d + 31;
        } else if (cont == 4 || cont == 6 || cont == 9 || cont == 11) {
            d = d + 30;
        } else if (cont == 2) {
            if ((a % 4 == 0 && a % 100 != 0) || (a % 400 == 0)) {
                d = d + 29; // Año bisiesto
            } else {
                d = d + 28; // Año no bisiesto
            }
        }
        cont++;
    } while (cont < m);

    // Ajuste para el día actual
    d = d - (31 - d1);


// Determinación si el año actual es bisiesto
    if ((a % 4 == 0 && a % 100 != 0) || (a % 400 == 0)) {
        db = 1;
    } else {
        db = 0;
    }
    d = d + d1 + db;

    // Recibir datos de latitud y longitud
    printf("------------------------------\n");
    printf("Escriba la latitud: ");
    scanf("%f", &lat);
    printf("------------------------------\n");
    printf("Escriba la longitud: ");
    scanf("%f", &Long);

    // Cálculo de la declinación solar
    ds = -23.44 * cos((((360.0 / 365) * (d + 10)) * pi) / 180);
    printf("------------------------------\n");
    printf("El angulo de declinacion solar es de %.2f grados\n", ds);

    // Fórmula de la ecuación del tiempo
    b = (((360.0 / 365) * (d - 81)) * pi) / 180;
    eot = 9.87 * sin(2 * b) - 7.53 * cos(b) - 1.5 * sin(b);

    // Cálculo del tiempo solar verdadero
    long_zon = trunc(Long / 15);
    long_est = long_zon * 15;
    tsv = hor_loc + (((4 * (Long - long_est)) + eot) / 60);

    // Cálculo de la altura solar
    h = (15 * (tsv - 12)) * (pi / 180);
    ds = ds * (pi / 180);
    lat = lat * (pi / 180);
    alt_sol = asin(sin(ds) * sin(lat) + cos(ds) * cos(lat) * cos(h));
    alt_sol_gra = alt_sol * (180 / pi);
    printf("------------------------------\n");
    printf("La altura solar es de %.2f grados\n", alt_sol_gra);

    // Cálculo del azimut
    printf("------------------------------\n");
    azim = acos((sin(ds) - sin(alt_sol) * sin(lat)) / (cos(alt_sol) * cos(lat)));
    if (h > 0) {
        azim = 2 * pi - azim;
    }
    azim = azim * (180 / pi);
    printf("El azimut o direccion solar es de %.2f grados\n", azim);

    return 0;
}
```

Figura 3. Código en C desarrollado por los estudiantes de primer ciclo

## VIII IMPLEMENTACIÓN EN UN SISTEMA DE PANELES SOLARES

Implementar este código en un sistema real de paneles solares implica integrarlo en un sistema más amplio que controle y optimice el funcionamiento de los paneles solares basándose en la posición del sol.

Los sistemas reales de paneles solares generalmente incluyen sensores para medir la radiación solar, la temperatura, y a veces la posición del sol directamente. Estos sensores pueden ser integrados con el código para obtener datos más precisos y en tiempo real sobre la condición solar.

El código actual calcula la posición del sol, pero en un sistema real sería necesario utilizar estos datos para ajustar automáticamente la orientación de los paneles solares para maximizar la captación de energía solar durante todo el día. Esto implica el uso de actuadores para mover los paneles según la posición calculada del sol.

Desarrollar una interfaz de usuario o un sistema de monitoreo que muestre los datos calculados en tiempo real, así como el rendimiento de los paneles solares en función de la posición del sol.

Asegurarse de que los cálculos astronómicos sean precisos y estén actualizados. Pueden ser necesarias actualizaciones periódicas o ajustes para considerar variaciones estacionales o cambios en los patrones climáticos que podrían afectar la posición del sol.

## IX POSIBLES MEJORAS FUTURAS

Utilizar técnicas de machine learning para optimizar automáticamente la orientación de los paneles solares basándose en datos históricos de radiación solar y rendimiento de los paneles.

Utilizar sensores de alta precisión y confiabilidad para obtener mediciones más precisas de la radiación solar y otros parámetros ambientales que afectan el rendimiento de los paneles solares.

Permitir al sistema adaptarse automáticamente a diferentes ubicaciones geográficas ingresando las coordenadas geográficas, o incluso utilizando GPS para determinar la posición exacta y ajustar los cálculos en consecuencia.

Utilizar los datos recopilados para desarrollar estrategias de mantenimiento predictivo que puedan anticipar y prevenir fallos en los paneles solares o en el sistema de seguimiento solar.

## X ELEMENTOS UTILIZADOS EN EL DESARROLLO:

- Panel Solar: 112 V voltaje, 5 W potencia máxima
- Servomotor SG5010: 65 kg-cm torque
- Microcontrolador Arduino: Programado en lenguajes C y Python
- Sensor Fotorresistor: Nivel de luz
- Sensor ACS712: Corriente eléctrica
- Resistencias de 10K (0.25 W)
- Placa de Pruebas: Integración de elementos
- Diodos Protectores: Protección contra la corriente inversa.

## XI SIMULADORES

SimulIDE y Proteus VSM, otros como Arduino Cloud, CircuitLab, Eagle, EasyEDA, Emulare, Fritzing, LTSpice.

## XII IMPLEMENTACIÓN PRÁCTICA

El proceso se da desde la selección de los componentes hasta la programación del sistema, proceso que abarca las siguientes fases claves:

- Objetivo: definir un propósito claro
- Componentes: elegir el material adecuado
- Diseño Mecánico: planificar la presentación del sistema
- Creación de un circuito eléctrico: armar el sistema
- Automatización: programar de acuerdo al algoritmo
- Pruebas: verificar que cumpla con la carga e idea
- Seguridad: proteger, aislar condiciones climatológicas.

## XIII RESULTADOS Y DISCUSIÓN

Resultados durante la simulación, se presentan los datos experimentales y los hallazgos se mostraron una diferencia significativa entre el caso de orientación óptima en función al sistema automatizado en comparación con los sistemas estáticos.

## XIV TASA DE CARGA

La diferencia ajena al uso propiamente de paneles solares maximiza el ángulo al sol para maximizar el aprovechamiento de la energía. El sensor y el microcontrolador son exitosos en cuanto a la implementación fuera del simulador.

## XV CONCLUSIÓN

El uso de un algoritmo competente y microcontroladores aporta mejoras inimaginables a los sistemas de aprovechamiento de energía verde. Los sistemas posibles de crear basados en electrónica y simulaciones reales y competentes pueden ser factibles en la realidad. La idea general del proyecto es que la optimización de los sistemas de energía a través de la automatización puede marcar una verdadera diferencia en la transición.

# Referencias


[1] IBM, «¿Qué es el machine learning (ML)?,» [En línea]. Available: https://www.ibm.com/es-es/topics/machine-learning.

[2] PVEducation, «Ángulo acimut,» [En línea]. Available: https://www.pveducation.org/es/fotovoltaica/2-propiedades-de-la-luz-del-sol/%C3%A1ngulo-acimut.

[3] PVEducation, «Ángulo de elevación,» [En línea]. Available: https://www.pveducation.org/es/fotovoltaica/2-propiedades-de-la-luz-del-sol/el-%C3%A1ngulo-de-elevaci%C3%B3n.