Trabajo de Fin de Grado
Grado en Ingeniería Electrónica, Robótica y Mecatrónica

Sistema de ayuda al posicionamiento de placas solares basado en microcontrolador

Autor: Alejandro Letrado Castellanos
Tutor: Manuel Ángel Perales Esteve

Dpto. de Ingeniería Electrónica
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2021

Trabajo de Fin de Grado
Grado en Ingeniería Electrónica, Robótica y Mecatrónica

Sistema de ayuda al posicionamiento de placas solares basado en microcontrolador

Autor:
Alejandro Letrado Castellanos

Tutor:
Manuel Ángel Perales Esteve
Profesor titular

Dpto. de Ingeniería Electrónica
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2021

Trabajo de Fin de Grado: Sistema de ayuda al posicionamiento de placas solares basado en microcontrolador

Autor: Alejandro Letrado Castellanos

Tutor: Manuel Ángel Perales Esteve

El tribunal nombrado para juzgar el Trabajo arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocal/es:

Secretario:

acuerdan otorgarle la calificación de:

El Secretario del Tribunal:

Fecha:

A mi familia

A mis maestros

Agradecimientos

Primero, por supuesto, agradecer a mi tutor, Manuel por toda la atención, ayuda y guía para traer aquí este trabajo. Y también a esos profesores por los que tanto he avanzado, Graci, Jesús, Mariángeles, Gil, …

Agradecer a todos mis amigos, los del pueblo y todos los acumulados en estos 4 años de grado. Especialmente a Marta y Alberto por su apoyo en la redacción de este trabajo; a Roberto y Elena por el aporte de material para la elaboración de las pruebas y a Thais que siempre me ha apoyado y a Carmen por sus ánimos mientras se desarrollaba este trabajo.

Por último, gracias a mis padres, mi hermana y toda mi familia sin los que no habría llegado hasta donde estoy.

Alejandro Letrado Castellanos

Argamasilla de Calatrava, 2021

Resumen

En el presente trabajo se realiza el estudio previo y desarrollo de un sistema simplificado que permita obtener la posición del Sol en el cielo terrestre a partir de la geometría Sol-Tierra. El objetivo de dicho sistema será servir de ayuda y principal apoyo para el posicionamiento de módulos de placas solares de forma genérica, independientemente del panel/es y su instalación.

Se busca su implementación mediante microcontroladores especializados en tecnologías del Internet de las cosas (IoT) para que desde unos datos limitados externos sobre los módulos de paneles, situación e instante temporal, el sistema calcule las coordenadas de posición de los paneles para asegurar su máximo rendimiento energético.

Abstract

In this work, a preliminary study and development of a simplified system is carried out to obtain the position of the Sun in the Earth's sky from the Sun-Earth geometry. The aim of this system will be to serve as an aid and main support for the positioning of solar panel modules in a generic way, independently of the panel/s and their installation.

Its implementation is sought through microcontrollers specialised in Internet of Things (IoT) technologies so that from limited external data on the panel modules, location and time instant, the system calculates the position coordinates of the panels to ensure their maximum energy performance.

Índice

Notación

DC	Duty Cycle
TSV	Tiempo solar verdadero
ω	Ángulo horario
TSM	Tiempo solar medio
E_t	Ecuación del tiempo
λ	Longitud
TSA	Tiempo solar aparente
TC	Tiempo civil
sen	Función seno
cos	Función coseno
TU	Tiempo Universal
TLE	Tiempo local estándar
λ_S	Longitud del meridiano solar del lugar
AO	Adelanto oficial
TO	Tiempo oficial
Φ	Latitud
Ψ	Acimut
A	Altura solar
θ_z	Ángulo cenital
Δ	Declinación
Γ	Ángulo diario de la Tierra
Γ	Orientación o ángulo acimutal del panel
B	Inclinación del panel
I	Ángulo de incidencia

IoT	Internet of Things/Internet de las cosas

1 INTRODUCCIÓN

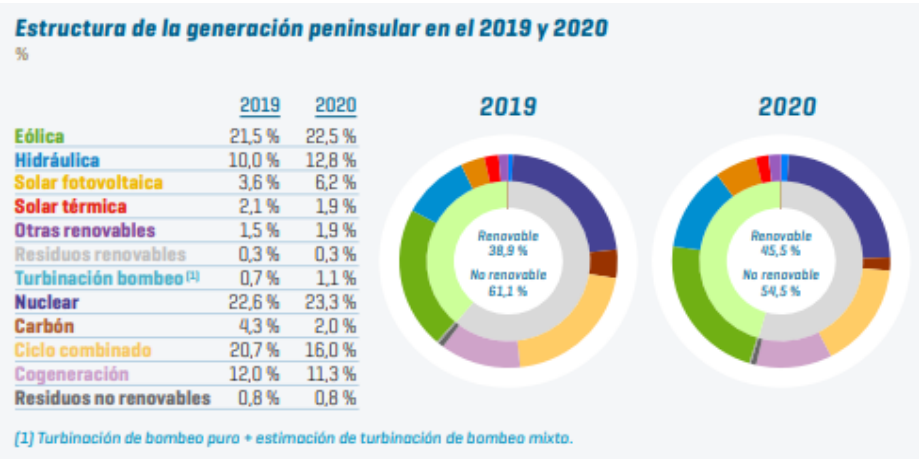
En los últimos 40 años y fruto de la creciente industrialización y el general aumento de la demanda mundial de energía eléctrica, la potencia eléctrica instalada ha aumentado en línea con esta demanda, que tradicionalmente ha sido cubierta principalmente a lo largo de las décadas con el uso de energías basadas en quema de hidrocarburos (carbón, gas, petróleo, …) y energía nuclear.

No obstante, es en estos 40 años que la dependencia de estas fuentes ha cambiado en favor de las energías renovables motivado por el desarrollo de las tecnologías que han mejorado enormemente la eficiencia de estas fuentes, así como la preocupación por la escasez de hidrocarburos, fruto bien del agotamiento de los yacimientos explotables o bien por bloqueos comerciales entre distintos países; y de la amenaza constante de los efectos nocivos del calentamiento global a causa de la acumulación excesiva de CO_2, expulsado a la atmósfera.

Es por eso, que se busca el reemplazo de estas energías por las renovables, que no están exentas de problemas, especialmente la intermitencia en su producción (caso de la solar o la eólica), el almacenamiento eléctrico y la mejora tecnológica en pos de aumentar el rendimiento energético. Será precisamente en este último campo en el que se centre el presente trabajo en el caso concreto de la energía solar.

1.1 Actualidad del sector

España actualmente cuenta con una gran infraestructura eléctrica renovable, actualmente el 45'5 % de la producción energética es renovable (1) siendo la principal fuente, la eólica. La energía solar, representa el 17'9 % de la producción renovable en España, repartida entre solar fotovoltaica (13'7 %) y solar térmica (4'2 %) (1) lo que representa algo más del 8 % de la generación energética en el país.



	2019	2020
Eólica	21,5 %	22,5 %
Hidráulica	10,0 %	12,8 %
Solar fotovoltaica	3,6 %	6,2 %
Solar térmica	2,1 %	1,9 %
Otras renovables	1,5 %	1,9 %
Residuos renovables	0,3 %	0,3 %
Turbinación bombeo (1)	0,7 %	1,1 %
Nuclear	22,6 %	23,3 %
Carbón	4,3 %	2,0 %
Ciclo combinado	20,7 %	16,0 %
Cogeneración	12,0 %	11,3 %
Residuos no renovables	0,8 %	0,8 %

Figura 1-1: Generación peninsular en los años 2019 y 2020

Además desde hace años se ha venido incrementando la instalación renovable total en España y la solar en particular (especialmente en el caso de la fotovoltaica).

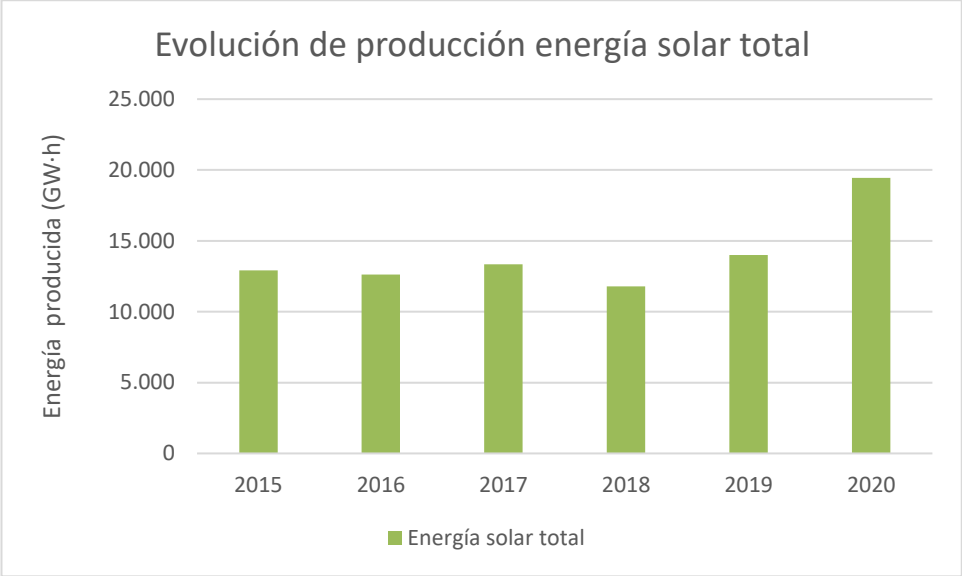


Figura 1-2: Evolución generación energía solar peninsular

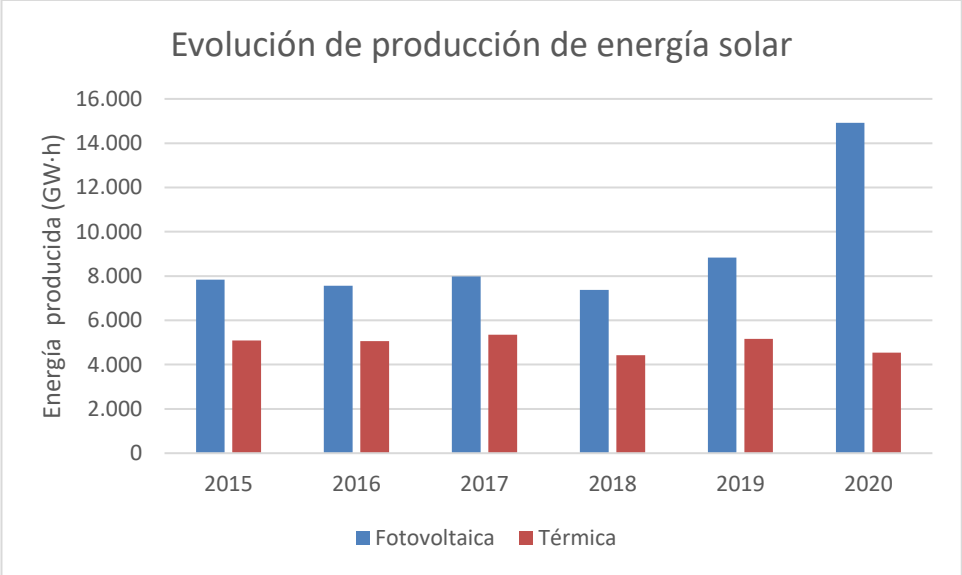


Figura 1-3: Evolución generación energía solar fotovoltaica y termosolar peninsulares

Si bien se ha conseguido una elevada producción, la energía solar presenta variados problemas de eficiencia entre ellos:

- Radiación recibida, los campos solares deben situarse en zonas con gran exposición solar y despejadas de objetos o edificios que generen sombras.
- Dispersión de la radiación solar, la radiación que recibe de la Tierra se ve atenuada al llegar a la superficie terrestre debido a la absorción atmosférica, la difusión a causa de las nubes y el reflejo en superficies blancas y/o pulidas.
- Ángulo de incidencia y orientación, dicho lo anterior, para un máximo aprovechamiento energético, lo ideal es que los rayos solares atraviesen la menor cantidad de atmósfera posible lo que dependerá de los ángulos relativos entre el Sol, la Tierra y los paneles.
- La suciedad de los paneles que dispersa y absorbe la radiación.

- En la fotovoltaica, la eficiencia de los propios paneles fotovoltaicos, por las características de las propias células fotovoltaicas, la dispersión del vidrio protector frente a ellas, el efecto de la temperatura, etc.

Es por todo esto, que se busca un cierto equilibrio en las instalaciones solares a fin de incrementar en tanto sea posible la eficiencia de los paneles.

1.2 Seguimiento del ángulo de incidencia

Como hemos indicado anteriormente, los paneles solares presentan una serie de problemas que afectan en gran medida a su rendimiento y utilidad. La mayoría de ellos dependen de la naturaleza, la situación o del propio panel, dejándonos como una variable a controlar la orientación del panel solar.

Este parámetro puede ser muy determinante, ya que nos puede asegurar una mayor captación de radiación solar en cualquier caso. Para que un panel consiga un aprovechamiento óptimo debe buscarse la orientación hacia el ecuador (dirección sur si se coloca en el hemisferio norte y viceversa) y una inclinación del panel tal que los rayos solares incidan perpendicularmente sobra la superficie del panel, minimizando toda pérdida por dispersión.

A tales efectos, se han desarrollado distintas configuraciones para los paneles en función tanto de la necesidad, la maximización de la producción energética y el presupuesto (2):

- Sistemas estáticos, en los que los paneles desde un primer momento se orientan al ecuador y se ajusta su inclinación en función de la latitud de tal forma que en las horas de mayor irradiación se consiga gran perpendicularidad. Estos sistemas suelen ser usados en las grandes centrales ya que las necesidades de los motores para poder mover los paneles exceden normalmente su presupuesto.
- Sistemas con eje horizontal variable, los paneles se reorientan respecto a un eje norte-sur mientras su inclinación permanece fija.
- Sistema con eje horizontal este-oeste en el que los paneles se orientan según un eje de sentido este-oeste.
- Sistema con eje inclinado λ, en los que los paneles se orientan según un eje inclinado un ángulo λ sobre el plano horizontal.

Sistema de doble eje en el que se puede o seguidor solar, en el que los paneles permiten cambiar su orientación e inclinación para seguir en todo momento la posición del Sol respecto al panel.

2 SISTEMA Y COMPOSICIÓN

Con todo lo comentado anteriormente y pensando en aplicaciones pequeñas como podría ser el consumo doméstico, en edificios comunitarios o pequeñas urbanizaciones dispersas, se desarrollará una calculadora de ángulos de posicionamiento de paneles en base a un microcontrolador que realizará un muestreo periódico de la situación en la zona para recalcular los ángulos. El sistema incluye un conjunto con 2 servomotores sobre los que se actuará a fin de visualizar los resultados y corroborar el funcionamiento.

1.1 Hardware

2.1.1 Microcontrolador

Como microcontrolador, se ha elegido la Wipy 3.0 desarrollada por la empresa Pycom, especializada para aplicaciones de IoT cuenta con una CPU con microprocesador Espressif ESP32 así como un módulo Bluetooth y una antena Wi-Fi integrada con un rango máximo de 1 km pudiendo actuar en modos tanto de estación como de emisor. Entre otras características, se optó por este microcontrolador por sus altas capacidades, rapidez y programación en MicroPython, que al ser código abierto nos da gran disponibilidad de información y facilidad de adaptar códigos similares disponibles.

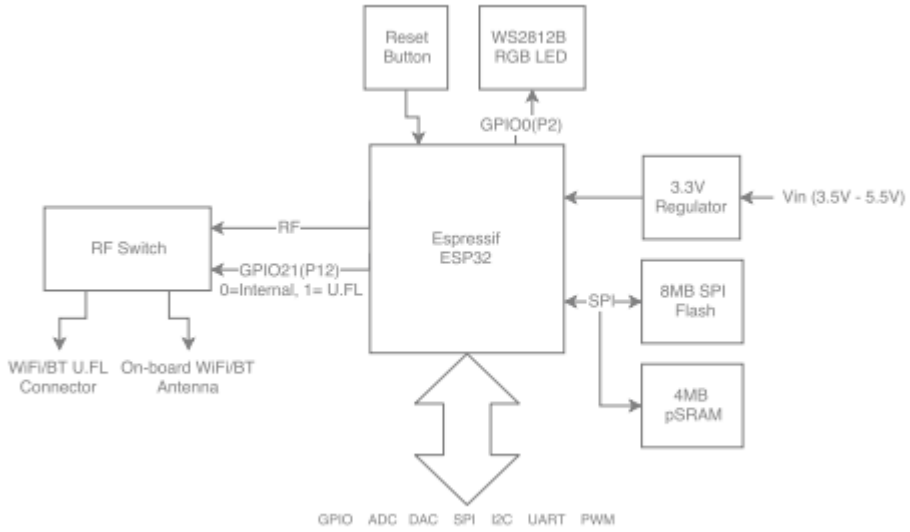


Figura 2-1: Diagrama de bloques de la Wipy 3.0

La Wipy además resulta ideal para estas aplicaciones dado su reducido tamaño para mantenerla resguardada al realizarse el sistema real y en el caso de ser necesario, se le puede añadir una antena externa para aumentar el rango de Wi-Fi permitiendo su uso en zonas alejadas.

Tal y como se ha comprobado la correcta funcionalidad de las conexiones Wi-Fi de la Wipy no dependen solo de esta si no del proveedor de Internet de la red a la que se conecte, ya que por la seguridad interna de estos, la Wipy puede no alcanzar los servidores DNS deseados o incluso no conectarse a la red correctamente.

Figura 2-2: Wipy 3.0

2.1.2 Expansion Board 3

Para el uso de la Wipy 3.0 se ha empleado, además, la Expansion Board 3 también fabricada por Pycom, pensada para manejar especialmente la Wipy y que cuenta además con enchufe para alimentación externa y ranura para SD donde se almacenarán los datos recogidos por los sensores.

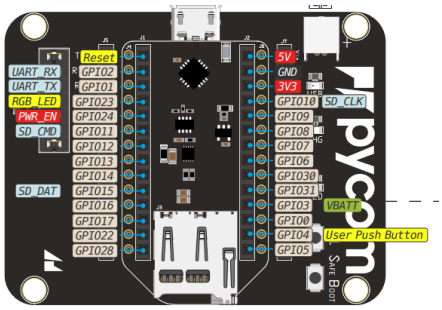


Figura 2-3: Expansion Board 3

2.1.3 Sensors Boosterpack

Para la toma de medidas ambientales necesarias, se ha elegido el BOOSTXL-SENSORS BoosterPack de Texas Instruments (TI) que combina una IMU (acelerómetro-giróscopo BMI160 y magnetómetro BMM150, ambos de Bosch), una unidad de medida ambiental (BME280 de Bosch), un sensor de temperatura (TMP007 de TI) y un sensor de luz ambiental (OPT3001 de TI). Por las características del sistema, se emplea solamente el OPT3001 y la IMU siendo necesaria la debida adaptación de sus drivers por el entorno de programación elegido.

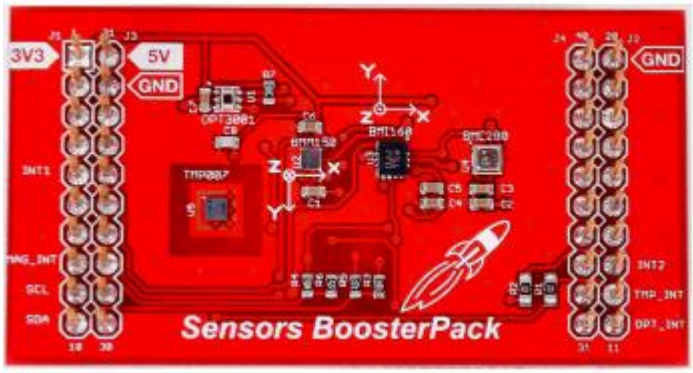


Figura 2-4: BOOSTXL-SENSORS BoosterPack

2.1.4 Servomotores

Para simular la actuación del sistema real se construye un pequeño modelo con 2 servomotores SG90 (3) montados en un soporte móvil de 2 grados de libertad.

Los servomotores tienen un rango de actuación de 180º (de -90º a 90º) y se controlan mediante PWM. El periodo del PWM para los servomotores es de 20 ms, pero su DC es muy limitado, tan solo entre 1 (-90º) y 2 (90º) ms. Además, hay que recalcar que por la definición del PWM dentro de la Wipy, el DC debe expresarse en tanto por 1, dejando el DC de los motores con valores entre el 0’05 y el 0’1 si bien experimentalmente se ha comprobado que se puede llegar a ampliar a 0’03 y 0’105 y que pueden llevar a cabo movimientos sin problemas ajustando el DC a 4 cifras decimales; estos dos factores serán los más importantes a tener en cuenta de cara a los errores y límites de movimiento que tendrá el sistema al realizar experimentos, los movimientos muy precisos no serán procesados por no ser apreciables o porque el motor no los realice y los bruscos o los cambios en el sentido de la irradiación tampoco, si es que superan los 90º que puede alcanzar el motor a cualquier sentido.

2.2 Software

2.2.1 MicroPython

Para el desarrollo de la aplicación (y por las exigencias de la Wipy) se ha elegido el lenguaje de programación MicroPython (4), una implementación de Python 3 creada expresamente para su uso en microcontroladores, el leguaje mantiene casi todas las características de Python adaptando algunas funciones al caso concreto de los microcontroladores (como puede ser la lectura/escritura en registros o el uso de *timers*, ya que lo común es que estén integrados en la propia máquina), incluye el objeto *machine* que integra los constructores y métodos integrados en el propio microcontrolador y excluye la gran mayoría de las bibliotecas propias de Python, que, no obstante, pueden añadirse o en forma reducida si fuera necesario.

La programación se realiza en el editor de texto plano Atom, desarrollado por GitHub para el que se integra el paquete '*Pymakr'* desarrollado por Pycom para sus placas.

2.2.2 Ipinfo.io

Como se explicará a continuación, será necesario saber la posición geográfica del sistema con cierta precisión, para ello, mediante los servicios Wi-Fi integrados en la Wipy se hará una llamada a la web ipinfo.io que permite la geolocalización de una IP dando datos tanto de su posición, área, proveedor de internet, etc.

Hay que destacar, que la IP que puede identificar geográficamente será la del router al que se conecte la Wipy y que será necesario conocer con anterioridad. También decir, que por la seguridad que ofrecen los proveedores de internet, se obtendrá la ubicación del nodo ADS al que pertenece la IP buscada y no la del router al que identifica.

3 PRINCIPIOS DE LA RADIACIÓN SOLAR

Para el desarrollo de nuestro sistema será necesario obtener un modelo matemático que nos permita describir con claridad la posición del Sol en el cielo terrestre, extraer de este el modelo matemático de la incidencia de los rayos solares y por último incluir el modelo de la orientación y la inclinación del panel solar sobre los que trabajará el microcontrolador (2) (5) (6).

3.1 Los movimientos terrestres

La Tierra se mantiene, como el resto de los planetas del Sistema Solar, realizando una órbita elíptica en torno al Sol que se localiza en uno de los focos de la elipse. La excentricidad de la elipse trazada por la Tierra es muy reducida por lo tanto la variación de la distancia entre la tierra y el Sol no es muy elevada. Esta distancia queda descrita por la ecuación:

$$r = r_0 \left\{ 1 + 0'017 sen \left[\frac{2\pi \cdot (d_n - 93)}{365} \right] \right\} \tag{3-1}$$

donde d_n es el día del año (de 1 a 365 desde el 1 de enero) y r_0 es la distancia promedio de la trayectoria:

$$r_0 = 1'495979 \cdot 10^8 \; km \tag{3-2}$$

El periodo orbital de la Tierra es de 365’25636 días.

3.1.1 Rotación terrestre

La Tierra se encuentra girando en torno al eje que atraviesa sus polos geográficos, este movimiento en torno a su propio eje es uniforme y tiene una duración de 23 h 56 min 4’099 s. El eje de rotación Terrestre no es perpendicular al plano de la eclíptica (curva de recorrido aparente del Sol en torno a la Tierra), sino que presenta una inclinación de 23º 27’, es esta inclinación la que provoca la variación de la duración de los periodos diurnos y nocturnos así como las estaciones a lo largo del movimiento de traslación y por tanto es culpable de mayores (o menores) tiempo de exposición de los rayos solares y de la inclinación de estos sobre la superficie terrestre.

3.1.2 Traslación Terrestre

Se llama así al movimiento orbital de la Tierra en torno al Sol, que como ya se ha comentado, no es uniforme, la Tierra es más veloz en el perihelio (punto más alejado al Sol) y más lenta en el afelio (punto más cercano). Así, la distancia Tierra-Sol en cualquier momento del año viene dada por la ecuación:

$$E_0 = 1 + 0'033cos\left(\frac{2\pi n}{365}\right) \tag{3-3}$$

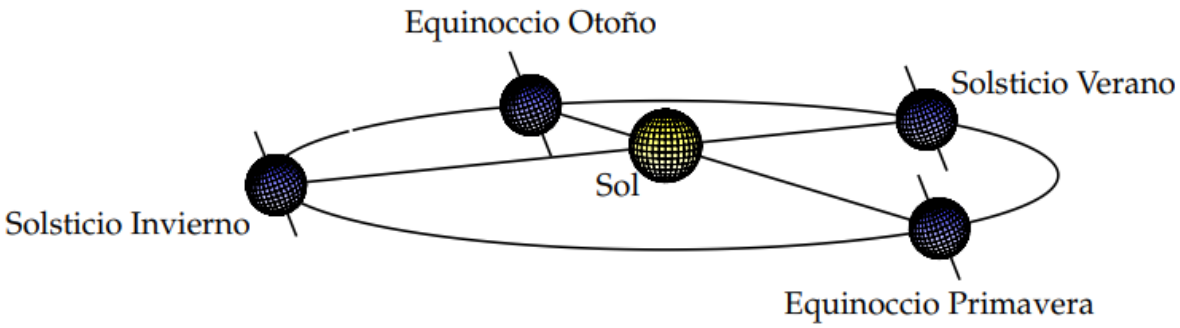


Figura 3-1: Traslación terrestre y puntos singulares en el hemisferio norte

3.2 Tiempo. Solar y oficial

Como paso previo a la situación del Sol en el cielo terrestre, será necesario modelar los instantes solares a partir de las coordenadas celestes, proceso no exento de problemas por la geometría propia de los movimientos terrestres. Una vez obtenidos los instantes solares a partir de la hora oficial (la del reloj) y la localización del sistema, posicionar el Sol en el cielo es trivial.

Definimos el tiempo solar verdadero, TSV, como el ángulo horario del Sol en horas, siendo el recorrido de cada hora de 15˚, de forma que un día solar verdadero es el tiempo que transcurre para que el Sol pase por el mismo meridiano (quedando dividido en 24 horas, siendo el 0 el mediodía):

$$TSV = \frac{\omega}{15} \tag{3-4}$$

Definimos el tiempo solar aparente, TSA como el verdadero más 12 horas de forma que coincide con el día natural.

No obstante, el movimiento aparente del Sol viene de la combinación de los movimientos de rotación y traslación terrestres, que en el caso de este último, no es uniforme en el tiempo dada la órbita elíptica de la Tierra. Por ello y por la necesidad de establecer una escala de tiempo invariante se calcula el tiempo solar medio como:

$$TSV = TSM + E_t \tag{3-5}$$

donde E_t es la ecuación del tiempo, que es el sumatorio de las correcciones necesarias para asegurar que el TSM sea uniforme. E_t es variable a lo largo del año y los valores de los máximos y mínimos pueden variar entre años, de forma que puede graficarse como:

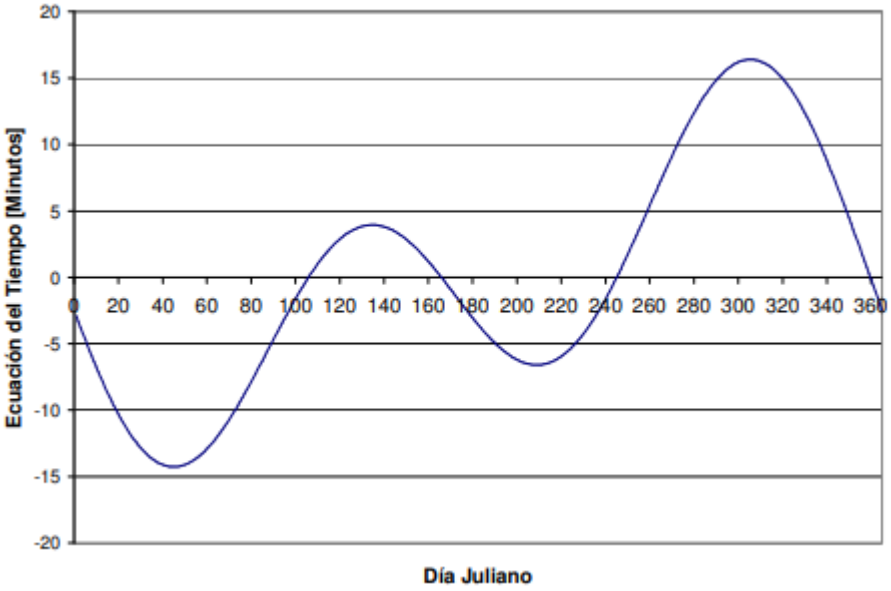


Figura 3-2: Gráfica de la ecuación del tiempo

Definimos además el tiempo civil, TC, como el TSM más 12 horas, contándose desde el 0 a medianoche. Para longitudes distintas de la Tierra existen por tanto un TC distinto, siendo la diferencia proporcional a la longitud; como referencia para distintas longitudes se define el tiempo universal, TU, como el TC del

meridiano de Greenwich[1], tal que:

$$TC = TU + \frac{\lambda}{15}^{2} \tag{3-6}$$

Dado que el TC tiene un aspecto local, la superficie terráquea se encuentra dividida en 24 husos horarios cada uno de 15º, en cada uno se define su tiempo local estándar, TLE, de tal forma que la diferencia entre TC y TLE en una zona es:

$$TC - TLE = \frac{(\lambda - \lambda_S)}{15} \tag{3-7}$$

donde λ es la longitud local y λ_S la del meridiano central del uso.

Se define, además el tiempo oficial, TO (aquel que emplean los relojes comunes) como:

$$TO = TLE + AO \tag{3-8}$$

donde AO es el adelanto oficial propio de cada huso horario, sometido además a las normativas de cada país (en el caso de España, en horario de invierno se suma 1 hora y en el de verano, 2). Así la relación entre TO y TSV se expresa como:

$$TSV = TO - AO + \frac{(\lambda - \lambda_S)}{15} + E_t - 12 \tag{3-9}$$

con el tiempo expresado en horas, al igual que E_t y las longitudes en grados.

[1] Meridiano 0 actualmente, en otras épocas se emplearon otras referencias.
[2] Siendo λ la longitud geográfica en sentido positivo hacia el Este

3.3 Movimiento aparente del Sol

La combinación de la rotación y traslación terrestres se ven desde la superficie terrestre como el movimiento aparente del Sol a lo largo de la esfera celeste. Será esta posición la fundamental para estimar la radiación que recibirá un módulo de paneles solares y en el caso del sistema, la posición a la que orientarse.

Todo punto en la superficie terrestre puede determinarse mediante 2 coordenadas geográficas:

Latitud, Φ, el ángulo que forma la recta normal al punto con plano ecuatorial que se mide de 0º a 90º con sentido positivo hacia el Norte.

Longitud, λ, el ángulo que forman el meridiano que pasa por el punto y el de Greenwich. Se mide de 0º a 180º con sentido positivo hacia el Este

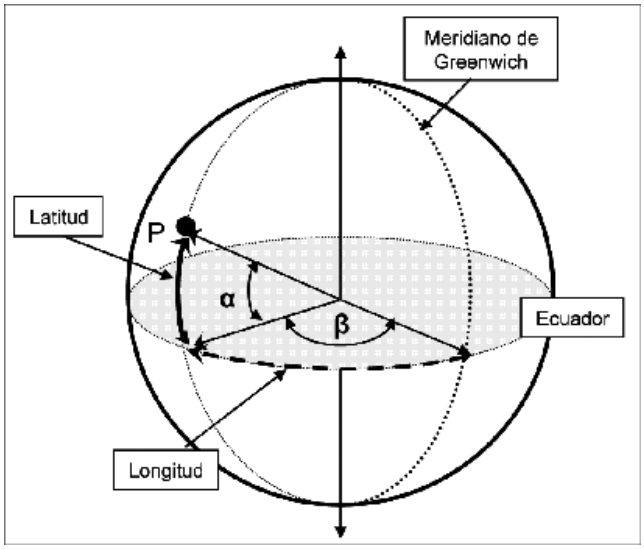


Figura 3-3: Latitud y longitud sobre esfera celeste

3.3.1 Sistema de coordenadas celestes horizontales

Para determinar la posición relativa del Sol, es necesario imponer un sistema de referencia que nos permita localizar las coordenadas angulares del Sol en la esfera celeste, se suelen emplear 2 posibles sistemas para situar cualquier astro en la esfera celeste:

El sistema de coordenadas celestes horizontales, aquel que tendría un observador situado en la superficie terrestre. Su plano principal es el del horizonte, el perpendicular a la recta vertical sobre el observador (el punto de corte de esta con la parte superior de la esfera es el denominado, cenit; mientras que el inferior es el nadir). Para este sistema, existen 2 coordenadas:

El acimut, Ψ, arco formado por la proyección del astro sobre el plano del horizonte y el Norte cardinal (en el hemisferio sur) o el Sur cardinal (hemisferio norte). Se mide positivamente en sentido horario desde el punto cardinal de referencia.

La altura solar, α, arco formado la recta que une el astro y el observador con el plano horizontal, se mide de 0º a 90º de forma positiva hacia el cenit. Habitualmente, no se emplea la altura solar, sino su complementario el ángulo cenital, θ_z, que además suele definirse entre 0º y 180º, de forma que la relación es:

$$\theta_z = 90 - \alpha \tag{3-10}$$

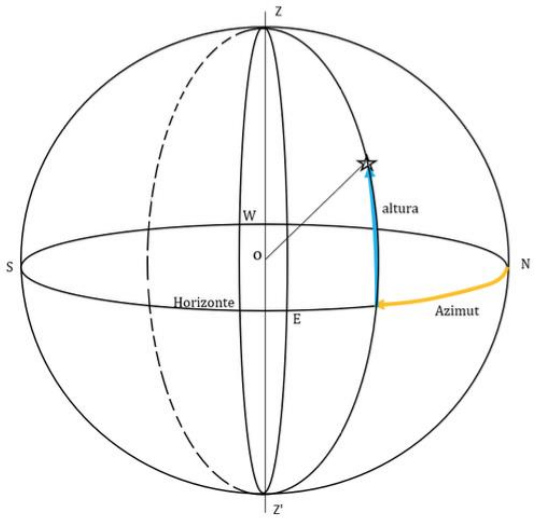


Figura 3-4: Coordenadas celestes horizontales

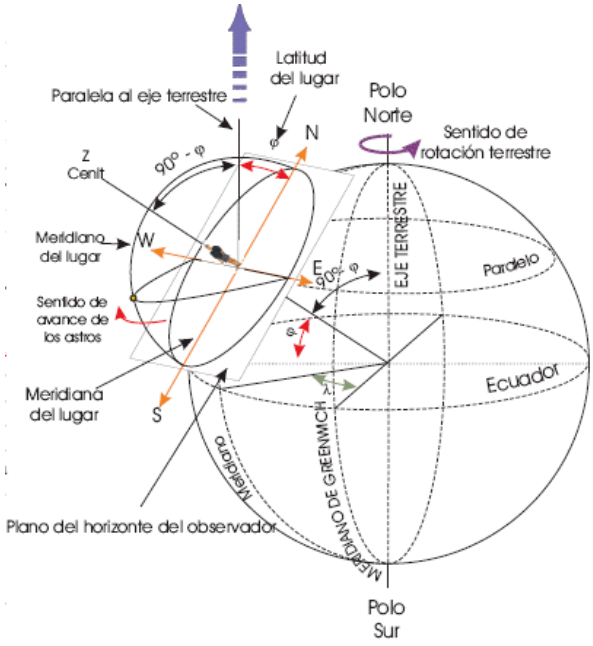


Figura 3-5: Coordenadas celestes horizontales vistas sobre la esfera terrestre

3.3.2 *Sistemas de coordenadas celestes horarias*

Por otro lado tenemos el sistema de coordenadas celestes horarias, que se basa en la rotación celeste, tomando el eje de rotación de la Tierra como referencia, así define los polos celestes (Norte y Sur del eje de giro) y el ecuador celeste, se toma además la referencia del meridiano superior del lugar, aquel que parte del Norte celeste y corta al plano del horizonte pasando por el cenit. Este sistema se basa también en dos coordenadas:

Ángulo horario, ω, ya explicado con anterioridad y que es el ángulo a lo largo del ecuador celeste que forman el meridiano superior del lugar y el circulo horario donde se sitúa el astro. Está definido entre 0º y 180º positivos hacia el Oeste.

Declinación, δ, arco entre el ecuador celeste y el astro, definido de 0º a 90º positivos en sentido Norte.

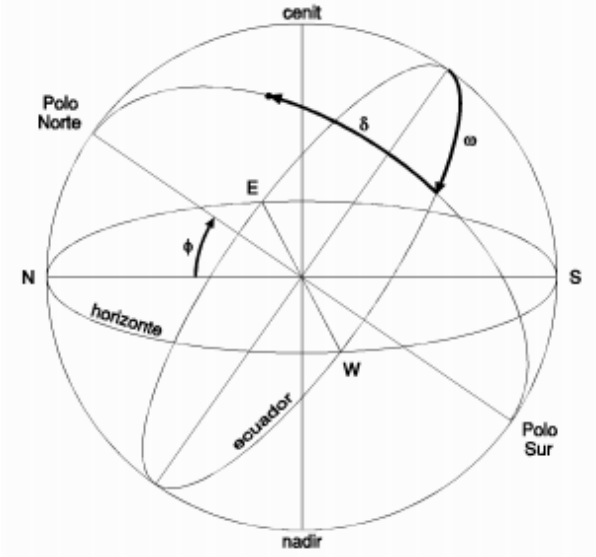


Figura 3-6: Coordenadas celestes horarias

El ángulo que forman los planos del horizonte y el ecuador celeste depende de la situación del observador, de hecho, la latitud del observador es la altura del Norte celeste respecto del horizonte. De este modo, es posible obtener la relación entre unas coordenadas y otras, así el ángulo cenital puede definirse como:

$$cos(\theta_z) = cos(\delta)\, cos(\omega)\, cos(\Phi) + sen(\delta) sen(\Phi) \tag{3-11}$$

mientras que el ángulo acimutal se puede definir como:

$$cos(\Psi) = signo(\Phi) \cdot \frac{cos(\delta)\, cos(\omega)\, sen(\Phi) - sen(\delta)\, cos(\Phi)}{sen(\theta_z)} \tag{3-12}$$

3.3.3 Ecuaciones aproximadas

En el análisis de datos anuales de radiación solar, se suelen emplear años de 365 días frente al calendario gregoriano, lo que simplifica los cálculos y no introduce un error muy severo (ya que la duración del año es 365 d 6 h 8 min aprox.). teniendo en cuenta esto y que la declinación a lo largo de un año es más o menos constante podemos aproximar la posición del Sol en el cielo según las ecuaciones :

$$\omega = 15 \cdot t \tag{3-13}$$

con t en horas TSV y ω en grados.

Y la declinación en grados por la expresión:

$$\delta = 0'3723 + 23'2567 sen\ \omega_t - 0'7580 cos\ \omega_t + 0'1149 sen\ \omega_t + \cdots \tag{3-14}$$
$$\ldots + 0'3656 cos\ \omega_t - 0'1712 sen\ \omega_t + 0'0201 cos\ \omega_t$$

donde

$$\omega_t = \frac{360}{365} \cdot (n - 79'436) \tag{3-15}$$

con n, el n.º del día del año.

Por último la E_t expresada en minutos se puede aproximar como:

$$E_t = \frac{229'2}{60} \cdot [0'000075 + 0'001868cos\,\Gamma - 0'032077sen\,\Gamma + \ldots - 0'014615cos(2\Gamma) - 0'04089sen(2\Gamma)] \quad (3\text{-}16)$$

con Γ:

$$\Gamma = \frac{2\pi \cdot (n-1)}{365} \quad (3\text{-}17)$$

3.4 Incidencia de los rayos solares en una superficie plana

Una vez caracterizada la posición relativa del Sol respecto a la Tierra, modelamos la posición relativa del Sol respecto a una superficie plana (los paneles solares en nuestro caso); que se obtiene de la del Sol (o cualquier astro en general) y de la posición del propio panel, expresada por 2 ángulos:

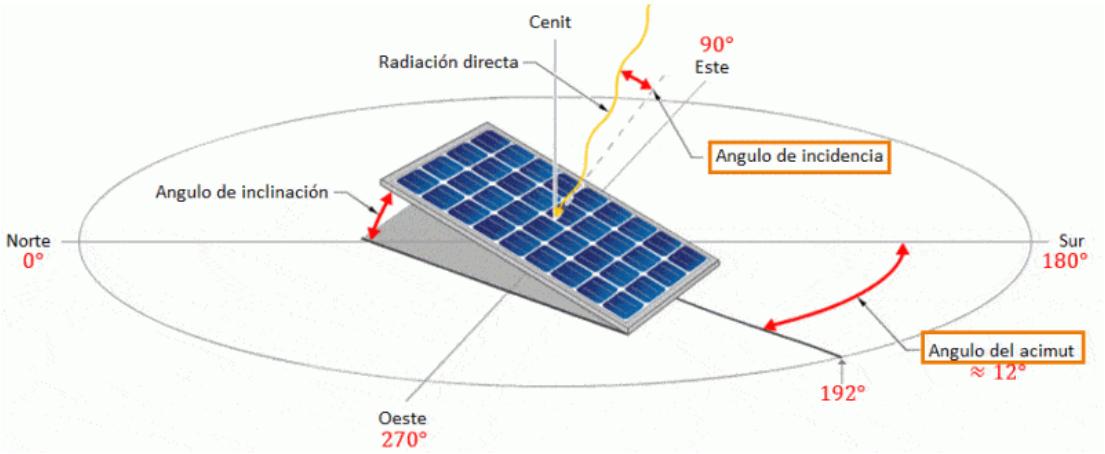


Figura 3-7: Ángulos de orientación en un panel solar

Orientación o ángulo acimutal, γ, definido como el ángulo formado entre el meridiano solar y la proyección horizontal de la perpendicular a la superficie. Se mide desde el Sur (si se está en el hemisferio norte y viceversa) variando entre 0º y 180º en sentido positivo hacia el Oeste.

Inclinación o pendiente, β, el que forman superficie y plano horizontal y definido positivo en sentido horario.

La irradiación sobre una superficie depende del ángulo de incidencia , *i*, de los rayos solares, aquel que forman los rayos y la recta perpendicular a dicha superficie y cuya expresión es:

$$\cos i = \cos\beta \cdot sen\,\alpha + sen\,\beta \cdot \cos\alpha \cdot \cos(\Psi - \gamma) \tag{3-18}$$

siendo β, α, Ψ y γ los distintos ángulos definidos anteriormente.

3.5 Principio de funcionamiento para el seguimiento con doble eje

En el caso general, es necesario tener en cuenta el ángulo *i*, ya que buscamos para cualquier panel buscamos el mayor aprovechamiento de la radiación. Esto se consigue asegurándose de que los rayos solares incidan perpendicularmente a la superficie, es decir, que la normal a la superficie del panel, sea paralela a los rayos que inciden directamente.

Sin embargo, en el caso de un seguidor de doble eje, se controla tanto la orientación como la inclinación del panel, esto soluciona enormemente el problema y simplifica los cálculos, ya que las ecuaciones del modelo del seguidor son de la forma:

$$\beta = \theta_z \tag{3-19}$$

$$\gamma = \Psi \tag{3-20}$$

Por tanto para asegurar el cálculo correcto de los ángulos y el movimiento del sistema será necesario obtener, la situación geográfica precisa del sistema y los ángulos horario y declinación para los que se necesitará conocer la fecha y hora de la zona donde se encuentre el sistema.

4 IMPLEMENTACIÓN DEL SISTEMA

En este capítulo se describirá la solución adoptada para el desarrollo del sistema y los programas realizados para su implementación. Se ha buscado una solución genérica para cualquier situación de uso de forma que con unos pequeños cambios el sistema esté preparado para distintos entornos

4.1 Estructura del sistema

Tal y como se refleja en el diagrama adjunto, el centro del sistema es la Wipy. El microcontrolador vía Wi-Fi obtiene los datos de geolocalización de la IP y la hora de la red y los guarda, con estos datos y los datos de valor del campo magnético aportados por el magnetómetro, calcula los ángulos locales de los rayos solares y genera el '*duty cycle'* a aplicar a los servomotores. Mientras tanto el sensor de luz tomará medidas como comprobación de la luz incidente. Todos los datos son enviados a una interfaz (bien en el *REPL* de Pycom, bien en PuTTY) para poder llevar su seguimiento, además los valores obtenidos se guardan en la SD extraíble incluida.

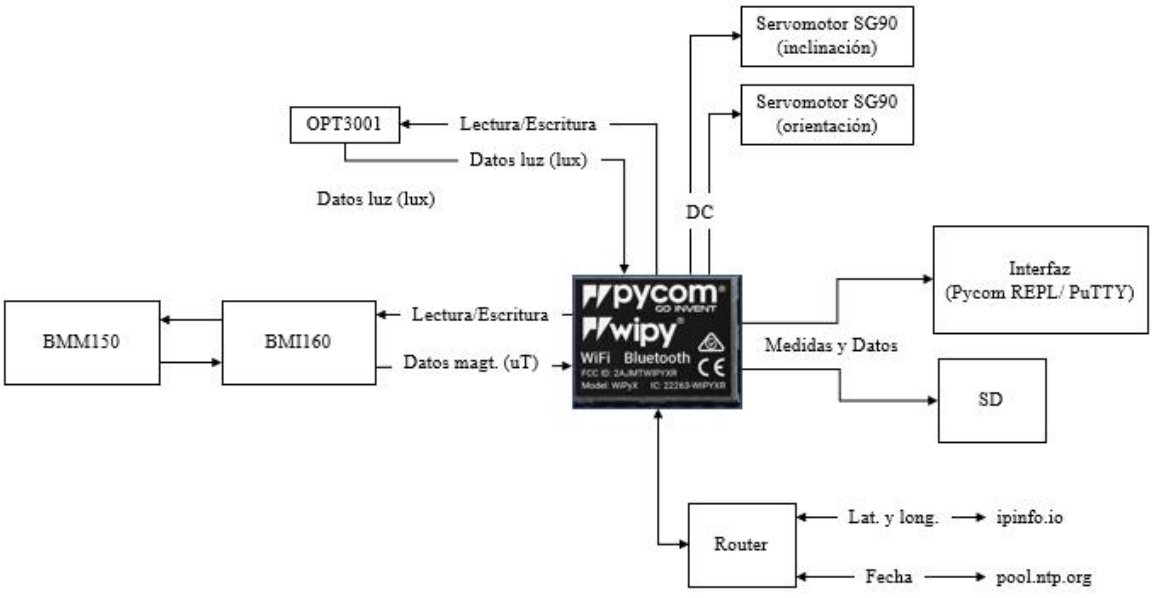


Figura 4-1: Diagrama del sistema y comunicaciones

Tal como se ve en la imagen, en el montaje del modelo, tenemos la base con 2 grados de libertad con los 2 servos y sobre esta, el Sensors Boosterpack de tal forma que a partir del magnetómetro se obtenga la orientación del panel y el sensor lumínico reciba la luz directamente. La Wipy está directamente conectada a los pines del Sensors Boosterpack y al cable PWM de los motores y a través de la *protoboard* a los pines de

tensión y tierra de los motores.

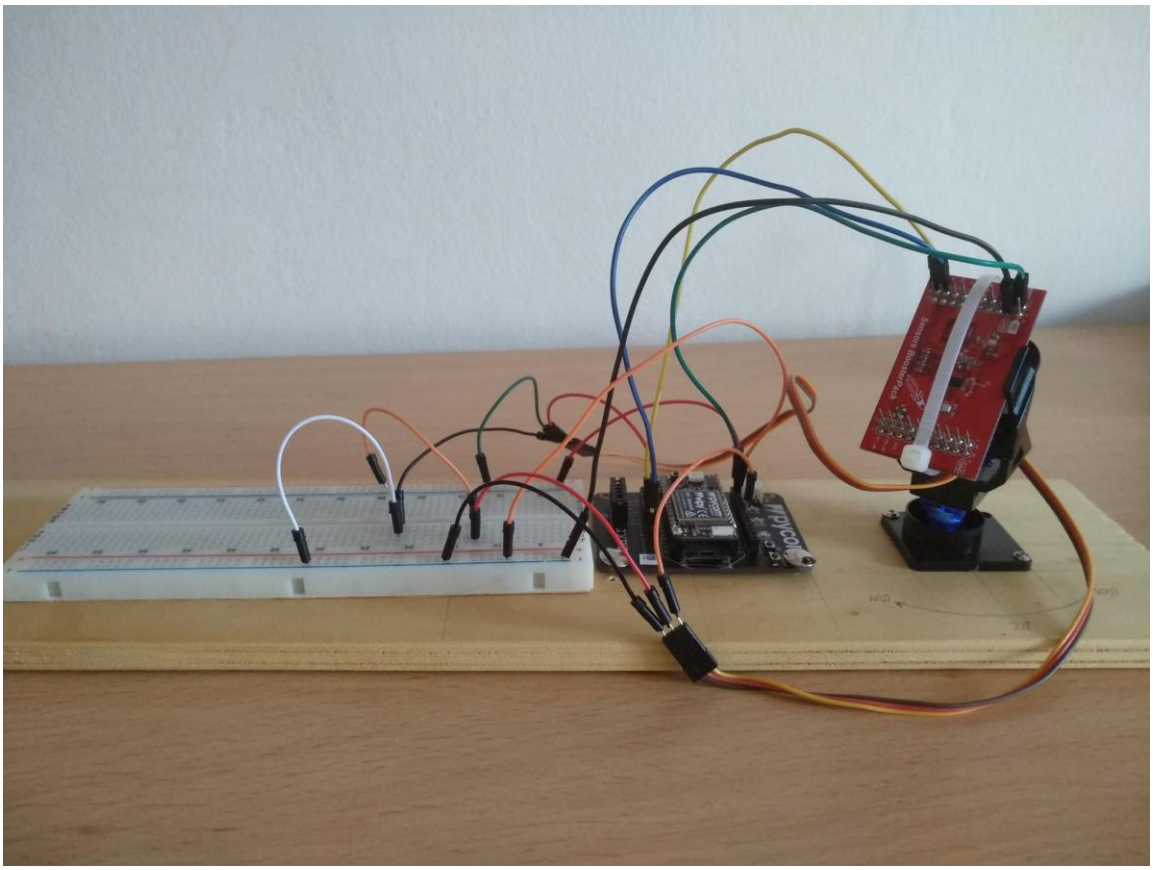


Figura 4-2: Montaje del modelo para el sistema

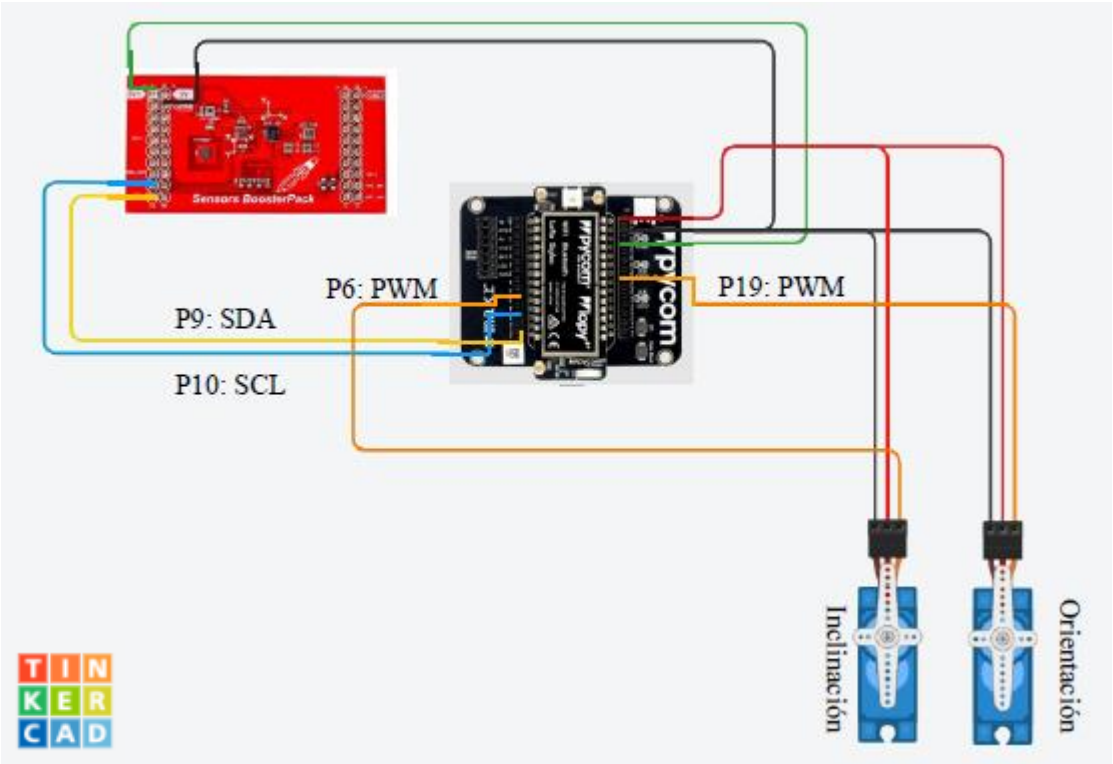


Figura 4-3: Diagrama de conexionado del sistema

4.2 Módulos de la Wipy 3.0

Principalmente se han usado 4 de los módulos incluidos con el microcontrolador, el bus I^2C para la comunicación con los sensores, los pines para comunicación y PWM, la tarjeta SD y los servicios WLAN.

4.2.1 Tarjeta SD

Se planteó el uso de la SD como modulo extraíble de almacenamiento de datos, para guardar los datos de las medidas y resultados para su futuro análisis. Tal y como se verá en el código, se crean 3 archivos de texto dentro de la tarjeta donde se guardarán, las medidas de los sensores (‘medidas.txt’), el valor de las coordenadas solares (‘angulos.txt’) y el valor dado a los servomotores como ángulo y como DC (‘motor.txt’).

4.3 Firmware

Para el uso de los distintos sensores, ha sido necesario la adaptación de los drivers existentes a la estructura de MicroPython, bien adaptando una API ya existente en Python o reescribiendo el driver desde otros lenguajes (C, C++ [Arduino], …). Como se comentó, se emplean 2 de los sensores de Sensors Boosterpak, el sensor de luz ambiental y el magnetómetro, ambos con comunicación vía bus I^2C.

4.3.1 Driver del OPT3001

En este caso se dispone de un pequeño driver en MicroPython (7) que incluye algunas de las funciones que puede realizar el driver de TI, entre ellas configurar el sensor a modo de máxima escala para la conversión de medidas, la lectura de los registros de medida e identificación del sensor. Solo se ha incluido la dirección del sensor en el Boosterpack (0x47) y las funciones de acción sobre los registros en I^2C, ya que el código encontrado estaba pensado para el SMBus de una Raspberry Pi.

4.3.2 Driver del BMM150

El magnetómetro incluido en el Boosterpack está conectado como sensor secundario y de apoyo al acelerómetro-giróscopo BMI160, de forma que es inaccesible directamente, para ello se parte de un driver en Python para el BMI160 (8) basado en el original de Boch para Arduino y se incluyen las funciones de manejo de registros propias de MicroPython. Para la acción sobre el magnetómetro, se toma de referencia este código (9), que añade a la API de Arduino antes mencionada el control del magnetómetro y se adapta el código a MicroPython. A continuación se muestra el código de las funciones que manipulan los registros, el resto del código que se ha insertado se muestra en el Anexo ya que consiste en llamadas a estas funciones con los registros específicos para el BMM150.

Código 4.1 Manipulación de registros BMI160/BMM150

```
class BMI160_I2C(BMI160):
    def __init__(self, i2c, addr=0x69):
        self.i2c = i2c
        self.addr = addr
        super().__init__()

    def _reg_write(self, reg, data):
        self.i2c.writeto(self.addr, bytes([reg, data]))

    def _reg_read(self, reg):
        return self._regs_read(reg, 1)[0]

    def _regs_read(self, reg, n):
        self.i2c.writeto(self.addr, bytes([reg]))
        sleep_us(2)
        return self.i2c.readfrom(self.addr, n)
```

4.4 Software

La aplicación se ha desarrollado en 2 archivos, también se incluye el archivo boot.py que el microcontrolador utiliza de referencia para establecer la comunicación por UART y que indica que archivo se considera *main*.

4.4.1 Ecuaciones.py

Ecuaciones.py es un archivo de cálculo que se encarga de las operaciones para la obtención de coordenadas solares, en concreto se implementan las ecuaciones (3-11), (3-12), (3-13), (3-14), (3-15), (3-16) y (3-17) [3] y la operación angulo_mag, que calcula la orientación actual del Sistema respecto del norte magnético a partir de las medidas del magnetómetro, para después cambiar la referencia (el 0º) al Sur, tal y como se define la orientación de un panel en el hemisferio norte.

4.4.2 Main.py

En este archivo se desarrolla toda la aplicación del sistema.[4]

El archivo es posible dividirlo en 3 grandes secciones.

- Las inicializaciones, donde activamos y configuramos el bus I^2C, 2 pines PWM, los servicios WLAN como estación (para conectarse a Internet), montar la SD y generar en ella los 3 archivos donde se almacenarán los datos.
- Las operaciones previas, donde se realiza la conexión a la red más cercana disponible y las operaciones que se mantendrán a lo largo de todo el tiempo de ejecución, como son la obtención de la hora, la geolocalización del sistema y la obtención de la ecuación del tiempo y la declinación (ambas ctes. para un día completo). La Wipy, tiene incluido un RTC, que por defecto y tras cada reinicio se inicia el 1 de enero de 1970 (fecha inicial de referencia para POSIX), por tanto es necesario actualizar su valor a la fecha actual para los cálculos necesarios. Para ello y como se ve en el código inferior, se accede a los servidores del proyecto en línea pool.ntp.org y se sincroniza al RTC con ellos; en concreto se sincroniza con la hora del meridiano 0, lo que se tendrá en cuenta para el resto de los cálculos.

Código 4.2 Sección de actualización de fecha del RTC

```
 rtc = machine.RTC()
 rtc.ntp_sync("pool.ntp.org")
 time.sleep_ms(3)
 while not rtc.synced():
     machine.idle()
 print("RTC synced with NTP time/es/")
 print("Son las {3}:{4}:{5} del {2} de {1} del
{0}".format(*time.localtime()))
 n = time.localtime()[7]
```

Para la geolocalización del sistema se emplean, como ya se comentó, los servicios de '*ipinfo.io'* y se

[3] Los cambios de unidades angulares se han especificado dentro de los propios argumentos de entrada
[4] Solo se muestra tramos de código de interés. El resto se encuentra en Anexo

desarrolla la búsqueda a partir de una aplicación obtenida de 10, que registra la IP especificada y obtiene un diccionario con los datos de la IP, localización geográfica aproximada (en realidad se obtiene normalmente la del nodo ADS del proveedor de Internet de una IP), nombre, nombre del proveedor, ciudad donde se localiza, etc.

En concreto, guardamos en una variable *LOC* el valor asociado a la clave 'loc' del diccionario que incluye una cadena de caracteres con la latitud y longitud separadas por una coma. Se localiza la posición de la coma y se separan las coordenadas a partir de esa posición realizando la conversión a flotante para los cálculos.

Código 4.3 Sección para obtener la localización del sistema

```
 def ipInfo(addr=""):
      if addr == "":
         url = "https://ipinfo.io/json"
      else:
         url = "https://ipinfo.io/" + addr + "/json"

      response = urlopen(url)
      data = ujson.load(response)
      return data


 # Localización del sistema
 LOC = ipInfo("85.137.11.21")['loc']
 coma = LOC.index(",")
 LAT = LOC[0:coma]
 LON = LOC[coma+1:]

 phi     = float(LAT) # Pasamos latitud y longitud de str a float
 lamda   = float(LON)
 print("El sistema se encuentra en la latitud: " + str(phi) + " y
longitud: " + str(lamda))
```

- Las operaciones continuas, que incluyen la lectura de los sensores, su escritura en la interfaz y almacenamiento de datos en la SD. La obtención de la hora solar, la orientación del sistema respecto al Sur, el ángulo cenital y el acimut realizando llamadas a las funciones incluidas en *'Ecuaciones.py'* así como su escritura tanto en la SD como en la interfaz. Por último, se incluye el ajuste a los servomotores para transformar el ángulo en DC en función del arco que sea capaz de recorrer el servomotor aproximadamente y las saturaciones dentro de los límites de DC que tienen los servos.

Código 4.4 Ajuste de los ángulos al DC

```
 # Ajustamos los ángulos al rango de los servos
 # considerando una posición más o menos cte.
 gamma = 0.1 - acimut*(0.1)/(PI)
 gamma = round(gamma, 4)
 beta = 0.075 - cenital*(0.075 - 0.05)/(PI/2)
 beta = round(beta, 4)

 # Redondeamos para el ajuste a los valores de los servos
```

```
if (gamma > 0.1):
    gamma = 0.1
elif(gamma < 0.03):
    gamma = 0.03

if (beta > 0.075):
    beta = 0.075
  elif(beta < 0.05):
      beta = 0.05
```

A modo aclarativo, se incluye el diagrama de flujo de la aplicación

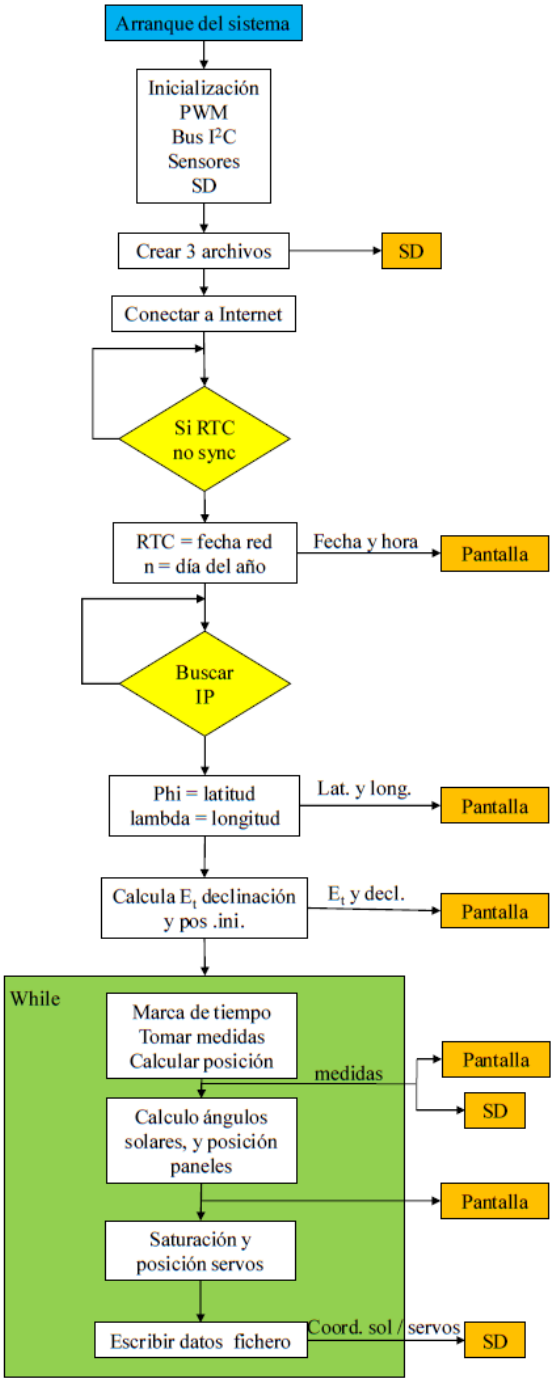


Figura 4-4: Diagrama de flujo del sistema

5 RESULTADOS EXPERIMENTALES

En este capítulo se mostrarán y explicarán los resultados obtenidos para una serie de pruebas sobre el sistema con tal de comprobar que este reacciona correctamente. De cara a cada experimento se ha tenido en cuenta la orientación del sistema y el rango de giro del servomotor inferior (correspondiente a la orientación de la base de 2 grados de libertad).

5.1 Metodología

Debido a la no disponibilidad de un entorno para probar correctamente el sistema (terraza, patio, azotea, etc.); el sistema se ha probado en interior por lo que la luz recibida por el OPT3001 no tendrá unos valores relevantes para la aplicación, que si se incluirán como documentación. Además, y dado que el movimiento relativo del Sol respecto a la Tierra es lento, necesitando al menos 10 min para obtener cambios apreciables en los ángulos y en el movimiento de los servos, la aplicación se ha manipulado para que realice los movimientos más rápidamente; se incluye un contador incremental que añade 1º o 2º, según el caso, al valor del ángulo horario, así, en 1 minuto de ejecución para el sistema habrán transcurrido 6º (aproximadamente 20 min).

A todo esto añadir, que realizar las pruebas de esta manera lleva al sistema a horas extremas (10, 11 de la noche) para las que el ángulo cenital tendrá valores por encima de 90º y que por tanto, no se tendrán en cuenta, ya que exceden la inclinación máxima que tiene un panel.

De cara a representar las distintas variables, se ha optado por emplear el TO de Sevilla calculado a partir de los ángulos solares dados y la ecuación (3-9).

5.2 Experimento 1.

Para este experimento, el sistema se coloca orientado 28’9º respecto al Sur hacia el Oeste, se incrementa el ángulo horario 1º en cada ejecución (cada 10 s) y se incrementa hasta alcanzar cerca de las 24 h (TO), equivalentes a 144’008º.

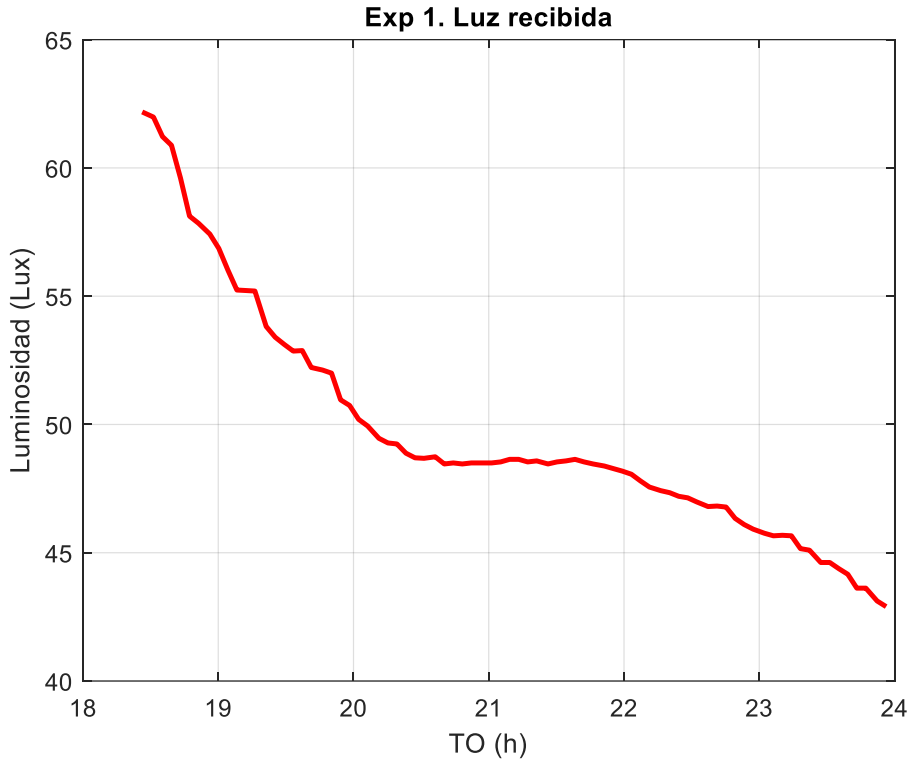


Figura 5-1: Luz recibida por el sensor en el experimento 1

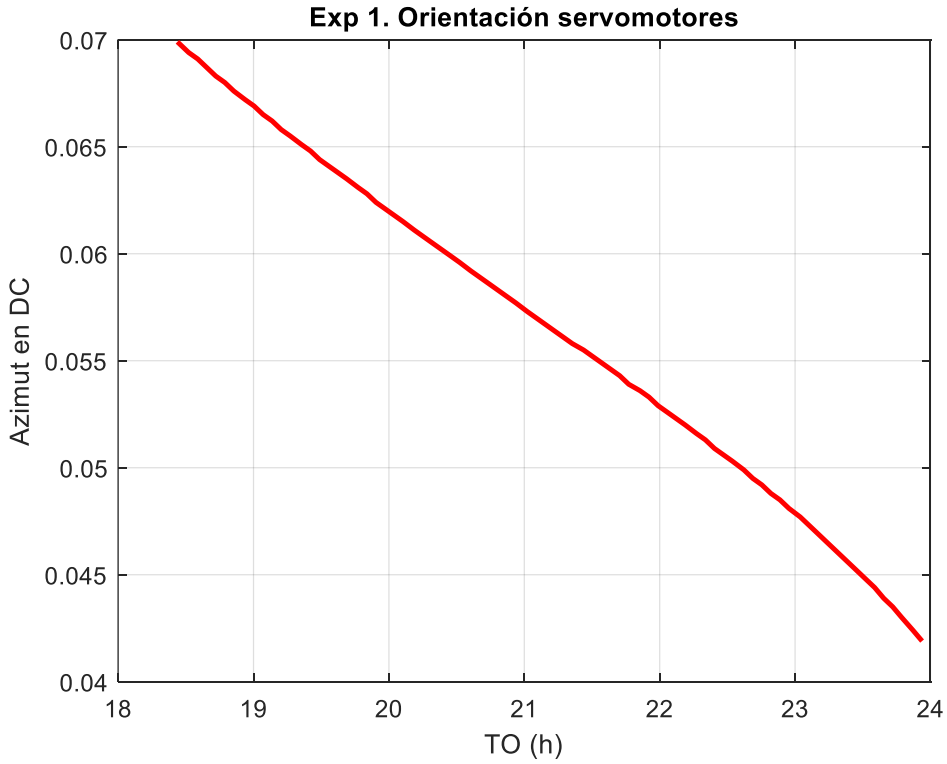


Figura 5-2: Orientación del servomotor inferior en DC

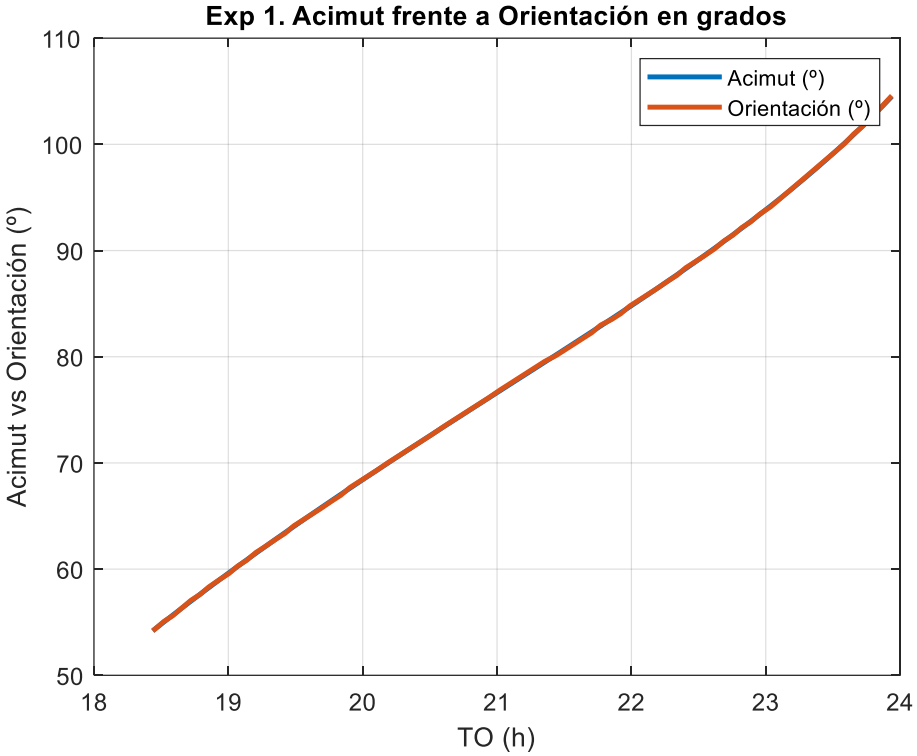


Figura 5-3: Comparativa entre acimut y orientación del motor inferior

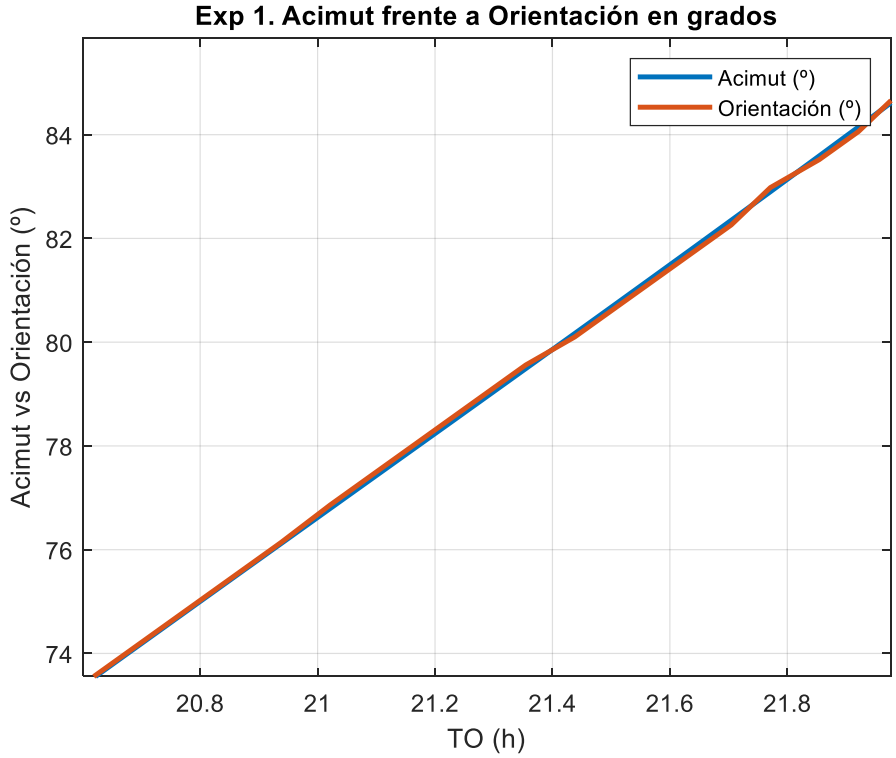


Figura 5-4: Detalle comparativa acimut y orientación

Como podemos ver en las gráficas anteriores, se recibe un valor reducido de luz que decrementa al cambiar la orientación (debido a que la posición inicial en este y los demás experimentos el sensor está orientado hacia una ventana). Por su parte vemos que el ángulo de interés, el acimut incrementa entre 55º y 105º y a señal dada al servomotor (transformada de DC a grados lo sigue con bastante precisión), si vemos en detalle, la diferencia entre ambas gráficas son unas pequeñas oscilaciones. El sentido en la gráfica de la orientación en DC está invertido ya que por la colocación de la base en el montaje, un DC de 0’1 se corresponde con 0º y 0’03 a 180º.

5.3 Experimento 2.

El experimento es similar al anterior, pero el sistema está orientado 50º respecto al Sur hacia el Oeste.

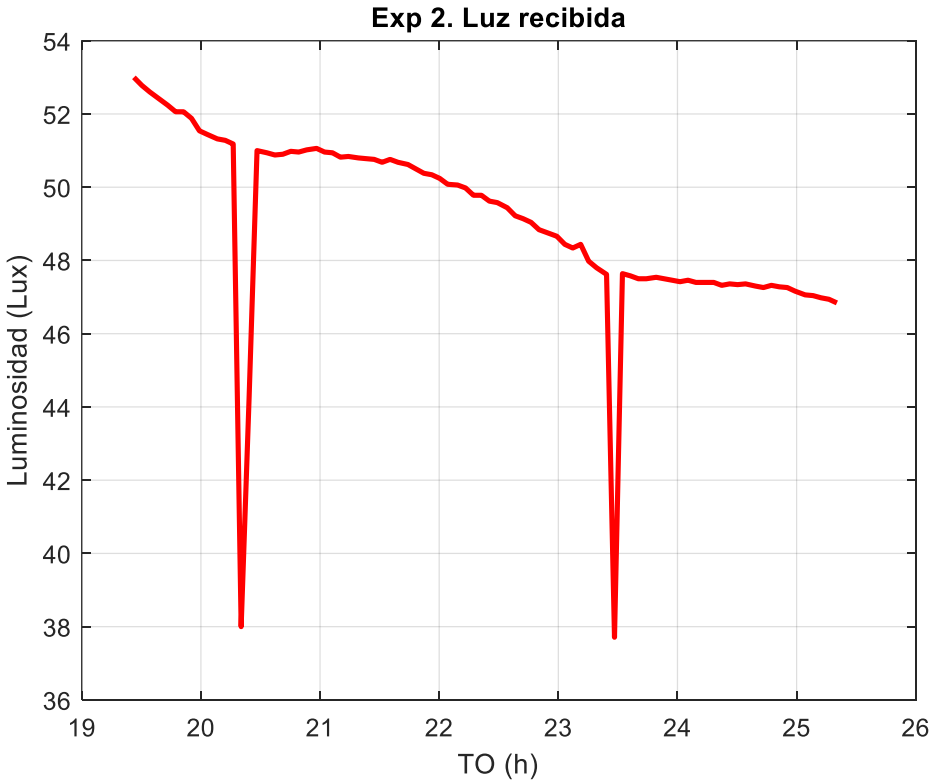


Figura 5-5: Luz recibida por el sensor en el experimento 2

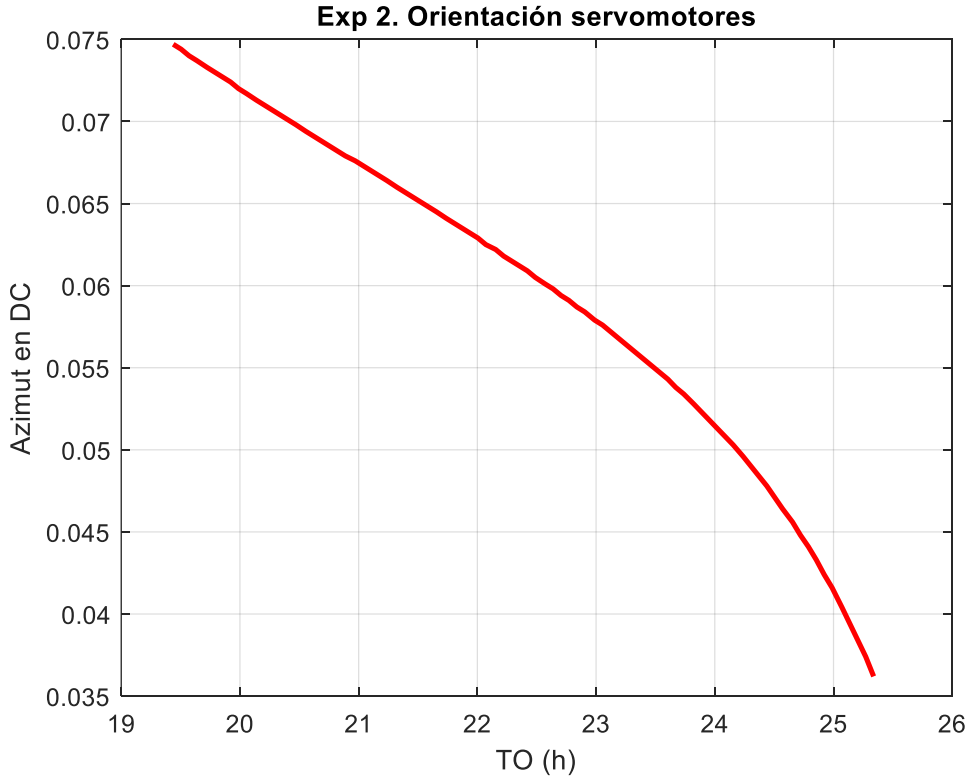


Figura 5-6: Orientación del servomotor en DC

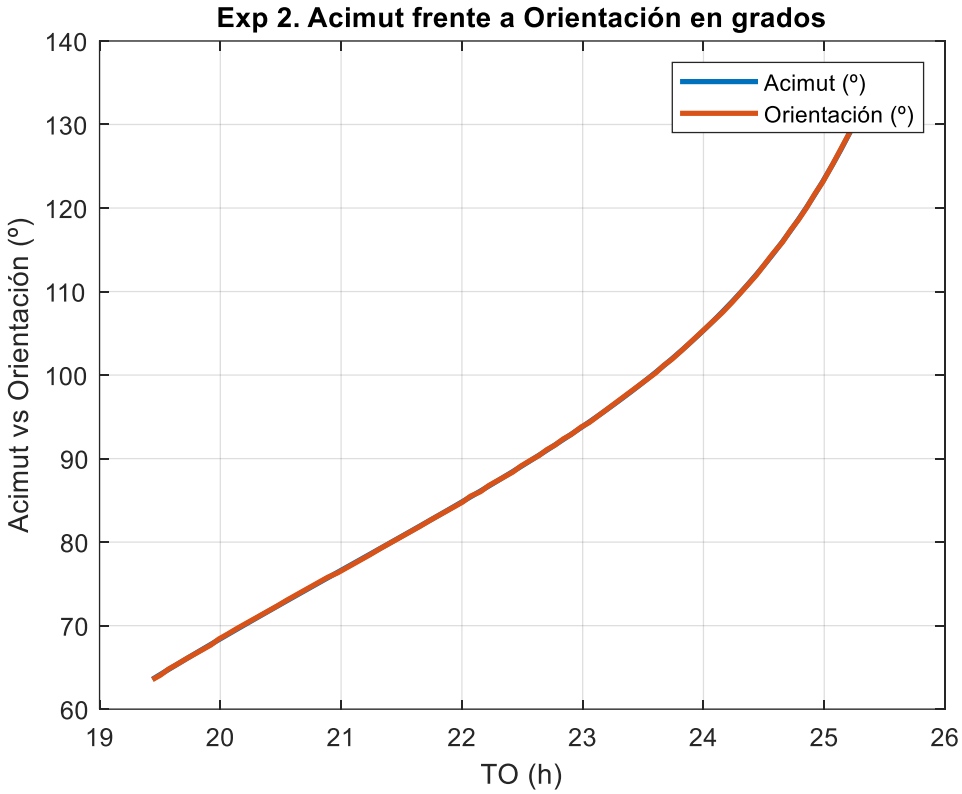


Figura 5-7: Acimut frente a orientación del motor en ángulos

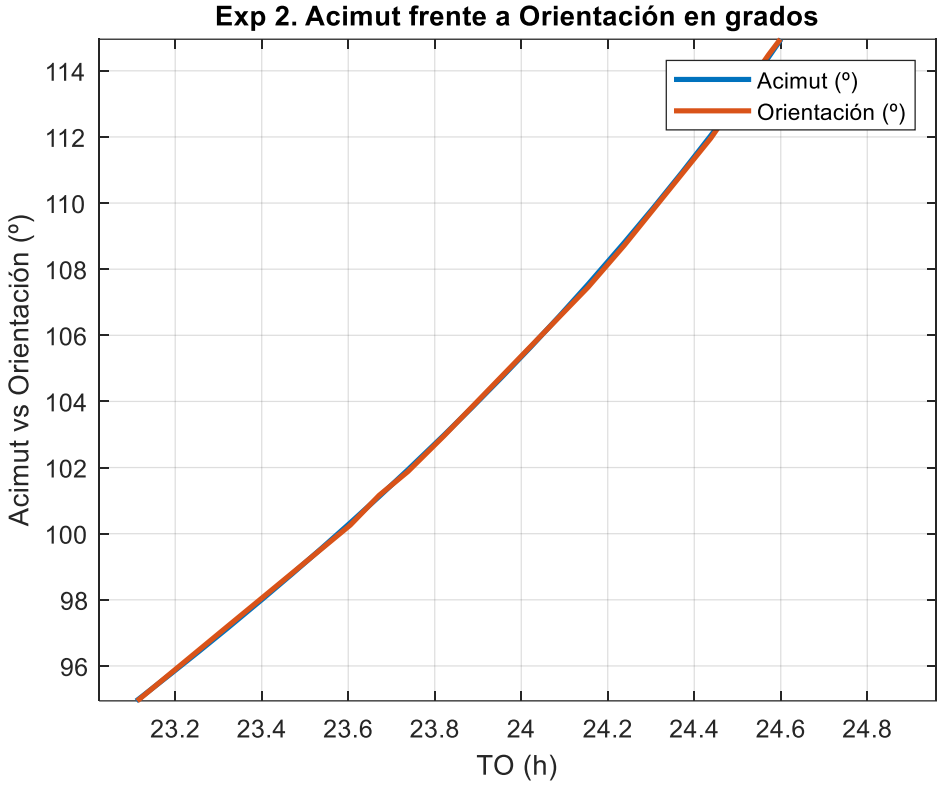


Figura 5-8: Detalle comparación entre acimut y orientación

Vemos que obtenemos resultados muy similares al caso anterior. Llegando en este caso a una posición mucho más extrema, llegando a aproximadamente la 1:20 h.

5.4 Experimento 3.

En este caso se buscaba observar la evolución del ángulo cenital. El sistema se orienta mirando hacia el Sur.

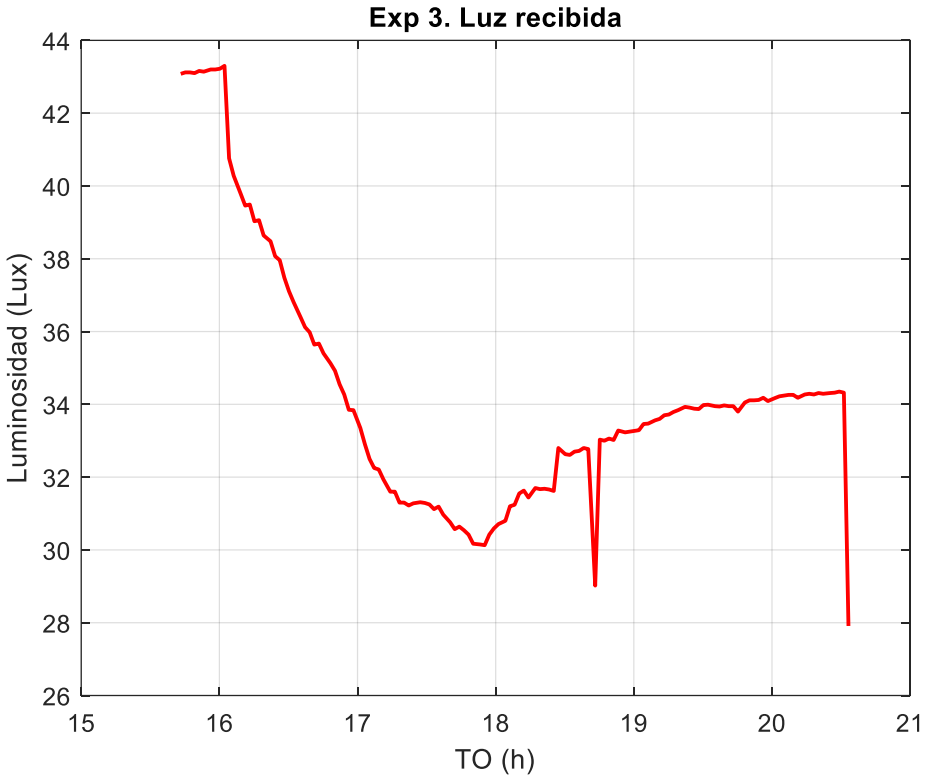


Figura 5-9: Luz recibida en el experimento 3

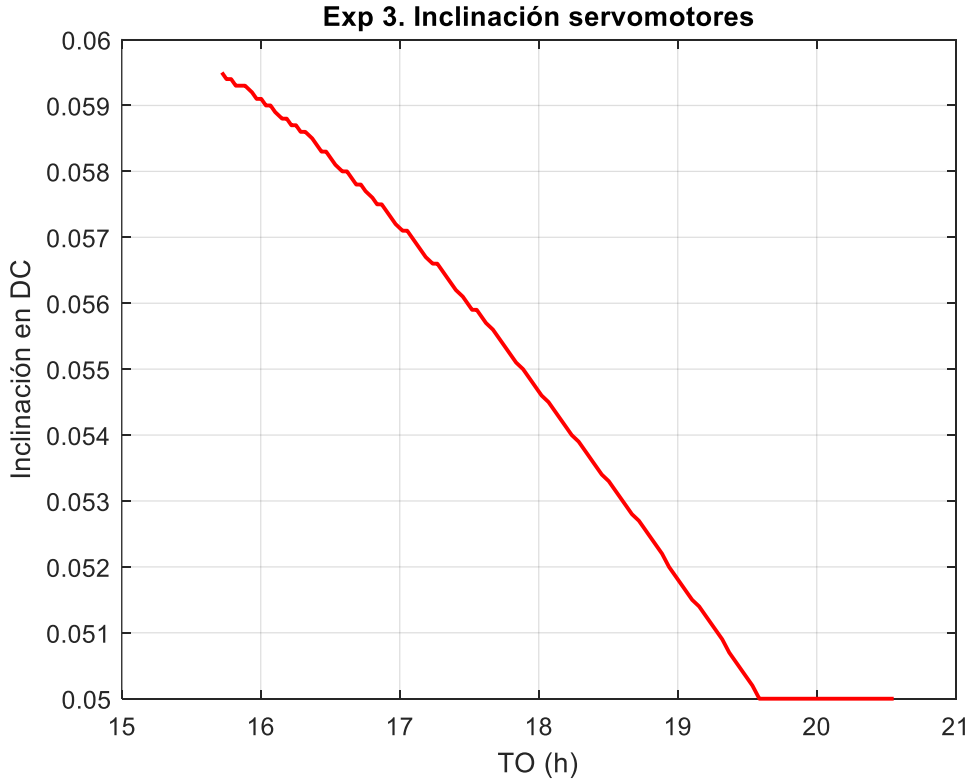


Figura 5-10: Inclinación del servomotor en DC

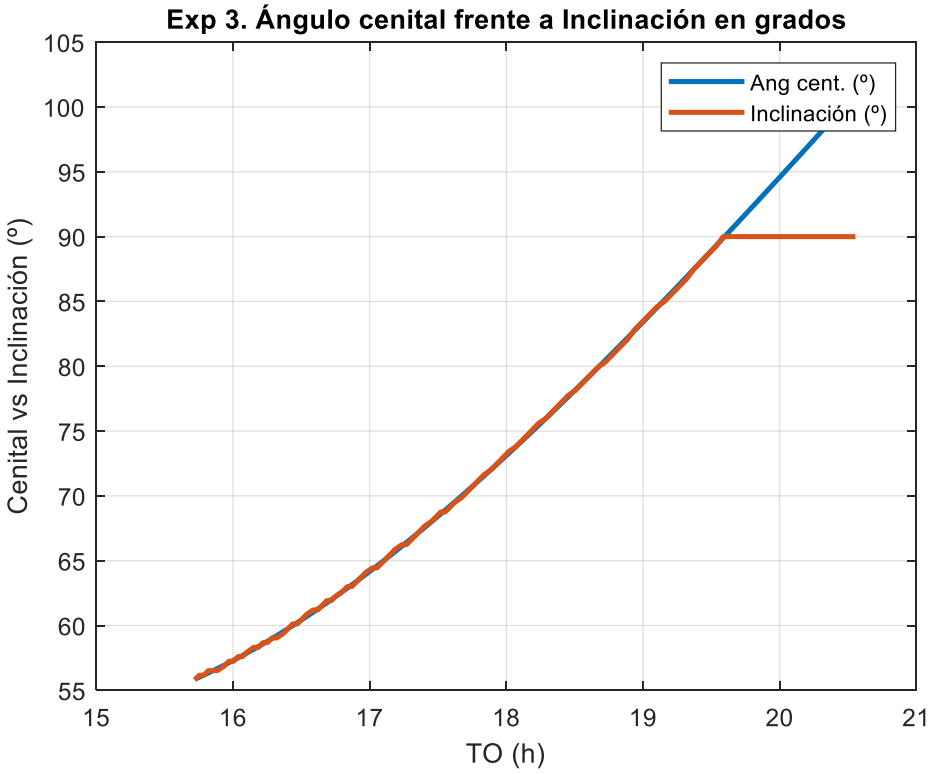


Figura 5-11: Evolución de la inclinación frente al ángulo cenital

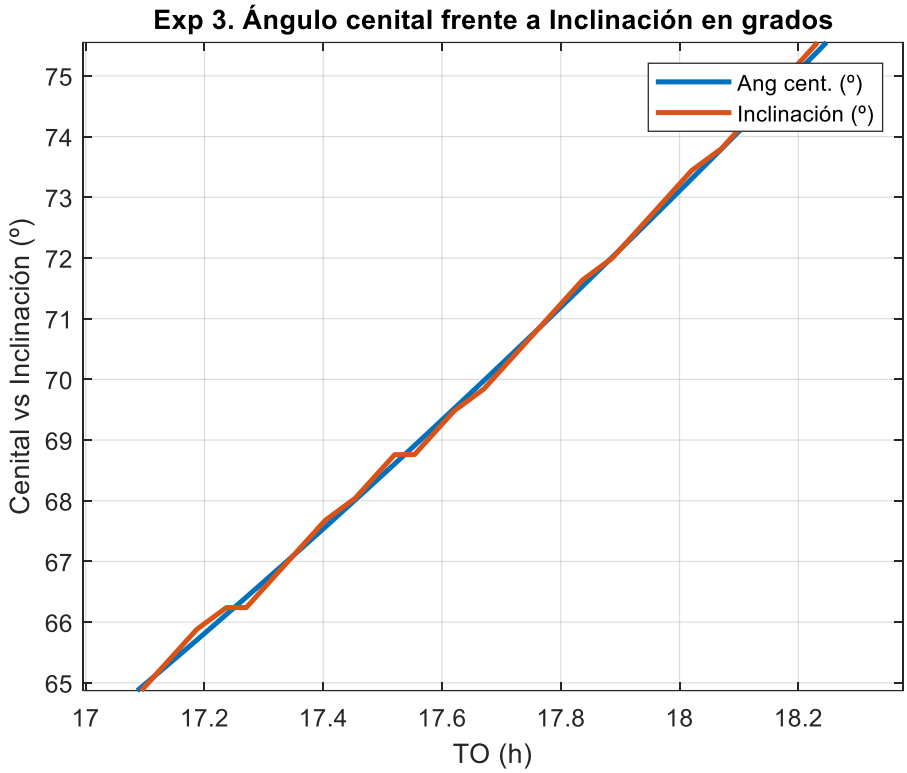


Figura 5-12: Detalle comparativa entre ángulo cenital e inclinación

En este experimento se ve que la diferencia entre inclinación y orientación, al alcanzarse los 90º, panel en vertical, el sistema satura el ángulo de inclinación, y si bien el ángulo cenital puede seguir tomando valores hasta 180º (valores que serían seguidos si se estuviera en el hemisferio sur) el panel no lo seguirá. Además se puede observar que hay mayores diferencias entre ángulo cenital e inclinación, esto se puede deber a que la variación del ángulo cenital a lo largo de las horas de Sol es menor que la del acimut y al aplicar el redondeo al

DC se consiguen valores con muy ligeras variaciones los unos de los otros, valores que, sin embargo, no se verán reflejados en la práctica al ser demasiado pequeños para que el servomotor se mueva.

6 CONCLUSIONES

Tal y como se ha desarrollado a lo largo de la teoría y demostrado con los experimentos, el sistema alcanza su objetivo de buscar la ubicación precisa del Sol en el cielo a partir de una cantidad limitada de datos, 4 en concreto, latitud, longitud, fecha y hora del día.

Con esto se cumplen las necesidades del bucle abierto para la aplicación sobre paneles solares reales para incrementar el rendimiento en pequeñas instalaciones. Añadir, que el sistema dependiendo únicamente de coordenadas geográficas y tiempo y sus limitadas necesidades, puede adaptarse a cualquier punto en la Tierra.

Entre las limitaciones que presenta el sistema, es necesario destacar que por los límites de los servomotores, la inclinación del panel queda totalmente caracterizada, no así la orientación que se limita a un arco máximo de 180 º en un único sentido, por lo que para su uso en bucle cerrado sobre un panel (o pequeña granja solar) será necesario replantear la obtención de la orientación para tener disponible todo el arco de giro.

Señalar, por otra parte, que el sistema trabaja en intervalos de tiempo de 10 s frente al caso práctico en el que se ejecutarían las operaciones en intervalos de minutos dada la dinámica del movimiento del Sol, por lo que si bien no se ha considerado, el sistema podría trabajar en modo de muy bajo consumo por lo que sería ideal para las aplicaciones a las que va orientado.

ÍNDICE DE FIGURAS

REFERENCIAS

1. **REE.** *Informe del Sistema Eléctrico Español 2020.* Alcobendas (Madrid) : Red Eléctrica de España, 2020.

2. **Lamigueiro, Oscar Perpiñán.** *Energía Solar Fotovoltaica.* 2011.

3. **TowerPro.** [En línea] [Citado el: 7 de Julio de 2021.] http://www.ee.ic.ac.uk/pcheung/teaching/DE1_EE/stores/sg90_datasheet.pdf.

4. **MicroPython - Python for microcontrollers. [En línea] 2014. [Citado el: 4 de Julio de 2021.] https://micropython.org/.**

5. Galán, Ricardo, y otros. ***Innovación conocimiento & Modelo de cuantificación del potencial fotovoltaico de España.*** **2015. 9788460843672.**

6. Tejera, Sara Moreno. ***Radiación Solar.***

7. Pöhlmann, Thomas. GitHub - perryrh0dan/opt3001: Python support library for the light sensor OPT3001 from Texas Instruments. [En línea] 12 de Septiembre de 2020. [Citado el: 4 de Julio de 2021.] https://github.com/perryrh0dan/opt3001.

8. Bessat, Matthieu. BMI160-i2c · PyPI. [En línea] [Citado el: 4 de Julio de 2021.] https://pypi.org/project/BMI160-i2c/.

9. KZSLAB: Kaizen Solutions. GitHub - KZSLAB/BMI160_Arduino_Library: An Arduino Library for the BMI160, a 6-Axis Sensor from BOSCH. [En línea] 13 de Abril de 2017. [Citado el: 4 de Julio de 2021.] https://github.com/KZSLAB/BMI160_Arduino_Library.

10. REE. ***El Sistema Eléctrico Español 2015.*** **Alcobendas (Madrid) : Red Eléctrica de España, 2015.**

11. —. ***El Sistema Eléctrico Español 2016.*** **Alcobendas (Madrid) : Red Eléctrica de España, 2016.**

12. —. ***El sistema electrico español 2017.*** **Alcobendas (Madrid) : Red Eléctrica de España, 2017.**

13. —. ***Informe del Sistema Eléctrico Español 2018.*** **Alcobendas (Madrid) : Red Eléctrica de España, 2018.**

14. —. ***Informe del Sistema Eléctrico Español 2019.*** **Alcobendas (Madrid) : Red Eléctrica de España, 2019.**

15. Pycom. Pycom_002_Specsheets_WiPy3.0_v2.pdf. [En línea] 19 de Diciembre de 2017. [Citado el: 7 de Julio de 2021.] https://docs.pycom.io/gitbook/assets/specsheets/Pycom_002_Specsheets_WiPy3.0_v2.pdf.

16. —. Introduction. [En línea] [Citado el: 4 de Julio de 2021.] https://docs.pycom.io/firmwareapi/.

17. —. expansionBoard3Specsheet180412. [En línea] 12 de Abril de 18. [Citado el: 4 de Julio de 2021.] https://docs.pycom.io/gitbook/assets/expansion3-specsheet-1.pdf.

18. Texas Instruments Incorporated. BOOSTXL-SENSORS Sensors BoosterPack Plug-in Module User's Guide BOOSTXL-SENSORS Sensors BoosterPack Plug-in Module. [En línea] Mayo de 2018. [Citado el: 4 de Julio de 2021.] https://www.ti.com/lit/ug/slau666b/slau666b.pdf.

19. Andrés, Pilar. 3.-Latitud y longitud. (α) ángulo de latitud; (β) ángulo de longitud.... | Download Scientific Diagram. *Evaluación y prevención de riesgos Ambientales en Centroamérica.* Girona : Documenta Universitaria, 2008, págs. 339-356.

20. Cerón, Hugo. Sistema de coordenadas astronómicas – Konrad Lorenz. [En línea] Fundación Universitaria Konrad Lorenz. [Citado el: 4 de Julio de 2021.] https://web.konradlorenz.edu.co/en/blog/sistema-de-coordenadas-astronomicas/.

21. Díaz, Vicente. Astronomia de posicion. *El cielo del mes.* [En línea] , de de 2007. [Citado el: 4 de Julio de 2021.] http://www.elcielodelmes.com/Curso_iniciacion/curso_1.php.

22. Germán Cordero, Raúl. ¿Como debe ser la orientación e inclinación de placas Solares? [En línea] SFE SOLAR LOGISTIC S.L, 7 de Agosto de 2019. [Citado el: 4 de Julio de 2021.] https://www.sfe-solar.com/noticias/articulos/como-varia-la-captacion-de-energia-solar-en-superficies-inclinadas/.

23. Mertens, Konrad. *Photovoltaics: Fundamentals, Technology, and Practice - Konrad Mertens - Google Libros.* Chichester : John Wiley & Sons Ltd, 2019. ISBN: 978-1-119-40104-9.

24. Li, Danny H W y Lam, Tony N T. Determining the Optimum Tilt Angle and Orientation for Solar Energy Collection Based on Measured Solar Radiance Data. [En línea] 28 de Noviembre de 2007. [Citado el: 4 de Julio de 2021.] https://downloads.hindawi.com/journals/ijp/2007/085402.pdf.

25. Altaruru. Python - Localizar IP. [En línea] 25 de Agosto de 2019. [Citado el: 4 de Julio de 2021.] https://www.altaruru.com/python-localizar-ip/.

26. Alonso Lorenzo, José Alfonso. Radiación, Irradiancia, Azimut y Hora sol Pico en Fotovoltaica. [En línea] SFE SOLAR LOGISTIC S.L, 5 de Junio de 2018. [Citado el: 4 de Julio de 2021.] https://www.sfe-solar.com/noticias/articulos/energia-fotovoltaica-radiacion-geometria-recorrido-optico-irradiancia-y-hsp/.

27. Texas Instruments Incorporated. OPT3001 Ambient Light Sensor (ALS) Datasheet (Rev. C). [En línea] Noviembre de 2017. [Citado el: 4 de Julio de 2021.] https://www.ti.com/lit/ds/symlink/opt3001.pdf?ts=1625416599497&ref_url=https%253A%252F%252Fwww.ti.com%252Fproduct%252FOPT3001.

28. Bosch Sensortec GmbH. BMM150 Datasheet. [En línea] Abril de 2020. [Citado el: 4 de Julio de 2021.] https://www.bosch-sensortec.com/media/boschsensortec/downloads/datasheets/bst-bmm150-ds001.pdf.

29. —. BMI160 Datasheet. [En línea] Noviembre de 2020. [Citado el: 4 de Julio de 2021.] https://www.bosch-sensortec.com/media/boschsensortec/downloads/datasheets/bst-bmi160-ds000.pdf.

ANEXO I. FIRMWARE BMM150

Parte I. Inicialización y activación del magnetómetro y su interfaz

```
# ***Activamos la interfaz del magnetómetro en modo 'normal power'***
self._reg_write(registers.CMD, registers.BMI160_CMD_MAG_MODE_NORMAL)
sleep_ms(2)
# ***Secuencia para activar el registro de pull-up***
self._reg_write(registers.FOC_CONF, registers.BMI160_FOC_CONF_DEFAULT)
self._reg_write(registers.CMD, registers.BMI160_EN_PULL_UP_REG_1)
self._reg_write(registers.CMD, registers.BMI160_EN_PULL_UP_REG_2)
self._reg_write(registers.CMD, registers.BMI160_EN_PULL_UP_REG_3)
self._reg_write(registers.BMI160_7F,
registers.BMI160_EN_PULL_UP_REG_4)

self._reg_write_bits(registers.BMI160_RA_MAG_X_H, 2, 4, 2)
self._reg_write(registers.BMI160_7F,
registers.BMI160_EN_PULL_UP_REG_5)

# Establecemos la dirección I2C del magnetómetro a 0x13
self._reg_write(registers.BMI160_MAG_IF_0,
registers.BMM150_BASED_I2C_ADDR)

# Activar el modo configuración, establecer el offset en la lectura a
la salida al max. y la longitud a 8
self._reg_write(registers.BMI160_MAG_IF_1,
registers.BMI160_MAG_MAN_EN)

# Activar la interfaz del magnetómetro
self._reg_write_bits(registers.BMI160_IF_CONF, 2, 4, 2)

        # ***Configurar el BMM150***
        # Activar el Sleep mode del BMM150
self._reg_write(registers.BMI160_MAG_IF_4,
registers.BMM150_EN_SLEEP_MODE)
self._reg_write(registers.BMI160_MAG_IF_3, registers.BMM150_POWER_REG)
sleep_ms(1)

        # Establecer las repeticiones para los ejes X e Y
self._reg_write(registers.BMI160_MAG_IF_4,
registers.BMM150_XY_REPETITIONS)
self._reg_write(registers.BMI160_MAG_IF_3,
registers.BMM150_XY_REP_REG)

        # Establecer las repeticiones para el eje Z
self._reg_write(registers.BMI160_MAG_IF_4,
registers.BMM150_Z_REPETITIONS)
self._reg_write(registers.BMI160_MAG_IF_3, registers.BMM150_Z_REP_REG)
```

```
          # ***Configurar la interfaz del MAG para el modo Datos***
          #/* Configure MAG write address and data to force mode of
BMM150 */
self._reg_write(registers.BMI160_MAG_IF_4,
registers.BMM150_OPMODE_REG_DEFAULT)
self._reg_write(registers.BMI160_MAG_IF_3,
registers.BMM150_OPMODE_REG)
          #/* Configure MAG interface data rate (25Hz) */
self._reg_write(registers.BMI160_MAG_IF_2,
registers.BMM150_R_DATA_ADDR)
          #/* Configure MAG read data address */
self._reg_write(registers.BMI160_RA_MAG_CONF,
registers.BMI160_MAG_CONF_25Hz)
          #/* Enable MAG data mode */
self._reg_write_bits(registers.BMI160_MAG_IF_1, 0, 7, 1)
          #/* Wait for power-up to complete */
while(0x1 != self._reg_read_bits(registers.PMU_STATUS,
definitions.BMI160_MAG_PMU_STATUS_BIT,
definitions.BMI160_MAG_PMU_STATUS_LEN)):
sleep_ms(1)
```

Parte II. Establecimiento de la tasa de muestreo

```
def getMagRate(self):
          return self._reg_read_bits(registers.BMI160_RA_MAG_CONF,
definitions.BMI160_MAG_RATE_SEL_BIT,
definitions.BMI160_MAG_RATE_SEL_LEN)
      #/** Set magnetometer output data rate.
      # * @param rate New output data rate
      # * @see getMagRate()
      # * @see BMI160_MAG_RATE_25HZ
      # * @see BMI160_RA_MAG_CONF
      # */
      def setMagRate(self, rate):
          self._reg_write_bits(registers.BMI160_RA_MAG_CONF, rate,
definitions.BMI160_MAG_RATE_SEL_BIT,
definitions.BMI160_MAG_RATE_SEL_LEN)
```

Parte III. Funciones de toma de medidas

```
def getMagneto(self):
          raw = self._regs_read(registers.BMI160_RA_MAG_X_L, 6)
          vals = unpack('<3h', bytes(raw))
          return (vals[0], vals[1], vals[2])

      #/** Get X-axis magnetometer reading.
      # * @return X-axis magnetometer measurement in 16-bit 2's
complement format
```

```
def getMagnetoX(self):
          raw = self._regs_read(registers.BMI160_RA_MAG_X_L, 2)
          vals = unpack('<h', bytes(raw))
          return vals

      #/** Get Y-axis magnetometer reading.
      # * @return Y-axis magnetometer measurement in 16-bit 2's
complement format
def getMagnetoY(self):
          raw = self._regs_read(registers.BMI160_RA_MAG_Y_L, 2)
          vals = unpack('<h', bytes(raw))
          return vals


      #/** Get Z-axis magnetometer reading.
      # * @return Z-axis magnetometer measurement in 16-bit 2's
complement format
def getMagnetoZ(self):
          raw = self._regs_read(registers.BMI160_RA_MAG_Z_L, 2)
          vals = unpack('<h', bytes(raw))
         return vals
```

Parte IV. Reescritura de las funciones de lectura/escritura del BMI160/BMM150

```
class BMI160_I2C(BMI160):
    def __init__(self, i2c, addr=0x69):
        self.i2c = i2c
        self.addr = addr
        super().__init__()

    def _reg_write(self, reg, data):
        self.i2c.writeto(self.addr, bytes([reg, data]))

    def _reg_read(self, reg):
        return self._regs_read(reg, 1)[0]

    def _regs_read(self, reg, n):
        self.i2c.writeto(self.addr, bytes([reg]))
        sleep_us(2)
        return self.i2c.readfrom(self.addr, n)
```

ANEXO II. CÓDIGOS SISTEMA

Ecuaciones.py

```
signo = lambda x : copysign(1, x)


def angulo_mag(Y,X):
      ang = atan2(Y,X) # Orientación según el magnetómetro
      if (ang < 0):
          ang = ang + 2*pi
      # Cambiamos la referencia para que sea el Sur tal y como se
define la orientacion
      if(ang > pi):
          orientacion = -(ang - pi)
      else:
          orientacion = pi - ang
      return orientacion

# Ecuación del tiempo aproximada
  ef Ec_tiempo(n):
      G = (2*pi*(n - 1))/365 # Ángulo diario en radianes de la orbita
terrestre

      # Ecuación del tiempo expresada en minutos
      Et_m = 229.2/60*(0.000075 + 0.001868*cos(G) - 0.032077*sin(G)\
      - 0.014615*cos(2*G) - 0.04089*sin(2*G))

      Et_h = Et_m/60 # Ecuación del tiempo expresada en horas
      return Et_h

  def omega(n, TO, AO, gamma, gamma_S, E_t):
      TSV = TO - AO + (gamma - gamma_S)/15 + E_t - 12
      ang_hor = 15*TSV
      return ang_hor

  def delta(n):
      w_t = 360/365*(n - 79.436)

      # Aproximación de la declinación
      delta = 0.3723 + 23.2567*sin(w_t) - 0.7580*cos(w_t)\
      + 0.1149*sin(w_t) + 0.3656*cos(w_t) - 0.1712*sin(w_t)\
      + 0.0201*cos(w_t)
      return delta

  # Función que obtiene el ángulo cenital
  def theta_z(delta, omega, phi):
      cos_3 = cos(delta)*cos(omega)*cos(phi)
      sen_2 = sin(delta)*sin(phi)
      theta_z = acos(cos_3 + sen_2)
      return theta_z
```

```
# Función que obtiene el acimut
def psi(delta, omega, phi, theta):
    CCS = cos(delta)*cos(omega)*sin(phi)
    CS = cos(phi)*sin(delta)

    cos_psi = signo(theta)*(CCS - CS)/sin(theta)
    psi = acos(cos_psi)
    return psi
```

Main.py

```
 from machine import Pin, I2C, PWM, SD
 from math import pi as PI
 from network import WLAN
 import ujson
 import time
 import os

 # Clases de los sensores y funciones matemática solar y otras
 from driver import BMI160_I2C
 from driver import driv
 from driver.struct import unpack
 import Ecuaciones
 from urllib.urequest import urlopen

 # Constantes dadas
 LONG_M = const(0) # Longitud del meridiano solar (en casi toda España
el 0)

 # Inicializaciones
 pwm1 = PWM(0, frequency=50)
 pwm2 = PWM(1, frequency=50)
 # Inicializamos el bus I2C con los pines (P9=SDA, P10=SCL), que son
sus pines por defecto
 i2c = I2C(1, pins=('P9','P10'))
 i2c.init(I2C.MASTER, baudrate=20000)

 # Inicializamos la Wipy en modo estación
 wlan = WLAN(mode=WLAN.STA)

 # Inicializamos los sensores
 bmi160 = BMI160_I2C(i2c)
 opt = driv.OPT3001(i2c)
 opt.write_config_reg(driv.I2C_LS_CONFIG_CONT_FULL_800MS)

 sd = SD()
 os.mount(sd, '/sd')

 def ipInfo(addr=""):
     if addr == "":
        url = "https://ipinfo.io/json"
     else:
        url = "https://ipinfo.io/" + addr + "/json"

    response = urlopen(url)
     data = ujson.load(response)
     return data

 # try some standard file operations
 f = open('/sd/medidas.txt', 'w')
 f.write('Medidas de los sensores:')
 f.close()
 time.sleep_ms(2)
 f = open('/sd/angulos.txt', 'w')
```

```
  f.write('Ángulos calculados')
  f.close()
  time.sleep_ms(2)
  f = open('/sd/motor.txt', 'w')
  f.write('Señal en los motores (en angulo y DC)')
  f.close()

  # Nos conectamos a la red
  time.sleep_ms(2)
  wlan.disconnect()
  time.sleep_ms(2)
  wlan.connect(ssid='View', auth=(WLAN.WPA2, '345fa47ba656'))
  time.sleep(1)

  # Actualizamos el RTC a la hora de la red
  rtc = machine.RTC()
  rtc.ntp_sync("pool.ntp.org")
  time.sleep_ms(3)
  while not rtc.synced():
      machine.idle()
  print("Son las {3}:{4}:{5} del {2} de {1} del
{0}".format(*time.localtime()))
  n = time.localtime()[7] # n° del día


  # Localización del sistema
  LOC = ipInfo("85.137.11.21")['loc']
  # Buscamos la posición de la coma y desde esta obtenemos latitud y
longitud
  coma = LOC.index(",")
  LAT = LOC[0:coma]
  LON = LOC[coma+1:]

  phi     = float(LAT) # Pasamos latitud y longitud de str a float
  lamda   = float(LON)
  print("El sistema se encuentra en la latitud: " + str(phi) + " y
longitud: " + str(lamda))


  # Constantes, lo serán para el resto del día.
  Et   = Ecuaciones.Ec_tiempo(n)  # Ecuación del tiempo de hoy
  decl = Ecuaciones.delta(n)      # Declinación de hoy

  print("Ecuación del tiempo del día " + str(n) + "/365: " + str(Et) +
" h")
  print("Declinación el día " + str(n) + "/365: " + str(decl) + "°")

  pwm_ori = pwm1.channel(0, pin='P19', duty_cycle=0.105)
  pwm_incl = pwm2.channel(1, pin='P6', duty_cycle=0.05)

  time.sleep(1)
  while(True):
  # Medimos por los sensores
    marca_t = (time.localtime()[3], time.localtime()[4],
time.localtime()[3])
  mx, my, mz = bmi160.getMagneto()
```

```
  mag = (mx, my, mz)
  Luz = opt.read_lux_float()
  print("Son las {0}:{1}:{2}".format(*marca_t))
  print("mx = {0:>8} uT; my = {1:>8} uT; mz = {2:>8} uT".format(*mag))
  print("Iluminación: ", + Luz, "lux")
  gamma_0 = Ecuaciones.angulo_mag(my, mx)
 f = open('/sd/medidas.txt', 'a') # Guardamos medidas
  f.write('\n' + '{0}:{1}:{2}'.format(*marca_t) +  ' ' + '{0:>8} {1:>8}
{2:>8}'.format(*mag) + ' ' + str(Luz))
  f.close()

  # Calculamos las horas solares y la orientacion del sistema
  TO = marca_t[0] + marca_t[1]/60 + marca_t[2]/3600
  hora_sol = Ecuaciones.omega(n, TO, 0, lamda, LONG_M, Et)
  gamma_0 = Ecuaciones.angulo_mag(my, mx)
  print("Orientación del sistema (respecto al Sur): " +
str(gamma_0*180/PI))
  print("Hora solar actual: " + str(hora_sol) + "°")

  # Se calculan el ángulo cenital y acimut
  cenital = Ecuaciones.theta_z(decl*PI/180, hora_sol*PI/180,
phi*PI/180)
  acimut = Ecuaciones.psi(decl*PI/180, hora_sol*PI/180, phi*PI/180,
cenital)

  angulos_solares = (hora_sol, decl, acimut*180/PI, cenital*180/PI)
  print("Angulos solares:")
  print("Hora solar = {0:.3f}°; declinación = {1:.3f}°; acimut =
{2:.3f}°; cenital = {3:.3f}°".format(*angulos_solares))

  # Ajustamos los ángulos al rango de los servos,
  # considerando una posición más o menos cte.
  gamma = 0.1 - acimut*(0.1)/(PI)
  gamma = round(gamma, 4)
  beta = 0.075 - cenital*(0.075 - 0.05)/(PI/2)
  beta = round(beta, 4)

  # Redondeamos para el ajuste a los valores de los servos
  if (gamma > 0.1):
      gamma = 0.1
  elif(gamma < 0.03):
      gamma = 0.03

  if (beta > 0.075):
      beta = 0.075
  elif(beta < 0.05):
      beta = 0.05

  pwm_ori.duty_cycle(gamma)
  pwm_incl.duty_cycle(beta)

# Guardamos en la SD
  f = open('/sd/angulos.txt', 'a')
  f.write('\n' + '{0}:{1}:{2}'.format(*marca_t) +  ' ' + '{0:.3f}
{1:.3f} {2:.3f} {3:.3f}'.format(*angulos_solares))
  f.close()
  f = open('/sd/motor.txt', 'a')
```

```
  f.write('\n' + '{0}:{1}:{2}'.format(*marca_t) +  ' '  +
str(angulos_solares[2]) + ' ' + str(gamma) + ' '\
  + str(angulos_solares[3]) + ' ' + str(beta))
  f.close()
   time.sleep(10)
```