Proyecto Fin de Máster
Máster en Ingeniería Electrónica, Robótica y Automática

Direccionamiento y control de gimbal para seguimiento solar en una planta solar térmica

Autora: Irene Luque Martínez
Tutor: José Ramón Domínguez Frejo

Dpto. de Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2022

Proyecto Fin de Máster
Máster en Ingeniería Electrónica, Robótica y Automática

Direccionamiento y control de gimbal para seguimiento solar en una planta solar térmica

Autora:
Irene Luque Martínez

Tutor:
Jose Ramón Domínguez Frejo
Juan de la Cierva-Incorporación

Dpto. de Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2022

Proyecto Fin de Máster: Direccionamiento y control de gimbal para seguimiento solar en una planta solar térmica

Autora: Irene Luque Martínez

Tutor: José Ramón Domínguez Frejo

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

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2022

El Secretario del Tribunal

A Malena, Paco, Andrea, Manuel

Agradecimientos

Sin duda agradecer a Eduardo, primeramente, la oportunidad de formar parte de este equipo y la confianza depositada desde el primer momento. Trabajar en un entorno como el que existe en este grupo es un gusto. Gracias a José Ramón, por la asistencia, la paciencia y la ayuda constantes. Y por supuesto gracias a todas las personas que han aportado algo a este proyecto de cualquier manera, tanto a nivel técnico como a nivel anímico; hubiera sido muy difícil sacarlo adelante sin ello.

Irene Luque Martínez
Máster en Ingeniería Electrónica, Robótica y Automática

Sevilla, 2022

Resumen

El siguiente proyecto trata sobre el control de un Pan&Tilt, robot capaz de rotar alrededor de tres ejes y que porta, en nuestro caso, un sensor para la toma de medidas solares en su extremo. El desarrollo del software para el seguimiento solar está enfocado a dotar al robot de independencia y autonomía en la tarea de la orientación del sensor de medición al punto de mayor energía solar a tiempo real, para realizar aquí una toma de medidas y poder así obtener el valor de la irradiancia solar.

En un contexto global, este sistema móvil enviará esta información al resto de componentes de una planta termosolar, que la utilizarán para ejecutar los sistemas de control diseñados y optimizar el rendimiento global de la planta.

Para llevar a cabo el proyecto se han desarrollado códigos de cálculos solares, ubicación, movimiento y medición solar a través de Arduino. Mediante el uso de varias placas y diferentes elementos hardware trabajando al mismo tiempo se creará una red de transmisión de la información, de manera que el sistema realizará el seguimiento a tiempo real.

Abstract

The following project deals with the control of a Pan&Tilt, a robot capable of rotating around three axes and carrying, in our case, a solar measurement sensor at its end. The development of the Software for solar tracking is focused on providing the robot with independence and autonomy in the task of orienting the sensor to the point of highest solar energy in real time, in order to take measurements here and thus obtain the value of the solar irradiance.

In a global context, this mobile system will send this information to the rest of the components of a solar thermal plant, which will use it to run the designed control systems and optimize the overall performance of the plant.

To carry out the project, solar calculation, location, movement and solar measurement codes have been developed through Arduino. By using several boards and different hardware elements working at the same time, a network of information transmission will be created, so that the system will monitor in real time.

Índice

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

Notación

API	*Application Programming Interface*
DNI	*Direct Normal Irradiance*
HW	Hardware
IDE	Entorno de Desarrollo Integrado
SM	Solar Mems
SW	Software
TES	*Thermal Energy Storage*
TFM	Trabajo de Fin de Máster
:	Tal que
=	Igual
<=	Menor o igual
<	Menor
>=	Mayor o igual
>	Mayor o igual
\	Backslash

1 INTRODUCCIÓN

EL presente proyecto trata sobre el diseño y desarrollo de un Pan&Tilt capaz de orientarse y tomar medidas de radiación solar a tiempo real, de cara a su futura reproducción e instalación en una flota de robots móviles terrestres. Estos robots, que estarán distribuidos a lo largo de toda una planta solar térmica, serán capaces de orientarse de forma autónoma e independiente hacia los puntos de mayor interés en cada momento, tomar medidas y enviar estos datos, de forma que se obtenga una estimación distribuida de la irradiancia permitiendo así mejorar el comportamiento de la planta y obtener una mayor potencia eléctrica.

La investigación se ha desarrollado en el seno del proyecto OCONTSOLAR – Optimal Control of Thermal Solar Energy Systems -, correspondiente a una ERC Advanced Grant concedida por el European Research Council. El objetivo principal de OCONTSOLAR es la inclusión de flotas de sensores móviles (agentes) como parte integral de sistemas de control, y su posterior aplicación a un contexto de plantas solares térmicas con colectores cilindro-parabólicos.

Figura 1-1. Logo OCONTSOLAR

En este capítulo se desarrollará una introducción teórica sobre el proyecto desarrollado durante los últimos meses como componente del mencionado grupo de investigación (clases y tipos de plantas solares, así como la motivación, objetivos y planteamiento de este), para luego dar pie a una exposición detallada del desarrollo de la investigación.

1.1 Irradiancia solar

El Sol constituye una grandísima fuente de energía para el Sistema Solar y, concretamente, para el planeta Tierra, que absorbe del orden de $1{,}1 \cdot 10^{17}$ W del flujo de esta energía (1). El origen de la radiación electromagnética es la reacción de fusión nuclear que se produce en el interior del Sol, donde una gran cantidad de energía térmica es liberada a altísimas temperaturas (2). Dado que el Sol es un cuerpo incandescente, la radiación electromagnética se emite en un amplio rango de longitudes de onda, de las cuales solo algunas son perceptibles por el ojo humano.

Al conjunto total de ondas se le conoce como espectro solar. Además de las visibles por los humanos, que constituyen el 40% de las mismas, encontramos las infrarrojas (50%) y las ultravioletas (10%), tal y como se aprecia en la Figura 1. Dentro de la categoría de rayos Ultravioletas, encontramos los rayos UVA – atraviesan la atmósfera y llegan a la superficie terrestre- , los UVB – tienen dificultad para atravesar la atmósfera – y los UVC – quedan atrapados en la capa de ozono y nunca llegan a la superficie.

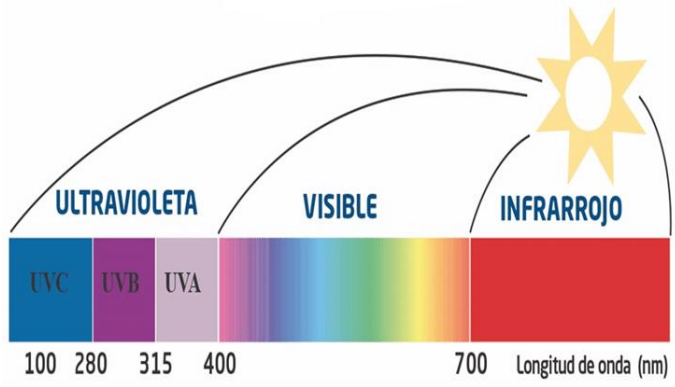


Figura 1-2. Espectro solar

De esta manera, concluimos que la radiación solar es la energía emitida por el Sol; conforma un concepto genérico que no se cuantifica con ninguna magnitud. Las magnitudes que describen la radiación solar que llega a la superficie terrestre por metro cuadrado son la irradiancia solar y la irradiación solar. La diferencia entre ellas radica en que la irradiancia solar mide la potencia en una superficie dada y en un instante concreto [W/m^2], mientras que la irradiación lo hace durante un tiempo [J/m^2 o Wh/m^2].

1.1.1 Componentes

Como resultado de su paso a través de la atmósfera, la distribución espectral de la radiación solar se ve afectada, a la vez que se reduce la magnitud de flujo radiante. Esto ocurre así como consecuencia de la absorción selectiva por las moléculas de los gases existentes en la atmósfera, de la difusión por moléculas de aire o partículas en suspensión y de la reflexión en las nubes. Así, este flujo radiante alcanza la atmósfera en diversas formas, que son las que se describen a continuación:

- ✓ **Radiación directa:** es aquella que proviene directamente del Sol y no sufre ningún cambio de dirección antes de llegar a la superficie terrestre. La irradiancia directa es igual a la irradiancia en el exterior de la atmósfera terrestre menos las pérdidas atmosféricas debidas a la absorción y dispersión. Esta será el objeto de estudio del presente proyecto.
- ✓ **Radiación difusa:** es el efecto generado cuando la radiación solar que alcanza la superficie atmosférica se dispersa de su dirección original, a causa de moléculas presentes en la atmósfera. Esta puede llegar a conformar un 15% de la radiación global en días soleados, siendo este porcentaje mucho mayor cuando el día es nublados.
- ✓ **Radiación reflejada:** es aquella que 'rebota' en la superficie terrestre, y es dependiente del coeficiente de reflexión o 'albedo'. Únicamente las superficies perpendiculares a la superficie terrestre son las que reciben esta radiación.
- ✓ **Radiación total:** está conformada por la suma de las tres irradiancias anteriores.

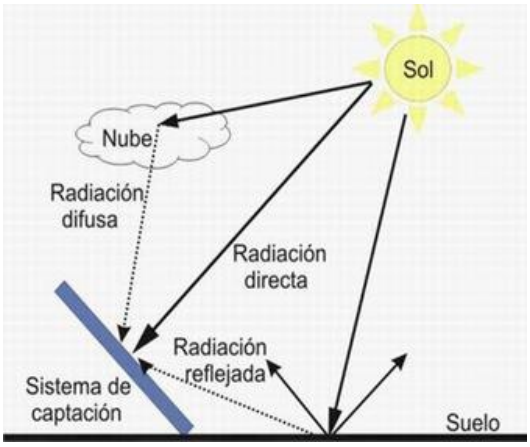


Figura 1-3. Tipos de radiación solar

1.1.2 Medición (sensores de radiación solar)

Atendiendo al principio físico en el que se basan, los sensores de radiación solar se pueden agrupar en diferentes tipos, entre los que destacan los sensores calorimétricos – la energía se transforma en flujo de calor, que es lo que se mide-, los sensores termomecánicos – el flujo se calcula a través de la magnitud de la flexión producida en una lámina bimetálica con coeficientes de dilatación térmica muy diferentes-, sensores termoeléctricos – basados en la idea de producir una fuerza electromotriz a través de dos alambres metálicos, manteniendo uno a la sombra y otro bajo el flujo radiante- y, por último, los sensores fotoeléctricos – se basan en la corriente eléctrica que se genera al incidir el flujo radiante en un elemento sensible compuesto por materiales semiconductores.

De esta forma, existen diferentes instrumentos para medir la radiación solar, que serán usados según el tipo de radiación a medir de las anteriormente descritas. Los principales medidores son los siguientes (3):

- ✓ **Piranómetros.** Los piranómetros son radiómetros diseñados para medir la irradiancia de una superficie plana, normalmente procedente de la radiación solar o de lámparas.
- ✓ **Pirgeómetros.** Los pirgeómetros están diseñados para efectuar mediciones de radiación IR (infrarroja); se utilizan tanto en la investigación atmosférica como en pruebas de materiales.
- ✓ **Albedómetros.** Los albedómetros son piranómetros dobles, capaces de medir la irradiancia solar, tanto global como reflejada, en un solo instrumento.
- ✓ **Pirheliómetros.** El pirheliómetro está concebido para efectuar mediciones de radiación solar directa, de incidencia normal, y sin supervisión.
- ✓ **Radiómetros UV** (con una respuesta espectral adaptada al espectro de acción eritemática), **Radiómetros Netos** (medir la radiación entrante y saliente, de onda corta -0.3 a 3 μm- y de onda larga -4.5 a >40 μm-), **Seguidores Solares**, etc.

Dado que el presente proyecto tiene como objetivo la medición de radiación normal directa, profundizaremos brevemente en el pirheliómetro, siendo este el instrumento encargado de seguir el movimiento del Sol y tomar mediciones de la energía que emite en cada momento (4). La luz solar entra en el instrumento a través de una ventana y es dirigida sobre una termopila, que convierte el calor en una señal eléctrica que se puede grabar, pudiendo obtener así la medida en W/m^2. Además, se utiliza junto con un sistema de seguimiento solar que nos permite mantener el instrumento orientado al sol.

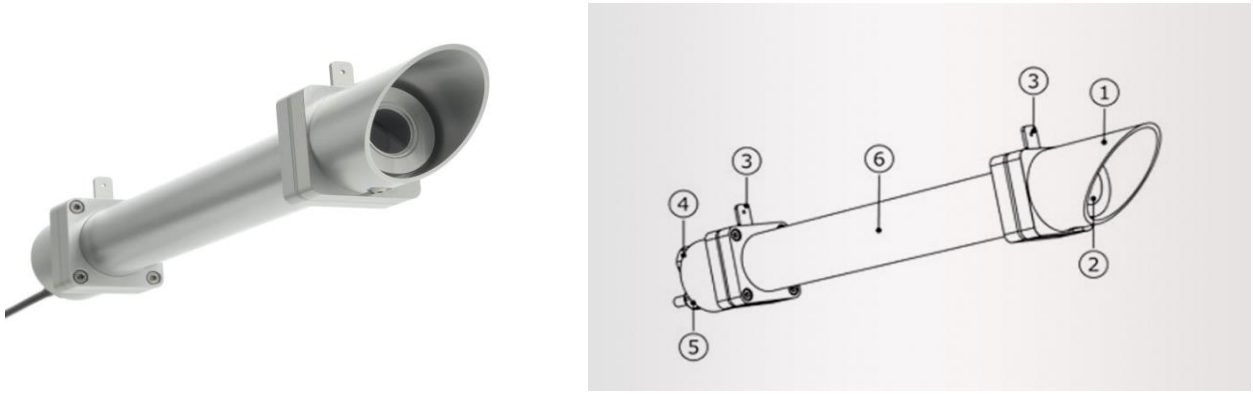


Figura 1-4: a) Pirheliómetro, b) Componentes del pirheliómetro: (6) cuerpo del instrumento, (4) tapa de protección, (5) ventana con calentador, (2) vista, (1) indicador de humedad, (7) cable

De esta manera, la Organización Meteorológica Mundial (OMM) ha estrablecido tres categorías diferentes para los pirheliómetros, atendiendo a factores como la sensibilidad o estabilidad de los factores de calibración:

- ✓ **Pirheliómetros absolutos:** son aquellos que se autocalibram, evaluándose la irradiación a partir de parámetros físicos del instrumento.
- ✓ **Pirheliómetros de referencia:** se calibran teniendo en cuenta el grupo de Patrones Mundiales en las comparaciones interpirheliométricas una vez cada cinco años.
- ✓ **Pirheliómetros secundarios:** son utilizados de manera operacional en las redes radiométricas; son más sencillos y robustos que los otros dos tipos, calibrándose por comparación con los patrones nacionales.

1.1.3 Sistemas y seguidores solares

Hoy en día, los beneficios que aportan las fuentes de energía renovables son ampliamente conocidos y reconocidos en cualquier ámbito y sector de la sociedad. Pese a que la energía termosolar lleva mucho tiempo siendo de gran importancia para el ser humano - antiguamente ya se utilizaban los rayos solares combinados con espejos para encender fuegos y calentar comidas-, no fue hasta la crisis del petróleo de la década de 1970 cuando la energía renovable empezó a considerarse una alternativa real a las fuentes de energía que hasta ese momento existían (5). Así, a partir de este momento la investigación y la inversión en ellas aumentaron considerablemente, pudiendo llegar a alcanzar el papel que tienen hoy en día en nuestras vidas y acarreando consigo una gran mejora en la eficiencia de las centrales termosolares y una disminución de costes. A continuación se realizará una exposición de las centrales termosolares, sus tipos y sus principales puntos de estudio, sirviendo de marco teórico para explicar los objetivos y pretensiones llevados a cabo durante este proyecto de investigación.

Una central termosolar es una instalación destinada a emplear la radiación procedente del Sol para generar energía eléctrica. Así, existen dos tipos de instalaciones que pueden cumplir este objetivo:

- ✓ **Sistema térmico-solar.** La radiación solar que incide en la superficie terrestre se utiliza para calentar un fluido. Tras las etapas iniciales en las que trabajan los equipos solares – tales como el receptor solar o el concentrador óptico- el proceso realiza la conversión mecánica del calor, calentando un flujo que más tarde moverá una turbina para generar la buscada energía eléctrica, a través del movimiento de un alternador. En la Figura 1-5 podemos encontrar un esquema del proceso descrito en un sistema de torre. Algunos de estos sistemas son las centrales solares térmicas con cilindros parabólicos, Heliostatos, receptores lineales de Fresnel, discos parabólicos de Stirling, etc., de lo se profundizará más adelante.
- ✓ **Sistema solar fotovoltaico.** Son los más conocidos, que nos aportan energía eléctrica gracias a los paneles fotovoltaicos que captan la energía luminosa del Sol y la transforman en eléctrica. Para realizar esta transformación, se emplean celdas fotovoltaicas formadas por metales sensibles a la luz y que desprenden electrones al recibirla, generando así la energía eléctrica (6).

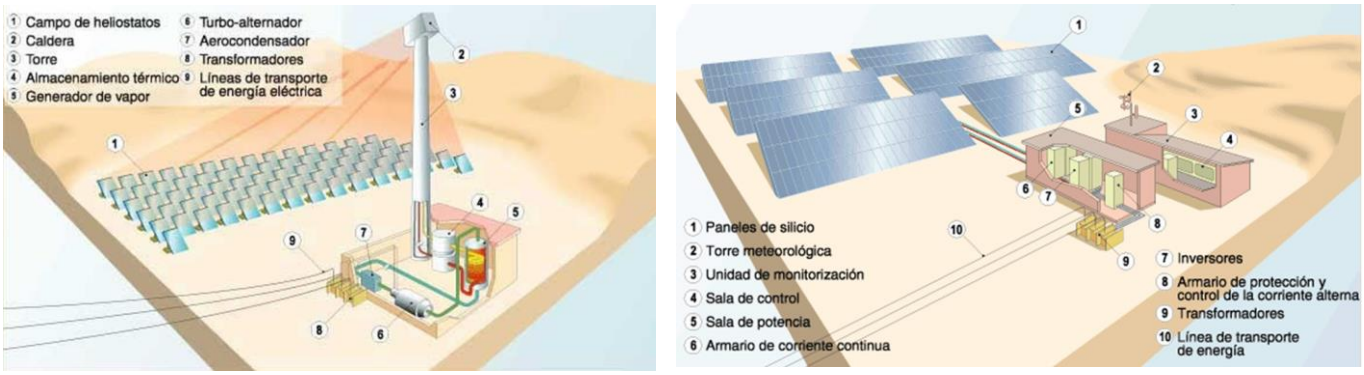


Figura 1-5: a) Central térmico-solar, b) Central solar-fotovoltaica

En algunas de estas últimas es interesante que los paneles solares se mantengan orientados hacia la dirección solar, con vistas a maximizar la producción de energía. Por otra parte, en las centrales térmico-solares, esta captación de energía se puede realizar por medio de espejos con orientación automática. Podemos entender cómo en ambos casos es importante y necesario un sistema de detección y seguimiento de la posicón del Sol.

Una vez las placas solares están situadas y expuestas al Sol, el ángulo de incidencia de estos rayos es el siguiente aspecto que determinará cuánta energía se genera. A mayor perpendicularidad del ángulo de incidencia, mayor energía podrá generar el panel fotovoltaico.

Así, según su capacidad de movimiento, también encontramos dos tipos de seguidores:

- ✓ **Seguidor de un eje.** El movimiento posible para el panel solar tiene un sólo grado de libertad; suele estar alineado Norte-Sur y permitir al panel el movimiento en la línea Este-Oeste. Son más económicos y simples, además de permitir una adaptación a cubiertas, pero evidentemente el seguimiento que pueden realizar será menos preciso.
- ✓ **Seguidor de dos ejes.** El movimiento del panel solar puede tener dos direcciones: Norte-Sur y Este-Oeste. La producción de energía a lo largo de todo el año se maximizará, ya que cambian la orientación no sólo a lo largo de un día, sino también teniendo en cuenta la época del año. El seguimiento solar será más preciso, acarreando esto un mayor coste del dispositivo.

En función del algoritmo utilizado para realizar dicho seguimiento también se distinguen varios tipos de seguidores solares:

- ✓ **Seguidores por punto luminoso.** Este tipo de seguidores cuentan con un sensor que les indica en todo momento cuál es el punto del cielo más luminoso, siendo por tanto hacia donde el medidor deberá orientarse de cara a obtener la máxima radiación. Este algoritmo genera una señal integrada por uno o varios sensores; dependiendo del contenido de dicha señal, se enviará un comando de control a los motores necesarios para conseguir el enfoque deseado.
- ✓ **Seguidores con programación astronómica.** En este segundo tipo de algoritmos, los seguidores llevan implementado un programa que conoce en qué punto debería estar el Sol en cada momento, de forma de puedan orientarse directamente allí. Es totalmente independiente de las condiciones meteorológicas, ya que no se dispone de sensores por punto luminoso, por lo que los cálculos se realizarán en base a las ecuaciones que predicen la ubicación del Sol a tiempo real.

Así, aunque la fotovoltaica es la más barata de las dos (es la fuente de energía más barata a día de hoy), la solar térmica cuenta con muchas otras ventajas. Es más eficiente que la fotovoltaica, con eficiencia de hasta el 70%, mientras que la fotovoltaica suele alcanzar el 21-22%. Además, la energía solar térmica se puede almacenar en un TES y producir durante la noche, además de producir un líquido a altas temperaturas (flujo), que se puede usar directamente en un proceso industrial.

1.1.4 Centrales termosolares

Una vez descritos los objetivos y el funcionamiento general de ellas, veremos que existen diversas maneras de llevar a cabo los principios expuestos (7):

- ✓ **Central termosolar de cilindros parabólicos.** Esta tecnología se basa en el uso de espejos cóncavos, montados en forma de canal, para dirigir los rayos del sol hacia el fluido que circula por un conducto situado en el foco de la parábola, tal y como se puede observar en la Figura 1-6. Estos espejos están construidos sobre una estructura que es capaz de girarlos, de manera que se pueda seguir la trayectoria solar de cara a obtener los rayos lo más perpendicularmente posible durante el máximo número de horas, optimizando así la obtención de energía. Este tipo de central es compleja, ya que la fabricación de los espejos cuenta con la dificultad de alcanzar la forma cóncava adecuada, primordial para obtener un aprovechamiento óptimo.

Figura 1-6. Central termosolar CPP

- ✓ **Central termosolar de Heliostatos.** Un heliostato es un conjunto de espejos que se mueven sobre dos ejes, normalmente en montura altacimutal – altura y acimut – con el objetivo de mantener el reflejo de los rayos solares que inciden sobre él en un determinado punto de su superficie (8). Las centrales termosolares formadas por heliostatos necesitan una gran superficie para ubicarlos, ya que además se precisa de una torre que será sobre la que se reflejen los rayos de Sol durante el día, alcanzando esta unas altísimas temperaturas. Se trata de una tecnología compleja, dado que cada espejo necesita un sistema electrónico independiente de control que calcule la trayectoria y posicionamiento del mismo, estando cada uno situado en un lugar diferente.

Figura 1-7. Planta solar de Heliostatos.

En la Figura 1-7 se puede observar el aspecto en la distancia de una planta de este tipo. En concreto, esa imagen pertenece a la Planta Solar PS10, la primera central térmica solar comercial de torre central y campo de heliostatos instalada en el mundo, en el año 2006, con una capacidad instalada de 11MW y una producción annual de 24GWh, ubicada en Sanlúcar la Mayor, Sevilla.

- ✓ **Central termosolar con receptores lineales Fresnel.** La gran diferencia con el resto de centrales expuestas radica en que el tipo de espejos empleados en este tipo de centrales son espejos planos. Esta característica reduce notablemente el precio de una central, siendo mucho más convencionales este tipo de espejos, aunque al final es coste por kWh producido es mayor hoy en día.

Figura 1-8. Central termosolar con receptores Fresnel

- ✓ **Central termosolar con discos parabólicos de Stirling.** En este caso, la central está formada por concentradores parabólicos independientes. En el foco existe un motor encargado de transformar la energía térmica recibida en un giro mecánico, y que a su vez lleva acoplado un generador eléctrico que convierte este giro mecánico en corriente eléctrica. De esta manera, cada unidad – concentrador – es capaz de producir energía eléctrica de forma independiente.

Figura 1-9. Receptor parabólico de Stirling

1.2 Antecedentes

Como se mencionó al inicio del documento, este proyecto se ha realizado en el seno de OCONTSOLAR, proyecto correspondiente a una Advance Grant con duración hasta febrero de 2024. OCONTSOLAR tiene como objetivo desarrollar nuevos métodos de control para utilizar sensores móviles montados en drones y vehículos terrestres no tripulados (UGV) como parte integral de los sistemas de control (9). Los sensores montados en vehículos han sido utilizados hasta la fecha para la vigilancia y la recopilación de información. Sin embargo, nunca como parte integral de los sistemas de control en el mundo de la energía solar térmica, siendo este el objetivo del proyecto.

Las plantas de energía solar se utilizarán como caso de estudio con el objetivo de optimizar su funcionamiento a través de estimaciones y predicciones de irradiancia espacial. Entre los principales objetivos del grupo se encuentran los siguientes: métodos para controlar flotas de sensores móviles e integrarlos como parte esencial de los sistemas de control global, métodos de estimación y predicción de la irradiancia solar distribuidos espacialmente utilizando una flota variable de sensores montados en drones y UGV o nuevos algoritmos de control predictivo (MPC) que usan estimaciones y predicciones de sensores solares móviles para lograr un funcionamiento más seguro y eficiente de las plantas. OCONTSOLAR incluye pruebas de concepto mediante experimentación en la Plataforma Solar de Almería, además de en una planta solar de climatización instalada en la institución anfitriona.

Figura 1-10. Plataforma Solar de Almería

Desde el inicio de este proyecto, en septiembre del 2018, han sido muchos los objetivos que se han cumplido y se han difundido en 27 artículos de revistas, 15 artículos de congresos, 4 capítulos de libros y 5 conferencias magistrales y plenarias en congresos. Muchos de ellos están relacionados con el control predictivo ((10), (11), (12)), con la estimación de irradiancia solar ((13), (14)) o incluso versan sobre la aplicación de los métodos de control desarrollados en otros ámbitos ((15), (16)). Dentro de un proyecto de esta envergadura es donde se sitúa la investigación presentada en este documento, que ha sido desarrollada mientras formaba parte de este grupo de investigación, teniendo así la información y documentación precedente disponible en todo momento.

1.3 Planteamiento y Objetivos

Como hemos visto, OCONTSOLAR es un proyecto de gran envergadura que abarca un sinfín de investigaciones de cara a conseguir sus objetivos finales. Un subobjetivo dentro de la investigación principal es aquel relacionado con el desarrollo de sensores móviles montados en vehículos terrestres no tripulados, con capacidad de orientación al Sol -cumpliendo todos los requisitos descritos en apartados anteriores- y toma de muestras de irradiancia, a tiempo real.

La Plataforma Solar utilizada para la verificación de métodos está situada en Tabernas (Almería) y cuenta con una superficie de más de 20.000 m² de espejos, instalados en un recinto de 400.000 m². Esta planta solar es del tipo cilindro-parabólica basando su funcionamiento, como ya se explicó, en el uso de espejos cóncavos montados en forma de canal, de manera que los rayos del Sol se puedan dirigir hacia el fluido circulante. La idea de la flota

de robots móviles es que cada uno de ellos lleve un sensor incorporado y así conseguir orientar dichos sensores hacia el punto exacto de procedencia de los rayos solares en cada momento y, así, la eficiencia de la planta sea óptima, ya que la temperatura del flujo será constantemente reajustada utilizando la información dada por estos sensores móviles. El tipo de seguidor que se va a desarrollar en este proyecto será capaz de moverse en la dirección Norte-Sur y la Este-Oeste, y además será un seguidor mixto: primeramente se ejecutará de forma interna un código 'por programación astronómica', que le dará la información sobre en qué punto debería estar el Sol en cada momento (altitud y azimut solar a tiempo real) independientemente de las condiciones climáticas. Tras esto, se hará uso del sensor de medición solar que llevará nuestro sistema instalado para hacer una correción del enfoque 'por punto luminoso' gracias a un código desarrollado que busca, dentro del margen en el que ya estamos, el punto exacto que nos aporta mayor intensidad luminosa.

Así, el objetivo del proyecto ha sido el diseño, montaje, codificación y puesta en marcha de un sistema de medición de irradiancia solar y orientación a tiempo real. El presente documento se estructura de la siguiente manera:

- ✓ Primeramente se expondrán los recursos empleados para la consecución del mismo: elementos Hardware, Software y conexiones realizadas entre ellos.
- ✓ Tras esto, se dedicará un apartado a cada uno de los tres grandes pasos que conforman el proyecto y que deben ser ejecutados en bucle para su funcionamiento: cálculo de ecuaciones solares en cada instante (Hora Solar, Elevación, Acimut), movimiento de la estructura física y toma de medidas de irradiancia solar, una vez encontrado y orientado hacia el punto objetivo.
- ✓ Tras todo el desarrollo técnico del proyecto, pasaremos a la muestra de los resultados experimentales del mismo y,
- ✓ Finalmente, a las conclusiones extraídas de la realización de todo este proceso

2 RECURSOS EMPLEADOS Y CONEXIONES

EN este apartado se describirán, de manera técnica y teórica, los elementos Hardware y Software empleados para el desarrollo de nuestro proyecto, desde el sensor de medición solar hasta la interfaz gráfica de usuario SimpleBGC, pasando por diferentes lenguajes, proyectos o aplicaciones que nos han servido de base para alcanzar nuestro objetivo. Finalmente, se mostrará un esquema de conexionado de nuestros elementos, para visualizar de manera clara la estructura del proyecto.

2.1 Hardware

2.1.1 Sensor Solar MEMS

Solar MEMS es una empresa que fabrica sensores solares para naves espaciales pequeñas y medianas. Así, dispone de distintos modelos de sensores de seguimiento solar. En este proyecto se usa el modelo ISS-AX (figura 2-1), que proporciona el ángulo incidente de los rayos solares, además de las 4 salidas analógicas con la medición de los 4 medidores instalados en el dispositivo (uno en cada cuadrante fotorreceptor).

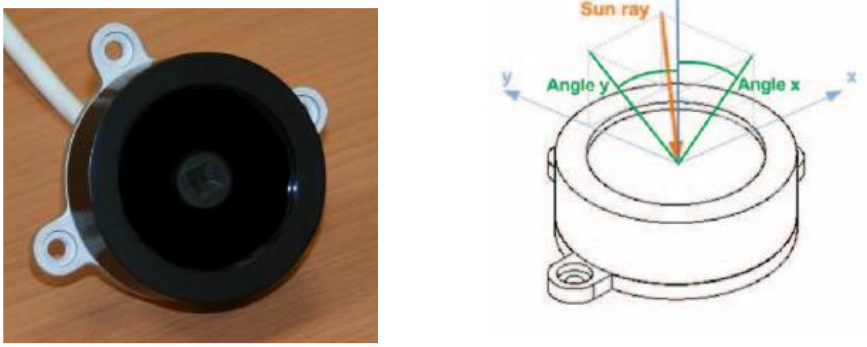


Figura 2-1. Solar MEMS, dispositivo y esquemático con sistema de referencia

El sensor tiene 6 pines de conexionado, dos de entrada (alimentación y GND) y 4 de salida (Vph1, Vph2, Vph3 y Vph4, los cuatro valores analógicos).

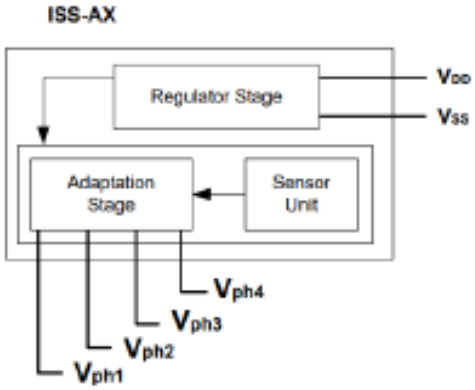


Figura 2-2. E/S del sensor ISS-AX

Algunas especificaciones técnicas del sensor son las que podemos ver en la Tabla

Tabla 2-1. Características técnicas sensor Solar MEMS

Característica	Valor
Tipo de sensor	Dos-ejes
Alimentación [V]	5-12
Consumo [mA]	11
Salidas analógicas [V/cuadrante]	4 señales: 0-4,5V
Temperatura [ºC]	-40 a 85
Protección	IP65 (polaridad inversa)

Logramos conseguir una alta sensibilidad a un bajo coste gracias a las dimensiones geométricas del dispositivo y la tecnología MEMS (MicroElectroMechanical Systems). A mayor perpendicularidad en el ángulo de incidencia, más exacta será la medida obtenida, ya que no se obtiene el valor de radiación de forma directa, sino de forma proporcional a partir de los cuatro valores analógicos obtenidos tal y como se explicará en el apartado de Medición Solar.

2.1.2 Arduino Pro Mini y FTDI Basic Breakout

La segunda gran tecnología empleada en este proyecto es aquella que engloba el mundo Arduino. Una de las placas de Arduino utilizadas es el Arduino Pro Mini. Se trata de una tarjeta pequeña, especialmente utilizada en proyectos en los que el tamaño es una limitante, con unas dimensiones de 18x33mm. Esta tecnología está dirigida a usuarias que requieran trabajar con sensores que usen tensiones de 3,3V.

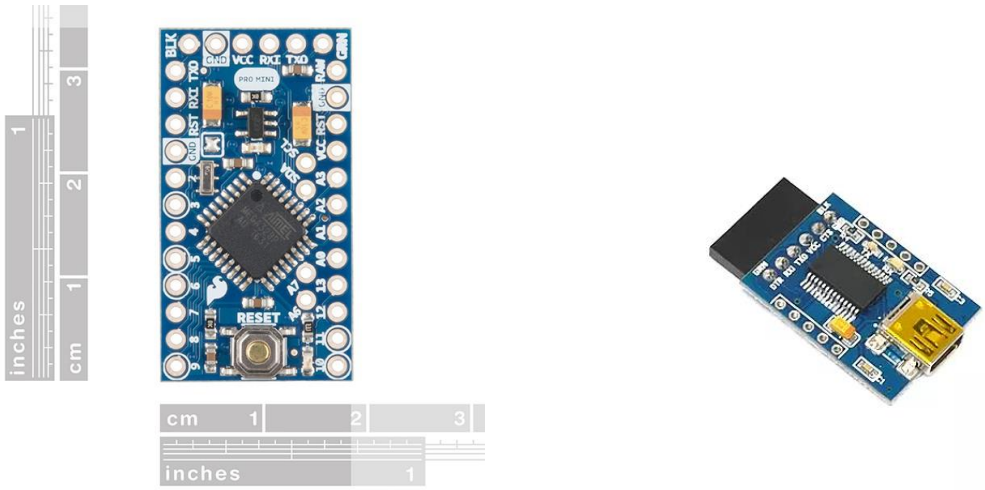


Figura 2-3. Arduino PRO MINI y FTDI USB-Serial

En particular, la tarjeta utilizada en este proyecto se corresponde con un Arduino Pro Mini 328 – 3.3V/8MHz. Como podemos observar en la imagen, cuenta con 14 entradas/salidas digitales y 6 entradas analógicas (17). La corriente máxima de salida es de 150mA, aunque cuenta así mismo con una protección contra sobre-corriente.

Para la programación de esta placa, dado que por su pequeño tamaño no posee conector USB para conectarlo al PC, es necesario utilizar una tarjeta FTDI, que realizará la conexión entre el Arduino Pro Mini y nuestro ordenador. Es importante que esta tarjeta incluya un pin RTS, que conectaremos al pin DTR de nuestro Arduino y que será el encargado de auto-reiniciar cuando un nuevo sketch se introduzca en la tarjeta. Esta característica es muy útil, ya que evita tener que pulsar manualmente el botón de reset al cargar un nuevo sketch en nuestra tarjeta.

En nuestro caso, este Arduino es un componente intrínseco del Pan&Tilt, por lo que se encuentra dentro del mismo y tiene una función explícita, como se especificará más adelante.

2.1.3 Arduino MEGA 2560

El otro Arduino empleado en el proyecto es el Arduino MEGA. Este dispositivo, al contrario que el que se acaba de presentar, tiene unas dimensiones mucho menos reducidas (102x53mm), que son precisamente las que le permiten las funcionalidades que lo diferencian del resto. Se trata de una tarjeta de desarrollo open-source construida con un microcontrolador modelo Atmega2560 con pines de E/S tanto analógicos como digitales. El Arduino MEGA tiene, entre otras cosas, 54 pines de E/S, 16 entradas análogas y 4 UARTs (puertos serial por HardWare) (18). Como implementación particular, esta tarjeta nos permite crear conexión por puerto serie con hasta 4 dispositivos diferentes (Serial0, Serial1, Serial2 y Serial3), tal y como podemos observar en la Figura 2-3. Esta característica nos ha permitido comunicar el Solar MEMS con el gimbal y visualizarlo todo en Arduino IDE, empleando tres de los puertos serie disponibles.

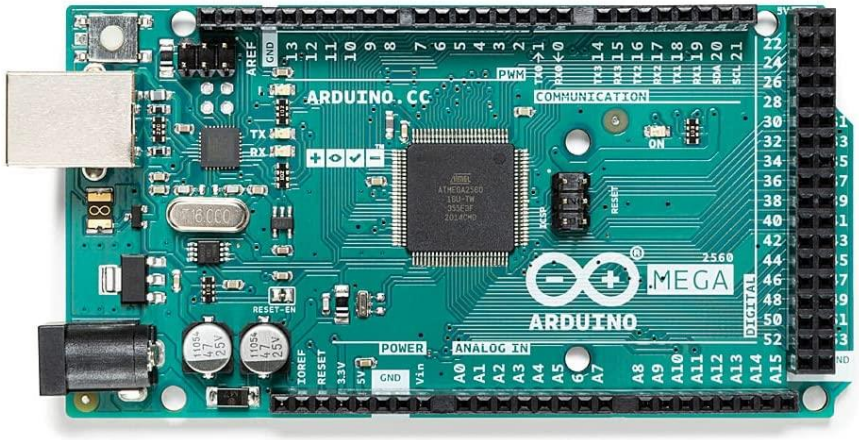


Figura 2-4. Arduino MEGA

2.1.4 Brújula Arduino QMC 5883L

La brújula de Arduino es un dispositivo independiente capaz de proporcionar las medidas x, y, z y acimut de su orientación a tiempo real. Tiene unas dimensiones muy reducidas (8.4x6x0.8cm) y un peso de 20gr, por lo que es factible su incorporación al extremo de nuestro gimbal. La brújula cuenta con 5 pines, de los cuales nosotros sólo necesitamos cuatro: Vcc, GND, SCL y SDA. Estos dos últimos son los que permiten crear la conexión: SCL (System Clock) es la línea de pulsos de reloj que sincroniza el sistema, y SDA (System Data) es la línea por la que se mueven los datos entre los dispositivos.

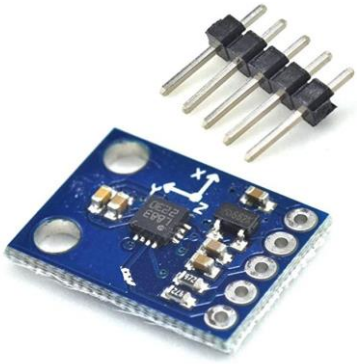


Figura 2-5. QMC5883L Compass Arduino

Si bien es importante localizar la placa en algún lugar del gimbal que no esté especialmente cercano ni al motor ni a las cámaras, ya que las propiedades ferromagnéticas de estos elementos pueden distorsionar las medidas de la brújula hasta hacerlas inválidas. Es por ello que nosotros hemos creado un soporte adicional para realizar las pruebas experimentales, que debería ser reforzado en caso de querer definitivamente instalar la brújula en el dispositivo.

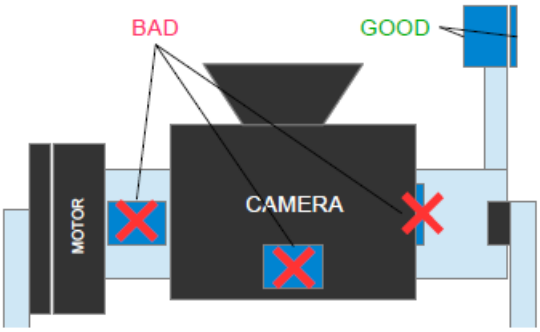


Figura 2-6. Localización del magnetómetro en el gimbal

- ✓ **IMU + Brújula Arduino**

Una posible opción para realizar esta implementación hubiera sido a través de la IMU que contiene el Pan&Tilt en su interior. La brújula se montaría en la misma plataforma que el sensor de medición, manteniendo cierta distancia como vemos en la Figura 2-6. A continuación, conectaríamos la IMU a la Brújula, siguiendo el esquema de la siguiente Figura, y lo podríamos configurar a través del GUI de Basecam, que desplegará nuevas opciones en el apartado 'Hardware' para habilitar la calibración e instalación del magnetómetro en nuestro gimbal. Los ejes de la brújula deben estar orientados de forma paralela a los del sensor IMU.

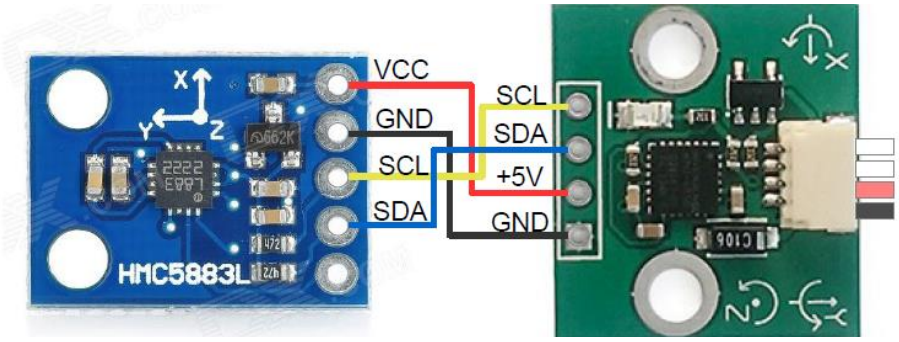


Figura 2-7. Conexión Brújula-IMU

Sin embargo, en nuestro caso esta opción no ha sido posible, ya que no existe documentación ni colaboración por parte de la empresa que diseñó el Pan&Tilt, de forma que no ha sido posible el acceso a la placa IMU ni la instalación de la conexión de la brújula.

2.1.5 Pan&Tilt

El *Pan&Tilt* es uno de los componentes principales de nuestro proyecto. Se trata de un dispostivo diseñado por la empresa sevilla DroneTools y originalmente pensado para su implementación en los drones del proyecto. DroneTools es una compañía de fabricación de RPAs (Remotely Piloted Aircraft Systems) que fabrica diferentes tipos de aeronaves con diseño a medida según la implementación buscada en cada caso.

Un *Pan & Tilt* es un dispositivo con capacidad de movimiento en el plano vertical (*Tilt*) y en el plano horizontal (*Panning*), que normalmente cuenta con una cámara para la toma de datos a tiempo real. En nuestro caso, la estructura tiene el Sensor Solar MEMS en su extremo superior, de forma que cuando el gimbal esté orientado al punto de mayor perpendicularidad solar, este pueda tomar la medida de irradiación. El dispositivo está formado por un compejo sistema electrónico, del que tan sólo son accesibles el Arduino y el sensor, además de las salidas por cable hacia el Arduino MEGA (una para el sensor de medición y otra para el movimiento del gimbal) y hacia la fuente de alimentación. Se puede observar en la Figura 2-4.

Figura 2-8. Pan & Tilt

Aunque se especificará con más detalle este procedimiento en el apartado de SW, es importante conocer que la placa controladora de nuestro gimbal es de BaseCam Electronics, una empresa que ofrece productos enfocados a la creación de sistemas de estabilización de alta calidad para cámaras fotográficas y de vídeo. Su tecnología está basada en motores brushless de accionamiento directo, que hacen que los dispositivos se puedan mover de forma suave y que pueda permanecer en quietud total cuando es necario. Ofrecen un manuel de instrucciones sobre cómo conectar, ajustar y calibrar la placa controladora de 3 ejes SimpleBGC 32bit. Para comenzar a utilizarla, se necesitan diferentes componentes: la placa controladora y adicionalmente una o dos unidades IMU (Unidad de Medición Inercial; dispositivo electrónico que mide e informa acerca de la velocidad, orientación y fuerzas gravitacionales del aparato), conexión USB a la placa o convertidor Bluetooth, un PC y el SW de Basecam. Una vez ensamblado el sistema, el gimbal podrá calibrarse y ajustarse para su corrector funcionamiento.

El gimbal, como se verá más adelante, necesita un cierto mantenimiento de cara a mantener su funcionamiento sin vibraciones, a una temperatura aceptable, etc. Para ello es necesario poder acceder a los parámetros de configuración del mismo y reajustarlo cuando sea necesario a través de la interfaz visual existente. La conexión necesaria entre el gimbal y el PC se realiza a través de una placa transmisora de información como la que vemos en la Figura 2-6. Se trata de una placa con conexión micro USB-UART (Rx, Tx, GND, 5V). Mientras que el

Micro USB va conectado al PC, el resto de cables van conectados a nuestro dispositivo, así podremos reajustar los parámetros una vez el PC reconozca la conexión.

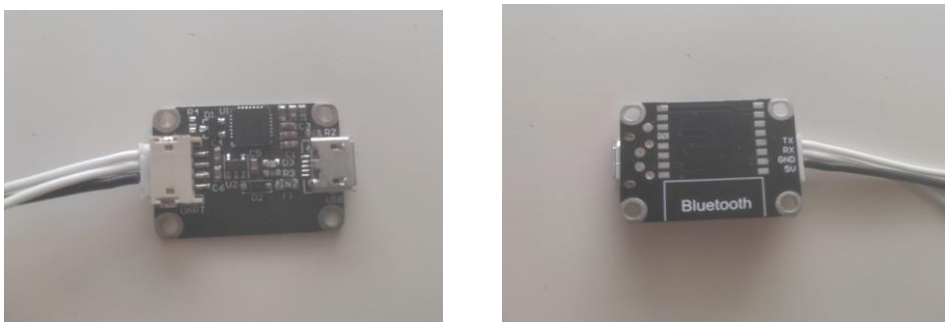


Figura 2-9. Placa conexión Pan&Tilt - PC

2.1.6 Batería LiPo

La fuente de alimentación de todo el dispositivo la conforma una Batería Lipo de 1800mAh, del tipo de la que se observa en la figura 2-4.

Figura 2-10. Batería LiPo

Las baterías LiPo (Litio y Polímero) son un tipo de baterías recargables muy habituales en el mundo de la robótica, dado que pueden almacenar una gran cantidad de energía y ofrecen una tasa de descarga muy alta. Las características de nuestra batería están elegidas acordes con la potencia y la independencia que necesitamos para este proyecto, y se pueden observar en la Tabla 2-1 (19).

Tabla 2-2. Características batería LiPo

Característica	**Valor**
Peso [g]	180
Número de celdas	3S
Capacidad [mAh]	1800
Tasa de descarga [C]	100

Paralelo [P]	1
Voltaje [V]	11.1 (3S)
Medidas [mm]	97x47x25
Longitud cable (C, D) [mm]	45, 65
Tipo conector	XT60
Tipo de conector de balanceo	JST-XH R

2.1.7 Linterna LED

De cara a realizar las pruebas experimentales con luz artificial, la idea es buscar el espectro más parecido, preferentemente con LED blanco de alta potencia. La linterna utilizada tiene un único haz de luz, lo cual también era un requisito importante, con unas dimensiones de 125 x 25 x 30 mm. Cuenta con diferentes opciones de rango de potencia y de alcance, dandonos la posibilidad de simular diferentes escenarios. La linterna lleva tres pilas AAA que le permiten

Figura 2-11. Linterna LED alta potencia

2.2 Software

2.2.1 Arduino IDE

El entorno de desarrollo integrado (IDE) de Arduino es una aplicación multiplataforma (disponible para Windows, macOS y Linux) escrita en lenguaje de programación Java, y que se utiliza para escribir y cargar programas en placas compatibles con Arduino (20). El IDE de Arduino admite los lenguajes C y C++, aunque con algunas reglas especiales de estructuración de códigos.

En el caso de este proyecto, el desarrollo del código en Arduino supone un gran porcentaje del trabajo de SW del mismo. Se ha estructurado en torno a un programa principal, desde el que se llama a las funciones definidas por usuario necesarias. El beneficio de realizarlo de esta manera es que todos los códigos quedan muy claros y ordenados, siendo mucho más sencillo para el futuro usuario del proyecto diseñado entenderlo e implementarlo. El esquema jerárquico de los programas y funciones que empleamos es el que se esquematiza en la Figura 2-6.

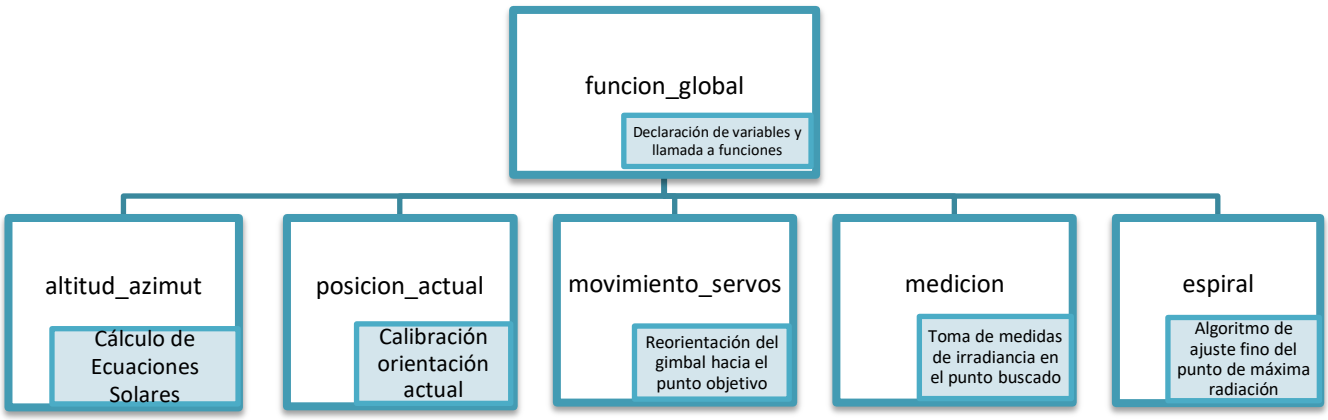


Figura 2-12. Esquemático de la jerarquía de funciones de Arduino

Función_global cumple varios objetivos fundamentales. El primero de ellos es la definición de todas las librerías y variables globales que son necesarias para llevar a cabo las tareas de cada una de las cinco funciones a las que se llama. Además, *funcion_global* es la encargada de encabezar y dirigir el resto del código, realizando las llamadas a las distintas funciones en un orden establecido y enviando, de unas a otras, los datos obtenidos por las anteriores.

Por último, todo esto se ejecuta dentro de un bucle que está programado para ejecutarse cada minuto. Esto se entiende si volvemos a plantear el objetivo final del proyecto. La idea es que nuestro sensor de medición solar esté en todo momento orientado hacia el punto de mayor radiación solar, y para ello es nuestro gimbal quien tiene que moverlo hacia ahí. Si bien el movimiento del Sol es apreciable por el ojo humano cada cierto tiempo, no es tan rápido como para tener que actualizar los cálculos constantemente, pudiendo así crear un espacio entre un cálculo y el siguiente, entre los que se ha observado que el movimiento solar es inapreciable. El periodo que hemos establecido ha sido de 60 segundos; es decir, será cada este tiempo que nuestra *funcion_global* vuelva a ejecutarse de nuevo y, con ella, las cinco subfunciones vuelvan a ponerse en funcionamiento para realizar todos los cálculos de nuevo, reorientando el sensor lo que sea necesario para volver a enfocar al punto de mayor intensidad solar.

Como podemos observar en la Figura 2-7, el procedimiento del bucle consiste en tomar la medida de tiempo en el momento de entrar al bucle a través de la función *millis*. Esta función no necesita ningún parámetro de entrada, y nos devuelve en un valor entero, de tipo *unsigned long*, la cantidad de milisegundos transcurridos desde que la placa fue encendida. De esta forma, actualizamos el valor de 'Tiempoahora' al entrar al bucle, y la condición para la próxima entrada será que *millis* sea mayor que 'Tiempoahora' + el periodo establecido, consiguiendo así la toma de medidas minuto a minuto que buscábamos.

```
//Definición de variables
  unsigned long periodo=60000; //Equivale a 60 secs
  unsigned long Tiempoahora=0;
  int hor=13;
  int minu=05;
 […]

  void loop ( ) {
  if ( millis ( ) >  Tiempoahora + periodo) {
     //Actualizo tiempo acutal en ms
     Tiempoahora = millis ( );
     […]

    //Actualizo variables para volver a empezar
    Minu = minu + 1;
    if  ( minu >= 60 ) {
    hor = hor + 1;
    minu = 0; }
}
}
```

Figura 2-13. Trozo de código que programa el bucle temporal para la toma de medidas

El valor medido por cada uno de los cuatro cuadrantes fotorreceptores del sensor de irradiación serán almacenados en una matriz, de cara a poder graficar los valores e incluso compararlos con los valores oficiales de irradiación solar de cara a realizar una transformación de los valores tomados (que se toman en valores entre 0-1024, y se transforman en el código a valores de 0-5V) a W/m^2. Así, se crea una matriz vacía con un tamaño capaz de almacenar hasta 480 tomas de medidas (que, a toma por minuto, se correspondería con 8h seguidas de experimentación) y una variable que va aumentando su valor para almacenar cada toma.

Antes de finalizar la presentación general del SW de Arduino –el desarrollo detallado de las 5 sub-funciones se hará más adelante, tras la exposición de los principios teóricos en los que se basan-, se van a exponer las diferentes librerías que se han instalado para poner en marcha todo el proyecto. Una librería es un trozo de código hecho por terceros, que facilita mucho la programación y permite la abstracción, simplificando la comprensión de nuestros programas.

- ✓ **Inttypes**

Esta librería nos permitirá crear variables de un determinado tipo de enteros, que podremos usar si necesitamos que estos estén representados en un número exacto de N bits. En concreto, los tipos que incluye son los siguientes: int8_t, int16_t, int32_t; uint8_t, uint16_t, uint32_t, con la pertinente correspondencia con el número de bits. Este tipo de variables han sido creadas en algunas de las sub-funciones a lo largo del código, como veremos conforme este se vaya explicando.

- ✓ **Timelib**

Incluye la funcionalidad de cronometraje de tiempo en Arduino. El código es derivado de la biblioteca 'Playground DateTime', aunque está actualizada para proporcionar una API más flexible y fácil de usar. Uno de los objetivos principales de esta librería es el de habilitar la funcionalidad de 'Fecha' y 'Hora', de donde salen muchísimas nuevas posibilidades de uso: reloj de tiempo real, datos de tiempo GPS, mensajes en Serie desde un PC para la sincronización de la hora, etc.

Las principales funciones disponibles son las que vemos en la Figura 2-7.

```
hour();            // the hour now  (0-23)
minute();          // the minute now (0-59)
second();          // the second now (0-59)
day();             // the day now (1-31)
weekday();         // day of the week (1-7), Sunday is day 1
month();           // the month now (1-12)
year();            // the full four digit year: (2009, 2010 etc)
```

Figura 2-14. Funcionalidades librería 'Timelib'

En nuestro caso, esta librería se utiliza en la función c*alculo_altitud_azimut*, donde se utiliza, por ejemplo, el dato *day* para calcular el ángulo diario (que depende del múmero de días del año que hayan pasado), y a raíz de ahí calcular declinación, ecuación del tiempo, etc.

- ✓ **Wire**

Esta librería permite la comunicación con dispositivos I2C/TWI. Para configurarla es importante que la placa cuente con las líneas SDA (línea de datos) y SCL (línea de reloj). En la siguiente tabla se muestra la localización de estos pines TWI en varias placas de Arduino:

Tabla 2-3. Localización pines TWI

Placa	**I2C/TWI pins**
UNO, Ethernet	A4 (SDA), A5 (SCL)
Mega2560	20 (SDA), 21 (SCL)
Leonardo	20 (SDA), 21 (SCL), SDA1, SCL1

En este proyecto, esta librería se ha instalado a partir de comenzar a usar la brújula de Arduino que, como se vio con anterioridad, contiene dos pines SCL y SDA que se han conectado a diferentes entradas de las distintas placas de Arduino a lo largo de todas las pruebas hechas durante la investigación.

- ✓ **MechaQMC**

Esta librería hace posible la implementación de la brújula QMC5883L, descrita en el apartado 2.1.4. Es una librería creada para el uso de las placas QMC5883L, como la que hemos implementado en nuestro dispositivo. Como se comentó, es capaz de devolverlos a tiempo real los valores XYZ y cálculo de azimut. También nos permite la obtención de 16 puntos de dirección de azimut (0-15) y sus nombres. Asísmimo, mediante el uso de un filtro nos puede devolver los valores XYZ suavizados, utilizando la media y la eliminación de mínimos y máximos. Su uso requiere la instalación de la librería Wire.h.

- ✓ **SBGC y SBGC_Arduino**

Estas librerías permiten implementar sketches de Arduino para controlar el gimbal de SimpleBGC de Basecam Electronics, que es el que utiliza nuestro Pan&Tilt. La API Serial permite que una aplicación o dispositivo se comunique con el controlador SimpleBGC a través del puerto UART (21). Cada controlador tiene uno o más puertos UART que pueden ser utilizados para enviar y recibir comandos de la API Serial. Los comandos pueden utilizarse para recuperar el estado actual del sistema y los datos en tiempo real, cambiar los ajustes, controlar el gimbal, activar el estado de los pines, ejecutar diversas acciones, obtener acceso a la EEPROM interna y al bus I2C, etc. Además, el software SimpleBGC GUI (del que hablamos en el siguiente apartado) utiliza la misma API Serial para comunicarse con la placa, por lo que todas sus funciones pueden ser implementadas en aplicaciones de terceros.

Así, estas librerías nos permiten controlar el movimiento del gimbal (necesario para orientarnos a la altitud y el azimut que necesitamos) al conectar la interfaz serie de Arduino, que en nuestro caso será el puerto Serial3 del Arduino Mega, al conector UART de la placa SimpleBGC (que está interna dentro del gimbal), a través de los puertos GND, Vcc y Rx → Tx, Tx →Rx.

2.2.2 Basecam GUI

El gimbal que se utiliza para nuestro proyecto tiene un controlador de BaseCam Electronics, cuya programación y ajuste se realiza a través del GUI (Interfaz Gráfica de Usuario) de SimpleBGC. La empresa proporciona un manual de instrucciones sobre cómo conectar, ajustar y calibrar la placa controladora de 3 ejes. Tras la instalación de la aplicación para Windows (*SimpleBGC_GUI.exe*), la secuencia de configuración conlleva varios pasos que, en nuestro caso, tuvieron que ser reajustados en varios momentos a lo largo del desarrollo de este proyecto:

A. Ajuste de la mecánica

Primeramente, es necesario realizar el ajuste de la mecánica equilibrando los tres ejes del gimbal. Este paso es importante ya que la calidad de la estabilización depende en gran medida de la calidad del equilibrio. Para ello, con el gimbal apagado se intenta mover de forma rápida a lo largo de los 3 ejes, y debemos llegar a conseguir el ajuste en el cual nos resulte difícil desequilibrarlo del punto orginial.

Hay que prestar especial atención a la instalación del sensor, cuyos ejes deben estar alineados en paralelo con los ejes de los motores, con enlaces rígidos y sin holgura para que el movimiento sea estable en condiciones reales (viento, vibraciones del vehículo,etc.)

B. Calibración del sensor.

Para este paso es necesario asegurarse de que el IMU esté conectado y sea reconocido por el sistema. A continuación, configuramos la orientación del sensor a través de los parámetros 'Axis TOP' Y 'Axis RIGHT'; la forma más sencilla de hacerlo es usando la herramienta de auto-detección.

- Giróscopo

La calibración del giróscopo se realiza cada ve que se enciende el controlador, y dura unos 4 segundos. De forma óptima, inmovilizaremos el sensor lo máximo posible en los primeros segundos tras el encendido (la auto-calibración comienza 1 segundo después de encenderlo).

- Acelerómetro

La calibración del acelerómetro, sin embargo, se realiza una sola vez, aunque es altamente recomendado recalibrarla cada cierto tiempo o cuando la temperatura del aparato comienza a cambiar significativamente.

C. Establecer parámetros básicos

Para ello hay que configurar la opción de 'POWER' de acuerdo a la configuración del motor, y tras esto conectar la fuente de alimentación. Los motores deben empezar a girar y, si todo está bien, el sensor/cámara se estabilizará. Existe la opción de auto-detectar el número de polos y motores; en caso de que el número de polos tenga mucho error, se puede insertar manualmente. Tras ello, los parámetros 'Gain multiplier' y 'Outer P' serán configurados a 1 y 100, respectivamente, para todos los ejes (valores por defecto), y se correrá el 'auto-tuning' para el controlador-PID, primeramente con los valores por defecto y después ajustándolo manualmente si es necesario. Algunos algoritmos sugeridos para el ajuste manual del PID son los siguientes:

1. Ajustar I=0.01, P=10, D=10 para todos los ejes. El gimbal debería ser estable; si no es así, disminuir P y D sutilmente y comenzar el ajuste de cada eje secuencialmente:
2. Aumentar gradualmente P hasta que el motor comience a oscilar (puede golpear la cámara y ver en el gráfico del giroscopio la rapidez con la que la oscilación decae). Aumentar D levemente - debería amortiguar las oscilaciones y reducir el tiempo de decaimiento (tratamos que sea lo menor posible).
3. Ajustar el paso anterior hasta que D alcance su máximo, que es cuando la vibración de alta frecuencia comienza a aparecer (se puede oír y notar la vibración, además de ver líneas ruidosas en el gráfico del giroscopio). En este momento, P y D estarán en los máximos para su configuración, por lo que se pueden disminuir.
4. Aumentar I hasta que comience la oscilación de baja frecuencia, manteniendo la estabilización del gimbal. En este momento deberíamos tener un máximo para todos los valores PID del eje seleccionado, pudiendo repetir estos mismos pasos para los otros ejes.
5. Cuando todos los ejes están sintonizados en estático, mover el marco del gimbal, emulando un entorno de trabajo real, notando que la influencia cruzada de los ejes puede hacer que el gimbal sea inestable. Si esto ocurre, disminuir sutilmente los valores del PID desde su máximo para los ejes requeridos.

Figura 2-15. Algortimo ajuste manual PID del gimbal

Si estos parámetros se establecen de la forma adecuada, el error de estabilización debería ser menor a un grado cuando se pone el aparato en funcionamiento.

D. Conexión y configuración

Esta configuración se realizará de forma individual para cada uno de los 3 ejes, siguiendo siempre los mismos pasos.

Seleccionaremos el canal, por ejemplo, 'RC_PITCH'. En la pestaña 'RC Settings' se harán los siguientes pasos: asignar la entrada 'RC_PITCH-PWM' al eje PITCH y dejar el resto de ejes como 'sin entrada'; al eje seleccionado se le ajustarán los parámetros MIN.ANGLE=-90, MAX.ANGLE=90, ANGLE MODE LPF=5, SPEED=50. A continuación, tras alimentar el sistema, comprobaremos que la entrada RC_PITCH recibe datos en la pestaña 'Monitoring', y así poder controlar el movimiento desde la emisora RC (Entre -90 y 90 grados). Se procederá de la misma forma para los dos ejes restantes.

E. Experimentación en condiciones reales

La idea de este último paso es encender el aparato mientras se tiene, momentáneamente, conectado al PC, de manera que se pueden comprobar las vibraciones a través de la pestaña de Monitorización. Se tratará de ajustar lo máximo posible cada detalle para reducir el nivel de vibraciones, y así confirmar que toda la calibración ha sido correcta antes de salir del GUI y poner en marcha el sistema.

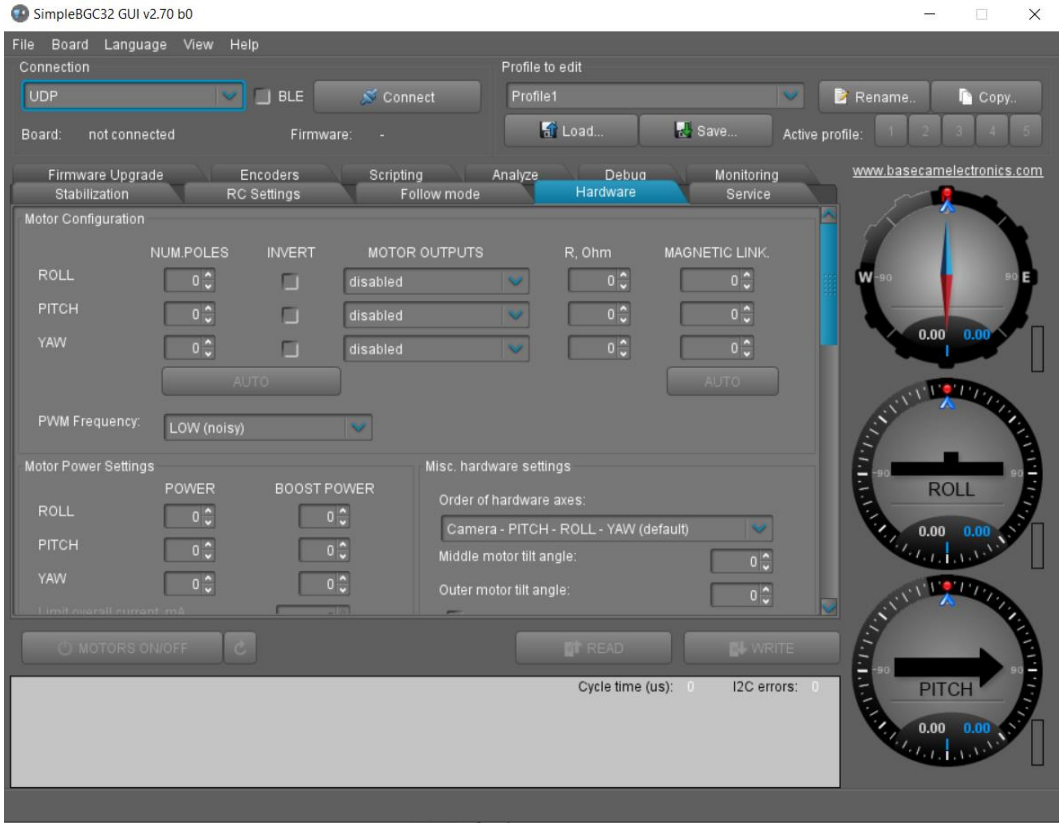


Figura 2-16. Apariencia visual del Basecam GUI

2.3 Conexiones

Es importante tener una idea clara de cómo son todas las conexiones del proyecto de cara a poder reproducirlo, ya que originalmente no existía ningún tipo de documentación que proporcionara información alguna sobre este tema. Para ello, vamos a basarnos en la fotografía que vemos en la Figura 2-17.

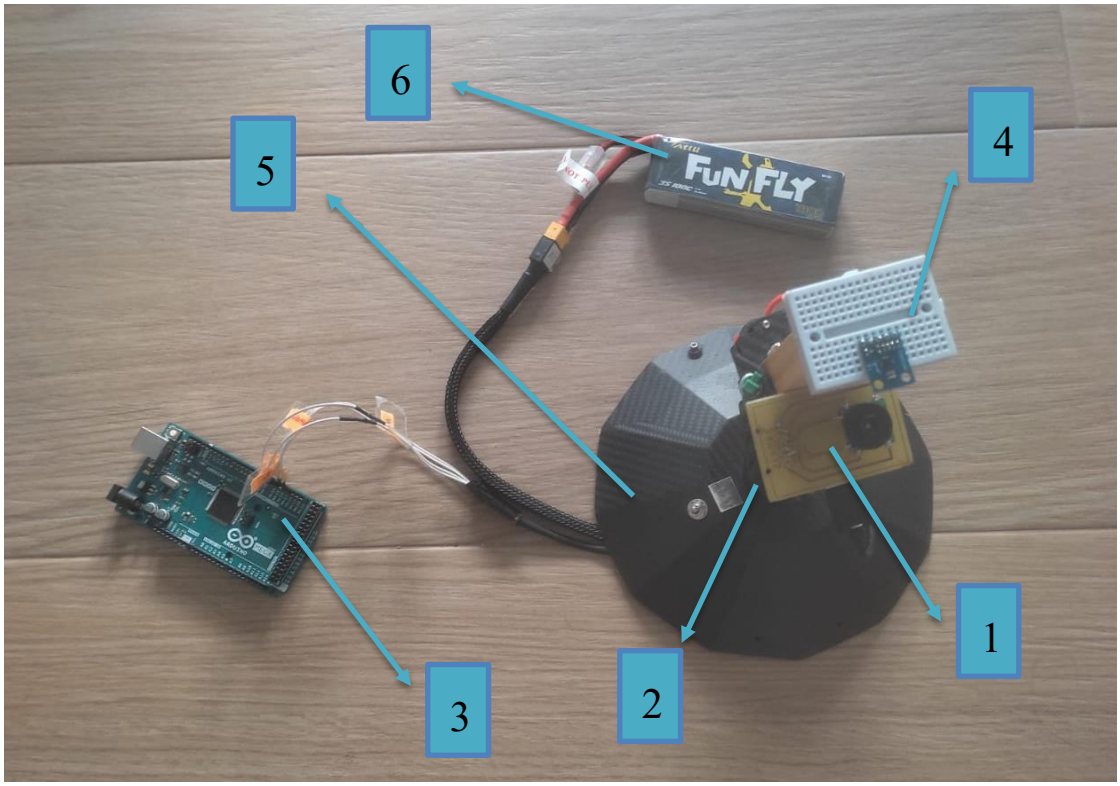


Figura 2-17. Fotografía esquematizada del montaje total del dispositivo

Los elementos numerados son los 6 componentes Hardware que ya han sido descritos con detalle en el Apartado 2.1, y que se especifican en la siguiente tabla:

Tabla 2-4. Relación numérica de los componentes HW

Numeración	Elemento HW
1	Sensor Solar MEMS
2	Arduino Pro Mini
3	Arduino MEGA
4	Brújula Arduino
5	Pan&Tilt
6	Batería LiPo

Como vemos, la conexión entre el Solar MEMS y el Arduino Pro Mini es directa, siendo este el Arduino que procesa los datos medidos, como veremos más adelante. El resto de conexiones son todas internas, de forma que del Pan&Tilt salen al exterior los cables necesarios para conectar al Arduino MEGA dos canales de datos diferentes: las mediciones del Solar MEMS (gestionadas a través del Arduino PRO Mini) y el movimiento del gimbal. Todo esto lo gestiona el Arduino MEGA gracias a la existencia de varios puertos serie. Como podemos comprobar, se ha creado una estructura que permite crear una distancia entre la brújula y el resto de aparatos que pudieran crear alteraciones en el campo magnético.

3 CÁLCULO DE ECUACIONES SOLARES

EN este capítulo vamos a describir, primeramente, los principios teóricos y matemáticos sobre los que se sustentan las Ecuaciones de Euler, que son la base utilizada en este proyecto para conseguir la orientación de nuestro Pan&Tilt al punto de mayor irradiancia solar a tiempo real. Tras ello, se prestará atención a la manera de plasmar esto en forma de código para conseguir, a tiempo real, la información solar necesaria.

3.1 Introducción

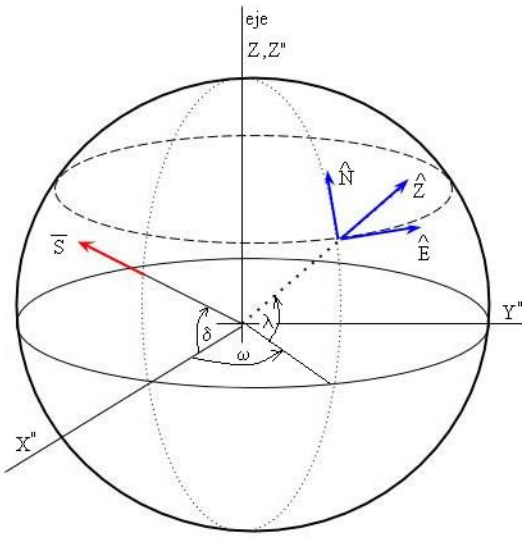


Figura 3-1. Sistema de Referencia Local

A continuación, se va a determinar de forma analítica la posición del Sol y se va a estudiar su movimiento aparente medido por un observador situado en algún lugar sobre la superficie de la Tierra. Este planteamiento matemático se basa en la idea de la existencia de un vector, S, que une el centro de la Tierra con el Sol. Para ello, consideraremos un punto situado sobre la superficie de la Tierra, con latitud λ y con ángulo horario w, de manera que definir estos dos valores será nuestro objetivo de cara a conseguir localizar este punto cualquiera sobre la superficie terrestre que hemos definido. En la Figura 3-1 se observan los sistemas de referencia existentes en este planteamietnto y el vector S definido.

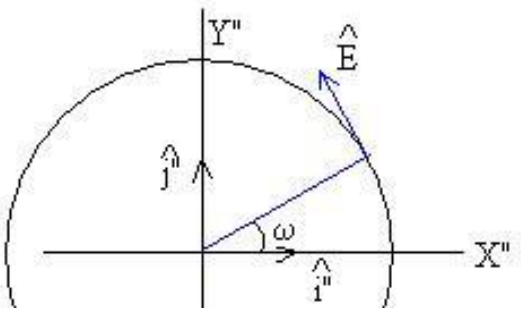


Figura 3-2. Vector E en ejes i'', j''

El Sistema de Referencia Local lo conforman los ejes E, Z, N que pueden ser expresados en términos de los vectores unitarios i'', j'', k'' (22):

$$E = -\sin\omega\cdot\hat{i}'' + \cos\omega\cdot\hat{j}'' \quad (1)$$

$$Z = \cos\lambda\cos\omega\cdot i'' + \cos\lambda\sin\omega\cdot j'' + \sin\lambda\cdot k'' \quad (2)$$

$$N = Z \times E = \begin{vmatrix} i'' & j'' & k'' \\ \cos\omega\cos\lambda & \cos\lambda\sin\omega & \sin\lambda \\ -\sin\omega & \cos\omega & 0 \end{vmatrix} =$$

$$= -cos\omega\cdot sin\lambda\cdot i'' - sin\omega\cdot cos\lambda\cdot j'' + cos\lambda\cdot k'' \quad (3)$$

Una vez tenemos estos tres vectores, se pueden sacar los componentes de S mediante el uso del producto escalar de los ejes por el vector S:

$$S_N = S\cdot N = -\cos\omega\cdot\sin\lambda\cdot\cos\delta + \cos\lambda\cdot\sin\delta \quad (4)$$

$$S_E = S\cdot E = -\sin\omega\cdot\cos\delta \quad (5)$$

$$S_Z = S\cdot Z = \cos\omega\cdot\cos\lambda\cdot\cos\delta + \sin\lambda\cdot\sin\delta \quad (6)$$

$$S = -\sin\omega\cdot\cos\delta\cdot E + (\cos\lambda\cdot\sin\delta - \cos\omega\cdot\sin\lambda\cdot\cos\delta)\cdot N + (\cos\omega\cdot\cos\lambda\cdot\cos\delta + \sin\lambda\cdot\sin\delta)\cdot Z \quad (7)$$

Por lo que sabemos que, para conseguir la definición de este vector, necesitamos obtener las fórmulas de dos parámetros distintos: la declinación, δ, y el ángulo horario, w.

3.2 Declinación, δ

Así, definimos la declinación solar como el ángulo que forma la línea Sol-Tierra y el plano del Ecuador, que es un valor que cambia no sólo a diario, sino en cada instante. Su variación a lo largo del año la observamos en la gráfica de la Figura 3-2, donde se observa que oscila desde un máximo de +23.45º en el solsticio de verano boreal, hasta un mínimo de –23.45º en el solsticio de invierno boreal. En los puntos en los que este valor es nulo, se trata de los equinoccios de primavera y de otoño.

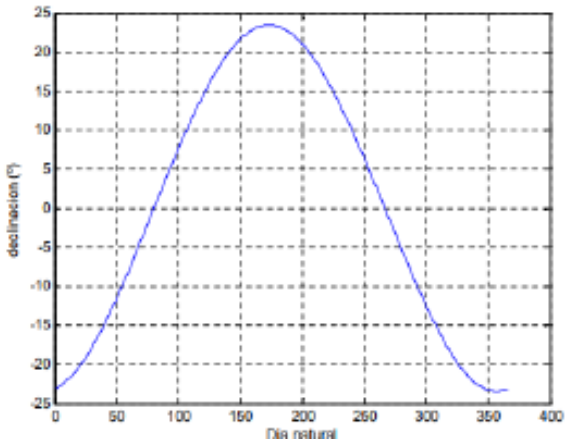


Figura 3-3. Variación declinación solar

Si expresamos el vector S en términos de los vectores unitarios i'', j'', k'', resulta que este forma un ángulo δ (declinación) con el eje X'', y se expresa de la siguiente manera:

$$S = \cos\delta \cdot i'' + \sin\delta \cdot k'' \quad (8)$$

La ecuación matemática que calcula el valor exacto de esta declinación es, por tanto, dependiente del ángulo diario, que se expresa en radianes y tiene la siguiente formulación,

$$\tau = \frac{2\pi}{365}(d_n - 1) \quad (9)$$

Donde dn expresa el día juliano del año, que irá entre 1 y 365. Así, la declinación se puede calcular siguiendo la siguiente fórmula:

$$\delta = (0.006918 - 0.399912\cos\tau + 0.070257\text{sen}\tau - 0.0067\cos 2\tau + 0.0009907\text{sen}2\tau - 0.002697\cos 3\tau + 0.00148\text{sen}3\tau) \quad (10)$$

3.3 Ecuación del tiempo, E_t

Si volvemos al origen de este desarrollo, vemos que para definir un punto sobre la superficie terrestre necesitamos su valor de latitud (λ) y su valor de ángulo horario (*w)*. Como ya sabemos, la latitud es un valor en grados que se mide desde el Ecuador, y que tiene los límites en $\pm 90^{o}$. Para conocer el valor de *w*, necesitamos hablar de la ecuación del tiempo. La Ecuación del tiempo, E_t, se define como la diferencia entre el tiempo solar verdadero y el tiempo solar medio.

El tiempo solar verdadero depende de la rotación de la Tierra sobre su eje polar y el movimiento de traslación alrededor del Sol. Está basado en el día solar verdadero, que es el intervalo de tiempo en el que el Sol completa un ciclo alrededor de un observador estacionario y la Tierra, y que no es uniforme y varía a lo largo del año. El efecto de estas variaciones es lo que produce desviaciones estacionales. Para evitarlo, se toma una esfera terrestre ficticia que posee un movimiento de rotación uniforme alrededor del Sol, creando así el término del tiempo solar medio. La Ecuación del tiempo es la diferencia entre ambas, y según las ecuaciones de Spencer, se puede obtener su valor numérico de la siguiente manera:

$$E_t = (0{,}000075 + 0.001868 cos\,\tau + 0.032077 sin\,\tau - 0.014615 cos\,2\tau - 0.04089 sin\,2\tau)(229.18) \quad (11)$$

Donde la multiplicación final está realizando la conversión de radianes a minutos. Teniendo en cuenta esta fórmula, la gráfica de la Ecuación del tiempo sería la que observamos en la Figura 3-3,

Figura 3-4. Alteración de la Ecuación del tiempo

donde observamos que la diferencia máxima será de unos 16 minutos.

Recordamos que estamos tratando de obtener la fórmula de w. Para ello, nos queda definir el concepto de horal local aparente, T_S:

$$T s = t i e m p o \ l o c a l \ e s t a n d a r + c o r r e c c i o n \ d e \ l o n g i t u d + E t$$

Donde el tiempo local estándar es la hora del día y la correción de la longitud equivale a 4·longitud. De esta manera, el ángulo w lo calcularemos mediante la siguiente fórmula:

$$\omega = ((T s - 12) \cdot 15) \cdot (\frac{\pi}{180}) \quad (12)$$

3.4 Elevación y azimut – Ecuaciones de Euler (cálculo)

La elevación solar se define como el ángulo de elevación del Sol; es decir, la distancia angular vertical que hay entre el cuerpo medido y el horizonte de la persona que lo está observando o el plano local del observador (23).

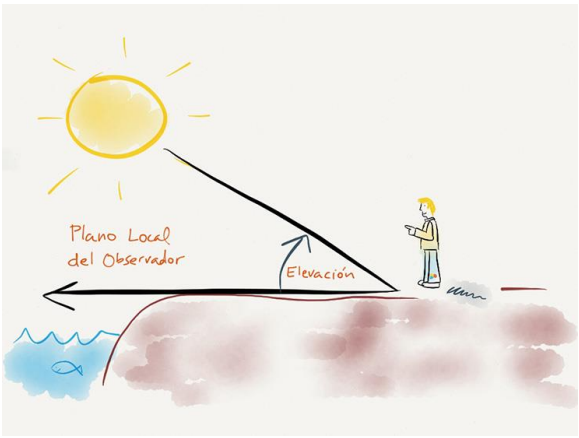


Figura 3-5. Descripción gráfica de la Elevación Solar

El azimut es el ángulo formado entre el Norte y el cuerpo en observación, medido en el sentido de rotación de las agujas del reloj alrededor del horizonte de la persona. Así, tal y como podemos observar en la Figura 3-6, el azimut de un cuerpo situado al Norte será de 0º, mientras que el azimut de un cuerpo situado a Oeste sería de 270º.

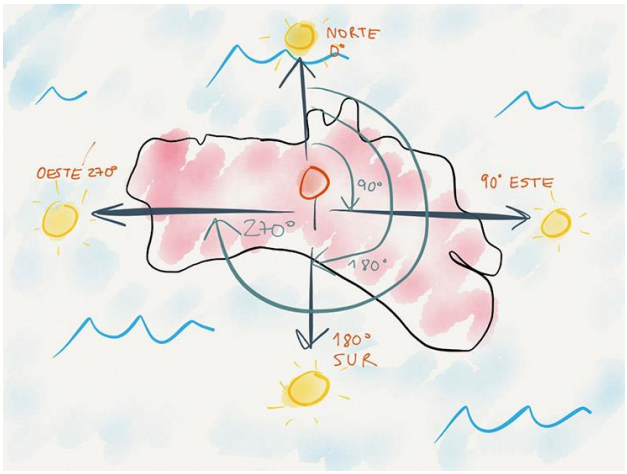


Figura 3-6. Descripción gráfica del Azimut Solar

Basándonos en las ecuaciones (4), (5), (6), en las que definíamos las componentes del vector S en función de ω, δ, vemos que este también se puede expresar en términos de dos ángulos: a_s(altitud del Sol) y γ_s (acimut del Sol), tal y como vemos en la Figura 3-7,

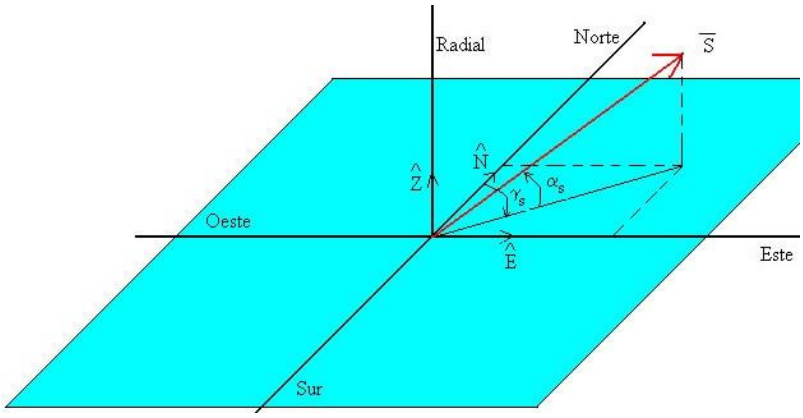


Figura 3-7. Vector S con Sistema de Referencia Local

quedando la siguiente ecuación; al igualar las componentes, obtenemos las relaciones de las ecuaciones (14), (15), (16):

$$S = \cos\alpha s \sin\gamma s \cdot E + \cos\alpha s \cos\gamma s \cdot N + \sin\alpha s \cdot Z \quad (13)$$

$$\cos\alpha s \cdot \sin\gamma s = -\sin\omega \cdot \cos\delta \quad (14)$$

$$\cos\alpha s \cdot \cos\gamma s = \cos\lambda \cdot \sin\delta - \sin\lambda \cdot \cos\omega \cdot \cos\delta \quad (15)$$

$$\sin\alpha s = \cos\lambda \cdot \cos\omega \cdot \cos\delta + \sin\lambda \cdot \sin\delta \quad (16)$$

De las últimas ecuaciones, podemos despejar la altitud (αs) y el acimut (γs):

$$\sin \alpha s = \cos\lambda \cdot \cos\omega \cdot \cos\delta + \sin\lambda \cdot \sin\delta \quad (17)$$

$$\cos\gamma s = \frac{cos\lambda \cdot sin\delta - cos\omega \cdot sin\lambda \cdot cos\delta}{\cos\alpha s} \quad (18)$$

Utilizando el arco coseno de la ecuación (18), podremos obtener γs en un ángulo entre 0-180º, pero ya sabemos que el acimut del Sol puede obtener un valor entre 0-360º. Es por ello que el cálculo del acimut incluirá lo siguiente:

$$\gamma c = \arccos\left(\frac{\cos\lambda\sin\delta - \cos\omega\sin\lambda\cos\delta}{\cos\alpha s}\right) \quad (19)$$

$$\gamma c = \arccos(\cos\lambda\sin\delta - \cos\omega\sin\lambda\cos\delta\cos\alpha s) \quad (20)$$

- Si el ángulo horario ω < 0 entonces γs = γc
- Si el ángulo horario ω > 0 entonces γs = 360 - γc

De forma que si el acimut pertenece al intervalo [0, 180º], entonces el Sol está al Este del observador, lo cual se correspondería con las horas entre las 00-12h (w < 0). Mientras que si el acimut se encuentra en el intervalo [180, 360º], el Sol estará al Oeste del observador y estaremos entre las 12-00h, donde w > 0.

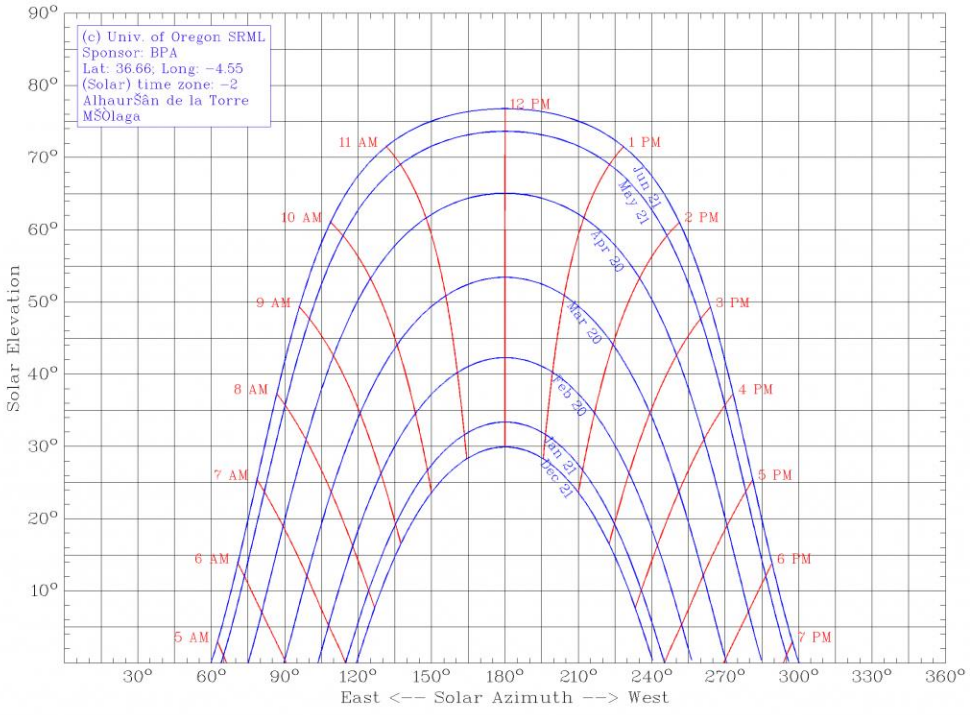


Figura 3-8. Gráfico valores altitud-azimut solar

3.5 Código: *ecs_solares*

Como ya sabemos, dentro de los objetivos de nuestro proyecto existen varios pasos. El primero de ellos es saber el valor numérico del acimut y de la altitud, para así poder orientar hacia ese punto a nuestro sensor y tras ello continuar con el procedimiento. Por esto era necesario la previa explicación teórica de estos conceptos, que incluyen los cálculos de la declinación y la hora solar, por lo que también han sido explicados.

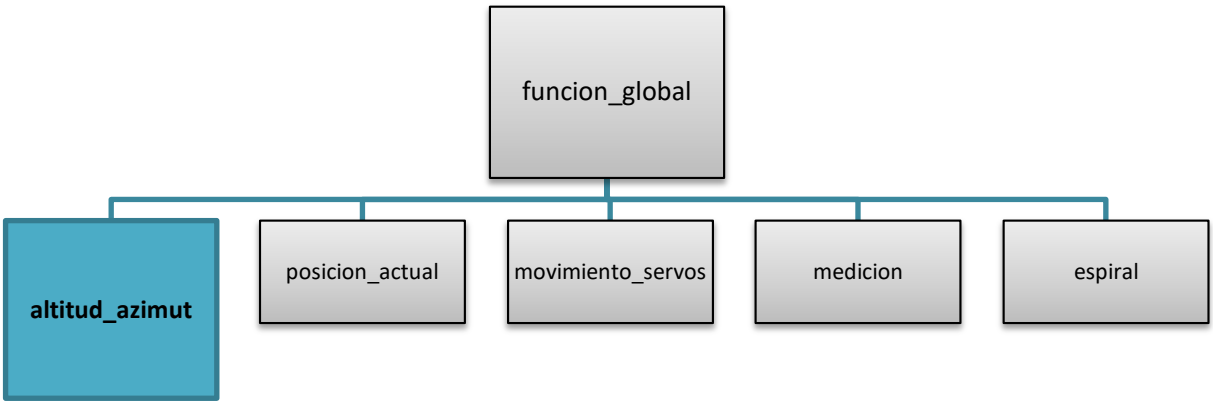


Figura 3-9. Función actual dentro del esquema general

Primeramente se realiza la definición de variables. Es importante en este caso definir un tipo de variable, 'time_t', que es lo que devuelve la función ponFecha. Esta función, que vemos a continuación, hace uso de la librería Time.h anteriormente explicada.

```
time_t ponFecha ( int y, int m, int d, int hh, int mm, int ss ) {
  tmElements_t f;
  f.Second = ss;
  f.Minute = mm;
  f.Hour = hh;
  f.Day = d;
  f.Month = m;
  f.Year = y - 1970;
  return makeTime (f);     //Crea tiempo Unix
}
```

Además de esto, se crea un vector con el número de días de cada mes del año, y también las variables que indican la latitud y longitud del lugar donde se van a realizar las pruebas. Así, la definición de variables sería la siguiente:

```
int tabla[ ]={31,29,31,30,31,30,31,31,30,31,30,31}; //días que tiene cada mes del año
int dias=0;
time_t t0,t1,dif;

//SEVILLA
float latit=37.38283*(PI/180); //En radianes
float longit=-5.973717; //En º
```

Para entrar en la ejecución de la función, *void ecs_solares*, necesitamos hacerle llegar determinados argumentos desde la función principal, que serán la información otorgada sobre hora, minuto, segundo, mes y día en donde comenzó la toma de medidas, y la información sobre latitud y longitud que acabamos de proporcionar. Tras algunas líneas de código en las que se hacen cálculos sobre el día juliano (guardado en la variable *dias*), actualización del tiempo, etc, pasamos a los cálculos de las variables ya presentadas.

Al cálculo de la declinación lo precede el cálculo del ángulo diario, que consiste en el ángulo (en radianes) que habrá girado la Tierra, partiendo de la idea de que en 365 días gira 2PI rad. Pese a que el cálculo se realiza en radianes, se traspasa a grados y se almacena en la variable *dec*.

```
float angulodiario=(2*PI/365)*(dias-1);
declinacion=0.006918-0.399912*cos(angulodiario)+0.070257*sin(angulodiario)-
0.006758*cos(2*angulodiario)+0.0009907*sin(2*angulodiario)-
0.002697*cos(3*angulodiario)+0.00148*sin(3*angulodiario); //declinación en radianes
float dec=declinacion*180/PI; //declinación en º
```

Tras ello, pasamos al cálculo del ángulo horario, w, cuya ecuación la podemos ver en (12). Recordamos que para ello necesitamos el cálculo de Ts, donde también entra en juego la Ecuación del tiempo ($Ts = tiempo\ local\ estandar + correccion\ de\ longitud + Et$). Con todo esto, la codificación quedaría como sigue:

```
float et=(0.000075+0.001868*cos(angulodiario)-0.032077*sin(angulodiario)-0.014615*cos(2*angulodiario)-
0.04089*sin(2*angulodiario))*229.18;
float diferenciatotal=diferenciames+diferenciapos-et; //Diferencia total en minutos
float Ts=horas-(diferenciatotal/60); //hora solar verdadera (minutos)
float w=((Ts-12)*15)*(PI/180); //En radianes
```

Es importante tener en cuenta las unidades. En este caso, tanto *w* como *altitud* y *acimut* se calcuarán en radianes durante todo el proceso (unas dependen de otras, así que se ha unificado a radianes), y una vez terminado todo el cálculo se realiza la transformación a grados. El azimut corregido es el ángulo que le enviaremos a la señal teniendo en cuenta dónde se encuetra en Pan&Tilt en el momento actual.

```
altitud=asin(cos(latitud)*cos(declinacion)*cos(w)+sin(declinacion)*sin(latitud));
acimut=acos((cos(latitud)*sin(declinacion)-cos(w)*sin(latitud)*cos(declinacion))/cos(altitud));
altitud=altitud*(180/PI);
acimut=acimut*(180/PI);
if(w>0) acimut=360-acimut;
acimut_corregido=acimut-angulo_inicial;
```

4 MOVIMIENTO PAN&TILT

Una vez tenemos calculados los valores de altitud y acimut, nuestro siguiente paso es orientar el Pan&Tilt –y, por tanto, el sensor de medición instalado en su extremo- hacia ese punto geográfico.

4.1 Introducción

Si nos paramos a determinar cómo se debe hacer realmente este proceso caemos en la cuenta de que, antes de enviar la señal de movimiento al gimbal, deberemos conocer hacia dónde está apuntando actualmente. Si bien existe la opción alternativa de orientar el gimbal hacia una posición determinada (por ejemplo, al Norte) y realizar el movimiento completo hacia el acimut calculado, entendemos que esto no es una solución óptima, ya que supondría dos movimientos añadidos tras cada entrada al bucle. La opción más eficiente, sin embargo, calcula la posición actual y, junto con el dato del acimut, calcula la diferencia de ángulos y ese es el movimiento que se le envía al gimbal.

Es por esto que el paso de orientación al gimbal consta de dos llamas a subfunciones desde la función principal, tal y como se ve en la Figura 4-1. Primeramente se calcula la posición actual y, después, la diferencia de ángulos y el envío de la señal de movimiento al gimbal.

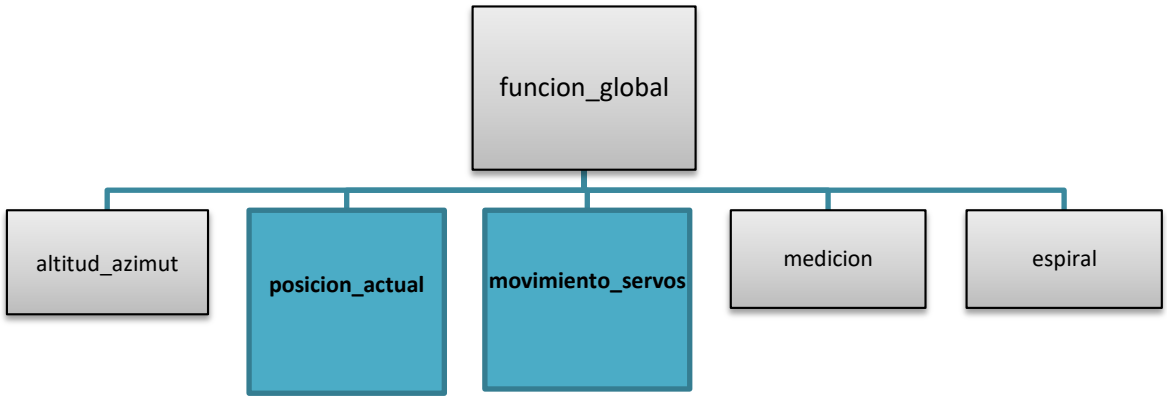


Figura 4-1. Funciones actuales dentro del esquema general

4.2 Código: *posicion_actual*

Para tomar la medida de la posición actual, se han probado diferentes métodos experimentales.

La opción más básica y que siempre estará disponible será la de orientar manualmente nuestro gimbal hacia el Norte (donde el azimut será de 0º) y enviarle la señal para que gire el ángulo del azimut actual medido. Así, si en ese instante el azimut es, por ejemplo, de 134º, le enviamos al gimbal que vaya directamente hacia ese ángulo. En este caso, el cálculo de la posición actual no sería necesario, ya que contamos con la idea de que angulo_actual será 0º.

Sin embargo, de cara a automatizar lo máximo posible el funcionamiento del proyecto, la idea es que exista un instrumento capaz de medir en cada momento dónde se encuentra nuestro sensor.

La implementación más sencilla y barata es a través del uso de la Brújula de Arduino anteriormente presentada. Existen varias librerías que podemos instalar para realizar un código válido. En nuestro caso, hemos instalado la librería <*QMC5883LCompass.h*>. El código en cuestión se almacena dentro de la función *posicion_actual*, que devuelve la variable global *angulo_actual* con el valor numérico de la orientación del Pan&Tilt.

```
float posicion_actual (float declinacion) {
   int16_t xS,yS,zS;
   float suma=0;
   float angulo_actual;

   for (int k=0; k<20; k++){ //Cálculo de la media de 20 medidas tomadas en este punto
   qmc.read(&xS,&yS,&zS);
   float angulo=atan2(yS,xS); //Azimut magnético al que está el sensor
   suma=suma+angulo_actual;
   }

   angulo_actual=suma/20;
   angulo_actual=angulo_actual * RAD_TO_DEG;
   angulo_actual=angulo_actual-declinacion; //Azimut geográfico
   if(angulo_actual<0) angulo_actual=angulo_actual +360;

   return angulo_actual;
}
```

Para tomar una medida válida, dado que la brújula recibe de forma aleatoria valores fuera del rango real, se toman 20 medidas y se hace la media entre ellos dentro del bucle for. La llamada a los valores se hace, como podemos ver en el código, a través de 'qmc.read', que nos almacena en las variables local x, y, z los valores medidos en este instante. Una vez fuera del bucle, se calcula la media y se implementa la función de Arduino encargada de transformar el valor a grados sexagesimales. Finalmente, tal y como dice la teoría de las Ecuaciones de Euler, se resta la declinación al ángulo medido, y se transforma a un ángulo perteneciente al intervalo [0, 360º].

4.3 Código: *movimiento_servos*

El código que procesa el movimiento de los servomotores lo alberga la función movimiento_servos. Desde el código principal, una vez tenemos el valor numérico de la variables global angulo_actual (que será 0º en el caso de que se realice una orientación manual del aparato hacia el Norte), se hace una llamada a esta función y se realiza el movimiento del Pan&Tilt. Como sabemos, el gimbal tiene tres ejes de movimiento – ROLL, PITCH, YAW- y todos son controlables a través de las funciones que nos habilitan las librerías <SBGC.h>, <SBGC_Arduino.h>.

En concreto, se crea una variable global en la función principal del tipo *SBGC_cmd_ccontrol*, llamada c. Esta variable contiene 6 valores numéricos que nos dan la siguiente información: (modo, velocidad_ROLL,

angulo_ROLL, velocidad_PITCH, angulo_PITCH, velocidad_YAW, angulo_yaw). En el momento en que se crea la variable, los seis valores están inicializados a 0.

En nuestro caso, las variables que querremos modificar serán angulo_PITCH y angulo_YAW, para orientar el sensor al punto deseado. El Pan&Tilt estará conectado al canal Serial 3 de nuestro ArduinoMEGA. En la función void setup, lo llevamos a una posición inicial y configuramos los parámetros definidos desde la funcion_global.

```
Serial3.begin(SERIAL_SPEED); //Movimiento Pan&Tilt
SBGC_Demo_setup(&Serial3);
delay(3000);
c.mode = SBGC_CONTROL_MODE_ANGLE;
c.speedROLL = c.speedPITCH = c.speedYAW = 50 * SBGC_SPEED_SCALE;
c.anglePITCH = SBGC_DEGREE_TO_ANGLE(-90);
c.angleYAW = SBGC_DEGREE_TO_ANGLE(0);
SBGC_cmd_control_send(c, sbgc_parser);
```

Una vez ya dentro de la función movimiento_servos, esta solamente recibirá los valores numéricos de altitud y acimut corregido y reescribirá sobre la variable c. Al igual que el giro en YAW es completo (360º), en PITCH será de [-180º,0º]. Sin embargo, tal y como vimos en la Figura 3-8, el valor de elevación solar se da entre 0-90º, de manera que el procedimiento será realizar primero el movimiento de azimut solar con una elevación estándar (apuntando al cielo con el ángulo máximo) y, una vez llegados a ese punto, ir al ángulo exacto:

```
float altitud_corregida;
void Servos(float altitud, float acimut_corregido){
    c.angleYAW = SBGC_DEGREE_TO_ANGLE(acimut_corregido);
    altitud_corregida=altitud-180; //Transformación de la altitud a los valores del gimbal
    c.anglePITCH = SBGC_DEGREE_TO_ANGLE(altitud_corregida);
    SBGC_cmd_control_send(c, sbgc_parser);
    delay(5000); //Movimiento del gimbal
 }
```

5 TOMA DE MEDIDAS DE IRRADIANCIA SOLAR

El objetivo final, si nos retrotraemos al inicio del documento, es el de habilitar una toma de medidas correcta para obtener una estimación distribuida de la estimación solar que nos permita reajustar en tiempo real el flujo de aceite que se aplica a cada lazo del campo de colectores. Para ello, evidentemente, una de las funciones últimas será la de la toma de medidas. Sin embargo, de cara a realizar un ajuste más fino aún de la orientación del Pan&Tilt hacia el Sol, la función final combinará el movimiento del gimbal con los valores de irradiancia medidos, siguiendo un algoritmo en espiral, para encontrar el punto de mayor valor numérico.

5.1 Introducción

Es por ello que el esquema que venimos mostrando en cada sección marca las dos últimas casillas: medicion y espiral, que nos permitirán terminar el ciclo.

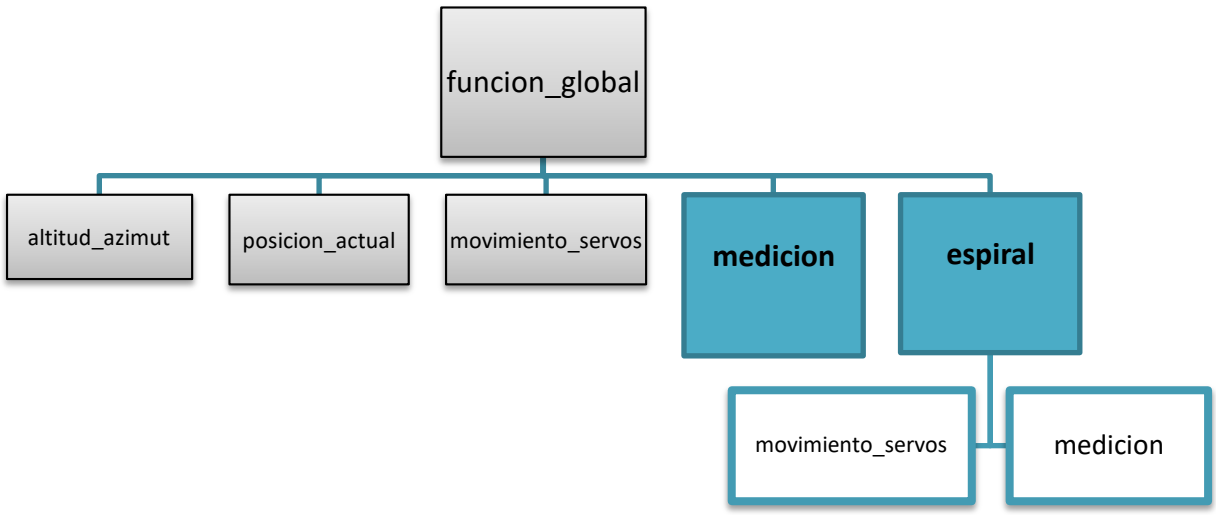


Figura 5-1. Funciones actuales dentro del esquema general

5.2 Código: *medicion*

El código programado se encarga de recibir los cuatro valores numéricos tomados por el sensor. Sin embargo, como ya se explicó, estas medidas las toma el Arduino Mini Pro que se encuentra instalado en el Pan&Tilt, por lo que la función de la que hablamos aquí, medicion, será la función complementaria a ese código, que recibirá por puerto serie en el Arduino MEGA esta información.

Para ello, primeramente debemos consultar el Datasheet del sensor Solar MEMS, de cara a entender las matemáticas que sustentan estas líneas de código. Ya vimos en apartados anteriores las características del sensor y su división en cuatro cuadrantes diferentes. Así, utilizando estas medidas y las ecuaciones que se plantean a continuación, podemos calcular el ángulo de incidencia solar:

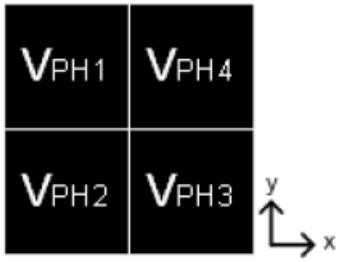


Figura 5-2. Fotodiodo del sensor y ejes

- Cálculo del ángulo X

$$X_1 = V_{PH3} + V_{PH4}$$
$$X_2 = V_{PH1} + V_{PH2}$$
$$F_X = \frac{X2 - X1}{X2 + X1}$$
$$AnguloX = arctg\ (C \cdot F_X)$$

(21)

- Cálculo del ángulo Y

$$Y_1 = V_{PH1} + V_{PH4}$$
$$Y_2 = V_{PH2} + V_{PH3}$$
$$F_Y = \frac{Y2 - Y1}{Y2 + Y1}$$
$$AnguloY = arctg\ (C \cdot F_Y)$$

(22)

V_{PH1}, V_{PH2}, V_{PH3} y V_{PH4} serán los valores obtenidos por los cuatro sensores. El valor de la constante C varía según el modelo del sensor. Para el nuestro, un ISS-A60, el valor de C será C=1.871.

Estos son los cálculos en los que nos basaremos para encontrar la mejor orientación del Pan&Tilt hacia el Sol, que será el punto de mayor irradiancia. Para ello, primeramente vamos a mostrar partes del código que el Arduino Pro Mini traía instalado en su interior, de manera que se entienda el código complementario creado por nosotros para transformar los datos leídos en estos valores:

```
int vph1;
int vph2;
int vph3;
int vph4;
byte vphPtr[2];
```

Tras inicializar el puerto serie, la función en bucle hace lo siguiente:

```
void loop() {
  if ( Serial.available() > 0 ) {
    char order = Serial.read();
    if ( order == 'a' ){
      vph1 = analogRead(A0);
      vph2 = analogRead(A1);
      vph3 = analogRead(A2);
      vph4 = analogRead(A3);

      Int16ToByteArray(vph1, vphPtr);
      Serial.write(vphPtr, 2);
      Int16ToByteArray(vph2, vphPtr);
      Serial.write(vphPtr, 2);
      Int16ToByteArray(vph3, vphPtr);
      Serial.write(vphPtr, 2);
      Int16ToByteArray(vph4, vphPtr);
      Serial.write(vphPtr, 2);
    }
  }
}
```

Si nos fijamos, sencillamente realiza la lectura de los pines A0, A1, A2, A3 a los que están conectadas las cuatro salidas del Solar MEMS, y pasa estos valores por la función Int16ToByteArray, que los transforma a un vector de dos componentes que nos dará su valor entre 0-1024.

```
void Int16ToByteArray(int _int, byte* _array){
  byte *ptrAux = _array;
  *(ptrAux++) = _int & 0xFF;
  *(ptrAux++) = (_int >> 8) & 0xFF;
}
```

De esta manera, va enviando por puerto serie estos valores conforme los va calculando, por lo que la función complementaria que busque recibir esta información, tendrá que tener en cuenta esto.

Una vez presentada la función implementada dentro del Arduino Pro Mini, pasamos a mostrar cómo es el código de la función que hemos implementado nosotros en el Arduino externo que ha ido recibiendo toda la información a lo largo del proyecto.

Se crearán tanto las variables iniciales como las que albergarán los valores finales buscados: Fx, Fy, ángulos, etc. Además se han creado las variables 'suma', ya que la medida final que se tomará de cada punto será una media entre varios valores tomados seguidamente, para evitar los fallos.

```
char ej;
float V1[2], V2[2], V3[2], V4[2];
float V1b, V2b, V3b, V4b;
float X1, X2, Y1, Y2, angX, angY;
float Fx, Fy, valor1, valor2;
float pi = 3.141593;
float V1_suma=0, V2_suma=0, V3_suma=0, V4_suma=0, anguloX_suma=0, anguloY_suma=0;
```

Como vimos en la función anterior, el Arduino queda a la espera de la entrada por teclado de un valor. Entonces lo almacena y comprueba que sea una 'a' para comenzar la toma de medidas. Es por eso que en este código, que es el que irá en las pruebas conectado al PC, se debe crear esta comunicación de cara a poner en funcionamiento el proceso. Recordamos que esa comunicación ocupa el canal Serial1 del Arduino MEGA.

```
Serial.println("Estamos midiendo irradiación, introduce 'a':");
ej = Serial.read();
[…]

if ( Serial1.available() > 0 ) {
 for (int k=0; k<20; k++){ //Cálculo de la media de 20 medidas tomadas en este punto
   Serial1.write(ej); //Envía el valor 'a' al Arduino Pro Mini
```

En este punto ya se ha 'activado' la toma de medidas, por lo que nos disponemos a leerlas en el mismo orden en que la otra placa las envía, y a traducirlas a los valores que nos interesan, teniendo en cuenta las ecuaciones (21) y (22). Mostraremos como ejemplo el proceso para el la medida Vph1, entendiendo que se repite para el resto de medidas:

```
//LECTURA DE MEDIDAS//
    //V1//
    V1[0] = Serial1.read();
    V1[1] = Serial1.read();
    V1b = transforma2(V1[0], V1[1]); //Saco el valor entre 0-1024
    V1b = (V1b * 5) / 1024; //Lo transformo a 0-5V
    V1_suma = V1_suma + V1b;
```

Una vez realizadas las lecturas, pasamos al cálculo de datos:

```
//CÁLCULO ÁNGULO INCIDENCIA//

    X1 = V3b + V4b; //Medidas entre 0-10V
    X2 = V1b + V2b;
    Y1 = V1b + V4b;
    Y2 = V2b + V3b;

    Fx = (X2 - X1) / (X2 + X1);
    Fy = (Y2 - Y1) / (Y2 + Y1);

    valor1 = Fx * 1.871; //Constante C por el modelo de SolarMems usado
    angX = atan(valor1);
    angX = (angX * 180) / pi; //Pasado a grados sexagesimales
    anguloX_suma = anguloX_suma + angX;

    valor2 = Fy * 1.871;
    angY = atan(valor2);
    angY = (angY * 180) / pi;
    anguloY_suma = anguloY_suma + angY;
```

Y, una vez fuera del bucle de toma de medidas y de comunicación con el otro Arduino, se realiza la media de todos los valores tomados para obtener V1_final, V2_final, V3_final, V4_final, anguloX_final, anguloY_final (los códigos completos se adjuntan en los Anexos). Observamos que se han hecho llamadas a otra subfunción, *transforma2*: realiza la conversión de los valores que nos ha enviado el otro Arduino a valores que nos son útiles: 0-1024.

5.3 Código: *espiral*

Podríamos dar por terminado todo el proceso llegados a este punto; sin embargo, para optimizar el funcionamiento del proyecto vamos a implementar un algoritmo de ajuste fino para asegurarnos que estamos tomando la medida en el punto exacto de mayor irradiancia. La idea es tomar diez medidas en puntos muy próximo entre sí que, en conjunto, dibujan una espiral en el espacio con un radio de 2º. Se irán comparando los valores de cada medida y se almacenará el mayor de todos ellos.

Como vemos en el esquema de la Figura 5-1, dentro de esta última función se hacen, a su vez, llamadas a dos de las funciones ya descritas anteriormente: *movimiento_servos* y *espiral*. Esto tiene como objetivo el mover el enfoque del sensor de medición sutilmente alrededor del punto actual, y ahí tomar la medida que se comparará con la que ya tenemos.

Primero creamos las variables que vamos a necesitar. A la función le llegan los valores V1, V2, V3, V4 que se almacenaron en la función *medicion* tras hacer la media de todas las medidas tomadas. Además, se crean dos vectores que almacenan los grados de movimiento que realizará el aparato antes de cada medida, ya que este movimiento en espiral será el mismo siempre.

```
void espiral (float V1_final, float V2_final, float V3_final, float V4_final) {
  int servo1[10]={0,1,1,-1,-1,2,2,-2,-2,2};
  int servo2[10]={0,0,-1,-1,1,1,-2,-2,2,2};
  float maximo1=V1_final,maximo2=V2_final,maximo3=V3_final,maximo4=V4_final;
  float altitud_espiral;
  float acimut_espiral;
  int n;
```

Una vez definido esto, se entra en el bucle for, que se repite 10 veces leyendo cada vez una componente diferente de los vectores servo1 y servo2. Estos valores son los que se mandarán como altitud y acimut a la función movimiento_servos en cada paso por el bucle, y tras el movimiento se realizará la toma de medidas. Dado que los valores

```
for (n=1; n<=10; n++){
    //PRIMERO MOVIMIENTO
    acimut_espiral = servo1[n];
    altitud_espiral = servo2[n];
    Servos(altitud_espiral, acimut_espiral);

    //SEGUNDO MEDIDA
    medicion ();
```

Por último, se toma la medida en esta nueva posición y se compara con el valor que había almacenado hasta ahora, actualizándose sólo si es mayor:

```
//TERCERO COMPARACIÓN Y ALMACENAMIENTO
 matriz[j][0] = j;

 if (V1_final> maximo1) {
   matriz[j][1]=V1_final;
   maximo1=V1_final;
}
 if  (V2_final> maximo2) {
   matriz[j][2]=V2_final;
   maximo2=V2_final;
}
 if (V3_final> maximo3) {
   matriz[j][3]=V3_final;
   maximo3=V3_final;
}
 if (V4_final> maximo4) {
   matriz[j][4]=V4_final;
   maximo4=V4_final;
  }
```

Al finalizar el bucle, se aumentará en uno el valor de la variable n y se repetirá desde arriba. Una vez se haya completado el bucle, tendremos la medida más alta almacenada en la matriz, que constará de cinco columnas e irá completándose con los valores máximos en cada vuelta al bucle de la funcion_global que, tal y como vimos, se repite cada 60 segundos. Esta matriz tiene un tamaño de 480 filas, pensadas para poder almacenar las variables durante 8 horas seguidas de seguimiento solar (60 * 8 = 48) y cinco columnas. Su aspecto tras este periodo será como el que vemos en la Figura 5-3. Cabe aclarar que en esta matriz se pueden almacenar los valores que nos vayan a ser más útiles para los resultados que posteriormente nos interesen. Ahora mismo el programa marca el almacenamiento de los cuatro valores por separado, pero dado que ya está implementado en el código el cálculo, por ejemplo, de Fx, Fy, AnguloX, AnguloY, etc., se podrían almacenar estos valores por ejemplo de cara a graficar el ángulo de incidencia solar durante las horas de la experimentación. Igualmente, sería interesante comprobar si los resultados son más fieles a la realidad ejecutando la función en espiral tal y como se ha explicado o, quizás, quedándonos con el punto en el que la suma de los cuatro valores sea máxima, lo cual supondría modificar sutilmente algunas líneas.

Minuto de la muestra	Vph1	Vph2	Vph3	Vph4
1	Valor máximo sensor 1 en el primer minuto	Valor máximo sensor 2 en el primer minuto	Valor máximo sensor 3 en el primer minuto	Valor máximo sensor 4 en el primer minuto
2	Valor máximo sensor 1 en el segundo minuto	(…)	(…)	(…)
3	Valor máximo sensor 1 en el tercer minuto	(…)	(…)	(…)
(…)				

Figura 5-3. Tabla simulación de la matriz almacenamiento

6 RESULTADOS EXPERIMENTALES

6.1 Introducción

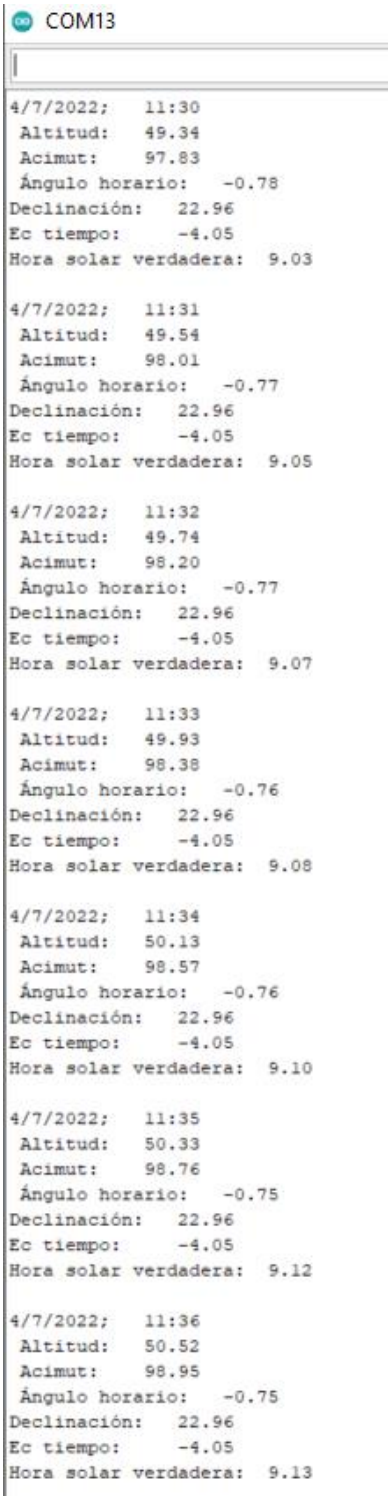
Como se ha ido apuntando, la puesta en marcha de todo este proyecto tiene la idea de hacer pruebas experimentales de larga duración. Para ello, todas estas funciones deberán ir repitiéndose cada cierto tiempo. Tras probar el periodo de tiempo idóneo, se configuró el bucle para ser repetido cada 60 segundos, tal y como se vio en el apartado 2.2.1.

Cada una de las funciones han sido codificadas y su funcionamiento verificado por separado. Sin embargo, para realizar las pruebas experimentales deseadas es necesario poder implementar todas las funcionalidades desarrolladas a la vez, dotando al Pan&Tilt de la independencia que necesita para realizar el seguimiento del Sol y la toma de medidas. Uno de los elementos principales del proyecto, el Pan&Tilt, ha sido el causante de que no se hayan podido completar las pruebas conjuntamente. Desde el principio de esta investigación no se nos entregó ningún tipo de documentación, esquemas de conexionado, SW ni tan siquiera alguna explicación sobre su funcionamiento por parte de la empresa a la que se le encargó el pedido del aparato. Es por eso que el proyecto se complicó y alargó más de lo esperado, ya que aunque se consiguió poner en marcha y montar el proyecto completo – ensamblar con el resto de componentes y realizar la debida programación y comunicación entre todos ellos, como se ha desarrollado en este documento- el Pan&Tilt comenzó a dar problemas de HW desde el primer momento. Si bien es cierto que la empresa se ocupó de lo ocurrido un par de veces, llegados a un cierto punto de la investigación (cuando solo quedaba la tarea de realizar las pruebas experimentales) el Pan&Tilt fue perdiendo rendimiento: ciertas conexiones dejaron de existir, el Arduino Pro Mini –que trasladaba los datos de las medidas solares- dejó de comunicarse con el resto del Pan&TIlt, etc., hasta que en las últimas semanas de proyecto, el Pan&Tilt dejó de encenderse, eliminando de raíz así la posibilidad de entregar ninguna prueba gráfica o audiovisual del funcionamiento conjunto de las funciones desarrolladas.

Así, en este apartado se va a mostrar el funcionamiento por separado de todas las funciones posibles (calculo_altitud_acimut, posicion_actual, medicion). La función más importante que no podrá mostrarse, aunque está probado que funciona correctamente y su detallada explicación y codificación lo puede demostrar, es la función movimiento_servos, que en tan solo unas líneas envía al servo al lugar buscado. Así mismo ocurre con la función espiral que, como vimos en la Figura 5-1, hace llamadas tanto a medicion como a movimiento_servos, y que tampoco podrá ser probada en este documento, ya que no se tomaron muestras durante la programación sino que se esperó a hacerlo con el dispositivo al completo.

6.2 Resultados cálculos elevación, azimut

Para hacer una muestra de los cálculos de elevación y azimut, se presentarán los resultados obtenidos al ejecutar nuestra función en un día y una hora determinados. Así mismo, se presentarán los resultados obtenidos en una web de cálculo de ecuaciones solares a tiempo real (24), para comprobar el correcto funcionamiento del código y la variedad de información que nos aporta. Como se dijo, el bucle se ejecuta a tiempo real y se actualizan los valores cada minuto, siendo así como lo vemos en la figura a continuación.



sol" posición	Elevación	Azimut	latitudes	longitudes
04/07/2022 11:30 | GMT1	49.21°	97.89°	37.3886303° N	5.9953403° W
crepúsculo	**Sunrise**	**Puesta de sol**	**Azimut Sunrise**	**Azimut Puesta de sol**
crepúsculo -0.833°	07:08:19	21:48:22	59.97°	299.95°
crepúsculo civil -6°	06:37:28	22:19:07	55.09°	304.82°
Náutica" crepúsculo -12°	05:59:00	22:57:30	48.49°	311.4°
El crepúsculo astronómico -18°	05:15:49	23:40:33	40.27°	319.58°
la luz del día	**hh:mm:ss**	**diff. dd+1**	**diff. dd-1**	**Mediodía**
04/07/2022	14:40:03	-00:00:44	00:00:40	14:28:20

sol" posición	Elevación	Azimut	latitudes	longitudes
04/07/2022 11:33 | GMT1	49.8°	98.45°	37.3886303° N	5.9953403° W
crepúsculo	**Sunrise**	**Puesta de sol**	**Azimut Sunrise**	**Azimut Puesta de sol**
crepúsculo -0.833°	07:08:19	21:48:22	59.97°	299.95°
crepúsculo civil -6°	06:37:28	22:19:07	55.09°	304.82°
Náutica" crepúsculo -12°	05:59:00	22:57:30	48.49°	311.4°
El crepúsculo astronómico -18°	05:15:49	23:40:33	40.27°	319.58°
la luz del día	**hh:mm:ss**	**diff. dd+1**	**diff. dd-1**	**Mediodía**
04/07/2022	14:40:03	-00:00:44	00:00:40	14:28:20

sol" posición	Elevación	Azimut	latitudes	longitudes
04/07/2022 11:36 | GMT1	50.39°	99.01°	37.3886303° N	5.9953403° W
crepúsculo	**Sunrise**	**Puesta de sol**	**Azimut Sunrise**	**Azimut Puesta de sol**
crepúsculo -0.833°	07:08:19	21:48:22	59.97°	299.95°
crepúsculo civil -6°	06:37:28	22:19:07	55.09°	304.82°
Náutica" crepúsculo -12°	05:59:00	22:57:30	48.49°	311.4°
El crepúsculo astronómico -18°	05:15:49	23:40:33	40.27°	319.58°
la luz del día	**hh:mm:ss**	**diff. dd+1**	**diff. dd-1**	**Mediodía**
04/07/2022	14:40:03	-00:00:44	00:00:40	14:28:20

Figura 6-1. Verificación cálculos solares

6.3 Resultados posición actual

Siguiendo con el procedimiento de muestra de resultados de manera independiente, vamos a comprobar la calidad de la orientación del sistema de brújulas desarrollado. Para ello, mostramos en la Figura 6-2 un gráfico que representa el tiempo en el eje X y el ángulo que nuestra brújula mide cuando se orienta en dirección al Norte, Sur, Este y Oeste, expresado en º.

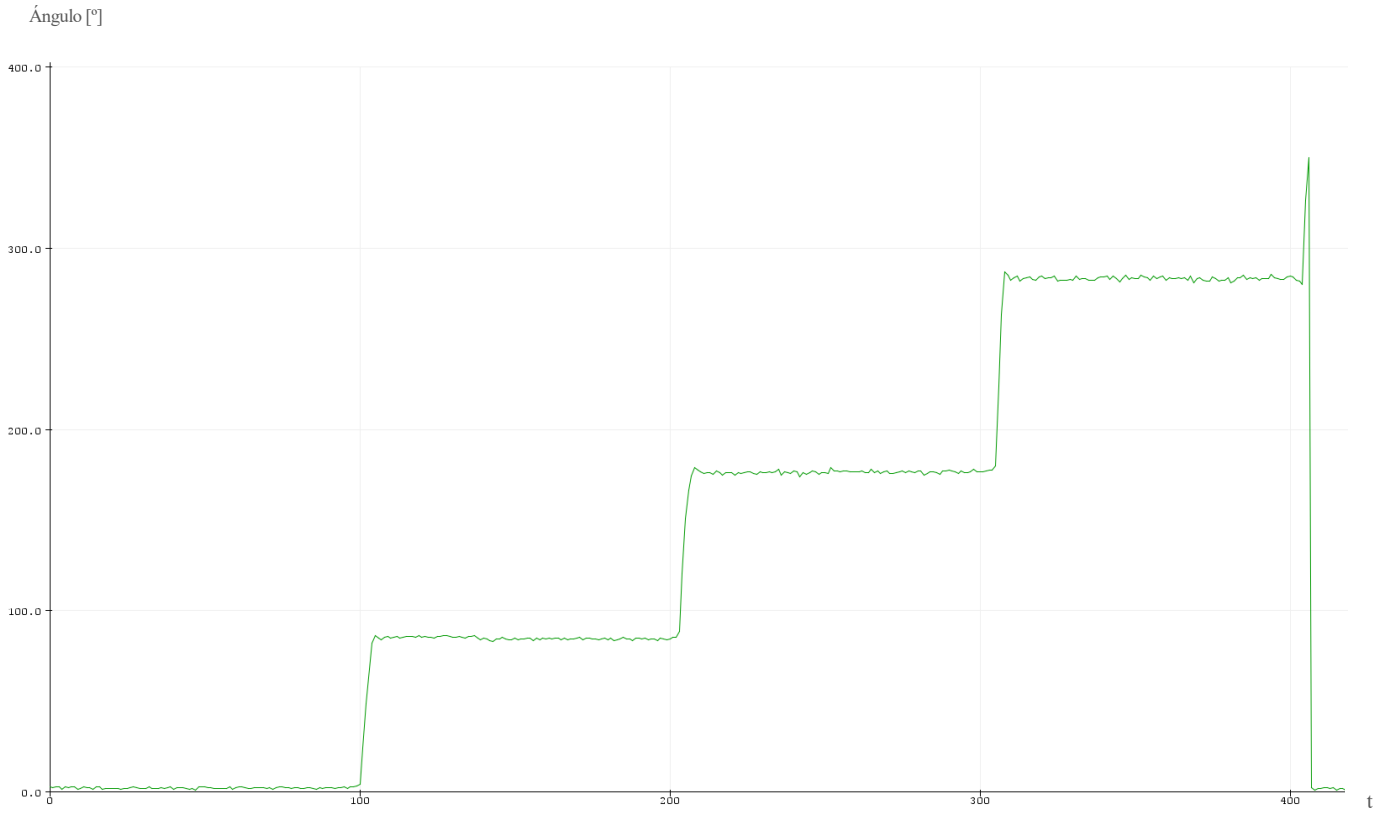


Figura 6-2. Gráfica ángulos medidos por la brújula de Arduino

Observamos que la brújula empleada para las pruebas es capaz de mantener estabilidad en los ángulos correspondientes: 0º, 90º, 180º, 270º, y que vuelve al valor nulo tras completar la circunferencia completa. Si bien podemos observar algo de ruido durante el tiempo que la brújula pasa estable. Es por ello que en el código se añadieron en su día unas líneas para tomar el valor medio entre diferentes medidas tomadas en un instante.

6.4 Resultados medicion solar

El sensor Solar MEMS consta de cuatro celdas de medición solar, dispuestas como vemos en la Figura 6-3. Vamos a probar, primero, que los cuatro sensores individualmente funcionan como es debido. Se hará primero un pequeño análisis del funcionamiento del sensor con luz artificial, y luego se mostrarán algunos ejemplos de la luz medida con la radiación solar.

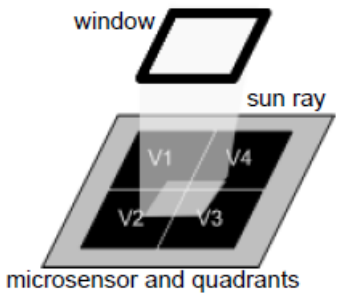


Figura 6-3. Cuadrantes Solar MEMS

Como vemos en el DataSheet, si la luz entra por una diagonal el sensor que tenga más cerca será el que menos luz reciba. El que más, será el que esté en la diagonal opuesta. Los dos sensores que queden a ambos lados recibirán una medida parecida si la luz viene totalmente en la diagonal. Vemos en las gráficas a continuación que el sensor que se ha implementado en el proyecto cumple con lo previsto. El título de cada gráfica indica por cuál de los cuatro sensores está entrando la luz artificial que se empleó para las pruebas. Por ejemplo, en la gráfica de la Figura 6-4, la luz entró por la diagonal del sensor V1. Así, es la línea amarilla que representa la luz medida por S3 (tal y como vemos en la leyenda) la que recibe mayor intensidad luminosa; V1, en azul, el sensor que menos recibe.

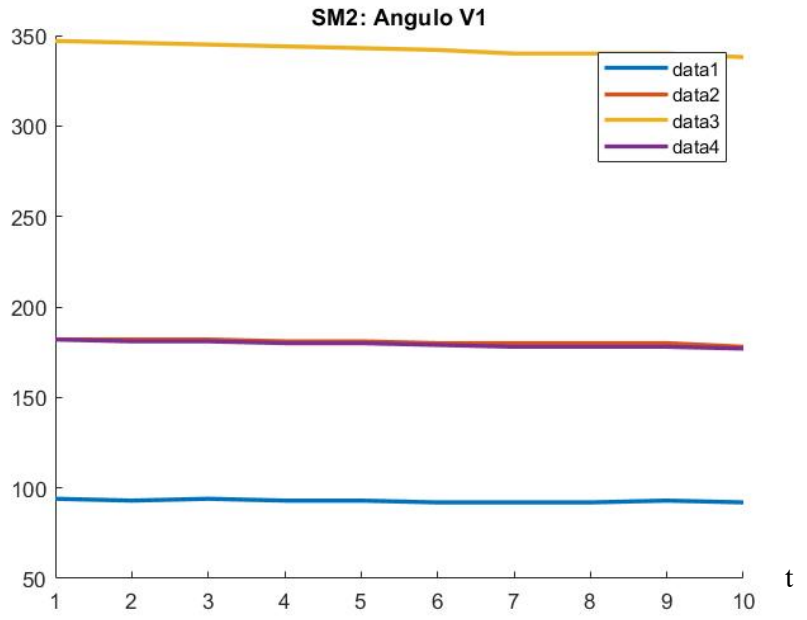


Figura 6-4. Gráfica medición solar, entrada de luz diagonalmente por la esquina del sensor V1

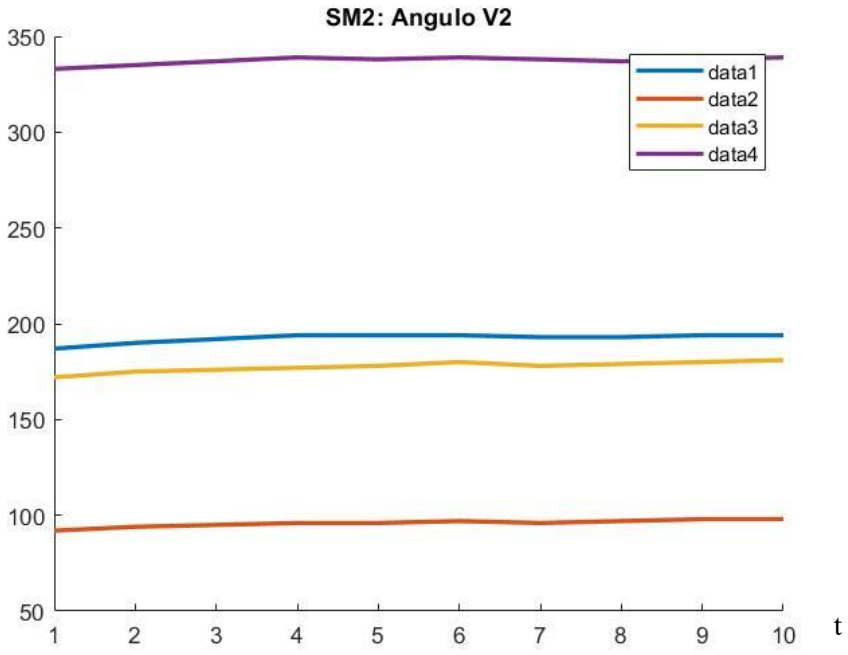


Figura 6-5. Gráfica medición solar, entrada entrada de luz diagonalmente por la esquina del sensor V1

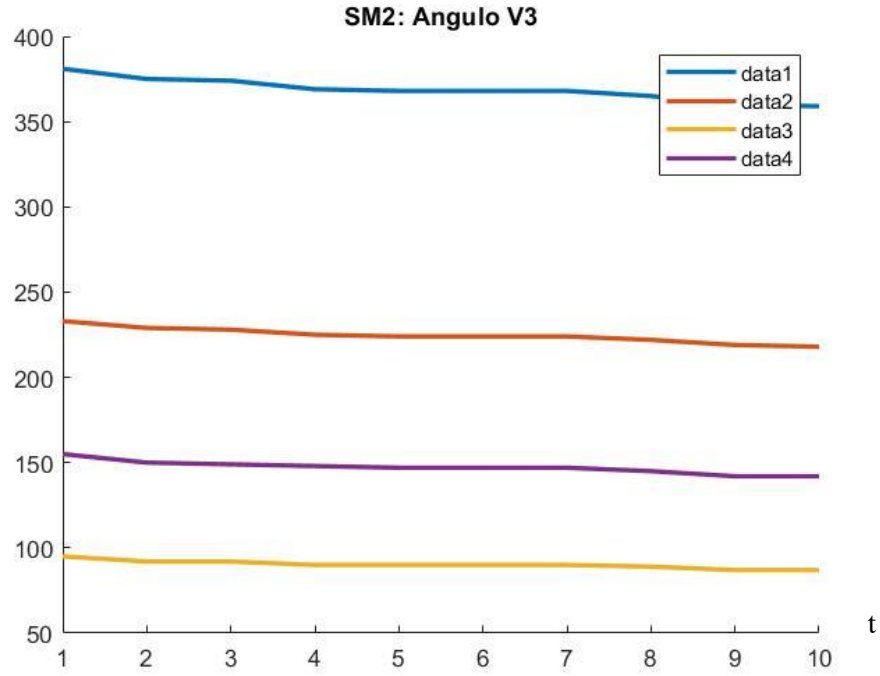


Figura 6-6. Gráfica medición solar, entrada entrada de luz diagonalmente por la esquina del sensor V1

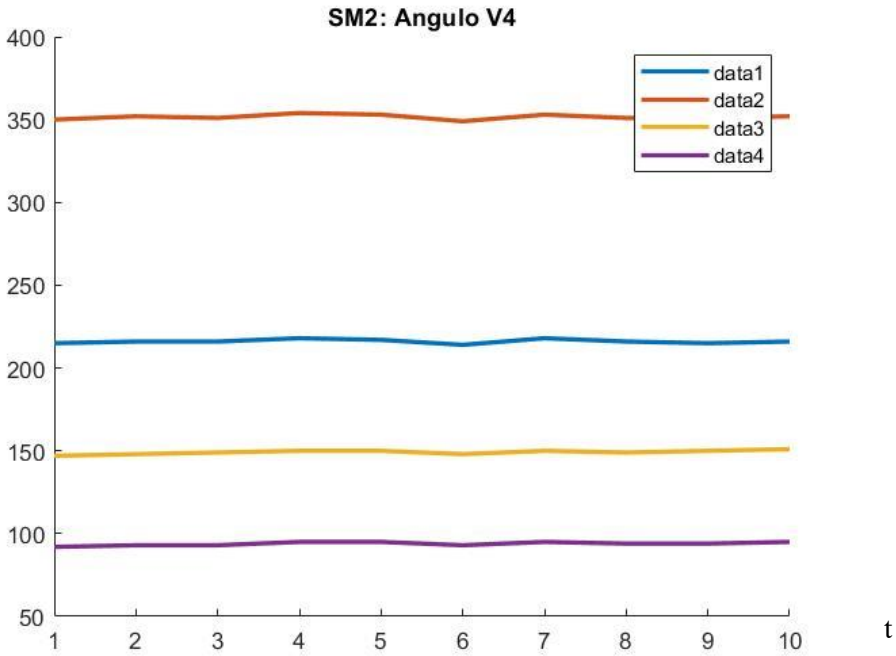


Figura 6-7. Gráfica medición solar, entrada entrada de luz diagonalmente por la esquina del sensor V1

Vemos que el pronóstico se cumple para las cuatro pruebas, validando así el funcionamiento de nuestro sensor. Además, el valor numérico nos indica la cantidad de luz medida en un rango 0-1023, que podrá luego transformarse al valor 0-5V. Vemos que el valor medido en el sensor de mayor luminosidad en cada caso está entre los 300-350, mientras que al sensor al que menos luz le llega está en todos los casos un poco por debajo de 100.

6.5 Dificultades encontradas

Las dificultades encontradas durante el desarrollo del presente proyecto no han sido pocas, ni fáciles de resolver. Pese a que la programación de los códigos y la búsqueda e instalación de las librerías y drivers necesarios no ha sido una tarea sencilla, la realidad es que todos los problemas que han resultado limitantes para alcanzar los propósitos del proyecto han estado relacionados con el HW. A continuación se hará un breve repaso de las dificultades que han ido apareciendo de forma cronológica desde el principio de la investigación.

Originalmente, con lo único con lo que contábamos era con el Pan&Tilt que la empresa Dronetools había elaborado para el Departamento. Sin embargo, como hemos visto anteriormente, esto es una estructura cerrada que en su interior cuenta con placas, conexiones y elementos ensamblados a nivel profesional, ya que la empresa se dedica al desarrollo de drones de carreras, y elaboró un Pan&Tilt independiente para este proyecto. Consecuentemente, este fue entregado sin ningún tipo de documentación que especificara nada sobre su funcionamiento, programación, conexiones, etc., o que nos dotara de independencia a la hora de solventar posibles problemas de HW que pudieran acontecer. De hecho, el dispositivo contaba con conexiones erróneas al principio que fueron muy difíciles de detectar y corregir, ya que no teníamos idea sobre su funcionamiento. Así mismo, el Pan&Tilt cuenta con un Arduino, como ya hemos visto, que traía el código de lectura de datos instalado. Sin embargo, tampoco se nos proporcionó este código hasta pasadas muchas semanas de proyecto, de forma que fue difícil crear el código complementario que enviara y recibiera toda la información (ya vimos que hay una comunicación constante entre ambos Arduinos). Otro de los principales problemas que tuvimos con este dispositivo antes de poder montar y programar el resto de elementos estuvo relacionado con su temperatura de funcionamiento. Teóricamente, el Pan&Tilt debía venir ajustado para funcionar sin problema. Sin embargo, se notó que las temperaturas alcanzaban valores altísimos – tal y como vemos en la Figura 7.1 – a los pocos minutos de funcionamiento, que impedían su puesta en marcha durante más de 4 o 5 minutos seguidos.

Figura 6-8. Medición temperatura Pan&Tilt

Una vez esta primera fase de problemas con el Pan&Tilt fue superada, comenzamos a dedicarnos al resto de componentes, donde fueron apareciendo nuevas dificultades, más accesibles. Con respecto al sensor Solar Mems, empezamos obteniendo muy buenos resultados numéricos por parte del sensor y sus cuatro fotoreceptores. Sin embargo, llegado un cierto momento, estos dejaron da calcular correctamente el ángulo Y. Cuando la luz le llegaba en círculos y se esperaba que la gráfica fuera una curva sinusoidal entre $\pm 62^{\circ}$ (este es el ángulo que recibe como ángulo máximo), obteníamos una curva como la que vemos en la Figura 7-2: además de tener parones y fallos en las medidas de forma aislada, en los momentos en los que el ángulo de incidencia era de 0º, el sensor medía -40º. Tras numerosas pruebas e intentos de arreglo y pese a nuestra intención y esfuerzo por evitarlo, nos hicimos con un nuevo SM que volvió a realizar los cálculos de forma adecuada.

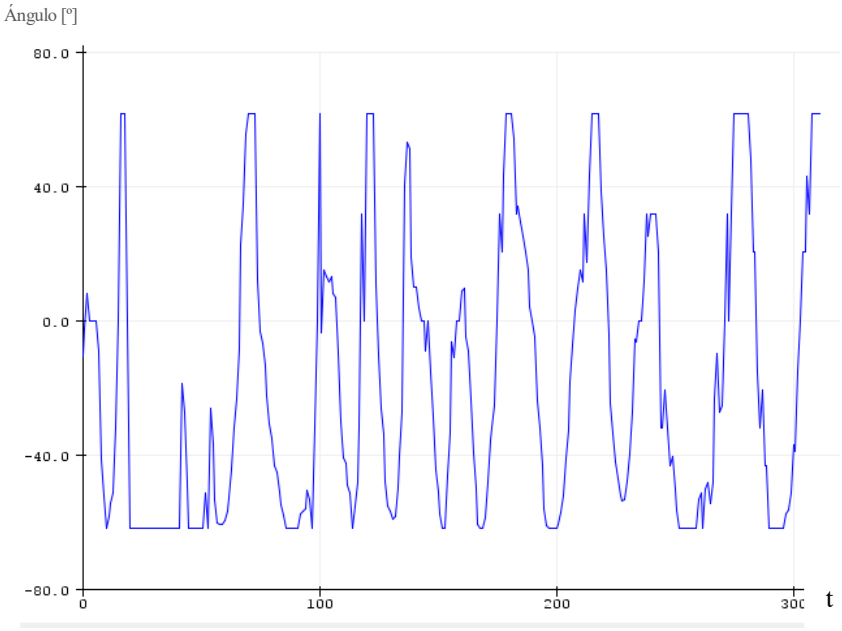


Figura 6-9. Gráfica medida errónea ángulo Y

Ahora sí, tuvimos un periodo de tiempo en el que los elementos HW parecía que funcionaban y pudimos desarrollar la mayoría de los códigos sin demasiadas interrupciones por problemas técnicos. Si bien es cierto que una mañana el Pan&Tilt dejó de encenderse. Tratamos de ponernos en contacto con la empresa y, finalmente, accedieron a revisar el Pan&Tilt. Tras desmontar toda su estructura y deshabilitar las conexiones, cables, etc., resultó que una de las piezas que habían instalado dentro del dispositivo estaba dando fallo e impedía todo funcionamiento del mismo. Se trata de una pieza muy específica, en forma de corona, que permitía cerrar las conexiones en cualquier ángulo de giro. Dado que la empresa se dedica a esto, tenían esta pieza tan específica para sustituirla y poder continuar con el proyecto.

Así que de nuevo continuamos con nuevos avances, en este caso llegando casi al final del proyecto: la instalación de la brújula. Se plantearon varios tipos de brújulas y finalmente nos decantamos por la que se ha descrito en la memoria. Pese a que finalmente tiene un funcionamiento aceptable, la realidad es que se han tenido que pedir hasta 9 brújulas diferentes, de las que sólo 3 han funcionado. Dos de ellas, además, dejaron de responder cuando fueron instaladas en el dispositivo, pese a que se tuvieron en cuenta los campos electromagnéticos y se posicionaron separada del resto de elementos, como vimos en la Figura 2-17. Esto se detectó porque sus ángulos comenzaron a desviarse. En la siguiente Figura se han graficado los ángulos que una de estas dos brújulas desconfiguradas marcaba al apuntarla a N, S, E, O tras haber sido instalada en el sistema completo.

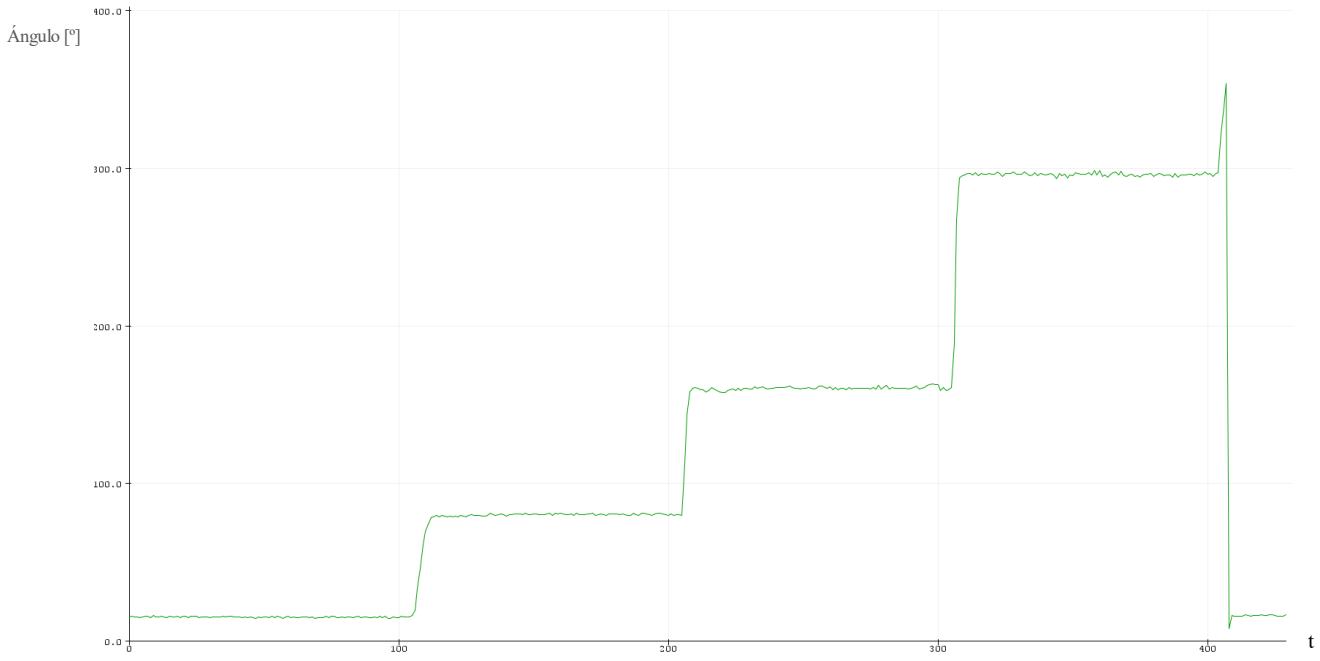


Figura 6-10. Gráfica medidas angulares brújula desconfigurada

Vemos cómo la brújula marca un ángulo de 14º cuando apunta al Norte y debería marcar 0º. Este error va incrementándose conforme la brújula va girando, de forma que cuando el ángulo real es de 270º, la brújula señala casi 300º. Es por esto que esta brújula se desestimó y se usó, finalmente, la única que quedaba que daba buenos resultados. Estos resultados, aún así, se hacen pasar por un filtro que nos da un valor medio que sí que concuerda con precisión con el ángulo real. Este filtro es necesario porque, aunque funciona de forma esperada y como decimos el resultado es bueno, si nos fijamos con detalle en las medidas que aporta vemos en la Figura 7-4 que estas tienen mucho ruido aún cuando la brújula está totalmente quieta durante un largo periodo de tiempo.

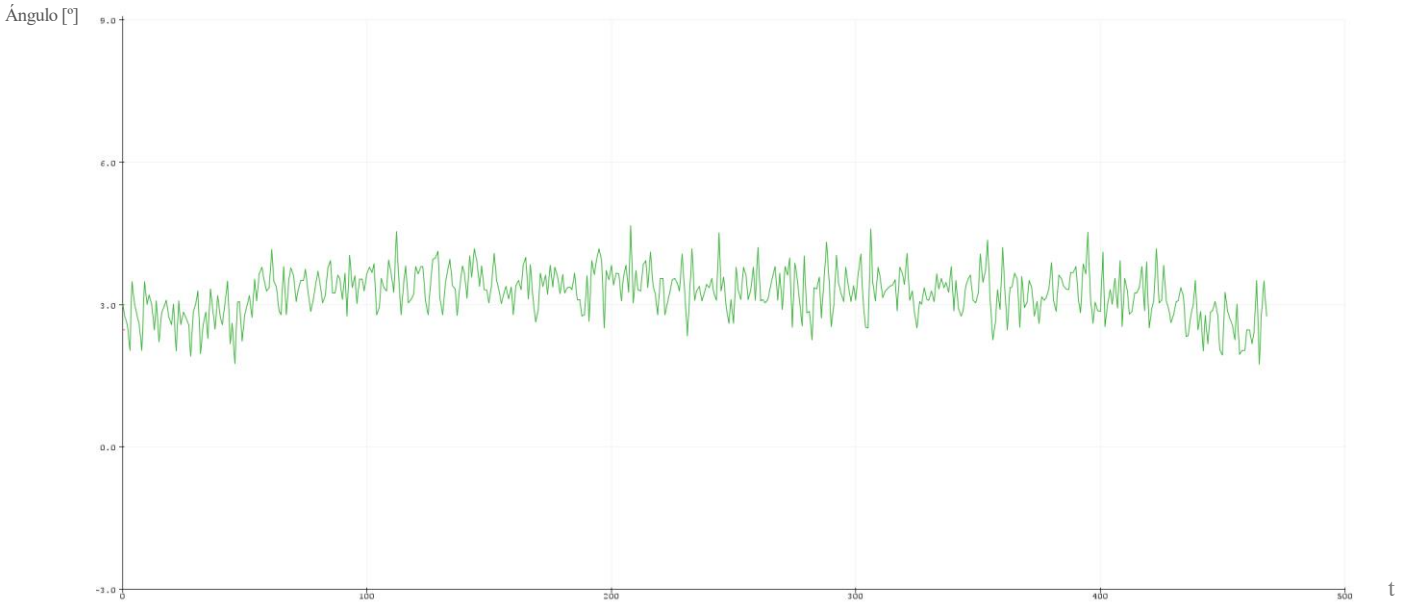


Figura 6-11. Ruido de las medidas angulares de la brújula instalada

Tras el filtro que se implementó, la orientación del gimbal era la adecuada. Aún así, para perfeccionar el punto exacto de mayor irradiación se creó otro algoritmo de ajuste fino del que ya se habló anteriormente. Se trató, así mismo, de conocer la información de si existía alguna placa en el interior de la estructura que pudiera, de alguna manera, leer el ángulo de giro. Sin embargo no se nos proporcionó esta información por parte de la empresa así que no se pudo continuar investigando por esa vía.

Una vez hechos los reajustes de código propios tras la puesta en marcha de la estructura, el gimbal conseguía girar y tomar las medidas tal y como se había previsto. Se dedicaron algunos días a perfeccionar estos giros con el gimbal antes de realizar las pruebas y documentación finales, y fue en este tiempo cuando empezaron a fallar de forma global las funciones del Pan&Tilt. En las últimas semanas de investigación, las conexiones internas y el envío de comunicación por parte del Arduino Pro Mini (instalado en el insterior del Pan&Tilt) cesaron por completo, y en este momento la empresa se desresponsabilizó completamente del mantenimiento del aparato. Mientras se intentaba buscar alguna solución a este fallo ya casi total de la estructura, de cara a poder realizar experimentos reales y concluir así con la investigación, el Pan&Tilt volvió a dejar de encenderse, tal y como había ocurrido semanas atrás. Ya se venía notando la pérdida de rendimiento de su funcionamiento, pero a partir de que se dejó de encender, el aparato cesaba por completo de cumplir sus funciones. Se volvió a intentar recibir mantenimiento o documentación sobre las conexiones internas por parte de la empresa, pero no se obtuvo respuesta. Sobra decir que cada una de estas interrupciones suponían un retraso sobre el plan temporal inicial del proyecto, que pretendía haber sido completado (incluyendo gráficos y pruebas experimentales) con mayor brevedad de lo que finalmente se pudo.

En este apartado se han tratado de reflejar de forma resumida los problemas de HardWare que ha sufrido este proyecto desde sus inicios: muchos de ellos propios de una investigación en la que se implementa de forma física lo que se está investigando; otros muchos de ellos, derivados de la desinformación y falta de mantenimiento y de documentación del aparato principal, que ha sido donde se ha absorbido más tiempo de forma inesperada y lo que, finalmente, ha impedido que se pudieran concluir las pruebas experimentales tal y como se pretendía.

7 CONCLUSIONES

7.1 Conclusiones

Como conclusión principal tras el desarrollo del proyecto vemos que no es viable utilizar más Pan&Tilt como este para instalarlos en la planta real e integrarlos en su funcionamiento diario, ya que requerirían de un mantenimiento excesivo y retrasarían constantemente la actividad de la planta. La idea es seguir esta misma línea de investigación con otros dos Pan&Tilt diferentes a este, y comparar así los resultados y el rendimiento que cada uno de ellos ofrece. A la vista está que este modelo no cumple con los requisitos que se necesitan de él, al menos para ser instalados en robots móviles terrestres. Pese a que quizás se deba a que estos Pan&Tilt están originalmente diseñados para ser utilizados en drones, la realidad es que en esta aplicación en particular habría que hacer numerosos cambios para poder implementarlos, por lo que pensamos que probablemente el uso de otros Pan&Tilt más sencillos, implementando eso sí el código aquí desarrollado, dará resultados más satisfactorios. Por lo tanto, concluimos que el trabajo realizado no ha sido en vano ya que el código, que es uno de los trabajos principales que se ha realizado, es correcto y tienen un buen funcionamiento, de manera que podrá ser implementado en un Pan&Tilt que tenga un mejor rendimiento. Además, la red de conexiones diseñada e implementada es capaz de transmitir en tiempo real todos los datos y reaccionar en consecuencia, por lo que vemos que aunque uno de los elementos de HW principales ha resultado fallido, el conjunto de la aplicación es correcta y funciona tal y como se esperaba, de manera que la única modificación será implementar este mismo proyecto con un dispositivo Pan&Tilt diferente.

7.2 Líneas de trabajo futuro

Sobre la posible continuación de un proyecto como este, sobra decir que la primera tarea sería la de utilizar un Pan&Tilt que no retrasara la investigación constantemente y del que se pudiera obtener información y/o mantenimiento. Una vez hecho esto y completadas las pruebas experimentales, podrían sacarse numerosas concluiones muy útiles para el proyecto global al que pertenece esta investigación.

Para empezar, el siguiente paso sería instalarlo en un robot terrestre. En concreto, en nuestro proyecto se trabaja con RosBot: robots terrestres que se desplazan de forma autónoma y que pueden incluso crear flotas de robots que se comunican entre ellos. Para ello se implementaría la conexión entre Arduino y ROS, ya existente, y según el valor de irradiación medida por cada robot en cada zona de la planta solar, se aumentaría o disminuiría la temperatura del flujo de aceite en esa zona. Esto puede cambiar de forma instantánea por el paso de nubes de mayor o menor tamaño, y aquí está la idea de la optimización de la planta.

Así mismo, tomando las medidas tomadas por el Solar Mems en el punto de mayor irradiancia solar se podría por ejemplo realizar un pequeño estudio comparativo con las medidas oficiales de la DNI (*Direct Normal Irradiance),* y encontrar así la transformación de las medidas que tomen nuestros sensores a W/m^2. Estos dos pasos, que se encuentran bastante próximos al trabajo que se ha realizado en esta investigación, supondrán un uso real del código desarrollado en nuestra planta.

7.3 Valoración personal

A nivel personal tanto esta investigación como todas las personas que han colaborado a lo largo de su desarrollo, han supuesto un gran aprendizaje para mí. Si bien se comenzó sin tener idea alguna sobre el funcionamiento del dispositivo que se nos ofreció, a día de hoy el camino recorrido ha sido largo y contamos con un amplio documento descriptivo sobre cómo realizar estas tareas. Además, este proyecto tiene una gran importancia dentro de OCONTSOLAR, ya que como se ha ido desarrollando, el conocimiento de la irradiancia solar exacta a tiempo real en toda la superficie de la planta nos permitirá implementar diferentes estrategias de Control desarrolladas por compañeros del grupo, que nos permitirán optimizar el funcionamiento de la planta. Sin embargo, es evidente que los múltiples imprevistos que ha ocasionado el Pan&Tilt y las consecuentes llamadas,

esperas y faltas de entendimiento con la empresa al final de la investigación, han supuesto para mí una sensación creciente de impotencia y agotamiento, que se ha traducido finalmente en la no realización de las pruebas experimentales que se esperaban para concluir la investigación.

Con todo, ha sido un proyecto muy satisfactorio que nos ha aportado información relevante para el grupo y la investigación general, de manera que me siento complacida de haber podido dedicarme a ello en los últimos meses.

REFERENCIAS

1. Mestre, A. (1994) «¿Cómo se mide la radiación solar?», Revista de meteorología, 4, 15-18.
2. Planas, O. Solar-energia (2019). https://solar-energia.net/que-es-energia-solar/radiacion-solar/irradiacion-solar.
3. «KippZonen. OTT HydroMet» (2022), https://www.kippzonen.es/ProductGroup/85/Instrumentos-Solares.
4. «Wikipedia» (2019), https://es.wikipedia.org/wiki/Pirheli%C3%B3metro#cite_note-2.
5. Machado Toranzo, N., Lussón Cervantes, A., Leysdian Oro Carralero, L., Bonzon Henríquez, J., Escalona Costa, O., (2015) «Seguidor Solar, optimizando el aprovechamiento de la energía solar», Energética, 36(2), 190-199, ISSN 1815-5901.
6. Arencibia-Carballo, G. (2016) «La importancia del uso de paneles solares en la generación de energía eléctrica.», RedVet, 17.
7. «Arsus Energia S.L.», https://arsusenergia.es/centrales-termosolares-su-funcionamiento-y-tipos/.
8. «Wikipedia» (2022), https://es.wikipedia.org/wiki/Heliostato.
9. «CORDIS – Resultados de Investigaciones de la UE» (2020), https://cordis.europa.eu/project/id/789051/es.
10. D. Frejo, J. R., F. Camacho, E. (2020) «Centralized and distributed Model Predictive Control for the maximization of the thermal power of solar parabolic-trough plants», Solar Energy, 204, 190-199.
11. Masero, E., D. Frejo, J. R., Maestre, J.M., F. Camacho, Eduardo (2021), «A light clustering model predictive control approach to maximize thermal power in solar parabolic-trough plants», Solar Energy, 214, 531-541
12. Masero, E., Fletscher, L. A., Maestre, J.M. (2020), «A coalitional model predictive control for the energy efficiency of next-generation cellular networks», Energies, 13(24), 6546.
13. Aguilar-López, J. M., García, R. A., Camacho, E. F. (2021), «Algoritmo para la detección de formas aplicable a la estimación solar», Revista Iberoamericana de Automática e Informática Industrial RIAI, 18, 277-287.
14. Escaño, J. M., Sánchez, A. J., Ceballos, M., Gallego, A. J., F. Camacho, E. (2021), «Estimador Neuro-Borroso, con reducción de complejidad, de las temperaturas de un campo solar cilindro-parabólico», Revista Iberoamericana de Automática e Informática industrial RIAI, 18, 138.
15. Muros, F. J. (2021), «Coalitional control in the framework of cooperative game theory», Revista Iberoamericana de Automática e Informática Industrial RIAI, 18, 97.
16. Maestre, J. M., Chanfreut, P., García Martín, J., Masero, E., Inoue, M. y F. Camacho, E. (2021), «Control predictivo de sistemas ciberfísicos», Revista Iberoamericana de Automática e Informática Industrial RIAI, 19, 1-12.
17. «Arduino», https://arduino.cl/producto/arduino-pro-mini-328-3-3v-8mhz/.
18. «Arduino», https://arduino.cl/producto/arduino-mega-2560/.
19. «RC-Innovations», https://rc-innovations.es/bateria-lipo-tattu-funfly-3s-11.1v-1800mah-100c-ta-ff-100c-1800-3s1p-gens-ace.
20. «Wikipedia», https://es.wikipedia.org/wiki/Arduino_IDE.
21. «Basecam Electronics», https://www.basecamelectronics.com/serialapi/.
22. Sproul, A. B., (2007), «Derivation of the solar geometric relationships using vector analysis», Renewable Energy, 32, 1187-1205

23. «Photopills», https://www.photopills.com/es/articulos/entendiendo-el-azimut-la-elevacion#:~:text=La%20elevaci%C3%B3n%20es%20la%20distancia,el%20horizonte%20local%20del%20observador.
24. «SunEarthTools», https://www.sunearthtools.com/dp/tools/pos_sun.php?lang=es.
25. «Wikipedia», https://es.wikipedia.org/wiki/Elevaci%C3%B3n_solar.
26. F. Camacho, E., Berenguel, M. (2012), «Control of Solar Energy Systems», IFAC Proceedings Volumes, 45(15), 848-855.

ANEXOS

ANEXO A: Archivos programados

funcion_global

```
//LIBRERIAS

#include <TimeLib.h>
#include <MechaQMC5883.h>
#include <inttypes.h>
#include <SBGC.h>
#include <SBGC_Arduino.h>
#include <Wire.h>

#define SERIAL_SPEED 115200
#define SBGC_CMD_DELAY 20

//VARIABLES

  unsigned long periodo=60000; //60 secs
  unsigned long Tiempoahora=0;

  //1. CALCULO ALTITUD, AZIMUT

  void ecs_solares(int hor, int minu, int sec, int mes, int dia, float latitud, float longitud);

  int hor=13;
  int minu=05;
  int sec=00;
  int mes=5;
  int dia=22;

  int j=0; //Es la variable que voy a utilizar para almacenar los datos en la matriz

  //SEVILLA
  float latitud=37.38283*(PI/180); //En RAD
  float longitud=-5.973717; //En °
  float altitud;
  float acimut;
  float acimut_corregido;
  float declinacion;
```

```
  //2 y 3. CÁLCULO POSICION ACTUAL Y ÁNGULO DE MOVIMIENTO

  float posicion_actual (float declinacion);
  void Servos(float angulo, float altitud, float acimut);

  int16_t x,y,z;
  float angulo_inicial=0;
  float suma_inicial=0;
  float angulo_final;
  float angulo_actual;
  float error=100; //Inicializado a 100 para que entre al bucle
  SBGC_cmd_control_t c = { 0, 0, 0, 0, 0, 0, 0 }; //mode, speedROLL, angleROLL, speedPITCH, anglePITCH, speedYAW, angleYAW

  MechaQMC5883 qmc;

   //4. Toma medidas irradiación

  void medicion ();
  float V1_final, V2_final, V3_final, V4_final, anguloX_final, anguloY_final;
  float matriz[480][5]; //Almacenamiento de los 4 valores de cada medida realizada


  //5. Ajuste fino (ESPIRAL)

 void espiral (float V1_final, float V2_final, float V3_final, float V4_final);



void setup() {

  Serial.begin(57600);
  while (!Serial) {}

  Serial1.begin(57600); //Solar MEMS
  Serial3.begin(SERIAL_SPEED); //Movimiento Pan&Tilt
  SBGC_Demo_setup(&Serial3);
  delay(3000);
  c.mode = SBGC_CONTROL_MODE_ANGLE;
  c.speedROLL = c.speedPITCH = c.speedYAW = 50 * SBGC_SPEED_SCALE;
  c.anglePITCH = SBGC_DEGREE_TO_ANGLE(-90);
  c.angleYAW = SBGC_DEGREE_TO_ANGLE(0);
  SBGC_cmd_control_send(c, sbgc_parser);

  if ( Serial.read() == 'a' ){ //Esperar a que el gimbal esté en la posición inicial para tomar medida

  //CALCULO ÁNGULO_INICIAL_REFERENCIA
  angulo_inicial = posicion_actual (declinacion);

  Wire.begin();
  qmc.init();

}
}


void loop() {

if(millis()> Tiempoahora + periodo){

    //Actualizo tiempo acutal en ms
    Tiempoahora=millis();
```

```
    //Oriento la altitud hacia arriba (el acimut lo dejo como esté ya que se puede cuantificar la posición)
    c.anglePITCH = SBGC_DEGREE_TO_ANGLE(-90);
    SBGC_cmd_control_send(c, sbgc_parser);

    //1. CALCULO ALTITUD, AZIMUT

     ecs_solares (hor, minu, sec, mes, dia, latitud, longitud); //Aquí dentro se calcula el acimut_corregido

    //2 y 3. BUCLE MOVIMIENTO PAN&TILT Y CALCULO POSICIÓN ACTUAL (ERROR <3°)

    while (error >= 3 || error <= -3) {

    Servos(altitud, acimut_corregido);
    angulo_actual = posicion_actual (declinacion);
    error = angulo_actual - acimut;
    }


    //4 y 5. TOMA MEDIDA IRRADIACIÓN FINAL, ALGORITMO AJUSTE FINO + ALMACENAMIENTO (ESPIRAL)

    medicion;
    espiral (V1_final, V2_final, V3_final, V4_final);

    j++; //Aumento en uno el valor de esta variable, para almacenar los próximos datos en una nueva fila de la matriz



    //ACTUALIZO VARIABLES PARA VOLVER A EMPEZAR

     minu=minu+1;
     if (minu>=60){
     hor=hor+1;
     minu=0;
     }
}
}


time_t ponFecha(int y, int m, int d, int hh, int mm, int ss){

    tmElements_t f;
    f.Second = ss;
    f.Minute = mm;
    f.Hour = hh;
    f.Day = d;
    f.Month = m;
    f.Year = y - 1970 ;
    return makeTime(f);      //Crea tiempo Unix

}
```

ecs_solares

```
  //1. CALCULO ALTITUD, AZIMUT

  int tabla[]={31,29,31,30,31,30,31,31,30,31,30,31}; //días que tiene cada mes del año
  int dias=0;
  time_t t0,t1,dif;

  //SEVILLA
  float latit=37.38283*(PI/180); //En RAD
  float longit=-5.973717; //En °


void ecs_solares (int hor, int minu, int sec, int mes, int dia, float latitud, float longitud){

  //Tiempo actual (año, mes, dia, hora, min, sec)
  t0 = ponFecha(2022, 1, 1, 0, 0, 0);
  t1= ponFecha(2022, mes, dia, hor, minu, sec);
  dif =  t1 - t0;
  float minutos= minute(dif) ;
  float resta= minutos/60;
  float horas=hour(dif)+resta;

  //Diferencia posición
  float diferenciapos= abs(longitud*4);

  //Diferencia por horario de verano
  int diferenciames;
  if(month(t1)<4 | month(t1)>10){ //1h si estamos en invierno
   diferenciames=60;}
  else{ //2h si estamos en verano
   diferenciames=120;
  }

  //Diferencia tabla de días del mes, por año bisiesto
  if(year(t1)%4==0){ //año bisiesto, febrero tiene 29 días
     tabla[1]=29;
    }
  else{
     tabla[1]=28;
    }


 //Cálculo del nº de días del año que llevamos, en la variable dias
 int suma=0;

 for(int i=0; i<(month(dif)-1);i++){ //dias de los meses completos
   suma=suma+tabla[i];
 }
 dias= suma + day(dif); //dias del mes incompleto en el que estamos
```

```
//CÁLCULOS REALES UNA VEZ TENEMOS TODOS LOS DATOS

  //Cálculo declinación (depende del día del año; equinoccio de primavera y otoño es 0°,
  //solsticio de verano e invierno es +-23.45°)
  float angulodiario=(2*PI/365)*(dias-1); //En 365 días gira 2pi
  declinacion=0.006918-0.399912*cos(angulodiario)+0.070257*sin(angulodiario)-0.006758*cos(2*angulodiario)
  +0.0009907*sin(2*angulodiario)-0.002697*cos(3*angulodiario)+0.00148*sin(3*angulodiario); //declinación en RAD
  float dec=declinacion*180/PI; //declinación en °

  //Ecuación del tiempo
  float et=(0.000075+0.001868*cos(angulodiario)-0.032077*sin(angulodiario)-0.014615*cos(2*angulodiario)
           -0.04089*sin(2*angulodiario))*229.18; //Diferencia entre el mvto aparente del sol y mvto medio (en minutos)
  float diferenciatotal=diferenciames+diferenciapos-et; //Diferencia total en minutos
  float Ts=horas-(diferenciatotal/60); //HORA SOLAR VERDADERA (minutos)


  //Ángulo horario (describe la rotación de la tierra alrededor de su eje)
  float w=((Ts-12)*15)*(PI/180); //En RAD

  //Altitud SOLAR
  altitud=asin(cos(latitud)*cos(declinacion)*cos(w)+sin(declinacion)*sin(latitud));

  //Acimut SOLAR
  acimut=acos((cos(latitud)*sin(declinacion)-cos(w)*sin(latitud)*cos(declinacion))/cos(altitud));

  altitud=altitud*(180/PI);
  acimut=acimut*(180/PI);

  if(w>0) acimut=360-acimut;

  acimut_corregido=acimut-angulo_inicial; //Cuánto tiene que moverse el gimbal, según la que sea su posición actual
}
```

posicion_actual

```
float posicion_actual (float declinacion) {

    int16_t xS,yS,zS;
    float suma=0;
    float angulo_actual;

    for (int k=0; k<20; k++){ //Cálculo de la media de 20 medidas tomadas en este punto

    qmc.read(&xS,&yS,&zS);

    //Ángulo en el que estoy (deberemos calcular el angulo del eje x respecto del Norte)
    float angulo=atan2(yS,xS); //Azimut magnético al que está el sensor
    suma=suma+angulo_actual;
    }

    angulo_actual=suma/20;
    angulo_actual=angulo_actual * RAD_TO_DEG;
    angulo_actual=angulo_actual-declinacion; //Azimut geográfico
    if(angulo_actual<0) angulo_actual=angulo_actual +360;

    return angulo_actual;
```

movimiento_servos

```
float altitud_corregida;
void Servos(float altitud, float acimut_corregido){

        c.angleYAW = SBGC_DEGREE_TO_ANGLE(acimut_corregido);

        altitud_corregida=altitud-180; //Transformación de la altitud a los valores del gimbal
        c.anglePITCH = SBGC_DEGREE_TO_ANGLE(altitud_corregida);
        SBGC_cmd_control_send(c, sbgc_parser);

        delay(5000); //Movimiento del gimbal
  }
```

medicion

```
void medicion (){

  char ej;
  float V1[2], V2[2], V3[2], V4[2];
  float V1b, V2b, V3b, V4b;
  float X1, X2, Y1, Y2, angX, angY;
  float Fx, Fy, valor1, valor2;
  float pi = 3.141593;

  //Actualizo a 0 las variables 'suma'
  float V1_suma=0, V2_suma=0, V3_suma=0, V4_suma=0, anguloX_suma=0, anguloY_suma=0;

  Serial.println("Estamos midiendo irradiación, introduce 'a':");
  delay(1000);
  ej = Serial.read();

  Serial.println("Valor introducido:");
  delay(500);
  Serial.write(ej);
  Serial.write("\n");
  delay(200);

  delay(1000);

  if ( Serial1.available() > 0 ) {

   for (int k=0; k<20; k++){ //Cálculo de la media de 20 medidas tomadas en este punto

      Serial1.write(ej); //Toma de medida en arduino pro mini

      //LECTURA DE MEDIDAS

      //V1//
      delay(50);
      V1[0] = Serial1.read();
      V1[1] = Serial1.read();
      delay(50);
      V1b = transforma2(V1[0], V1[1]); //Saco el valor entre 0-1024
      V1b = (V1b * 5) / 1024; //Lo transformo a 0-5V
      V1_suma = V1_suma + V1b;

      //V2//
      delay(50);

      V2[0] = Serial1.read();
      V2[1] = Serial1.read();
      delay(50);
      V2b = transforma2(V2[0], V2[1]);
      V2b = (V2b * 5) / 1024;
      V2_suma = V2_suma + V2b;
```

```
  //V3//
  delay(50);

  V3[0] = Serial1.read();
  V3[1] = Serial1.read();
  delay(50);
  V3b = transforma2(V3[0], V3[1]);
  V3b = (V3b * 5) / 1024;
  V3_suma = V3_suma + V3b;

  //V4//
  delay(50);

  V4[0] = Serial1.read();

  V4[1] = Serial1.read();
  delay(50);
  V4b = transforma2(V4[0], V4[1]);
  V4b = (V4b * 5) / 1024;
  V4_suma = V4_suma + V4b;

  //CÁLCULO ÁNGULO INCIDENCIA

  X1 = V3b + V4b; //Medidas entre 0-10V
  X2 = V1b + V2b;
  Y1 = V1b + V4b;
  Y2 = V2b + V3b;

  Fx = (X2 - X1) / (X2 + X1);
  Fy = (Y2 - Y1) / (Y2 + Y1);

  valor1 = Fx * 1.871; //Constante C por el modelo de SolarMems usado
  angX = atan(valor1);
  angX = (angX * 180) / pi; //Pasado a grados sexagesimales
  anguloX_suma = anguloX_suma + angX;

  valor2 = Fy * 1.871;
  angY = atan(valor2);
  angY = (angY * 180) / pi;
  anguloY_suma = anguloY_suma + angY;

      Serial1.write(ej); //Si después de todas las medidas no se dice nada, vuelvo a medir de nuevo
    }
  }

   V1_final= V1_suma/20;
   V2_final= V2_suma/20;
   V3_final= V3_suma/20;
   V4_final= V4_suma/20;
   anguloX_final = anguloX_suma/20;
   anguloY_final = anguloY_suma/20;

}
```

```
unsigned long transforma2 (byte a, byte b) { //Check la entrada entre 0-1023

  unsigned long valor = 0;

  if (b == 0) {
    valor = a;
  }

  if (b == 1) {
    valor = (a + 256);
  }
  if (b == 2) {
    valor = (a + 256 * 2);
  }
  if (b == 3) {
    valor = (a + 256 * 3);
  }
  return valor;
}
```

espiral

```
void espiral (float V1_final, float V2_final, float V3_final, float V4_final) {
  //Le llegan los valores de la medida que se acaba de realizar en el punto (0,0) de la espiral

  int servo1[10]={0,1,1,-1,-1,2,2,-2,-2,2};
  int servo2[10]={0,0,-1,-1,1,1,-2,-2,2,2};

  float maximo1=V1_final,maximo2=V2_final,maximo3=V3_final,maximo4=V4_final;
  float altitud_espiral;
  float acimut_espiral;

  int n;

  for (n=1; n<=10; n++){

    //PRIMERO HAGO EL MOVIMIENTO
    acimut_espiral = servo1[n];
    altitud_espiral = servo2[n];

    Servos(altitud_espiral, acimut_espiral);


    //SEGUNDO TOMO MEDIDA
    medicion ();
```

```
    //TERCERO COMPARO CON EL MÁXIMO, Y SI LO SUPERA ACTUALIZO ESA VARIABLE
    //Y LA ALMACENO EN LA MATRIZ FINAL
     matriz[j][0]=j;

     if(V1_final> maximo1){
       matriz[j][1]=V1_final;
       maximo1=V1_final;
    }
     if (V2_final> maximo2){
       matriz[j][2]=V2_final;
       maximo2=V2_final;
    }
     if (V3_final> maximo3){
       matriz[j][3]=V3_final;
       maximo3=V3_final;
    }
     if (V4_final> maximo4){
       matriz[j][4]=V4_final;
       maximo4=V4_final;
      }

    n++;
  }

}
```

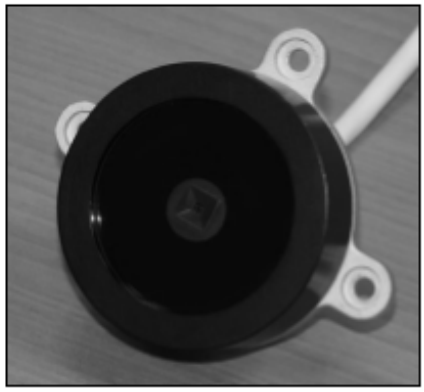
ANEXO B: Datasheet Sensor ISS-AX

Solar MEMS Technologies S.L.

Sun Sensor ISS-AX

Analog sensor

Technical Specifications



Features

Two orthogonal axes sun sensor
Wide or narrow field of view
High accuracy
4 analog outputs
Low power consumption
Wide operating voltage range: 5÷12 V
Industrial temperature range: - 40° to 85°
Reduced size
Low weight
IP65 protection
Reverse polarity protection

Applications

Sun tracking/pointing systems
Solar Trackers
Heliostats
Photovoltaic
CSP, CPV and HCPV
Stirling

ISS-AX sun sensor measures the incident angle of a sun ray in both orthogonal axes. The high sensitivity reached is based on the geometrical dimensions of the design.

Its characteristics make it a suitable tool for high accurate sun-tracking and positioning systems, with low power consumption and high reliability.

ISS-AX sun sensor has been designed with a unique and novel own technology based on MEMS fabrication processes to achieve high integrated sensing structures at low cost.

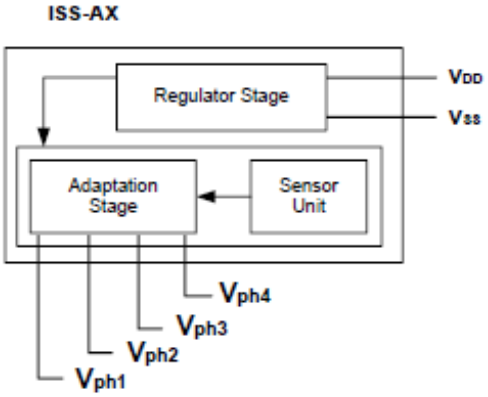


Fig 1. Block Diagram

Contents

Figures

Tables

1. General Specifications

Parameter	ISS-A60	ISS-A25	ISS-A15	ISS-A5	Unit	Comments
Sensor type	2 axes	2 axes	2 axes	2 axes	-	Orthogonal
Field of view (FOV)	120	50	30	10	º	Aperture of the cone of view
Accuracy (*)	< 10	< 10	< 10	< 10	%	3σ
Precision (*)	< 0,06	< 0,04	< 0,02	< 0,01	º	Sensitivity
Average consumption	9	9	9	9	mA	
Dimensions						
Diameter	80	80	80	80	mm	
Height	27	27	27	27	mm	
Weight	100	100	100	100	g	
Level of protection	IP65	IP65	IP65	IP65		CEI 60529 Standard
Expected lifetime of 10 years + Pressure test at 0,05 mbar and 25ºC						

(*) Accuracy test in Laboratory: cable of 2 meters, collimated light source, radiation of 900 W/m², CAD resolution of 10 bits, and filter stage with sampling frequency of 50 Hz and bandwidth of 0,4 Hz.

Table 1. General Specifications

Different models of the ISS-AX are offered, differing in the field of view (FOV) of the sensor. The accuracy of the sensor is inversely proportional to the field of view. All these models have been tested on solar trackers with Solar MEMS Helios Controller.

2. Electrical characteristics

Symbol	Parameter	Min	Typical	Max	Unit
V_{DD}	Supply voltage	5	5	12	V
I_{DD}	Feed current	-	9	-	mA
Vph	Photodiode voltages (analog outputs)	0	-	4,5	V
Recommended					
V_{DD}	Supply voltage	5	-	12	V
Vr	Supply voltage ripple	0	-	100	mVpp
T_{OP}	Operating temperature	-40	-	85	ºC
Absolute maximum					
V_{DD}	Supply voltage	0	-	16	V
T_{OP}	Operating temperature	-40	-	85	ºC

Table 2. Electrical characteristics

Reverse polarity protection.

3. Sun Sensor ISS-AX

ISS-AX sensor measures the incidence angles of a solar radiation respect to its perpendicular. This information is provided through 4 analog outputs.

3.1. Description

ISS-AX measures the incidence angle of a sun ray in both axes, based on a quadrant photodetector device. The sunlight is guided to the detector through a window above the sensor. Dependent of the angle of incidence, the sunlight induces photocurrents in the four quadrants of the detector.

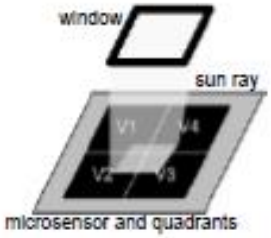


Fig 2. Microsensor of ISS-AX

3.2. Reference Axes

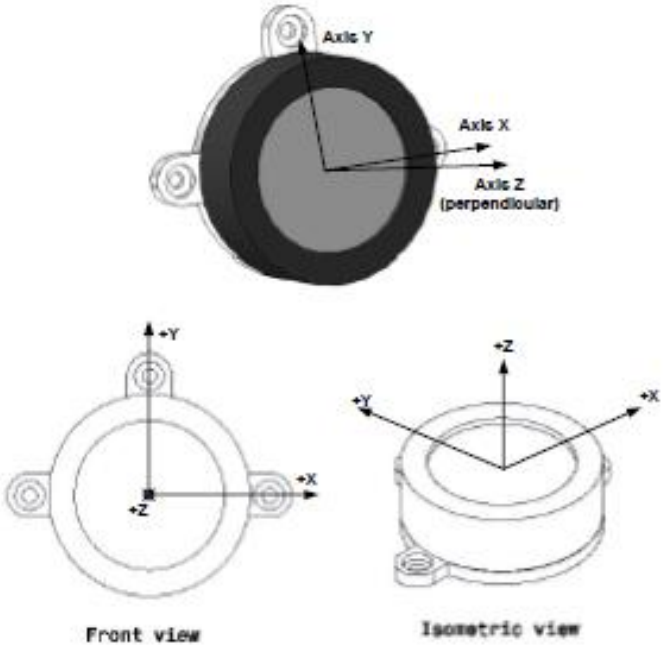


Fig 3. ISS-AX reference system

Z axis is perpendicular to the sensor base plane.

3.3. Measurements

The *Angle X* and *Angle Y* specify the angular position of the incident sun ray inside the field of view of the ISS-AX sensor.

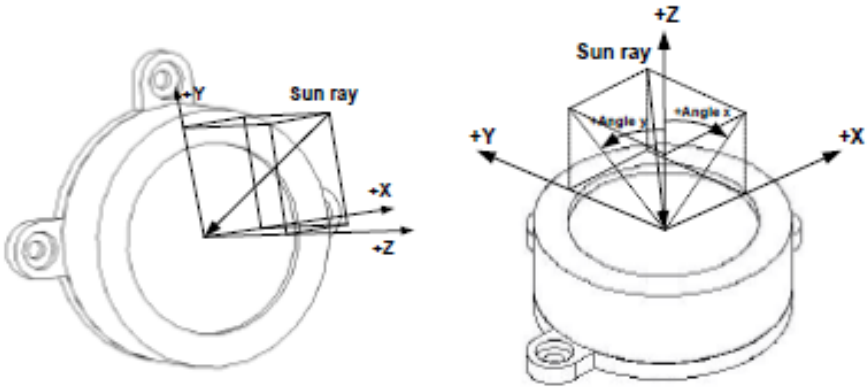


Fig 4. References for measured angles

Angle X and *Angle Y* of the incident ray can be obtained with a simple **set of equations** involving the four photodiode voltages generated by the sensor (V_{PH1}, V_{PH2}, V_{PH3}, and V_{PH4}):

$$X_1 = V_{PH3} + V_{PH4} \qquad Y_1 = V_{PH1} + V_{PH4}$$

$$X_2 = V_{PH1} + V_{PH2} \qquad Y_2 = V_{PH2} + V_{PH3}$$

$$F_X = \frac{X_2 - X_1}{X_2 + X_1} \qquad F_Y = \frac{Y_2 - Y_1}{Y_2 + Y_1}$$

$$Angle\ X = arctg\left(C \cdot F_X\right) \qquad Angle\ Y = arctg\left(C \cdot F_Y\right)$$

Type	Value
ISS-A60	1,871
ISS-A25	0,477
ISS-A15	0,324
ISS-A5	0,130

Table 3. Values of the parameter C according to the type of sensor ISS-AX (Geometric Correction)

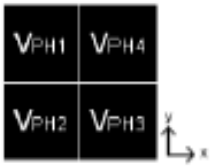


Fig 5. References for the photodiodes

The accuracy of the sensor increases when receiving a radiation perpendicular to the sensor, close to zero degrees in X and Y. This is an outstanding feature that makes it suitable for tracking applications. The **accuracy** can be **increased** in more than one order of magnitude by compensating the offset error after the installation of the sensor by means of **Calibration or Alignment**.

The use of a **filtering stage is recommended** (for example: 50 Hz sampling frequency and 0,4 Hz bandwidth).

4. Calibration or Alignment

Calibration process increases the accuracy of the ISS-AX sun sensor. The zero degrees position is the pair of angles X/Y that it is measured on the installation according to, for example, the maximum power generated of a CPV panel.

Example of calibration process: Solar Tracker:

1. Install the Sun Sensor ISS-AX on the tracker.
   The sensor must be installed according to the solar panels, in the same plane, as well as possible. This will improve the use of the sun sensor field of view.

2. Control the solar tracking to get the maximum power generated.
   Get the Angle X and Angle Y of the sun sensor ISS-AX. This pair will be your zero degrees position for the maximum power generation.

3. Control the solar tracker in closed-loop using your new zero degrees position of the ISS-AX sun sensor as reference: rectify the angles measured by the sensor according to this zero degrees position.

5. Recommendations

Depending on the application of the Sun Sensor ISS-AX, we recommend the use of the following models:

- Solar Tracker with Photovoltaic:
  The accuracy requirements are not demanding, so it's recommended to **use the ISS-A60** model, **to get a wide field of view**.
- Solar Tracker with CPV or similar:
  The accuracy requirements are very demanding, so it's recommended to **use the ISS-A5** model, **to get high accuracy and narrow field of view**, because a wide field of view increases the effects of the **environmental conditions** on the accuracy of the sun sensor: clouds effect and seeing of the ground.
- Other applications:
  It depends on the demanding of the field of view and the accuracy.

6. Electrical interface

Colour	Terminal	Type	Comments
Red	VDD	Power	Power Supply
Blue	VSS	Power	Ground
Yellow	Vph1	Analog output	Photodiode 1: upper-left
Green	Vph2	Analog output	Photodiode 2: lower-left
Brown	Vph3	Analog output	Photodiode 3: lower-right
White	Vph4	Analog output	Photodiode 4: upper-right
Grey	Rtn	Analog reference	Signal return
Pink	-	-	Not connected
Shield	-	-	Connect to the blue wire

Table 4. Electrical interface

The housing of the sun sensor ISS-AX is isolated electrically.

7. Mechanical data

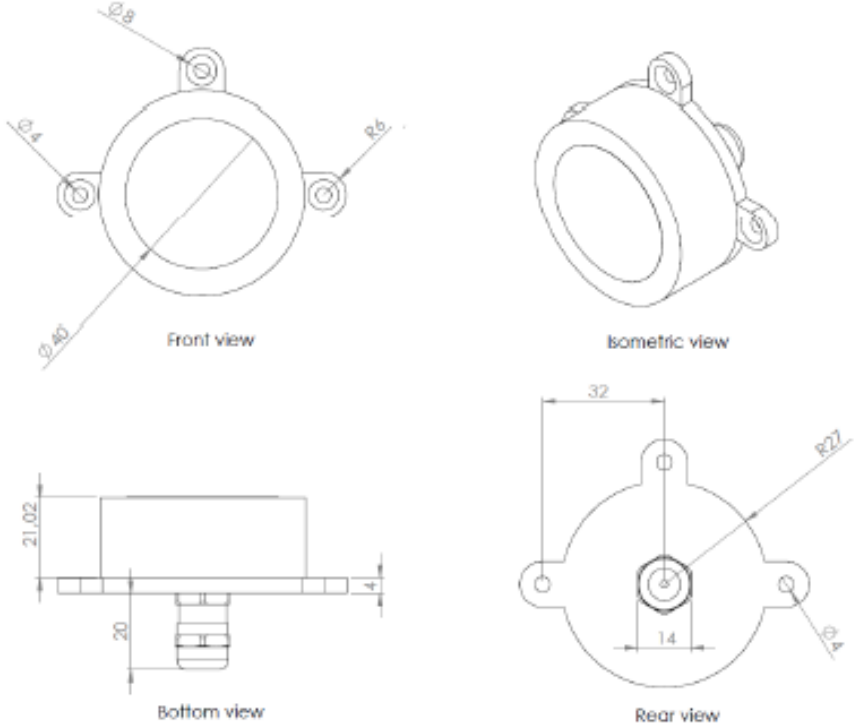


Fig 6. ISS-AX dimensions

The box of the ISS-AX sensor is composed of a top and bottom housing, both made of Aluminum 6082: it has good corrosion resistance. The top housing has a protective coating of anodizing and it is black lacquered, and the bottom housing has a protective coating of matt anodizing.

8. Warranty

Solar MEMS Technologies S.L. warrants the ISS-AX sun sensor to the original consumer purchaser any product that is determined to be defective for the following terms will be repaired or replaced.

The warranty is one year from date of purchase.

The product in question must be sent to Solar MEMS Technologies S.L. (address is shown below) within the warranty period and the original consumer purchaser must comply with the following conditions, to be eligible for repair or replacement under this warranty:

- The product must not have been modified or altered in any way by an unauthorized source.
- The product must have been installed in accordance with the installation instructions and the technical specifications.

This limited warranty does not cover:

- Damage due to improper installation.
- Accidental or intentional damages.
- Misuse, abuse, corrosion, or neglect.
- Product impaired by severe conditions, such as excessive wind, ice, storms, lightning strikes or other natural occurrences.
- Damage due to improper packaging on return shipment.

Any and all labor charges for troubleshooting, removal or replacement of the product are not covered by this warranty and will not be honored by Solar MEMS Technologies S.L.

Return shipping to Solar MEMS Technologies S.L. must be pre-paid by the original consumer purchaser. Solar MEMS Technologies S.L. will pay the normal return shipping charges to original consumer purchaser within the European Union countries only.

Address of Solar MEMS Technologies S.L.
Solar MEMS Technologies S.L.
C/Early Ovington 24, nave 1.
41300, La Rinconada,
Seville, Spain.
E-mail: smt@solar-mems.com
Phone: (+34) 954 460 113

Solar MEMS has a system of quality and environment according to the ISO 9001 and ISO 14001 standards, provided by the certification company Applus CTC.